TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Internet věcí

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Zbyněk Novák**
Vedoucí práce: Ing. Tomáš Martinec, Ph.D.

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Internet of Things

Bachelor thesis

Study programme: B2646 – Information technology

Study branch: 1802R007 – Information technology

Author: **Zbyněk Novák**

Supervisor: Ing. Tomáš Martinec, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Zbyněk Novák**

Osobní číslo: **M14000060**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Internet věcí**

Zadávající katedra: **Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s počítačem Raspberry Pi a možnostmi použití různých OS a vývojových prostředí a dále s možnostmi použití tohoto počítače v rámci aplikací principů internetu věcí.
2. Seznamte se s možnostmi a limity komunikace v Mesh sítích a nalezněte nebo navrhěte protokol pro vytvoření Mesh sítě pomocí technologie Wifi.
3. Vytvořte reálnou aplikaci internetu věcí, která bude komunikovat s centrálním databázovým systémem a zároveň budou jednotlivé body spolu komunikovat pomocí principů Mesh sítě.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**30–40 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Eben Upton, Gareth Halfacree: Raspberry Pi Uživatelská příručka, Computer Press, 2013**

[2] **Yan Zhang, Jijun Luo, Honglin Hu: Wireless Mesh Networking: Architectures, Protocols and Standards (Wireless Networks and Mobile Communications). Auerbach Publications; 1 edition (December 13, 2006)**

Vedoucí bakalářské práce:	**Ing. Tomáš Martinec, Ph.D.**
	Ústav mechatroniky a technické informatiky
Konzultant bakalářské práce:	**Ing. David Krčmařík, Ph.D.**
Datum zadání bakalářské práce:	**10. října 2016**
Termín odevzdání bakalářské práce:	**15. května 2017**

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 9.5.2017

Podpis: 

Poděkování

Rád bych poděkoval Ing. Tomáši Martincovi, Ph.D., za jeho cenné rady a trpělivost při vedení mé bakalářské práce. Rovněž bych chtěl poděkovat Ing. Davidu Krčmaříkovi, Ph.D., za vstřícnost a pomoc při získávání potřebných informací a podkladů.

Abstrakt

Cílem této bakalářské práce je seznámení se s počítačem Raspberry Pi a prozkoumání možností jeho využití v rámci aplikací Internetu věcí. Poté seznámení se s možnostmi a limity fungování Mesh sítí a nalezení či navržení protokolu pro vytvoření aplikace Mesh sítě. Na základě těchto poznatků bude vytvořena reálná aplikace Mesh sítě, která bude komunikovat s centrálním databázovým systémem, ze kterého bude ovládána. Pomocí desky NodeMcu V3 byla vytvořena funkční WiFi Mesh síť. Část připojení Mesh sítě k MQTT serveru řeší řídící aplikace, která přeposílá do Mesh sítě nastavení z MQTT serveru. Nakonec byla vytvořena klientská aplikace, která odesílá požadované nastavení na MQTT server. Cíle práce byly splněny. Bylo vytvořeno funkční řešení Mesh sítě ovládané přes MQTT server pomocí klientské aplikace. Pro realizaci byl použit hardware s nízkou pořizovací cenou, výsledné řešení je tudíž ekonomicky výhodné. Přínosem této bakalářské práce je zhodnocení aktuálního stavu Mesh sítí v oblasti Internetu věcí a možností jejich ovládání.

Klíčová slova:

ESP8266, MQTT, NoceMcu, Raspberry Pi, WiFi Mesh síť

Abstract

The aim of this work is familiarization with the Raspberry Pi computer and the research of the options of its use within the Internet of Things. Further aims are familiarization with the options and limits of a Mesh network and the principles of its operation as well as the finding or projection of protocol which creates the Mesh network's applications. Based on these findings, a real application will be created. It will communicate with a central database and will be operated by it. A functional WiFi Mesh network was created with the use of a NodeMcu V3 board. The Mesh network's connection to the MQTT server is secured by a client application that forwards settings from MQTT server to the Mesh network. At the end of this work, a client application, which sends off the requested settings to MQTT server, was created. The aims of this work were fulfilled. A functional Mesh network operated by client application through central database was created. For realization in this work, an inexpensive hardware was used. Therefore, the final solution is economical. The contribution of this work can be seen in evaluation of the actual Mesh networks' situation in the field of Internet of Things and of their controlling options.

Key words:

ESP8266, MQTT, NoceMcu, Raspberry Pi, WiFi Mesh network

Obsah

Seznam obrázků

Seznam zkratek

IoT	Internet of Things
AP	Access point
MQTT	Message Queuing Telemetry Transport
M2M	Machine to machine
LED	Light-Emitting Diode
WPF	Windows Presentation Foundation
UWP	Universal Windows Platform

1 Úvod

Ve své bakalářské práci se zabývám tématem Internet věcí, a to konkrétně vytvořením WiFi Mesh sítě, která bude komunikovat s centrálním databázovým systémem, ze kterého bude ovládaná.

Toto téma jsem si zvolil na základě svého bakalářského projektu. V rámci tohoto projektu jsem se zabýval vývojem aplikací pro Raspberry Pi, díky čemuž jsem měl možnost se s tímto počítačem podrobněji seznámit a v návaznosti se obeznámit také s tématem Internetu věcí (IoT). Téma Internetu věcí mne velice zaujalo a rozhodl jsem se v prohlubování svých znalostí v tomto odvětví pokračovat.

V teoretické části představím základní pojmy. Budu se zabývat Internetem věcí, dále WiFi Mesh sítěmi, jejich limity a omezeními při použití v reálném provozu, komunikací mezi jednotlivými uzly sítě a možnostmi využití. Popíši funkčnost centrálního databázového systému, jaký typ databáze jsem zvolil a proč jsem jej zvolil. Blíže specifikuji vybraný hardware, jeho vlastnosti a možnosti použití. Opět zdůvodním také proč jsem daný hardware zvolil.

V praktické části nejprve vytvořím jednoduchou aplikaci (blikání diody na ESP) pro seznámení se s WiFi čipem ESP8266, konkrétně

s deskou NodeMcu V3 a vývojovým prostředím Arduino IDE. Následně nainstaluji MQTT server na počítač Raspberry Pi a implementuji ESP jako klienta MQTT serveru. Poté otestuji možné knihovny pro implementaci Mesh sítě s ESP, pokusím se implementovat ESP MQTT klienta a propojit Mesh sítě. Výsledná Mesh síť bude mít přístup na MQTT server, který bude nainstalován na počítači Raspberry Pi, a bude mít přístup k internetu. Na závěr pro demonstraci použití vytvořím jednoduchou aplikaci v jazyku `C#` s technologií WPF, která bude zapínat a vypínat LED diodu na ESP v Mesh síti. Aplikace bude posílat požadované nastavení LED na MQTT server, odkud si Mesh síť bude toto nastavení stahovat.

S využitím Mesh sítě a IoT by poté bylo možné vytvořit například chytrou domácnost. V jejím rámci by uživatel mohl ovládat osvětlení, teplotu vytápění nebo garážová vrata bez nutnosti rozšiřování domácí WiFi sítě tak, aby všechny jednotlivé stanice byly v jejím dosahu. Jednoduše by připojil nové zařízení do sítě v dosahu alespoň jedné další stanice a zařízení by bylo připraveno k použití.

2 Teoretická část

2.1 IoT

Pojem „Internet věcí“ je jen souhrnným zastřešujícím označením. Jedná se o kontrolu a komunikaci předmětů, jako jsou různá čidla, ovládací prvky, mobilní zařízení a další, mezi sebou nebo s člověkem prostřednictvím bezdrátových technologií a internetu. V dnešní době se v praxi využívá již nespočet zařízení, a to např. jako dálkově ovládané zásuvky a osvětlení, kamery, meteostanice a jiné. Prozatím však nespolupracují pod jednou technologií ani jedním společným protokolem [1].

IoT a zabezpečení

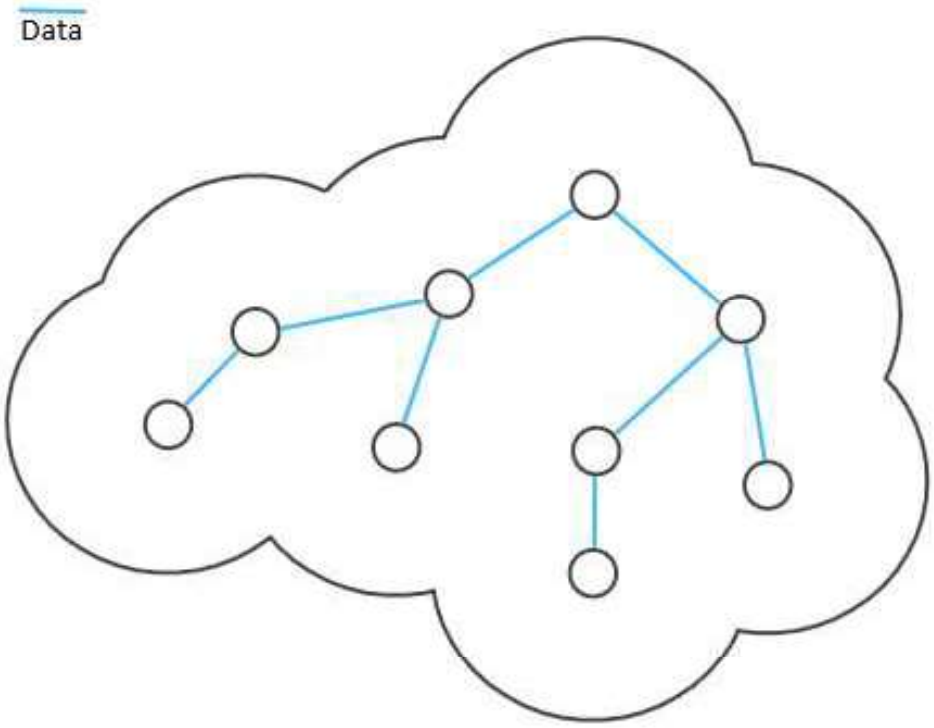
Nedávná studie společnosti HP [2] uvádí, že 70 % nejběžněji užívaných zařízení v rámci IoT je napadnutelných. Tato zranitelnost se týká hesel, šifrování nebo nedostatečné granularity přístupových práv. Zástupci HP použili svůj nástroj HP Fortify on Demand k proskenování deseti nejpopulárnějších zařízení IoT a podle svých slov objevili v průměru 25 „zranitelností“ na jedno zařízení. Týkaly se přitom zařízení známých výrobců, konkrétně televizorů,

webových kamer, domácích termostatů, dálkově ovládaných elektrických zásuvek, kontrolních jednotek sprinklerů nebo třeba elektronických zámků dveří.

Většina zařízení podle zástupců HP vyžadovala od uživatelů irelevantní osobní informace nebo nevyžadovala dostatečně bezpečné heslo. 70 % zařízení nešifrovalo přenášená data, přičemž stejný problém měla i polovina k nim příslušících mobilních aplikací. U 60 % zařízení byly objeveny nedostatečně zabezpečené ovládací webové stránky, stejný počet nešifroval softwarové updaty.

2.2 Mesh síť

WiFi Mesh síť je bezdrátová síť, která dodržuje topologii Mesh sítě. Jedná se o síť skládající se z více uzlů, kde všechny uzly sítě jsou si rovny (ad hoc). Každý uzel sítě funguje jako přístupový bod (AP - access point) a zároveň jako klient (client). Mezi těmito uzly sítě musí být zajištěno spolehlivé směrování a automatická konfigurace struktury sítě. Další nutností je také automatické začlenění nově přidaného uzlu do již stávající sítě uzlů. Jednotlivé uzly tak nevyžadují předem vytvořenou infrastrukturu, aby spolu mohli začít komunikovat, a samy si zajišťují nezbytné funkce pro řízení sítě.



Obrázek 2.1: Smíšená topologie sítě - Mesh (zdroj: autor)

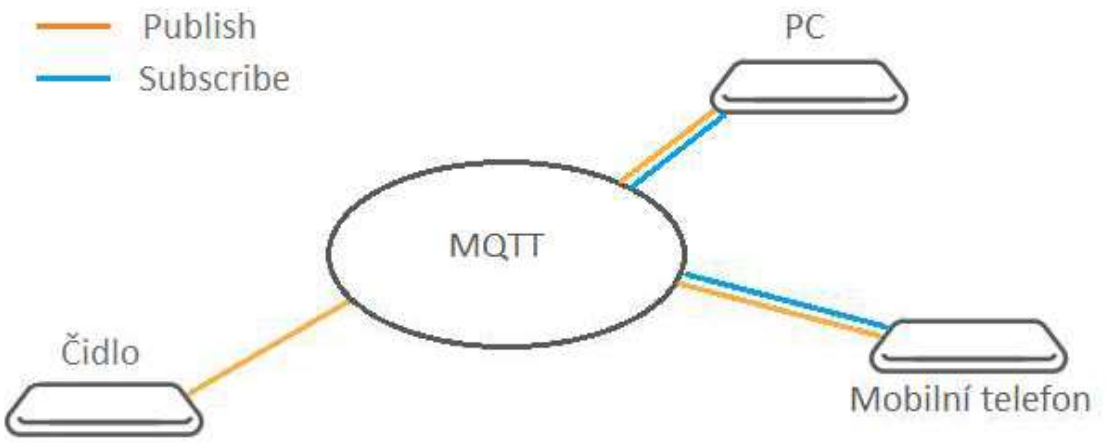
2.3 MQTT server

MQTT patří do skupiny komunikací, jež nesou souhrnné označení M2M (Machine to machine). Způsob, jakým MQTT pracuje, a jeho vlastnosti plně vyhovují jak pro účely této práce, tak pro obecné využití v oblasti IoT.

Jedná se o komunikační protokol postavený nad TCP/IP, který umožňuje mezi jednotlivými subjekty posílat krátké zprávy. MQTT komunikace se skládá z centra (broker) a klientů. Klientem může být čidlo, display, aplikace nebo webová služba. Klienti mohou buď posílat do brokeru data/zprávy (publish), nebo data/zprávy odebírat (subscribe). Každá zpráva je publikována pod názvem tématu (topic). Toto téma po jeho publikaci broker rozešle všem klientům, kteří si zaregistrovali jeho odebírání (obr. 2.2). Protokol sám nespecifikuje, jakého typu mají data být, z toho důvodu lze posílat libovolná data, jako např. textové zprávy, binární data nebo obrázky. Velikost posílaných dat je poté omezena samotným brokerem [3].

2.4 Hardware

Původním záměrem bylo použít jako klienty Mesh sítě počítače Raspberry Pi 3, jež mají vestavěný WiFi čip. Avšak po prozkoumání možných alternativ byl pro realizaci WiFi Mesh sítě použit čip ESP8266, konkrétně tímto čipem osazená neoficiální varianta desky



Obrázek 2.2: Funkcionální diagram MQTT (zdroj: autor)

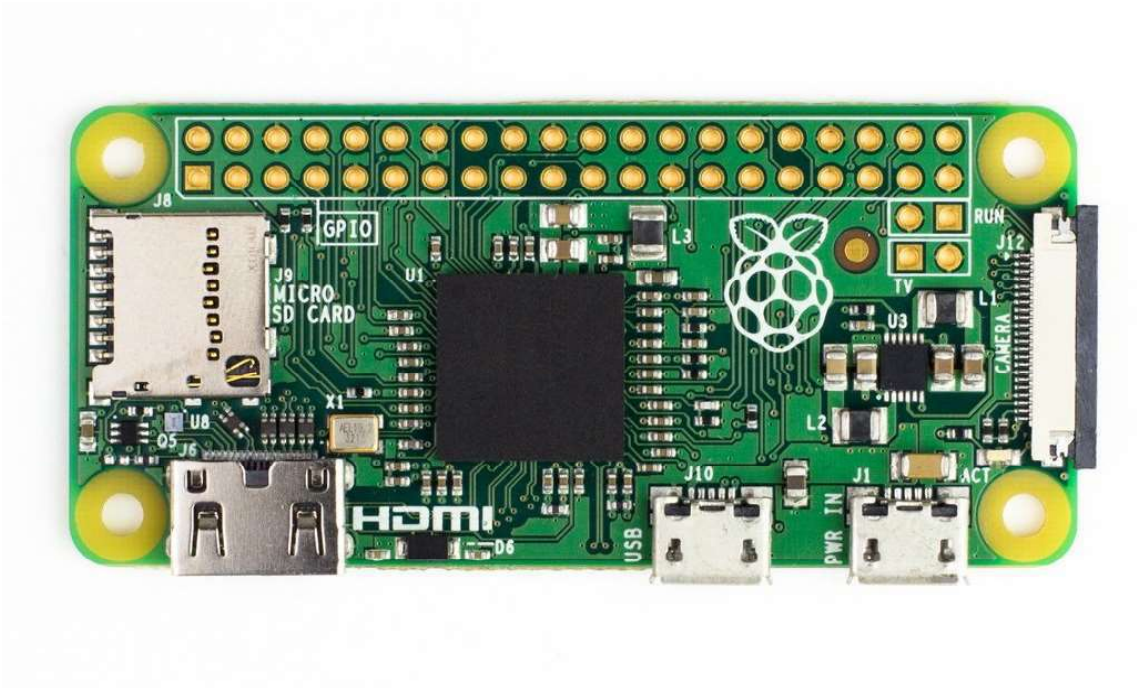
NodeMcu V3 od společnosti LoLin (obr. 2.4).

2.4.1 Raspberry Pi

Jedná se o malý jednodeskový počítač zhruba o velikosti platební karty, osazený ARM procesorem. Na Raspberry Pi je možné provozovat jak různé distribuce Linuxu, jako například Raspbian, tak i distribuce Windows, konkrétně Windows 10 IoT Core.

Operační systémy a vývojová prostředí

Raspbian jako operační systém pro Raspberry Pi nabízí plno možností využití, které bychom očekávali od plnohodnotného operačního systému. Na tomto systému lze vyvíjet aplikace v programovacím jazyku Python, Java, C/C++ a další. Při použití vývojového prostředí monodevelop lze vyvíjet i .NET aplikace. Většina těchto programovacích jazyků a příslušných vývojových prostředí je součástí základní instalace.



Obrázek 2.3: Raspberry Pi Zero (zdroj: https://shop.pimoroni.com)

Oproti tomu Windows 10 IoT Core je systém, který uživatelské prostředí nemá. Nelze tuto absenci ovšem pokládat za nevýhodu, pro některé typy projektů může byt naopak výhodou. Na Windows 10 IoT je možné spouštět pouze UWP aplikace vyvíjené v prostředí Visual Studio 2015 a vyšší.

Raspberry Pi tedy může sloužit k ovládání různých zařízení, ale také k vývoji příslušných aplikací. Vzhledem k širokým možnostem, které systém Raspbian nabízí, je spolu s multiplatformním programovacím jazykem Python vhodnou volbou pro účely této práce.

Typy Raspberry Pi

Počítač Raspberry Pi je dostupný v několika variantách a generacích [4]. Původním záměrem bylo pro potřeby této práce použít Raspberry Pi 3, který oproti předchozím generacím nabízí vestavěný WiFi čip, a tudíž není potřeba pro připojení k WiFi dokupovat WiFi modul.

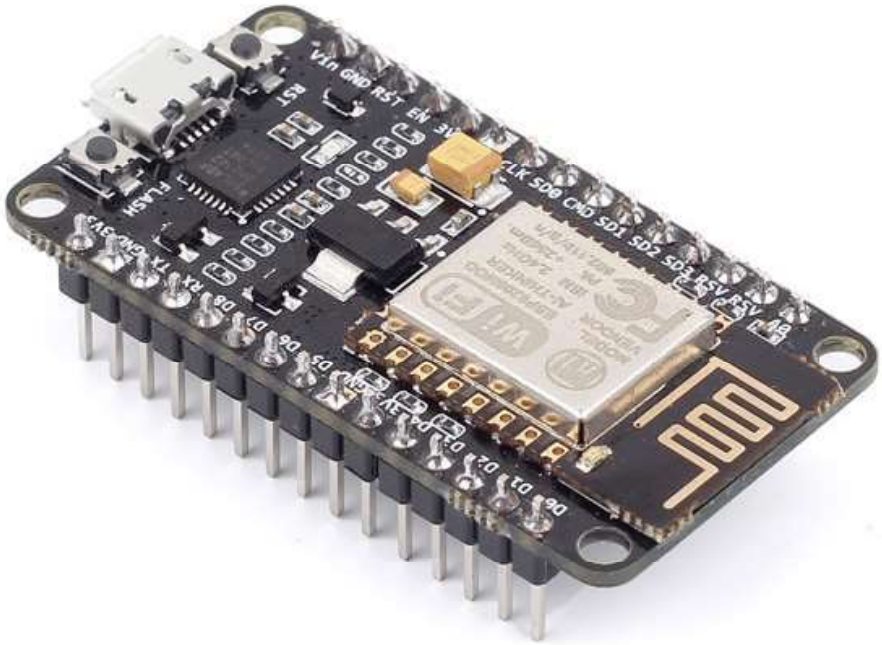
Prodejní cena Raspberry Pi 3 se nicméně pohybuje okolo 35 dolarů (v přepočtu cca 900 korun), vytvoření Mesh sítě by tedy z tohoto důvodu bylo poměrně nákladné.

Vhodnou alternativou je Raspberry Pi Zero (obr. 2.3). Rozměrově je o polovinu menší než předchozí počítače Raspberry Pi. Nedisponuje sice vestavěným WiFi čipem, avšak jeho prodejní cena se pohybuje okolo 5 dolarů (cca 125 korun). Včetně USB WiFi modu za dalších 5 dolarů ho lze pořídit za 10 dolarů (250 korun), vychází tudíž o 25 dolarů (600 korun) levněji [5].

V této práci bude tedy použito jako zařízení, na němž bude nainstalován centrální databázový systém, jenž bude ovládat WiFi Mesh síť, Raspberry Pi Zero.

2.4.2 ESP8266 - NodeMcu V3

NodeMcu V3 je prototypová deska, která se inspirovala podobným projektem, který se nazývá Arduino. Cílem obou projektů je nabídnout levnou univerzální vývojovou platformu, která je přístupná i méně technicky zdatným uživatelům.



Obrázek 2.4: NoceMcu V3 od společnosti Lolin (zdroj: https://www.aliexpress.com)

Obsahuje USB konektor, který slouží pro napájení a pro komunikaci s PC, dále pak piny pro analogovou a digitální komunikaci. NodeMcu V3 se prodává za 3 dolary, což je v přepočtu asi 80 korun. Základem této prototypové desky je WiFi čip ESP8266.

ESP8266 je malý výkonný WiFi čip navržen tak, aby zvládl obstarat bezdrátovou komunikaci, ale i kód dalšího programu, který bude mít k dispozici až 80 % prostředků. Může být využit jako WiFi klient, hotspot, webserver atd [6].

Díky své nízké pořizovací ceně v kombinaci s širokými možnostmi využití v oblasti Internetu věcí je deska NodeMcu V3 osazená čipem ESP8266 ideální pro použití jako klient WiFi Mesh sítě.

Dostupné varianty

Existuje několik prototypových počítačů postavených na čipu ESP8266. Všechny tyto mikropočítače lze jednoduše programovat z prostředí Arduino IDE, čímž se stávají nejen snadno dostupné, ale i snadno programovatelné. Patří mezi ně například Wemos D1 R2, Wemos D1 mini, WiFiMcu nebo desky NodeMcu.

Desky NodeMcu se vyrábí v několika variantách. Pro potřeby této práce byla zvolena varianta V3. Je k dostání v oficiální a neoficiální verzi. Neoficiální verze je mimo pracovního 3,3V zdroje napětí obohacena i o 5V napětí z připojeného USB [7].

3 Cíl práce

Cílem práce je vytvořit samostatně fungující WiFi Mesh síť, která bude připojena na centrální databázový systém, skrze který bude přes internet ovládána pomocí klientských zařízení.

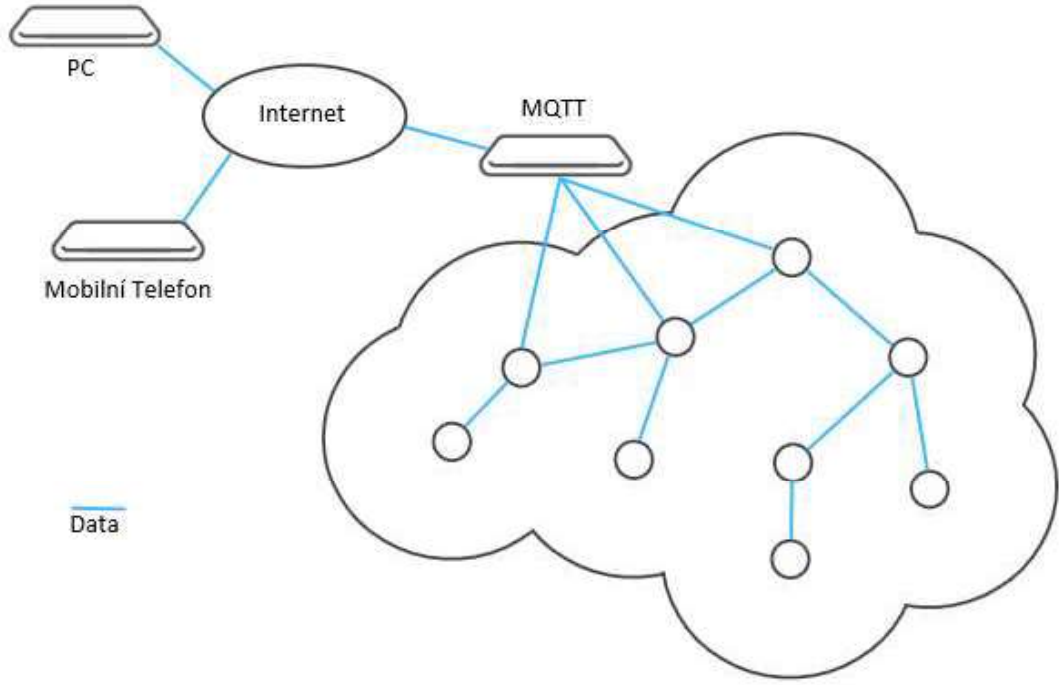
WiFi Mesh síť bude realizována pomocí čipu ESP (desky NodeMcu V3). Mesh síť bude samostatná a všechny uzly sítě si budou rovny. Při připojení nového uzlu do Mesh sítě se tato musí sama automaticky překonfigurovat a začlenit nový uzel.

Na počítači Raspberry Pi bude nainstalován ovládací MQTT server, k němuž bude připojena Mesh síť, která bude odebírat topik, přes nějž bude řízena.

Jako koncový klient bude vytvořena aplikace, jejímž prostřednictvím bude posíláno požadované nastavení Mesh sítě na MQTT server. Tímto způsobem bude aplikace Mesh síť ovládat. Pro jednoduchou demonstraci bude moci uživatel pomocí klienta ovládat LED v uzlech sítě.

V průběhu práce bude dodržen následujícího postupu.

Prvním krokem bude vytvoření jednoduché aplikace pro čip ESP8266 za účelem seznámení se s vývojovým prostředím a principem jeho fungování. V dalším kroku bude zprovozněn řídící



Obrázek 3.1: Zamýšlené zpracování řešení (zdroj: autor)

MQTT server na počítači Raspberry Pi a implementován klient pro připojení k MQTT serveru na čipu ESP8266. Následně bude vytvořena samostatná WiFi Mesh sít s použitím ESP8266 (desek NodeMcu V3), jež bude poté patřičně otestována. V dalším průběhu práce bude propojena vytvořená Mesh sít s MQTT serverem nainstalovaným na počítači Raspberry Pi pomocí dříve implementovaného klienta. Konečným úkolem bude vytvoření klientské aplikace, jež bude prostřednictvím MQTT serveru ovládat Mesh síť (obr. 3.1).

4 Praktická část

4.1 NodeMcu V3 seznámení

Pro seznámení se s deskou NodeMcu byla vytvořena aplikace blikající LED. Tento první krok byl učiněn především pro potřeby obeznámení se s vývojovými prostředími dostupnými pro programování desky a s použitým programovacím jazykem C/C++.

Po avizovaném seznámení se s vývojovým prostředím byla implementována jednoduchá aplikace, jež zapíná a vypíná LED diodu na desce v náhodném intervalu od 0 do 10. Skrze tuto aplikaci bylo možné seznámit se s principem nahrávání kódu do desky a s komunikací desky přes sériový monitor.

Vývojové prostředí

Desku NodeMcu lze programovat stejně jako Arduino desky přímo z prostředí Arduino IDE. Je pouze zapotřebí zvolit v nastavení Arduino IDE dodatečný zdroj programovatelných desek a následně nainstalovat balíček pro ESP8266, který zajistí potřebná data pro překládání programů pro tuto architekturu. [8] Po provedení uvedených úkonů je již možné desku začít programovat.

Bylo otestováno také vývojové prostředí Atom s rozšířením Atom.io, nicméně se vyskytly problémy s instalací knihoven. Knihovny, ačkoli se v projektu nacházely, vývojové prostředí při kompilaci nevidělo. Tento problém se u Arduino IDE nevyskytl ani jednou.

4.2 MQTT server

Po aplikaci blikající LED následoval krok zprovoznění MQTT serveru na počítači Raspberry Pi. Na Raspberry Pi byl nainstalován jak MQTT broker, tak i MQTT klient pro otestování funkčnosti serveru. Port MQTT serveru je v základu nastaven: pro TCP/IP port 1883 a pro SSL komunikaci port 8883.

4.2.1 Instalace MQTT serveru na Raspberry Pi

Instalace MQTT byla provedena přes terminál.[9] Nainstalován byl nejprve MQTT broker a poté klient. Po restartování zařízení byl následujícím příkazem spuštěn klient s parametrem -d (zobrazování debugovacích zpráv) a parametrem -t (název topiku), který z MQTT serveru odebírá topik „test_topic“

```
mosquitto_sub -d -t test_topic
```

Pro otestování posílání a odebírání zpráv z MQTT serveru byla v druhém okně terminálu následujícím příkazem s parametry -d (zobrazování debugovacích zpráv), -t (název topiku) a -m (zpráva

k odeslání) poslána zpráva na MQTT server do topiku „test_topic“ s testovací zprávou „Testovací zpráva“

```
mosquitto_pub -d -t test_topic -m \Testovací zpráva"
```

4.2.2 Použití MQTT serveru na NodeMcu V3

Po zprovoznění MQTT serveru na počítači Raspberry Pi byla implementována aplikace klienta na ESP. Pro implementaci byla použita knihovna PubSubClient. K připojení bylo zapotřebí nastavit název WiFi sítě (ssid), heslo pro WiFi síť a IP adresu MQTT serveru. Port je v základním nastavení nastaven takto: pro TCP/IP port 1883 a pro SSL komunikaci port 8883. Aplikace odebírá zprávy z MQTT serveru na Raspberry Pi a na základě zprávy, kterou příjme, zapíná nebo vypíná vestavěnou LED diodu. Po nastartování se ESP připojí na MQTT server a do topiku „esp“ odesílá zprávu, že je připojeno k MQTT serveru. Pro obsluhu příchozích zpráv z MQTT serveru je zde zaregistrovaná callback funkce, jež po přijetí nové zprávy vykoná daný kód pro ovládání LED. Pro ovládání LED diody odebírá topik „ESP_LED“, který funguje takto – pokud je příchozí zpráva 1, zapíná LED, pokud je zpráva 0, vypíná LED. Po nastavení stavu LED odesílá do topiku „esp“ zprávu o aktuálním nastavení LED. Celý program běží ve smyčce a kontroluje, jestli připojení k MQTT serveru neselhalo. Pokud připojení selže, klient se opětovně pokusí o připojení k serveru.

4.3 WiFi Mesh síť

Pro vytvoření WiFi Mesh sítě z desek NodeMcu postavených na WiFi čipu ESP8266 byla využita knihovna easyMesh. Jedná se o knihovnu navrženou jako True Ad-hoc síť. V případě tohoto typu sítě není potřeba předem stanovit strukturu sítě, není zapotřebí žádný centrální bod sítě, ani router nebo jemu podobné síťové prvky. Každý uzel systému může pracovat jako samostatné zařízení a automaticky organizovat strukturu sítě tak, aby byla vytvořena funkční a stabilní WiFi Mesh síť. Pro veškerou komunikaci a posílání zpráv používá tato knihovna JSON objekty. Knihovna nepoužívá k rozeznávání jednotlivých uzlů sítě IP adresy. Namísto toho je každý uzel sítě identifikován číslem čipu (chipID), které je pro každé ESP unikátní. Zprávy mohou být posílány buď broadcastem všem uzlům sítě, nebo mohou být podle zmiňovaného chipID posílány přímo na určitý uzel sítě. WiFi a bezdrátová komunikace je postavena tím způsobem, aby byla kompatibilní s Arduinem. Nepoužívá však Arduino WiFi knihovny, které měly problémy s latencemi, nýbrž nativní ESP8266 knihovny, které jsou dostupné prostřednictvím Arduino IDE.

Aplikace pro otestování funkčnosti spojení jednotlivých uzlů sítě počítala, kolik uzlů sítě aktuální uzel vidí v síti, a pomocí zablikání LED byl signalizován jejich počet. To znamená např. pokud uzel vidí připojené další dva uzly, blikne třikrát za sebou (celkově tři

uzly v síti). Toto blikání je opakováno pravidelně s 1vteřinovou pauzou mezi bliknutím.

Dalším úkonem, který uzel provádí, je odesílání zprávy broadcastem všem uzlům sítě v náhodném čase od 1 do 5 vteřin. Tato zpráva obsahuje číslo uzlu (vlastní číslování jednotlivých ESP), číslo čipu (chipID) a zbývající volnou paměť v daném uzlu.

Počet uzlů sítě je podle popisu knihovny omezen volnou pamětí, ve které se ukládá počet uzlů připojených k danému uzlu. Po nahrání programu do paměti ESP zbývá v paměti přibližně 25 kB. Při připojení nového uzlu do sítě byl zaznamenán úbytek paměti nejprve přibližně 1-2 kB a po ustálení sítě přibližně 1 kB. Odhadem by se tak k jednomu ESP mohlo připojit, včetně rezervy při kolísání využité paměti při připojení nového bodu do sítě, přibližně 10-15 uzlů. Zmíněný odhad je založen na vlastním testování se čtyřmi ESP. Popis knihovny však blíže nespecifikuje množství možných připojených uzlů. Výsledný možný počet se tedy od výše uvedených údajů může lišit.

4.3.1 Knihovna PainlessMesh

Při použití knihovny EasyMesh se vyskytlo několik problémů. Prvním problémem bylo nestabilní připojení. Po připojení nového uzlu do již stávající sítě se uzly začaly odpojovat a přepojovat k jiným uzlům s cílem rekonfigurace sítě na co nejstabilnější strukturu. Nicméně implementace této části knihovny obsahovala chyby

a po připojení nového uzlu se rekonfigurace sítě opakovala stále dokola. Mesh síť se při větším množství připojených uzlů neustálila na stabilní struktuře a při ustavičném přepojování jednotlivých uzlů tak byla v podstatě nepoužitelná. Dalším problémem byl fakt, že knihovna EasyMesh se již dále nevyvíjela, tudíž nebylo možné do budoucna počítat s opravami stávajících chyb.

Po prozkoumání alternativ pro nahrazení dosavadní knihovny EasyMesh byla nalezena knihovna PainlessMesh. Tato knihovna vychází z knihovny EasyMesh. Opravuje předešlé chyby a je aktivně opravována a rozšiřována. Knihovna PainlessMesh opravuje také nestabilitu, se kterou měla problém knihovna EasyMesh. V prvotní fázi testování pro tuto bakalářskou práci však byla dostupná pouze verze, která obsahovala chybu – jednotlivé uzly se k sobě vůbec nepřipojovaly. Tato chyba byla ale poměrně záhy opravena.

Dalším problémem, jejž knihovna PainlessMesh opravuje, je problém s názvy sítí, které ESP vytvářela. V původní knihovně vytvářelo každé ESP WiFi síť s názvem sítě a číslem čipu (chipID), takže každé ESP vytvářelo novou viditelnou WiFi síť. Oproti tomu v knihovně PainlessMesh vytváří všechny uzly WiFi síť pouze s názvem sítě, takže po připojení většího množství uzlů je viditelná pouze jedna WiFi síť. Díky nové knihovně tak byla Mesh síť stabilní a připravena pro další práci.

4.3.2 Připojení Mesh sítě k MQTT serveru

Při testování připojení Mesh sítě k MQTT serveru se objevil problém. Některé uzly sítě nebylo možné připojit k WiFi síti s přístupem k internetu. Přestože je možné uzel naprogramovat tak, aby byl připojen k WiFi síti, jež je připojena k internetu, a zároveň fungoval jako uzel sítě, k němuž je možno se připojit, neexistuje prozatím způsob, jak přinutit ostatní uzly v síti, aby se připojovaly primárně k tomuto uzlu.

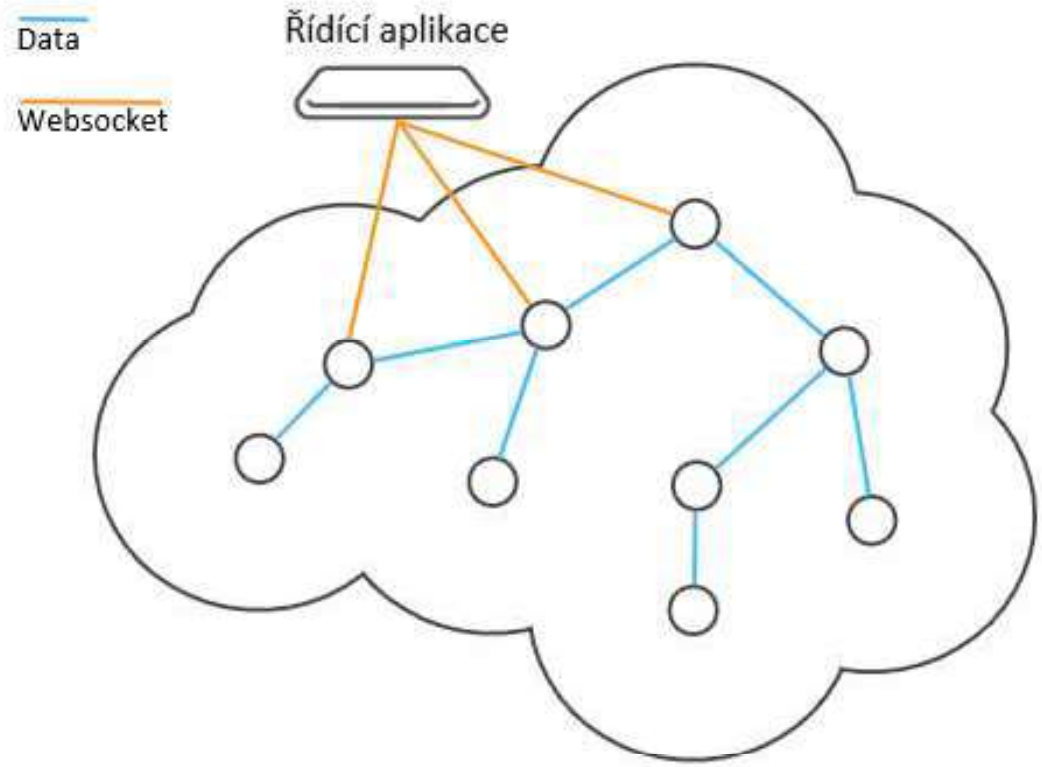
První možností, jak připojit Mesh síť k MQTT serveru, připojeného do internetu, je opakovaně odpojovat a připojovat WiFi na některém z uzlů a tímto způsobem střídat WiFi připojení z Mesh sítě a připojení k MQTT. V takovém případě by však mohlo docházet ke ztrátě dat.

Druhým možným řešením je připojit k Raspberry Pi dva WiFi adaptéry. Jeden adaptér by byl připojen do internetu a druhý k WiFi síti, kterou vytváří Mesh síť. Vzhledem k možnosti ztráty dat u prvního řešení se tato práce přiklání k řešení s dvěma adaptéry.

4.3.3 Komunikace s Mesh sítí pomocí websocketu

Další problém se vyskytl při použití knihovny pro připojení k MQTT serveru a knihovny pro WiFi Mesh.

Knihovna pro připojení k MQTT serveru využívala vlastní knihovny pro připojení k WiFi síti a nedala se tedy použít jen pro komunikaci se serverem bez použití integrovaného připojení



Obrázek 4.1: Připojení k Mesh síti pomoci websocketu (zdroj: autor)

k WiFi. Knihovny se tudíž nedaly použít zároveň.

Jako řešení problému byla zvolena websocketová komunikace. Na počítači Raspberry Pi s nainstalovaným MQTT serverem bylo nutné vytvořit aplikaci, která komunikuje přes websockety s Mesh sítí a přeposílá zprávy z MQTT serveru přímo do sítě, v níž si zprávu jednotlivé uzly sítě následně rozšiřují mezi sebou. Díky tomuto řešení bylo možné připojit se do sítě přímo a komunikovat tak i bez použití MQTT serveru. Takový způsob komunikace by se dal využít například pro správu Mesh sítě (bez nutnosti připojení k MQTT serveru)(obr. 4.1).

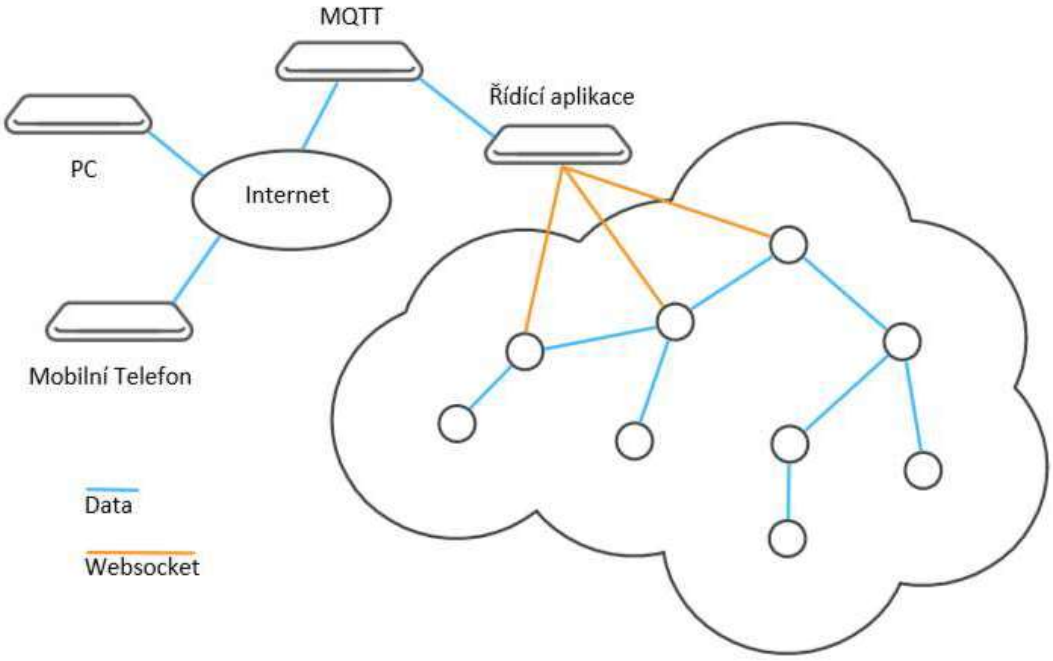
4.4 Řídící a synchronizační program Mesh sítě

MQTT server sám o sobě neuchovává poslední zadané hodnoty. Po připojení nového uzlu do Mesh sítě tento uzel nezná aktuální nastavení, které v síti existuje. Tento problém zároveň s výše zmiňovaným problémem komunikace MQTT serveru a Mesh sítě řeší řídící a synchronizační program Mesh sítě.

Jedná se o řídící aplikaci Mesh sítě, která bude odebírá daný topik z MQTT serveru, jehož prostřednictvím je možné tuto síť ovládat, uchovávat si aktuálně nastavenou hodnotu a s Mesh sítí ji synchronizovat, aby nastavení této sítě bylo aktuální. Při pokusech o naprogramování této synchronizace přímo v Mesh síti v rámci této práce vznikaly problémy se souběhem a zacyklením synchronizace. Naopak při synchronizaci pomocí řídící aplikace byla síť vždy aktuální a nedoházelo zde k žádným problémům. Aplikace byla napsána v programovacím jazyku Python, a je tudíž multiplatformní.

Po přijetí zprávy z MQTT serveru si aplikace uloží dané nastavení a následně se připojí k některému z uzlů Mesh sítě, který je aktuálně v dosahu, a přepošle mu pomocí websocketu zprávu, kterou obdržela. Zpráva je poté dál šířena v Mesh síti (obr. 4.2).

Současně aplikace provádí odesláním aktuálního nastavením, které má v paměti, každých 5 vteřin synchronizaci, a tím zajišťuje že jsou všechny připojené uzly v síti aktuální. Aplikace také přeposílá



Obrázek 4.2: Výsledná funkčnost s prostředníkem (zdroj: autor)

příchozí zprávy z Mesh sítě na MQTT server. Pro připojení k uzlům sítě musí mít program v paměti IP adresy jednotlivých uzlů, jež jsou v dosahu.

Aplikaci lze při spuštění nastavit několik parametrů. Těmi jsou adresa a port MQTT serveru, název topiku, ze kterého bude aplikace odebírat aktuální nastavení, a volba pro testovací mód. Nápovědu k aplikaci lze vyvolat parametrem -h.

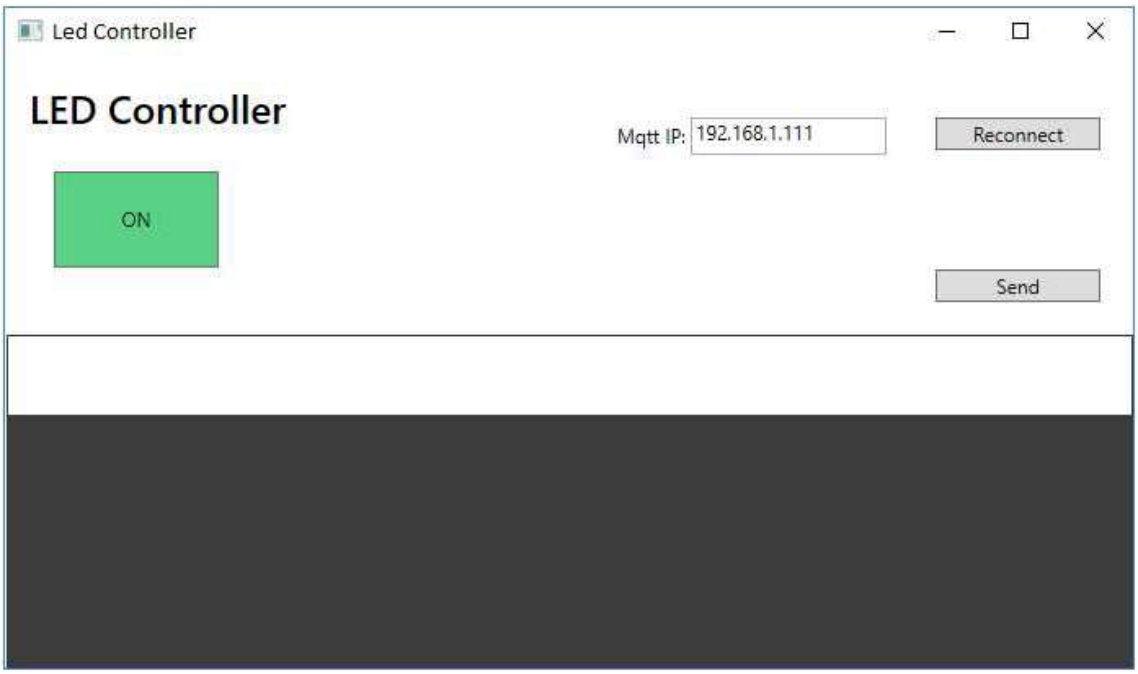
Testovací mód slouží pro testování spojení mezi jednotlivými uzly sítě. Po zapnutí v testovacím módu aplikace neodebírá nastavení MQTT serveru, ale každých 5 vteřin posílá do Mesh sítě příkaz pro vypnutí a zapnutí LED na všech zařízeních. Všechny uzly sítě, jež jsou do společné sítě připojeny, se tedy rozsvěcí a zhasínají s intervalem 5 vteřin. Díky tomuto testovacímu módu bylo snadné

ověřit, zda jsou všechny uzly připojené do sítě a zda komunikace dosahuje do všech větví sítě.

4.5 WPF aplikace pro ovládání Mesh sítě

Ovládací aplikace sítě ovládá nastavení LED v Mesh síti. Ovládání je zajištěno odesíláním aktuálního nastavení na řídící MQTT server. Aplikace obsahuje tlačítko pro vypnutí a zapnutí LED, textové pole pro posílání textových zpráv, možnosti nastavení adresy MQTT serveru, tlačítko pro obnovení spojení s MQTT serverem a jednoduchou logovací konzoli. Logovací konzole zobrazuje stav připojení k MQTT serveru, odeslané zprávy, příchozí zprávy a po přepnutí LED také odpověď, zda LED byla skutečně přepnuta, a informaci o její aktuální hodnotě.

Po zapnutí aplikace stačí pouze zadat adresu řídícího MQTT serveru, tlačítkem se připojit k serveru a LED v Mesh síti může být ovládána (obr. 4.3).



Obrázek 4.3: WPF aplikace pro ovládání Mesh sítě (zdroj: autor)

5 Vyhodnocení řešení

Výsledkem této práce bylo rozšířené řešení původního návrhu.

Pomocí ovládací aplikace je možné změnit nastavení LED v Mesh síti. Ovládací aplikace pošle přes internet dané nastavení na centrální MQTT server, který je nainstalován na lokálním počítači Raspberry Pi. Na tomto počítači se zároveň nachází řídící a synchronizační aplikace Mesh sítě. Tato aplikace příjme z MQTT serveru zprávu, která byla předtím odeslána z ovládací aplikace, a přepošle ji pomocí websocketu do Mesh sítě. Následně si uloží aktuální nastavení LED do své paměti, aby mohla poté každých pět vteřin provádět synchronizaci posíláním této hodnoty do všech uzlů Mesh sítě, a tím zajistit její aktuálnost. V Mesh síti se zpráva šíří přes všechny uzly sítě a nastaví LED na požadovanou hodnotu.

Oproti původně zamýšlenému řešení se výsledná funkčnost celku liší v několika bodech. Prvním z nich je nutnost prostředníka mezi Mesh sítí a MQTT serverem. Původní plán pracoval s Mesh sítí připojenou na stejnou síť jako MQTT server a s tím, že by tato síť měla připojení k internetu. Prostředníka bylo nutné vytvořit z důvodů omezené funkčnosti knihovny použité pro vytvoření Mesh sítě.

Druhým bodem je komunikace Mesh sítě s MQTT serverem, kterou ve výsledném zpracování také řeší prostředník mezi Mesh sítí a MQTT serverem. Tento problém byl způsoben nekompatibilitou knihovny Mesh sítě a knihovny pro komunikaci s MQTT serverem. Obě tyto knihovny používají vlastní knihovny pro připojení k WiFi síti bez možnosti použití pouze komunikačních prvků. Pokud by knihovna pro připojení k MQTT serveru umožňovala vlastní implementaci připojení k WiFi síti a poté nabízela pouze sadu nástrojů pro komunikaci s MQTT serverem, daly by se tyto knihovny použít společně. V takovém případě by nebyla potřeba websocketová komunikace prostředníka a Mesh sítě a Mesh sít by tak byla připojena přímo na MQTT server. Ostatní části výsledného řešení byli implementovány podle původního návrhu.

5.1 Řešené problémy

Při přechodu z knihovny pro WiFi Mesh síť EasyMesh na knihovnu PainlessMesh, byla knihovna PainlessMesh v nefunkčním stavu. Obsahovala chyby, které bránily připojení jednotlivých uzlů sítě k sobě. Z toho důvodu bylo v danou chvíli nutné pracovat s knihovou EasyMesh, která ovšem nefungovala správně při připojení více jak dvou uzlů sítě, a to až do doby kdy byla chyba v knihovně PainlessMesh opravena, což ovšem vedlo k zdržení v průběhu práce.

Dalším problémem, který se u Mesh sítě vyskytuje, je zabezpečení komunikace. Websocketovou komunikaci mezi Mesh sítí

a řídící aplikací, která je nainstalovaná na počítači Raspberry Pi, je možné zabezpečit pomocí ssl. Komunikace uvnitř Mesh sítě však není šifrovaná a jako zprávy posílá JSON objekty, tudíž je možné komunikaci v síti odposlouchávat. Vzhledem k tomu není Mesh síť vhodná pro použití v reálném provozu, a to až do doby, dokud komunikace nebude šifrovaná.

5.2 SW nároky

Co se týče paměťové náročnosti, nejdůležitějším bodem je Mesh síť. Velikost Mesh sítě je závislá na volné paměti v jednotlivých uzlech sítě. Uzel sítě musí uchovávat přehled o svém připojení a zároveň přehled všech uzlů, jež jsou na něj připojené. Po nahrání programu do desky zbývá přibližně 25 kB paměti. Připojení nového uzlu paměť sníží zhruba o 1 kB paměti. Odhadem by se k jednomu bodu sítě včetně rezervy pro příjem a odesílání zpráv a vnitřní proměnné mohlo připojit bezpečně zhruba 10-15 dalších uzlů sítě, avšak toto tvrzení vyžaduje nejprve podložení opravdovým testem. V případě této práce byly použity pouze 4 uzly.

Paměťová náročnost aplikace závisí na počtu použitých uzlů sítě v dosahu Raspberry Pi, jež je k uzlům sítě připojeno pomocí WiFi adaptéru. Aplikace uchovává pouze aktuální nastavení sítě, zprávy, které přeposílá a IP adresy dostupných uzlů sítě.

MQTT server a ovládací aplikace slouží pouze k přeposílání zpráv, a proto jsou jejich paměťové nároky zanedbatelné.

5.3 HW nároky

Desku NodeMcu i počítač Raspberry Pi je možné napájet buď síťovým adaptérem, nebo z baterie. Pro napájení obou hardwarů lze použít napětí 5 V a proud minimálně 250 mA. Pokud by byl k Raspberry Pi připojen monitor, je nutností použít elektrický proud o výšce minimálně 500 mA. Knihovna, využitá pro vytvoření Mesh sítě, má možnost nastavení módu s omezenou funkčností, který používá funkci sleep, pro napájení z bateriového zdroje.

Výhodou výsledného řešení je cena. Pořizovací cena počítače Raspberry Pi Zero, který byl použit a na kterém je nainstalovaný MQTT server a řídící aplikace sítě, je 5 dolarů (v přepočtu 125 korun). Na tomto počítači byly použity také dva WiFi moduly, jejichž pořizovací cena je taktéž 5 dolarů za kus.

Pořizovací cena desek NodeMcu V3, které byly použity na vytvoření Mesh sítě, činí 3 dolary (80 korun). V součtu vychází cena řešení, při kterém Mesh síť obsahuje jeden uzel, přibližně na 18 dolarů, což lze přepočíst na cca 450 korun. Každý další rozšiřující uzel sítě je možné dokoupit za 3 dolary (80 korun).

6 Závěr

Cíle bakalářské práce byly splněny. Byly prozkoumány možnosti Mesh sítí v Internetu věcí. Byla vytvořena reálná aplikace využívající principy Mesh sítí, ovládaná přes centrální MQTT server pomocí klientské aplikace. Oproti původnímu návrhu bylo kvůli nedostatkům některých z knihoven nutné vytvořit prostředníka mezi Mesh sítí a MQTT serverem, který komunikoval s Mesh sítí pomocí websocketu. Tento prostředník následně sloužil pro synchronizaci hodnot a složitější logiku ovládání sítě. Ovládání Mesh sítě pomocí websocketu nabízí možnost přímého připojení se do Mesh sítě. Otevírá také další možnosti, například vytvoření aplikace pro správu Mesh sítě nebo různých ladících nástrojů bez nutnosti komunikace s MQTT serverem. Implementace knihovny pro Mesh síť také postrádá zabezpečení komunikace v Mesh síti, vzhledem k čemuž se až do doby doplnění zabezpečení komunikace uvnitř sítě nehodí pro využití v reálném provozu.

Vzhledem k rychlému tempu, s jakým se tato oblast vyvíjí, očekávám v blízké budoucnosti doplnění dalších funkcí a knihoven. V průběhu práce již některé funkce použitých knihoven doplněny byly, ovšem až v takové fázi, že již nebylo možné je do ní začlenit.

Tato práce přináší přínos nejen obecný, týkající se zhodnocení aktuálního stavu Mesh sítí v oblasti Internetu věcí, nýbrž také obrovský přínos pro mou osobu, a to z hlediska zkušeností, jež jsem měl při vypracovávání možnost načerpat. V budoucnosti se hodlám tomuto tématu dále věnovat, buď jako domácímu koníčku, nebo v souvislosti s mou diplomovou prací.

Literatura

[1] Co je IoT?
IoT portál [online], [vid. 10. 4. 2017]. Dostupné z: https://www.iot-portal.cz

[2] HP study reveals 70 percent of internet of things devices vulnerable to attack.
HP Development Company [online], [vid. 10. 4. 2017]. Dostupné z: http://www8.hp.com/us/en/hp-news/press-release.html?id=1744676#.WRRrXt2UfDe

[3] Komáři se ženili aneb něco o MQTT
4 Makers [online], [vid. 10. 4. 2017]. Dostupné z: http://www.4makers.info/komari-se-zenili-aneb-neco-o-mqtt/

[4] PRODUCTS
RASPBERRY PI FOUNDATION [online], [vid. 10. 4. 2017]. Dostupné z: https://www.raspberrypi.org/products/

[5] *Pimoroni* [online], [vid. 10. 4. 2017]. Dostupné z: https://shop.pimoroni.com

[6] Pojďme programovat elektroniku: Vyrobíme Wi-Fi světýlko s nastavitelným jasem
Pimoroni [online], [vid. 10. 4. 2017]. Dostupné z: http://www.zive.cz/clanky/pojdme-programovat-elektroniku-vyrobime-wi-fi-svetylko-s-nastavitelnym-jasem/sc-3-a-184579/default.aspx

[7] Comparison of ESP8266 NodeMCU development boards
my2cents [online], [vid. 10. 4. 2017]. Dostupné z: https://frightanic.com/iot/comparison-of-esp8266-nodemcu-development-boards/

[8] ESP8266 - Easiest way to program so far (Using Arduino IDE)
What I Made Today [online], [vid. 10. 4. 2017]. Dostupné z: http://www.whatimade.today/esp8266-easiest-way-to-program-so-far/

[9] Instalace Mosquitto ze zdrojových kódů na Raspi a FreeBSD
4 Makers [online], [vid. 10. 4. 2017]. Dostupné z: http://www.4makers.info/instalace-mosquitto-na-raspi-ze-zdrojovych-kodu/

[10] Komár (mosquitto) na Raspi
4 Makers [online], [vid. 10. 4. 2017]. Dostupné z: http://www.4makers.info/komar-mosquitto-na-raspi/

[11] Upton, E.; Halfacree, G.: *Raspberry Pi: uživatelská příručka.* Computer Press, 2013.

A Obsah přiloženého CD

- Text bakalářské práce
  - `bakalarska_prace_2017_Zbynek_Novak.pdf`
  - `kopie_zadani_bakalarske_prace_2017_Zbynek_Novak.pdf`
- Fotografie
- Zdrojové kódy