Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informačních technologií

Bezdrátová senzorová síť s využitím protokolů ESP-NOW a ESP-MESH

Bakalářská práce

Autor: Daniel Mikš
Studijní obor: Aplikovaná informatika (ai3s-p)

Vedoucí práce: Ing. Patrik Urbaník

Hradec Králové Duben 2025

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 28.4.2025 Daniel Mikš

Poděkování:

Rád bych poděkoval svému vedoucímu bakalářské práce Ing. Patriku Urbaníkovi za cenné rady, připomínky při zpracování práce a věnovaný čas během konzultací.

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací bezdrátové senzorové sítě s využitím komunikačních protokolů ESP-NOW a ESP-MESH. V úvodních kapitolách je čtenář seznámen s konceptem internetu věcí, jeho výhodami a výzvami. Následně jsou podrobně popsány moduly ESP8266 a ESP32, které tvoří základ navržené sítě. Práce se zaměřuje na charakteristiku a praktické použití protokolů ESP-NOW a ESP-MESH, včetně jejich výhod, omezení a možností kombinovaného nasazení. V experimentální části je popsána realizace sítě, měření dat pomocí různých senzorů (BME280, BH1750, DS18B20), a odesílání těchto dat do Raspberry Pi. Součástí je rovněž vyhodnocení spolehlivosti přenosu a efektivity zvolených topologií. Výsledky ukazují, že použití multihop komunikace umožňuje efektivní sběr dat i v rozsáhlejším prostoru bez přímé viditelnosti mezi zařízeními.

Abstract

Title: Wireless Sensor Network Using ESP-NOW and ESP-MESH Protocols

This Bachelor Thesis focuses on the design and implementation of a wireless sensor network using ESP-NOW and ESP-MESH communication protocols. The introductory chapters provide an overview of the Internet of Things, including its benefits and challenges. The thesis then discusses the ESP8266 and ESP32 modules that form the hardware basis of the proposed network. Emphasis is placed on the characteristics and practical applications of ESP-NOW and ESP-MESH, highlighting their strengths, limitations, and potential for combined deployment. The practical section describes the implementation of the network, data collection using various sensors (BME280, BH1750, DS18B20), and transmission to a Raspberry Pi. The work includes an evaluation of transmission reliability and the efficiency of different topologies. The results demonstrate that multihop communication allows effective data collection even in environments without direct visibility between nodes.

Key words: ESP-NOW, ESP-MESH, wireless sensor network, Internet of Things, ESP32, Raspberry Pi

Obsah

1 Úvod 1

2 Cíl práce 2

3 Metodika zpracování 3

4 Úvod do internetu věcí 4

4.1 Historický vývoj a současný význam 4

4.2 Oblasti využití IoT 4

4.3 Přínosy a výzvy internetu věcí 5

5 Bezdrátové technologie pro komunikaci 6

5.1 Bezdrátové senzorové sítě 6

5.1.1 Komunikační protokoly v bezdrátových senzorových sítích 6

5.2 Internet věcí a jeho propojení s WSN 7

5.3 Výzvy a budoucnost bezdrátových senzorových sítí 7

6 Platformy Espressif 8

6.1 ESP8266 9

6.1.1 Varianty ESP8266 9

6.2 ESP32 10

7 Technologie pro vzájemnou komunikaci mezi ESP 11

7.1 Protokol ESP-NOW 12

7.1.1 Klíčové vlastnosti ESP-NOW 12

7.1.2 Komunikace pomocí ESP-NOW 12

7.1.3 Topologie – One Master, Multiple Slaves 12

7.1.4 Topologie – One Slave, Multiple Masters 13

7.1.5 Topologie – Multiple Masters, Multiple Slaves 14

7.1.6 Vrstvy ESP-NOW 15

7.2 Protokol ESP-MESH ... 16
7.2.1 Klíčové vlastnosti ESP-MESH ... 16
7.2.2 Komunikace pomocí ESP-MESH ... 16
7.3 Porovnání ESP-NOW a ESP-MESH ... 17
7.4 Multihop komunikace v ESP sítích ... 18
7.4.1 Úvod do multihopu a jeho význam v IoT ... 18
7.4.2 Topologie sítí ... 19
7.4.3 Způsoby multihop komunikace ... 20
7.4.4 Výhody a nevýhody jednotlivých metod ... 20
7.4.5 Využití v ESP-MESH a ESP-NOW ... 21
8 Typy komunikačních sběrnic ... 21
8.1 I2C ... 21
8.2 SPI ... 22
8.3 One-Wire ... 23
9 Implementace sítě ... 24
9.1 Senzory pro sběr dat ... 25
9.2 Analogové senzory ... 25
9.3 Digitální senzory ... 25
9.4 Porovnání analogových a digitálních senzorů ... 26
9.4.1 Senzor teploty (DS18B20) ... 26
9.4.2 Senzor tlaku, teploty a vlhkosti (BME280) ... 27
9.4.3 Senzor intenzity světla (BH1750) ... 28
9.5 Zprovoznění nodů ... 29
9.6 Implementace sítě ESP-NOW ... 30
9.6.1 Implementace sítě ESP-MESH ... 32
9.7 Měření jednotlivých veličin ... 33

9.7.1 Měření senzorem BME280 ... 33
9.7.2 Měření senzorem BH1750 ... 34
9.7.3 Měření senzorem DS18B20 ... 35
9.8 Příprava Raspberry Pi ... 35
9.8.1 Instalace Raspbianu ... 36
9.8.2 Instalace potřebného softwaru ... 37
9.9 Odesílání dat do Raspberry Pi ... 38
9.10 Ukládání naměřených hodnot ... 39
9.11 Výstup měřených hodnot ... 40
10 Shrnutí a diskuse výsledků ... 42
11 Závěry a doporučení ... 43
12 Seznam použité literatury ... 44
13 Seznam použítých zkratek ... 49
14 Přílohy ... 50
14.1 Příloha č. 1: Zdrojový kód ESP-NOW Slave ... 50
14.2 Příloha č. 2: Zdrojový kód ESP-NOW Master ... 53
14.3 Příloha č. 3: Zdrojový kód ESP-MESH Node ... 56
14.4 Příloha č. 4: Zdrojový kód ESP-MESH Root node ... 58
15 Zadání práce z IS (eVŠKP) ... 60

Seznam Obrázků

Obrázek 1 One master, Multiple slaves 13

Obrázek 2 One slave, Multiple Masters 14

Obrázek 3 Vrstvy ESP-NOW 15

Obrázek 4 ESP-MESH Síť 17

Obrázek 5 Topologie sítí 19

Obrázek 6 Blokové schéma implementace 24

Obrázek 7 Výsledná síť ESP 25

Obrázek 8 Senzor 18B20 27

Obrázek 9 Senzor BME280 28

Obrázek 10 Senzor BH1750 ve Variantě I2C 28

Obrázek 11 Schéma zapojení senzorů 29

Obrázek 12 Instalace Raspberry Pi 36

Obrázek 13 Graf naměřených hodnot 41

Seznam zdrojových kódů

Zdrojový kód 1 Formát odesílané zprávy 30

Zdrojový kód 2 Odeslání zprávy ESP-NOW 31

Zdrojový kód 3 Měření senzorem BME280 34

Zdrojový kód 4 Měření senzorem DS18B20 35

Zdrojový kód 5 Globální proměnné pro odesílání dat 38

Zdrojový kód 6 Připojení k access pointu 38

Zdrojový kód 7 Vzor přijatých dat na API serveru 39

1 Úvod

Moderní společnost se stále více opírá o technologie, které umožňují automatizaci, sběr údajů a jejich sdílení mezi různými zařízeními. Tato zařízení se postupně stávají nedílnou součástí domácností, podniků i veřejných institucí, a jejich počet i rozmanitost každoročně roste. Vzniká tak prostředí, ve kterém dochází k neustálé výměně informací, a to často bez zásahu člověka. Tato propojenost přináší řadu výhod, jako je vyšší efektivita, pohodlí či úspora nákladů, ale zároveň klade i nové požadavky na způsob, jakým tato zařízení komunikují a spolupracují.

Zvláštní pozornost je věnována situacím, kdy není možné využít klasické infrastruktury pro přenos informací, a je tedy třeba hledat jiná, samostatná řešení. Právě zde nacházejí uplatnění systémy, které jsou schopny fungovat nezávisle a zároveň poskytovat důležité údaje v reálném čase. Pro jejich efektivní provoz je zásadní schopnost jednotlivých jednotek vzájemně komunikovat a vytvářet logický celek, který dokáže fungovat i v prostředí s omezeným přístupem ke stávající síti.

Vzhledem k těmto výzvám a potřebám se práce zaměřuje na oblast bezdrátového přenosu údajů mezi zařízeními a na možnosti, jak vytvořit síť, která je nejen funkční, ale zároveň flexibilní a snadno rozšiřitelná. Cílem práce je přiblížit fungování těchto systémů, popsat jejich základní principy a zároveň navrhnout praktické řešení, které demonstruje využití těchto technologií v jednoduchém a srozumitelném modelu.

2 Cíl práce

Cílem této bakalářské práce je navrhnout, realizovat a ověřit fungování bezdrátové sítě určené pro přenos údajů mezi jednotlivými prvky. V práci je kladen důraz na popis principů bezdrátové komunikace, možností zapojení více jednotek do jednoho celku a způsobů, jakými lze data efektivně předávat v prostředí bez pevného připojení.

Teoretická část se zaměřuje na přehled technologií používaných pro bezdrátové propojení zařízení, jejich výhody, omezení a vhodnost nasazení v různých podmínkách. Praktická část obsahuje návrh sítě, její sestavení a ověření funkčnosti pomocí sběru a přenosu údajů. Výsledkem práce je funkční model, který slouží jako ukázka možností bezdrátové komunikace v moderních systémech.

3 Metodika zpracování

Tato bakalářská práce je založena na kombinaci rešeršní a praktické části. V první fázi byla provedena důkladná analýza dostupných odborných zdrojů zaměřených na oblast bezdrátové komunikace, síťových struktur a jejich využití v praxi. Zdroje zahrnovaly jak akademické publikace, tak technickou dokumentaci k používaným technologiím. Cílem této fáze bylo pochopit základní principy bezdrátového přenosu dat, klasifikaci sítí a topologií, a současně identifikovat klíčové faktory ovlivňující návrh a provoz bezdrátových senzorových sítí.

Na základě provedené analýzy došlo k syntéze získaných poznatků, které posloužily jako východisko pro návrh vlastního modelu sítě. Tento model byl navržen s ohledem na specifika a požadavky související s bezdrátovou komunikací a se zaměřením na efektivní přenos dat, robustnost a možnosti škálování.

V praktické části byl realizován návrh sítě, jehož funkčnost byla ověřena testováním. Průběh realizace i výsledky byly zaznamenány a následně vyhodnoceny.

Při psaní bakalářské práce byly využity nástroje umělé inteligence, zejména ChatGPT, a to především jako podpora při formulaci textu, kontrole gramatiky, ověřování odborných pojmů a při strukturování a rozpracování obsahu práce.

4 Úvod do internetu věcí

Internet věcí představuje dynamicky se rozvíjející oblast informačních technologií, která propojuje fyzické objekty prostřednictvím internetu. Tato zařízení jsou vybavena senzory, čidly, softwarem a dalšími technologiemi, které jim umožňují nejen vzájemnou komunikaci, ale i komunikaci s uživateli a cloudovými systémy [1]. Hlavním cílem IoT (Internet of Things) je sběr, zpracování a přenos dat v reálném čase, což otevírá nové možnosti v oblasti automatizace, optimalizace procesů a inteligentního rozhodování [2].

4.1 Historický vývoj a současný význam

Myšlenka propojení zařízení prostřednictvím sítě sahá až do 80. let 20. století, ale skutečný rozvoj IoT začal až s nástupem bezdrátových technologií, miniaturizace čipů a rozšířením internetu vysoké rychlosti. Termín „Internet of Things“ byl poprvé použit v roce 1999 Kevinem Ashtonem v souvislosti s RFID (Radio Frequency Identification) technologiemi [3]. Od té doby se koncept IoT rychle rozvinul a stal se nedílnou součástí moderních technologií a průmyslových strategií, například v rámci koncepce Průmysl 4.0.

V současné době se IoT stává jedním z klíčových pilířů digitalizace. Organizace i jednotlivci využívají propojená zařízení ke zvýšení efektivity, snížení provozních nákladů a získání cenných dat pro další analýzu. Díky rozvoji cloudových služeb, umělé inteligence (AI) a edge computingu se navíc možnosti internetu věcí dále rozšiřují [4] a stávají se dostupnějšími i pro menší podniky a domácnosti. Zatímco cloud computing spoléhá na vzdálené servery, kde jsou data zpracovávána a ukládána, edge computing přesouvá část výpočetního výkonu blíže ke koncovým zařízením, jako jsou například chytré senzory nebo domácí asistenti. Díky tomu je možné reagovat rychleji, snížit zatížení sítě a zvýšit spolehlivost systémů, což je zásadní zejména u zařízení internetu věcí v reálném čase [5].

4.2 Oblasti využití IoT

IoT má široké spektrum využití napříč různými sektory:

- **Chytré domácnosti:** Termostaty, osvětlení, bezpečnostní kamery a další zařízení umožňují uživatelům ovládat své prostředí na dálku a přizpůsobovat ho svým potřebám.

- **Průmysl a výroba (Průmysl 4.0):** IoT senzory sledují výrobní linky, detekují poruchy a optimalizují výrobní procesy.
- **Zemědělství:** Chytrá čidla monitorují půdní vlhkost, teplotu a další podmínky, čímž pomáhají zvyšovat výnosy a efektivněji hospodařit s vodními zdroji.
- **Zdravotnictví:** Nositelná zařízení měří vitální funkce pacientů a odesílají data lékařům v reálném čase.
- **Městská infrastruktura:** IoT přispívá k rozvoji tzv. *smart cities*, například skrze inteligentní řízení dopravy, odpadového hospodářství či veřejného osvětlení.

4.3 Přínosy a výzvy internetu věcí

Internet věcí představuje revoluční přístup k propojení fyzického a digitálního světa. Díky schopnosti sbírat a analyzovat data v reálném čase umožňuje zvýšení efektivity, úsporu nákladů a zlepšení uživatelského komfortu [3]. Příkladem může být chytrá domácnost, kde systém automaticky reguluje teplotu podle přítomnosti osob, nebo průmyslové zařízení, které identifikuje závadu ještě před jejím vznikem.

Se zaváděním IoT systémů však přicházejí také nové výzvy, zejména v oblasti bezpečnosti, ochrany soukromí, standardizace a interoperability [5]. Například únik osobních údajů z chytrých zařízení nebo prolomení zabezpečení sítě může vést ke ztrátě citlivých dat či k ohrožení bezpečnosti uživatelů. Autonomní operace zařízení bez nutnosti přímého lidského zásahu je umožněna integrací senzorů, akčních členů, výpočetní logiky a síťového připojení [5]. Tento model umožňuje dynamickou reakci na měnící se podmínky, optimalizaci chování systémů a zvýšení jejich efektivity i spolehlivosti.

Správná volba komunikační technologie je klíčová pro zajištění funkčnosti celého IoT ekosystému. Při návrhu systému je třeba zohlednit požadavky na dosah, energetickou náročnost, přenosovou rychlost a schopnost zvládat vysokou hustotu připojených zařízení [1].

V posledních letech lze navíc sledovat trend směřující k edge computingu, tedy k přesunu výpočetních operací blíže ke zdroji dat. Tento přístup snižuje latenci, odlehčuje centrální infrastrukturu a umožňuje rychlejší a bezpečnější reakce zařízení v reálném čase. Současně se rozvíjí i technologie zaměřené na úsporné přenosy dat, například LPWAN (Low-Power Wide-Area Network), které jsou vhodné pro zařízení s omezenými energetickými zdroji [5].

5 Bezdrátové technologie pro komunikaci

Bezdrátová komunikace se stala klíčovým prvkem moderního propojeného světa. Umožňuje přenos dat mezi zařízeními bez použití fyzických vodičů, což přináší výhody jako mobilitu, flexibilitu a jednoduchost instalace. Tyto technologie se využívají v různých oblastech, včetně průmyslu, domácností, dopravy a medicíny. Mezi hlavní představitele bezdrátové komunikace patří Wi-Fi, mobilní sítě, technologie krátkého dosahu jako Bluetooth a bezdrátové senzorové sítě (Wireless Sensor Networks - WSN), které se uplatňují především v monitorovacích a automatizačních systémech [1].

5.1 Bezdrátové senzorové sítě

Bezdrátové senzorové sítě tvoří skupina autonomních senzorových uzlů, které monitorují a přenášejí data o svém okolí. Senzory mohou měřit nejrůznější fyzikální veličiny, například teplotu, vlhkost, tlak, světlo, vibrace nebo chemické složení prostředí. Tyto sítě nacházejí uplatnění v průmyslové automatizaci, kde sledují stav strojů a přispívají k prediktivní údržbě, v zemědělství, kde optimalizují zavlažování na základě měření vlhkosti půdy, a v chytrých městech, kde monitorují dopravu, parkovací místa či kvalitu ovzduší [1]. Dále se využívají ve zdravotnictví, kde umožňují sledování vitálních funkcí pacientů pomocí nositelných senzorů, a v oblasti obrany a bezpečnosti, kde se podílejí na detekci neoprávněného vstupu nebo sledování pohybu v citlivých oblastech. WSN se vyznačují nízkou spotřebou energie, distribuovaným zpracováním dat a schopností samoorganizace. Klíčovým problémem těchto sítí je efektivní správa energie, protože senzory často pracují na bateriové napájení a je nutné minimalizovat jejich spotřebu pro maximální prodloužení životnosti [7]. Neméně důležité je však i zajištění spolehlivé komunikace – tedy schopnost přenášet data ze senzorů i v prostředí s minimálním pokrytím, což vyžaduje promyšlené řešení síťové topologie a výběr vhodného komunikačního protokolu.

5.1.1 Komunikační protokoly v bezdrátových senzorových sítích

Pro efektivní fungování WSN se využívají specializované komunikační protokoly, které umožňují přenos dat mezi senzory a centrálním uzlem. Jedním z nejpoužívanějších protokolů je Zigbee, který je optimalizován pro nízkou spotřebu energie a krátké vzdálenosti, což ho činí ideálním pro aplikace v oblasti internetu věcí. Další možností je LoRaWAN (Long Range Wide Area Network), který je vhodný pro přenos dat na dlouhé vzdálenosti při zachování nízké spotřeby energie, a proto se často využívá ve

vzdálených oblastech bez dostupnosti jiných komunikačních sítí. Pro krátké vzdálenosti se využívá BLE (Bluetooth Low Energy), který je oblíbený zejména v nositelných zařízeních. Zajímavou variantou je také 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks), který umožňuje použití protokolu IPv6 (Internet Protocol version 6) v nízkoenergetických sítích a zajišťuje kompatibilitu s internetovou infrastrukturou. Každý z těchto protokolů je optimalizován pro různé scénáře použití a jejich výběr závisí na konkrétních požadavcích na dosah, spotřebu energie a datovou propustnost [7]. Způsoby, jakými tyto protokoly přistupují k předávání dat v síti, například prostřednictvím záplav (Flooding), šeptání (Gossiping) nebo hierarchických struktur s clustery, jsou detailněji rozebrány v části 6.6.3.

5.2 Internet věcí a jeho propojení s WSN

Bezdrátové senzorové sítě jsou nedílnou součástí konceptu internetu věcí, který umožňuje propojení různých zařízení do jedné komunikační sítě. V této síti mohou zařízení vzájemně komunikovat a reagovat na změny prostředí v reálném čase. Ve spojení s cloud computingem a AI mohou WSN zajišťovat autonomní řízení procesů bez nutnosti lidského zásahu [8]. To se uplatňuje například v chytrých budovách, kde senzory monitorují spotřebu energie, teplotu a vlhkost a na základě těchto údajů automaticky upravují vytápění a ventilaci. Podobně v oblasti průmyslu 4.0 umožňují WSN automatizaci výrobních procesů a optimalizaci logistiky tím, že neustále sledují stav materiálů, strojů a výrobků [9][10].

5.3 Výzvy a budoucnost bezdrátových senzorových sítí

Bezdrátové senzorové sítě nabízejí široké možnosti využití napříč různými oblastmi, avšak zároveň čelí několika klíčovým výzvám. Mezi nejzásadnější patří optimalizace spotřeby energie, zabezpečení dat a škálovatelnost sítě. Energetická efektivita je zásadní pro dlouhou životnost senzorových uzlů, a proto se výzkum intenzivně zaměřuje na minimalizaci spotřeby energie a na vývoj energeticky úsporných komunikačních protokolů [7]. Bezpečnost je dalším kritickým faktorem, jelikož senzorové sítě často přenášejí citlivá data, která musí být chráněna pomocí šifrování a autentizace. Škálovatelnost představuje výzvu v souvislosti s rostoucím počtem zařízení, která je třeba efektivně integrovat a spravovat v rámci stávajících systémů.

Se zvyšující se komplexitou a rozsahem těchto sítí roste také náročnost na spolehlivou a efektivní komunikaci mezi jednotlivými prvky. V prostředí internetu věcí se stále

častěji uplatňují mikrokontroléry, jako jsou jednotky ESP (Espressif Systems Platform). Tyto moduly zajišťují nejen sběr dat ze senzorů, ale také jejich následné zpracování a bezdrátový přenos. Díky své nízké spotřebě energie, dostupnosti a silné komunitní podpoře se ESP moduly stávají klíčovou součástí mnoha IoT řešení, a to jak v akademické sféře, tak v praktických aplikacích [9][11].

Technologie umožňující vzájemnou komunikaci mezi těmito mikrokontroléry hrají zásadní roli při vytváření decentralizovaných a inteligentních sítí, které dokážou efektivně fungovat i v prostředích s omezenými zdroji. Budoucnost bezdrátových senzorových sítí směřuje k ještě těsnější integraci s prvky umělé inteligence, která umožní lokální zpracování dat přímo na úrovni senzorů, čímž dojde ke snížení objemu přenášených informací a ke zvýšení rychlosti odezvy systému.

Dalším významným trendem je snaha o energetickou soběstačnost senzorových uzlů. Využití technologií pro sklizeň energie z okolního prostředí, jako je solární napájení nebo sběr energie z vibrací a teplotních rozdílů, umožní provoz zařízení po dlouhou dobu bez nutnosti výměny baterií.

Bezdrátové technologie pro komunikaci, a zejména bezdrátové senzorové sítě, tak budou hrát stále významnější roli v rozvoji internetu věcí, chytrých měst a průmyslové automatizace. Jejich další vývoj přinese nové možnosti pro efektivnější a inteligentnější správu systémů v různých oblastech lidské činnosti.

6 Platformy Espressif

Mikročipy ESP od společnosti Espressif Systems patří mezi nejpopulárnější a nejdostupnější řešení v oblasti vestavných systémů s integrovanou Wi-Fi a Bluetooth konektivitou. Díky své flexibilitě, nízké ceně a široké podpoře ze strany open-source komunit se staly klíčovou volbou pro vývojáře i hobby nadšence v oblasti internetu věcí. ESP mikrokontroléry nabízejí nejen dostatečný výpočetní výkon, ale také podporu rozšířených protokolů, nízkou spotřebu energie a snadnou integraci s různými vývojovými platformami, včetně Arduino IDE (Integrated Development Environment), MicroPython a ESP-IDF (Espressif IoT Development Framework) [5].

Významnou výhodou těchto čipů je jejich modulární struktura a široké spektrum variant, které umožňují uživatelům vybrat si vhodný model podle požadavků daného projektu, od jednoduchých senzorových uzlů po pokročilé aplikace využívající AI nebo edge computing.

6.1 ESP8266

ESP8266 je historicky jedním z prvních dostupných mikrokontrolérů, který přinesl Wi-Fi konektivitu do světa DIY (Do It Yourself) elektroniky a otevřel dveře široké škále inovativních projektů. Díky své cenové dostupnosti a snadnému programování se rychle stal oblíbenou volbou pro aplikace v domácí automatizaci, inteligentních senzorech nebo jednoduchých webových serverech běžících na vestavných zařízeních [12].

Tento čip je postaven na jednojádrovém procesoru Tensilica L106 s frekvencí až 80 MHz a podporuje standard Wi-Fi 802.11 b/g/n. Má integrovaný TCP/IP protokolový stack, což znamená, že může fungovat jako samostatný Wi-Fi klient nebo dokonce jako přístupový bod (AP). Programovatelný je nejen v jazycích Lua a C/C++, ale také v prostředích jako Arduino IDE nebo PlatformIO, což rozšiřuje jeho využití pro různé typy vývojářů [13].

ESP8266 má relativně omezený počet GPIO (General Purpose Input/Output) pinů, což může být limitující při složitějších aplikacích, ale stále je dostatečný pro základní senzorické aplikace a menší projekty. K dispozici má i jednoduchý analogový vstup, který umožňuje měření senzorových hodnot bez potřeby externích převodníků. I přes své omezení zůstává ESP8266 jedním z nejpoužívanějších čipů v komunitě IoT díky svému poměru cena/výkon a snadné integraci s bezdrátovými sítěmi [14].

6.1.1 Varianty ESP8266

ESP8266 je dostupný v různých modulech a vývojových deskách, které se liší počtem GPIO pinů, rozměry, podporou periferií a dalšími funkcemi. Některé z nejpopulárnějších variant zahrnují:

ESP-01 – Základní modul s minimálním počtem pinů (2 GPIO), vhodný pro jednoduché Wi-Fi aplikace. Má kompaktní rozměry, ale omezené možnosti připojení dalších periferií.

ESP-07 – Modul s externím anténním konektorem a větším počtem GPIO pinů než ESP-01, vhodný pro aplikace vyžadující delší dosah Wi-Fi signálu.

ESP-12E / ESP-12F – Pokročilé varianty s více GPIO piny, lepší integrovanou anténou a větší pamětí. Tyto moduly se často používají na vývojových deskách a v hotových IoT zařízeních.

Wemos D1 Mini – Velmi populární vývojová deska kompatibilní s Arduino ekosystémem. Má micro USB (Universal serial bus) konektor, vestavěný převodník USB-UART a kompaktní rozměry, což usnadňuje vývoj IoT projektů.

NodeMCU – Jedna z nejznámějších vývojových desek s ESP8266. Obsahuje micro USB port, více GPIO pinů a podporuje programování v Lua i Arduino IDE. Má integrovaný převodník USB-UART, což zjednodušuje programování.

LOLIN (dříve Wemos) D1 R2 – Vývojová deska ve formátu podobném Arduino Uno, která umožňuje snadné propojení s Arduino shieldy a nabízí více GPIO pinů než Wemos D1 Mini.

ESP-14 – Modul kombinující ESP8266 s mikrokontrolérem STM8S003, což umožňuje hybridní aplikace s výhodami obou čipů.

ESP8266 CH340 – Varianta ESP8266 s integrovaným převodníkem USB-UART (Universal Asynchronous Receiver-Transmitter) CH340**,** který umožňuje snadné připojení k počítači bez potřeby externích adaptérů. Tato verze je oblíbená mezi vývojáři, protože eliminuje problémy s kompatibilitou, které se někdy vyskytují u jiných převodníků, jako je CP2102.

6.2 ESP32

ESP32 představuje významný krok vpřed oproti ESP8266 a nabízí rozšířenou funkcionalitu, která umožňuje jeho využití i v pokročilejších aplikacích. Tento čip přináší vyšší výpočetní výkon, podporu Bluetooth a větší množství GPIO pinů, což ho činí ideálním pro náročnější projekty v oblasti IoT, průmyslové automatizace a chytrých domácností.

ESP32 disponuje dvoujádrovým procesorem Xtensa LX6 (některé varianty mají jednojádrovou verzi) s taktovací frekvencí až 240 MHz. Oproti ESP8266 má také výrazně vylepšené možnosti správy spotřeby energie, což umožňuje jeho nasazení v aplikacích napájených bateriemi. Kromě standardního Wi-Fi 802.11 b/g/n nabízí také podporu Bluetooth 4.2 (Classic a BLE), což umožňuje širší spektrum použití v oblasti bezdrátové komunikace [15].

Postupem času vznikly různé varianty ESP32, z nichž každá je optimalizována pro specifické aplikace:

ESP32-S2 je zaměřený na aplikace s vysokými nároky na USB konektivitu a podporu pro zpracování obrazu a zvuku. Oproti standardnímu ESP32 mu však chybí Bluetooth.

ESP32-C3 je založen na otevřené architektuře RISC-V (Reduced Instruction Set Computer) a nabízí podporu Wi-Fi a Bluetooth 5.0, čímž poskytuje moderní alternativu ke klasickým ESP32 modelům.

ESP32-S3 přináší rozšířené možnosti pro AI a strojové učení (ML – Machine Learning) a je optimalizovaný pro aplikace zpracování obrazu a zvuku. Zároveň podporuje Bluetooth 5 a Wi-Fi, což umožňuje jeho využití v pokročilých IoT řešeních.

Každý z těchto modelů je navržen tak, aby splňoval specifické požadavky různých typů projektů, od jednoduchých senzorových systémů po komplexní aplikace s vysokými výpočetními nároky. ESP32 tak představuje flexibilní platformu s širokou podporou v komunitě a rozsáhlou dokumentací, což usnadňuje vývoj i méně zkušeným programátorům. Porovnání klíčových vlastností obou modelů je uvedeno v tabulce níže.

Vlastnost	ESP8266	ESP32
Procesor	Tensilica L106, až 160 MHz	Tensilica Xtensa LX6, až 240 MHz
Počet jader	1x	2x
Bluetooth	Není podporováno	Bluetooth v4.2 (Classic + BLE)
Paměť RAM	Až 50 kB RAM	až 520 kB SRAM
Externí FLASH	Podpora SPI (viz 8.2) flash	Podpora SPI flash + PSRAM
ADC	1 kanál (10 bitů)	Až 18 kanálů (12 bitů)
Šifrování Wi-Fi	Pouze softwarově	Hardwarové AES, SHA-2, RSA, ECC
Spotřeba energie	Nízká (vyšší než optimalizované ESP32 varianty)	Standardně vyšší, ale režimy hlubokého efektivnější než ESP8266
Cena	Nízká	O něco vyšší, ale stále nízká

Tabulka 1 Srovnání ESP8266 a ESP32

Bez ohledu na to, zda vývojář potřebuje jednoduchý mikrokontrolér pro základní Wi-Fi aplikace nebo výkonnější řešení pro pokročilé IoT projekty, ESP mikročipy od Espressif Systems nabízejí širokou škálu možností, které umožňují realizovat téměř jakýkoli bezdrátový projekt [15].

7 Technologie pro vzájemnou komunikaci mezi ESP

Schopnost zařízení vzájemně komunikovat je klíčovým předpokladem pro vytváření efektivních a spolehlivých bezdrátových sítí v prostředí internetu věcí. Tato kapitola se zaměřuje na technologie určené pro přímou komunikaci mezi moduly ESP, konkrétně

na protokoly ESP-NOW a ESP-MESH. Tyto protokoly umožňují vytváření sítí s nízkou latencí, nízkou spotřebou energie a vysokou škálovatelností. V jednotlivých podkapitolách budou popsány jejich klíčové vlastnosti, způsob komunikace, používané topologie i možnosti využití v praxi.

7.1 Protokol ESP-NOW

ESP-NOW je protokol pro bezdrátovou komunikaci vyvinutý firmou Espressif Systems. Tento protokol umožňuje přímou komunikaci mezi zařízeními na platformách jako ESP8266 nebo ESP32 bez potřeby připojení k Wi-Fi síti. ESP-NOW je obzvláště užitečný pro aplikace Internetu věcí, protože umožňuje zařízením komunikovat s extrémně nízkou latencí a spotřebou energie [12].

7.1.1 Klíčové vlastnosti ESP-NOW

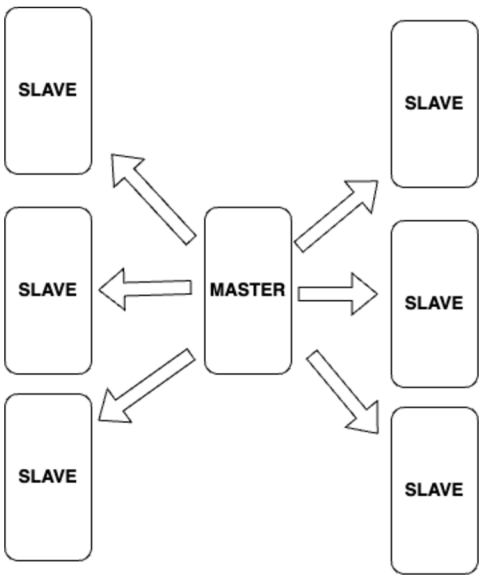
Díky schopnosti přenášet data rychleji než běžné Wi-Fi spojení je tento protokol vhodný pro aplikace, které vyžadují okamžitou odezvu. Jednou z klíčových výhod ESP-NOW je jeho snadná konfigurace, která umožňuje fungování bez nutnosti složité infrastruktury. Protokol také podporuje šifrování, což zajišťuje vysokou úroveň bezpečnosti přenášených dat. Flexibilita ESP-NOW je dalším významným atributem, protože umožňuje zařízením udržovat běžné Wi-Fi připojení, zatímco paralelně komunikují pomocí tohoto protokolu, což rozšiřuje možnosti jejich využití.

7.1.2 Komunikace pomocí ESP-NOW

ESP-NOW využívá peer-to-peer komunikaci, což umožňuje přímou výměnu dat mezi zařízeními bez nutnosti připojení k Wi-Fi síti. Tento protokol podporuje několik topologií přenosu dat, včetně jednoho mastera s více slave zařízeními (one master, multiple slaves), jednoho slave s více master zařízeními (one slave, multiple masters) a více masterů komunikujících s více zařízeními (multiple masters, multiple slaves) [16].

7.1.3 Topologie – One Master, Multiple Slaves

V tomto režimu jedno hlavní zařízení (master) komunikuje s více podřízenými zařízeními (slaves). Master odesílá data jednotlivým slave zařízením, která na ně mohou okamžitě reagovat nebo odpovědět zpět. Tento model je běžně využíván například v bezdrátových senzorových sítích, kde master přijímá data z různých senzorů a centrálně je zpracovává [17].



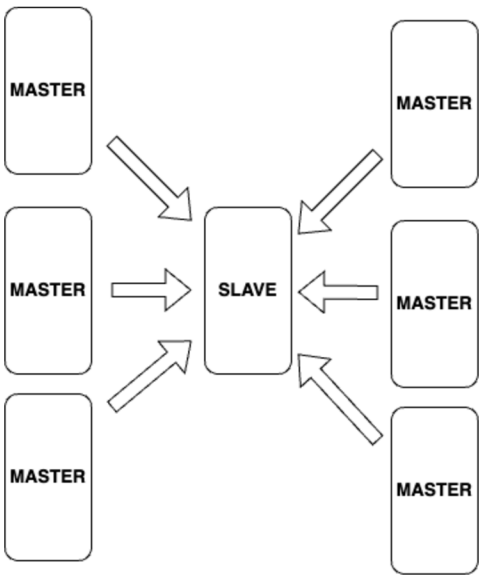
Obrázek 1 One master, Multiple slaves

Zdroj: Vlastní

- Master posílá data slave uzlům, každé slave zařízení je identifikováno pomocí své MAC adresy.
- Slaves odpovídají na základě požadavků, mohou data pouze přijímat nebo i odesílat zpět.
- **Výhody**: nízká latence, efektivní přenos dat, nízká spotřeba energie.
- **Nevýhody**: master musí spravovat adresář zařízení, což vyžaduje více paměti a výpočetního výkonu.

7.1.4 Topologie – One Slave, Multiple Masters

Tato topologie umožňuje jednomu slave zařízení přijímat data z více master zařízení. Tento režim je užitečný například v aplikacích, kde více masterů musí kontrolovat a řídit jeden senzor nebo akční člen.



Obrázek 2 One slave, Multiple Masters

Zdroj: Vlastní

- Slave přijímá příkazy nebo data od různých master uzlů, vhodné pro systémy, kde více entit musí monitorovat stejné zařízení.
- Každý master může nezávisle komunikovat se stejným slave zařízením, to umožňuje redundantní řízení nebo záznam dat z více zdrojů.
- **Výhody**: Vyšší dostupnost zařízení, možnost více vstupních bodů do systému.
- **Nevýhody**: Složitější synchronizace příkazů, potenciální konflikty mezi master uzly [17].

7.1.5 Topologie – Multiple Masters, Multiple Slaves

Tento režim umožňuje více hlavním zařízením (masters) vzájemně komunikovat mezi sebou i s více podřízenými zařízeními (slaves). Každý master může spravovat svou skupinu slave zařízení, přičemž jednotlivé slaves mohou přijímat data od různých masterů. Tento model je vhodný pro rozsáhlé sítě, kde je potřeba decentralizovaná správa dat, například v chytrých budovách nebo průmyslových aplikacích [18].

- Každý master komunikuje s vlastní skupinou slave zařízení, umožňuje rozdělení úloh mezi více jednotek.
- Slaves mohou přijímat příkazy od různých masterů, podporuje vyšší flexibilitu v síti.
- **Výhody**: Větší škálovatelnost, možnost decentralizovaného řízení, vyšší redundance.

- **Nevýhody**: Složitější správa komunikace mezi více uzly, nutnost řízení kolizí při přenosu.

7.1.6 Vrstvy ESP-NOW

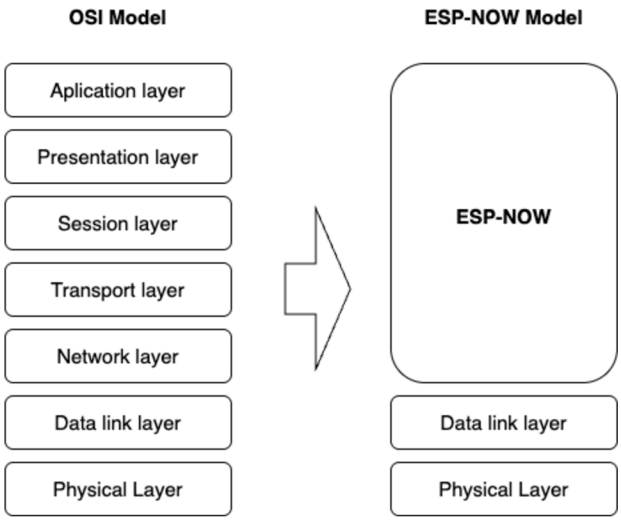
Model ISO/OSI (International Organization for Standardization/Open Systems Interconnection) je tradiční síťový model, který rozděluje komunikaci do sedmi vrstev od fyzického přenosu až po aplikaci. V praxi se používá spíš jako referenční rámec.

Na rozdíl od toho ESP-NOW je zjednodušený komunikační protokol, určený hlavně pro mikrokontroléry ESP32 a ESP8266. V tomto modelu se téměř všechny vyšší vrstvy (aplikační, prezentační, relační, transportní a síťová) slévají do jediné vrstvy označené jako ESP-NOW.

ESP-NOW funguje přímo nad linkovou vrstvou [19] , bez nutnosti použít Wi-Fi síť nebo TCP/IP protokol. Klíčové vlastnosti:

- **Nízká latence** – velmi rychlá komunikace mezi zařízeními.
- **Nízká spotřeba energie** – ideální pro bateriová zařízení a IoT aplikace.
- **Komunikace typu peer-to-peer** – zařízení si posílají data přímo mezi sebou.
- **Šifrování** – možnost zabezpečeného přenosu pomocí šifrovacích klíčů.
- **Bez připojení k routeru** – zařízení nemusí být připojena do stejné Wi-Fi sítě.

Tím, že ESP-NOW přebírá roli více vrstev najednou, výrazně zjednodušuje vývoj a zrychluje komunikaci v jednoduchých bezdrátových sítích, například při přenosu senzorových dat mezi zařízeními [13].



Obrázek 3 Vrstvy ESP-NOW

Zdroj: Vlastní

7.2 Protokol ESP-MESH

Technologie ESP-MESH je síťová komunikační technologie, která umožňuje vzájemnou komunikaci mezi zařízeními ESP8266 nebo ESP32. Tato technologie je zvlášť vhodná pro aplikace v oblastech, kde je potřeba spojení velkého počtu zařízení, například v automatizaci domácnosti, inteligentních budovách nebo v rozsáhlých IoT projektech [20].

7.2.1 Klíčové vlastnosti ESP-MESH

Technologie ESP-MESH se vyznačuje samoorganizačními schopnostmi, které umožňují automatickou organizaci sítě a snadné přidávání nových zařízení bez potřeby ruční konfigurace. Je samoopravná, což znamená, že v případě výpadku jednoho nebo více uzlů dokáže automaticky rekonfigurovat své spojení a trasy pro zachování nepřetržité komunikace. Díky škálovatelnosti může obsahovat a efektivně spravovat tisíce uzlů, což je ideální pro rozsáhlé systémy. Nízká spotřeba energie je další klíčovou vlastností, která je zásadní pro zařízení na baterie a zajišťuje jejich delší výdrž. Vícecestné směrování umožňuje přenos dat různými cestami k cílovému uzlu, což zvyšuje spolehlivost a rychlost přenosu dat v síti [21].

7.2.2 Komunikace pomocí ESP-MESH

Komunikace v síti ESP-MESH probíhá bez potřeby centrálního routeru nebo přístupového bodu, což umožňuje vytvoření decentralizované a robustní sítě. Každé zařízení v síti může fungovat jako uzel (node), který nejen přijímá a odesílá data, ale také předává zprávy mezi ostatními zařízeními. To umožňuje efektivní pokrytí rozsáhlých oblastí a vyšší odolnost vůči výpadkům jednotlivých uzlů.

ESP-MESH využívá Wi-Fi protokol k přenosu dat mezi zařízeními. Data mohou být směrována dynamicky na základě nejvýhodnější cesty mezi odesílatelem a příjemcem. Každý uzel v síti si udržuje seznam sousedních uzlů a optimalizuje svou trasu podle aktuálních podmínek, což minimalizuje latenci a zajišťuje stabilní připojení.

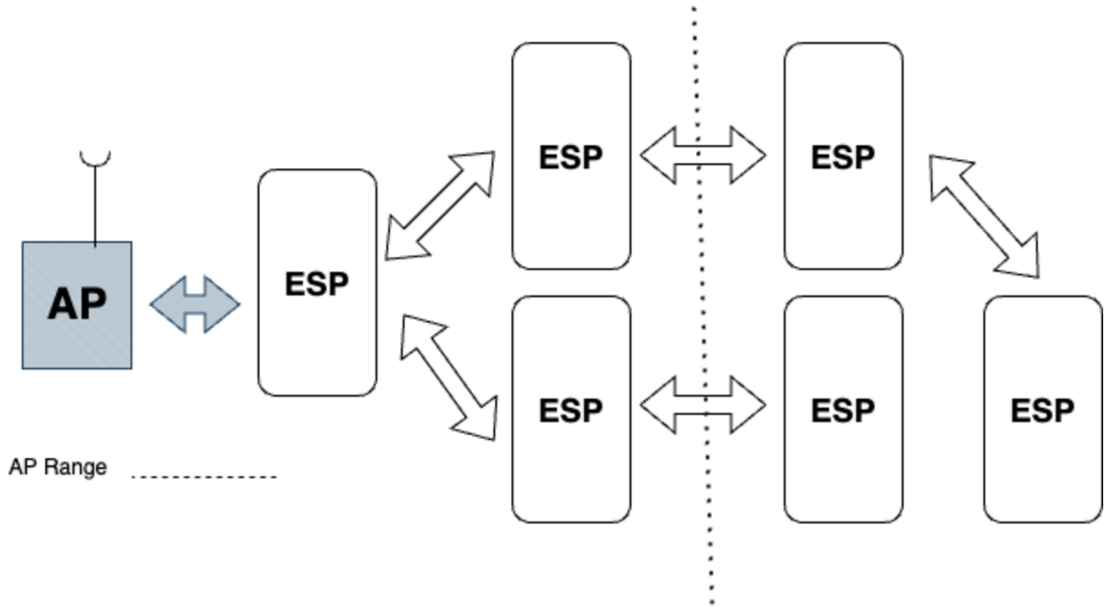
Síť ESP-MESH podporuje různé topologie komunikace:

- **Přímá komunikace** – když dvě zařízení mohou komunikovat přímo, bez zprostředkování jinými uzly.

- **Vícehopová (multi-hop) komunikace** – pokud je mezi zdrojovým a cílovým zařízením více uzlů, pak zpráva prochází přes několik mezilehlých zařízení, která ji přeposílají dál.
- **Broadcast a multicast zprávy** – umožňují jedním příkazem oslovit více zařízení najednou, což je užitečné například při synchronizaci nebo hromadné aktualizaci nastavení.

ESP-MESH umožňuje bezpečnou komunikaci díky šifrování dat a autentizaci mezi zařízeními. Tím je zajištěno, že zprávy v síti nejsou zachytávány nebo falšovány neoprávněnými uzly.

Použití ESP-MESH je výhodné zejména v prostředích, kde je nutné překonávat fyzické překážky (např. ve velkých budovách nebo venkovních oblastech), kde tradiční Wi-Fi signál nemusí mít dostatečný dosah. Díky své flexibilitě a škálovatelnosti je tato technologie ideální pro chytré domy, automatizaci průmyslových procesů a rozsáhlé IoT aplikace [22].



Obrázek 4 ESP-MESH Síť

Zdroj: Vlastní

7.3 Porovnání ESP-NOW a ESP-MESH

ESP-MESH narozdíl od ESP-NOW vytváří rozsáhlé síťové topologie s více uzly, kde každý uzel může komunikovat s mnoha dalšími a předávat zprávy. Je to ideální pro aplikace vyžadující robustní datový přenos. Naopak, ESP-NOW umožňuje přímou komunikaci mezi zařízeními bez potřeby Wi-Fi připojení k routeru, což je vhodné pro

rychlé reakce a nízkou spotřebu energie, jako jsou dálková ovládání a senzorové sítě. ESP-MESH využívá Wi-Fi pro komunikaci (Proto je taky někdy nazývána jako ESP WIFI MESH), vyžaduje ale specifickou konfiguraci a správu sítě. ESP-NOW na druhou stranu může efektivně koexistovat s Wi-Fi a Bluetooth LE, což zařízením umožňuje udržovat nízkou energetickou spotřebu a zároveň nabízet rychlou odezvu.

7.4 Multihop komunikace v ESP sítích

Multihop komunikace představuje důležitý koncept v bezdrátových sítích, kde data nejsou přenášena pouze mezi dvěma body, ale mohou projít přes více uzlů (nody), než dosáhnou svého cíle. Tento přístup se využívá především v rozsáhlejších sítích, kde není přímé spojení mezi všemi zařízeními, což je častý případ u IoT řešení postavených na platformách jako ESP8266 a ESP32 [23].

Díky multihop komunikaci může síť pokrýt větší oblast a zároveň dosahuje vyšší spolehlivosti přenosu dat. Pokud dojde k výpadku určitého uzlu, síť je schopna dynamicky najít alternativní cestu pro doručení zprávy. To je zásadní například u senzorových sítí nasazených v prostředí s proměnlivými podmínkami nebo s překážkami v přímé viditelnosti. V některých scénářích dokonce může multihop přenos prodloužit životnost zařízení tím, že umožní odesílat data na kratší vzdálenosti, a tím šetří baterii.

7.4.1 Úvod do multihopu a jeho význam v IoT

V prostředí internetu věcí často vznikají situace, kdy senzory a akční členy nejsou v dosahu centrálního uzlu nebo základnové stanice. Multihop umožňuje vytvořit flexibilní a rozšiřitelnou síť, kde každý uzel může fungovat jako směrovač a přeposílat data dále. Tím se zvyšuje dosah celé sítě a zároveň její spolehlivost, v případě výpadku jednoho uzlu mohou data najít alternativní cestu.

Význam multihop komunikace roste s rostoucím počtem zařízení v síti. Například ve scénáři měření kvality ovzduší ve městě mohou senzory rozmístěné na velké ploše odesílat data přes „meziuzly“ až do centrální brány připojené k internetu. Bez multihopu by bylo nutné zajistit přímou komunikaci každého senzoru s bránou, což je technicky a ekonomicky náročné [22].

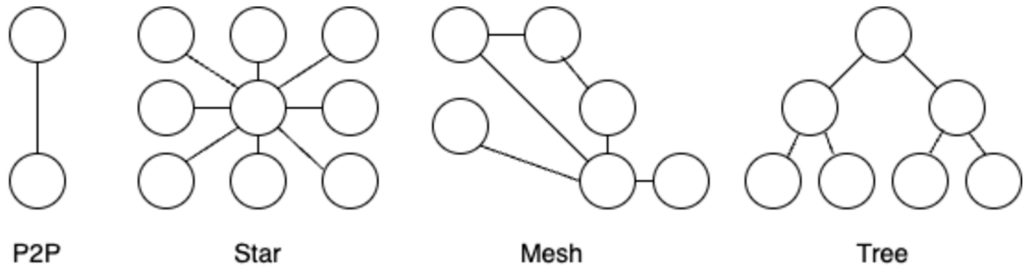
Multihop také přispívá k lepší energetické efektivitě v sítích typu WSN, kde uzly s omezeným výkonem přenášejí data v několika krátkých skocích, namísto jednoho energeticky náročného přenosu na dlouhou vzdálenost [24]. V některých návrzích sítí

se záměrně vytváří „energeticky úsporné trasy“, které snižují spotřebu baterií a prodlužují životnost celé sítě.

7.4.2 Topologie sítí

Topologie sítě definuje způsob propojení jednotlivých uzlů. V kontextu ESP zařízení a IoT sítí se nejčastěji uplatňují následující topologie:

- **Point-to-Point (P2P):** Přímé spojení mezi dvěma zařízeními. Jednoduché, ale omezené dosahem.
- **Star (hvězdicová topologie):** Všechna zařízení komunikují s jedním centrálním uzlem. Výhodou je jednoduchost řízení, nevýhodou je centrální bod selhání [25].
- **Mesh (síťová topologie):** Každý uzel může komunikovat s více sousedními uzly. Poskytuje redundanci, flexibilitu a je ideální pro multihop komunikaci [22][25].
- **Tree (stromová topologie):** Hierarchická struktura, kde data tečou od listů přes větve až ke kořeni. Používá se u některých typů clusterových systémů [25].



Obrázek 5 Topologie sítí

Zdroj: Vlastní

Schémata výše popsaných topologií jsou vyobrazena na Obrázku 5. Topologie mesh je ideální pro multihop přenos, protože umožňuje každému uzlu předávat data dále, a síť je schopna se přizpůsobit změnám v rozmístění zařízení. Tree topologie může být výhodná pro strukturované aplikace, kde je přirozená hierarchie (například měření v několika patrech budovy). Některé moderní sítě kombinují různé topologie, například kombinaci mesh s hvězdicovou strukturou pro větší robustnost a snadnější řízení provozu [25].

7.4.3 Způsoby multihop komunikace

Existuje několik metod, jak realizovat multihop komunikaci. Každá má své specifické vlastnosti, výhody a omezení:

Záplavy (Flooding): Jednoduchá technika, kde každý uzel, který přijme zprávu, ji přepošle všem svým sousedům. Výhodou je vysoká spolehlivost a jednoduchá implementace. Nevýhodou je však zahlcení sítě, opakované přenosy stejné zprávy a vysoká spotřeba energie. Flooding je vhodný především pro menší sítě nebo pro přenos zpráv s vysokou prioritou, kde je důležitější spolehlivost než efektivita [26].

Šeptání (Gossiping): Efektivnější varianta floodingu. Každý uzel si náhodně vybírá jednoho souseda, kterému zprávu pošle. Tím se snižuje zatížení sítě a dochází k rovnoměrnějšímu šíření informací. Tento princip se hodí do méně časově kritických aplikací, kde nevadí určité zpoždění. Gossiping lze navíc upravit tak, aby určité zprávy měly vyšší prioritu a byly šířeny rychleji [26].

Hierarchická síť – clustery a clusterheady: Sítě lze dále členit na menší skupiny zvané clustery, přičemž každá z nich má svůj hlavní uzel (clusterhead). Tyto hlavní uzly zajišťují komunikaci mezi jednotlivými clustery. Tento model snižuje počet přenosů a zvyšuje efektivitu řízení sítě. Vyžaduje však složitější správu a výběr clusterheadů. Využití clusterheadů je běžné například v energeticky úsporných WSN sítích, kde se role hlavního uzlu dynamicky mění.

Další možností je využití adaptivních algoritmů, které kombinují výše uvedené přístupy. Například síť může přepnout z gossiping do clusterové struktury při zvýšené zátěži. Některé pokročilé návrhy sítí využívají i prediktivní směrování na základě historie přenosů nebo očekávaného pohybu uzlů [26].

7.4.4 Výhody a nevýhody jednotlivých metod

Volba metody závisí na konkrétní aplikaci. Například pro monitoring prostředí může být gossiping ideální, zatímco pro průmyslové aplikace se uplatní hierarchický přístup s důrazem na spolehlivost a řízení. Flooding je vhodný jako záložní metoda při nouzových situacích, kdy je nutné doručit zprávu za každou cenu.

V praxi bývá často kompromisem použití omezeného floodingu, kdy se zpráva šíří pouze omezený počet skoků, což snižuje zátěž a zároveň zachovává robustnost. Také lze zavést tzv. TTL (Time To Live), kdy každá zpráva nese informaci o maximálním počtu skoků [26][27].

7.4.5 Využití v ESP-MESH a ESP-NOW

ESP-NOW je určen spíše pro jednoduché přímé komunikace s minimální režijní zátěží. Neimplementuje automaticky multihop, ale lze jej k tomuto účelu využít za cenu ručního směrování dat. Při správném návrhu lze vytvořit přeposílací logiku, kdy každý uzel zná své sousedy a předává zprávy směrem ke konkrétnímu cíli.

ESP-MESH naopak přirozeně podporuje multihop komunikaci. Každý uzel může fungovat jako přeposílací bod, síť se sama rekonfiguruje při výpadku uzlů a umožňuje velké škálování bez nutnosti přímého Wi-Fi připojení každého zařízení k routeru. Umožňuje také nastavení priorit, přesměrování provozu a distribuci zátěže mezi jednotlivé uzly.

Obecně lze říct, že multihop komunikace v ESP sítích představuje vysoce efektivní nástroj pro budování spolehlivých, škálovatelných a energeticky úsporných IoT řešení. Výběr správné topologie a metody přenosu je klíčový pro úspěšné nasazení v praxi a měl by být vždy přizpůsoben konkrétnímu účelu a provozním podmínkám dané aplikace.

8 Typy komunikačních sběrnic

Komunikační sběrnice jsou základní způsoby používané pro komunikaci mezi mikrokontroléry a periferiemi v elektronických zařízeních. Tyto sběrnice umožňují výměnu dat mezi různými součástmi, jako jsou senzory, paměťové moduly, displeje nebo jiná zařízení, a hrají klíčovou roli v oblasti vestavěných systémů a IoT.

8.1 I2C

Sběrnice I2C (Inter-Integrated Circuit) je sériová sběrnice, která umožňuje komunikaci mezi různými zařízeními na základě master-slave architektury. Sběrnice I2C je velmi flexibilní a běžně se používá pro připojení různých periferií k mikrokontrolérům, jako jsou již zmiňované senzory, displeje, EEPROM (Electrically Erasable Programmable Read-Only Memory) paměti a další zařízení. Je známá svou jednoduchou implementací a schopností propojit více zařízení na jedinou sběrnici. I2C se používá v aplikacích, kde je třeba minimalizovat počet vodičů potřebných pro propojení komponent. I2C používá pouze dva vodiče, jeden pro data (SDA - Serial Data Line) a jeden pro hodinový signál (SCL - Serial Clock Line), což zjednodušuje zapojení a snižuje náklady [28].

I2C sběrnice nabízí několik klíčových funkcí, které umožňují efektivní a flexibilní komunikaci mezi integrovanými obvody. Jednou z těchto funkcí jsou multi-master schopnosti, což znamená, že více "master" zařízení může současně ovládat sběrnici. Díky integrovanému arbitrážnímu mechanismu se přitom předejde možným kolizím, které by mohly vzniknout, když dvě zařízení v roli „master“ zahájí komunikaci ve stejný čas. Další důležitou vlastností je adresování zařízení, kde každé "slave" zařízení na sběrnici má svou unikátní adresu, umožňující masterům zasílat data přímo určeným zařízením.

Co se týče rychlostních režimů, I2C umožňuje pracovat v několika módech: standardní režim s rychlostí 100 kHz, rychlý režim s 400 kHz a vysokorychlostní režim s 3.4 MHz, což poskytuje různé možnosti pro aplikace vyžadující různé komunikační rychlosti. Tyto vlastnosti činí I2C sběrnici velmi flexibilním a široce používaným řešením pro komunikaci mezi komponentami v elektronických zařízeních. Nevýhodou může být omezená délka sběrnice a nižší rychlost přenosu dat ve srovnání s jinými protokoly [29].

8.2 SPI

Komunikační sběrnice SPI (Serial Peripheral Interface) je rozhraní určené pro vysokorychlostní komunikaci mezi mikrokontroléry a různými periferiemi, jako jsou senzory nebo paměti. Toto rozhraní, které původně vyvinula společnost Motorola, se vyznačuje svou jednoduchostí a efektivitou.

SPI pracuje na principu master-slave, kde master zařízení generuje hodinový signál a řídí celkovou komunikaci a datové výměny se slave zařízeními. Jde o synchronní systém, kde jsou data přenášena v přesném časovém souladu s hodinovými pulzy, což umožňuje velmi rychlou výměnu dat. SPI také podporuje plně duplexní přenos, což znamená, že data mohou být odesílána a přijímána současně. Typická konfigurace SPI zahrnuje čtyři signálové vodiče: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock) a SS (Slave Select). Vodič Slave Select umožňuje masteru vybrat, se kterým slave zařízením chce komunikovat, což zajišťuje flexibilitu v zapojení více periferií. Mezi hlavní výhody SPI patří jeho vysoká rychlost přenosu dat a jednoduchost implementace. Na druhé straně, hlavní nevýhody zahrnují omezení na počet zařízení, které lze připojit bez dodatečné logiky pro rozšíření, a nutnost použít více vodičů než u některých jiných sběrnic, jako je I2C [30].

SPI je vhodné pro aplikace, kde je prioritou rychlost a spolehlivost přenosu dat, ačkoliv vyžaduje více vodičů pro komunikaci. Toto rozhraní je běžně využíváno v zařízeních jako jsou digitální kamery, SD (Secure Digital) karty a různé typy senzorů, kde jsou tyto charakteristiky klíčové.

8.3 One-Wire

One-Wire je komunikační protokol, který umožňuje efektivní přenos dat mezi jedním master zařízením a jedním nebo více slave zařízeními pouze pomocí jediného komunikačního vodiče. Vyvinutý firmou Dallas Semiconductor je široce využíván v různých aplikacích díky své jednoduchosti a nízkým nákladům na implementaci.

Hlavní výhodou One-Wire sběrnice je schopnost komunikace i napájení zařízení pomocí jediného vodiče. To je zvláště přínosné v prostředích s omezeným prostorem nebo tam, kde je cílem minimalizace nákladů na hardwarové komponenty. Pro běžnou funkčnost jsou potřebné pouze dva vodiče a to datová linka a země, což výrazně zjednodušuje návrh zapojení [31].

Další specifickou vlastností je možnost využití tzv. parasitic power, kdy mnoho zařízení připojených na sběrnici je napájeno přímo z datového vodiče. Tento přístup eliminuje potřebu samostatného napájecího vodiče, což dále snižuje náklady a zvyšuje jednoduchost konstrukce [32]

Každé zařízení komunikující přes One-Wire je vybaveno unikátním 64bitovým sériovým číslem, které umožňuje master zařízení jednoznačně identifikovat a adresovat jednotlivé uzly na sběrnici. Díky tomu je možné bezchybně komunikovat i ve složitějších systémech s větším počtem připojených zařízení.

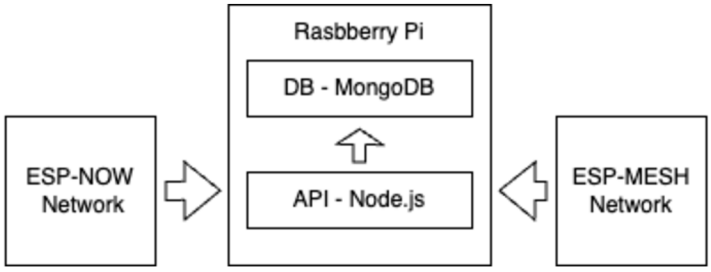
Typickými aplikacemi, kde se One-Wire využívá, jsou například měření teploty, čtení dat z EEPROM pamětí nebo autentizační zařízení v bezpečnostních systémech. Příkladem je teplotní senzor DS18B20 od společnosti Dallas Semiconductor, který využívá právě tento komunikační protokol pro přenos naměřených hodnot do nadřazeného systému [32].

Mezi hlavní výhody One-Wire sběrnice patří jednoduché zapojení, minimální počet vodičů a možnost napájení zařízení přímo z datové linky. Na druhé straně je třeba počítat s určitými omezeními, jako je nižší rychlost přenosu dat oproti sběrnicím typu I2C nebo SPI, a omezená délka vodiče a počet připojených zařízení, které mohou být limitovány kapacitními a odporovými ztrátami při delších vzdálenostech.

9 Implementace sítě

Tato část práce se detailně zaměřuje na samotnou implementaci dvou bezdrátových komunikačních protokolů, konkrétně ESP-NOW a ESP-MESH, které byly zvoleny pro svou vhodnost v prostředí bezdrátových senzorových sítí. Cílem této části je podrobně popsat celý proces nasazení, který začíná fyzickou instalací jednotlivých senzorů, pokračuje konfigurací příslušných ESP modulů a končí kompletním nastavením komunikace, ukládání a vizualizace naměřených dat prostřednictvím zařízení Raspberry Pi.

V rámci navržené implementace byly vytvořeny dvě samostatně fungující bezdrátové sítě, z nichž každá zajišťuje sběr naměřených dat z připojených senzorů a následný přenos těchto dat na společný API server. Tento server je implementován v prostředí Node.js a běží přímo na zařízení Raspberry Pi. Přijatá data jsou na serveru dále zpracována a pomocí ORM (Object-Relational Mapping) rozhraní jsou uložena do objektově orientované databáze MongoDB, která umožňuje flexibilní a efektivní práci se strukturovanými záznamy. Data uložená v databázi jsou následně zobrazována ve formě přehledných grafů v uživatelském dashboardu.



Obrázek 6 Blokové schéma implementace

Zdroj: Vlastní

Obě sítě využívají tzv. root node, tedy centrální uzel, který hraje klíčovou roli při sběru a přeposílání dat z ostatních zařízení (nodů) v síti. Tento root node (viz Obrázek 7), který je vizuálně odlišen bílou krabičkou, nemá připojené žádné senzory. Jeho jedinou funkcí je zajištění příjmu dat z ostatních zařízení a jejich následné odesílání na vzdálený API server.

Obrázek 7 Výsledná síť ESP

Zdroj: Vlastní

9.1 Senzory pro sběr dat

Senzory hrají klíčovou roli v měření fyzikálních veličin a sběru dat v elektronických systémech. V závislosti na způsobu zpracování výstupního signálu je dělíme na analogové a digitální. Každý typ má své specifické výhody a nevýhody, které je třeba zohlednit při návrhu elektronických zařízení.

9.2 Analogové senzory

Analogové senzory představují klasický způsob měření fyzikálních veličin, při kterém je výstupním signálem nejčastěji napětí či proud. Tyto hodnoty se následně převádějí pomocí A/D (Analog/Digital) převodníku (ADC – Analog Digital Converter) na digitální čísla, se kterými může mikrokontrolér dále pracovat. Například u 10bitového ADC odpovídá výstupní napětí hodnotám v rozsahu 0 až 1023 [33]. Mezi běžné analogové senzory patří například termistory, jejichž odpor se mění v závislosti na teplotě [34], nebo fotorezistory reagující na intenzitu světla.

Použití analogových senzorů vyžaduje určitou míru dodatečného zpracování. Signál je potřeba upravit, škálovat a často i filtrovat, aby bylo možné získat přesná a spolehlivá data [35]. Kromě toho jsou tyto senzory náchylnější k elektrickému rušení, jelikož analogové signály nejsou tak odolné jako digitální. Přesto jsou díky své jednoduchosti, nízké ceně a nenáročnosti na hardware stále používány v mnoha základních aplikacích.

9.3 Digitální senzory

Digitální senzory poskytují modernější přístup k měření a přenosu dat. Naměřené hodnoty jsou v těchto zařízeních digitalizovány interně a odesílány ve formě binárního

kódu prostřednictvím komunikačních protokolů, jako jsou I2C, SPI nebo One-Wire do mikrokontroléru, kde dojde k dalšímu zpracování. To znamená, že senzory nepotřebují externí A/D převodník, a jejich výstupy jsou již připravené k přímému využití bez nutnosti škálování nebo složitého zpracování.

Mezi hlavní výhody digitálních senzorů patří vyšší přesnost, stabilita a odolnost vůči rušení [36]. Na rozdíl od analogových senzorů totiž nejsou citlivé na změny napětí nebo elektromagnetický šum, což je činí spolehlivějšími zejména v náročných podmínkách. Digitální senzory také často umožňují připojení více zařízení na jednu sběrnici, čímž šetří počet potřebných vstupních pinů a usnadňují návrh zapojení. Vzhledem ke své přesnosti a univerzálnosti jsou dnes digitální senzory běžnou součástí zařízení v oblasti IoT, meteorologie, automatizace, zdravotnictví či průmyslové techniky [37].

9.4 Porovnání analogových a digitálních senzorů

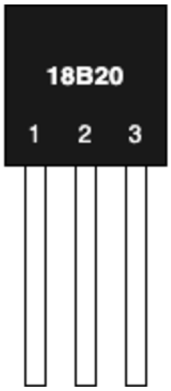
Analogové a digitální senzory se liší především způsobem zpracování a přenosu dat. Zatímco analogové senzory generují spojitý elektrický signál, který je potřeba dále digitalizovat a škálovat, digitální senzory poskytují přímo čitelná data bez nutnosti dalších úprav. Analogové senzory jsou jednodušší z hlediska zapojení a levnější, ale trpí vyšší náchylností k rušení a nižší přesností. Digitální senzory naopak nabízejí vyšší míru spolehlivosti, možnost sofistikované komunikace a snadnou integraci do komplexních systémů. V praxi tak volba mezi analogovým a digitálním senzorem závisí na konkrétní aplikaci, jednoduché a nákladově efektivní řešení si stále vystačí s analogovým senzorem, zatímco složitější a přesnější systémy dnes spoléhají převážně na senzory digitální. Jedním z typických představitelů digitálních senzorů, který vyniká svou přesností a snadnou implementací, je teplotní senzor DS18B20, jehož vlastnostem a možnostem použití se věnuje následující kapitola.

9.4.1 Senzor teploty (DS18B20)

Teplotní senzor DS18B20, vyráběný společností Maxim (dříve Dallas), je oblíbeným řešením pro měření teploty díky své jednoduchosti a dostupnosti. Lze jej zakoupit ve dvou variantách, jako třípinový senzor ve tvaru tranzistoru nebo ve vodotěsném provedení s prodlužovacím kabelem, což umožňuje jeho použití i v náročnějších venkovních podmínkách.

Senzor disponuje třemi vodiči: GND (zemnící pin), VCC (napájecí pin) a Data (komunikační pin). Díky jedinému datovému pinu senzor využívá One-Wire sběrnici,

která umožňuje připojení více senzorů na jeden signálový vodič, čímž lze výrazně snížit množství potřebné kabeláže. Kromě klasického třívodičového zapojení podporuje i tzv. parazitní režim, kdy je napájení a komunikace vedena po stejném vodiči [38].



Obrázek 8 Senzor 18B20

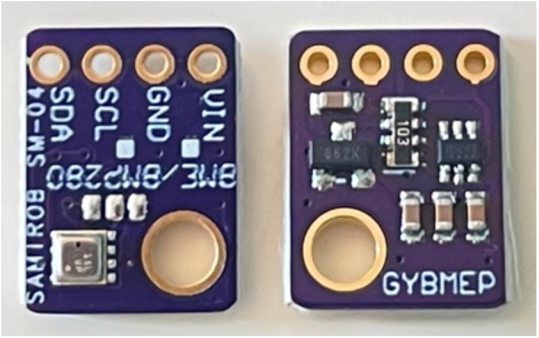
Zdroj: Vlastní

DS18B20 nabízí široký rozsah měření od -55 °C do +125 °C, přičemž v užším pásmu od -10 °C do +85 °C je zajištěna vysoká přesnost s maximální odchylkou pouhých 0,5 °C. Napájecí napětí se pohybuje mezi 3.0 V a 5.5 V, což umožňuje snadnou integraci do různých elektronických aplikací. Cena standardní varianty senzoru se pohybuje okolo 50 Kč (2025), zatímco verze s metrovým kabelem je dostupná přibližně za 100 Kč [38].

Díky své jednoduché konstrukci, nízké ceně a širokým možnostem využití je DS18B20 skvělou volbou pro začínající i pokročilé nadšence. Lze jej využít v domácí automatizaci, průmyslových aplikacích, IoT projektech nebo pro měření teploty v různých experimentech a senzorech.

9.4.2 Senzor tlaku, teploty a vlhkosti (BME280)

Mezi další senzory je možné zařadit modul BME280 od společnosti Bosch, který umožňuje měření teploty, atmosférického tlaku a relativní vlhkosti. Tento senzor podporuje komunikaci prostřednictvím sběrnic I2C nebo SPI, v závislosti na konkrétní variantě. Díky své kompaktní velikosti jej lze snadno umístit do různých zařízení, a díky integrovanému napěťovému regulátoru je možné jej napájet napětím 1.8 – 5V (například při připojení k Arduinu) bez nutnosti externí regulace [39].



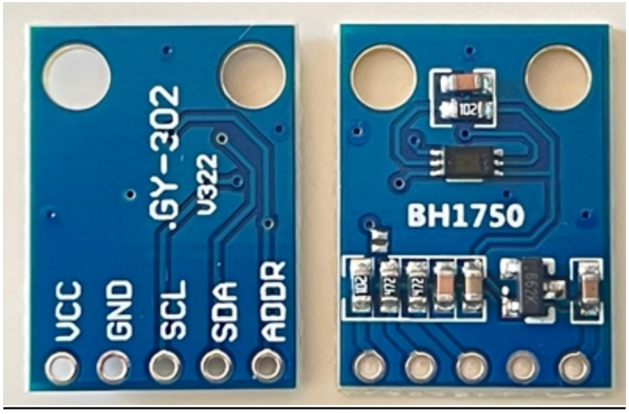
Obrázek 9 Senzor BME280

Zdroj: Vlastní

BME280 nabízí rozsah měření teploty od -40 °C do +85 °C s maximální odchylkou 1 °C, relativní vlhkosti od 0 % do 100 % s odchylkou 3 %, a tlaku v rozmezí 300 hPa až 1100 hPa s přesností 1 hPa [39]. Existují i podobné senzory, jako BMP280, BMP180 a BMP085, které mají obdobné vlastnosti, avšak neumožňují měření relativní vlhkosti. Cena tohoto modulu se pohybuje kolem 140 Kč (v roce 2025).

9.4.3 Senzor intenzity světla (BH1750)

Senzor BH1750 je určen pro měření intenzity osvětlení dopadajícího na jeho povrch. Naměřená data odesílá prostřednictvím I2C nebo SPI sběrnice, přičemž výstupní hodnota je celočíselná a udává se v luxech (lux). Díky jednoduchému použití a dostupnosti je tento modul vhodný zejména pro začínající nadšence, kteří chtějí měřit intenzitu světla ve svých projektech [40].



Obrázek 10 Senzor BH1750 ve Variantě I2C

Zdroj: Vlastní

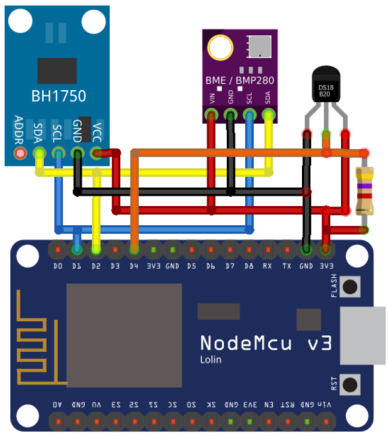
Senzor pracuje s napájecím napětím od 3.3 V do 5.5 V, přičemž umožňuje měření intenzity světla v rozsahu 0 až 65 535 luxů. Jeho provozní teplota se pohybuje mezi -40 °C a +85 °C, což umožňuje použití v různých prostředích. Proudový odběr senzoru

je pouze 0.12 mA, takže je vhodný i pro nízkoenergetické aplikace. Měření probíhá v intervalu 120–180 ms, což poskytuje dostatečně rychlou odezvu pro většinu běžných použití. Cena modulu se v roce 2025 pohybuje kolem 60 Kč [40].

Díky své přesnosti a snadné integraci nachází BH1750 uplatnění v automatizaci osvětlení, chytrých domácnostech, meteorologických stanicích a dalších projektech, kde je potřeba měřit intenzitu světla.

9.5 Zprovoznění nodů

Prvním krokem je fyzické zapojení senzorů. V této demonstraci je využito typu ESP 8266 Node MCU V3 a senzorů BME280, DS18B20 a BH1750 v I2C verzích. Jelikož senzory BME280 a BH1750 podporují sběrnici I2C bude využito právě této sběrnice. Propojením D1 (Digitální pin 1) s pinem SCL na senzorech a D2 s pinem SDA na senzorech se zajistí plnohodnotná komunikace. Kromě komunikace je třeba zajistit napájení, jelikož Node MCU pracuje na 3,3V je výhodné využít právě tohoto napájení z některého z pinů ESP. Dále je třeba zajistit i propojení nulového vodiče (GND). Senzor DS18B20 je připojen na datový pin D4 (GPIO2), přičemž je třeba nezapomenout na připojení pull-up rezistoru mezi datový pin a napájecí napětí.



Obrázek 11 Schéma zapojení senzorů

Zdroj: Vlastní

Vypracováno v softwaru: Fritzing [41]

Následně je nutné nainstalovat vhodné vývojové prostředí pro práci s deskami ESP8266. Zde bude demonstrováno Arduino IDE verze 2 které je open source a využívá se nejčastěji i z důvodu největší kompatibility, nicméně lze využít např. Visual Studio Code s doplňkem pro Arduino. Instalační balíček, pro aktuální verzi Arduino IDE 2, je

možné stáhnout z oficiálních stránek developera [42]. Díky tomuto softwaru je možné psát a upravovat kódy, nicméně hlavně je možné kód zkompilovat a nahrát do mikrokontroleru.

Před samotným nahráním kódu se je třeba ujistit že máme potřebné ovladače k sériovému převodníku dané desky, zde v případě originální verze Node MCU V3 je ovladač součástí většiny systémů, nicméně v případě modelu s převodníkem CH340 je třeba instalace ovladače pro převodník CH340.

9.6 Implementace sítě ESP-NOW

Každé zařízení typu **slave** představuje jeden uzel vybavený senzory, který měří fyzikální veličiny v tomto případě: teplotu, vlhkost, tlak, intenzitu světla. Následně tyto hodnoty odesílá zařízení typu **master**. Master funguje jako centrální uzel, který přijímá data ze všech podřízených jednotek. Tato struktura umožňuje snadno škálovat celý systém, přidáním dalších slave uzlů se zvyšuje počet měřených míst bez nutnosti zásahu do architektury masteru. Každé zařízení potřebuje znát **MAC adresu** masteru, aby mohlo odesílat data.

Konfigurace slave zařízení

Na straně slave probíhá v úvodu kódu inicializace všech potřebných knihoven a proměnných. Každé zařízení má přiděleno jednoznačné ID, které je později součástí odesílaných dat.

```
typedef struct send_message {
   short id;
   float temperature;
   float temperature2;
   float humidity;
   float pressure;
   float intensity;
} send_message;
send_message sendMessage;
```

Zdrojový kód 1 Formát odesílané zprávy

Zdroj: Vlastní

Odeslaná zpráva obsahuje všechny naměřené hodnoty, které jsou odesílány ve formě binárního bloku. V programu na zařízení ESP je nitné importovat knihovnu *espnow* [43]. Následuje funkce *setup()* s přepnutím zařízení do režimu Wi-Fi stanice pomocí příkazu *WiFi.mode(WIFI_STA).* Následně je spuštěna knihovna ESP-NOW, která

umožňuje bezdrátovou komunikaci mezi zařízeními bez nutnosti připojení k Wi-Fi síti. To se provede příkazem *esp_now_init()*.

Dalším krokem je registrace cílového zařízení (tzv. peer), které bude přijímat zprávy. Zde se specifikuje MAC adresa zařízení a role příjemce, v tomto případě jako **ESP_NOW_ROLE_SLAVE**, tedy příjemce zpráv. K registraci slouží funkce:

```
esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);
```

Pro správné fungování komunikace je zároveň potřeba registrovat funkci zpětného volání, která bude informovat, zda bylo odeslání zprávy úspěšné. Tuto roli plní příkaz:

```
esp_now_register_send_cb(OnDataSent);
```

Ve funkci *loop()* probíhá samotné čtení hodnot ze senzorů a odeslání zprávy v pravidelných intervalech:

```
sendMessage.id = 1;
sendMessage.temperature = bme.readTemperature();
sendMessage.humidity = bme.readHumidity();
sendMessage.pressure = bme.readPressure() / 100.0F;
sendMessage.intensity = bh.readLightLevel();
esp_now_send(broadcastAddress, (uint8_t *) &sendMessage,
sizeof(sendMessage));
```

Zdrojový kód 2 Odeslání zprávy ESP-NOW

Zdroj: Vlastní

Každé zařízení má jiný ID, což umožňuje masteru identifikovat zdroj dat.

Konfigurace master zařízení

Na straně mastera je nutné především poslouchat příchozí data od všech registrovaných slave zařízení. Struktura pro příjem dat je definována stejně jako na straně slave, ale přejmenována ze *send_message* na *received_message*.

Master inicializuje Wi-Fi a ESP-NOW stejně jako slave, ale kromě toho registruje funkci zpětného volání pro příjem dat:

```
esp_now_register_recv_cb(DataReceivedCallback);
```

Funkce *DataReceivedCallback()* je zavolána pokaždé, když některý ze slave uzlů odešle zprávu. V rámci této funkce se přijatá binární data zkopírují do struktury

receivedMessage, kde je s nimi dále pracováno. Master díky přítomnosti *receivedMessage.id* ví, ze kterého konkrétního uzlu data pocházejí.

Spolupráce mezi masterem a slaves

Každý slave periodicky měří hodnoty ze senzorů a následně sestavuje strukturovaný datový blok, který obsahuje aktuální stav prostředí. Tento blok je pomocí ESP-NOW zaslán přímo master zařízení, bez potřeby prostředníka nebo připojení k routeru.

Master přijímá data od všech uzlů nezávisle a díky unikátním ID může každému z nich přiřadit identitu. Systém je robustní a snadno rozšiřitelný, pro přidání dalšího měřicího uzlu stačí definovat jeho MAC adresu na straně slave a použít ji při registraci peera.

9.6.1 Implementace sítě ESP-MESH

Implementace ESP-MESH sítě je rozdělena do dvou základních částí, zařízení plnící roli brány (*root node*) a jednotlivé uzly (*nodes*), které sbírají data ze senzorů a posílají je dále v rámci sítě. K realizaci byla využita knihovna *painlessMesh [44]*, která umožňuje snadno vytvářet dynamické sítě mezi zařízeními ESP8266 nebo ESP32.

Root node

Brána představuje hlavní sběrný bod sítě. Je připojena k internetu prostřednictvím lokální Wi-Fi a současně je plně zapojená do ESP-MESH sítě. Její hlavní funkcí je příjem dat z uzlů a jejich případné předání dál, například na server.

Inicializace sítě probíhá pomocí metody *mesh.init()*, kde se kromě přihlašovacích údajů a portu specifikuje i režim připojení. V tomto případě je použito *WIFI_AP_STA*, tedy režim, který umožňuje být současně přístupovým bodem i klientem jiné Wi-Fi sítě. V rámci funkce *setup()* je zároveň nastavena funkce zpětného volání pomocí *mesh.onReceive()*, která se automaticky spustí při přijetí zprávy od jiného zařízení v síti.

Zprávy přijaté bránou jsou předávány do uživatelsky definované funkce *SendToApi()*. Tato funkce slouží k odesílání dat na webové rozhraní API.

Pro správný běh celé sítě je nutné v nekonečné smyčce *loop()* pravidelně volat metodu *mesh.update()*. Tato metoda zajišťuje všechny vnitřní procesy spojené s komunikací, synchronizací mezi uzly a případným směrováním zpráv v síti.

Uzly (Nodes)

Každý uzel je vybaven senzory a jeho hlavní úlohou je sběr a pravidelné odesílání environmentálních dat v podobě strukturované zprávy ve formátu JSON. Kromě

inicializace senzorů v metodě *setup()* je klíčová funkce *sendMessage()*, která je spouštěna periodicky pomocí plánovače úloh (*scheduler*).

Plánovač umožňuje naplánovat určitou akci, například odeslání zprávy, v definovaném intervalu bez použití blokujících funkcí jako *delay()*. V tomto případě je pomocí objektu *Task* nastavena úloha, která se aktivuje každých 5 sekund.

Ve funkci *sendMessage()* jsou nejprve načteny hodnoty ze senzorů, a poté sestavena JSON zpráva. Ta obsahuje údaje o teplotě, vlhkosti, tlaku, intenzitě osvětlení a ID uzlu. Výsledná zpráva je následně odeslána do celé sítě pomocí *mesh.sendBroadcast()*, což zajistí, že ji obdrží i root node.

Také v případě uzlu je hlavní smyčka *loop()* velmi jednoduchá, obsahuje pouze volání *mesh.update()*. Tím je zajištěno nejen samotné fungování sítě, ale také běh naplánovaných úloh. Díky této architektuře je chování zařízení predikovatelné, nenáročné na výkon a přehledně oddělené. Čtení senzorů je řešeno nezávisle na ostatní síťové logice.

9.7 Měření jednotlivých veličin

Pro měření teploty, vlhkosti, atmosférického tlaku a intenzity osvětlení byly použity senzory BME280, DS18B20 a BH1750. Všechny senzory jsou zapojeny k mikrokontroléru ESP pomocí digitálních sběrnic. Senzory BME280 a BH1750 využívají sběrnici I2C, zatímco DS18B20 komunikuje přes One-Wire protokol.

Měření probíhá periodicky, přičemž hodnoty jsou načítány ze senzorů, zpracovány a následně připraveny k dalšímu přenosu. Při každém cyklu jsou snímány všechny dostupné veličiny v daném čase, což umožňuje jejich vzájemné porovnání a sledování vývoje v čase. Důraz je kladen na správnou inicializaci senzorů, stabilní komunikaci a konzistentní čtení dat.

9.7.1 Měření senzorem BME280

Senzor BME280 slouží k měření tří základních fyzikálních veličin: teploty, relativní vlhkosti a atmosférického tlaku. Komunikuje pomocí sběrnice I2C, což umožňuje jednoduché připojení více zařízení na stejné linky SDA a SCL. Pro potřeby měření byla zvolena I2C varianta senzoru s defaultní adresou senzoru nastavenou na hodnotu *0x76*.

Na začátku kódu je nutné připojit knihovnu *Adafruit_BME280 [45]*. Pro správné fungování senzoru je nejprve nutné definovat jeho adresu a vytvořit instanci objektu typu *Adafruit_BME280*, která slouží k ovládání senzoru.

Inicializace senzoru probíhá během spouštění systému, obvykle v metodě *setup()*. Při inicializaci se naváže komunikace se senzorem na uvedené adrese. Pokud je inicializace úspěšná, lze začít se čtením dat.

```
#include <Adafruit_BME280.h>
#define BME280_ADDRESS (0x76)
Adafruit_BME280 bme;
setup(){
   bme.begin(BME280_ADDRESS);
}
getValues(){
   … = bme.readTemperature();
   … = bme.readHumidity();
   … = bme.readPressure() / 100.0F;
}
```

Zdrojový kód 3 Měření senzorem BME280

Zdroj: Vlastní

Při každém měřicím cyklu se načítají hodnoty teploty, vlhkosti a tlaku. Hodnota tlaku je převedena z Pascalů na hektopascaly (*/100.0F*).

Měření pomocí senzoru BME280 je spolehlivé a rychlé, přičemž výsledné hodnoty umožňují sledování environmentálních podmínek v reálném čase.

9.7.2 Měření senzorem BH1750

Senzor BH1750 slouží k měření intenzity osvětlení. Jedná se o digitální světelný senzor, který komunikuje pomocí I2C sběrnice a vrací přímo hodnotu v jednotkách lux, což výrazně zjednodušuje zpracování dat bez nutnosti dalších výpočtů nebo kalibrací. Díky své nízké spotřebě a spolehlivosti se často používá v projektech pro monitorování světelných podmínek v interiéru i exteriéru.

Pro správnou funkci senzoru je nutné nejprve importovat příslušné knihovny, nejprve *Wire [46]* pro I2C komunikaci a *BH1750 [47]* pro samotný senzor. Následuje vytvoření instance objektu *BH1750*, která slouží k ovládání senzoru. Inicializace senzoru se provádí ve funkci *setup()* pomocí metody *begin()*. Touto metodou se zahájí komunikace a připraví senzor na měření.

Samotné měření probíhá jednoduše, stačí zavolat metodu *readLightLevel()*, která vrátí aktuální intenzitu světla v luxech.

9.7.3 Měření senzorem DS18B20

Senzor DS18B20 je digitální teplotní čidlo, které komunikuje pomocí One-Wire sběrnice. Hlavní výhodou tohoto senzoru je možnost připojení více zařízení na jediný datový pin, což umožňuje vytvářet jednoduché a efektivní měřicí sítě. DS18B20 poskytuje přesné teplotní údaje s rozlišením až 12 bitů a je široce používán v aplikacích, kde je kladen důraz na nízkou spotřebu a spolehlivost.

Pro komunikaci se senzorem je nutné použít dvě knihovny: *OneWire* [48], která zajišťuje podporu One-Wire protokolu, a *DallasTemperature* [49], která poskytuje nadstavbu pro pohodlnou práci s teplotními senzory DS18x20.

Datový pin, ke kterému je senzor připojen, je definován jako konstanta. Následuje vytvoření instance *OneWire* na daném pinu a následné předání této instance knihovně *DallasTemperature.*

V metodě *setup()* je třeba inicializovat senzor pomocí metody *begin()*, čímž se naváže komunikace s připojenými zařízeními.

Měření teploty probíhá ve dvou krocích. Nejprve se pomocí *requestTemperatures()* požádá o aktualizaci hodnot ze senzoru, následně se pomocí *getTempCByIndex(0)* získá teplota prvního nalezeného senzoru (index 0).

```
#include <OneWire.h>
#include <DallasTemperature.h>
const int ds_pin = 4;
OneWire oneWireDS(ds_pin);
DallasTemperature dsSensor(&oneWireDS);
setup(){
   dsSensor.begin();
}
getValues(){
   dsSensor.requestTemperatures();
   … = dsSensor.getTempCByIndex(0);
}
```

Zdrojový kód 4 Měření senzorem DS18B20

Zdroj: Vlastní

9.8 Příprava Raspberry Pi

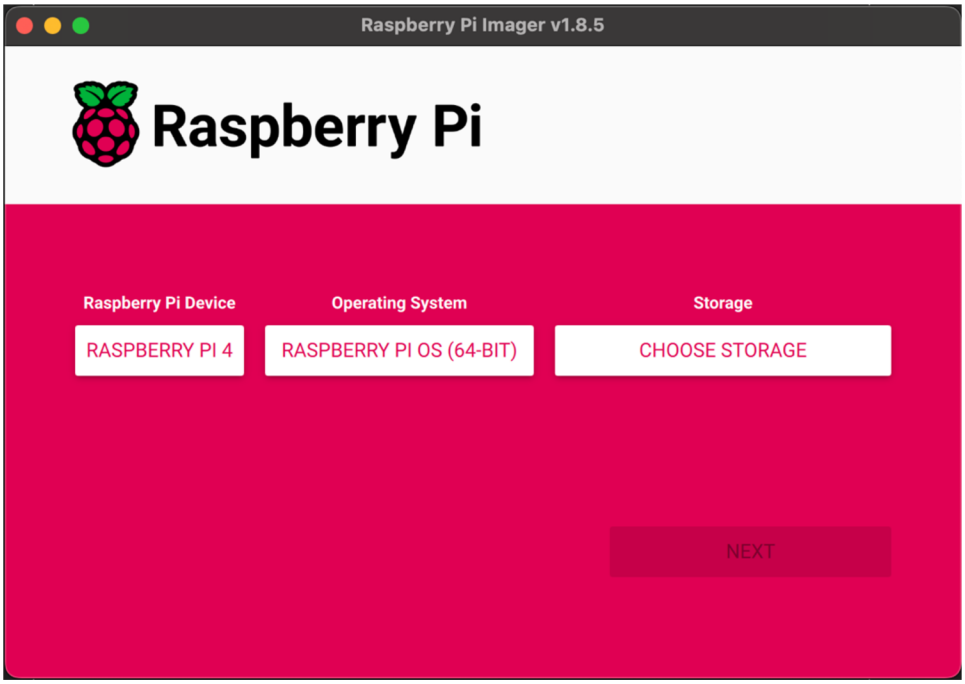
K praktické implementaci je použito Raspberry Pi (Model 4B 8GB) opatřené micro SD kartou (32 GB). Při dlouhodobém provozu je vhodné zajistit ochranu a dostatečné chlazení zařízení. Doporučuje se použití ochranného obalu, který nejen chrání desku

před mechanickým poškozením a prachem, ale také pomáhá s odvodem tepla, pokud má ventilační otvory.

Raspberry Pi generuje při intenzivním zatížení značné množství tepla, a proto je vhodné instalovat pasivní chladiče na procesor a další klíčové komponenty. Pro trvalý provoz se však doporučuje aktivní chlazení, například pomocí ventilátoru, který zajistí efektivnější odvod tepla. Aktivní chlazení výrazně snižuje teplotu zařízení, čímž minimalizuje riziko přehřívání a prodlužuje životnost Raspberry Pi. Při nasazení ve výpočetně náročných aplikacích nebo nepřetržitém provozu je tedy aktivní chlazení téměř nezbytné pro stabilní chod a dlouhou životnost zařízení.

9.8.1 Instalace Raspbianu

Nejprve je třeba vytvořit instalaci operačního systému Raspbian. Po vložení micro SD karty do čtečky a spuštění softwaru „Raspberry Pi Imager“, je nutné vybrat vyloženou micro SD kartu a požadovaný obraz systému, kromě toho je možné nastavit i heslo pro administrátorský účet „pi“. Proběhne zápis a kontrola dat. Poté je možné micro SD kartu vyjmout a vložit do zařízení Raspberry Pi.



Obrázek 12 Instalace Raspberry Pi

Zdroj: Vlastní

Po zapojení síťového adapteru se Raspberry pi spustí a přihlásí pod administrátorským účtem „pi“. Nejprve provedeme aktualizaci již existujícího softwaru. Příkaz *sudo apt*

update provede aktualizaci seznamu dostupných balíčků z repozitáře a příkaz *sudo apt upgrade* provede samotnou instalaci aktualizovaného seznamu.

9.8.2 Instalace potřebného softwaru

K fungování API serveru, který bude zpracovávat HTTP, je nezbytné nainstalovat webový server a databázový systém. V tomto případě byl zvolen serverový runtime Node.js a jako databáze byla zvolena objektově orientovaná MongoDB, která umožňuje flexibilní ukládání strukturovaných dat ve formátu JSON.

Instalace Node.js

Node.js je open-source JavaScriptový runtime, který umožňuje spouštět serverové aplikace. Jeho instalace se provádí pomocí několika příkazů v terminálu. Nejprve je potřeba spustit příkaz *sudo curl -sL https://deb.nodesource.com/setup_16.x | sudo bash -*, kterým se přidá potřebný repozitář. Poté se samotný Node.js nainstaluje příkazem *sudo apt-get -y install nodejs*. Po dokončení instalace lze ověřit, že vše proběhlo úspěšně, příkazem *node -v*, který zobrazí aktuálně nainstalovanou verzi Node.js.

Instalace MongoDB

Pro instalaci databázového systému MongoDB se používá příkaz *sudo apt-get install -y mongodb-org*. Po jeho dokončení je možné databázový server spustit pomocí příkazu *sudo systemctl start mongod* a následně zkontrolovat jeho stav příkazem *sudo systemctl status mongod.*

Správa databáze pomocí MongoDB Compass

Pro usnadnění práce s databází je vhodné využít grafické uživatelské rozhraní MongoDB Compass. Tento nástroj umožňuje snadno prohlížet, upravovat a mazat data v databázi bez nutnosti psaní příkazů v terminálu. Kromě toho poskytuje přehledné zobrazení struktury kolekcí, možnost vizualizace dat nebo tvorbu jednoduchých dotazů.

MongoDB Compass lze stáhnout zdarma z oficiálních stránek, a je dostupný pro většinu operačních systémů (Windows, macOS, Linux).

Po instalaci a spuštění MongoDB Compass stačí zadat připojovací řetězec (např. *mongodb://localhost:27017*) a po připojení lze ihned pracovat s databází.

9.9 Odesílání dat do Raspberry Pi

K výslednému odeslání dat z obou root Nodů do Raspberry Pi je využito Http requestů. Proto je nezbytné nejprve importovat knihovnu *ArduinoHttpClient.*

Samotné nastavení parametrů nezbytných pro fungování odesílání dat je definováno v globálních proměnných ESP (viz Zdrojový kód 5). V proměnných *ssid, password* jsou definovány údaje k Access Pointu, přes který bude probíhat přenos. Poté je nezbytné určit *serverName a serverPort,* kde se jedná o IP adresu/doménu a port která bude zpracovávat zaslané requesty. V tomto případě se jedná o IP adresu Raspberry Pi. Následně jsou zde vytvořeny instance třídy *WifiClient* a *HttpClient*.

```
// Údaje k wifi
const char* ssid = "BpSsid";
const char* password = "BpPass";

// Api server IP a port
const char* serverName = "192.168.1.102";
const int serverPort = 4000;

// Api server cesta
const char* serverPath = "/api/setSensorValues";

// WifiClient instance
WiFiClient wifiClient;

// HttpClient instance
HttpClient httpClient = HttpClient(wifiClient, serverName,
serverPort);
```

Zdrojový kód 5 Globální proměnné pro odesílání dat

Zdroj: Vlastní

Nastaly změny i ve funkci *setup*() kde probíhá k samotnému připojení ESP8266 k wifi Access Pointu. Je zde použit while cyklus který zaručuje připojení i přes to že se jeden nebo více pokusů nemusí povést.

```
void setup() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting...");
  }
  Serial.println("Connected to WiFi!");
}
```

Zdrojový kód 6 Připojení k access pointu

Zdroj: Vlastní

9.10 Ukládání naměřených hodnot

ESP gateway obdrží ze sítě ESP data o teplotě, vlhkosti, tlaku a světelné intenzitě. Následně tato data odesílá formou HTTP POST požadavku ve formátu JSON na API server, konkrétně na endpoint */api/setSensorValues*. Tento požadavek je zpracován na straně Node.js serveru, kde jsou data validována, strukturována a následně uložena do MongoDB. ESP tedy vystupuje jako klient, který pravidelně odesílá měření na server.

Příkladem požadavku může být:

```
{
  "node_id": 1
  "temperature": 23.7,
 "temperature2": 23.5,
  "humidity": 55.4,
  "pressure": 1011,
  "intensity": 735,
}
```

Zdrojový kód 7 Vzor přijatých dat na API serveru

Zdroj: Vlastní

Struktura Node.js aplikace

Struktura serverové aplikace je rozdělena do několika souborů, z nichž každý plní specifickou úlohu. Soubor *app.js* slouží k inicializaci Express aplikace, nastavení základního middleware (například pro zpracování JSON požadavků) a načtení směrovacích modulů (rout). Soubor *server.js* zajišťuje spuštění samotného HTTP serveru a případné připojení modulu Socket.IO pro práci s WebSockety, ty slouží v aplikaci k aktualizaci dat ve výsledném grafu. V souboru *routes/api.js* je implementována logika API endpointů, včetně nejdůležitějšího */setSensorValues*, který zpracovává příchozí data ze zařízení ESP. Model *SensorData.js*, umístěný ve složce *models*, definuje Mongoose schéma, které určuje strukturu a datové typy hodnot, jež budou ukládány do databáze. Nakonec *mongoose.js*, uložený ve složce *config*, zajišťuje navázání spojení s MongoDB databází prostřednictvím knihovny Mongoose a zpracování případných chyb při připojování.

Spuštění serveru

Pro spuštění serveru je potřeba mít nainstalovaný Node.js, MongoDB a všechny potřebné závislosti uvedené v souboru *package.json*. Nejprve je nutné nainstalovat závislosti pomocí příkazu *npm install*. Dále je třeba zajistit, aby běžela instance MongoDB. Databáze musí být spuštěná a musí existovat uživatel s přístupem. Poté lze

server spustit příkazem *node server.js*. Po úspěšném spuštění se server připojí k databázi a začne naslouchat na portu 4000.

V konzoli se zobrazí hlášky „Connected to MongoDB“ a „Server is running on http://localhost:4000“.

Příjem dat na API

V souboru *api.js* se nachází hlavní POST endpoint */api/setSensorValues*, který přijímá data z ESP. Po obdržení dat je vytvořena nová instance modelu *SensorData* a následně uložena do databáze metodou *save()*. Celý zdrojový kód API serveru je veřejně dostupný na Gitlab.com (https://gitlab.com/2risa.dan/fim-uhk-bp-nodejs-api)

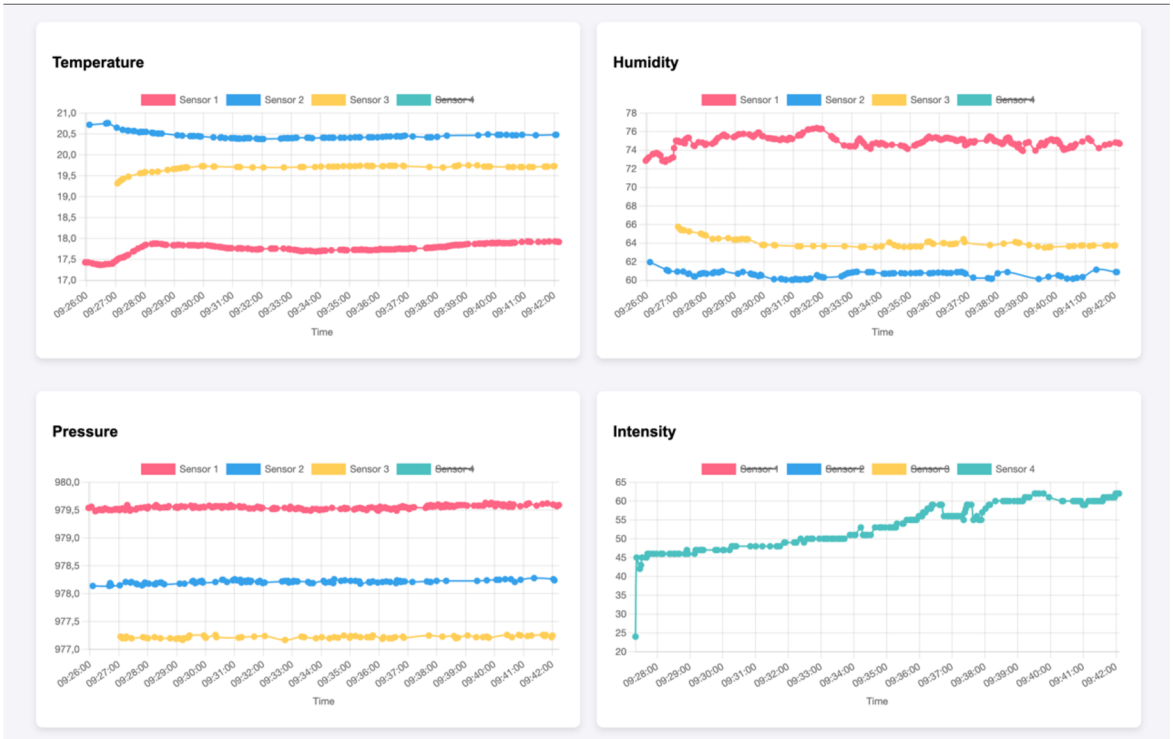
Uložení dat do databáze

Uložená data jsou spravována pomocí MongoDB [50] a knihovny **Mongoose** [51], která umožňuje definovat schéma v tomto případě *sensorSchema* Mongoose komunikuje s databází pomocí objektově orientovaného přístupu. Každý nový záznam je uložen jako dokument obsahující všechny přijaté hodnoty včetně časových razítek *createdAt* a *updatedAt*, které jsou generovány automaticky a udávají vytvoření a případnou aktualizaci záznamu.

9.11 Výstup měřených hodnot

Výsledná data byla získána ze dvou nezávislých sítí, přičemž jedna využívala komunikační protokol **ESP-NOW** a druhá **ESP-MESH**. V obou případech data pocházela ze senzorů připojených k jednotlivým nodům, přičemž hodnoty byly následně odesílány na API server běžící na Raspberry Pi. Díky jednotnému formátu dat a porovnatelné kvalitě přenosu mezi oběma technologiemi bylo možné tato data spojit do jednoho společného zobrazení.

Výsledky jsou znázorněny v grafech rozdělených dle jednotlivých veličin (viz Obrázek 13). Přístup k tomuto výstupu je realizován prostřednictvím webového rozhraní po zadání IP adresy Raspberry Pi do webového prohlížeče, v tomto případě na adrese http://192.168.0.102. Pro vykreslení grafů byla využita knihovna **Chart.js** [52].



Obrázek 13 Graf naměřených hodnot

Zdroj: Vlastní

Stránka je responzivní proto je možné ji zobrazit i na mobilních zařízeních. Graf zobrazuje měření čtyř různých senzorů v čase. Teplota a tlak jsou relativně stabilní s mírnými rozdíly mezi senzory. Stejně tak i vlhkost vykazuje rozdílné hodnoty mezi senzory. Tyto rozdíly jsou však způsobeny tím, že senzory jsou umístěny na různých místech s odlišnou vzdáleností od ESP gateway, což může ovlivňovat naměřené hodnoty.

10 Shrnutí a diskuse výsledků

Cílem této práce bylo navrhnout a implementovat bezdrátovou senzorovou síť s využitím protokolů ESP-NOW a ESP-MESH. V rámci experimentální části byl vytvořen funkční prototyp sítě složené z více uzlů na platformě ESP8266, přičemž jednotlivé uzly komunikovaly bez nutnosti Wi-Fi routeru. Testování probíhalo v simulovaném prostředí se zaměřením na stabilitu přenosu, energetickou náročnost, latenci a dosah.

Při použití **ESP-NOW** se ukázalo, že jde o velmi rychlý a energeticky efektivní způsob přenosu dat typu point-to-point nebo broadcast v rámci omezeného počtu zařízení. Výhodou je nízká latence, nevýhodou je však absence automatického směrování paketů a nutnost explicitní znalosti MAC adres všech komunikujících zařízení. Tento přístup je vhodný zejména pro jednoduché aplikace s přímou viditelností a omezeným počtem uzlů.

Naopak **ESP-MESH** umožňuje vytvořit hierarchickou síť s dynamickým směrováním zpráv. Výsledky ukázaly, že při větším počtu zařízení je tento protokol flexibilnější a umožňuje pokrýt větší plochu. Síť je schopna automaticky rekonfigurovat topologii při výpadku některých uzlů, což zvyšuje její spolehlivost. Nevýhodou je vyšší energetická náročnost, delší doba inicializace a vyšší latence oproti ESP-NOW.

Z pohledu stability přenosu a kvality signálu byly oba protokoly vyhodnoceny jako vhodné pro nasazení v IoT aplikacích. ESP-NOW se ukázal jako ideální pro časově kritické aplikace, zatímco ESP-MESH je vhodnější pro rozsáhlejší, robustní sítě. Výsledky také potvrzují, že volba konkrétního protokolu by měla záviset především na charakteru aplikace, požadované škálovatelnosti a provozních podmínkách.

11 Závěry a doporučení

Byla provedena analýza a praktická implementace bezdrátové senzorové sítě založené na komunikačních protokolech ESP-NOW a ESP-MESH. Obě technologie představují dostupná řešení pro bezdrátovou komunikaci v rámci sítí zařízení typu IoT. ESP-NOW se ukázal jako vhodný pro přímou a energeticky úspornou komunikaci mezi menším počtem zařízení, zatímco ESP-MESH umožňuje vytvoření rozsáhlejší sítě s dynamickým směrováním a vyšší odolností vůči výpadkům jednotlivých uzlů.

Z obecného hlediska jsou bezdrátové senzorové sítě s využitím těchto technologií aplikovatelné v různých oblastech, například v automatizaci budov, monitorování životního prostředí, zemědělství nebo logistice. Vzhledem k nízké ceně a otevřenosti řešení postavených na platformě ESP8266 nebo ESP32 lze považovat tyto technologie za perspektivní pro široké využití.

Doporučuje se další ověřování navržené architektury v náročnějších provozních podmínkách, například v průmyslovém prostředí. Takové nasazení by umožnilo získat detailnější informace o stabilitě, škálovatelnosti a reálné spolehlivosti sítě při dlouhodobém provozu. Současně lze zvážit rozšíření systému o pokročilé funkce, jako je šifrování komunikace nebo napojení na cloudové.

12 Seznam použité literatury

[1] SOHRABY, Kazem, Daniel MINOLI a Taieb ZNATI. *Wireless sensor networks: technology, protocols, and applications*. Hoboken, N.J: Wiley-Interscience, 2007. Wiley-IEEE Press. ISBN 978-0-471-74300-2.

[2] SCHWARTZ, Marco. *Internet of Things with ESP8266: build amazing Internet of Things projects using the ESP8266 Wi-Fi chip*. 1st ed. Place of publication not identified: Packt Publishing, 2016. Community Experience Distilled. ISBN 978-1-78646-667-9.

[3] GUBBI, Jayavardhana, Rajkumar BUYYA, Slaven MARUSIC a Marimuthu PALANISWAMI. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* [online]. 2013, **29**(7), 1645–1660. ISSN 0167739X. Dostupné z: doi:10.1016/j.future.2013.01.010

[4] AL-FUQAHA, Ala, Mohsen GUIZANI, Mehdi MOHAMMADI, Mohammed ALEDHARI a Moussa AYYASH. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials* [online]. 2015, **17**(4), 2347–2376. ISSN 1553-877X, 2373-745X. Dostupné z: doi:10.1109/COMST.2015.2444095

[5] *ESP32 prakticky - Edice CZ.NIC: knihy nejen pro IT odborníky - knihy* [online]. [vid. 2025-04-19]. Dostupné z: https://knihy.nic.cz/cs/detail/32/?utm_source=chatgpt.com

[6] ZHANG, Yan, Jijun LUO a Honglin HU, ed. *Wireless mesh networking: architectures, protocols and standards*. Boca Raton: Auerbach Publications, 2007. Wireless networks and mobile communications series. ISBN 978-0-8493-7399-2.

[7] RAULT, Tifenn, Abdelmadjid BOUABDALLAH a Yacine CHALLAL. Energy efficiency in wireless sensor networks: A top-down survey. *Computer Networks* [online]. 2014, **67**, 104–122. ISSN 13891286. Dostupné z: doi:10.1016/j.comnet.2014.03.027

[8] RAZZAQUE, Mohammad Abdur, Marija MILOJEVIC-JEVRIC, Andrei PALADE a Siobhan CLARKE. Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal* [online]. 2016, **3**(1), 70–95. ISSN 2327-4662. Dostupné z: doi:10.1109/JIOT.2015.2498900

[9] MEKKI, Kais, Eddy BAJIC, Frederic CHAXEL a Fernand MEYER. A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express* [online]. 2019, **5**(1), 1–7. ISSN 24059595. Dostupné z: doi:10.1016/j.icte.2017.12.005

[10] ZACHOS, Christos K., Jonathan LOO a Shafiullah KHAN. Wireless Mesh Network: Architecture and Protocols. In: Jonathan LOO, Jaime Lloret MAURI a Jesús Hamilton ORTIZ*Mobile Ad Hoc Networks* [online]. 1. vyd. Boca Raton: CRC Press, 2016 [vid. 2025-04-20], s. 425–447. ISBN 978-0-429-19222-7. Dostupné z: doi:10.1201/b11447-19

[11] ZHAO, Feng a Leonidas J. GUIBAS. *Wireless Sensor Networks: An Information Processing Approach*. B.m.: Morgan Kaufmann, 2004. ISBN 978-1-55860-914-3.

[12] NHAN, Nguyen Chi, Thai Viet HOANG a Nguyen Phuoc Hoang KHANG. Design and implementation of a low-power wireless sensor network using power-saving techniques. In: *2024 18th International Conference on Advanced Computing and Analytics (ACOMPA)*: *2024 18th International Conference on Advanced Computing and Analytics (ACOMPA)* [online]. Ben Cat, Vietnam: IEEE, 2024, s. 16–20 [vid. 2025-04-21]. ISBN 979-8-3315-4246-7. Dostupné z: doi:10.1109/ACOMPA64883.2024.00010

[13] esp-now/User_Guide.md at master · espressif/esp-now. *GitHub* [online]. [vid. 2025-04-21]. Dostupné z: https://github.com/espressif/esp-now/blob/master/User_Guide.md

[14] LEA, Perry. *Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security*. 1. vyd. Birmingham: Packt Publishing Limited, 2018. ISBN 978-1-78847-059-9.

[15] *Technical Documents | Espressif Systems* [online]. [vid. 2025-04-21]. Dostupné z: https://www.espressif.com/en/support/documents/technical-documents

[16] *ESP-NOW - ESP32 - — ESP-IDF Programming Guide v5.4.1 documentation* [online]. [vid. 2025-04-21]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_now.html#

[17] ERIDANI, Dania, Adian Fatchur ROCHIM a Faiz Noerdiyan CESARA. Comparative Performance Study of ESP-NOW, Wi-Fi, Bluetooth Protocols based on Range, Transmission Speed, Latency, Energy Usage and Barrier Resistance. In: *2021 International Seminar on Application for Technology of Information and Communication (iSemantic)*: *2021 International Seminar on Application for Technology of Information and Communication (iSemantic)* [online]. Semarangin, Indonesia: IEEE, 2021, s. 322–328 [vid. 2025-04-21]. ISBN 978-1-6654-2804-0. Dostupné z: doi:10.1109/iSemantic52711.2021.9573246

[18] (PDF) An efficient networking solution for extending and controlling wireless sensor networks using low-energy technologies. *ResearchGate* [online]. nedatováno [vid. 2025-04-22]. Dostupné z: doi:10.7717/peerj-cs.780

[19] SCHWAB, Klaus. *The fourth industrial revolution*. First U.S. edition. New York: Crown Business, 2016. ISBN 978-1-5247-5886-8.

[20] (PDF) Evaluation Method of Mesh Protocol over ESP32 and ESP8266. *ResearchGate* [online]. 2025 [vid. 2025-04-21]. Dostupné z: doi:10.21123/bsj.2021.18.4(Suppl.).1397

[21] *ESP-WIFI-MESH | Espressif Systems* [online]. [vid. 2025-04-21]. Dostupné z: https://www.espressif.com/en/products/sdks/esp-wifi-mesh/overview

[22] (PDF) An Efficient Wireless Sensor Network Based on the ESP-MESH Protocol for Indoor and Outdoor Air Quality Monitoring. *ResearchGate* [online]. 2025 [vid. 2025-04-21]. Dostupné z: doi:10.3390/su142416630

[23] (PDF) IoT: A Mobile Application and Multi-hop Communication in Wireless Sensor Network for Water Monitoring. *ResearchGate* [online]. 2024 [vid. 2025-04-21]. Dostupné z: doi:10.3991/ijim.v14i11.13681

[24] (PDF) Development of Power Consumption Models for ESP8266-Enabled Low-Cost IoT Monitoring Nodes. *ResearchGate* [online]. 2024 [vid. 2025-04-21]. Dostupné z: doi:10.4236/ait.2019.91001

[25] (PDF) Performance Comparison among (Star, Tree and Mesh) Topologies for Large Scale WSN based IEEE 802.15.4 Standard. *ResearchGate* [online]. nedatováno [vid. 2025-04-22]. Dostupné z: doi:10.5120/ijca2015905515

[26] (PDF) A Survey of Flooding, Gossip Routing, and Related Schemes for Wireless Multi-hop Networks. *ResearchGate* [online]. [vid. 2025-04-22]. Dostupné z: https://www.researchgate.net/publication/262492073_A_Survey_of_Flooding_Gossip_Routing_and_Related_Schemes_for_Wireless_Multi-hop_Networks

[27] (PDF) Multihop Routing In Self-Organizing Wireless Sensor Networks. *ResearchGate* [online]. nedatováno [vid. 2025-04-22]. Dostupné z: https://www.researchgate.net/publication/285697924_Multihop_Routing_In_Self-Organizing_Wireless_Sensor_Networks

[28] I2C-bus specification and user manual. 2021, **2021**.

[29] (PDF) A Review on The Inter-Processor Communication: I2C, UART, and SPI interfacing techniques. *ResearchGate* [online]. [vid. 2025-04-22]. Dostupné z: https://www.researchgate.net/publication/356556000_A_Review_on_The_Inter-Processor_Communication_I2C_UART_and_SPI_interfacing_techniques

[30] (PDF) Design & Verification of Serial Peripheral Interface (SPI) Protocol. *ResearchGate* [online]. nedatováno [vid. 2025-04-22]. Dostupné z: doi:10.35940/ijrte.F7356.038620

[31] (PDF) A Study of Communication Protocols for Internet of Things (IoT) Devices: Review. In: *ResearchGate* [online]. 2024 [vid. 2025-04-22]. Dostupné z: doi:10.2991/ahis.k.210913.033

[32] (PDF) Modeling and Analysis of the 1-Wire Communication Protocol Using Timed Colored Petri Nets. *ResearchGate* [online]. 2024 [vid. 2025-04-22]. Dostupné z: doi:10.1109/ACCESS.2018.2833213

[33] WOLF, Marilyn. Chapter 6 - Analog/Digital and Digital/Analog Conversion. In: Marilyn WOLF, ed. *Embedded System Interfacing* [online]. B.m.: Morgan Kaufmann, 2019 [vid. 2025-04-22], s. 129–139. ISBN 978-0-12-817402-9. Dostupné z: doi:10.1016/B978-0-12-817402-9.00006-6

[34] DAHIYA, Abhishek Singh, Yogeenth KUMARESAN, Oliver OZIOKO, Markellos NTAGIOS a Ravinder DAHIYA. Soft Sensors for Electronic Skin. In: Roger NARAYAN, ed. *Encyclopedia of Sensors and Biosensors (First Edition)* [online]. Oxford: Elsevier, 2023 [vid. 2025-04-22], s. 51–67. ISBN 978-0-12-822549-3. Dostupné z: doi:10.1016/B978-0-12-822548-6.00069-8

[35] DEMPSTER, JOHN. CHAPTER ONE - Introduction. In: JOHN DEMPSTER, ed. *The Laboratory Computer* [online]. London: Academic Press, 2001 [vid. 2025-04-22], Biological Techniques Series, s. 1–11. Dostupné z: doi:10.1016/B978-012209551-1/50034-9

[36] *Noise Immunity - an overview | ScienceDirect Topics* [online]. [vid. 2025-04-22]. Dostupné z: https://www.sciencedirect.com/topics/engineering/noise-immunity?utm_source=chatgpt.com

[37] RAJAK, Prem, Abhratanu GANGULY, Satadal ADHIKARY a Suchandra BHATTACHARYA. Internet of Things and smart sensors in agriculture: Scopes and challenges. *Journal of Agriculture and Food Research* [online]. 2023, **14**, 100776. ISSN 2666-1543. Dostupné z: doi:10.1016/j.jafr.2023.100776

[38] DS18B20 - Programmable Resolution 1-Wire Digital Thermometer. nedatováno.

[39] Humidity Sensor BME280. *Bosch Sensortec* [online]. [vid. 2025-04-22]. Dostupné z: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/

[40] *BH1750 16-bit Ambient Light Sensor* [online]. 18. listopad 2020 [vid. 2025-04-22]. Dostupné z: https://cz.mouser.com/adafruit-bh1750-ambient-light-sensor

[41] *Welcome to Fritzing* [online]. [vid. 2025-04-20]. Dostupné z: https://fritzing.org/

[42] *Software* [online]. [vid. 2025-04-20]. Dostupné z: https://www.arduino.cc/en/software/

[43] Arduino/tools/sdk/include/espnow.h at master · esp8266/Arduino. *GitHub* [online]. [vid. 2025-04-22]. Dostupné z: https://github.com/esp8266/Arduino/blob/master/tools/sdk/include/espnow.h

[44] painlessMesh/README.md at master · gmag11/painlessMesh. *GitHub* [online]. [vid. 2025-04-22]. Dostupné z: https://github.com/gmag11/painlessMesh/blob/master/README.md

[45] *adafruit/Adafruit_BME280_Library* [online]. C++. B.m.: Adafruit Industries. 22. duben 2025 [vid. 2025-04-22]. Dostupné z: https://github.com/adafruit/Adafruit_BME280_Library

[46] Arduino/libraries/Wire/Wire.h at master · esp8266/Arduino. *GitHub* [online]. [vid. 2025-04-22]. Dostupné z: https://github.com/esp8266/Arduino/blob/master/libraries/Wire/Wire.h

[47] *BH1750 | Arduino Documentation* [online]. [vid. 2025-04-22]. Dostupné z: https://docs.arduino.cc/libraries/bh1750/

[48] OneWire/OneWire.h at master · PaulStoffregen/OneWire. *GitHub* [online]. [vid. 2025-04-19]. Dostupné z: https://github.com/PaulStoffregen/OneWire/blob/master/OneWire.h

[49] BURTON, Miles. *milesburton/Arduino-Temperature-Control-Library* [online]. C++. 17. duben 2025 [vid. 2025-04-19]. Dostupné z: https://github.com/milesburton/Arduino-Temperature-Control-Library

[50] MongoDB: The World's Leading Modern Database. *MongoDB* [online]. [vid. 2025-04-22]. Dostupné z: https://www.mongodb.com/

[51] *Mongoose v8.13.2: Getting Started* [online]. [vid. 2025-04-22]. Dostupné z: https://mongoosejs.com/docs/

[52] *chartjs/Chart.js* [online]. JavaScript. B.m.: Chart.js. 19. duben 2025 [vid. 2025-04-19]. Dostupné z: https://github.com/chartjs/Chart.js

13 Seznam použítých zkratek

IoT – Internet of Things (internet věcí)
ESP – Espressif Systems Platform
WSN – Wireless Sensor Network (Bezdrátová senzorová síť)
RFID – Radio Frequency Identification
BLE – Bluetooth Low Energy
IPv6 – Internet Protocol version 6
6LoWPAN – IPv6 over Low power Wireless Personal Area Networks
LoRaWAN – Long Range Wide Area Network
LPWAN – Low-Power Wide-Area Network
AI – Artificial Intelligence
ESP-IDF – Espressif IoT Development Framework
DIY – Do It Yourself
TCP/IP – Transmission Control Protocol / Internet Protocol
GPIO – General Purpose Input/Output
USB – Universal Serial Bus
UART – Universal Asynchronous Receiver-Transmitter
ISO/OSI – Open Systems Interconnection model (podle ISO normy)
TTL – Time To Live
I2C – Inter-Integrated Circuit
EEPROM – Electrically Erasable Programmable Read-Only Memory
SDA – Serial Data Line
SCL – Serial Clock Line
SPI – Serial Peripheral Interface
MOSI – Master Out Slave In
MISO – Master In Slave Out
SCK – Serial Clock
SS – Slave Select
SD – Secure Digital
ADC – Analog-to-Digital Converter

14 Přílohy

14.1 Příloha č. 1: Zdrojový kód ESP-NOW Slave

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

// Knihovny pro BME 280
#include <Adafruit_BME280.h>

// Knihovny pro BH1750
#include <Wire.h>
#include <BH1750.h>

// Knihovny pro DS18B20
#include <OneWire.h>
#include <DallasTemperature.h>
const int ds_pin = 4;


// Instance OneWire
OneWire oneWireDS(ds_pin);

// Instance DS
DallasTemperature dsSensor(&oneWireDS);

// Adresa BME
#define BME280_ADDRESS (0x76)

// BME Instance
Adafruit_BME280 bme;

// BH Instance
BH1750 bh;

// Mac Adresa příjemce
uint8_t broadcastAddress[] = {0xAC, 0x0B, 0xFB, 0xC2, 0x95, 0x9D};

// Definice odeslané zprávy
typedef struct send_message {
    short id;
    float temperature;
    float temperature2;
    float humidity;
    float pressure;
    float intensity;
} send_message;
```

```
// Proměnná typu send_message
send_message sendMessage;

// Timer odesílání dat
unsigned long lastTime = 0;
unsigned long timerDelay = 2000;

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0){
    Serial.println("Delivery success");
  }
  else{
    Serial.println("Delivery fail");
  }
}

void setup() {
  // Inicializace serioveho monitoru
  Serial.begin(115200);

  // Nastavení ESP jako "Wifi station"
  WiFi.mode(WIFI_STA);

  // Inicializace ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Inicializace BME
  bme.begin(BME280_ADDRESS);

  // Inicializace BH
  bh.begin();

  // Inicializace DS18B20
  dsSensor.begin();

  // Registace ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);
  esp_now_register_send_cb(OnDataSent);

  // Registerace master peera
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);
}
```

```
void loop() {
  if ((millis() - lastTime) > timerDelay) {
    // Nastavení hodnot objektu
    sendMessage.id = 4;
    sendMessage.temperature = bme.readTemperature();
    // Získání hodnot DS18B20 ze sběrnice
    dsSensor.requestTemperatures();
    sendMessage.temperature2 = dsSensor.getTempCByIndex(0);
    sendMessage.humidity = bme.readHumidity();
    sendMessage.pressure = bme.readPressure() / 100.0F;
    sendMessage.intensity = bh.readLightLevel();
    Serial.println(sendMessage.intensity);
    // odeslání zprávy přes ESP-NOW
    esp_now_send(broadcastAddress, (uint8_t *) &sendMessage, sizeof(sendMessage));
    lastTime = millis();
  }
```

14.2 Příloha č. 2: Zdrojový kód ESP-NOW Master

```
#include <ArduinoHttpClient.h>
#include <ESP8266WiFi.h>
#include <espnow.h>

// Definice přijaté zprávy
typedef struct received_message {
    short id;
    float temperature;
    float temperature2;
    float humidity;
    float pressure;
    float intensity;
} received_message;

// Definice odeslané zprávy
typedef struct send_message {
    int interval;
} send_message;

// Proměnná typu received_message
received_message receivedMessage;

// Proměnná typu send_message
send_message sendMessage;

// Boolean která definuje zda odeslat data
bool sendToApi = 0;

// Údaje k wifi
const char* ssid = "BpSsid";
const char* password = "BpPass";

// Api server IP a port
const char* serverName = "192.168.1.102";
const int serverPort = 4000;

// Api server cesta
const char* serverPath = "/api/setSensorValues";

// WifiClient instance
WiFiClient wifiClient;

// HttpClient instance
HttpClient httpClient = HttpClient(wifiClient, serverName, serverPort);

// Callback pro přijetí dat
```

```
void DataReceivedCallback(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
  memcpy(&receivedMessage, incomingData, sizeof(receivedMessage));
  Serial.print("received from node: ");
  Serial.println(receivedMessage.id);
  Serial.println();
  sendToApi = 1;
}

void DataSendCallback(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0){
    Serial.println("Delivery success");
  }
  else{
    Serial.println("Delivery fail");
  }
}

void setup() {
  // Načtení sériového monitoru
  Serial.begin(115200);

  // Připojení k wifi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting...");
  }
  Serial.println("Connected to WiFi!");

  // Nastaveni ESP jako Wi-Fi Stanice
  WiFi.mode(WIFI_STA);

  // Načtení ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("ESP-NOW failed to load!");
    return;
  }

  // Nastaveni callback funkcí
  esp_now_register_send_cb(DataSendCallback);
  esp_now_register_recv_cb((DataReceivedCallback));
}

void loop() {
    // Ověření zda je třeba odeslat data
    if(sendToApi){
      // Odeslání dat
      SendToApi();
```

```
    }
}

void SendToApi(){
  // Vytvoření jsonu
  String jsonData = "{";
  jsonData += "\"node_id\":" + String(receivedMessage.id);
  jsonData += ",\"mode\":" + String(1);
  jsonData += ",\"temperature\":" + String(receivedMessage.temperature);
  jsonData += ",\"temperature2\":" + String(receivedMessage.temperature);
  jsonData += ",\"humidity\":" + String(receivedMessage.humidity);
  jsonData += ",\"pressure\":" + String(receivedMessage.pressure);
  jsonData += ",\"intensity\":" + String(receivedMessage.intensity);
  jsonData += "}";

  // Odeslání Jsoonu na http server (NodeJs)
  httpClient.beginRequest();
  httpClient.post(serverPath);
  httpClient.sendHeader("Content-Type", "application/json");
  httpClient.sendHeader("Content-Length", jsonData.length());
  httpClient.beginBody();
  httpClient.print(jsonData);
  httpClient.endRequest();

  // Vypsání odpovědi
  int status_code = httpClient.responseStatusCode();
  String response = httpClient.responseBody();
  Serial.print("HTTP Response Code: ");
  Serial.println(status_code);
  Serial.print("Response Body: ");
  Serial.println(response);
  sendToApi = 0;
}
```

14.3 Příloha č. 3: Zdrojový kód ESP-MESH Node

```
#include "painlessMesh.h"

// ESP-MESH Zabezpečení
#define   M_LOGIN     "DmBp2025"
#define   M_PASS      "DmBp2025SuperPass"
#define   M_PORT      5555

// Knihovny pro BME 280
#include <Adafruit_BME280.h>

// Knihovny pro DS18B20
#include <OneWire.h>
#include <DallasTemperature.h>

// Knihovny pro BH1750
#include <Wire.h>
#include <BH1750.h>

// Instance OneWire
OneWire oneWireDS(ds_pin);

// Instance DS
DallasTemperature dsSensor(&oneWireDS);

// Adresa BME
#define BME280_ADDRESS (0x76)

// BME Instance
Adafruit_BME280 bme;

// BH Instance
BH1750 bh;

// Definice mesh třídy
painlessMesh  mesh;

// Scheduler pro ESP-MESH
Scheduler scheduler;

// Callback pro odeslání zprávy
void sendMessage() {
  String jsonData = "{";
  jsonData += "\"node_id\": 3";
  jsonData += ",\"mode\":" + String(1);
  jsonData += ",\"temperature\":" + String(bme.readTemperature());
  dsSensor.requestTemperatures();
```

```
  jsonData += ",\"temperature2\":" + String(dsSensor.getTempCByIndex(0));
  jsonData += ",\"humidity\":" + String(bme.readHumidity());
  jsonData += ",\"pressure\":" + String(bme.readPressure() / 100.0F);
  jsonData += ",\"intensity\":" + String(bh.readLightLevel());
  jsonData += ",\"mesh_id\": " + String(mesh.getNodeId());
  jsonData += "}";
  mesh.sendBroadcast(jsonData);
}

// Uloha pro odesílání zprávy
Task taskSendMessage( TASK_SECOND * 5 , TASK_FOREVER, &sendMessage );

// Callback po přijetí zprávy
void receivedCallback( uint32_t from, String &msg ) {
  // Vypsání příchozí zprávy
  Serial.println(msg);
}
void setup() {
  // Inicializace seriové komunikace
  Serial.begin(115200);

  // Inicializace BME
  bme.begin(BME280_ADDRESS);

  // Inicializace BH
  bh.begin();

  // Inicializace DS18B20
  dsSensor.begin();

  // Zapnutí výpisu o komunikaci do sériového portu
  mesh.setDebugMsgTypes(COMMUNICATION);

  // Inicializcaze ESP-MESH
  mesh.init( M_LOGIN, M_PASS, &scheduler, M_PORT );

  // Callback pro přijetí zprávy
  mesh.onReceive(&receivedCallback);

  // Přidání tasku na odeslání zprávy do scheduleru
  scheduler.addTask(taskSendMessage);

  // Zapnutí tasku pro odeslání zprávy
  taskSendMessage.enable();
}
void loop() {
  // Aktualizave ESP-MESH
  mesh.update();
}
```

14.4 Příloha č. 4: Zdrojový kód ESP-MESH Root node

```
#include <ArduinoHttpClient.h>
#include "painlessMesh.h"

// ESP-MESH Zabezpečení
#define   M_LOGIN     "DmBp2025"
#define   M_PASS      "DmBp2025SuperPass"
#define   M_PORT      5555
#define   STATION_SSID     "BpSsid"
#define   STATION_PASSWORD "BpPass"

// Definice mesh třídy
painlessMesh  mesh;

// Údaje k wifi
const char* ssid = "BpSsid";
const char* password = "BpPass";

// Api server IP a port
const char* serverName = "192.168.1.102";
const int serverPort = 4000;

// Api server cesta
const char* serverPath = "/api/setSensorValues";

// WifiClient instance
WiFiClient wifiClient;

// HttpClient instance
HttpClient httpClient = HttpClient(wifiClient, serverName, serverPort);

// Scheduler pro ESP-MESH
Scheduler scheduler;

// Callback pro přijetí zprávy
void receivedCallback( uint32_t received_id, String &received_message ) {
  Serial.println(received_message);
  SendToApi(received_message);
}

void setup() {
  // Inicializace seriové komunikace
  Serial.begin(115200);

  // Inicializace Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(1000);
    Serial.println("Connecting...");
  }
  Serial.println("Connected to WiFi!");

  // Zapnutí výpisu o komunikaci do sériového portu
  mesh.setDebugMsgTypes(COMMUNICATION);

  // Inicializcaze ESP-MESH
  mesh.init( M_LOGIN, M_PASS, &scheduler, M_PORT, WIFI_AP_STA, 6 );

  // Callback pro přijetí zprávy
  mesh.onReceive(&receivedCallback);
  mesh.stationManual(STATION_SSID, STATION_PASSWORD);
}

void loop() {
  // Aktualizave ESP-MESH
  mesh.update();
}

void SendToApi(String jsonData){
  Serial.println("SENT TO API");

  Serial.println(x);
  Serial.println(jsonData);
  // Odeslání Jsonu na http server (NodeJs)
  httpClient.beginRequest();
  httpClient.post(serverPath);
  httpClient.sendHeader("Content-Type", "application/json");
  httpClient.sendHeader("Content-Length", jsonData.length());
  httpClient.beginBody();
  httpClient.print(jsonData);
  httpClient.endRequest();

  // Vypsání odpovědi
  int status_code = httpClient.responseStatusCode();
  String response = httpClient.responseBody();
  Serial.print("HTTP Response Code: ");
  Serial.println(status_code);
  Serial.print("Response Body: ");
  Serial.println(response);
}
```

15 Zadání práce z IS (eVŠKP)

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Daniel Mikš**
Studium:	I2200701
Studijní program:	B0613A140033 Aplikovaná informatika
Studijní obor:	Softwarové inženýrství
Název bakalářské práce:	**Bezdrátová senzorová síť s využitím protokolů ESP-NOW a ESP-MESH**
Název bakalářské práce AJ:	Wireless sensor network using ESP-NOW and ESP-MESH protocols

Cíl, metody, literatura, předpoklady:

Cílem bakalářské práce je popsat, navrhnout a prakticky otestovat bezdrátovou senzorovou síť na protokolech ESP-NOW a ESP-MESH. Součástí práce bude také porovnání těchto dvou protokolů.

1. Úvod do Internetu věcí

2. Systém Internetu věcí

3. Komunikační topologie

4. Bezdrátová senzorová síť (WSN)

5. Protokoly ESP-NOW a ESP-MESH

6. Popis použitého hardwaru a softwaru

7. Implementace navrženého řešení

SCHWARTZ, Marco. *Internet of Things with ESP8266: Build amazing Internet of Things projects using the ESP8266 Wi-Fi chip*. 1. 2016, 226 s. ISBN 9781786468024

CAMERON, Neil. *ESP32 Formats and Communication: Application of Communication Protocols with ESP32 Microcontroller*. 1. Apres, 2023, 754 s. ISBN 9781484293782.

PRADER, Lizzie a Peter HODDIE. *IoT Development for ESP32 and ESP8266 with JavaScript*. 1. Apress, 2020, 593 s. ISBN 1484250699.

Zadávající pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Patrik Urbaník

Datum zadání závěrečné práce: 15.10.2021