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

13. března 2020

Declaration of Authorship

Česky, bude stačit kratší - starší bakalářky ze ZČU. Možná je dokonce oficiální, které se k textu přidá až po tisku, vyřešíme až na konci...

I, Patrik NACHTMANN, declare that this thesis titled, „Senzorické řešení chytré domácnosti s automatickou diagnostikou komunikace“ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

„*Something supersmart.*“

Your hero

ZÁPADOČESKÁ UNIVERZITA

Abstrakt

Fakulta aplikovaných věd

Katedra kybernetiky

Bakalář (Bc.)

Senzorické řešení chytré domácnosti s automatickou diagnostikou komunikace

by Patrik Nachtmann

Your abstract goes here...

Acknowledgements

Your acknowledgements go here...

Obsah

Seznam obrázků

Seznam tabulek

List of Abbreviations

IoT **I**nternet **o**f **T**hings

Kapitola 1

Introduction

Přeložit do češtiny

Your intro... Thesis ref example: **bulin:2016**, Misc ref example: **smidl:pc**,
Article ref example: **mcculloch:neuron**, Online webpage ref example: **online:xor_solution**

1.1 State of the Art

1.2 Thesis Objectives

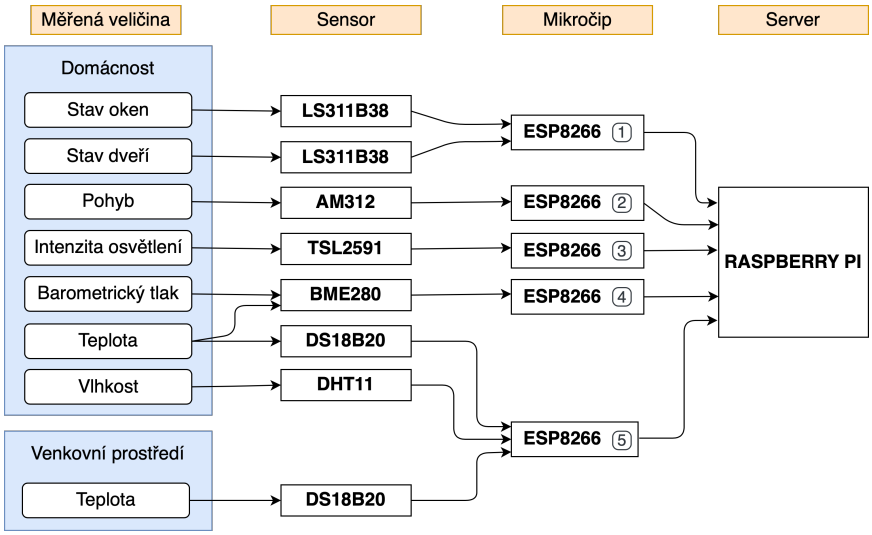
1.3 Thesis Outline

Kapitola 2

Hardwarové 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, barometrický tlak a existence pohybu. 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 nejefektivnější využití hardwaru může jeden mikročip číst data z několika čidel. Děje se to hlavně u čidel teploty a vlhkosti a u čidel pro monitorování stavu oken a dveří. U ostatních čidel je zpravidla využit jeden mikročip pro čtení dat z jednoho specifického čidla. Tento přístup byl využit zejména z důvodů specifického fyzického umístění senzoru (např.: pohybový senzor musí být umístěn v rohu místnosti, kde není vhodné současně měřit jiné veličiny) nebo z důvodu výpočetní náročnosti, kdy jeden mikročip zvládne načítat data pouze z jednoho čidla.

2.1 Mikročip ESP8266

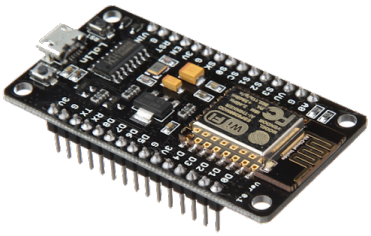
Základním stavebním prvkem senzorů v této chytré domácnosti je mikročip **ESP8266**. Existuje celá řada mírně odlišných mikrokontrolérů založených na tomto modulu, v principu se ale jedná o velmi levný mikročip s procesorem o frekvenci 80 Hz, flash pamětí typicky od 512 KiB do 4 MiB, 16 vstupně-výstupních GPIO piny a podporou sběrnic SPI, I^2C a I^2S [1].

Vlastnosti mikročipu

Hlavní předností toho 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 rozšiřují 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 chytré domácnosti jsem využil výhradně vývojovou platformu **NodeMCU** (na Obr. 2.2[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[3] a propojovací kabely. V fázi vytváření prototypu tedy není nutné pájet.



Obrázek 2.2: Vývojová platforma NodeMCU s modulem ESP8266

[1] https://en.wikipedia.org/wiki/ESP8266

[2] Převzato z http://pdacontrolen.com/introduction-platform-iot-cayenne-mydevices-esp8266/

[3] Nepájivé kontaktní pole. Komponenta, která umožňuje sestavit prototyp obvodu a je vhodná pro prvotní experimentování.

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 vyžaduje ovladač CP2102 (alternativně CH340). Pro nahrání samotného firmwaru do mikročipu jsem zvolil nástroj esptool.py[4]. 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

Tento mikročip není plnohodnotným počítačem ve smyslu jakéhokoliv uživatelského rozhraní. Veškerá komunikace s čipem probíhá po USB rozhraní a je založena na principu nahrání skriptů do paměti čipu a následném restartování pro spuštění programu. Jakákoliv 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[5]. Mpfshell 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 jevil jako logická volba. Nespornou výhodou tohoto jazyku je velká podpora v rámci komunity a množství návodů a knihoven pro komunikaci s jednotlivými čidly.

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

[4] https://github.com/espressif/esptool
[5] https://github.com/wendlers/mpfshell

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.

Architektura kódu na mikročipu

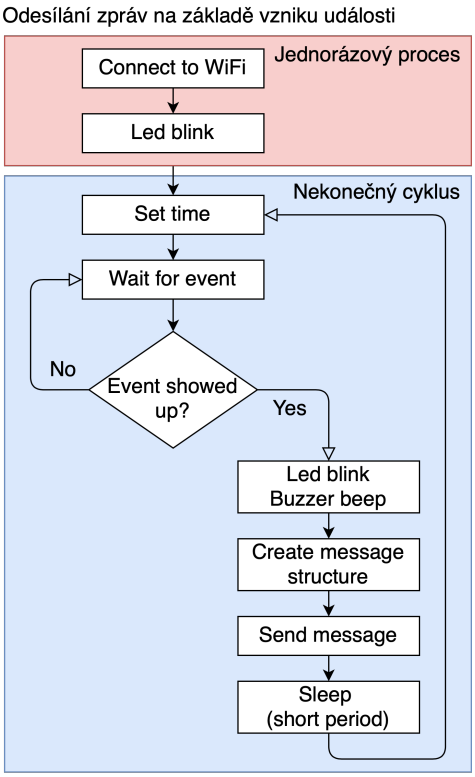
Kód v MicroPythonu pro mikrokontrolér ESP8266 sestává z několik částí. Primárně se jedná o soubor *boot.py* a *main.py*. Soubor *boot.py* se po zapnutí mikrokontroléru spustí vždy jako první nezávisle na jakémkoliv dalším souboru. V tomto skriptu dochází k připojení k místní Wi-Fi síti. Po úspěšném připojení k síti senzor čtyřikrát zabliká interní led a následně i externí led diodou pro vizuální kontrolu. Druhým skriptem, který naběhne ihned po *boot.py* je *main.py*. Toto pořadí a dané dva skripty jsou pevně stanovené programovacím jazykem nezávisle na implementaci dalších skriptů. 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 problikne 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[6]. 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 této části kódu tedy čidlo setrvává naprostou většinu času. V momentě, kdy událost nastane, čidlo problikne externí led diodou (v případě pohybového čidla se pro indikaci zapne ještě externí bzučák) a přejde do fáze vytváření zprávy. Po vytvoření struktury zprávy, která sestává z několika atributů (více v section 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 někde nazacyklil. V případě pohybového senzoru je krátké uspání vhodné také pro to, aby čidlo neposílalo zbytečně moc zpráv, když se v místnosti vyskytuje pohyb. Šetří se tím vytíženost sítě a pro účel detekce pohybuje není nutné odesílat informaci několikrát za vteřinu.

více v sec 3.1 přeložit do češtiny

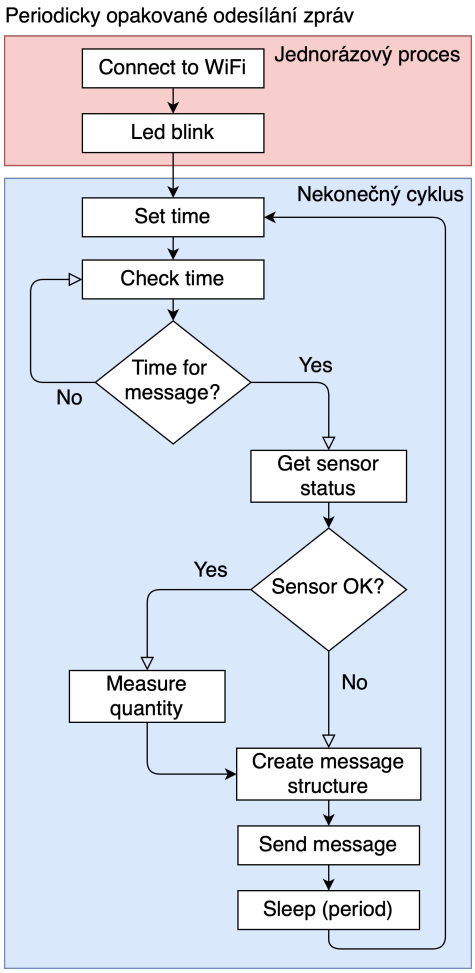
[6] **N**etwork **T**ime **P**rotocol - protokol pro synchronizaci času počítače po internetu



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, které měří 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 fyzikálních veličin opakovat za účelem získání dostatečného množství dat pro následné trénování modelů a klasifikaci.



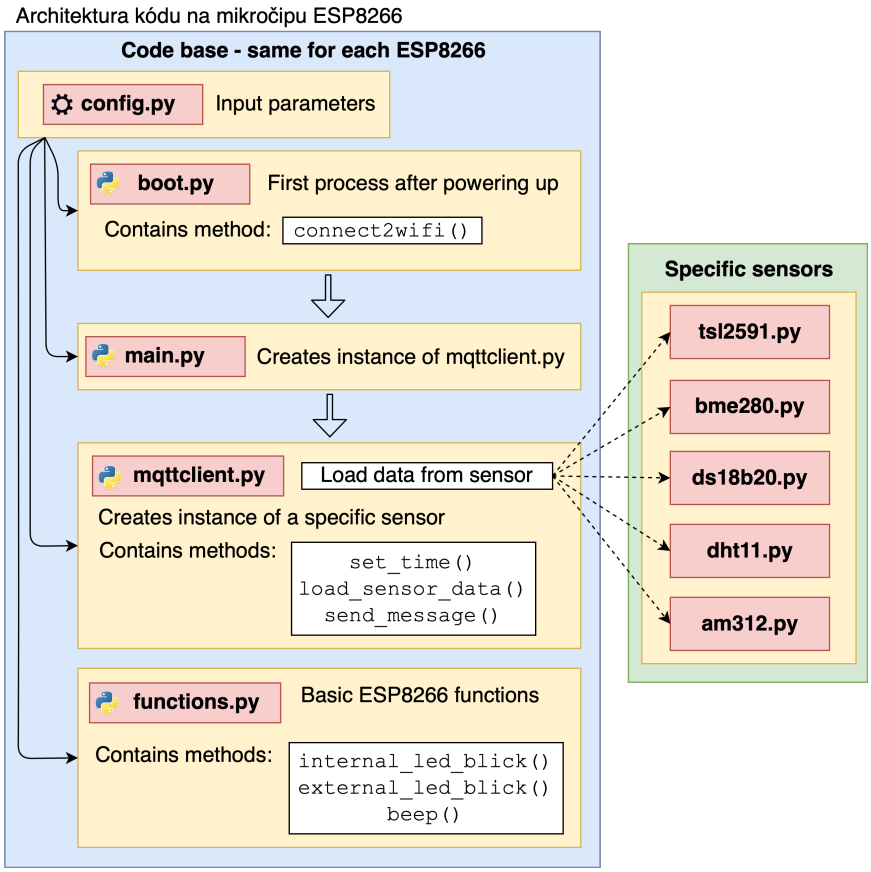
Obrázek 2.4: Diagram periodicky opakovaného odesílání zpráv

Vytvoření designu a vytvoření pevného obvodu - ESP + čidlo na PCB desce
Popis logiky komunikace mezi ESP, jednotlivými čidly a odesíláním zpráv (prvně připojit k wifi -> načíst hodnotu měřené veličiny -> v případě, že hodnota OK -> odeslat zprávu na broker -> uspat se)
Naprogramování logiky ESP - dva použité principy (periodické odesílání zpráv nebo odesílání zpráv založené na události)
Objektově orientovaná struktura kódu na ESP - boot.py, main.py, config.py, mqttclient.py, skript pro načítání dat z konkrétního senzoru (důraz kladen na objektové programování - přehlednost kódu, snadné změny, konfigurační soubor pro nastavení vstupních parametrů, univerzální kód pro všechna esp, snadná rozšiřitelnost kódu do budoucna, snadná implementace dalších senzorů, ...)

Detail architektury kódu na mikročipu

Architektura kódu všech senzorů sestává ze stejného základu. Při návrhu programu pro mikročipy byl kladen 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 znovu nebo složitě vytrhávat části jinde použitého kódu, ale použiji stávající program, který pouze doplním o implementaci komunikace s konkrétním čidlem.



Obrázek 2.5: Diagram architektury kódu na microchipu ESP8266

2.2 Senzory

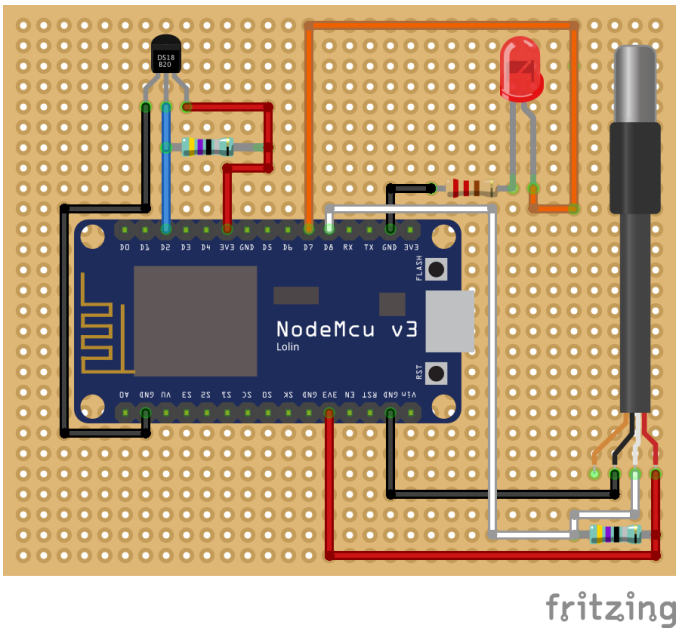
V projektu chytré domácnosti bylo použito celkem 7 senzorů pro měření fyzikálních veličin ...
Které veličiny měřeny (využití naměřených hodnot, smysl měřených veličin, ...)
Proč vybrány tyto konkrétní senzory

2.2.1 Teplotní čidlo DS18B20

esp-temphumid
Senzor pro měření teploty
Základní informace o senzoru - rozsah měření, přesnost, napájení, způsob komunikace, životnost, spolehlivost, ...

Více variant - vnitřní, venkovní
Obvod zapojení - esp, senzor, rezistor, napájení obvodu na PCB desku, umístění v místnosti
Zprovoznění komunikace s teplotními čidlem ds18b20 (room a outside)
Schéma zapojení senzoru, esp8266 a dalších periferií (led světlo, bzučák, ...)
Fotka reálného zkonstruovaného čidla

Senzor DS18B20 je jedním z nejdostupnějších senzorů použitých v projektu chytré domácnosti ...

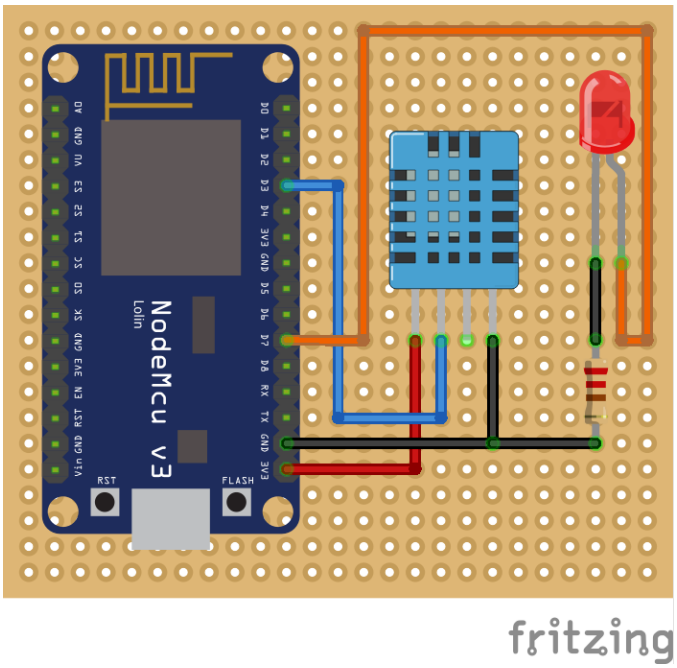


Obrázek 2.6: POPIS

2.2.2 Vlhkoměr DHT11

esp-temphumid
Senzor pro měření teploty a vlhkosti v místnosti
Základní informace o senzoru - rozsah měření, přesnost, napájení, způsob komunikace, životnost, spolehlivost, ...
Využití měřených veličin (používáme pouze naměřenou vlhkost, protože teplota má oproti ds18b20 nebo bme280 menší přesnost, ...)
Obvod zapojení - esp, senzor, rezistor, napájení obvodu na PCB desku, umístění v místnosti
Zprovoznění komunikace s vlhkostním čidlem dht11
Schéma zapojení senzoru, esp8266 a dalších periferií (led světlo, bzučák, ...)
Fotka reálného zkonstruovaného čidla

Senzor DHT11 je ...

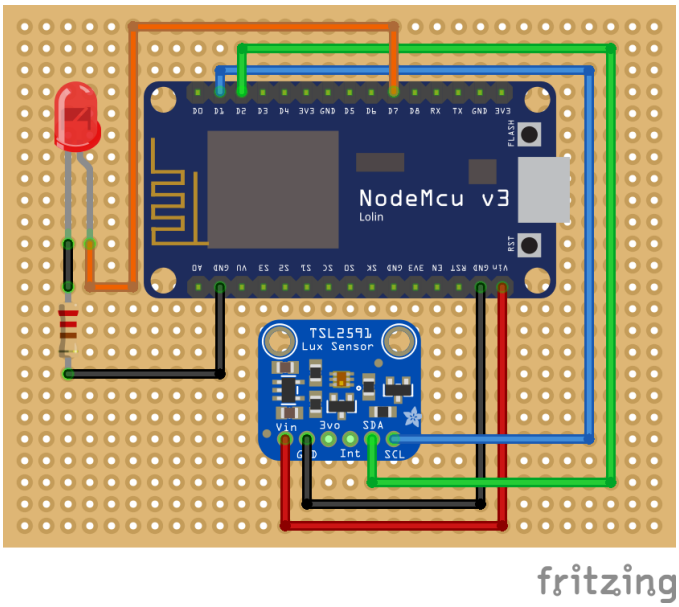


Obrázek 2.7: POPIS

2.2.3 Čidlo intenzity osvětlení TSL2591

esp-lux
Čidlo pro měření intenzity osvětlení
Základní informace o senzoru - rozsah měření, přesnost, napájení, způsob komunikace, životnost, spolehlivost, ...
Obvod zapojení - esp, senzor, rezistor, napájení obvodu na PCB desku, umístění v místnosti
Zprovoznění komunikace se senzorem tsl2591
Schéma zapojení senzoru, esp8266 a dalších periferií (led světlo, bzučák, ...)
Fotka reálného zkonstruovaného čidla

Senzor TSL2591 je ...

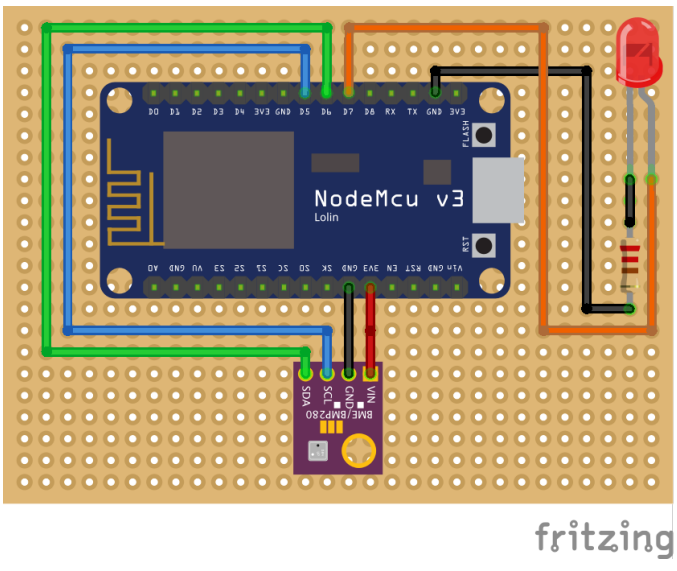


Obrázek 2.8: POPIS

2.2.4 Čidlo barometrického tlaku a teploty BME280

esp-pressure
Čidlo pro monitorování vnitřní teploty a barometrického tlaku
Základní informace o senzoru - rozsah měření, přesnost, napájení, způsob komunikace, životnost, spolehlivost, ...
Obvod zapojení - esp, senzor, rezistor, napájení obvodu na PCB desku, umístění v místnosti, alternativní senzory
Zprovoznění komunikace s čidlem bme280
Schéma zapojení senzoru, esp8266 a dalších periferií (led světlo, bzučák, ...)
Fotka reálného zkonstruovaného čidla

Senzor BME280 je ...

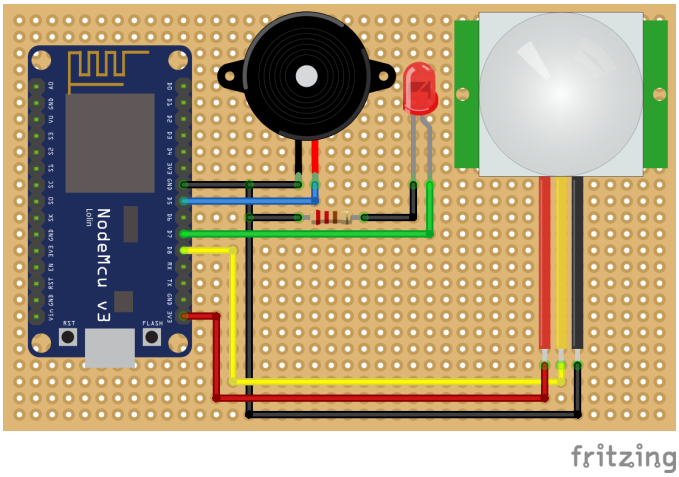


Obrázek 2.9: POPIS

2.2.5 Pohybové čidlo AM312

esp-pir
Čidlo pro monitorování pohybu v místnosti
Základní informace o senzoru - rozsah měření, přesnost, napájení, způsob komunikace, životnost, spolehlivost, ...
Obvod zapojení - esp, senzor, rezistor, napájení obvodu na PCB desku, umístění v místnosti, alternativní senzory - potíže s čidlem hc-sr501
Zprovoznění komunikace se senzorem am312
Schéma zapojení senzoru, esp8266 a dalších periferií (led světlo, bzučák, ...)
Fotka reálného zkonstruovaného čidla

Senzor AM312 je ...

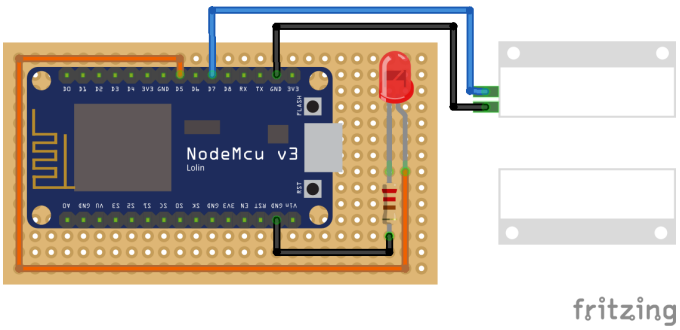


Obrázek 2.10: POPIS

2.2.6 Magnetické čidlo LS311B38

esp-magnet
Čidlo pro monitorování stavu dveří a oken
Základní informace o senzoru - rozsah měření, přesnost, napájení, způsob komunikace, životnost, spolehlivost, ...
Obvod zapojení - esp, senzor, rezistor, napájení obvodu na PCB desku, umístění v místnosti
Zprovoznění komunikace s magnetickým senzorem ls311b38
Schéma zapojení senzoru, esp8266 a dalších periferií (led světlo, bzučák, ...)
Fotka reálného zkonstruovaného čidla

Senzor LS311B38 je ...



Obrázek 2.11: POPIS

2.3 Raspberry Pi

Co je to Raspberry Pi
Výhody, nevýhody oproti klasickému pc (výpočetní výkon, spotřeba proudu, ...)
Využití v projektu chytré domácnosti - MQTT broker, databázový server, webserver, klasifikace příchozích zpráv
Výhody, nevýhody univerzálního použití Raspberry Pi pro několik účelů, ...

Kapitola 3

Síťová komunikace a databáze

Smyslem zkonstruovaných čidel je jejich snadná implementace do bytu či domu.
ESP8266 potřebuje jen napájení, celá komunikace mezi mikročipem a brokerem probíhá po WiFi

3.1 Protokol MQTT

Co je to protokol MQTT (vznik, využití)
Proč zrovna tento protokol
Princip použití protokolu MQTT (v síti je broker, který čeká na příchozí zprávy, ostatní zařízení jsou v módu publish - odesílají zprávy na broker, hierarchie zpráv, ...)
Zprovoznění komunikace přes MQTT
Sestavení struktury zpráv + identifikátorů čidel (popis jednotlivých atributů ve zprávě, proč jedno esp může posílat více zpráv do různých topiců, popis zvolené hierarchie topiců, ...)
ESP má aktuální čas pomocí NTP protokolu - díky tomu je možné přidávat do zpráv timestampy Popis logiky odesílání zpráv z ESP - ESP odešle zprávu periodicky vždy v polovině minuty (např.: 14:30:30, 14:31:30, ...) - lehké nepřesnosti způsobené dobou načítání dat ze senzorů, ...

3.2 Ukládání dat do databáze

Základní princip databáze, proč jsme se rozhodli pro použití databáze, ...
Co je MongoDB (určení, využití, funkčnost, ...)
Ukládání hodnot měřených veličin do MongoDB

3.3 Webserver

Popis backendu engine.py Backend - z hlediska programování - jeden soubor engine.py, který zajišťuje serverování webové stránky, ukládání dat do databáze, sensor check, ...

Kapitola 4

Diagnostika a detekce anomálií

Proč řešit diagnostiku čidel v rámci chytré domácnosti, ...
Úvod do detekce anomálií, základní principy, smysl využití, ...
Diagnostika a detekce anomálií v projektu chytré domácnosti probíhá na 3 úrovních ...

4.1 Detekce chyb na úrovni ESP8266

První a nejnižší úroveň diagnostiky jednotlivých senzorů
Implementace detekce chyb na úrovni samotného microchipu
Popis základního principu - při načítání ze dat ze senzoru čidlo pozná, když je senzor nefunkční - nevrací žádné hodnoty a ESP pošle info o nefunkčním senzoru)
Robustnost - odolnost ESP vůči výpadkům senzorů - esp nespadne kvůli nefunkčnímu senzoru, jen změní status zprávy, ...
Nonstop provoz - senzory lze k ESP připojovat a odpojovat v reálném čase bez nutnosti vypínání nebo restartu microchipu, pokud senzor v jakémkoliv časovém okamžiku odpojím - změní se status zprávy na "error", ve chvíli kdy senzor zase připojím na desku - esp začne měřit a posílat, ...

4.2 Detekce anomálií na základě klasifikace

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 anomálií 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

Implementace detekce anomálií na úrovni serveru (brokeru)
Implementace funkce sensor-check()
Periodická kontrola odesílání zpráv jednotlivých čidel - pokud čidlo z neznámého důvodu neodešle zprávu nebo pokud se zpráva nepřenese k brokeru - informace o anomálii se přenese do webového rozhraní
Vysvětlení stavů čidel - 3 možné stavy čidla - ok, value-error, error

Závěr diagnostiky: kontrola stavu jednotlivých čidel + kontrola věrohodnosti posílaných hodnot + kontrola pravidelnosti odesílaných zpráv

Kapitola 5

Webové rozhraní

Podle toho jak to půjde.. Kapitola 5 Interakce systému z uživatelem - Webové rozhraní - Hlasové ovládání

Důvod využití webové vizualizace
Frontend - popis struktury webu - proč 3 záložky, popis Overview, Analytics, About, ...
Smysl Overwiev - nonstop zobrazení dat na monitoru, ...
Přijímání aktuálních zpráv z ESP bez nutnosti refresh stránky přes websockets - co je websockets, důvod využití této technologie, alternativa v podobě json souboru - nevýhody, ...

Zajištění integrity celého webu (nově přijaté zprávy se propisují na web, po znovunačtení se ihned ukazují poslední přijaté hodnoty - má smysl hlavně u neperidicky posílaných veličin, ...)

Kapitola 6

Conclusion

Conclusion and Discussion

Conclusion text...

6.1 Future Work

Outlook...

Příloha A1

Structure of the Workspace