



## BAKALÁŘSKÁ PRÁCE

---

# Senzorické řešení chytré domácnosti s automatickou diagnostikou komunikace

---

*Autor:*  
Patrik NACHTMANN

*Vedoucí práce:*  
Ing. Martin BULÍN, MSc.

Závěrečná práce k získání akademického titulu *Bakalář (Bc.)*  
*v oboru Systémy pro identifikaci, bezpečnost a komunikaci*

Katedra kybernetiky

20. dubna 2020

## *Prohlášení*

Prohlašuji, že jsem tuto bakalarskou práci vypracoval samostatně pod vedením pana Ing. Martina Bulína, MSc. Všechny použité podklady, ze kterých jsem čerpal informace, jsou uvedeny v seznamu použité literatury a citovány v textu.

V Plzni dne 20. dubna 2020

.....

Patrik NACHTMANN

*„When Henry Ford made cheap, reliable cars, people said, ‘Nah, what’s wrong with a horse?’ That was a huge bet he made, and it worked.“*

Elon Musk

ZÁPADOČESKÁ UNIVERZITA

*Abstrakt*

Fakulta aplikovaných věd

Katedra kybernetiky

Bakalář (Bc.)

**Senzorické řešení chytré domácnosti s automatickou diagnostikou  
komunikace**

Patrik NACHTMANN

Abstrakt text ...

## *Poděkování*

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce panu Ing. Martinu Bulínovi, MSc. za odborné konzultace a trpělivost v průběhu celého projektu. Především za hodnotné rady a poznámky v průběhu návrhu řešení a následnou cennou zpětnou vazbu při psaní této práce.

Dále děkuji panu Ing. Janu Švecovi, Ph.D za nápady při tvorbě zadání této práce.

# Obsah

<b>Abstrakt</b>	<b>iii</b>
<b>1 Úvod</b>	<b>1</b>
1.1 State of the Art . . . . .	1
1.2 Thesis Objectives . . . . .	1
1.3 Thesis Outline . . . . .	1
<b>2 Hardwarové komponenty</b>	<b>2</b>
2.1 Mikročip ESP8266 . . . . .	3
2.2 Senzory . . . . .	9
2.2.1 Teplotní čidlo DS18B20 . . . . .	9
2.2.2 Vlhkoměr DHT11 . . . . .	10
2.2.3 Čidlo intenzity osvětlení TSL2591 . . . . .	12
2.2.4 Čidlo barometrického tlaku a teploty BME280 . . . . .	13
2.2.5 Pohybové čidlo AM312 . . . . .	14
2.2.6 Magnetické čidlo LS311B38 . . . . .	15
2.3 Raspberry Pi . . . . .	16
<b>3 Síťová komunikace a databáze</b>	<b>19</b>
3.1 Protokol MQTT . . . . .	20
3.1.1 Struktura zpráv . . . . .	21
3.1.2 Hierarchie zpráv . . . . .	23
3.2 Ukládání dat do databáze . . . . .	24
3.3 Webserver . . . . .	24
<b>4 Diagnostika a detekce anomálií</b>	<b>27</b>
4.1 Detekce chyb na úrovni ESP8266 . . . . .	28
4.2 Detekce anomálií na základě klasifikace . . . . .	29
4.3 Diagnostika stavu čidel na serveru . . . . .	29
<b>5 Webové rozhraní</b>	<b>31</b>
5.1 Programování frontendu . . . . .	31
5.2 Overview . . . . .	32
5.3 Analytics . . . . .	33
5.4 About . . . . .	36
<b>6 Závěr</b>	<b>38</b>
6.1 Future Work . . . . .	38
<b>Bibliography</b>	<b>39</b>
<b>A1 Structure of the Workspace</b>	<b>40</b>

# Seznam obrázků

2.1	Vztahy mezi použitým hardwarem a měřenými veličinami . . . . .	2
2.2	Vývojová platforma NodeMCU s modulem ESP8266 . . . . .	3
2.3	Diagram odesílání zpráv na základě vzniku události . . . . .	5
2.4	Diagram periodicky opakovaného odesílání zpráv . . . . .	6
2.5	Diagram architektury kódu na microchipu ESP8266 . . . . .	7
2.6	Schéma obvodu platformy NodeMCU, dvou čidel DS18B20 a led diody na plošném spoji . . . . .	10
2.7	Fyzická realizace senzoru s moduly DS18B20 a DHT11 . . . . .	10
2.8	Schéma obvodu platformy NodeMCU, modulu DHT11 a led diody na plošném spoji . . . . .	11
2.9	Schéma obvodu platformy NodeMCU, modulu TSL2591 a led diody na plošném spoji . . . . .	12
2.10	Fyzická realizace senzoru s modulem TSL2591 . . . . .	13
2.11	Schéma obvodu platformy NodeMCU, modulu BME280 a led diody na plošném spoji . . . . .	13
2.12	Fyzická realizace senzoru s modulem BME280 . . . . .	14
2.13	Schéma obvodu platformy NodeMCU, modulu AM312, led diody a externího bzučáku na plošném spoji . . . . .	15
2.14	Fyzická realizace senzoru s modulem AM312 . . . . .	15
2.15	Schéma obvodu platformy NodeMCU, magnetického kon- taktu LS311B38 a led diody na plošném spoji . . . . .	16
2.16	Fyzická realizace senzoru s magnetickým kontaktem LS311B38	16
2.17	Jednočipový počítač Raspberry Pi 4 Model B . . . . .	17
3.1	Rozložení jednotlivých komponent a schéma datových toků v celém projektu . . . . .	20
3.2	Princip komunikace přes MQTT protokol . . . . .	21
3.3	Schéma procesů s popisy portů běžících na Raspberry Pi . .	25
3.4	Diagram architektury kódu backendu na Raspberry Pi . . .	25
4.1	Markovův řetězec popisující přechody mezi jednotlivými stavami senzorů . . . . .	28
4.2	Zobrazení ve webové vizualizaci v případě chyby senzoru . .	29
5.1	Stránka Overview ve webovém rozhraní . . . . .	32
5.2	Detail dlaždice . . . . .	33
5.3	Zobrazení všech ikon popisujících stavu senzoru . . . . .	33
5.4	Stránka Analytics se zobrazeným modelem pro vnitřní tep- lotu z čidla DS18B20_01 . . . . .	34
5.5	Proces výběru z předvoleb a specifikace zobrazení dat . . . .	35
5.6	Popis jednotlivých dlaždic na stránce Analytics . . . . .	36
5.7	Stránka About ve webovém rozhraní . . . . .	37

# Seznam tabulek

3.1	Atributy zprávy odesílané z mikročipu na broker . . . . .	23
3.2	Výčet všech topiců, do kterých jsou odesílány zprávy . . . . .	23

# Použité zkratky a termíny

<b>IoT</b>	Internet of Things
<b>RPi</b>	Raspberry Pi
<b>ESP8266</b>	Mikročip, mikrokontrolér
<b>NodeMCU</b>	Vývojová platforma s čipem ESP8266
<b>Čidlo</b>	Komponenta k mikročipu (např.: vlhkoměr DHT11)
<b>Senzor</b>	Celek skládající se z mikročipu a čidla (např.: obvod ESP8266 + DHT11)

# Kapitola 1

## Úvod

Přeložit do češtiny

Your intro... Thesis ref example: Bulín, 2016, Misc ref example: Šmídl, 2017, Article ref example: McCulloch a Pitts, 1943, Online webpage ref example: Bradley, 2006

### 1.1 State of the Art

### 1.2 Thesis Objectives

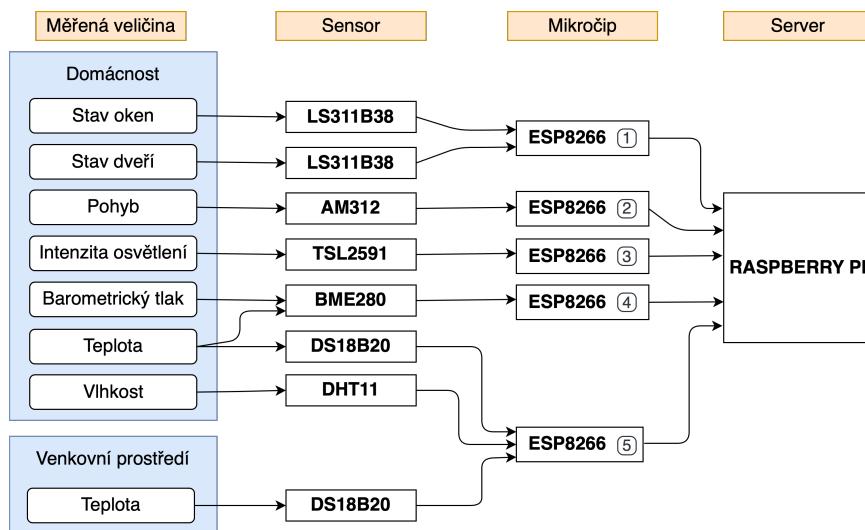
### 1.3 Thesis Outline

## Kapitola 2

# Hardware komponenty

V této kapitole je popsán veškerý použitý hardware v projektu senzorického řešení chytré domácnosti. V projektu je využito 5 mikročipů ESP8266, 8 čidel pro měření fyzikálních veličin a jeden počítač Raspberry Pi.

Měřené fyzikální veličiny lze rozdělit do dvou základních kategorií - veličiny, které jsou měřeny uvnitř domácnosti v rámci místnosti a veličiny ve venkovním prostředí. Mezi veličiny měřené uvnitř místnosti patří teplota, vlhkost, intenzita osvětlení, stav dveří a oken a barometrický tlak. V místnosti je dále monitorován pohyb osob. Ve venkovním prostředí je měřena teplota. Na Obr. 2.1 je zobrazen použitý hardware v závislosti na měřených veličinách.



OBRÁZEK 2.1: Vztahy mezi použitým hardwarem a měřenými veličinami

Ve snaze o co nejfektivnější využití hardwaru může jeden mikročip číst data z několika čidel. Tento přístup se využívá hlavně u čidel teploty a vlhkosti a u čidel pro monitorování stavu oken a dveří. U ostatních senzorů je zpravidla využitý jeden mikročip pro čtení dat z jednoho specifického čidla. Využitelnost jednoho mikročipu pro více čidel je závislá zejména na fyzickém umístění senzoru (např.: pohybový senzor musí být umístěn v rohu místnosti, kde není vhodné současně měřit jiné veličiny) a na výpočetní náročnosti. Jeden mikročip často zvládne načítat data pouze z jednoho čidla.

## 2.1 Mikročip ESP8266

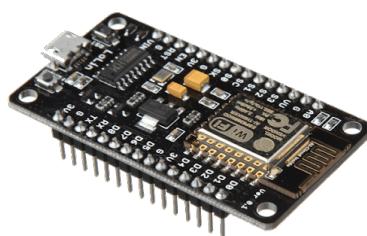
Základním stavebním prvkem senzorů v tomto projektu je mikročip *ESP8266*. Existuje celá řada mírně odlišných mikrokontrolérů založených na tomto modulu, v principu se ale jedná o velmi levný mikročip s procesorem o frekvenci 80 Hz, flash pamětí typicky od 512 KiB do 4 MiB, 16 vstupně-výstupními piny a podporou sběrnic SPI, I<sup>2</sup>C a I<sup>2</sup>S.

### Vlastnosti mikročipu

Hlavní předností tohoto čipu je podpora Wi-Fi standardu IEEE 802.11 b/g/n. K mikročipu stačí kabelově přivést pouze napájení a veškerá komunikace může probíhat bezdrátově po místní síti. Tím odpadá nutnost předem připravené kabeláže v domě a rozšířují se možnosti využití i v méně dostupných prostorech. Tento druh mikročipu byl zvolen hlavně kvůli kompatibilitě s celou řadou čidel, podpoře několika programovacích jazyků a v neposlední řadě kvůli široce rozšířené komunitě a dostupnosti nejrůznějších návodů pro DIY projekty.

### Vývojové platformy

Vývojové desky s mikročipem ESP8266 jsou vyráběny v několika provedeních. V projektu senzorického řešení chytré domácnosti jsem využil výhradně vývojovou platformu *NodeMCU* (na Obr. 2.2). Tato platforma je přizpůsobena pro pohodlnou komunikaci s mikročipem ESP8266 pomocí sériové linky. Platforma disponuje USB konektorem a vývody GPIO pinů. USB konektor slouží k napájení a zároveň k přenosu dat do paměti mikročipu. Pro zapojení externích komponent a vytváření obvodů lze pohodlně využít breadboard<sup>1</sup> a propojovací kably. Ve fázi vytváření prototypu tedy není nutné pájet.



OBRÁZEK 2.2: Vývojová platforma NodeMCU s modulem  
ESP8266

### Firmware mikročipu

Po zakoupení vývojové platformy je prvně potřeba nahrát aktuální firmware do flash paměti mikročipu. Před nahráním firmwaru je nutné nainstalovat ovladač pro komunikaci s USB rozhraním na desce. NodeMCU zpravidla

<sup>1</sup>Nepájivé kontaktní pole. Komponenta, která umožňuje sestavit prototyp obvodu a je vhodná pro první experimentování.

vyžaduje ovladač CP2102 (alternativně CH340). Pro nahrání samotného firmware do mikročipu jsem zvolil nástroj *esptool.py*. Esptool je program postavený na Pythonu, který umí načíst informace o připojeném mikročipu, vymazat flash paměť, nahrát nový firmware a případně zálohovat flash paměť.

## Vývojové prostředí

Veškerá komunikace s čipem probíhá po USB rozhraní a je založena na principu nahrání skriptů do paměti čipu a následném restartování pro spuštění programu. Jakkoliv změna kódu znamená vymazání původního skriptu z paměti čipu a následném nahrání nového skriptu. Tato skutečnost mírně ztěžuje ladění kódu a odchytávání chyb, ale samotný skript pro mikrokontrolér nebývá příliš dlouhý a k ladění kódu lze efektivně využít příkazovou řádku. Pro ukládání skriptů do mikročipu je potřeba vhodný nástroj pro přístup do adresářové části paměti na čipu. Jedním z těchto nástrojů je *Mpfshell*. Mpfshehell je balíček postavený na Pythonu, který umožňuje editaci souborů uložených v paměti (nahrávání, mazání, vytváření složek apod.) a zpřístupňuje příkazovou řádku na mikročipu. Právě příkazová řádka je zásadní pro rychlé spuštění kódu a odladění chyb, je možné v ní přímo psát části kódu nebo spouštět jednotlivé skripty bez nutnosti restartu.

## Programování mikročipu

Samotné ESP8266 může být programováno v několika jazycích, předně v Arduino IDE, MicroPythonu nebo LUA. V tomto projektu jsou všechny mikrokontroléry programovány v jazyce MicroPython. V rámci sjednocení programovacích jazyků v celém projektu jsem vybral MicroPython právě z důvodu univerzálnosti. MicroPython vychází z Pythonu 3 a je přímo přizpůsoben pro programování mikrokontrolérů. Další části projektu (například backend webové vizualizace nebo klasifikátor dat) jsou postaveny na Pythonu 3, proto se MicroPython v rámci zachování jednotné struktury jeví jako logická volba. Nespornou výhodou tohoto jazyku je velká podpora v rámci komunity a množství návodů a knihoven pro komunikaci s jednotlivými čidly.

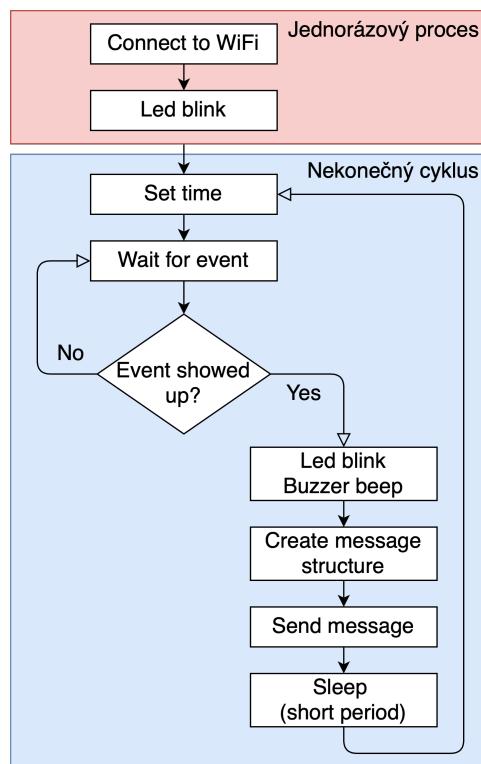
## Odesílání dat z mikročipu

V senzorech pro chytrou domácnost se uplatňují 2 základní přístupy odesílání dat na server. První je založen na odeslání zprávy na základě vzniku události a druhý přístup odesílá zprávy pravidelně s předem zvolenou periodou odesílání. V obou případech se po zapnutí senzoru mikročip nejprve připojí k Wi-Fi a problkne led dioadami. Tento proces proběhne jednorázově a pak senzor přejde do stavu, kdy je připraven načítat data ze senzorů a publikovat zprávy.

## Odesílání zpráv na základě vzniku události

Odesílání zpráv ze senzoru na server na základě vzniku události používá senzor pro detekci pohybu v místnosti a senzor pro monitorování stavu oken a dveří. Tento přístup je popsán v diagramu na Obr. 2.3. V nekonečném cyklu,

do kterého se senzor dostane po připojení k internetu, se prvně sesynchronizuje čas se světovým časem pomocí protokolu NTP<sup>2</sup>. Po synchronizaci senzor čeká, dokud se neobjeví událost, kterou má zaznamenat. Touto událostí je například otevření či zavření okna nebo dveří a nebo pohyb člověka v místnosti. V momentě, kdy událost nastane, čidlo problikne externí led diodou (v případě pohybového čidla se pro indikaci zapne ještě externí bzučák) a přejede do fáze vytváření zprávy. Po vytvoření struktury zprávy, která sestává z několika atributů (více v Kap. 3.1), se senzor připojí k MQTT brokeru a odešle zprávu. Po odeslání zprávy se uspí na 3 vteřiny a celý cyklus se opakuje. Uspání je v programu spíše jako preventivní opatření, aby se kód nežádaným způsobem nezacyklil. V případě pohybového senzoru je krátké uspání vhodné také pro to, aby čidlo neposílalo zbytečně moc zpráv, když v místnosti detekuje pohyb. Šetří se tím vytíženosť sítě a pro účel detekce pohybu není nutné odesílat informaci několikrát za vteřinu.



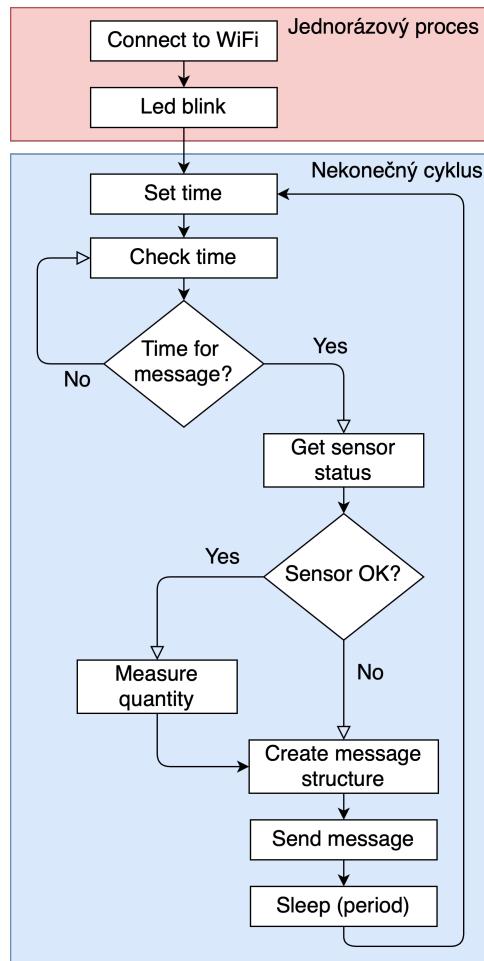
OBRÁZEK 2.3: Diagram odesílání zpráv na základě vzniku události

### Periodické odesílání zpráv

Pravidelné odesílání zpráv s předem danou frekvencí odesílání používají všechny ostatní senzory měřící veličiny, u kterých je vhodné měření po určité době opakovat. Jde o čidla vnitřní a venkovní teploty, vlhkosti, intenzity osvětlení a barometrického tlaku. U těchto senzorů je vhodné měření fyzičkálních veličin opakovat za účelem získání dostatečného množství dat pro následné trénování modelů a pro zajištění neustálé aktuálnosti hodnot ve webové vizualizaci. Vstupními parametry toho přístupu je perioda odesílání dat a určení okamžiku, ve kterém má proběhnout odeslání dat. V případě

<sup>2</sup>Network Time Protocol - protokol pro synchronizaci času po internetu

senzorů pro chytrou domácnost je perioda odesílání 1 minuta a okamžik, kdy se posílají data je vždy ve 30. vteřině (např.: po sobě jdoucí sekvence dat odeslaná v časech 14:25:30, 14:26:30, 14:27:30, ...). Minutová perioda odesílání dat je kompromis mezi dostatečným množstvím dat a vytížeností výpočetní techniky (vytíženost přenosové sítě, množství potřebné kapacity pro ukládání těchto dat, opotřebení čidel vlivem neustálého načítání dat apod.). Odesílání dat každou minutu vytvoří 1440 vzorků dat za 24 hodin. Po několika dnech sbírání dat už lze z tohoto množství dat počítat statistiky a trénovat klasifikátory. Specifická volba odesílání dat ve 30. vteřině je spíše návrhové rozhodnutí, které sjednocuje odesílání dat ze všech senzorů ve stejném okamžiku. Přístup periodického odesílání dat je popsán v diagramu na Obr. 2.4.



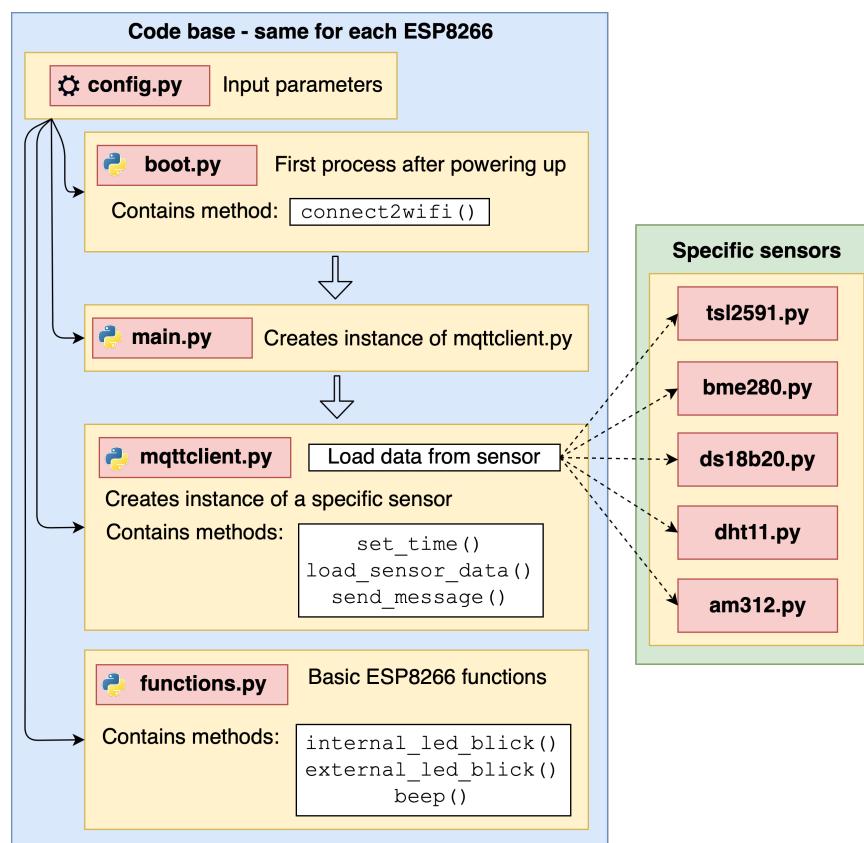
OBRÁZEK 2.4: Diagram periodicky opakovaného odesílání zpráv

V nekonečném cyklu, do kterého se senzor dostane po připojení k Wi-Fi, se nejdříve sesynchronizují vnitřní hodiny mikročipu se světovým časem pomocí protokolu NTP. Následně senzor čeká v cyklu na okamžik, kdy má odeslat zprávu (30. vteřina každé minuty). V momentě, kdy je podmínka splněna a je čas na zprávu, mikročip zjistí status senzoru a načte data (zjištění stavu čidla je přímo spojeno s načtením dat - pokud se podaří načíst data z čidla, stav čidla je "OK", pokud dojde k chybě při pokusu o načtení dat, stav čidla je "ERROR"). Více o detekci chyb na úrovni mikročipu v Kap. 4.1). Po

načtení dat z čidla se vytvoří struktura zprávy (více o struktuře zprávy v Kap. 3.1) a zpráva se odešle na MQTT broker. Po odeslání se senzor uspí na čas  $t - 10[\text{sec}]$ , kde  $t$  je perioda odesílání. Například pro periodu odesílání 1 minuta se senzor uspí na 50 vteřin. Zbylých 10 vteřin, kdy senzor nespí, je určeno pro proces načtení dat ze senzoru (např.: u teplotního senzoru, který načítá data z několika čidel tento proces trvá v průměru 3 až 4 vteřiny), vytvoření zprávy a odeslání zprávy.

### Architektura kódu na mikročipu

Architektura kódu všech senzorů sestává ze stejného jádra. Při návrhu programu pro mikročipy byl kláden důraz na objektovou strukturu programování a co nejuniverzálnější využití. Proto je základ programu pro všechny senzory stejný a liší se jen komunikací s konkrétním čidlem. Tímto přístupem se podařilo sjednotit verze kódu ve všech senzorech a hlavně umožnit rychlé přidání dalších senzorů v budoucnu. Pokud bych do chytré domácnosti chtěl přidat další senzor, který bude měřit jinou fyzikální veličinu, nemusím programovat celou logiku senzoru znova nebo složitě vytrhávat části jinde použitého kódu, ale použiji stávající program, který pouze doplním o implementaci komunikace s konkrétním čidlem. Vizuální pohled na architekturu kódu na mikročipu je na Obr. 2.5.



OBRÁZEK 2.5: Diagram architektury kódu na microchipu  
ESP8266

Struktura kódu pro mikročip ESP8266 se skládá z 6 souborů - *boot.py*,

*main.py*, *mqttclient.py*, *functions.py*, jednoho skriptu pro komunikaci s čidlem a jednoho konfiguračního souboru. Skripty *boot.py* a *main.py* jsou pevně dané programovacím jazykem MicroPython.

- *config.py*

V souboru *config.py* jsou nadefinovány vstupní parametry konkrétního senzoru. Jsou zde uloženy přístupové údaje pro připojení k Wi-Fi, parametry pro MQTT komunikaci a označení GPIO pinů podle reálného využití (mikročip a čidlo je napájené na plošném spoji a využití GPIO je pevně dané touto fyzickou realizací). Všechny parametry jsou uloženy v tomto konfiguračním souboru a zbytek kódu pouze odkazuje na tyto parametry.

- *boot.py*

Skript *boot.py* se po zapnutí mikrokontroléru spustí vždy jako první nezávisle na jakémkoliv jiném souboru. V tomto skriptu dochází k připojení k místní Wi-Fi síti. Po úspěšném připojení k síti senzor čtyřikrát zabliká interní led a následně i externí led diodou pro vizuální kontrolu.

- *main.py*

Druhým skriptem, který naběhne ihned po *boot.py* je *main.py*. Toto pořadí spuštění skriptů je pevně stanovené programovacím jazykem. Ve skriptu *main.py* se vytvoří instance třídy `MQTTClient`.

- *mqttclient.py*

Tato třída je v podstatě jádrem celého programu pro mikročip. Dochází zde k synchronizaci vnitřních hodin se světovým časem, načítání dat z čidla, vytváření struktury zprávy a publikování zprávy. Ve třídě `MQTTClient` se vytvoří instance třídy konkrétního čidla, která obstarává veškerou komunikaci s čidlem. Třída `MQTTClient` je tedy pro všechny mikročipy stejná, jednotlivé mikročipy se liší jen třídou pro komunikaci s konkrétním čidlem.

- *functions.py*

Soubor *functions.py* obsahuje obecné funkce mikročipu ESP8266. Umožňuje zapnutí interní a externí led diody a zapnutí externího bzučáku. Funkce jsou napsané univerzálně a načítají vstupní parametry ze souboru *config.py*. Z konfiguračního souboru se načítá například číslo GPIO pinu externí led diody - každý senzor může mít externí led diodu umístěnou na jiném GPIO pinu, kvůli využití místa na plošném spoji, ale metoda pro rozsvícení diody je vždy stejná.

Soubory *tsl2591.py*, *bme280.py*, *ds18b20.py*, *dht11.py* a *am312.py* jsou skripty pro komunikaci s konkrétním čidly a mikročip obsahuje vždy jen ten komunikační skript, který pro obsluhu svého čidla potřebuje.

Využitím objektově orientovaného programování a implementací konfiguračních souborů bylo dosaženo maximální univerzálnosti a přehlednosti kódu skrze všechny senzory v chytré domácnosti. Hlavní výhodou tohoto přístupu je snadná rozšiřitelnost kódu v budoucnu a díky udržení jedné vývojové větve

je na každém mikročipu možné snadno aktualizovat kód. Díky konfiguračnímu souboru je možné měnit vstupní parametry bez nutnosti zásahu do samotného kódu.

## 2.2 Senzory

V projektu chytré domácnosti bylo použito celkem 7 typů čidel pro měření fyzikálních veličin. Mezi tyto čidla patří teplotní čidlo *DS18B20*, vlhkoměr *DHT11*, čidlo intenzity osvětlení *TSL2591*, čidlo barometrického tlaku a teploty *BME280*, pohybové čidlo *AM312* a magnetické čidlo pro monitorování stavu oken a dveří *LS311B38*. Tyto fyzikální veličiny mají z pohledu chytré domácnosti význam pro automatizaci, zajištění bezpečnosti objektu a přehled o podmírkách uvnitř domu a okolí. Data z těchto senzorů jsou následně využita pro trénovaní modelů, klasifikaci jednotlivých hodnot a pro predikci měřených hodnot. Konkrétní senzory byly vybrány na základě kompromisu mezi cenou, funkčností a přesností měření.

### Postup fyzické realizace čidel

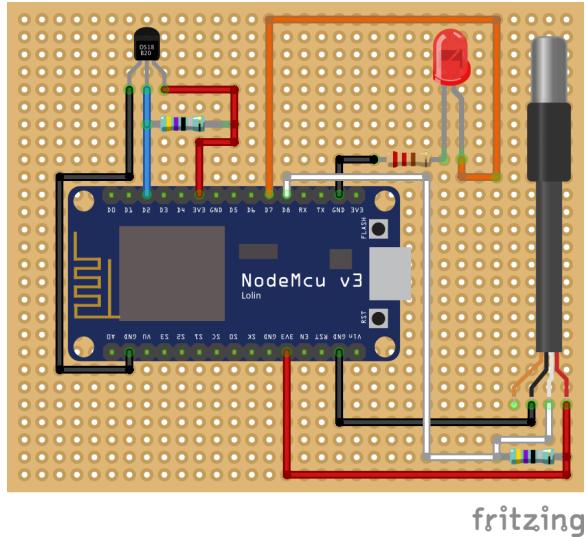
Prvním krokem při návrhu chytrých čidel je stanovení fyzikálních veličin, které chceme sledovat a otestovat spolehlivost čidel pro měření těchto veličin. Po vybrání konkrétních senzorů následuje sestrojení prototypu - zapojení obvodu na breadboardu. Po vytvoření funkčního obvodu může začít experimentování a ladění kódu pro načítání dat z čidla. Po úspěšném naprogramování komunikace s čidlem jsem přešel k vytvoření pevného fyzického obvodu napájením jednotlivých součástí na plošný spoj. Tento plošný spoj má nespornou výhodu ve své životnosti a odolnosti, na rozdíl od obvodu na breadboardu, kde se kabely mohou vlivem neopatrnosti vypojovat. Ke každému senzoru jsem přidal červenou externí led diodu pro indikaci různých změn a stavů. Uživatel v podstatě nemá žádný jiný způsob vizuální kontroly funkčnosti senzoru. Led diodu využívám jako signalizaci při úspěšném připojení k Wi-Fi a jako signalizaci odeslání zprávy. Vždy, když senzor odešle zprávu, dioda jednou blikne.

#### 2.2.1 Teplotní čidlo DS18B20

Teplotní čidlo Dallas DS18B20 se vyrábí ve více provedení, v projektu chytré domácnosti jsou použity 2 čidla, jedno v klasickém provedení pro měření vnitřní teploty a jedno voděodolné pro měření venkovní teploty. Rozsah měření tohoto čidla je od  $-55^{\circ}\text{C}$  do  $+125^{\circ}\text{C}$  s přesností  $\pm 0,5^{\circ}\text{C}$  v mezích od  $-10^{\circ}\text{C}$  do  $+85^{\circ}\text{C}$ . Doba nutná pro konverzi 12 bitové hodnoty teploty do digitálního čísla je 750 ms. Datová komunikace probíhá přes rozhraní One-Wire a pro přenos dat je využitý jeden pin. Každé čidlo DS18B20 má svoji unikátní 64 bitovou sériovou adresu. Díky tomu je umožněno vytvořit One-Wire obvod s několika čidly, které jsou připojeny k jednomu digitálnímu GPIO pinu a pomocí toho sériového čísla lze čidlo snadno identifikovat při programování komunikace čidla s mikročipem.

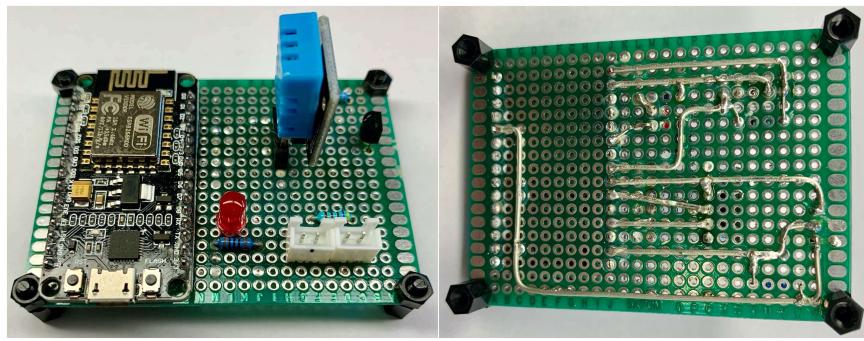
Čidlo je k mikročipu připojeno třemi dráty - jeden drát pro napájení, jeden pro uzemnění a jeden pro datovou komunikaci. Napájecí drát je na ESP8266 připájen k pinu 3,3V, zemnící drát ke GND a datová linka je připájena k digitálnímu pinu D4. Do obvodu je potřeba připojit rezistor s odporem

$4,7k\ \Omega$ , který spojuje datový a napájecí drát. Schéma zapojení obou čidel DS18B20 je zobrazeno na Obr. 2.6.



OBRÁZEK 2.6: Schéma obvodu platformy NodeMCU, dvou čidel DS18B20 a led diody na plošném spoji

Čidlo DS18B20 se osvědčilo přesností měření a důvěryhodností naměřených hodnot. Při prvotním testování jsem zapojil najednou více těchto čidel blízko sebe a naměřené hodnoty ze všech čidel se lišily v rámci desetin stupně. V projektu chytré domácnosti jsou hodnoty teplot zaokrouhlovány na 2 desetinná místa, tudíž přesnost v rámci desetin dostačuje a odchyly jsou zanedbatelné. V tomto projektu jsou teplotní čidla DS18B20 a vlhkostní snímač DHT11 z důvodu efektivního využití hardwaru a stejněmu umístění v místnosti připojeny k jednomu mikročipu ESP8266. Senzor skládající se z jedné platformy NodeMCU, jednoho vnitřního čidla DS18B20, jednoho venkovního čidla DS18B20, jednoho vlhkoměru DHT11 a externí led diody je umístěn v rohu místnosti. Venkovní čidlo má 1 m dlouhý kabel a je umístěno za oknem. Na Obr. 2.7 je vyfocena fyzická realizace zkonstruovaného senzoru.



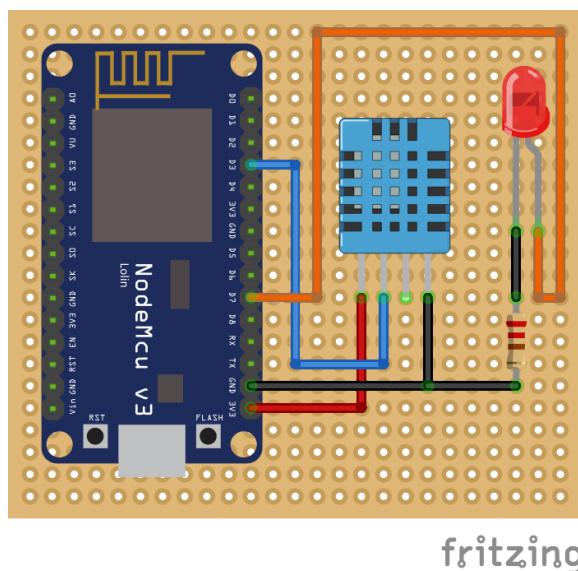
OBRÁZEK 2.7: Fyzická realizace senzoru s moduly DS18B20 a DHT11

## 2.2.2 Vlhkoměr DHT11

Modul DHT11 je určen pro měření vlhkosti a teploty uvnitř místnosti. Měřit teplotu umožňuje v rozmezí od  $0\ ^\circ\text{C}$  do  $+50\ ^\circ\text{C}$  s přesností  $\pm 2,0\ ^\circ\text{C}$ .

Z důvodu menšího rozsahu měření a hlavně nižší přesnosti je toto čidlo v projektu chytré domácnosti používáno pouze pro měření vlhkosti. Vnitřní vlhkost je měřena v rozmezí 20 % až 90 % s přesností  $\pm 5\%$ . Chyba měření vlhkosti také není zcela zanedbatelná, nicméně pro základní představu o vlhkosti v místnosti postačuje. Často nezáleží na konkrétní hodnotě vlhkosti, ale spíše na změně této hodnoty v průběhu dne. Pokud je například celý den domácnost bez přítomnosti lidí, drží se vlhkost celý den na stejně hodnotě. Po příchodu člověka do místnosti se hodnota vlhkosti zvýší (sice mírně v rámci jednotek procent, ale s touto informací už je možné dále pracovat).

Čidlo DHT11 je k mikročipu připojeno třemi dráty - jeden drát pro napájení, jeden pro uzemnění a jeden pro přenos dat. Modul vyžaduje napětí v rozmezí 3,0 V až 5,5 V. Po zavedení napětí k modulu je před měřením vyžadováno minimálně 1 vteřinu počkat na kalibraci čidla. Tuto prodlevu je nutné brát v úvahu při každém načítání dat z čidla. Ze zkušenosti s komunikací s čidlem DHT11 raději používám 2 vteřinovou pauzu, při kratším intervalu se často stávalo, že čidlo nestihlo odpovědět a kód spadl do chyby. Metoda pro čtení hodnot z čidla DHT11 vrací hodnotu v procentech zaokrouhlenou na celá čísla. Schéma zapojení modulu DHT11 a vývojové platformy NodeMCU je zobrazeno na Obr. 2.8.



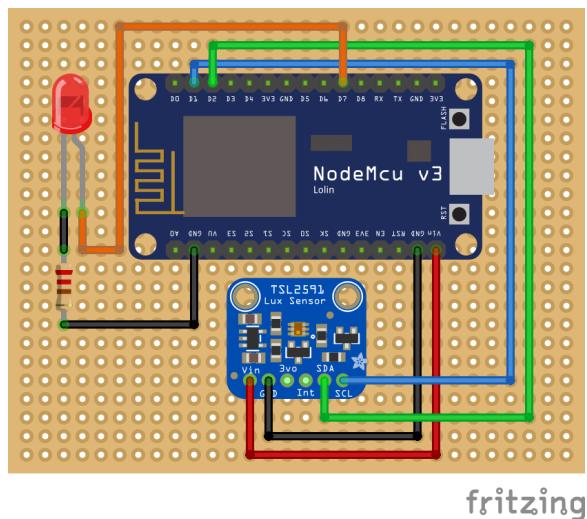
OBRÁZEK 2.8: Schéma obvodu platformy NodeMCU, modulu DHT11 a led diody na plošném spoji

Fotografie fyzické realizace senzoru využívající modul DHT11 je na Obr. 2.7. Modul DHT11 není úmyslně připájen přímo k plošnému spoji, ale je připojen přes konektor na desce. Během několika měsíců testovacího provozu přestalo fungovat jedno čidlo DHT11 a muselo být nahrazeno za jiné. Díky připojení přes dutinkovou lištu byla výměna modulu záležitostí několika vteřin. Tato výměna senzoru probíhá za provozu bez nutnosti restartu senzoru, senzor pomocí automatické detekce chyb pozná nefunkční čidlo a po výměně za nový kus sám naběhne do režimu čtení hodnot z modulu (více v Kap. 4.1).

### 2.2.3 Čidlo intenzity osvětlení TSL2591

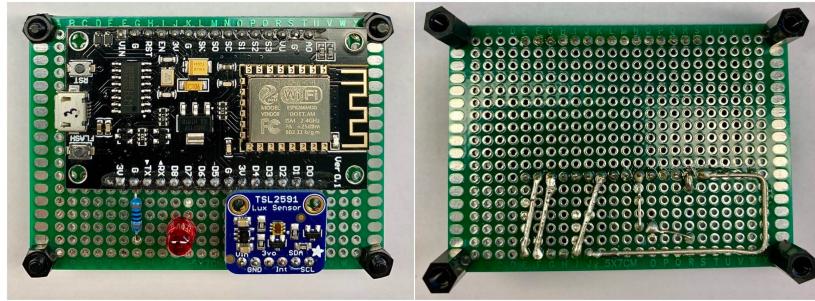
TSL2591 je čidlo sloužící pro měření intenzity osvětlení, které konvertuje intenzitu světla do digitálního výstupu přenášeného sběrnicí I<sup>2</sup>C. Výstupem modulu je hodnota intenzity osvětlení v jednotce lux s přesností na 4 desetinná místa. Toto čidlo je jedno z nejdražších v projektu chytré domácnosti, ale nabízí víceméně dokonalé rozpoznání intenzity osvětlení od 188  $\mu$ Lux do 88 000 Lux. Hodnota s přesností na 4 desetinná místa je zaokrouhlována na celá čísla, protože větší přesnost není vyžadována. Intenzita osvětlení v místnosti během dne se mění řádově o stovky luxů, v noci je osvětlení blízké 0 lux, přes den v rozmezí přibližně 30 až 500 lux a na přímém slunci je to několik desítek tisíc lux. Modul garantuje přesnost měření v teplotních podmínkách od -30 °C do +80 °C. Rozsah napájení desky s čidlem TSL2591 je od 3,3 V do 5,0 V. Velkou předností tohoto čidla je přítomnost diod pro měření infračerveného, celospektrálního a viditelného světla.

Každý modul TSL2591 má svou unikátní 7 bitovou adresu a komunikace s mikročipem ESP8266 probíhá přes I<sup>2</sup>C sběrnicí. K čidlu je přivedeno napájení z 3,3 V výstupu na mikročipu, dále je uzemněno GND drátem a I<sup>2</sup>C komunikaci (výstupy SDA a SCL) s mikrokontrolérem zajišťují digitální GPIO piny D1 a D2. Schéma zapojení čidla TSL2591, platformy NodeMCU a externí led diody je zobrazeno na Obr. 2.9.



OBRÁZEK 2.9: Schéma obvodu platformy NodeMCU, modulu TSL2591 a led diody na plošném spoji

Modul TSL2591 se během testování v projektu chytré domácnosti osvědčil pro svou kvalitu zpracování, přesnost naměřených hodnot, neobvykle velký rozsah měření a spolehlivost. Datová komunikace s tímto čidlem je výpočetně náročná, proto je použitý jeden samostatný mikročip ESP8266 pro obsluhu jen tohoto modulu. Fotografie zkonstruovaného senzoru je na Obr. 2.10.



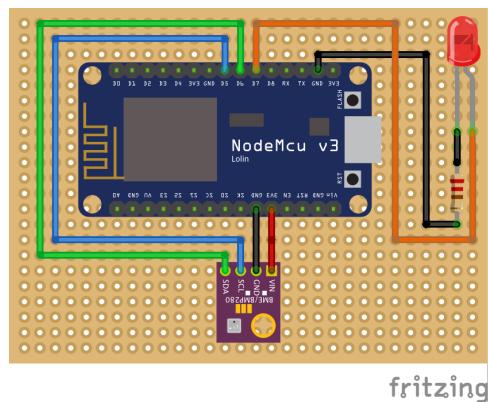
OBRÁZEK 2.10: Fyzická realizace senzoru s modulem TSL2591

#### 2.2.4 Čidlo barometrického tlaku a teploty BME280

Modul BME280 od výrobce BOSH slouží k měření vnitřní teploty a barometrického tlaku. Čidlo je schopno měřit okolní teplotu v rozmezí  $-40^{\circ}\text{C}$  do  $+85^{\circ}\text{C}$  a barometrický tlak v rozsahu 300 hPa až 1100 hPa. Hodnoty teploty jsou měřeny s přesností  $\pm 1^{\circ}\text{C}$  a hodnoty barometrického tlaku s přesností  $\pm 1 \text{ Pa}$ . Modul vrací zkonzervovanou hodnotu obou veličin s rozlišením na 2 desetinná místa.

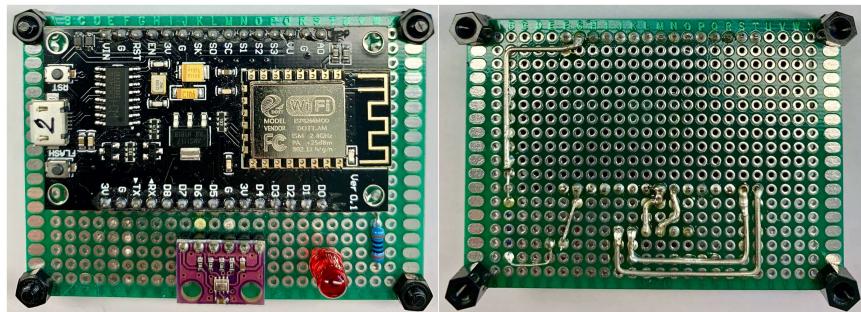
Při porovnání naměřených hodnot teploty z čidla BME280 a z DS18B20 jsem zjistil, že hodnoty se liší v průměru o méně než  $0,5^{\circ}\text{C}$ . Tento rozdíl je pro účely chytré domácnosti zanedbatelný a lze tedy předpokládat, že hodnoty teploty v místnosti jsou velmi přesné a čidlo DS18B20 pro běžné použití dostačuje, přestože je několikanásobně levnější, než modul BME280. Hodnoty barometrického tlaku je vhodné monitorovat s přesností na 2 desetinná místa, protože tato veličina se v místnosti v průběhu dne mění jen v minimálních mezích, za několik měsíců měření to průměrně bylo od 970 hPa do 980 hPa.

Komunikace s mikrokontrolérem probíhá po I<sup>2</sup>C sběrnicí. Reakční doba modulu od požadavku na změření hodnoty k odeslání hodnoty činí 1 vteřinu. S tímto zpožděním je opět nutno počítat při každém načítání hodnot. Kvůli výpočetní náročnosti při komunikaci s modulem a velikosti knihovny ovlaďovače čidla je v tomto projektu využitý jeden samostatný mikročip ESP8266, který obsluhuje pouze toto čidlo. Schéma zapojení čidla BME280, platformy NodeMCU a externí led diody je zobrazeno na Obr. 2.11.



OBRÁZEK 2.11: Schéma obvodu platformy NodeMCU, modulu BME280 a led diody na plošném spoji

Modul je k mikročipu ESP8266 připojen čtyřmi dráty - na výstup 3,3V pro napájení, GND pro uzemnění a dva digitální piny D5 a D6, které obstarávají přenos dat po sběrnici. Rozsah napájení desky s modulem BME280 je od 1,8 do 3,6 V. Senzor je umístěn na stole v rohu místnosti. Fotografie fyzické realizace senzoru je na Obr. 2.12.



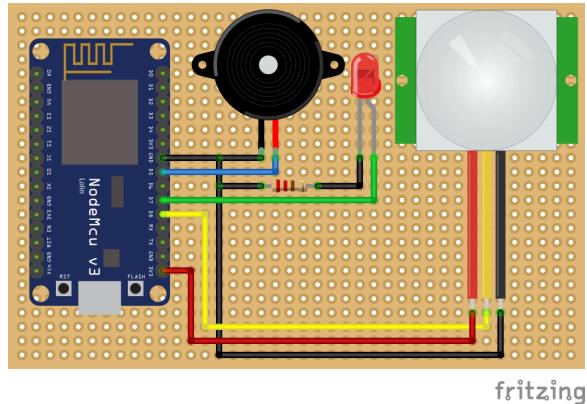
OBRÁZEK 2.12: Fyzická realizace senzoru s modulem BME280

### 2.2.5 Pohybové čidlo AM312

Modul pyroelektrické detekce pohybu AM312 slouží k monitorování pohybu osob v místnosti. Výstupem čidla je logická 1 v případě detekce pohybu a logická 0, pokud v místnosti není detekován pohyb. Jestliže v místnosti není detekován pohyb, čidlo neposílá žádnou informaci, výstupem na datovém pinu je trvale 0 a senzor čeká na vznik události (událost je v tomto případě detekce pohybu, více v Kap. 2.1). Ve chvíli, kdy je detekován pohyb, dochází ke vzniku události a výstupem čidla je 1. Tento modul je rozměrově velmi kompaktní a pohyb detekuje ve vzdálenosti do 5 metrů. Úhel detekce pohybu je do 100 stupňů. V případě místností v chytré domácnosti je tato detekční vzdálenost a úhel dostačující.

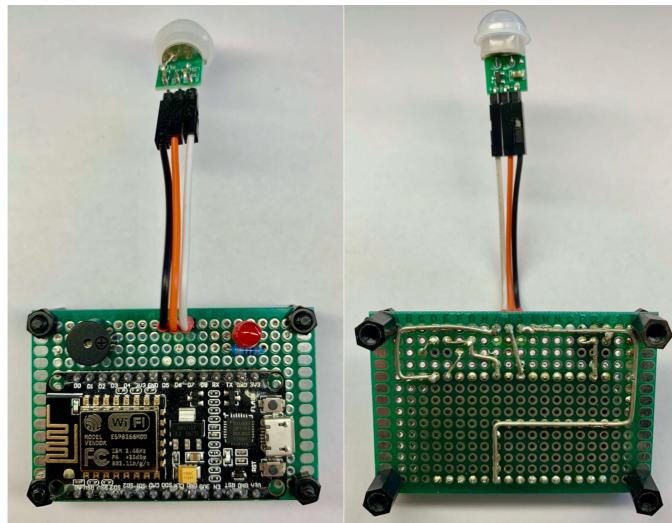
Modul AM312 se během několikaměsíčního provozu osvědčil pro svou spolehlivost. Pohyb byl detekován vždy, když jsem vstoupil do místnosti a zároveň modul neposílal falešné zprávy o pohybu v případě, že v domácnosti nikdo nebyl. Při experimentování s pohybovými čidly jsem testoval ještě modul HC-SR501, který má lepší specifikaci ve smyslu větší detekční vzdálenosti 7 m a úhlu do 120 stupňů, nicméně s tímto modulem byl problém právě s falešnými detekcemi.

Čidlo AM312 je konstruováno pro vstupní napětí od 2,7 do 12 V a pro prostřední s okolní teplotou od -20 °C do +60 °C. K mikrokontroléru ESP8266 je připojeno třemi dráty - jeden k vývodu 3,3 V pro napájení, jeden k GND výstupu pro uzemnění a jeden k digitálnímu pinu D8 pro přenos informace. Na plošném spoji tohoto senzoru je navíc přidán externí bzučák, který má využití jako alarm v případě detekce pohybu. Tento zvukový signál se hodil spíše pro prvotní ladění a testování na falešné poplachy, po odladění se jako vizuální signalizace detekce pohybu využívala externí led dioda, která se při zaznamenání pohybu rozsvítí na dobu 2 vteřin. Touto indikací jsem si ověřoval správnou funkčnost senzoru. Schéma zapojení pohybového čidla AM312, platformy NodeMCU, externí led diody a externího bzučáku na plošném spoji je na Obr. 2.13.



OBRÁZEK 2.13: Schéma obvodu platformy NodeMCU, modulu AM312, led diody a externího bzučáku na plošném spoji

Za účelem efektivní detekce je senzor umístěn v rohu místnosti naproti dveřím. Toto specifické umístění nedovoluje využití mikročipu ESP8266 současně s jiným senzorem, ale v tomto případě by to ani nebylo žádáné. Pohybový senzor je jedním ze dvou senzorů, které odesílají zprávy na základě vzniku události (více v Kap. 2.1). Tento přístup nelze efektivně kombinovat s periodickým odesíláním další veličiny. Fotografie fyzické realizace senzoru je na Obr. 2.14.



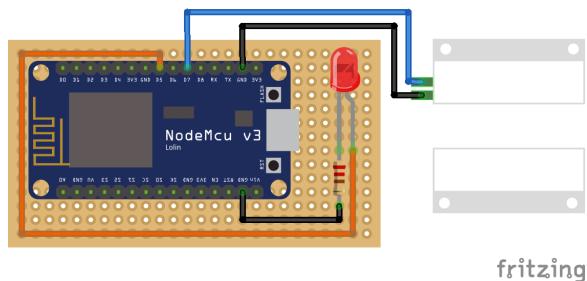
OBRÁZEK 2.14: Fyzická realizace senzoru s modulem AM312

### 2.2.6 Magnetické čidlo LS311B38

Magnetický jazýčkový kontakt LS311B38 slouží k detekci otevřeného okna nebo otevřených dveří. Toto čidlo je jedním z nejjednodušších čidel v projektu chytré domácnosti. Jedná se o 2 kusy magnetu, jedna část je umístěna na okně nebo dveřích a druhá část je připevněna k okennímu nebo nebo dveřnímu rámu. Magnety musejí být umístěny tak, že když je okno zavřené, oba kusy magnetu jsou blízko u sebe. Výstupem tohoto čidla je logická 1 v případě, že jsou magnety blízko u sebe (zavřené okno nebo zavřené dveře) nebo logická 0 v případě, kdy jsou magnetické jazýčky rozpojeny. V projektu chytré domácnosti je monitorován stav jednoho okna a jedněch dveří.

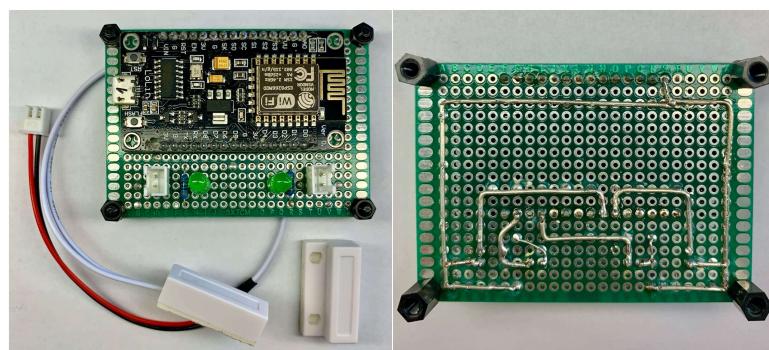
Pro tuto realizaci je využitý jeden mikrokontrolér ESP8266, který obsluhuje oba magnetické kontakty. Na plošném spoji jsou dvě externí led diody, jedna slouží pro vizuální indikaci otevřených dveří, druhá pro indikaci otevřeného okna.

Spínací vzdálenost magnetů je od 15 mm do 25 mm. Princip zapojení magnetických jazýčku k mikročipu ESP8266 je velmi podobný zapojení tlačítka. Z magnetu vedou dva kabely, které jsou připájeny k výstupům GND pro uzemnění a digitálnímu pinu D7. Oba kabely vedou jen z jednoho kusu magnetu a mikročip čeká na přiblížení druhého magnetu, čímž se obvod spojí a signál je přes digitální pin přenesen do mikročipu. Schéma zapojení magnetického kontaktu LS311B38, platformy NodeMCU a externí led diody je zobrazen na Obr. 2.15.



OBRÁZEK 2.15: Schéma obvodu platformy NodeMCU, magnetického kontaktu LS311B38 a led diody na plošném spoji

Vzhledem k jednoduchosti čidla lze předpokládat velmi dlouhou životnost a spolehlivost. Fotografie fyzické realizace senzoru je na Obr. 2.16.



OBRÁZEK 2.16: Fyzická realizace senzoru s magnetickým kontaktem LS311B38

## 2.3 Raspberry Pi

*Raspberry Pi* je jednočipový počítač s operačním systémem Raspbian. Raspbian OS je založený na Linuxu a po instalaci nabízí podporu několika programovacích jazyků a je přímo přizpůsoben na DIY projekty. Základní deska umožňuje připojit externí monitor pomocí microHDMI a externí komponenty jako je klávesnice a myš přes USB. K místní síti se připojuje klasickým LAN kabelem s koncovkou RJ45 nebo bezdrátově přes Wi-Fi. Oproti mikrokontrolérům jako jsou čipy ESP8266 nebo Arduino nabízí kromě ovládání

příslušenství pomocí GPIO kontaktů hlavně možnost programovat samotné aplikace. V projektu chytré domácnosti byla použita v současné době nejvýkonnější verze počítače Raspberry Pi 4 - Model B s 4 GB RAM, který byl uveden do prodeje v červnu 2019. Tento model je osazen 1,5 GHz čtyřjádrovým procesorem ARM Cortex-A72 a jako externí monitor lze použít displej s rozlišením 4K při 60 snímcích za vteřinu. Operační systém je nainstalován na externí microSD kartě s kapacitou 32 GB. Tato nejvýkonnější konfigurace byla zvolena z důvodu vytíženosti RPi v projektu chytré domácnosti.



OBRÁZEK 2.17: Jednočipový počítač Raspberry Pi 4 Model B

### Využití v chytré domácnosti

Počítač Raspberry Pi má v projektu chytré domácnosti několik využití.

- MQTT Broker
- Databázový server
- Trénování modelů a klasifikace
- Web server

Primárně slouží jako MQTT broker, který přijímá všechny příchozí zprávy ze senzorů (více v Kap. 3.1). Dále na RPi běží databázový server za účelem ukládání všech příchozích zpráv do databáze (více v Kap. 3.2). Současně jsou na RPi trénovány modely, na základě kterých je prováděna klasifikace jednotlivých příchozích zpráv (více v Kap. 4.2). Datově náročnější modely lze trénovat na externích výkonnějších počítačích a následně na RPi použít pro klasifikaci již natrénované modely. Za účelem grafické vizualizace naměřených dat a aktuálních hodnot veličin běží na RPi webový server (více v Kap. 3.3).

Velkou výhodou RPi jsou minimální rozměry a hlavně malý odběr proudu. V těchto aplikacích je nutné, aby počítač byl neustále zapnutý. Klasické PC by se k těmto účelům nehodilo kvůli vysoké spotřebě a nepotřebně velkému výkonu. Náklady na provoz a pořízení počítače RPi jsou ve srovnání s klasickém PC serverem několikanásobně nižší.

Raspberry Pi 4 Model B má i v tomto případě, kdy na RPi probíhá několik procesů paralelně vedle sebe, dostatečný výkon. Doba počítání modelů a klasifikace zpráv je v rámci vteřin, narozdíl od starších modelů RPi. Nevýhodou tohoto přístupu, kdy jeden server slouží k několika účelům, je nebezpečí pádu celého systému, pokud server přestane pracovat. V případě, že

z nějakého důvodu přestane pracovat RPi, přestanou se do databáze ukládat všechny příchozí zprávy, nově příchozí zprávy tudíž nebudou ani klasifikovány a webová vizualizace bude nedostupná. V případě nedostupnosti webové vizualizace pravděpodobně nejde o velký problém, ale pokud nejsou příchozí zprávy ukládány do databáze delší dobu, jde o závažnější problém a výsledné modely mohou být zkresleny. Tuto nevýhodu je vhodné vyřešit rozdělením procesů na více strojů - na více RPi. V této práci byla všechna data zálohována do cloutu.

## Kapitola 3

# Síťová komunikace a databáze

Smyslem zkonstruovaných čidel je jejich snadná implementace do bytu či domu. Senzory potřebují kabelově připojit jen napájení a celá komunikace mezi mikročipy a brokerem probíhá po WiFi. Tato schopnost bezdrátové komunikace zásadně rozšiřuje možnosti rozmístění senzorů po domácnosti. Komunikace senzorů se serverem využívá principů *IoT* a síťový protokol *MQTT*.

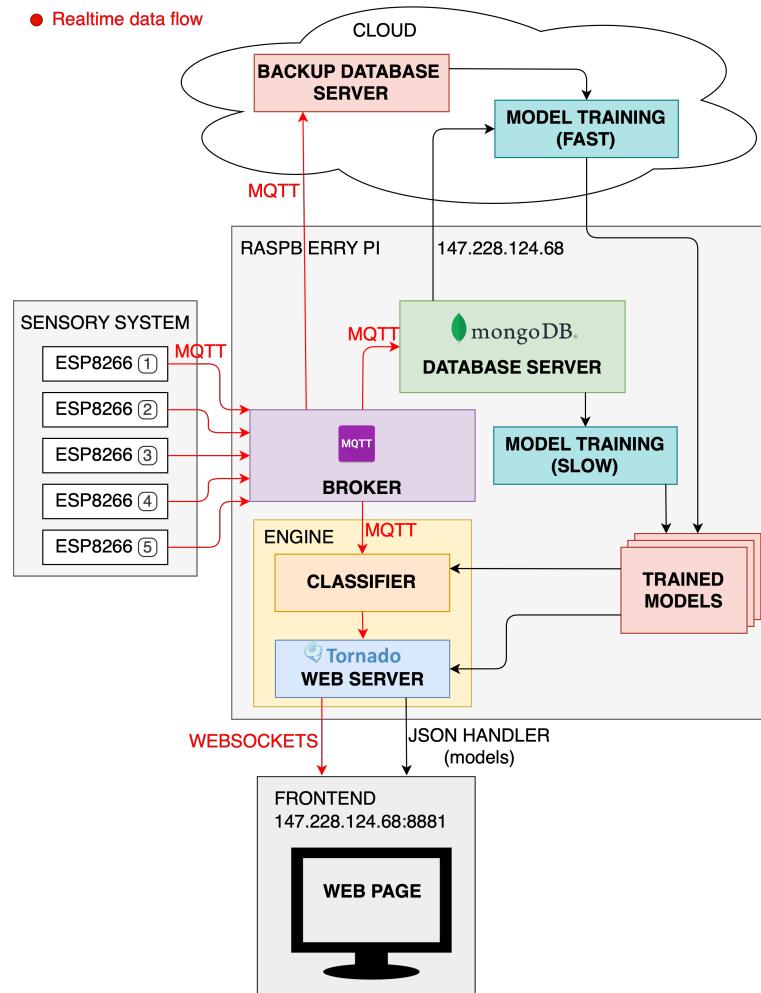
### Síť IoT

Síť IoT<sup>1</sup> je architektura na bázi internetu, která obecně slouží ke komunikaci a výměně dat. Senzory v projektu chytré domácnosti spadají do komponent v internetu věcí, protože veškerá komunikace probíhá po internetu a výhradně bez účasti uživatele. Technologie pro chytrou domácnost v tomto projektu byla navržena tak, aby uživatel měl přísun aktuálních dat ze senzorů bez nutnosti znalosti principů fungování celého systému.

Na Obr. 3.1 se nachází diagram, na kterém je zobrazen přehled všech komponent, jejich propojení a datové toky v celém projektu. Červenými šipkami jsou znázorněny datové toky, které probíhají v reálném čase. V cloudové části je zálohována databáze a jelikož je cloudový počítač výkonnější než Raspberry Pi, může být využitý k trénování modelů, které jsou pak jednorázově při načtení webu nahrány do RPi. Samotné modely mohou být trénovány i přímo na RPi, ale proces trénování modelů všech měřených veličin trvá déle. Komunikace mezi RPi a frontendem webu je popsána v Kap. 3.3.

---

<sup>1</sup>Internet of Things - internet věcí



OBRÁZEK 3.1: Rozložení jednotlivých komponent a schéma datových toků v celém projektu

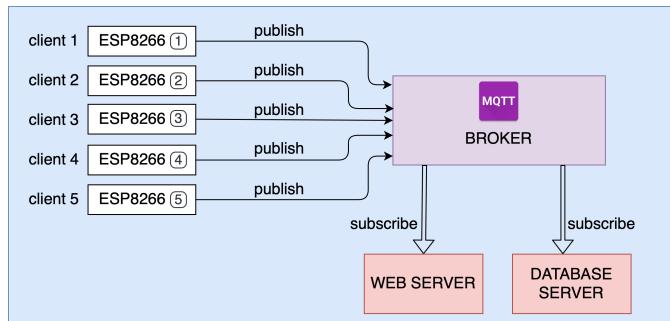
### 3.1 Protokol MQTT

MQTT<sup>2</sup> je internetový protokol, který slouží k výměně zpráv mezi subjekty v síti. Tento protokol vznikl už v roce 1999, ale k významnému využití dochází i v dnešní době díky IoT aplikacím. MQTT pracuje na TCP/IP vrstvě a hodí se především v aplikacích, které vyžadují minimální datový přenos po síti.

Při komunikaci prostřednictvím MQTT protokolu jsou v síti definovány dva typy entit - server a klient. V síti se nachází jeden server - *broker* a libovoľné množství klientů. Veškerá komunikace probíhá prostřednictvím brokeru. Broker je centrální místo v síti, na které ostatní zařízení publikují zprávy a který ostatním subjektům v síti umožňuje číst tyto zprávy. Klientem může být jakékoli zařízení, které má přístup na internet a podporu MQTT protokolu. Zpravidla jde o chytré senzory, které publikují zprávy. V tomto projektu je broker spuštěn na počítači Raspberry Pi (více v Kap. 2.3) a jako klienti figurují jednotlivé senzory - mikročipy ESP8266. Princip protokolu MQTT je zobrazen na Obr. 3.2. Jednotlivé senzory publikují zprávy na broker (mód *publish*) a broker přijímá všechny příchozí zprávy. Další

<sup>2</sup>Message Queuing Telemetry Transport

aplikace jako je webový a databázový server odebírají příchozí zprávy od brokeru (mód *subscribe*) za účelem dalšího zpracování dat. Z hlediska protokolu může být jeden klient současně *publisher* i *subscriber*, ale často bývají tyto role rozděleny. *Publisher* je zpravidla senzor, který měří určitou fyzikální veličinu a hodnoty odesílá na broker. *Subscriber* je většinou zařízení, které čte data zaslaná od *publisher*a s těmito daty pak dále pracuje (například je zobrazuje ve webovém rozhraní nebo ukládá do databáze).



OBRÁZEK 3.2: Princip komunikace přes MQTT protokol

### Bezpečnost přenosu dat

Identifikace klienta probíhá prostřednictvím uživatelského jména a hesla. Přes protokol MQTT se přenášejí textové řetězce v kódování UTF-8, které ve výchozím stavu bez použití SSL<sup>3</sup> nejsou nijak šifrované. Na TCP portu 1883 používá protokol nešifrovanou komunikaci, která se hodí pro přenos necitlivých dat. Alternativně lze na přednastaveném portu 8883 použít přenos šifrovaný protokolem SSL. Jelikož v projektu chytré domácnosti jsou senzory i broker na místní domácí síti a nepřenášejí se citlivá data, komunikace není šifrována.

#### 3.1.1 Struktura zpráv

Pomocí protokolu MQTT je možné odesílat libovolné formáty dat s omezením na maximální velikost jedné zprávy 256 MB. Senzory odesírají na broker zprávy v datovém formátu *JSON*<sup>4</sup>. Tento objektový formát umožňuje vytvářet hierarchické struktury zpráv ve formě řetězce znaků. JSON byl zvolen pro svou jednoduchost a přehlednost. Formát dat je pro člověka čitelný a výsledná zpráva je datově úsporná, tudíž vhodná pro pravidelné přenášení po síti bez zbytečného zvětšování datového toku. Níže je uvedena obecná struktura zprávy ve formátu JSON a jeden konkrétní příklad zprávy ze senzoru s vlhkostním čidlem.

```
{
  "sensor_id": string
  "location": string
  "owner": string
  "status": string
  "quantity": string
  "value": float
  "timestamp": string }
```

<sup>3</sup>Secure Sockets Layer - šifrovací protokol mezi transportní a aplikaci vrstvou

<sup>4</sup>JavaScript Object Notation - JavaScriptový objektový zápis

```
{
  "sensor_id": "dht11_01",
  "location": "room",
  "owner": "pn",
  "status": "ok",
  "quantity": "humidity",
  "value": 46.0,
  "timestamp": "2020-04-09 11:32:30" }
```

Obsah zprávy je uzavřen do složených závorek a zpráva se skládá z dvojic klíč-hodnota, kde klíčem je vždy první řetězec znaků ve dvojici (*location*, *owner*, atd.) a hodnota je řetězec znaků za dvojtečkou. Jednotlivé páry klíč-hodnota jsou od sebe odděleny čárkou. Každá zpráva se skládá z následujících atributů.

- *location* je atribut udávající umístění senzoru; Může nabývat hodnot *room* v případě, že je senzor v místnosti nebo *outside* u senzoru, který je venku.
- *owner* udává provozovatele senzoru; Tento atribut slouží k filtraci senzorů podle majitele a využití má v případě, že je v jedné síti větší množství senzorů od různých vývojářů.
- *status* nabývá obecně dvou hodnot - *ok* nebo *error*; Senzor posílá v každé zprávě status čidla, ze kterého načítá data o měřené veličině; Tento atribut má význam při automatické detekci chyb na úrovni mikročipu ESP8266 (více v Kap. 4.1).
- *sensor\_id* je unikátní identifikátor, který se vztahuje přímo k danému čidlu; Například senzor obsluhující současně teplotní čidlo *ds18b20* a vlhkostní čidlo *dht11* má pro každé čidlo samostatný identifikátor - *ds18b20\_01* a *dht11\_01*.
- *quantity* je atribut popisující měřenou fyzikální veličinu; Význam tohoto atributu je v identifikaci měřené veličiny, protože senzory posílají hodnoty veličin bez jejich fyzikálních jednotek.
- *timestamp* je časová známka, která je přiložena ke každé zprávě a její hodnotou je okamžik odeslání zprávy; Jelikož časová synchronizace je řešena na úrovni samotného mikročipu, je možné ke každé zprávě přidávat časovou známku a díky tomu snadno filtrovat příchozí zprávy podle času a data, čímž odpadá problematika určování pořadí zpráv na brokeru.
- *value* je atribut, ve kterém se přenáší hodnota měřené fyzikální veličiny; Jde vždy o číselnou hodnotu bez jednotky; Například u senzoru měřícího teplotu může mít atribut hodnotu 25,3; U senzoru monitorujícího pohyb v místnosti nabývá atribut value hodnoty 1 nebo 0.

V Tab. 3.1 je po řádcích zobrazen výčet všech kombinací hodnot atributů, kterých mohou zprávy v projektu chytré domácnosti nabývat.

location	owner	status <sup>5</sup>	sensor_id	quantity
room	pn	ok	ds18b20_01	temperature
outside	pn	ok	ds18b20_02	temperature
room	pn	ok	dht11_01	humidity
room	pn	ok	tsl2591_01	illuminance
room	pn	ok	bme280_01	pressure
room	pn	ok	bme280_01	temperature
room	pn	ok	am312_01	motion
room	pn	ok	ls311b38_01	door_open
room	pn	ok	ls311b38_02	window_open

TABULKA 3.1: Atributy zprávy odesílané z mikročipu na broker

### 3.1.2 Hierarchie zpráv

Při přenosu dat dochází k datovému toku ve směru od senzorů na broker. Broker pouze přijímá všechny zprávy, ale sám neposílá žádná data senzorům. Kvůli přehlednosti a jednotné struktuře posílaných zpráv je definována jasná hierarchie zpráv. Zprávy jsou odesílány do *topiců*. Topic určuje téma dané zprávy a slouží k oddělení zpráv podle toho, od jakého odesílatele pocházejí, jakou fyzikální veličinu popisují a kde jsou měřené (v místnosti nebo venku). Každá zpráva je přiřazena právě jednomu tématu a název tohoto tématu určuje sám odesíatel zprávy. Příjemce zprávy pak jen musí znát název tématu, do kterého odesíatel publikuje zprávy. Témata jsou řetězce v kódování UTF-8 oddělená lomítky a jejich hierarchie není samotným protokolem nijak určená. Při návrhu hierarchie témat v projektu senzorického řešení domácnosti byl kláden důraz na univerzálnost a přirozenost s možností snadného rozšíření v budoucnu.

Základem názvů všech témat je slovo *smarthome* označující název projektu. Za tímto slovem se přidá umístění senzoru a název měřené veličiny. Po spojení těchto třech slov vzniká název konkrétního topicu, kam senzor odesílá zprávy. V Tab. 3.2 je uveden výčet všech topiců používaných v této práci.

smarthome/<location>/<quantity>
smarthome/room/temperature
smarthome/outside/temperature
smarthome/room/humidity
smarthome/room/illuminance
smarthome/room/pressure
smarthome/room/motion
smarthome/room/door_open
smarthome/room/window_open

TABULKA 3.2: Výčet všech topiců, do kterých jsou odesílány zprávy

Hierarchie témat byla navržena tak, aby každá měřená fyzikální veličina měla svůj topic v závislosti na umístění. Díky této volbě je možné strukturu

<sup>5</sup>Hodnota atributu *status* se může samozřejmě změnit z *ok* na *error*. Hodnoty atributů *timestamp* a *value* jsou logicky v každé zprávě jiné, proto nejsou uvedeny v tabulce.

témata snadno rozšířit jak z hlediska lokalit (přidání dalších pokojů v chytré domácnosti - *livingroom*, *bedroom*, atd.), tak z hlediska veličin (například *smoke\_detection*). Tento model hierarchie topiců také umožňuje spojovat téma do souvisejících celků. K těmto účelům slouží znaky + a #. Symbol + nahrazuje jednu úroveň v topicu, pro odebírání například všech teplot nezávisle na lokaci nebo čidlu lze použít příkaz *smarthome/+/temperature*. Symbol # nahrazuje slovo za posledním lomítkem, pro odebírání například všech veličin, které jsou měřeny v místnosti lze použít příkaz *smarthome/room/#*.

## 3.2 Ukládání dat do databáze

Jelikož senzory v této práci odesílají zprávy velmi často - s periodou jedné minuty, je potřeba data vhodným způsobem ukládat. K tomu slouží datová základna (databáze), ve které jsou data s pevnou strukturou záznamů propojena pomocí klíčů a která umožňuje snadno filtrovat záznamy podle zadaných kritérií. V tomto projektu jsou data ukládána do databáze *MongoDB*.

### MongoDB

MongoDB je multiplatformní dokumentová databáze, která na rozdíl od relačních databází využívajících systém tabulek, ukládá data do kolekcí - souborů, které jsou podobné formátu JSON. Tato databáze je pro projekt chytré domácnosti vhodná právě díky tomu, že nevyužívá klasický přístup ukládání dat do tabulek. Jednou z hlavních výhod klasických relačních databází je možnost provázat tabulky mezi sebou pomocí klíčů. To se hodí zpravidla na obsáhlější záznamy, které je potřeba strukturovat a pomocí cizích klíčů provozovat s ostatními tabulkami. Data získaná ze senzorů mají jednotnou strukturu a ukládání do tabulek by bylo neefektivní, proto byla zvolena databáze MongoDB, která je pro archivaci tohoto typu dat vhodnější. Databáze MongoDB má širokou podporu programovacích jazyků včetně Pythonu.

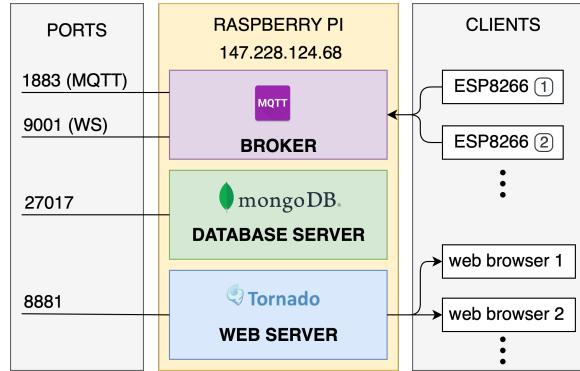
Databázový server odebírá nejvíce nadřazený topic *smarthome/#*, který zahrnuje všechna téma vypsaná v Tab. 3.2. Do databáze jsou ukládány všechny příchozí zprávy, jejichž *status* je "ok" (tedy v případě, kdy čidlo funguje bezproblémově a senzor odesílá relevantní data). Z každé příchozí zprávy s tímto statusem jsou uloženy hodnoty všech atributů - *location*, *owner*, *status*, *sensor\_id*, *quantity*, *value* a *timestamp*.

## 3.3 Webserver

V backendové<sup>6</sup> části serveru pro chytrou domácnost běží současně několik procesů. Celý systém byl navržen tak, aby se spuštěním jednoho hlavního skriptu rozběhly všechny části backendu. Tímto hlavním programem je skript *engine.py*. Engine.py zajišťuje spuštění MQTT Brokeru (Kap. 3.1), serverování webové stránky (více v Kap. 5), ukládání dat do MongoDB, trénování datových modelů pro následnou klasifikaci (více v Kap. 4.2) a diagnostiku čidel na úrovni serveru (více v Kap. 4.3). Na Obr. 3.3 jsou zobrazeny procesy, které běží na počítači Raspberry Pi. Na RPi je spuštěn MQTT Broker na portu 9001 pro websockets a 1883 pro klasický *subscribe* k odebírání

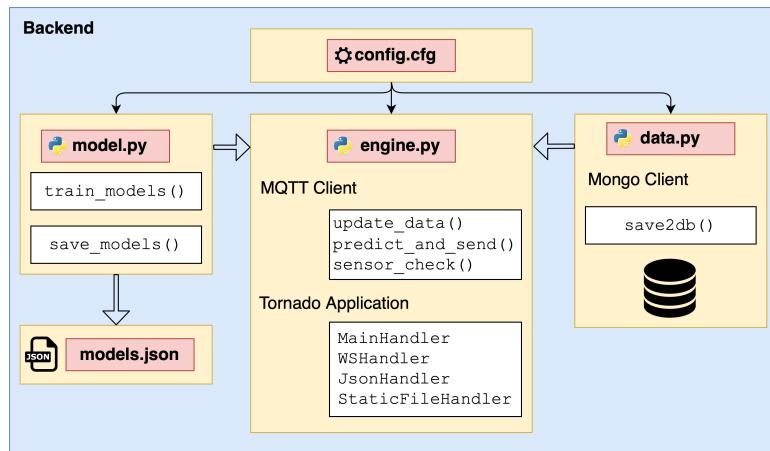
<sup>6</sup>Backend je v klient-server modelu vrstva operující nad daty. Jde o výpočetní logiku, která se skrývá pod uživatelským rozhraním (frontend).

publikovaných zpráv. Databázový server na RPi běží na výchozím portu 27017 a web server pro webovou vizualizaci na portu 8881.



OBRÁZEK 3.3: Schéma procesů s popisy portů běžících na Raspberry Pi

K zajištění všech těchto procesů je architektura kódu na RPi s využitím principů objektově orientovaného programování rozdělena do souborů *engine.py*, *model.py*, *data.py*, *models.json*, *config.cfg*. Diagram architektury kódu s rozdělením do souborů je zobrazen na Obr. 3.4.



OBRÁZEK 3.4: Diagram architektury kódu backendu na Raspberry Pi

Nad všemi skripty je konfigurační soubor *config.cfg* se vstupními parametry. Konfigurační soubor obsahuje parametry pro připojení k MQTT brokeru, porty pro webový a databázový server a parametry pro trénování modelů. *Engine.py* je hlavní skript, který využívá metod ze skriptů *model.py* a *data.py*. Funkce pro ukládání všech příchozích zpráv do databáze jsou nadeřízeny ve skriptu *data.py*. V tomto skriptu je vytvořena instance objektu *MongoClient* se vstupními parametry (název hosta a port). Jelikož databázový server (MongoDB) a webový server (jehož součástí je script *data.py*) běží fyzicky na stejném RPi, je název hosta "localhost" a výchozí port pro MongoDB je 27017. Hodnoty atributů ze zpráv jsou v databázi ukládány do kolekcí podle názvu topicu. Ve skriptu *model.py* probíhá trénování modelů pro predikci a klasifikaci naměřených hodnot (více v Kap. 4.2). Skript

model.py pracuje se souborem *model.json*, ve kterém jsou uloženy parametry pro trénování modelů - například časové období od kdy do kdy jsou brány vzorky dat pro trénování modelu. Skript engine.py se skládá ze dvou hlavních částí - MQTT Client a Tornado Application. Ve třídě MQTT Client jsou odebírána data z brokeru, tato data jsou klasifikována a následně přes WebSocket odesílána na webovou stránku. Třída Tornado Application je tvorena několika "Handlery". *MainHandler* se stará o renderování webové stránky - zpřístupňuje webovou stránku pod IP adresou ve webovém prohlížeči. Třída *WSHandler* obstarává komunikaci mezi webovým prohlížečem a serverem (např.: když uživatel volí, který model chce zobrazit). *JsonHandler* je třída, která pracuje s json soubory v backendové části serveru. Tento handler je využity k propagaci denních statistik natrénovaných klasifikátorů (denní statistika je zobrazena např.: na Obr. 5.4). Tyto statistiky se nahávají na web pouze jednou při načtení webové stránky, proto není nutné používat protokol WebSocket. Vhodnejší je využití json souboru, se kterým pracuje JsonHandler.

## websocket

*WebSocket* je síťový komunikační protokol, který umožňuje obousměrnou komunikaci po TCP protokolu. Jedná se o protokol, který má využití především ve webových aplikacích při komunikaci mezi klientem a serverem. V projektu chytré domácnosti je tento protokol využíván pro komunikaci mezi serverem a webovou vizualizací. Webová stránka potřebuje neustále aktuální data (naměřené hodnoty veličin) ze senzorů. WebSocket je ideální řešení přenosu dat ze serveru do webového klienta v reálném čase s minimální výpočetní náročností. Ve chvíli, kdy na broker přijde nová zpráva, je tato zpráva ihned odeslána protokolem WebSocket na webového klienta a webová stránka zobrazuje neustále aktuální data bez nutnosti obnovy stránky. Alternativou tohoto protokolu může být například načítání dat z externího souboru. Je možné vytvořit json soubor, do kterého budou nově příchozí zprávy ukládány a webová stránka bude periodicky načítat data z tohoto souboru. Hlavní nevýhodou tohoto přístupu je nutnost periodické kontroly externího souboru s daty. K tomu, aby na webové stránce byla neustále aktuální data, by bylo potřeba načítat soubor velmi často - například každé 2 až 3 vteřiny. To je velmi neefektivní, protože nová data ze senzoru chodí pravidelně po 1 minutě. Webový klient by tedy pořád otevíral soubor s daty a kontroloval, zda v něm jsou nová data, přičemž u naprosté většiny těchto kontrol by zjistil, že data jsou stále stejná. Tuto neefektivitu a zbytečnou výpočetní složitost nahrazuje protokol WebSocket. Zásadní výhodou protokolu WebSocket je možnost odesílání dat ze serveru do webového prohlížeče, aniž by si klient tuto zprávu musel vyžádat (tj. klient se nemusí každé 2-3 vteřiny dotazovat serveru zda nemá nová data. Nová data dostane automaticky ve chvíli, kdy je server obdrží). Současně je možné využít WebSocket i pro komunikaci opačným směrem - z klienta na server a uživateli tím umožnit interagovat s daty v backendu prostřednictvím webového prohlížeče. Tento směr komunikace lze využít například pro přetrénování modelů na základě vstupních parametrů, které uživatel zadá na webové stránce (uživatel může měnit časové rozpětí dat - od kdy do kdy, která jsou použita pro trénování modelu).

## Kapitola 4

# Diagnostika a detekce anomálií

Systém diagnostiky chyb a detekce anomálií je v projektu chytré domácnosti rozdělen do tří úrovní - detekce chyb na úrovni mikročipu ESP8266, detekce anomálií na základě klasifikace z natrénovaného modelu a diagnostika stavu čidel na serveru. Všechny části diagnostického systému jsou popsány v této kapitole.

### Význam diagnostického systému

Smyslem využití systému detekce anomálií je automatická diagnostika senzorů na základě příchozích zpráv. Výstupem tohoto systému je informace o stavu klientů. Stavy jednotlivých senzorů se propisují do webové vizualizace (více v Kap. 5) a díky ikonám vedle měřených veličin dostává uživatel okamžitý a kompletní přehled o stavu senzorů v chytré domácnosti. Zároveň má jistotu, že data zobrazovaná na webové stránce jsou aktuální, protože diagnostický systém ví, že data ze senzorů mají chodit pravidelně s pevnou periodou a když nové hodnoty nepřijdou, upozorní na to změnou ikony. Uživatel se tedy dozví, že senzor nepracuje stoprocentně a zobrazená data nemusejí být relevantní.

- Proč řešit diagnostiku čidel v rámci chytré domácnosti, ...
- Úvod do detekce anomálií, základní principy, smysl využití, ... - možná přidat ještě nějakou větu..

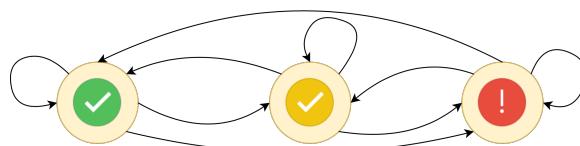
### Výstup systému detekce anomálií

Výstupem systému diagnostiky chyb a detekce anomálií jsou tři stavy přiřazené jednotlivým senzorům. Senzor se může nacházet v jednom z těchto tří stavů.

- *Sensor OK* je stav senzoru, kdy všechno funguje bezproblémově; Senzor odesílá data tak, jak se očekává - periodicky každou minutu a naměřená hodnota odpovídá natrénovanému modelu (více v Kap. 4.2); Ikona pro tento stav je zelená.
- Stav *Value ERROR* nastává, pokud senzor odesílá data periodicky - tak, jak se očekává, ale naměřená hodnota je mimo předpovězený rozsah - neodpovídá natrénovaném modelu; Tento stav je tedy mezi stavami "Sensor OK"- kdy je vše v pořádku a "Sensor ERROR"- kdy senzor nefunguje vůbec; Ikona pro tento stav má žlutou barvu.

- Sensor *ERROR* je status čidla, který nastává v momentě, kdy čidlo neposílá žádná data; Tento stav je způsobený pravděpodobně výpadkem čidla nebo nedostupností internetu; Tento status senzoru popisuje červená ikona.

Na Obr. 4.1 je zobrazeno schéma Markovova řetězce, který popisuje přechody mezi jednotlivými stavami senzorů. Z tohoto řetězce je patrné, že každý stav může přejít do libovolného jiného stavu nebo zůstat v aktuálním stavu. To znamená, že pokud se čidlo nachází v jednom momentě například ve stavu "sensor OK", v následujícím momentě se může nacházet opět ve stavu "OK" nebo v libovolném jiném stavu - "value ERROR" nebo "sensor ERROR".



OBRÁZEK 4.1: Markovův řetězec popisující přechody mezi jednotlivými stavami senzorů

## 4.1 Detekce chyb na úrovni ESP8266

První a nejnižší úroveň diagnostiky jednotlivých senzorů je detekce chyb na úrovni samotného mikročipu. Mikročip ESP8266 při každém načítání dat z čidla zkouší, jestli z daného čidla lze načíst data a pozná, když je čidlo nefunkční. Čidlo se považuje za nefunkční, pokud při vyžádání naměřených dat nevrací žádné hodnoty. Tento stav čidla může být způsobený špatným zapojením, vypojením čidla nebo rozbitým čidlem. Implementace detekce chyb na úrovni mikročipu spočívá v obsluze výjimek v kódu. Část kódu, která řeší komunikaci s čidlem je obalena v *try-except* bloku - mikročip se snaží načíst data z čidla a v případě, že to nelze, spadne kód do části *except*, kde se obslouží výjimka tím, že se do proměnné, kam se ukládá naměřená hodnota, přiřadí číslo -1. Cyklus v kódu pokračuje dále k vytváření struktury zprávy, kde se testuje, zda se hodnota v proměnné nerovná -1. Pokud ano, ve zprávě se změní hodnota atributu "status" na "error" a senzor odešle zprávu (více o programování mikročipu v Kap. 2.1). Senzor tedy odesílá zprávy na broker vždy - s funkčním i nefunkčním čidlem a status zprávy v detekci chyb na této úrovni nabývá pouze dvou hodnot - "ok" a "error".

Implementací detekce chyb na úrovni mikročipu byla zajištěna robustnost senzorů. Jednotlivé senzory jsou odolné vůči výpadkům čidel - kód na mikročipu nespadne kvůli nefunkčnímu čidlu, jen se změní status zprávy. Díky tomu je také zajištěn nonstop provoz - čidla lze k ESP8266 připojovat a odpovídat (například čidlo teploty nebo magnetické kontakty, které jsou připojeny přes konektor) v reálném čase bez nutnosti restartu mikročipu. Pokud čidlo v jakémkoliv časovém okamžiku odpojí a po chvíli zase připojí, mikročip si s tím poradí. Změní se status zprávy a celý systém funguje dál.

Výhoda implementace detekce chyb na úrovni mikročipu se během testování ukázala několikrát. Například když jsem potřeboval vyměnit nefunkční čidlo vlhkosti DHT11, které je připojené přes dutinkovou lištu (více v Kap. 2.2.2),

jen jsem vyměnil kus za kus a senzor okamžitě začal načítat data z čidla a posílat naměřené hodnoty.

## 4.2 Detekce anomalií na základě klasifikace

Ve skriptu *engine.py* jsou příchozí data z brokeru klasifikována a následně odesílána na webovou stránku. Klasifikace -1 nebo 1 podle... Webová stránka dále pracuje s klasifikací - zobrazení ikon u jednotlivých veličin (zelená, žlutá, červená)

- pozn.: vlhkost v místnosti se mění s ročním obdobím - v zimně kolem 40-50 procent, v létě kolem 20-30 procent..
- Implementace funkcí scikit-learn pro natrénovaní modelů
- Nastavení modelu a natrénování modelů pro jednotlivé veličiny
- Porovnání schémat natrénovaných modelů
- Implementace do projektu - klasifikace jednotlivých příchozích zpráv (přidání atributu "classification" do každé zprávy) - Implementace detekce anomalií přijatých zpráv na základě rozhodnutí klasifikace - klasifikace 1 nebo -1 (1 v případě, že přijatá zpráva svou hodnotou "odpovídá" natrénovanému modelu, -1 v případě, že se vychyluje od natrénovaného modelu)
- Přenos této informace o klasifikaci do webového rozhraní

## 4.3 Diagnostika stavu čidel na serveru

Diagnostika stavu čidel na úrovni serveru spočívá v implementaci funkce *sensor\_check*, která běží ve skriptu *engine.py* v backendové části serveru (popsáno v Kap. 3.3). Systém diagnostiky na této úrovni spojuje předešlé dvě úrovně detekce chyb. Pracuje současně s informací o stavu na úrovni mikročipu (popsáno v Kap. 4.1) a s informací o klasifikaci (popsáno v Kap. 4.2). Funkce *sensor\_check* kontroluje pravidelnost odesílání dat ze senzoru na broker. Pokud čidlo z neznámého důvodu neodešle zprávu nebo pokud se zpráva nepřenese k brokeru, informace o anomalií se přenese do webového rozhraní. Na Obr. 4.2 je ukázka příkladu zobrazení chyby senzoru ve webové vizualizaci. V případě chyby senzoru se změní ikona statusu a hodnota měřené veličiny je automaticky 0. Více o webové vizualizaci v Kap. 5.2.



OBRÁZEK 4.2: Zobrazení ve webové vizualizaci v případě chyby senzoru

Funkce *sensor\_check* je implementována ve třídě MQTT Client a v novém vlákně běží paralelně vedle ostatních metod. Tato funkce čte všechny příchozí zprávy a porovnává čas poslední přijaté zprávy od daného čidla s aktuálním časem. Jestliže je rozdíl těchto dvou proměnných (čas poslední přijaté zprávy minus aktuální čas) větší než zvolená mez, funkce detekuje chybu, do terminálu vypíše ID senzoru, ve kterém se vyskytuje chyba a status tohoto

senzoru změní na "sensor ERROR". V tomto projektu byla zvolena 60 vteřinová mez, což znamená, že stačí, aby senzor jednou neodeslal zprávu a hned je považován za vadný. Toto kritérium je relativně přísné, v běžném provozu by stačilo mez volit například 180 vteřinovou, tudíž by se ve webovém rozhraní zobrazoval senzor jako aktivní i když by dvakrát neodeslal zprávu. Tři minuty staré data lze považovat za stále aktuální a stav senzorů by byl více konzistentní - občas se stane (ne příliš často, například jednou za 2 dny provozu), že senzor z nějakého důvodu nestihne odeslat zprávu v dané minutě a webová vizualizace ho ihned vyhodnotí jako nefunkční, přitom senzor v dalších minutách opět zprávy odesílá. Status senzoru se zbytečně rychle změnil a uživatel by mohl být zmatený.

## Hierarchie statusů

Ve webovém rozhraní se mění ikony statusu podle tří stavů, které jsou popsány na výše definovaných úrovních. Statusy se mohou navzájem přepisovat podle hierarchie důležitosti, která je následující.

Sensor ERROR > Value ERROR > Sensor OK
--

To znamená, že pokud senzor například odešle zprávu se statusem "sensor OK", ale hodnota měřené veličiny je klasifikována jako -1, na webu se zobrazí status odpovídající klasifikaci -1 - "value ERROR", protože status "value ERROR" má větší váhu, než status "sensor OK".

Pokud senzor odešle zprávu se statusem "sensor ERROR", tato zpráva není dále klasifikována a na webové stránce se zobrazí ikona "sensor ERROR". Váha tohoto statusu je nejvyšší.

Pokud senzor odešle zprávu se statusem "sensor OK" a klasifikátor přiřadí hodnotu 1, ikona statusu na webu zůstává "sensor OK".

Když senzor odešle zprávu se statusem "sensor OK", ale pak několik minut po sobě přestane odesílat zprávy, na webu se status změní na "sensor ERROR".

Diagnostika a změny ikon zpráv jsou zkrátka řešeny tak, aby to bylo pro uživatele přirozené a na první pohled pochopitelné. Zelená ikona znamená, že je všechno v pořádku. Žlutá ikona uživateli říká, že je všechno víceméně v pořádku, ale něco není úplně stoprocentně správně. Červená ikona uživatele varuje a říká, že je senzor nefunkční. Diagnostický systém kombinuje vstupy ze systému kontroly stavu čidel s kontrolou věrohodnosti posílaných zpráv a kontrolou pravidelností odesílaných zpráv. Kombinací těchto tří subsystémů generuje výstupní ikony na webové stránce, které informují uživatele.

## Kapitola 5

# Webové rozhraní

Hlavním výstupem této práce je grafická vizualizace pro přehledné zobrazení veškerých dat a komunikaci s uživatelem. Za účelem této prezentace naměřených dat a modelů byla zvolena vizualizace ve formě webové stránky. Webová stránka je dostupná na adresě 147.228.124.68:8881 a její struktura je rozdělena do tří hlavních podstránek. Základem webu je horní horizontální menu, které se skládá z těchto tří tlačítek a uživateli umožňuje proklikávání mezi jednotlivými stránkami.

Overview	Analytics	About
----------	-----------	-------

Z hlediska web designu mají tlačítka v horizontálním menu úmyslně šedý odstín fontu písma (oproti bílému fontu ve zbytku stránky). Odlišují se tím tlačítka a text, na který není možné klikat. Jednotlivá tlačítka se po přejetí kurzorem mírně zvětší a nabádají uživatele ke kliknutí. *Overview* je stránka, která uživateli poskytuje kompletní a stručný přehled o veškerém dění v chytré domácnosti. Stránka *Analytics* umožňuje interakci s uživatelem a nabízí detailní informace o jednotlivých senzorech, stavu senzorů a zobrazuje natřenované modely. *About* je doplňující stránka, která velmi stručně popisuje tento projekt a dodává vysvětlivky k ikonám na webu.

První akce, kterou webová stránka po zobrazení v internetovém prohlížeči vykoná, je načtení dat ze souboru *webpage\_data.json*. V tomto souboru jsou uloženy poslední hodnoty senzorů. Význam tohoto datového souboru je v okamžitém zobrazení posledních hodnot po načtení stránky bez nutnosti čekání, než senzory odešlou data. Hlavní význam to má u senzorů, které odesílají data na základě události (více v Kap. 2.1) - například změna stavu okna (otevřené nebo zavřené) se děje jen několikrát denně a kdyby webová stránka neměla k dispozici soubor *webpage\_data.json*, pro zobrazení stavu okna by uživatel musel počkat, než se tento stav změní a senzor odešle stav, který se následně propíše na webovou stránku. Po obnovení webové stránky by se poslední hodnota opět ztratila. Soubor *webpage\_data.json* je používán pouze při prvním načtení webu, od této chvíle už webový klient přijímá aktuální hodnoty přes WebSocket.

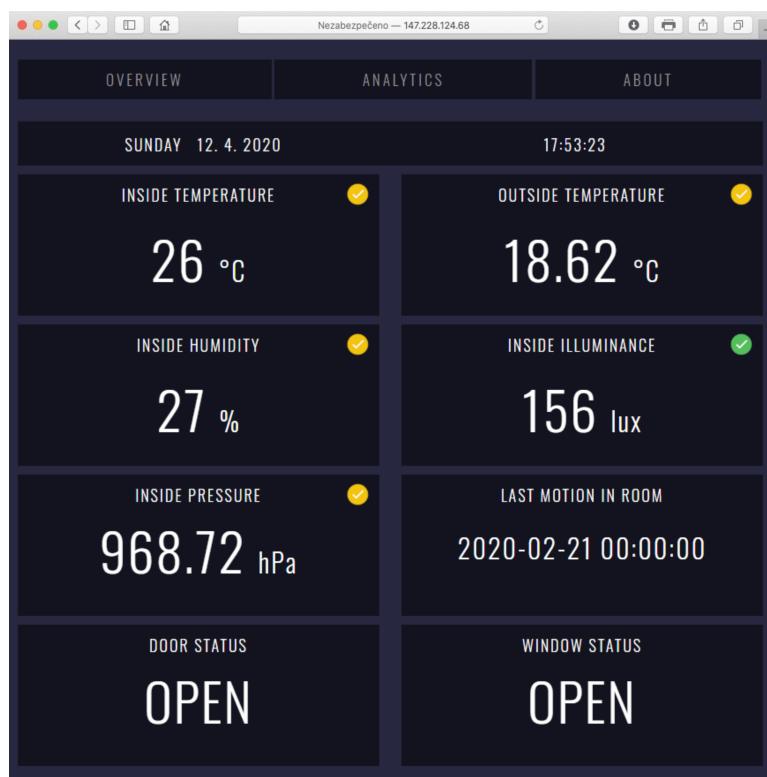
### 5.1 Programování frontendu

Webová stránka byla naprogramována v *HTML* s využitím *JavaScriptu* a kaskádového stylu *CSS*. Struktura kódu je rozdělena podle tří hlavních stránek. Každá stránka má svůj soubor s JavaScriptovým kódem a skriptem v CSS. *JavaScript* je zde využitý pro zajištěný potřebných funkcionalit webu

- například nově příchozí zpráva se automaticky propíše do požadovaného místa, při uživatelské volbě parametrů pro načtení natrénovaných modelů JavaScript zobrazuje data v požadovaném místě, dále JavaScript mění ikony statusu senzorů podle změn těchto statusů atd. V CSS souborech je na definován design webu. Jsou zde uloženy parametry pro vykreslení dlaždic - rozměry, barvy, fonty písma, rozmístění na stránce atd. Kombinací těchto tří programovacích jazyků bylo dosaženo interaktivní webové vizualizace pro prezentaci naměřených dat v projektu chytré domácnosti.

## 5.2 Overview

Overview je podstránka, která se načte jako výchozí při návštěvě webu. V záhlaví stránky se zobrazuje aktuální čas a datum a pod pruhem s těmito informace se nacházejí dlaždice obsahující informace o naměřených hodnotách. Na Obr. 5.1 je zobrazen náhled stránky Overview.



OBRÁZEK 5.1: Stránka Overview ve webovém rozhraní

Šablona dlaždice (Obr. 5.2) je pro každou veličinu stejná - skládá se z názvu měřené veličiny, ikony zobrazující stav senzoru (více v Kap. 4.3) a číselné hodnoty s fyzikální jednotkou.



OBRÁZEK 5.2: Detail dlaždice

Ikona v pravém horním rohu dlaždice se mění podle stavu senzoru. Status senzoru může nabývat tří hodnot. Jednotlivé ikony jsou na Obr. 5.3 a význam těchto stavů je popsán v Kap. 4.3.

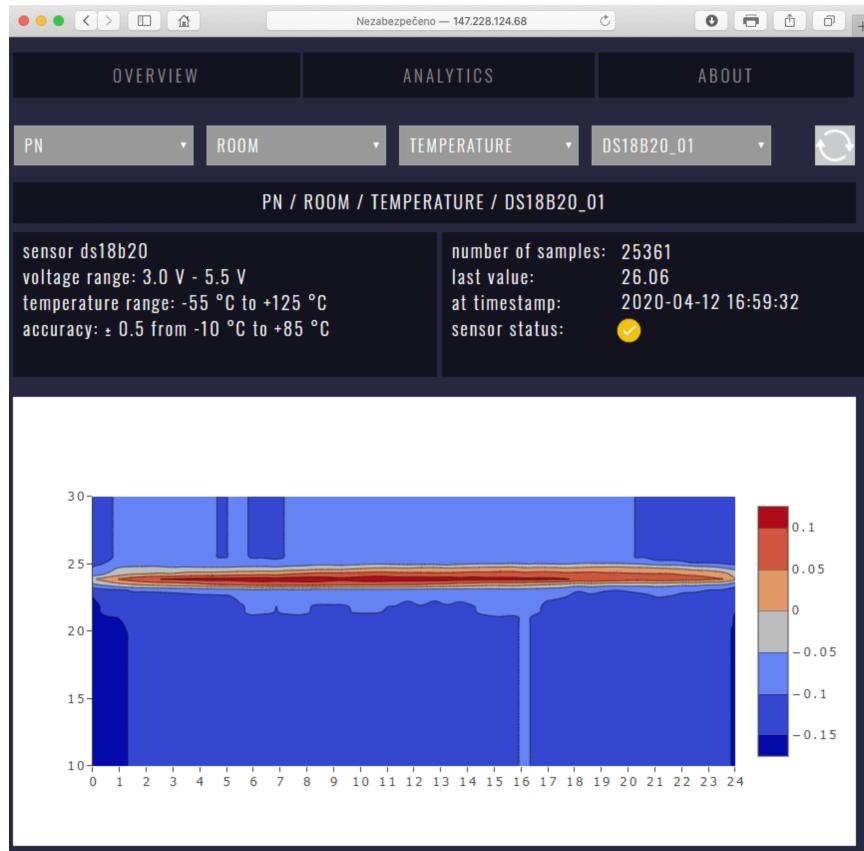


OBRÁZEK 5.3: Zobrazení všech ikon popisujících stavu senzoru

Záměrem stránky Overview je neustálý přísun aktuálních dat bez nutnosti jakékoliv obsluhy. Tato stránka může být otevřená nonstop na monitoru a uživatel díky ní má okamžitý přehled o veškerém dění v chytré domácnosti. Na stránku se pomocí protokolu WebSocket (více v Kap. 3.3) automaticky propisují nejnovější data bez nutnosti obnovy webu. Účelem je uživateli uceleně informovat o aktuálních hodnotách měřených veličin v chytré domácnosti.

### 5.3 Analytics

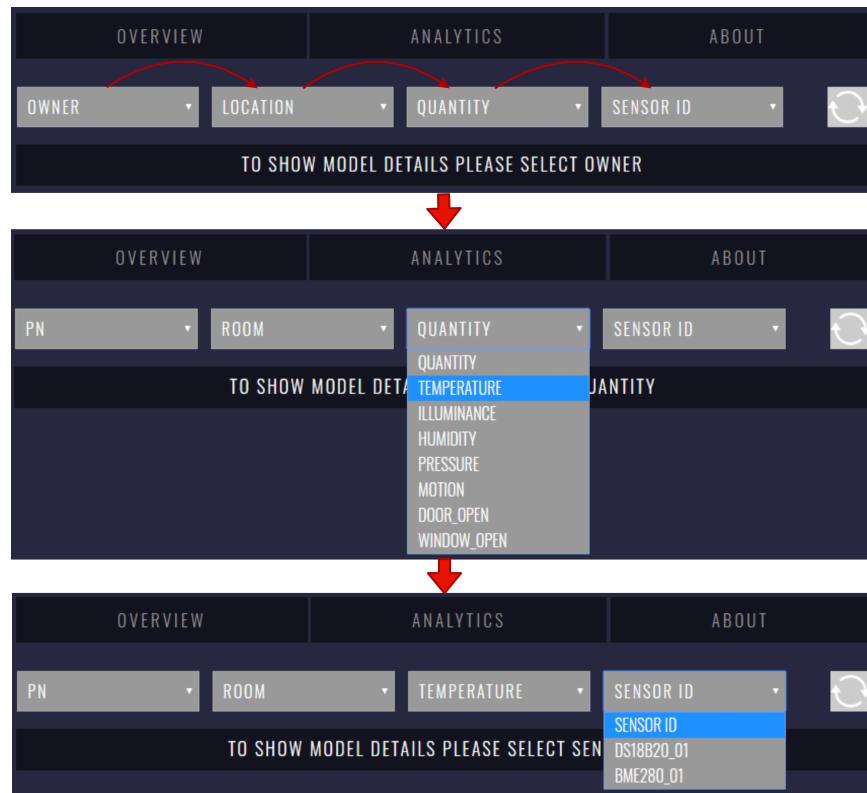
Stránka Analytics umožňuje výběr konkrétní měřené veličiny a nabízí náhled na natréновaný model, na jehož základě jsou naměřené hodnoty klasifikovány. Prvním krokem po vstupu do sekce Analytics je výběr z předvoleb v horizontálním panelové sekci pod hlavním menu. Po vyplnění všech předvoleb dostane uživatel informace o daném senzoru a zobrazí se natrénovaný model.



OBRÁZEK 5.4: Stránka Analytics se zobrazeným modelem pro vnitřní teplotu z čidla DS18B20\_01

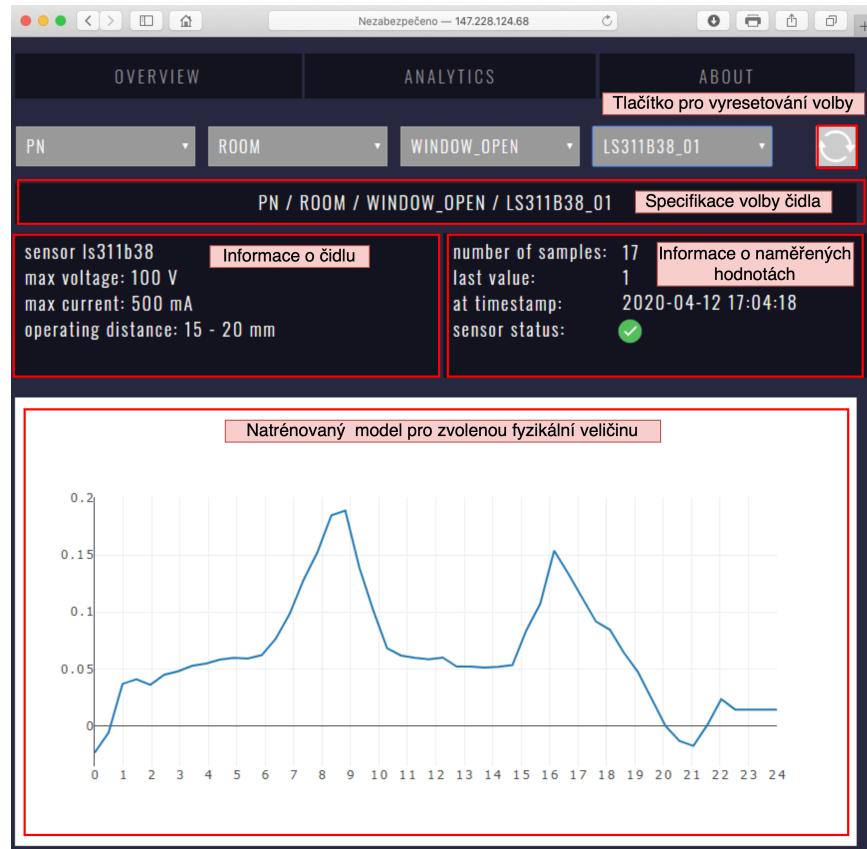
Volba z předvoleb v horizontální sekci probíhá zleva doprava. Uživatel musí nejprve zvolit vlastníka senzoru. Dokud nezvolí vlastníka, nemá možnost kliknout na další políčka. Po volbě vlastníka senzoru zvolí umístění senzoru (opět dokud nezvolí lokaci, nemá možnost kliknout na tlačítko "Quantity"). Po volbě lokace vybere fyzikální veličinu, jejíž data chce zobrazit. Konečně po volbě veličiny vybere konkrétní senzor, který měří zvolenou fyzikální veličinu. Po vyplnění tlačítka ("Sensor ID") se volba odešle na server a pod panelovou sekcí se zobrazí požadovaná data. Poslední tlačítko v panelové sekci vpravo slouží k vyresetování předem zadaných voleb. Po kliknutí na toto tlačítko dostane uživatel možnost vybrat například jinou fyzikální veličinu, která ho zajímá, aniž by musel obnovovat celou stránku. Předvolby, které uživatel volí, se mění v průběhu procesu výběru - například pokud uživatel zvolí jako lokaci "room", v dalším kroku se mu nabídnu pouze fyzikální veličiny, které jsou měřeny v rámci místnosti. Pokud jako lokaci vybere "outside", v dalším kroku se mu zobrazí možnost vybrat pouze fyzikální veličiny ve venkovním prostředí (v tomto případě venkovní teplota). Proces specifikace voleb je znázorněn na Obr. 5.5. Stejným způsobem funguje poslední volba na tlačítku "Sensor ID". Množina senzorů, které je možno v tomto tlačítce vybrat je kompletně závislá na všech předešlých volbách - uživatel musí zvolit vlastníka, lokaci a měřenou veličinu, aby bylo možné identifikovat senzory, které splňují tyto podmínky. Na Obr. 5.5 byl zvolen vlastník *PN*, lokace *ROOM*, veličina *TEMPERATURE* a v poslední volbě "Sensor ID" je možné vybrat jen *DS18B20\_01* nebo *BME280\_01*, protože pouze tyto dvě čidla

jsou od vlastníka PN a měří teplotu v místnosti.



OBRÁZEK 5.5: Proces výběru z předvoleb a specifikace zobrazení dat

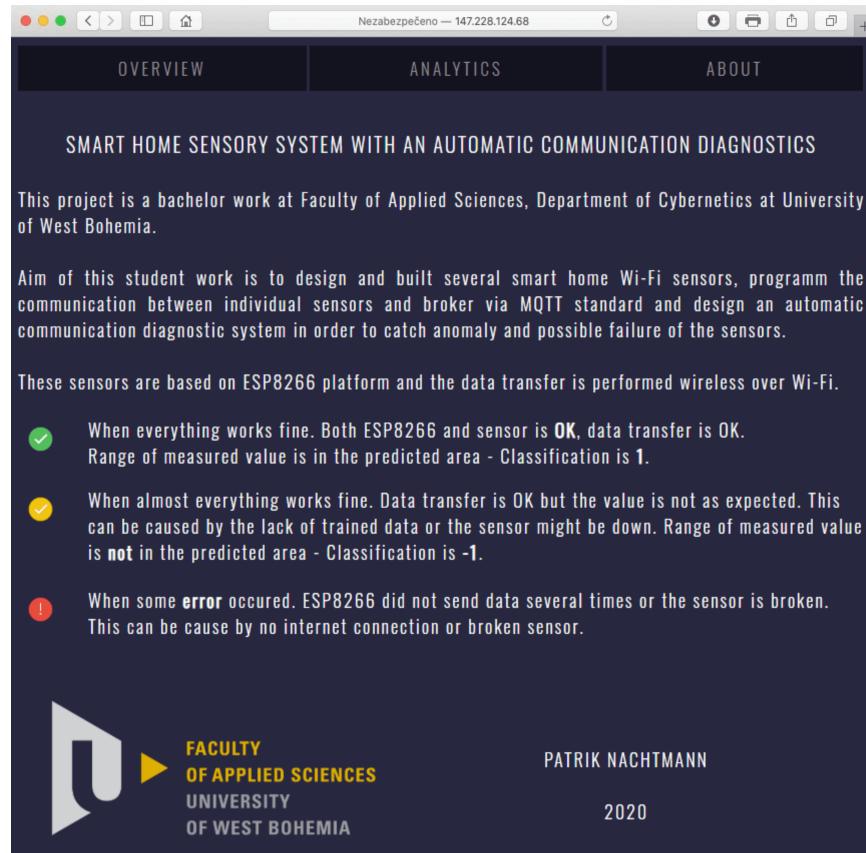
Na Obr. 5.6 jsou popsány jednotlivé dlaždice na stránce Analytics. Po uživatelské volbě všech vstupních parametrů je pod panelovou sekcí zobrazena specifikace zvoleného čidla, v levé dlaždici uprostřed stránky je stručný popis parametrů daného čidla, vedle se nacházejí základní informace o čidle a ve velké dlaždici je zobrazen natrénovaný model pro danou fyzikální veličinu. V dlaždici s informacemi o naměřených hodnotách dostane uživatel stručný přehled o datech ze senzoru - kolik vzorků dat se nachází v databázi, jaká byla poslední hodnota, kdy byla tato hodnota naměřena a konečně status senzoru.



OBRÁZEK 5.6: Popis jednotlivých dlaždic na stránce Analytics

## 5.4 About

About je doplňující stránka, která poskytuje stručné informace o projektu a vysvětlivky k používaným ikonám. Na Obr. 5.7 je zobrazena kompletní stránka About.



OBRÁZEK 5.7: Stránka About ve webovém rozhraní

Celá webová vizualizace byla navržena tak, aby nabízela jak stručný a výstižný pohled na aktuální dění v chytré domácnosti, tak pokročilý náhled na měřené veličiny s možností vykreslení natrénovaných modelů, které slouží primárně pro klasifikaci dalších zpráv, ale i například k vytvoření představy o tom, jak se měřená veličina vyvíjela a měnila v čase.

# Kapitola 6

## Závěr

Conclusion and Discussion

Conclusion text...

### 6.1 Future Work

Outlook...

# Bibliography

- [1] Warren S McCulloch a Walter Pitts. „A logical calculus of the ideas immanent in nervous activity“. In: *The bulletin of mathematical biophysics* 5.4 (1943), s. 115–133.
- [2] Peter Bradley. *The XOR Problem and Solution*. 2006. URL: <http://mind.ilstu.edu/>.
- [3] Martin Bulín. „Classification of terrain based on proprioception and tactile sensing for multi-legged walking robot“. Dipl. Campusvej 55, 5230 Odense M: University of Southern Denmark, červ. 2016.
- [4] Luboš Šmídl. personal communication. supervision of the thesis. 2017.

## Příloha A1

# Structure of the Workspace

