TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

# IoT - sběr a zpracování dat snímačů s modulem ESP8266

## Bakalářská práce

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

*Autor práce:* **Marek Fišera**
*Vedoucí práce:* Ing. Miloš Hernych

Liberec 2019

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

# IoT - the data collection with the ESP8266 module

## Bachelor thesis

*Study programme:* B2646 – Information Technology
*Study branch:* 1802R007 – Information Technology

*Author:* **Marek Fišera**
*Supervisor:* Ing. Miloš Hernych

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

# IoT – sběr a zpracování dat snímačů s modulem ESP8266

*Jméno a příjmení:* **Marek Fišera**
*Osobní číslo:* M14000021
*Studijní program:* B2646 Informační technologie
*Studijní obor:* Informační technologie
*Zadávající katedra:* Ústav mechatroniky a technické informatiky
*Akademický rok:* **2018/2019**

**Zásady pro vypracování:**

1. Seznamte se s aplikací obvodu ESP8266 jako modulu – snímače vybraných fyzikálních veličin. Dále se seznamte s jeho dostupnými verzemi firmware a programováním.
2. Navrhněte vhodný způsob sběru, archivace a přenosu modulem naměřených dat k dalšímu zpracování na vybrané serverové aplikaci.
3. Vytvořte vzorovou aplikaci, která bude navržený způsob sběru, archivace a přenosu demonstrovat.

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

**Seznam odborné literatury:**


[1] ESP8266 Technical Reference [online], 127 [cit. 2017-10-18]. Dostupné z: https://espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf

[2] ESP8266EX Datasheet [online], 27 [cit. 2017-10-18]. Dostupné z: https://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[3] IERUSALIMSCHY, Roberto. Programming in Lua [online], 348 [cit. 2017-10-18]. ISBN 859037981. Dostupné z: http://www.lua.org/pil/contents.html

[4] NodeMCU Documentation [online]. [cit. 2017-10-18]. Dostupné z: http://nodemcu.readthedocs.io/en/master/

[5] Getting started with MicroPython on the ESP8266 [online]. [cit. 2017-10-18]. Dostupné z: https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/intro.html


| | |
|---|---|
| *Vedoucí práce:* | Ing. Miloš Hernych<br>Ústav mechatroniky a technické informatiky |
| *Datum zadání práce:* | 28. listopadu 2018 |
| *Předpokládaný termín odevzdání:* | 30. dubna 2019 |

L. S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci 28. listopadu 2018

# 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 texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

28. 4. 2019 Marek Fišera

# Poděkování

Tímto bych rád poděkoval panu Ing. Miloši Hernychovi za poskytnuté konzultace a rady při zpracování této práce.

# Abstrakt

Cílem této bakalářské práce je seznámit se s aplikací obvodu ESP8266 jako snímače vybraných fyzikálních veličin. Dále se seznámit s jeho dostupnými verzemi firmware a programováním. Poté navrhnout a implementovat vhodný způsob sběru vybraných fyzikálních veličin za pomocí modulu, data následně archivovat, přenést je a zpracovat na vybrané serverové aplikaci. Pro archivaci a zpracování dat je naprogramována vlastní webová aplikace v jazyce C# a Typescript. Úvodní část této práce popisuje základní informace o použitém modulu a technologie použité k samostatnému vývoji aplikace. Aplikace pro odesílání dat z modulu je naprogramována v jazyce MicroPython.



# Klíčová slova

Mikropočítač ESP8266, HTTP protokol, MQTT protokol, programování v jazyce MicroPython, IoT platformy

# Abstract

The aim of this bachelor thesis is to familiarize with the application of the ESP8266 circuit as a sensor of physical quantities. Then get familiar with its available firmware versions and programming. After that design and implement a suitable method of collecting selected physical quantities using the module, then archiving, transferring and processing the data toa selected server application. Custom web application using C # and Typescript has been programmed for archiving and processing the data. The first part of this thesis describes the basic information about the module and technologies used for the development of the application. The application for sending the data from module is programmed in MicroPython language.



# Key words

Microcomputer ESP8266, HTTP protocol, MQTT protocol, MicroPython programming, IoT platforms

# Obsah

Úvod ... 12
1 Teoretická část ... 13
1.1 Internet of Things ... 13
1.2 ESP8266 ... 13
1.2.1 ESP-12F ... 13
1.3 Firmware ... 13
1.4 MicroPython ... 14
1.5 Lua ... 14
1.6 MQTT protokol ... 15
1.7 HTTP protokol ... 16
1.8 Přídavné moduly ... 16
1.8.1 DS18B20 ... 16
1.8.2 DHT11 ... 17
1.8.3 DHT22 ... 17
1.9 IoT platformy ... 18
1.9.1 ThingSpeak ... 18
1.9.2 Grove Streams ... 19
1.9.3 Cayenne myDevices ... 20
1.9.4 Ubidots ... 20
1.9.5 Adafruit ... 21
1.10 Srovnání IoT platforem ... 22
2 Webová aplikace ... 23
2.1 Požadovaná funkcionalita ... 23
2.2 Výběr vhodných technologií ... 23
2.2.1 ASP .NET Core ... 23
2.2.2 Angular ... 24
2.2.3 REST API ... 24
2.3 Entity Framework Core ... 24
2.4 Návrh databáze ... 25
2.4.1 Tabulka User ... 25
2.4.2 Tabulka Device ... 26
2.4.3 Tabulka Sensor ... 26
2.4.4 Tabulka SensorType ... 26
2.4.5 Tabulka SensorData ... 27
2.5 Zabezpečení hesel ... 27
2.6 Použité knihovny třetích stran ... 28
2.6.1 AlertifyJS ... 28
2.6.2 Angular JWT ... 28
2.6.3 Bootstrap ... 29
2.6.4 Chart.js ... 29
2.7 Odesílaná data zařízení ... 29
2.8 Testování ... 30
2.9 Výsledná aplikace ... 30
2.9.1 Stránka Home ... 31
2.9.2 Stránka Správa zařízení ... 32
2.9.3 Stránka Dashboard ... 33
2.10 Nasazení aplikace na MS Azure ... 34
3 Aplikace pro ESP8266 ... 36

3.1 Výběr firmware ..... 36
3.2 Výběr vývojového prostředí ..... 36
3.3 Sběr fyzikálních veličin ..... 36
3.4 Archivace a přenos dat ..... 37
3.5 Návrh aplikace ..... 37
3.6 Nahrání firmware do modulu ..... 38
3.7 Zapojení snímačů ..... 39
3.7.1 Zapojení DS18B20 ..... 39
3.7.2 Zapojení DHT11 a DHT22 ..... 39
3.8 Aplikace pro sběr a přenos dat ..... 40
Závěr ..... 41
Seznam použité literatury ..... 42

# Seznam obrázků

Obrázek 1: Modul ESP-12F ..... 14
Obrázek 2: MQTT publisher – subscribe [17] ..... 15
Obrázek 3: HTTP dotaz – odpověď ..... 16
Obrázek 4: Senzor DS18B20 ..... 17
Obrázek 5: Senzor DHT11 ..... 17
Obrázek 6: Senzor DHT22 ..... 18
Obrázek 7: Výchozí vykreslování hodnot platformy ThingSpeak ..... 19
Obrázek 8: Grove Streams graf maximálních a průměrných hodnot teploty ze všech dat ..... 19
Obrázek 9: Cayenne indikátor RH ..... 20
Obrázek 10: Ubidots graf změny teploty za týden ..... 21
Obrázek 11: Adafruit graf změny RH za 24 h ..... 21
Obrázek 12: Výsledné schéma relační databáze ..... 27
Obrázek 13: Ukázka notifikace knihovny AlertifyJS ..... 28
Obrázek 14: Stránka Home ..... 31
Obrázek 15: Registrační formulář ..... 32
Obrázek 16: Stránka Správa zařízení ..... 33
Obrázek 17: Stránka Dashboard ..... 34
Obrázek 18: Skupina prostředků aplikace ..... 35
Obrázek 19: Nastavení programu NodeMCU PyFlasher pro nahrání MicroPython firmware ..... 38
Obrázek 20: Schéma pinů senzoru DS18B20 [7] ..... 39
Obrázek 21: Schéma pinů senzoru DS18B20 [8] ..... 39

# Seznam tabulek

Tabulka 1: Porovnání IoT platforem ..... 22

# Seznam zkratek

| | |
|---|---|
| API | Application Programming Interface, rozhraní pro programování aplikací |
| CRUD | Create, Read, Update, Delete, základní databázové operace |
| GND | Ground, uzemnění |
| GPIO | General-purpose input/output, obecný pin používaný u mikropočítačů |
| HTML | HyperText Markup Language, značkovací jazyk |
| HTTP | Hypertext Transfer Protocol, internetový protokol |
| I²C | Inter-Integrated Circuit, počítačová sériová sběrnice |
| IDE | Integrated Development Environment, vývojové prostředí |
| IoT | Internet of Things, Internet věcí |
| IP | Internet Protocol, protokol používaný v počítačových sítích |
| JSON | JavaScript Object Notation, datový formát |
| MCU | Microcontroller, jednočipový počítač |
| MQTT | Message Queuing Telemetry Transport, komunikační protokol |
| QoS | Quality of Service, rezervace a řízení datových toků v počítačových sítích |
| REST | Representational state transfer, architektura rozhraní |
| SHA | Secure Hash Algorithm, hashovací funkce |
| SQL | Structured Query Language, dotazovací jazyk |
| TCP | Transmission Control Protocol, spojově orientovaný protokol |
| TLS | Transport Layer Security, protokol zajišťující zabezpečenou komunikaci |
| VDD | Voltage Drain Drain, napájení napětí |

# Úvod

Internet věcí je v dnešní době velmi často diskutovaným tématem v oblasti informatiky. Zařízení spadající do této oblasti, jako jsou mikrokontroléry, si dnes může velice snadno a levně sehnat téměř kdokoliv. Tato zařízení jsou energeticky velmi úsporná a lze je využít k vývoji různých aplikací. Zařízení nabízí podobnou funkcionalitu jako osobní počítače. Tyto mikrokontroléry lze dále rozšiřovat o další moduly, které umožňují sběr různých fyzikálních veličin, jako je např. teplota nebo relativní vlhkost vzduchu.

Teoretická část práce pojednává o vlastnostech modulu ESP8266 a jeho dostupných verzích firmware. Dále jsou v této části popsány nejpoužívanější programovací jazyky, které lze využít pro vývoj aplikací pro tento modul. K přenosu dat z modulu na server a zpět je nutné použít komunikačních protokolů, a tak jsou zde popsány protokoly, které lze využít pro vývoj aplikace. Pro sběr fyzikálních veličin jsou dále popsány přídavné senzory a v neposlední řadě je zde popsáno několik nejpoužívanějších IoT platforem, které umožňují ukládání a další zpracování odeslaných dat na serveru.

V druhé kapitole práce je popsán návrh a všechny potřebné technologie a prostředky pro vývoj webové aplikace, která slouží k ukládání dat zařízení ESP8266 a jejich vizualizaci. V další části je popsána implementace vlastního řešení aplikace. V neposlední řadě tato část práce obsahuje celkový popis výsledné aplikace a její nasazení na produkční server.

V třetí kapitole je popsán návrh řešení aplikace pro sběr a odesílání dat ze zařízení ESP8266 a výběr vhodných technologií a prostředků nutných k samostatnému vývoji aplikace. Dále je v této části práce popsáno zapojení senzorů pro sběr fyzikálních veličin. Posledním bodem této části práce je popis řešení vytvořené aplikace pro ESP8266.

# 1 Teoretická část

## 1.1 Internet of Things

Internet věcí je v dnešní době často probíraným tématem v oblasti informatiky. Pro tento termín neexistuje žádná ustálená definice, ale dá se pochopit jako síť fyzických zařízení, které jsou opatřeny senzory a softwarem, který umožňuje jejich vzájemnou komunikaci. Síť nemusí znamenat pouze internet, ale může jít také o lokální síť, v rámci, které si zařízení vyměňují data. Ze sítě je zajištěna následná možnost sdílení výsledků do internetu [1].

## 1.2 ESP8266

ESP8266 je jedním z nejlevnějších Wi-Fi modulů na trhu vyráběných v Číně. Modul může samostatně fungovat jako mikropočítač, nebo ho lze použít jako Wi-Fi modul např. k jednodeskovým počítačům Arduino. Modul lze programovat v mnoha programovacích jazycích, mezi nejpoužívanější patří jazyk LuA, MicroPython a Arduino. ESP8266 umí pracovat se soubory, podporuje protokol HTTP a umožňuje komunikaci s přídavnými zařízeními prostřednictvím 1-Wire, SPI nebo I²C sběrnice. Existuje velké množství variant tohoto modulu a jednou z nich je ESP-12F, který byl použit pro účely bakalářské práce.

### 1.2.1 ESP-12F

ESP-12F je variantou modulu ESP8266 vyráběný čínskou společností Ai-Thinker. Obsahuje 32bitovou MCU jednotku. Podporované taktovací frekvence modulu jsou 80 MHz a 160 MHz. Modul používá standard IEEE802.11 b/g/n s TCP/IP protokolem. Wi-Fi pracuje při frekvenci 2,4 GHz v režimu vysílače i přijímače se zabezpečením WPA/WPA2. Dále modul disponuje flash pamětí s velikostí 4 MiB. Modul obsahuje 16 GPIO pinů, které jsou používány pro ovládání modulu, nebo pro připojení vstupních a výstupních zařízení. Modul má zabudovaný 10bitový A/D převodník. Operační napětí je v rozmezí od 3 V do 3,6 V a provozní teplota od −40 °C do 125 °C [2].

## 1.3 Firmware

Firmware je software, který je určen pro chod určitého zařízení. Mezi nejpoužívanější verze firmware používaných u mikrokontrolérů ESP8266 patří NodeMCU, což je firmware používaný pro spuštění skriptů v jazyce LuA, další verzí je MicroPython, který jak již napovídá název dokáže spuštět skripty jazyku MicroPython. V neposlední řadě je

tu také upravený Arduino firmware, který má optimalizovanou funkcionalitu pro modul ESP8266. Pro nahrání firmware do modulu poslouží nejlépe programy, jako je např. NodeMCU flasher nebo NodeMCU PyFlasher, které i mimo jiné dokáží vymazat flash paměť, čímž odstraní případné přebytečné soubory, které by mohly v paměti zůstat po předchozí verzi firmware.


*Obrázek 1: Modul ESP-12F*

## 1.4 MicroPython

MicroPython je programovací jazyk vycházející z jazyku Python 3, který obsahuje menší množství standardních Python knihoven. Jazyk je optimalizovaný pro hladký chod na mikrokontrolérech a jejich vývojových prostředích. MicroPython dále obsahuje knihovny specifické pro mikrokontroléry, jako je např. „machine“ obsahující funkce upravené pro daný hardware na konkrétní vývojové desce. Jazyk se v současné době stále rozšiřuje o další knihovny, jako jsou knihovny pro konkrétní přídavné moduly, o které lze mikrokontroléry rozšířit. MicroPython je dostupný pod open source licencí a jeho zdrojový kód je dostupný na serveru GitHub. Lze ho také upravit podle potřeb pro vlastní účely [3].

## 1.5 Lua

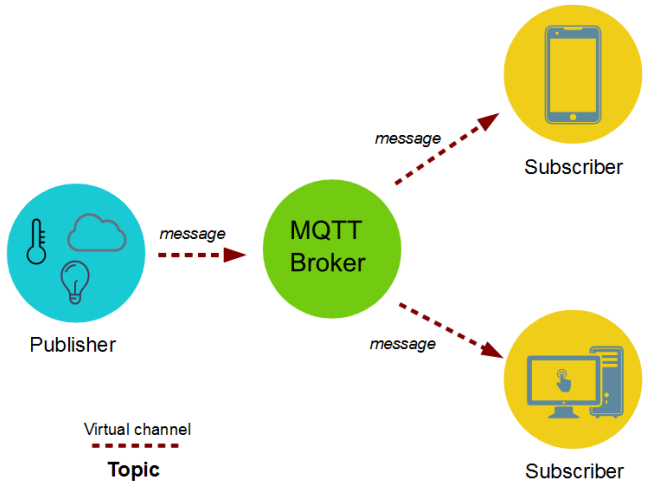
Lua je odlehčený skriptovací jazyk podporující procedurální a objektově orientované programování. Jeho jméno pochází z portugalštiny, ve které znamená „Měsíc“. Vznikl roku 1993 a je dostupný v open source licenci. Lua má implementovanou automatickou správu paměti za použití garbage collector [4].

## 1.6 MQTT protokol

MQTT (Message Queuing Telemetry Transport) je nenáročný komunikační protokol sloužící k přenosu zpráv mezi klienty za použití centrálního bodu – broker. Byl navržen především pro zařízení s nižším výkonem a pro sítě s nízkou šířkou pásma. Je navržen tak, aby minimalizoval využití sítě a minimalizoval využití hardware. Jelikož je protokol nenáročný a jednoduchý a velmi snadno se implementuje do zařízení jako jsou mikrokontroléry, velmi rychle se tak dostal do obliby. Navržen byl ve společnosti IBM roku 1999.

Přenos dat probíhá pomocí protokolu TCP a funguje na bázi publisher – subscriber. Pro komunikaci existuje centrální bod (broker), ten se stará výměnu zpráv. Zprávy jsou roztříděny do tzv. témat (topic). Klient následně do daného tématu publikuje (publish) nebo od něj odebírá (subscribe). Klient může být přihlášen k odběru více témat a zároveň také do více témat publikovat.

Obsah zpráv není nijak předem určený, nejčastěji se však používá formát JSON, BSON a data ve formě textu. Maximální velikost obsahu zprávy je omezena na 256 MB. MQTT využívá třech úrovní QoS (Quality of Service). Při použití nejnižší úrovně se zpráva odesílá bez potvrzení a ani doručení zprávy není zaručeno. Druhá úroveň zaručuje doručení zprávy alespoň jednou a nejvyšší úroveň zaručí dodání zprávy právě jednou. Pro nešifrované připojení používá protokol nejčastěji port 1883 a pro TLS spojení port 8883 [5].

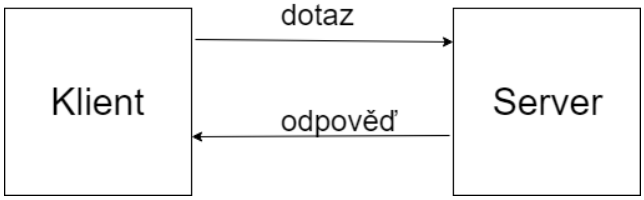



*Obrázek 2: MQTT publisher – subscribe [17]*

## 1.7 HTTP protokol

HTTP (Hypertext Transfer Protocol) je protokol určený pro přenos dat v internetu. První verze (HTTP/0.9) vznikla roku 1991 a dnes (2019) používaná verze (HTTP/1.1) roku 1997. HTTP protokol je bezstavový, což znamená, že každý příkaz je vykonaný nezávisle na předchozích příkazech. Protokol používá pro komunikaci zpravidla port TCP 80, ale může využít i jiné porty. Protokol nepodporuje šifrování dat a pro zabezpečení se nejčastěji používá TLS spojení nad TCP, tato varianta je označována jako HTTPS.

Funkce protokolu je založena na bázi dotaz – odpověď. Uživatel nejprve posílá serveru dotaz v podobě čistého textu, který obsahuje informace o verzi protokolu, označení požadovaného dokumentu a další informace např. o schopnostech prohlížeče. Server následně odešle odpověď ve formě textu, která obsahuje verzi protokolu a zprávu o úspěchu nebo chybový kód. Při úspěchu jsou ke zprávě připojena požadovaná data, např. ve formě požadovaného dokumentu [6].




*Obrázek 3: HTTP dotaz – odpověď*

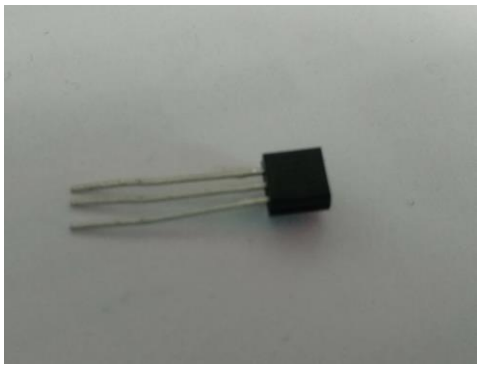
## 1.8 Přídavné moduly

Na trhu existuje mnoho externích modulů, které mohou být použity např. pro měření teploty, relativní vlhkosti vzduchu, barometrického tlaku a dalších fyzikálních veličin. Mezi často používané u modulu ESP8266 patří teplotní senzor DS18B20 a senzor DHT kombinující měření relativní vlhkosti vzduchu a teploty ve verzích DHT11 a DHT22.

### 1.8.1 DS18B20

Modul DS18B20 je teplotní senzor od firmy Maxim. Se zařízením komunikuje pomocí 1-Wire sběrnice, což je sběrnice využívající pouze jednoho pinu pro komunikaci. Senzor je schopen měřit teploty v rozmezí od −55 °C do 125 °C s přesností ± 0,5 °C v intervalu od −10 °C do 85 °C. Přesnost měřené teploty lze nastavit na 9 až 12 bitů. Senzor lze napájet přímo z komunikačního pinu díky módu „parasite power“, který eliminuje potřebu externího napájení. Každý modul má svůj 64bitový sériový kód, který umožnuje

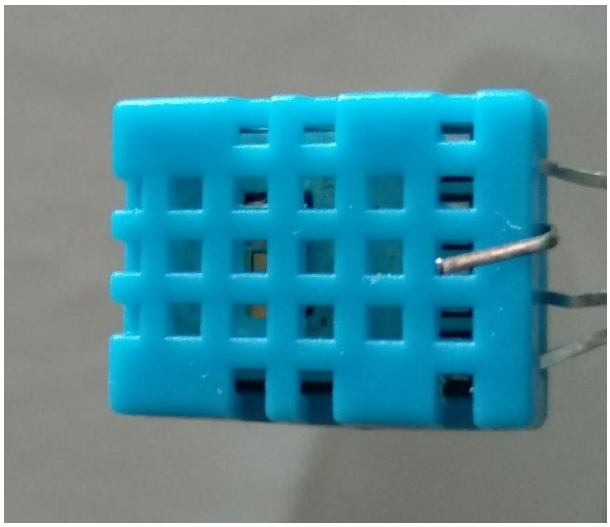
komunikaci více DS18B20 senzorů pomocí stejné 1-Wire sběrnice. Senzor je předem kalibrován a otestován výrobcem a výdrž kalibrace je garantována po jeho celou dobu životnosti [7].



*Obrázek 4: Senzor DS18B20*

### 1.8.2 DHT11

DHT11 je senzor pro měření teploty a relativní vlhkosti vzduchu. Rozsah měření teploty je od 0 °C do 50 °C s přesností ± 2 °C a rozsah měření vlhkosti od 20 % do 90 % s přesností ± 5 % RH [8]. Senzor používá jeden pin k napájení, jeden k uzemnění a jeden pin pro přenos dat. Každý DHT11 modul je striktně testován v laboratořích a výrobce zaručuje jeho přesnou kalibraci. K napájení modulu je potřebné 3-5,5 V [9].

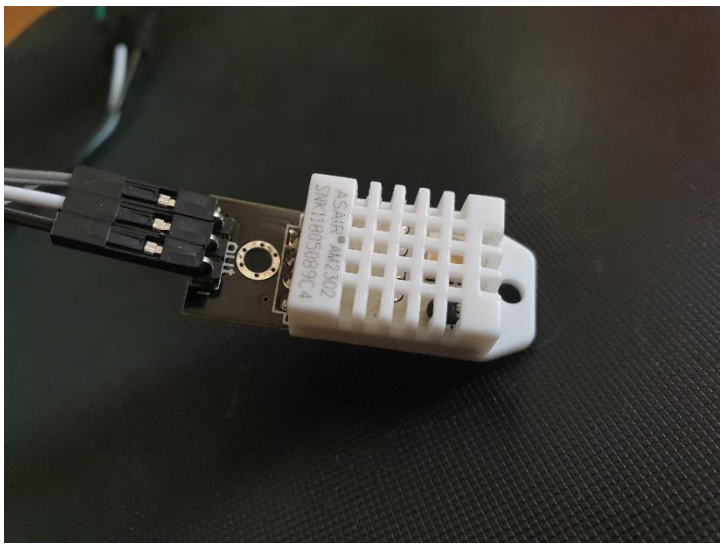


*Obrázek 5: Senzor DHT11*

### 1.8.3 DHT22

DHT22 je digitální senzor určený pro měření teploty a relativní vlhkosti vzduchu. Senzor vychází z DHT11, ale disponuje širším rozsahem měřených hodnot s vyšší přesností. Rozsah měření teploty je od −40 °C do 80 °C s přesností ± 0,5 °C. Rozsah měření relativní vlhkosti senzorem je od 0 % do 100 % s přesností ± 2 % RH. Senzor používá

jeden pin k přenosu dat, jeden k napájení a jeden k uzemnění. K napájení senzoru je nutné 3,3-6 V [10].




*Obrázek 6: Senzor DHT22*

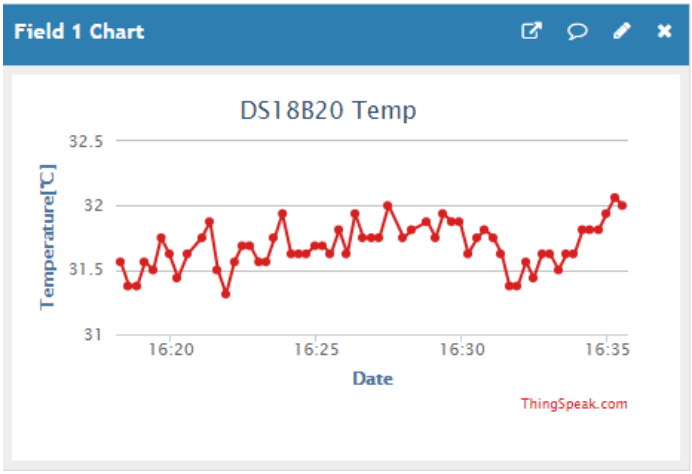
## 1.9 IoT platformy

IoT platforma je platforma, která umožňuje komunikaci mezi IoT zařízeními. Mezi další důležité vlastnosti patří uchovávání dat, jejich analýza a vizualizace. V dnešní době (2019) existují stovky takových platforem [11]. V následujících odstavcích budou popsány některé z často pužívaných v oblasti mikropočítačů, které nabízí bezplatné využívání některých jejich služeb.

### 1.9.1 ThingSpeak

ThingSpeak je IoT platforma, která je určena ke sběru dat senzorů a jejich archivaci v cloudovém úložišti. Pro zpracování dat lze použít předpřipravených grafů, které vykreslují vždy jednu hodnotu uloženou v konkrétním kanálu. Ve výchozím nastavení vykresluje posledních 60 uložených hodnot do spojnicového grafu. Na ose X vypisuje čas, kdy přišla data na server a na ose Y odeslanou hodnotu. Pro pokročilejší zpracování dat lze buď použít šablon, nebo lze data vizualizovat podle konkrétní potřeby v jazyce MATLAB. Na výběr je buď analýza dat vypsaná ve formě textu, nebo vykreslení požadovaných hodnot do grafů. Grafy lze nechat vykreslovat ve vybraném kanálu uživatele. Platforma nabízí soukromé i veřejné kanály pro zobrazování dat.

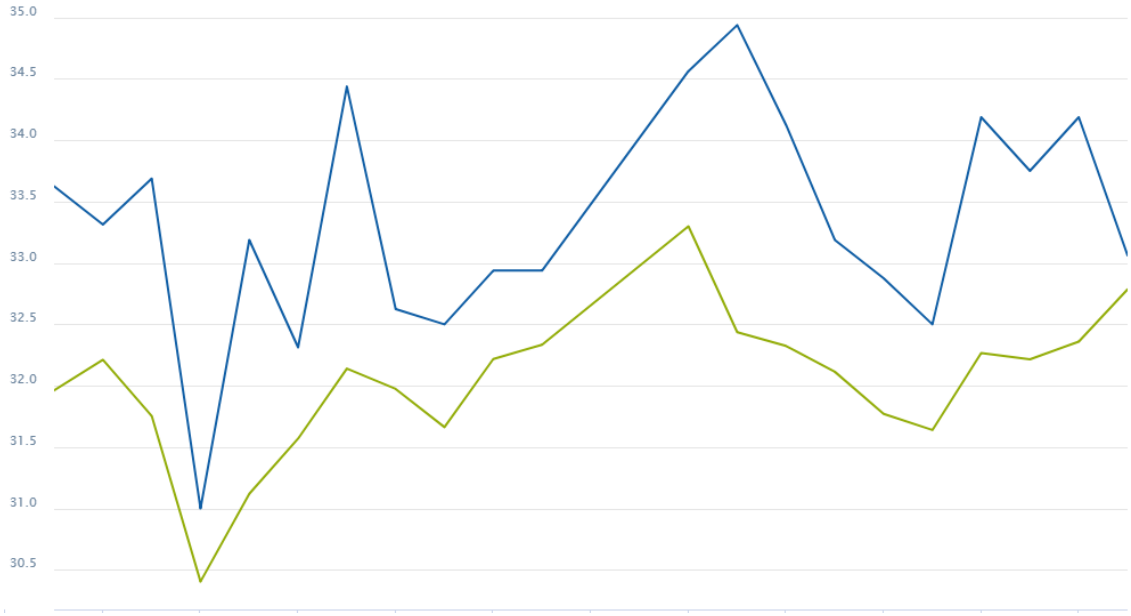
Data mohou být odesílána ze zařízení jako je Arduino, Raspberry Pi a podobného hardware. ThingSpeak využívá technologie REST a MQTT API pro přenos dat na server [12]. Výchozí vykreslování hodnot je zobrazeno na obrázku č. 7.




*Obrázek 7: Výchozí vykreslování hodnot platformy ThingSpeak*

### 1.9.2 Grove Streams

Grove Streams je IoT platforma pro analýzu dat, která vznikla roku 2011. Platforma byla vybudována pro vývojáře, kteří zde můžou budovat menší projekty i pro společnosti, které potřebují využívat milióny datových proudů ze senzorů a dalších mobilních zařízení. Služba Grove Streams nabízí pouze využití REST API pro přenos dat [13].




*Obrázek 8: Grove Streams graf maximálních a průměrných hodnot teploty ze všech dat*

### 1.9.3 Cayenne myDevices

Cayenne je IoT platforma vyvinutá společností myDevices. Jejich řešení si zakládá na jednoduchém drag and drop vývoji IoT projektů. Vývojáři mohou použít IoT řešení s minimální potřebou programování, i řešení bez nutnosti programování. Cayenne obsahuje katalog certifikovaných IoT zařízení, mezi která patří např. Arduino, ESP8266 a Raspberry Pi. Pro komunikaci se zařízeními využívá MQTT API.

Služba umožňuje vypisování uložených hodnot do připravených šablon pro určitá zařízení. Pro zobrazení aktuální hodnoty slouží jednoduché indikátory, které lze nastavit pro jakoukoliv veličinu. K dalšímu zpracování jsou k dispozici grafy, které lze použít k vykreslení hodnot ve vybraném intervalu. Dále lze na platformě využít SQL dotazů k filtrování určitých dat [14]. Na obrázku č. 9 je zobrazen ukázkový indikátor aktuální hodnoty relativní vlhkosti vzduchu.


*Obrázek 9: Cayenne indikátor RH*

### 1.9.4 Ubidots

Ubidots je platforma, která využívá mnoha protokolů pro komunikaci, jako je např. HTTP, MQTT nebo libovolný industriální protokol. Pro připojení zařízení do cloudového úložiště existuje na platformě více než 200 otestovaných knihoven s tutoriály. Pro vizualizaci dat nabízí indikátorů, grafů, tabulek a map za použití technologie HTML canvas. Dále nabízí sdílení dat veřejnými odkazy, které jsou přístupné z webových prohlížečů nebo aplikací pro mobilní telefony.

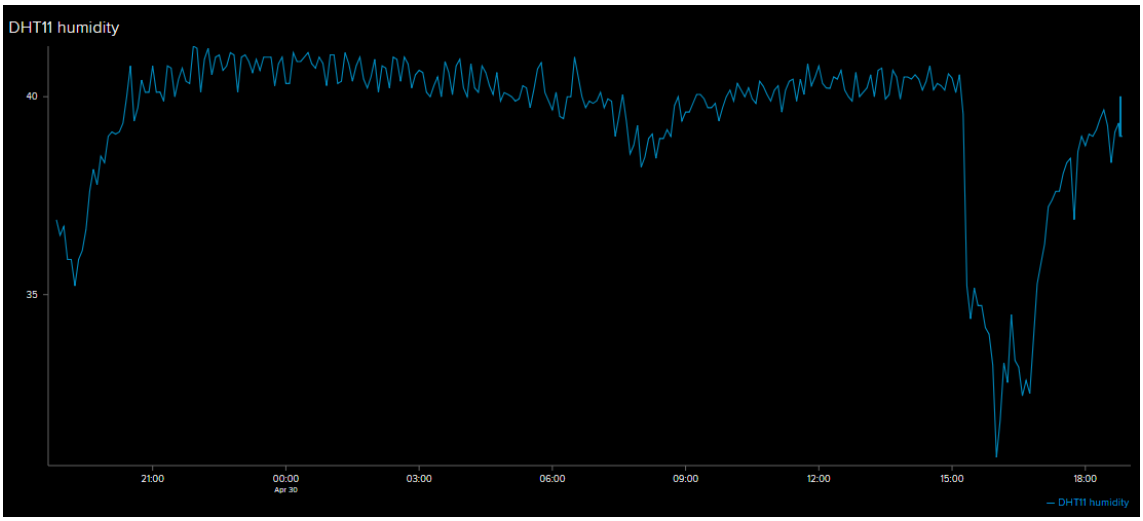
Ubidots dále nabízí zasílání notifikačních emailů, či SMS. Zasílání notifikací si může uživatel nastavit např. na událost při překročení stanovené limitní hodnoty nebo v případě, kdy data nepřicházejí na server po určitou časovou dobu. Služba je dostupná v placené verzi pro firemní účely i ve verzi zdarma pro edukativní využití. Použití edukativní verze je limitováno kredity, které slouží na platformě jako měna pro placení

služeb za příjem, uchovávání dat a odesílání notifikačních SMS zpráv. Po vyčerpání kreditů získaných za registraci musí uživatel nadále za služby platit [15]. Na obrázku č. 10 je zobrazena změna teploty za jeden týden, která je vykreslena grafem platformy Ubidots.


*Obrázek 10: Ubidots graf změny teploty za týden*

### 1.9.5 Adafruit

Adafruit je IoT platformou, která umožňuje využívat klientských knihoven pro práci se zařízeními jako je ESP8266, Raspberry Pi, Arduino a dalšími. Ke komunikaci se zařízeními využívá HTTP a MQTT protokolu. Pro vizualizaci dat nabízí uživatelsky konfigurovatelné tabulky, grafy a indikátory. Ve verzi zdarma jsou data uchovávána na serveru po dobu jednoho měsíce [16]. Na obrázku č. 11 je zobrazena změna relativní vlhkosti vzduchu za posleních 24 hodin, která je vykreslena grafem služby Adafruit.




*Obrázek 11: Adafruit graf změny RH za 24 h*

## 1.10 Srovnání IoT platforem

V této práci jsou srovnány pouze neplacené verze IoT platforem ThingSpeak, Grove Streams, Cayenne myDevices, Ubidots a Adafruit, protože mohly být otestovány. U většiny platforem je možné použít pro komunikaci se zařízeními MQTT nebo HTTP protokol. Neplacené verze jsou často limitovány množstvím odesílaných požadavků na server za určitý časový interval. Doba uchování dat může omezovat uživatele z hlediska dlouhodobější analýzy. Některé IoT platformy uchovávají data ve verzích zdarma pouze po dobu jednoho měsíce, proto nemusí být pro toto použití vhodné neplacené verze.

V tabulce č. 1 jsou porovnány parametry otestovaných IoT platforem. Mezi porovnávané parametry patří možnost způsobu komunikace se serverem, limity odesílaných dat na server, doba uchování dat a druhy dostupných triggerů. Všechny parametry jsou porovnávány v neplacených verzích uživatelských účtů. Z hlediska porovnávaných parametrů vychází ThingSpeak nejlépe, protože nabízí uchování dat až na 3 roky a jeho hlavní velkou výhodou je zpracování dat přímo na serveru v jazyce MATLAB, což ostatní platformy nenabízí.

| Server | ThingSpeak | Grove Streams | Cayenne myDevices | Ubidots | Adafruit |
|---|---|---|---|---|---|
| Komunikace (protokol) | HTTP a MQTT | HTTP | MQTT | HTTP a MQTT | HTTP a MQTT |
| Limit dat (uložených hodnot) | 3 000 000 zpráv za rok | 10 000 (string) 550 000 (ostatní datové typy) | 60 zpráv za minutu | Limitováno kredity | 30 zpráv za minutu |
| Doba uchování dat | 3 roky | neomezeně | 1 měsíc | 3 měsíce | 1 měsíc |
| Trigger (způsob notifikace) | MATLAB analýza, Tweet, HTTP request | Email, HTTP request, Upozornění, nastavení proměnné | Email, SMS | SMS, Email, Telegram | Email, HTTP request, nastavení proměnné |

*Tabulka 1: Porovnání IoT platforem*

# 2 Webová aplikace

V této části práce bude popsán návrh a řešení webové aplikace, která bude sloužit pro ukládání a zpracovaní přijatých dat z modulu ESP8266.

## 2.1 Požadovaná funkcionalita

Prvním důležitým krokem je stanovit, které funkce by měla aplikace umožňovat. Jelikož se bude jednat o aplikaci pro více uživatelů, bude důležité uživatelům umožnit registraci a následně je ověřovat pomocí hesla. Každý uživatel by měl mít možnost vytvářet nové moduly a k nim dále mít možnost připojit libovolné senzory. Uživatelé by měli mít možnost nahrávat hodnoty naměřené ze senzorů zařízení, k čemuž je nutné každému uživateli poskytnout unikátní API klíč. Další důležitou funkcí je umožnit uživatelům vymazávat data senzorů, či odstranit senzory ze zařízení, nebo odstranit celé zařízení se všemi jeho daty. Uživatelská data je nutné ukládat v centralizovaném úložišti, k čemuž lze využít databáze. Dále je nutné data vizualizovat, k čemuž lze využít tabulek nebo grafů. Pro zobrazení dat z určitého časového intervalu by mělo být uživatelům umožněno data filtrovat.

## 2.2 Výběr vhodných technologií

Pro vytvoření požadované aplikace existuje v dnešní době (2019) veliké množství technologií, z kterých lze vybírat. Nejdůležitější volbou je vhodný programovací jazyk nebo framework pro vývoj webové aplikace. Od něj se dále budou odvíjet další možnosti volby technologií.

### 2.2.1 ASP .NET Core

ASP .NET Core je multiplatformní, open source framework vyvíjený společností Microsoft pro vytváření moderních cloudových aplikací. Umožňuje vytvářet webové aplikace a služby, IoT aplikace a mobilní aplikace. Aplikace lze vyvíjet v operačních systémech Windows, macOS a Linux. Framework obsahuje integrovaný Kestrel server, který lze používat v době vývoje a jednoduše spustit příkazem *dotnet run*. V době vývoje aplikace byla poslední dostupná stabilní verze ASP .NET Core 2.1. Pro programování aplikací v tomto frameworku lze použít jazyk C#, Visual Basic nebo F# [18]. K vývoji aplikace byl zvolen programovací jazyk C#. Jazyk je pro tuto platformu nejpoužívanější, a proto je k němu dostupná velmi detailní dokumentace. V aplikaci je použit pro naprogramování backendové části.

### 2.2.2 Angular

Angular je frontendový open source webový framework, který je určen pro tvorbu single-page aplikací. Je postaven na komponentové architektuře a jako programovací jazyk používá Typescript. Hlavním prvkem každé komponenty je programová třída. Ke každé třídě je dále navázána HTML šablona a soubor obsahující CSS styly. Následně je v HTML šabloně určeno pomocí speciálních formátovacích značek, kam mají být vložena data. Tento způsob vkládání dat se označuje jako data-binding. Framework je založen na principech objektového programování. Framework je v neustálém vývoji, na kterém se nejvíce podílí společnost Google [19]. Aktuální verze frameworku použitá pro vývoj aplikace je Angular 6. V aplikaci je použit pro naprogramování frontendové části.

### 2.2.3 REST API

REST (REpresentational State Transfer) je architektura API umožňující přístup k datům, nad kterými provádí CRUD operace. Tyto operace jsou implementovány pomocí odpovídajících HTTP protokolu. Pro získání dat se používá metoda GET, což odpovídá operaci read. K odeslání dat na server se se používá metoda POST, kde jsou data obsažena v těle požadavku. Další operací je DELETE, která se používá k mazání záznamů. Poslední metodou je PUT, která je využívána pro aktualizování záznamů. REST architektura definuje v aplikaci přístup k datům.

## 2.3 Entity Framework Core

Entity Framework (EF) Core je odlehčený open source framework vyvíjený společností Microsoft, který vychází z Entity Framework 6. EF Core může sloužit k objektově-relačnímu mapování, což je technika, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem. EF Core podporuje velké množství databázových systémů, mezi které patří velmi často používané systémy MySQL a Microsoft SQL Server [20].

EF Core umožňuje provádění CRUD operací bez nutnosti psaní vlastních SQL dotazů, což usnadňuje a zpřehledňuje vývoj aplikace. Pro vývoj je nutné zvolit jeden z dostupných databázových systémů. Většina databázových systémů funguje jako služba, se kterou server typicky komunikuje prostřednictvím TCP/IP. Jedním z odlišných databázových systémů je SQLite, který ukládá všechny informace o databázi do souborů, ze kterých následně informace získává zpět při čtení. Tato skutečnost eliminuje potřebu další spuštěné databázové služby, a také lze databázi snadno zálohovat při použití

verzovacího systému, jako je např. Git. Použití SQLite je vhodné zejména pro vývoj aplikace. Po otestování aplikace a ukončení vývoje lze použít jakýkoliv jiný databázový systém z podporovaných. Velkou výhodou EF Core je, že přechod na jiný databázový systém lze provést změnou minimálního množství kódu a může být tak hotový za několik minut.

## 2.4 Návrh databáze

Před vlastní implementací je nutné vhodně navrhnout strukturu databáze tak, aby odpovídala standardům relačního modelu. Proto je nutné se držet několika pravidel. Pro předejití problému s uchováváním duplicitních dat je potřeba databázi normalizovat. Duplicita dat vzniká zejména, když je ukládáno mnoho dat do jedné tabulky a stejná data jsou následně ukládána na více místech z důvodů chybících vazeb mezi entitami. K předejití tomuto stavu je nutné entity rozdělit na menší, kde se data nebudou opakovat a vytvořit mezi nimi vazby pomocí klíčů. Tento krok dále pomáhá předejít problému s nekonzistencí ukládaných dat. Prvním důležitým klíčem je klíč primární, který jednoznačně identifikuje určitou instanci relace z databázové tabulky. Je tedy důležité zvolit primární klíč tak, aby byla zaručena jeho jedinečnost. V praxi je často používanou praktikou vytvořit uměle primární klíč, který se obvykle označuje jako id. Id se většinou automaticky inkrementuje pro každý nově přidaný záznam do databáze. Databáze se považuje za normalizovanou, pokud je převedena alespoň do třetí normální formy. Normalizovaná databáze většinou zabírá méně místa na úložišti.

Jelikož bude aplikace navržena pro více uživatelů, bude nutné si o každém z nich ukládat informace, které je dokáží rozlišit od ostatních uživatelů. První tabulkou bude tedy *User*. Každý uživatel potřebuje uchovávat informace o svých zařízeních, další tabulkou databáze bude *Device*. Zařízení zpravidla obsahují jeden, či více senzorů, k ukládání příslušných dat o nich bude vytvořena tabulka *Sensor*. K rozlišení různých druhů senzorů a jejich ukládaných dat bude vytvořena tabulka *SensorType*. Poslední a velice podstatnou tabulkou databáze bude *SensorData*, ve které budou uložena data všech senzorů zařízení. Tabulky budou detailněji popsány v následujících podkapitolách. Výsledné schéma relační databáze je zobrazeno na obrázku č. 12.

### 2.4.1 Tabulka User

Tabulka *User* bude ukládat informace o jednotlivých uživatelích. Prvním uchovávaným údajem bude uživatelské jméno, které bude uživatel zadávat při jeho registraci. Dalším

údajem bude heslo, které bude při registraci jednosměrně převedeno na kryptografickou sůl a hash. Tento krok je důležitý zejména z důvodů bezpečnosti ukládaných dat, aby hesla při případném úniku dat nebyla uložena v čitelné podobě. Dalším atributem tabulky bude API klíč, který bude uživatel používat k ověření při odesílání naměřených dat senzorů na server. API klíč bude vygenerován každému uživateli při registraci. Posledním atributem tabulky bude id, které bude jednoznačně identifikovat uživatele a bude tak primárním klíčem tabulky. Primárním klíčem tabulky by také mohl být API klíč, protože je unikátní pro každého uživatele. Z důvodu osobní preference bylo zvoleno id, protože zpřehledňuje kód při vývoji aplikace.

### 2.4.2 Tabulka Device

Tabulka *Device* bude obsahovat čtyři atributy. Prvním atributem bude název zařízení, které bude uživatel zadávat při jeho vytvoření. Dalším atributem bude popis zařízení, tento atribut bude nepovinný a uživatel si bude moci zvolit, zda ho bude chtít při vytváření vyplňovat. Primárním klíčem tabulky bude id, které bude zařízení jednoznačně identifikovat. Dalším atributem tabulky bude id uživatele, což bude cizí klíč z tabulky *User*. Tento atribut je zejména důležitý, aby bylo jasně stanoveno, komu zařízení patří. Každý uživatel bude mít možnost vlastnit více zařízení, a proto bude mezi tabulkou *User* a *Device* vytvořena vazba 1 : N.

### 2.4.3 Tabulka Sensor

Prvním atributem tabulky *Sensor* bude název senzoru. Dalším atributem bude popis, stejně jako u tabulky *Device*. Dále bude obsahovat dva cizí klíče, prvním z nich bude id zařízení, aby bylo možné zjistit, ke kterému zařízení senzor náleží. Dalším cizím klíčem bude *SensorTypeId*, který bude do tabulky vložen při vytváření nového senzoru, kdy uživatel zvolí, o jaký typ senzoru se jedná. Tento cizí klíč bude odpovídat číselné hodnotě id typu senzoru z tabulky *SensorType*. Primárním klíčem tabulky bude id senzoru. K jednotlivým zařízením bude možné připojit libovolný počet senzorů, a proto bude mezi tabulkou *Device* a *Sensor* vytvořena vazba 1 : N.

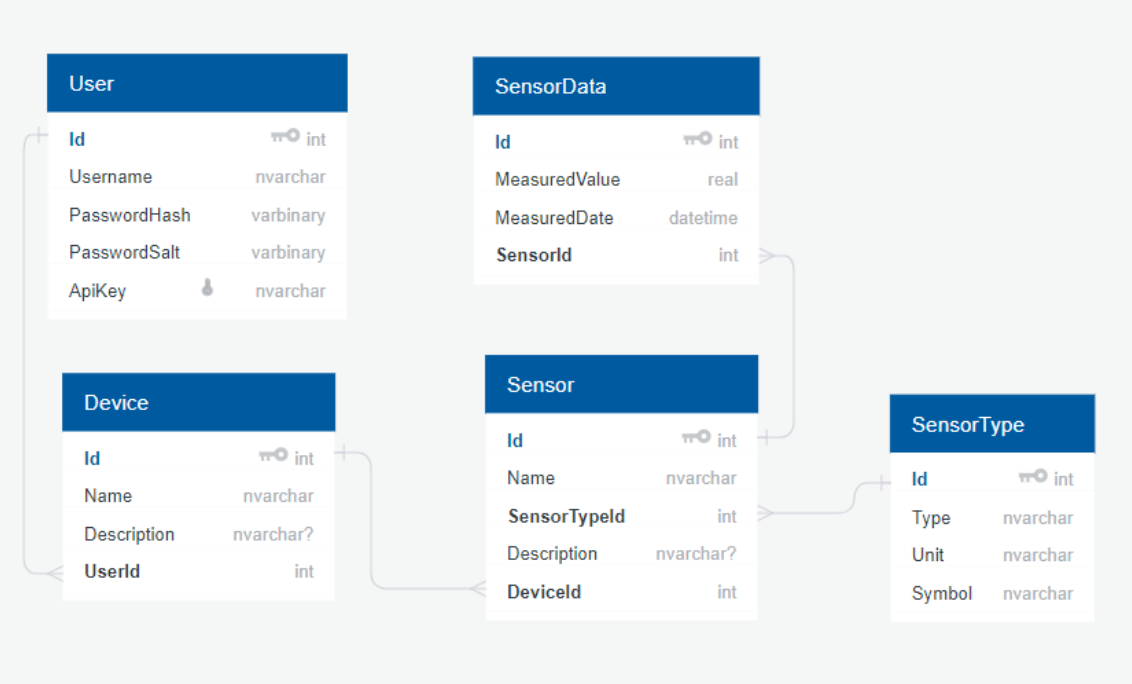
### 2.4.4 Tabulka SensorType

Tabulka *SensorType* bude sloužit k odlišování typů senzorů. Prvním atributem tabulky bude typ senzoru, který bude důležitý k rozpoznání, jakou veličinu daný senzor měří. Dalším atributem bude fyzikální jednotka a dalším značka jednotky. Tyto dva atributy budou důležité pro určení typu senzoru a při vykreslování dat senzorů do grafů.

Primárním klíčem tabulky bude id typu senzoru. Uživatelé nebudou mít oprávnění do tabulky přidávat nové typy senzorů, a budou tak pevně stanoveny. Mezi tabulkou *SensorType* a *Sensor* bude vytvořena vazba 1 : N.

### 2.4.5 Tabulka SensorData

V tabulce *SensorData* budou uložena data všech senzorů. Prvním atributem tabulky bude změřená hodnota senzoru. Dalším atributem bude datum, kdy byla hodnota změřena, tento údaj budou odesílat zařízení v čas změření v zařízení, nebude tedy generován při přijetí změřené hodnoty na server. Dále bude v tabulce cizí klíč *SensorId*, kvůli identifikaci, kterému senzoru ukládaná data náleží. Primárním klíčem tabulky bude id.




*Obrázek 12: Výsledné schéma relační databáze*

## 2.5 Zabezpečení hesel

Důležitým krokem aplikace je, v jaké podobě uchovávat hesla uživatelů v databázi. Ukládání hesel ve formě textu by bylo nebezpečné kvůli případnému úniku dat z databáze. Proto bylo nutné hesla uchovávat v bezpečnější formě. K uložení hesla se může použít hashovacích funkcí, jako je např. SHA nebo MD5, které heslo převedou do podoby hexadecimálního čísla. Použití samotné hashovací funkce ale z důvodu bezpečnosti nestačí, jelikož pro stejný vstup vždy vygeneruje stejný výstup a existují tabulky s již vypočítanými hesly, které se používají při útoku silou při úniku dat z databáze.

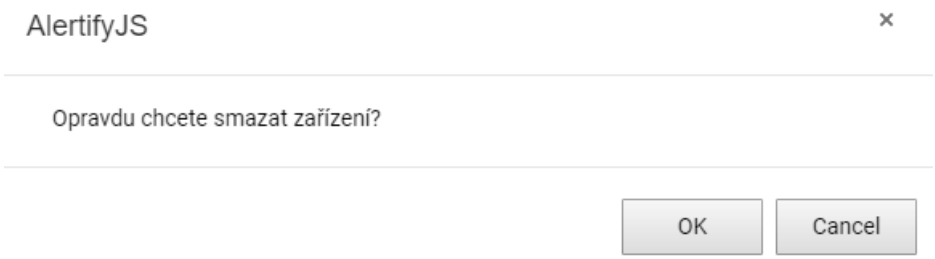
Pro uložení hesel bylo tedy využito kryptografické soli, která k heslu v podobě hash přidá ještě další unikátní vygenerovanou sekvenci znaků. Tento přístup zvyšuje náročnost na výpočetní výkon při případném útoku a zaručuje, že i stejná hesla jsou převedena na unikátní hash. V aplikaci bylo využito hashovací funkce SHA512.

## 2.6 Použité knihovny třetích stran

V aplikaci bylo použito několik knihoven třetích stran, jelikož implementace některých funkcionalit by byla příliš náročná. Ke správě balíčků ve frontendové (Angular) části aplikace je využit NPM (Node Package Manager). NPM je nástroj, který umožňuje instalaci javascriptových knihoven a jejich odinstalaci. V následujících podkapitolách jsou vypsané nejdůležitější použité knihovny a jejich podrobnější využití. Knihovny se do projektu nainstalují příkazem *npm install [název knihovny]*.

### 2.6.1 AlertifyJS

AlertifyJS je javascriptový framework pro zobrazování dialogů a notifikací v prohlížeči. V aplikaci je využit zejména k poskytnutí zpětné vazby uživatelům při provádění různých operací. Mezi operace patří např. vytváření nových zařízení, či jejich mazání. Uživateli je při těchto operacích zobrazen potvrzovací dialog, aby se předešlo provedení nechtěných operací. Dále je knihovna využita pro zobrazování notifikací při úspěchu, či neúspěchu přihlášení a odhlášení. Posledním využitím je pro zobrazování notifikací, pokud se na serveru vyskytne chyba.




*Obrázek 13: Ukázka notifikace knihovny AlertifyJS*

### 2.6.2 Angular JWT

Knihovna Angular JWT slouží k ověření platnosti JWT tokenů. V projektu je tato knihovna použita k přihlašování uživatelů. JWT tokeny jsou ukládány v tzv. „local storage“ na straně uživatele. Uživatel je přihlášen, pokud local storage obsahuje jeho

neexpirovaný a validní JWT token. Pokud je token označený jako nevalidní, potom není uživatel přihlášen a nemá přístup ke svým datům, ani k ostatním funkcionalitám aplikace.

### 2.6.3 Bootstrap

Bootstrap je open source knihovna pro tvorbu responzivních webových aplikací. Obsahuje mnoho hotových stylů, které lze použít při vytváření layoutu stránky. Návrh layoutu stránky je založen na tzv. „grid systému“. Hlavním prvkem grid systému je kontejner, který obsahuje řádky a sloupce. Tento systém umožňuje přesně stanovit umístění všech grafických prvků na webové stránce [21].

V aplikaci je použit pro kompletní vytvoření layoutu webových stránek. Dále je použit pro veškeré formuláře a tlačítka. Použití knihovny Bootstrap zajišťuje v aplikaci uživatelsky přívětivý grafický design.

### 2.6.4 Chart.js

Chart.js je open source knihovna napsaná v jazyce Javascript. Knihovna umožňuje vizualizaci dat s využitím širokého množství grafů. V aplikaci je knihovna využita pro vizualizaci dat přijatých z uživatelských zařízení. Konkrétně je využit spojnicový graf na stránce *Dashboard*. Grafy nabízí velké množství možností konfigurace, což umožňuje popsat osy grafu, přidat popisky, či nastavit různá barevná schémata a styly vykreslování dat. Spojnicový graf implementuje při vykreslování Bézierovu křivku.

## 2.7 Odesílaná data zařízení

Data odesílaná ze zařízení na server webové aplikace jsou ve formátu JSON. Jsou posílána metodou POST, kde jsou obsažena v těle zprávy. Metoda, která přijímá data na serveru zpracovává přijatá data jako pole, proto je možné ze zařízení odesílat více naměřených hodnot v jednom požadavku. Hlavička zprávy obsahuje API klíč uživatele, podle kterého následně server ověří, zda uživatel vlastní senzory v databázi, do kterých se odesílají naměřená data. Metoda vrací tři stavové kódy, v závislosti, jestli proběhla v pořádku, nebo si při průběhu přijímání dat vyskytl problém. V případě, že byl problém s ověřením API klíče uživatele nebo se uživatel pokusila zapsat data do senzorů, které mu nepatří metoda vrátí stavový kód *401 Unauthorized*. Pokud byla přijata data senzorů ve špatném formátu, vrátí metoda stavový kód *400 Bad Request* s textovou informací „Špatný formát dat“. V případě, že všechno proběhne v pořádku a data se uloží do databáze metoda vrátí stavový kód *200 OK*, kde v těle zprávy zobrazí všechna nově uložená data.

**Adresa URL**

*api/devices/send-data*

**Vzor hlavičky požadavku**

*Content-Type: application/json*

*apiKey: sGJUSswB1YE8VL1G*

**Vzor těla požadavku**

```
[
      {
      "MeasuredDate": "02/23/2019 11:00:00",
      "MeasuredValue": "22.35",
      "SensorId": "1"
      },
      {
      "MeasuredDate": "02/23/2019 11:00:00",
      "MeasuredValue": "23.35",
      "SensorId": "2"
      },
      {
      "MeasuredDate": "02/23/2019 11:00:00",
      "MeasuredValue": "24.35",
      "SensorId": "3"
      }
]
```

## 2.8 Testování

Testování aplikace bylo prováděno v průběhu vývoje i po dokončení celé aplikace. Testování bylo zejména důležité, jelikož se tím ověřila správná funkčnost aplikace a odhalilo se tím několik chyb, které byly následně opraveny. Pro testování REST API byla použita aplikace Postman. Aplikace umožnila otestování všech metod sloužících k získávání, či úpravě a mazání dat. Jelikož bylo potřeba v průběhu vývoje data několikrát smazat a upravit, tak bylo vhodné vytvořit testovací data. Testovací data byla vygenerována do samostatných souborů, kde byla uložena ve formátu JSON. K použití testovacích dat byly vytvořeny pomocné metody, které data ze souborů deserializovaly a následně se uložila data do databáze vždy, když databáze žádná data neobsahovala.

## 2.9 Výsledná aplikace

V následujících podkapitolách budou detailněji popsány všechny stránky výsledné aplikace.

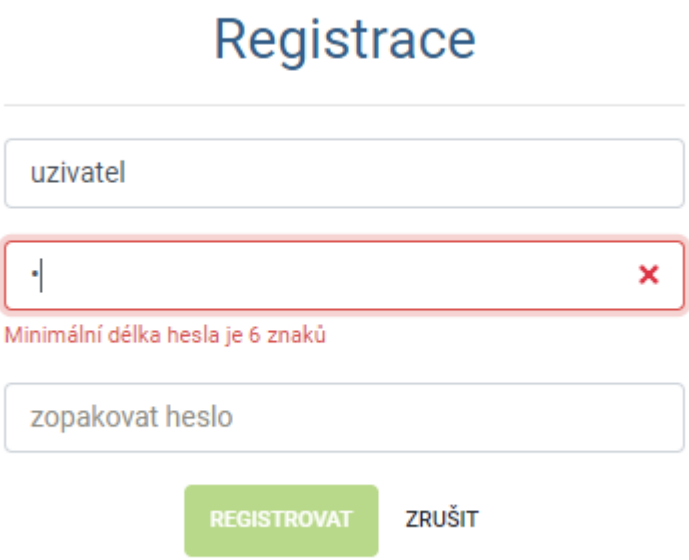
### 2.9.1 Stránka Home

Stránka Home je výchozí stránkou aplikace. Tato stránka se zobrazí i uživatelům, kteří nejsou registrováni a přihlášeni. Ostatní stránky aplikace se uživatelům zobrazí v rozcestníku pouze po úspěšné registraci a jejich přihlášení. Stránka obsahuje tlačítko *registrovat*, které při kliknutí zobrazí příslušnou komponentu. Komponenta obsahuje formulář, ve kterém uživatel zadá jméno, pod kterým se následně bude přihlašovat a heslo, které musí dvakrát potvrdit.

Vstupy formuláře jsou ověřovány pomocí validátorů. První vstup uživatelské jméno je povinný. Pokud tento vstup uživatel nevyplní, zobrazí se pod vstupem upozornění „Zadejte uživatelské jméno“. Další vstup heslo je také povinný, pokud ho uživatel nevyplní, objeví se pod vstupem hláška „Zadejte heslo“. Minimální délka hesla je z důvodů bezpečnosti stanovena na alespoň 6 znaků, proto pokud tento požadavek heslo nesplní, zobrazí se upozornění „Minimální délka hesla je 6 znaků“. Posledním vstupem formuláře je požadavavek pro zopakování hesla uživatelem, aby nedošlo k případnému překlepu. Tento vstup porovnává obsah s předchozím vstupem heslo. Pokud se hesla obou vstupů neshodují, zobrazí ze uživateli hláška „Hesla se musí shodovat“.

Formulář je označen za validní pouze v případě, kdy jsou jeho všechny vstupy zároveň také validní. Pokud je formulář validní, zvýrazní se uživateli tlačítko *registrovat*, kterým může všechny zadané údaje odeslat na server. Uživatelské jméno je na serveru převedeno na malá písmena, aby se předešlo registrování uživatelů se stejným uživatelským jménem, které by se pouze lišilo ve velikosti písmen. Server po odeslání požadavku zkontroluje, zda databáze již neobsahuje uživatelské jméno. Pokud jméno obsahuje, vypíše se uživateli upozornění, že uživatelské jméno existuje. Pokud požadavek proběhne v pořádku, je uživatel úspěšně registrován a jeho údaje se zapíší do databáze.


*Obrázek 14: Stránka Home*



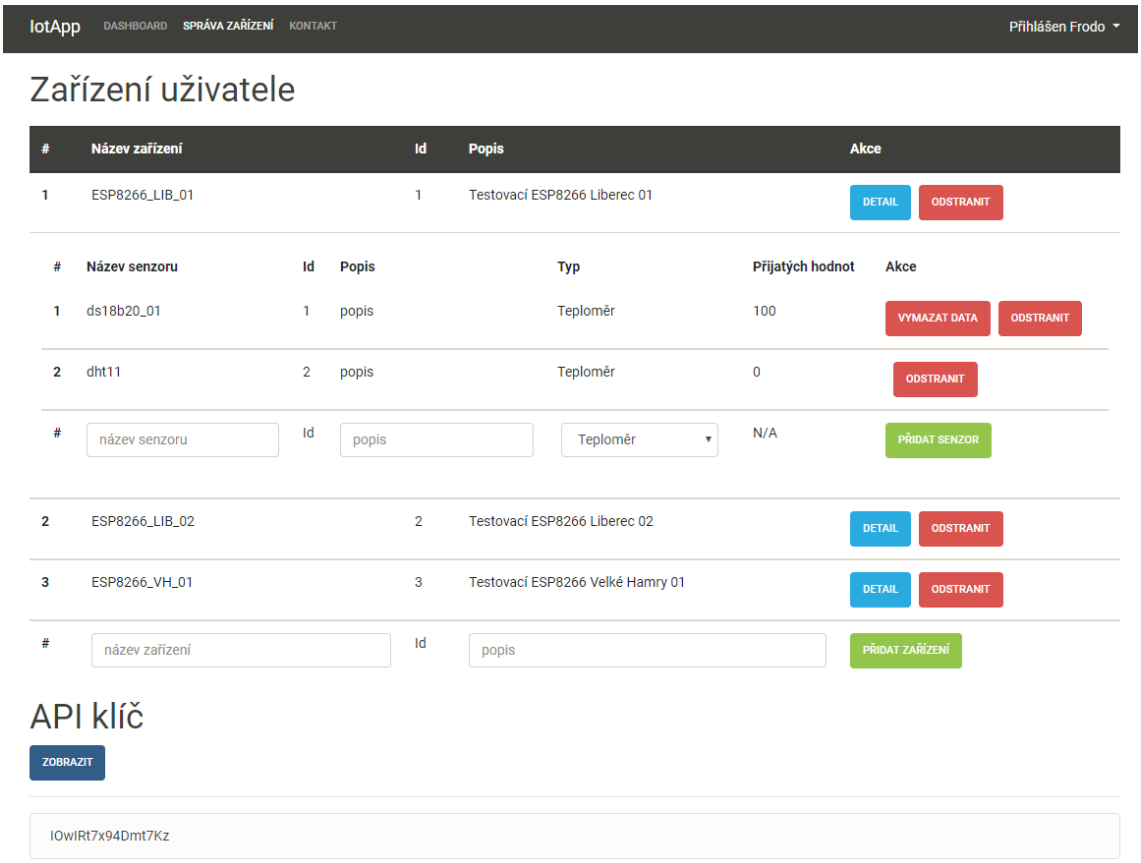

*Obrázek 15: Registrační formulář*

### 2.9.2 Stránka Správa zařízení

Stránka Správa zařízení je pro uživatele přístupná pouze po přihlášení. Uživatelům slouží k vytváření a úpravě zařízení a senzorů. Po registrování uživatele a jeho prvotním přihlášení stránka neobsahuje žádná zařízení ani senzory. Stránka pouze obsahuje API klíč, který slouží uživateli pro odesílání dat ze zařízení. Pokud tedy uživatel ještě nevytvořil žádné zařízení, stránka obsahuje pouze dva vstupy a tlačítko sloužící k vytvoření prvního zařízení. Prvním vstupem je název zařízení a druhým je jeho popis. Pokud uživatel vyplní tyto vstupy a stiskne tlačítko *Přidat zařízení*, vytvoří se první zařízení, jehož údaje se zapíšou do databáze.

Pokud uživatel vytvořil nějaká zařízení, zobrazí se všechna v tabulce, kde jsou o nich zobrazené všechny údaje. Mezi údaje zařízení patří jeho název, id a popis. V posledním sloupci tabulky jsou zobrazena dvě tlačítka. Červeně označené tlačítko *Odstranit* slouží k odstranění zařízení a jeho všech případných senzorů a dat z databáze. Po použití tohoto tlačítka je zobrazen uživatelům dialog, který musí potvrdit, pokud chce zařízení opravdu smazat. Druhé, modře označené tlačítko *Detail* slouží k zobrazení všech náležících senzorů k danému zařízení.

V detailu zařízení se o senzorech zobrazuje jejich název, id, popis, typ senzoru a počet přijatých hodnot ze senzoru zařízení. V posledním sloupci tabulky je tlačítko *Odstranit*, které odebere senzor zařízení a všechna jeho data. Pokud senzoru obsahuje alespoň jednu přijatou hodnotu, zobrazí se ještě ve sloupci tlačítko *Vymazat data*, které odstraní všechna přijatá data vybraného senzoru. Obě operace jsou ošetřeny dialogem podobně, jako u odstranění zařízení. Pod všemi vypsanými senzory zařízení je řádek, který slouží k přidání nového senzoru k zařízení. V tomto řádku uživatel vyplní vstupy s údaji o názvu senzoru a popis a v posledním vybere typ senzoru z již vytvořených předpisů uložených v databázi. Po vyplnění těchto údajů a stisku tlačítka *Přidat senzor* se v databázi vytvoří senzor se zadanými údaji a s id vybraného zařízení. Pod tabulkou je tlačítko, které po stisku zobrazí API klíč uživatele.




*Obrázek 16: Stránka Správa zařízení*

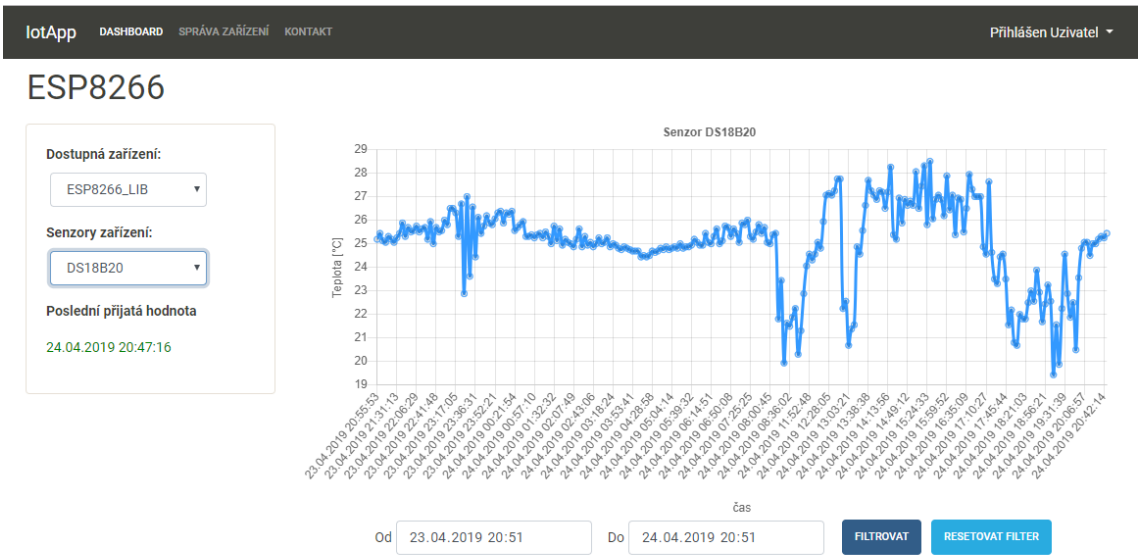
### 2.9.3 Stránka Dashboard

Stránka Dashboard je uživatelům přístupná pouze po úspěšném přihlášení. Stránka slouží uživatelům k zobrazování dat ze senzorů zařízení. Na tuto stránku je uživatel vždy přesměrován po úspěšném přihlášení. Na levé straně stránky je panel, který obsahuje přepínač všech zařízení uživatele. Pod výběrem zařízení je zobrazen další přepínač, který

slouží k výběru senzoru zařízení. Pokud senzor obsahuje přijatá data, je v panelu ještě zobrazen datum a čas poslední naměřené hodnoty.

Pravá strana stránky obsahuje graf, který je vykreslován knihovnou *Chart.js*. Graf zobrazuje data vybraného senzoru zařízení. Nad grafem je zobrazen název aktuálně vybraného senzoru. Na ose X je zobrazen datum, kdy byla hodnota naměřena senzorem. Osa Y představuje hodnotu naměřené fyzikální veličiny. Při načtení stránky jsou uživateli vždy vykreslena data z prvního senzoru prvního zařízení naměřená za posledních 24 hodin.

Pod grafem jsou dvě tlačítka k ovládání filtrování dat a dva vstupy. Vstupy slouží k nastavení uzavřeného časového intervalu, z kterého chce uživatel vykreslit hodnoty do grafu. Prvním tlačítkem uživatel potvrzuje časový interval a odesílá požadavek na server, kde jsou hodnoty času v požadavku předány jako tzv. query parametry. Server vyfiltruje všechny hodnoty vybraného senzoru podle zadaných parametrů, a následně je vykreslí do grafu. Druhé tlačítko slouží k resetování filtrování a při stisku vykreslí všechna data aktuálně zvoleného senzoru.




*Obrázek 17: Stránka Dashboard*

## 2.10 Nasazení aplikace na MS Azure

Po řádném otestování aplikace byla aplikace nasazena na produkční server. Pro připravení souborů Angular části aplikace byl použit příkaz *ng build --prod*. Jelikož příkaz ve výchozím nastavení používá nástrojů pro redukci souborů, občas způsobuje drobné chyby v aplikaci. V tomto případě způsoboval chybné zobrazování CSS stylů

v některých částích aplikace. Proto byl použit příkaz *ng build –prod –build-optimizer=false*, který těchto nástrojů nevyužívá, tudíž nezpůsoboval nadále problémy s grafickým zobrazením. Použití příkazu zapříčinilo větší výslednou velikost souborů o několik desítek kilobajtů, což v tomto případě nemá žádný dopad na funkčnost a běh aplikace.

Dalším krokem byla změna databázového systému ze SQLite na Microsoft SQL Server, jelikož nabízí více databázových operací a možnost správy databáze nástrojem Microsoft SQL Server Management Studio. Databáze nyní nebude nadále načítána ze souboru, ale bude se serverem komunikovat prostřednictvím protokolu TCP/IP. Jelikož Entity Framework podporuje mnoho databázových systémů, umožňuje mezi nimi jednoduchou konverzi za pomoci migrací. Proto byla vytvořena příkazem *dotnet ef migrations add* nová migrace a příkazem *dotnet ef database update* vytvořena nová databáze včetně všech tabulek pro databázový server Microsoft SQL Server.

Po připravení všech potřebných souborů aplikace bylo ještě nutné nakonfigurovat prostředky Microsoft Azure portálu, které jsou nutné pro chod aplikace. Prvním krokem bylo vytvoření tzv. skupiny prostředků, která představuje „kontejner“ obsahující všechny prostředky používané pro chod aplikace. Prvním přidaným prostředkem byl databázový server SQL Server. Dalším prostředkem byla databáze s vybranou základní kapacitou 2 GB, což bez problému pro tuto aplikaci postačí. Dalším konfigurovaným prostředkem byl „Plán služby App Service“, ve kterém se nastavuje, jaký výpočetní výkon aplikace potřebuje a jaký operační systém bude používat server aplikace. Poslední potřebný prostředek byl „App Service“, který obsahuje veškerý kód aplikace. Pro nahrání aplikace na server bylo využito verzovacího systému Git. Na obrázku č. 21 jsou vyobrazeny všechny použité prostředky na portálu Microsoft Azure.

| NÁZEV | TYP | UMÍSTĚNÍ | |
|---|---|---|---|
| iotapp-fisera | App Service | Západní Evropa | ••• |
| iot-app-service-plan | Plán služby App Service | Západní Evropa | ••• |
| iota-sql | SQL Server | Západní Evropa | ••• |
| IotAppDb (iota-sql/IotAppDb) | Databáze SQL | Západní Evropa | ••• |
| sqlvaivc2pd2mxmavy | Účet úložiště | Západní Evropa | ••• |

*Obrázek 18: Skupina prostředků aplikace*

# 3 Aplikace pro ESP8266

V této části bakalářské práce bude popsán výběr vhodných technologií a prostředků pro vývoj aplikace, která bude sbírat a odesílat data ze senzorů modulu ESP8266. Po popisu výběru technologií a návrhu aplikace zde bude popsána implementace vlastního řešení.

## 3.1 Výběr firmware

Existuje mnoho verzí firmware, již zmíněních v rešeršní částí bakalářské práce. Nejvhodnější variantou pro tento typ práce byl firmware MicroPython upravený pro modul ESP8266 ve verzi 1.10, který podporuje objektové programování a obsahuje mnoho knihoven, které usnadňují práci s externími moduly a zpřehledňují tak výsledný kód aplikace.

## 3.2 Výběr vývojového prostředí

Pro samostatné naprogramování aplikace bylo potřeba najít vhodné vývojové prostředí (IDE), které umožňuje komunikaci s modulem ESP8266. Dále bylo nutné, aby IDE umožňovalo základní operace se soubory uloženými v paměti modulu. Prvním takovým vývojovým prostředím bylo ESPlorer, které je vytvořeno pro psaní skriptů v jazyce Lua a MicroPython. Bohužel však působilo potíže při nahrávání skriptů do modulu a občas v prostředí selhávalo grafické rozhraní. Dalším otestovaným IDE bylo PyCharm od společnosti JetBrains, které je výborné pro programování v jazyce Python, jelikož obsahuje množství zabudovaných funkcí urychlujících programování aplikací. Pro spojení s modulem ESP8266 nabízelo MicroPython plugin verze 1.0, který však byl v první vydané verzi a stále ve fázi vývoje. Přes několik pokusů se nepodařilo navázat spojení s modulem. Dalším otestovaným IDE bylo uPyCraft, které nabízelo manipulaci se soubory uloženými v paměti modulu a kontrolu správné syntaxe jazyku MicroPython, což postačilo k naprogramování aplikace. Žádné komplikace, které nastaly s předešlými IDE už neproběhly.

## 3.3 Sběr fyzikálních veličin

První fyzikální veličinou, která šla měřit bez použití přídavných modulů bylo napětí. Napětí lze měřit přímo z modulu díky již zabudovanému A/D převodníku. Pro sběr dalších fyzikálních veličin bylo nutné připojení dalších senzorů. Prvním zvoleným byl teplotní senzor DS18B20, protože je snadno dostupný, a i přes jeho nízkou cenu dokáže měřit s přesností ± 0,5 °C. Dalším modulem byl DHT11, který je velmi levný a umožňuje

kromě snímání teploty i měření relativní vlhkosti vzduchu. Posledním použitým senzorem byl DHT22. Senzor DS18B20 komunikuje s modulem ESP8266 prostřednictvím 1-Wire sběrnice.

## 3.4 Archivace a přenos dat

Protože modul ESP8266 obsahuje poměrně nízkou vnitřní paměť (4 MiB), bylo nutné data ukládat na externí úložiště. K ukládání naměřených dat bude sloužit databáze webové aplikace. Pro přenos dat bude využito HTTP protokolu.

## 3.5 Návrh aplikace

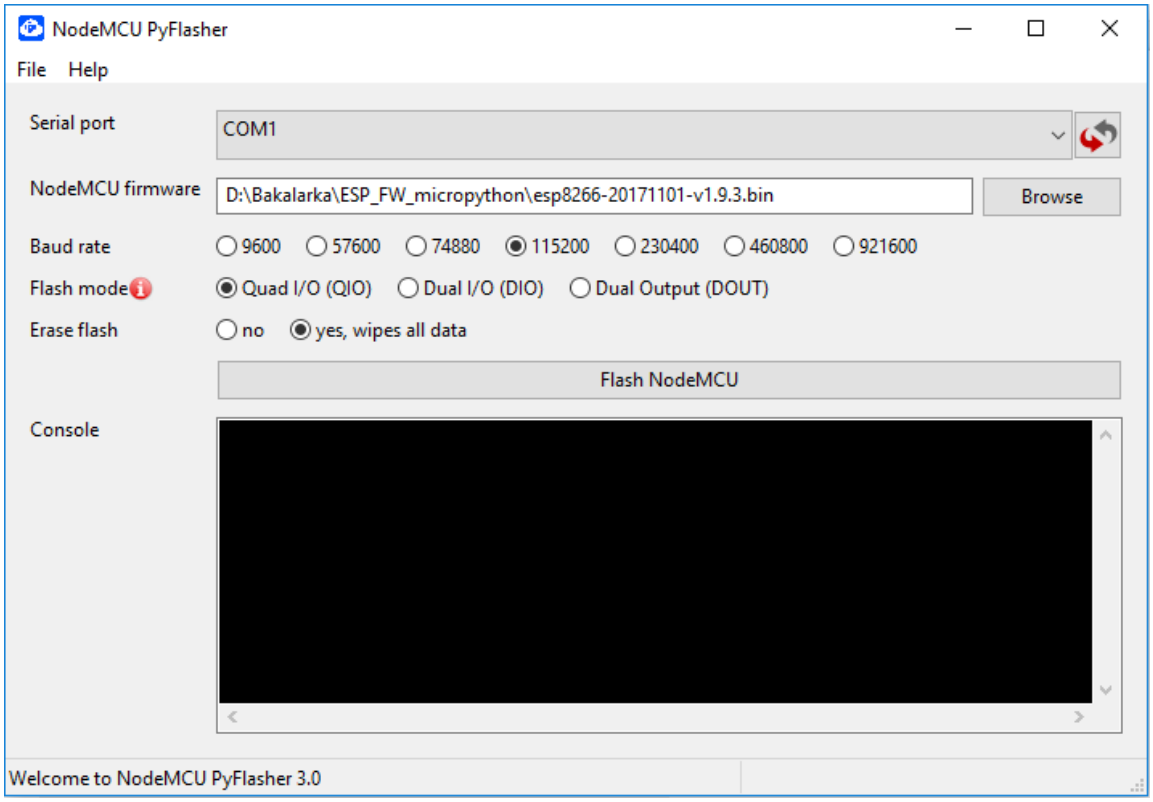
Pro návrh aplikace je nutné stanovit několik základních bodů. Nejprve je nutné naměřit požadované fyzikální veličiny. Pro změření napětí modulu existuje MicroPython knihovna *machine*, která využívá zabudovaného A/D převodníku. Pro sběr teploty senzoru DS18B20 je knihovna *ds18x20*. Pro sběr dat ze senzorů DHT11 a DHT22 existuje knihovna *dht*, která umožňuje změření teploty a relativní vlhkosti vzduchu.

Dále je nutné, aby byl modul ESP8266 připojen k internetu, kvůli odesílání dat do externího úložiště. V MicroPython slouží k tomuto účelu knihovna *network*, která umožňuje připojení do sítě prostřednictvím Wi-Fi. Po připojení modulu k internetu je také nutné nastavit správný čas a datum, protože modul neobsahuje baterii a nemá tak možnost udržovat informaci o aktuálním času, když je odpojen od napájení.

Dalším nutným bodem je samotné odesílání dat do externího úložiště. Jelikož je v této práci použito REST API, bude k přenosu dat použito HTTP protokolu a knihovna *urequests*. Pro odesílání dat na server v určené časové periodě je za potřebí využít funkce časovače z knihovny *time*.

## 3.6 Nahrání firmware do modulu

Pro nahrání firmware do modulu bylo nutné zvolit vhodný program. Prvním takovým byl velmi často používaný program NodeMCU flasher, který však po nahrání MicroPython firmware do modulu způsoboval náhodné resetování modulu a pády aplikace. Dalším otestovaným programem byl NodeMCU PyFlasher, který nepůsobil žádné podobné potíže. Pro samostatné nahrání firmware bylo nutné správně zvolit sériový port, který byl použit k přenosu dat. Dále bylo nutné zadat cestu k binárnímu souboru se samostatným firmware, přenosovou rychlost, flash mód a zda chceme vymazat všechna data z flash paměti modulu. Vymazání flash paměti modulu je dobrá praktika, protože se tímto krokem může předejít problémům s funkčností modulu, kdy může flash paměť obsahovat zbytky kódu staré aplikace. Nastavení zvolená pro toto konkrétní řešení jsou vyobrazena na obrázku č. 11.



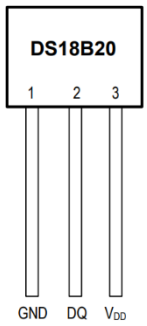

*Obrázek 19: Nastavení programu NodeMCU PyFlasher pro nahrání MicroPython firmware*

## 3.7 Zapojení snímačů

Snímače DS18B20, DHT11 a DHT22 bylo nutné správně zapojit k modulu ESP8266, aby bylo zajištěno jejich napájení a přenos nasbíraných dat. Jejich zapojení je stručně popsáno v následujících odstavcích.

### 3.7.1 Zapojení DS18B20

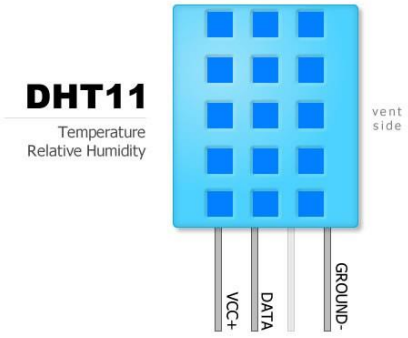
Prvním senzorem, který byl potřeba zapojit k modulu ESP8266 je teplotní čidlo DS18B20. Senzor byl zapojen k modulu podle obrázku č.12 z oficiální dokumentace senzoru. První pin GND (Ground) byl připojen k uzemnění modulu ESP8266, druhý pin DQ byl připojen k GPIO pinu modulu, který zajišťuje přenos dat a poslední VDD pin k napájení.




*Obrázek 20: Schéma pinů senzoru DS18B20 [7]*

### 3.7.2 Zapojení DHT11 a DHT22

Další senzory, které bylo nutno správně připojit jsou DHT11 a DHT22. Senzory mají 4 piny, z nichž se jeden nezapojuje. První pin senzoru slouží k napájení, druhý pro přenos dat a poslední pro připojení k uzemnění. Schéma pinů obou senzorů je identické, proto je na obrázku č. 13 zobrazeno pouze schéma DHT11.




*Obrázek 21: Schéma pinů senzoru DS18B20 [8]*

## 3.8 Aplikace pro sběr a přenos dat

Aplikace pro sběr a přenos dat byla naprogramována v jazyce MicroPython za použití vývojového prostředí uPycraft. V následujících odstavcích budou popsány všechny body aplikace.

- measure_values.py

  Soubor obsahuje funkce pro měření požadovaných fyzikálních veličin. První funkce získává hodnotu teploty z teplotního čidla DS18B20. Teploměr je připojen k GPIO pinu na pozici 13. Pro konvertování hodnoty teploty po čtení musela být nastavena prodleva 750 ms, kvůli omezení časového intervalu čtení hodnoty. Funkce vrací hodnotu teploty datového typu float. Dále je zde funkce pro získávání dat ze senzoru DHT22, která vrací teplotu a relativní vlhkost vzduchu. Funkce vrací hodnoty v datovém typu tuple. Poslední funkce získává hodnotu napětí ze zabudovaného A/D převodníku v modulu ESP8266.

- wifi.py

  Ve skriptu je obsaženo nastavení nutné pro připojení do Wi-Fi sítě. Pro připojení do sítě je nutné vyplnit identifikátor a heslo. Dále obsahuje funkci, která slouží k připojení do sítě Wi-Fi i kontrolu, zda je již spojení zprostředkováno.

- main.py

  Tato část aplikace obsahuje implementované všechny funkce potřebné k měření požadovaných fyzikálních veličin a funkci k připojení do sítě prostřednictvím Wi-Fi. Nejprve se aplikace pokusí modul připojit k internetu, pokus opakuje, dokud se nezdaří. Po připojení k internetu je nastaven správný aktuální čas a datum. Poté je část aplikace pro sběr a přenos dat spouštěna cyklicky. Každý cyklus proběhne nejdříve kontrola připojení k internetu, potom proběhne změření teploty z teplotního čidla DS18B20, změření teploty a relativní vlhkosti z čidla DHT22 a změření hodnoty napětí modulu. Změřená data jsou následně společně s aktuálním časem převedena do formátu JSON. Poté jsou data odeslána na server metodou POST, kde hlavička metody obsahuje uživatelův API klíč a tělo obsahuje naměřená data ve formátu JSON. Pro případ pádu aplikace je implementována výjimka, která restartuje modul ESP8266 a zavede ho základních nastavení pro znovuuvedení chodu aplikace. Měření a odesílání dat je prováděno každých pět minut.

# Závěr

Výsledkem této práce je aplikace napsaná v programovacím MicroPython, která zajišťuje sběr dat ze snímačů připojených k modulu ESP8266. Pro měření napětí modulu bylo využito integrovaného A/D převodníku. K měření teploty byl použit externí senzor DS18B20 a k měření relativní vlhkosti vzduchu senzor DHT22. Naměřená data jsou odesílána ve formátu JSON na vytvořenou serverovou aplikaci. Komunikace mezi modulem ESP8266 a serverem je zajištěna protokolem HTTP.

Backend serverové aplikace je naprogramován v jazyce C# za použití frameworku ASP .NET Core 2.1. Frontendová část aplikace je napsána v jazyce Typescript ve frameworku Angular 6. Aplikace je navržena pro více uživatelů, které ověřuje pomocí JWT tokenů. Uživatelům je po úspěšné autentizaci umožněna správa vlastních zařízení a vizualizace naměřených dat. Správa zahrnuje přidávání nových zařízení a senzorů, či jejich mazání. Data každého senzoru zařízení jsou vykreslována do spojnicových grafů, mezi kterými lze přepínat. Zobrazené hodnoty grafu lze filtrovat uzavřeným časovým intervalem.

Aplikace byla v době vývoje průběžně testována, což odhalilo několik chyb, které byly následně opraveny. Po dokončení vývoje byla aplikace otestována na lokálním serveru a opravena o několik dalších chyb. Nakonec byla aplikace ve finální verzi nasazena na cloudovou platformu Microsoft Azure.

Do budoucna by mohla být webová aplikace rozšířena o automatizované testy. Další užitečné rozšíření by bylo přidání uživatelských rolí, což by např. umožnilo přidání role administrátora, který by za pomoci nově přidaných prostředků mohl spravovat všechny uživatelské účty a data. Modul ESP8266 by mohl být rozšířen o baterii, která by umožňovala chod ve venkovním prostředí. Aplikace s modulem by mohla posloužit jako obdoba meteorologické stanice.

Vývoj webové aplikace i aplikace pro modul ESP8266 byl cennou zkušeností, jelikož využití podobných modulů se stále rozrůstá a v budoucnu budou jistě mít velmi široké uplatnění.

# Seznam použité literatury


[1] POHANKA, Pavel. Internet věcí [online]. [cit.2018-05-14] vyd. Dostupné z: http://i2ot.eu/internet-of-things/

[2] ESP-12F WiFi Module. *Elecrow* [online]. b.r. [cit. 2018-05-14]. Dostupné z: https://www.elecrow.com/download/ESP-12F.pdf

[3] *MicroPython - Python for microcontrollers* [online]. b.r. [cit. 2018-05-14]. Dostupné z: https://micropython.org/

[4] Lua. *Lua.org* [online]. b.r. [cit. 2018-05-14]. Dostupné z: https://www.lua.org/

[5] MALÝ, Martin. Protokol MQTT: komunikační standard pro IoT. *Root.cz* [online]. b.r. [cit. 2018-05-14]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/

[6] *HTTP Tutorial* [online]. b.r. [cit. 2018-05-14]. Dostupné z: https://www.tutorialspoint.com/http/index.htm

[7] DS18B20. *Maxim Integrated* [online]. b.r. [cit. 2018-05-14]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[8] DHT11 teploměr a vlhkoměr digitální. *Arduino-Shop.cz* [online]. b.r. [cit. 2018-05-14]. Dostupné z: https://arduino-shop.cz/docs/produkty/0/133/1500635986.pdf

[9] Temperature and humidity module DHT11 Product Manual. *Aosong* [online]. b.r. [cit. 2018-05-14]. Dostupné z: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf

[10] *Digital-output relative humidity & temperature sensor/module DHT22* [online]. Aosong Electronics Co.,Ltd, b.r. [cit. 2019-04-26]. Dostupné z: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[11] SCULLY, Padraig. 5 Things To Know About The IoT Platform Ecosystem [online]. online. vyd. [cit.2018-05-14]. Dostupné z: https://iot-analytics.com/5-things-knowabout-iot-platform/

[12] ThingSpeak [online]. online. vyd. [cit.2018-05-14]. Dostupné z: https://www.mathworks.com/help/thingspeak/

[13] Grove Streams, LLC [online]. online. vyd. [cit.2018-05-14]. Dostupné z: https://cz.linkedin.com/company/grove-streams-llc

[14] About - myDevices.com. Creators of Cayenne IoT Project Builder - myDevices.com [online]. online. vyd. [cit.2018-05-14]. Dostupné z: https://mydevices.com/about/

[15] IoT platform features. IoT platform | Internet of Things [online]. online. vyd. [cit.201805-14]. Dostupné z: https://ubidots.com/platform/

[16] IO - Adafruit. IO [online]. online. vyd. [cit.2018-05-14]. Dostupné z: https://io.adafruit.com/

[17] VOJÁČEK, Antonín. IoT MQTT prakticky v automatizaci - 1.díl - úvod [online]. online. vyd. [cit.2018-05-14]. Dostupné z: https://automatizace.hw.cz/iot-mqtt-prakticky-vautomatizaci-1dil-uvod.html

[18] *Introduction to ASP.NET Core* [online]. Redmond, Washington, USA: Microsoft, 2019 [cit. 2019-04-26]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/?view=aspnetcore-2.1

[19] *Angular Features & Benefits* [online]. Google, b.r. [cit. 2019-04-26]. Dostupné z: https://angular.io/features

[20] *Entity Framework Core* [online]. Microsoft, 2016 [cit. 2019-04-26]. Dostupné z: https://docs.microsoft.com/cs-cz/ef/core/

[21] *Get started with Bootstrap* [online]. b.r. [cit. 2019-04-26]. Dostupné z: https://getbootstrap.com/docs/4.3/getting-started/introduction/