Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Vzdálený monitoring venkovního prostředí

Vojtěch Hudec

Bakalářská práce
2022

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Vojtěch Hudec**
Osobní číslo:	**I18130**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Vzdálený monitoring venkovního prostředí**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Monitorování stavu venkovního prostředí potřebujeme provádět z mnoha důvodů ať už se jedná o osobní komfort, tedy vliv počasí, kdy nás zajímá teplota, vlhkost, případně intenzita větru, nebo o kvalitu prostředí, kdy nás může zajímat intenzita záření/UV záření, prašnost, případně jiné veličiny. Klasická dostupná řešení nabízí možnost měření vybraných veličin, ale většinou neumožnují vzdálený přístup, nebo můžeme využít služeb třetích stran, které neměří veličiny přímo v bodě zájmu. Cílem práce je navrhnout stanici pro měření vybraných meteorologických případně environmentálních veličin s možností vzdáleného přístupu k naměřeným datům. Praktická část by měla obsahovat řešení měřící stanice, sw řešení vzdáleného přístupu, data by měli být ukládány mimo stanici na vzdáleném serveru. Součástí řešení by měla být základní vizualizace dat. Teoretická část práce bude obsahovat stručný popis měřených, nebo měřitelných veličin s ohledem na jejich elektrická měření. Dále bude obsahovat rozbor možných řešení pro komunikaci stanice se serverem, rozbor použitelných protokolů a implementace z pohledu náročnosti implementace na straně klienta/serveru (http –get, MQTT…).

Rozsah pracovní zprávy: **30-60**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

[1] VÁŇA, V. Mikrokontroléry ATMEL AVR: popis procesoru a instrukční soubor. Praha: BEN technická literatura, 2003.336 s. ISBN 978-80-7300-083-0.
[2] VÁŇA, V. Mikrokontroléry ATMEL AVR: programování v jazyce C. Praha: BEN technická literatura, 2003. 216 s. ISBN 978-80-7300-102-0.
[3] VLACH, J. Řízení a vizualizace technologických procesů. Praha: BEN technická literatura, 2002. 160 s. ISBN 978-80-86056-66-X.
[4] BRTNÍK, B. Základní elektronické obvody. Praha:BEN technická literatura, 2011. 156s. ISBN 978--80-7300-408-8
[5] RIPKA, P.; TIPEK, A. Master Book of Sensors. Praha : BEN, 2003. ISBN 0-12-752184

Vedoucí bakalářské práce: **Ing. Pavel Rozsíval**
Katedra elektrotechniky

Datum zadání bakalářské práce: **31. října 2020**
Termín odevzdání bakalářské práce: **14. května 2021**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 26. února 2021

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 14. 8. 2022

Vojtěch Hudec

ANOTACE

Bakalářská práce se zaměřuje na monitorování stavu venkovního prostředí. Monitorování provádíme z mnoha důvodů ať už se jedná o osobní komfort, tedy vliv počasí, kdy nás zajímá teplota, vlhkost, případně intenzita větru, nebo o kvalitu prostředí, kdy nás může zajímat intenzita záření/UV záření, prašnost, případně jiné veličiny. Klasická dostupná řešení nabízí možnost měření vybraných veličin, ale většinou neumožňují vzdálený přístup. Případně můžeme využít služeb třetích stran, které neměří veličiny přímo v bodě zájmu. Cílem práce je navrhnout stanici pro měření vybraných meteorologických případně environmentálních veličin s možností vzdáleného přístupu k naměřeným datům.

KLÍČOVÁ SLOVA

Vzdálený monitoring, vzdálený přístup, meteostanice, měřící stanice, měření environmentálních veličin, vizualizace dat, ukládání na vzdálený server, počasí, teplota, stav venkovního prostředí

TITLE

Weather station with remote access

ANNOTATION

The bachelor thesis focuses on monitoring the state of the outdoor environment. We perform monitoring for many reasons, whether it is personal comfort, we are interested in temperature, humidity, or wind intensity, or the quality of the environment, when we may be interested in radiation intensity / UV radiation, dust, or other quantities. Classic available solutions offer the possibility of measuring selected quantities, but usually do not allow remote access, or we can use the services of third parties that do not measure quantities directly at the point of interest. The purpose of the work is to design a station for measuring selected meteorological or environmental quantities with the possibility of remote access to the measured data.

KEYWORDS

Remote monitoring, remote access, weather station, measuring station, measurement of environmental variables, data visualization, storing data on a remote server, weather, temperature, the state of the outdoor environment

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK

ACK	Acknowledgement
API	Application Programming Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I^2C	Inter Integrated Circuit
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MQTT	Message Queuing Telementry Transport
PCB	Printed Circuit Board
REST	Representational State Transfer
ROM	Read Only Memory
SOAP	Simple Object Access Protocol
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SSID	Service set Identifier
TCP	Transmission Control Protocol
XML	Extensible Markup Language

ÚVOD

Lidé už od pradávna sledují počasí, protože je nedílnou součástí života a je nutné se mu přizpůsobit. Nejčastěji sledují teplotu, podle které si vybírají oblečení, ve kterém vyjít ven. Dříve za oknem umístěný rtuťový nebo bimetalový teploměr nahradily modernější meteostanice nebo alespoň digitální teploměry. Ty jsou v dnešní době součástí téměř každé domácnosti. Typická meteostanice se běžně skládá z venkovní a vnitřní jednotky. Venkovní jednotka je určená pro měření teploty a vlhkosti, méně často měří i rychlost a směr větru. Naměřené hodnoty jsou přenášeny bezdrátově do vnitřní jednotky, která přijaté hodnoty zobrazuje a také často sama měří teplotu a vlhkost v jejím okolí, případně i tlak. Díky tomu je uživatel schopen získat přesnější údaje o počasí nezávisle na globální předpovědi.

K vypracování bakalářské práce je zapotřebí prostudovat problematiku meteorologie, běžně měřené veličiny a principy senzorů k měření těchto veličin určených. Nasbírané informace využít při návrhu a realizaci zařízení a poté ověřit funkčnost celého systému.

Cílem bakalářské práce je navrhnout meteostanici, která autonomně v pravidelných intervalech odesílá zaznamenané hodnoty na databázový server. V aplikaci by měl být uživateli poskytnut přístup nejen k aktuálním hodnotám, ale také jejich minimu, maximu a trendu za určité období. Pro lepší přehled bude možné zobrazit historii hodnot za určité období v podobě grafu. Tyto informace poskytnou uživateli kvalitnější a přesnější přehled o počasí ve sledované oblasti.

1 METEOROLOGIE

Hlavním cílem meteorologie, jakožto vědy, je zkoumání zemské atmosféry. Studuje a popisuje složení, stavbu, vlastnosti ale také jevy a děje v atmosféře. Mimo jiné se zabývá i měřícími přístroji. Ty využívá ke sběru dat, pomocí kterých vyhotovuje předpovědi počasí. Výsledky aplikuje v zemědělství, lékařství ale i letectví nebo například námořnictví. Mezi nejobvyklejší sledované veličiny patří teplota, tlak, vlhkost vzduchu, oblačnost, vítr nebo například srážky. První dochované pravidelné záznamy počasí pochází z roku 1337. Meteorologie se však začala rozvíjet až v 17. století, kdy byly vynalezeny potřebné přístroje jako je teploměr a tlakoměr. [1]

1.1 Meteorologická stanice

Zpravidla obsahuje teploměry, extrémní teploměry, vlasový vlhkoměr, termograf a hydrograf. Data se získávají odečítáním hodnot z umístěných zařízení, a to zpravidla v dohodnutých hodinách. Tyto hodnoty jsou dále zpracovávány a slouží k diagnóze a předpovědi počasí. K co nejpřesnější předpovědi je zapotřebí udržovat síť měřících bodů rozmístěnou po celém sledovaném území. Za zřizování a udržování těchto stanic v České republice je zodpovědný Český hydrometeorologický ústav. Meteostanice nemusí být vždy pevně umístěná na konkrétní místě. Běžné jsou i meteostanice na lodích, v letadlech nebo družicích. Mohou být plně automatické nebo vyžadovat součinnost obsluhy. Ta může být profesionální nebo dobrovolnická. [2]

1.2 Měření vybraných meteorologických veličin

1.2.1 Teplota vzduchu

Přístroj určený k měření teploty se nazývá teploměr. Pro co nejpřesnější měření musí být umístěn ve volném prostoru, chráněn před deštěm, sněhem i přímým slunečním zářením. Ideální umístění teploměru je v meteorologické budce ve výšce dvou metrů. Mezi sledované hodnoty se mimo aktuální teplotu řadí i extrémy. Tedy minimální a maximální teplota. Také můžeme sledovat přízemní teplotu ve výšce pěti centimetrů. Pro záznam průběhu naměřených hodnot je určeno zařízení zvané termograf. Na vývoj teploty má vliv nejen roční období. V běžných podmínkách teplota klesá se zvyšující se výškou. Při jasné obloze se v noci půda rychleji ochlazuje a tím se ochlazuje i vzduch v přízemní vrstvě. Tím dochází k inverzi, kdy je teplota vzduchu ve vyšší výšce vyšší. [3]

1.2.2 Vlhkost vzduchu

Vlhkostí vzduchu se rozumí poměr vodní páry ve vzduchu. Ve značné míře ovlivňuje počasí, srážky a oblačnost. Vlhkost vzduchu můžeme vyjádřit několika způsoby. Mezi nejběžnější z nich patří hmotnost vodní páry v určitém objemu nebo schopností přijímat další vodní páru. Měření se provádí psychrometry umístěnými v meteorologické budce. Vlhkost se určuje rozdílem teplot odečtených z dvou teploměrů. Jeden z nich je obalený tkaninou namočenou ve vodě. Čím sušší vzduch je, tím rychleji se voda z tkaniny vypařuje a tím nižší je teplota, který tento teploměr ukazuje. Vlhkost lze měřit i napřímo s omezenější přesností pomocí vlasového vlhkoměru. Využívá faktu, že lidský vlas zbavený tuku mění při změně okolní vlhkosti vzduchu svou délku. Tato změna však není dostatečně patrná, proto je obvykle vlas veden přes kladku, ke které je připevněna delší ručka. Ta na stupnici zobrazuje relativní vlhkost vzduchu v procentech. Vlhkost vzduchu má opačný chod než denní teplota. Nejvyšší vlhkost lze naměřit před východem slunce a nejnižší mezi čtrnáctou a šestnáctou hodinou. [4]

1.2.3 Atmosférický tlak

Tíha vzduchového sloupce od sledované hladiny až po horní hranici atmosféry se nazývá atmosférický tlak. Měří se v pascalech. Je ovlivňován teplotou i vlhkostí vzduchu, nadmořskou výškou a zeměpisnou šířkou. Z hlediska porovnávání tlaků je potřeba přepočítat údaje na 0 °C a nadmořskou výšku 0 metrů nad mořem. K měření se používají rtuťové tlakoměry. Z toho důvodu se dříve atmosférický tlak udával jednotkou milimetr rtuťového sloupce. Mezi modernější zařízení se řadí hypsometr nebo barograf. Naměřená data můžeme použít ke krátkodobé předpovědi počasí v místě měření. Atmosférický tlak se snižuje v závislosti na zvyšující se nadmořské výšce. Tento stav je zapříčiněn tím, že nad daným místem leží nižší vzduchový sloupec. Jevu je využíváno také při měření nadmořské výšky. [5]

1.2.4 Intenzita osvětlení

Intenzitou osvětlení je označován světelný tok, měřený v lumenech, který dopadá na plochu o velikosti jednoho metru čtverečního. Měření je vyjadřováno jednotkou zvanou Lux. I přesto, že se jedná o fyzikální, a ne meteorologickou veličinu, poskytuje nám představu o aktuálních povětrnostních podmínkách v místě měření. Jasné sluneční světlo má intenzitu přes sto tisíc luxů, v typický zatažený den lze běžně naměřit tisíc luxů, při úplně zatažené obloze však intenzita klesne až ke čtyřiceti luxům a při měsíčním svitu je intenzita nižší než jeden lux. [6]

2 HARDWARE

Kapitola se věnuje popisu jednotlivých komponent zvolených pro konstrukci měřícího zařízení. Při výběru modulů a senzorů byla zohledněna kvalita, přesnost měření, jejich rozlišení a cena.

2.1 Vývojová deska ESP32

Jedná se o všestranný mikrokontroler cílený především na vývoj IoT aplikací, a to zejména díky vestavěnému Wi-Fi modulu. Můžeme ho považovat za následníka neméně populárního modulu ESP8266, který způsobil revoluci na trhu a za velmi příznivou cenu umožnil nadšencům vytvářet projekty připojené k internetu. ESP32 je sice následníkem, ale ne však pokračovatelem linie. Oproti zmíněnému ESP8266 nabízí bohatší možnosti Wi-Fi, podporu Bluetooth, více digitálních vstupů a výstupů, a především větší výkon. Také se při konstrukci opustilo od způsobu komunikace v režimu modemu, se kterým se pracovalo pomocí AT příkazů. [7]

2.1.1 Specifikace

- Procesor: Tensilica Xtensa 32-bit LX6
    - Frekvence: 240 MHz
- Bezdrátové připojení:
    - WiFi: 802.11 b/g/n/e/i až do rychlosti 150 Mbit/s
    - Bluetooth: v4.2 BR/EDR a podpora Bluetooth Low Energy
- Paměť:
    - ROM: 448 KiB
    - SRAM: 520 KiB
    - Flash: až do velikosti 16 MB v závislosti na verzi [8]

Při výběru vývojové desky byla zvažována i varianta, která počítala s využitím desky Arduino. Hlavní motivací byla nativní podpora v Arduino IDE. Všechny ostatní aspekty však hrály ve prospěch ESP32. Poskytuje vyšší výkon a přímo v sobě integruje Wi-Fi modul. V případě Arduina by bylo zapotřebí konektivitu vyřešit Ethernet Shieldem, zaintegrovat ESP8266 nebo data posílat přes mobilní signál za použití modulu SIM800L. Při sečtení nákladů vycházelo v době rozhodování ESP32 o poznání příznivěji.

2.2 Senzory

Senzor je součástka, která měří veličiny fyzikálního nebo technického charakteru. Jejich hodnotu reprezentuje signálem, nejčastěji jako průběh napětí nebo proudu, který je přenositelný a zpracovatelný v řídícím nebo měřícím systému.

2.2.1 Senzor AHT10

Digitální teplotní a vlhkostní senzor od výrobce Asair. Mezi hlavní přednosti patří malé rozměry, nízká spotřeba a vysoká přesnost měření. Senzor kaliruje přímo v průběhu výroby a při měření teploty dosahuje přesnosti s odchylkou 0,3 °C v rozmezí od 0 °C do 55 °C. Senzor je schopný provádět měření od -40 °C až po 85 °C. Mimo ideální spektrum se však odchylka měření zvyšuje až k 1 °C. Při měření vlhkosti se za optimální oblast považuje 20 % až 80 %, kde senzor poskytuje přesnost 2 %. V jeho možnostech je však měření celé spektra od 0 % až po 100 %. Odchylka se mimo optimální oblast zvedne v nejhorším případě až na 3 %. Senzor pracuje v rozmezí napětí 1,8 V až 3,6 V. V průběhu měření k provozu potřebuje 23 µA a při spánku 0.25 µA. Komunikace probíhá přes sériovou sběrnici I^2C. [9]

Konkurujícím modulem byl mimo jiné DHT22. Za vyšší cenu však poskytuje horší parametry. Při měření teploty může docházet k odchylce 0,5 °C a při měření vlhkosti je udávaná tolerance v rozpětí 2 % až 5 %. Dalším důležitým parametrem sledovaným při porovnávání je proudová náročnost, kdy DHT22 při měření požaduje 2.5 mA. [10]

Pro Aruino language je k dispozici knihovna od společnosti Adafruit. Součástí je vzorový příklad, jak navázat komunikaci a odečíst data z modulu. V prvé řadě je zapotřebí naimportovat zmíněnou knihovnu, inicializovat jí vytvořením objektu a zahájit komunikaci funkcí begin(). Výsledkem metody je informace, zda se komunikaci podařilo navázat. Ve funkci loop jsou periodicky čteny a zobrazovány hodnoty z modulu. Lehce modifikovaná forma je využita ke čtení dat i v implementaci zařízení.

```
#include <Adafruit_AHT10.h>
Adafruit_AHT10 aht;
void setup(){
  Serial.begin(115200);
  if (!aht.begin()) {
    Serial.println("Sensor not found");
    while(1);
  }
}
void loop(){
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);
  Serial.println(temp.temperature);
  Serial.println(humidity.relative_humidity);
  delay(500);
}
```

Zdrojový kód 1: Čtení dat ze senzoru AHT10

2.2.2 Senzor BMP280

Senzor od společnosti Bosch, určený k měření atmosférického tlaku. Měření je schopen provádět v rozmezí 300 hPa až 1100 hPa s odchylkou 0.12 hPa. Měření tlaku je prováděno piezorezistory, na které se přenáší deformace membrány, způsobená atmosférickým tlakem. Při vzorovací frekvenci 1 Hz senzor požaduje 2.7 µA. Maximální frekvence odečítání hodnot je 157 Hz. Teplotní provozní podmínky výrobce udává od -40 °C až po 85 °C. Komunikace probíhá přes sběrnici I^2C, kdy je možné data přenášet se vzorkovací frekvencí až 3,4 MHz. Při komunikaci přes sběrnici SPI zvládne senzor komunikovat rychleji, a to až do vzorkovací frekvence 10 MHz. Při výběru byl zvažován i senzor BMP180 od stejného výrobce. Ten má ale vyšší nároky na proud a nižší rozlišení. [11]

2.2.3 Senzor VEML7700

Senzor z dílny Vishay Semiconductors měří intenzitu osvětlení pomocí fotodiody s vysokou přesností. Je možné měřit hodnotu až do 120 000 Lux. Udávané rozlišení je 0,0036 Lux. Za vhodné vstupní napětí se považuje rozmezí 2,5 V až 3,6 V a proudová náročnost se v závislosti na zvoleném pracovním módu může vyšplhat až na 45 µA. Za vhodné provozní podmínky se považuje teplota od -25 °C do 85 °C. Přenos dat zajišťuje sběrnice I^2C. [12]

Obrázek 1: Modul s VEML7700 [13]

Za podobnou cenu je k dostání senzor MAX44009. Fotodioda použitá na tomto modulu má nižší rozměry, ale rozlišení je 0,045 Lux, tedy nižší než u nakonec zvoleného modulu. Jelikož rozměr nebyl při výběru hlavním kritériem, nebyl důvod přiklonit se k této variantě. [14]

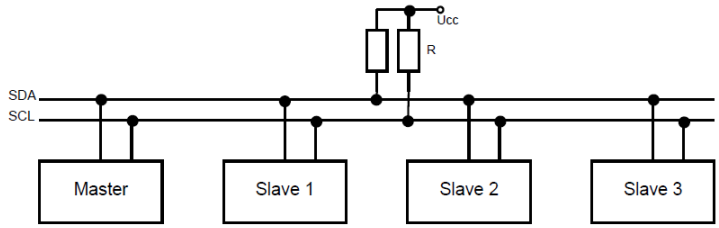
2.3 Přenos dat

V této kapitole budou popsány způsoby komunikace použité při konstrukci měřícího zařízení.

2.3.1 I^2C

Jedná se o komunikační sběrnici od společnosti Philips. Komunikace probíhá sériově po dvou vodičích, kdy jeden je určený pro přenos dat, ten je označovaný jako SDA – Serial Data. Druhý vodič s označením SCL – Serial Clock se využívá pro přenos hodinového signálu. Z důvodu koncipování sběrnice jako master – slave je nutnost řešení arbitrace. Adresování je řešeno vestavěným systémem, kdy má každé zařízení unikátní adresu o délce 7 až 10 bitů. Maximální počet zařízení připojených do sběrnice je 128. Za využívání sběrnice nejsou účtovány žádné poplatky, ale přidělování unikátních adres zařízením zpoplatněno je.

Pokud neprobíhá komunikace, oba vodiče jsou v úrovni high, tedy logická 1. Změna hodnoty na vodiči SDA může probíhat pouze v případě, že vodič SCL je v úrovni low. Zde platí výjimka při vysílání podmínek START a STOP, které slouží k zahájení arbitrace a ukončení komunikace. Pokud zařízení zachytí podmínku START, porovná svou adresu s adresou vysílanou na sběrnici. Zařízení, které vyhodnotí shodu, potvrdí komunikaci bitem ACK. [15]



Obrázek 2: Typické zapojení sběrnice I²C [16]

2.3.2 SPI

SPI neboli Serial Peripheral Interface je sériová sběrnice typu master – slave. Ke komunikace jsou použity čtyři vodiče. Před začátkem master nastaví logickou 0 skrz vodič CS – Chip Select na zařízení, se kterým chce začít komunikaci. Na vodič SCLK – Serial Clock začne generovat hodinový signál. V tuto chvíli zahájí obě zařízení přes zbylé dva vodiče komunikaci a přenos dat. Směrem z master proudí data vodičem MOSI – Master Out, Slave In a vybrané zařízení přenáší data vodičem MISO – Master In, Slave Out. Délka vysílaných dat může být 8 nebo 16 bitů. Komunikace končí zastavením vysílání hodinového signálu a nastavení logické 1 na CS. [17]

2.3.3 Wi-Fi

Wi-Fi je síťová technologie, která umožňuje bezdrátový přenos dat napříč zařízeními. Určitá zařízení síť vytváří a ostatní se k této síti připojují. Nejčastěji se technologie používá k internetovému připojení. Nejběžnější klientská zařízení jsou notebooky, mobilní telefony, tablety nebo i prvky IoT. Síťové komponenty jsou popsány standardy IEEE 802.11. Liší se frekvencí, šířkou pásma, rychlostí přenosu nebo počtem dostupných kanálů. Frekvence vysílaného signálu jsou buď 2.4 GHz nebo 5 GHz. Výhodou nižší frekvence je lepší pokrytí plochy signálem a snadnější prostup skrze objekty jako jsou například zdi. Obvyklý dosah signálu je 30 m. Vyšší frekvence umožňuje rychlejší přenos dat. Většina moderních Wi-Fi routerů šíří signál na obou frekvencích jako dvě nezávislé sítě a uživatel, pokud to jeho zařízení podporuje, si může vybrat ke které z nich se připojí podle jeho preferencí rychlosti nebo dosahu signálu. [18]

Modul ESP32 podporuje standardy 802.11 b/g/n/e/i. Nejrychlejším typem je 802.11n. Za ideálních podmínek dosahuje rychlosti 600 Mbps, běžně však 100 Mbps na frekvenci 2.4 GHz.

Pro použití Wi-Fi v zařízení je dostupná knihovna WiFi.h, obsahující všechny potřebné funkce. Funkce begin přijímá dva parametry, prvním je SSID sítě, ke které se má zařízení připojit

a druhým parametrem je heslo. Po zavolání této funkce se zařízení pokouší o připojení a podle potřeby aktualizuje hodnotu status. Ve chvíli, kdy je status roven konstantě WL_CONNECTED, je zařízení připojeno k síti.

```
1  #include <WiFi.h>
2  const char* ssid = "SSID";
3  const char* pass = "PASSWORD";
4  void setup(){
5    Serial.begin(115200);
6    WiFi.begin(ssid, pass);
7    Serial.printf("Connecting to %s ", ssid);
8    while(WiFi.status() != WL_CONNECTED){
9      delay(1000);
10     Serial.print(".");
11   }
12   Serial.println("CONNECTED");
13 }
```

Zdrojový kód 2: Připojení k Wi-Fi síti

3 SOFTWARE

3.1 Vývojová prostředí

3.1.1 Arduino IDE

Open source vývojové prostředí, primárně určené pro desky Arduino. Avšak díky velké komunitě okolo ESP32 a ESP8266 zanedlouho po jejich uvedení na trh vznikly knihovny pro toto vývojové prostředí. Hlavní funkcionality jsou textový editor kódu, kompilování a nahrávání programů do vývojových desek. Jazyk používaný pro programování vývojových desek vychází z programovacích jazyků C a C++. Tato nádstavba, zvaná Arduino language, je rozšířena o specifické metody a funkce. Každý program musí obsahovat funkci setup(), která je vykonána po spuštění zařízení. Slouží k nastavení počátečních hodnot, stavů a inicializaci. Funkce loop() obsahuje logiku programu. Tato funkce se vykonává ve smyčce. Vývojové prostředí obsahuje množství knihoven a vzorové příklady k nim. V menu je k dispozici správce, skrz kterého lze instalovat další externí knihovny pro moduly, senzory a čidla. [19]

3.1.2 IntelliJ IDEA

Vývojové prostředí pro JVM jazyky. Hlavním cílem vývojářů je co nejvíce zvýšit produktivitu práce při psaní kódu. Zvládne nahradit rutinní operace a tím šetřit čas, obsahuje chytré dokončování kódu a má pokročilé možnosti analýzy a refaktorování kódu. I přesto, že primárně cílí na JVM jazyky jako je Java, Kotlin, Scala nebo Groovy, podporuje programování i v ostatních populárních jazycích. Například Ruby, PHP nebo JavaScript. Mimo to nabízí také funkčnosti pokročilého textového editoru a zvýrazňování pro populární typy souborů jako JSON, XML a HTML. K dispozici je ve třech verzích. Edu obsahuje vestavěné lekce a výukové materiály Javy, Kotlinu a Scaly. Verze Community Edition je k dispozici zdarma pro JVM jazyky a vývoj pro Android. Nejvyšší verze Ultimate je zpoplatněna, ale nabízí přístup ke všem funkcím. Mezi nejzajímavější z nich patří vývoj pro jiné než JVM jazyky, podpora frameworku, aplikačních serveru a integrované nástroje ke správě databází. [20]

3.2 Roto 3T

Grafický správce Mongo databáze. Jedná se o open source nástroj, poskytující inovativní funkce pro nadšence i širší komunitu. Nabízí přístup k MongoDB shellu, analyzuje správnost psaného dotazu a je schopen psaný dotaz dokončovat. Mezi samozřejmost patří práce s dokumenty jako je přidávání, mazání, upravování a vyhledávání podle specifických kritérií a jejich následné řazení. [21]

3.3 Insomnia

API klient určený primárně k odesílání REST dotazů. Podporuje však i jiné API založené na HTTP protokolu. Například SOAP, GraphQL a GRPC. Mezi hlavní přednosti patří možnost udržování dotazů v kolekcích, podpora množství způsobů autentizací, včetně nejběžnějších tokenů nebo podpora více konfigurací pro různá prostředí. [22]

3.4 Docker

Zajišťuje virtualizaci a izolaci procesů, zvaných kontejnerů. Běžná virtualizace funguje na principu modelu hardware – hostitelský OS – hypervisor – klientský OS – aplikace. V případě Dockeru jsou všechny kontejnery spuštěny nad jednou linuxovou instancí. Výsledkem je model hardware – OS – docker engine – aplikace. Z hlediska výkonu virtualizace je tento přístup bezesporu efektivnější. Hlavní myšlenkou je snadnost nasazování aplikací. Ta se nepopiratelně propisuje do jeho uživatelského a API rozhraní, do celé filozofie i dokumentace. Zajímavou funkcí pro vývojáře je automatické sestavení aplikace. [23]

3.5 Použité technologie

3.5.1 Java

Objektově orientovaný jazyk navrhnutý společností Sun Microsystems v roce 1995. Primárním cílem bylo snížit implementační závislosti. Mezi hlavní přednosti patří rychlost, bezpečnost a spolehlivost. Je hojně využívaná v aplikacích pro desktop, herní konzole, mobilní telefony a své uplatnění najde i na superpočítačích a cloudu.

Java využívá své vlastní běhové prostředí zvané Java Virtual Machine. Stará se o překlad Java bytecodu do strojového jazyka v závislosti na prostředí, na kterém je spouštěné. Tímto způsobem je zajištěna nezávislost na cílovém systému, pro který se aplikace vyvíjí. JVM v sobě má vestavěný kompilátor typu just in time. To znamená, že je zkompilovaná pouze taková část aplikace, která je právě nutná k běhu. O to se stará běhové prostředí, které je součástí Java Runtime Environment. To v sobě mimo jiné obsahuje knihovní třídy a zavaděče. Pro vývoj v Javě je nutné mít nainstalovaný Java Development Kit. Součástí je především kompilátor a Java application launcher. [24]

3.5.2 Spring Boot

Spring Boot usnadňuje vytváření aplikací založených na Spring Frameworku. Obsahuje webový server Tomcat a také pomáhá s konfigurací závislostí a knihoven.

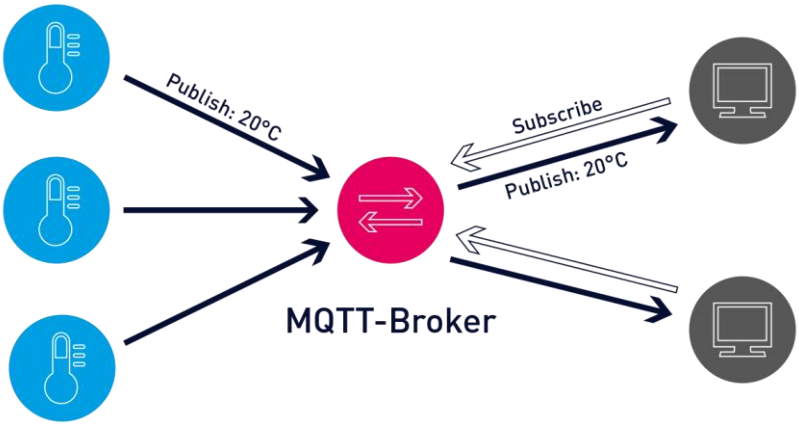
Spring Framework má za cíl usnadnit vývoj aplikace a dopřát programátorovi větší prostor pro soustředění se na implementaci funkčních požadavků. Nejdůležitější pro potřeby této práce je vyřešení přístupu do databáze, vlastní webový server a zprostředkování REST volání. [25]

3.5.3 React

Knihovna JavaScriptu určená k vytváření uživatelských rozhraní ve webovém prohlížeči. Název vychází z hlavní myšlenky celého Reactu. Reaguje na změny stavů a tím efektivně aktualizuje a překresluje pouze komponenty, ve kterých došlo k těmto změnám. Komponenty jsou základem celé knihovny. Jedná se o nezávislé a znovupoužitelné části kódu, které je možné integrovat do dalších, větších, obalových komponent a tím tvořit rozsáhlé a komplexní uživatelské rozhraní. I přesto, že React je knihovna pro uživatelské rozhraní, podporuje uvnitř svého kódu Node.js, který vzhledu dodá i potřebnou logiku aplikace. [26]

3.6 Příjem dat

Předávání zpráv mezi zařízením a obslužnou aplikací je zajištěno pomocí MQTT. Jedná se o protokol určený k předávání zpráv. Díky jeho nenáročnosti a jednoduchosti se perfektně hodí i pro zařízení s nízkým výpočetním výkonem, jako jsou různé IoT senzory nebo zařízení limitovaná velikostí a spotřebou. Přenos probíhá přes TCP podle návrhového vzoru publisher – subscriber. Srdcem je centrální bod, zvaný broker. Jeho cílem je zachytávat zprávy od publisherů a předávat je subcriberům. Filtrování zpráv zajišťuje rozdělení do topiců. To znamená, že subscriber obrdží pouze takové zprávy, jejichž topicům nebo jejich nadřazeným celkům naslouchá. Topicy jsou hierarchicky oddělené lomítky podobně, jako jsou uspořádaný složky na pevném disku počítače. Pomocí zástupných znaků může subscriber naslouchat více topicům současně. Znak + nahrazuje jednu úroveň. Znak # jednu nebo více úrovní, proto pokud je přítomný, musí být vždy až na konci řetězce. Nejčastěji zprávy obsahují data ve formátu JSON nebo se jedná o prostý text. Formát dat ale protokol specificky neudává a je pouze na uživateli, jaká data pošle. Jediné omezení je velikost zaslínaých dat, kdy maximální objem je 256 MB. Dostupné jsou 3 úrovně zpráv zvané Quality of Service. At-most-once má za cíl ohlídat, že byla zpráva odeslána, ale už neověřuje, jestli byla doručena. At-least-once hlídá jak odeslání zprávy, tak i její doručení. Nejvyšší úroveň zajistí exaktní doručení zprávy přesně jednou. [27]



Obrázek 3: Komunikace mezi zařízeními pomocí MQTT [28]

V zařízení je odesílání dat zajištěno knihovnou PubSubClient. Při inicializici je zapotřebí poskytnout isntanci klienta, který zajišťuje přístup k síti. Zpravidla se jedná o instanci třídy WifiClient nebo EthernetClient. Dále je zaptořebí poskytnout adresu a port MQTT serveru a tím se klient stane připravený na odesílání zpráv použitím metody publish. V této metodě je v prvním parametru specifikovaný topic, na který chceme zprávu odeslat a druhým paraemtrem je samotná zpráva. Před tím je zapotřebí ověřit, že existuje spojení s brokerem a pokud ne, připojení provést.

```
1   #include <WiFi.h>
2   #include <PubSubClient.h>
3   WiFiClient wifiClient;
4   PubSubClient client(wifiClient);
5   void setup() {
6     WiFi.begin(ssid, password);
7     client.setServer(server, 1883);
8   }
9
10  void loop() {
11    if (!client.connected()) {
12      reconnect();
13    }
14    client.publish(topic, message);
15  }
```

Zdrojový kód 3: Odeslání dat přes MQTT

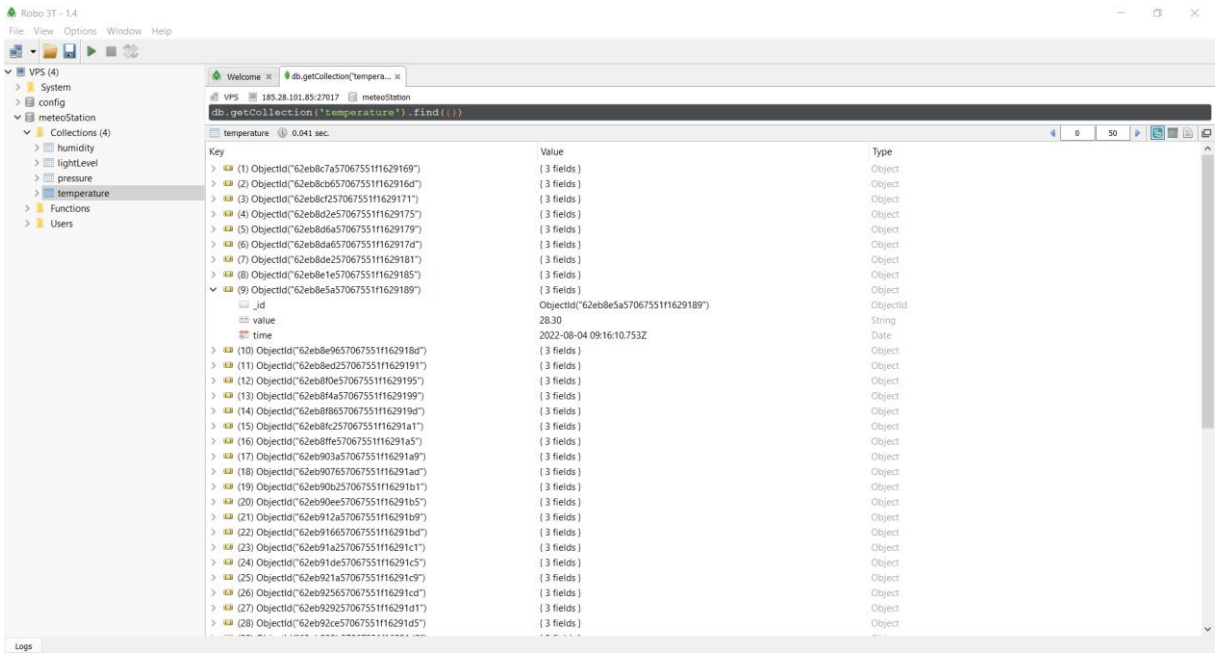
3.7 Persistence dat

Zachycená data ze zařízení je zapotřebí ukládat do databáze pro pozdější analýzu a zobrazení historie. Kvůli povaze dat a budoucí práci s nimi byla zvolena nerelační databáze. Tento typ databází neuchovává data v pevně definovaných tabulkách. Nejjednodušším typem databáze je

klíč – hodnota, kdy každá uložená hodnota má svůj unikátní klíč. Avšak rozšířenější je dokumentově orientovaný typ nerelačních databází. Nejčastěji jsou dokumenty reprezentovány takovým formátem, který opět pracuje na principu klíč – hodnota. Například JSON nebo XML. Obecně jsou nerelační databáze lépe škálovatelné a flexibilnější. Naopak relační databáze disponují vysokou integritou dat a konzistencí. [29]

Jako databáze byla vybrána MongoDB. Od roku 2009 je vyvíjena pod licencí open source a postupem času se stala nejpopulárnější NoSQL databází. Data ukládá ve strukturách dokumentů, jejichž formát je podobný formátu JSON. [30]

Každá hodnota je uložena do příslušné kolekce v dokumentu, který obsahuje unikátní identifikátor, čas vložení a hodnotu samotnou.



Obrázek 4: Okno Robo 3T s ukázkou struktury dokumentu

3.8 Komunikace napříč mikroslužbami

Komunikaci zajišťuje REST API. Jedná se o rozhraní, které primárně slouží k přístupu k datům aplikace. Dle popisu autora je API rozloženo do čtyř úrovní. Nultá úroveň zajišťuje přenos dat. Zpravidla je využit HTTP, ale není to podmínkou. První úroveň definuje zdroje. Tedy cesty ke koncovým bodům. Například zdroj temperature/getLatest je navázán na koncový bod, metodu programu, která na výstup vrátí poslední záznam teploty z databáze. Ve druhé úrovni se skrývá informace o použité HTTP metodě. Nejběžněji využívanými jsou GET pro získání dat a POST pro jejich vytváření. HTTP však obsahuje i PUT pro úpravy, DELETE pro mazání a PATCH pro částečné úpravy. Ve třetí úrovni je popsán model, kdy je cílem, aby uživatel znal pouze

základní koncový bod aplikace, který spolu s daty vrací odkazy na další zdroje. Tím vzniká stromová struktura odkazů na zdroje. Výhodou je, že klient není závislý na URL adresách. Autor rozhraní sice propaguje myšlenku, že REST API musí být řízeno odkazy, v praxi se ale uplatňuje málokdy.

Při komunikaci se využívá HTTP stavových kódu. Tím klient získává zpětnou vazbu ze serveru. Obecně jsou kódy rozděleny do pěti kategorií. Pokud začíná číslem 1, jedná se o informační odpověď. Počáteční číslo 2 značí úspěch, 3 přesměrování. Pod číslem 4 se skrývají chyby na straně klienta a pod číslem 5 chyby na straně serveru. Nejběžnější je 200 OK. Tím server oznamuje, že požadavek proběhl v pořádku. Při vytváření obsahu přes POST metodu je při úspěšném dokončení navrácen kód 201 created. Z chybových stavů lze vyčíst, že byl zaslán špatný požadavek. Informovat nás o tom bude kód 400 Bad Request. Obdobně pokud klient není ověřen, je navrácen kód 401 Unauthorized nebo při neexistujícím zdroji 404 Not Found.

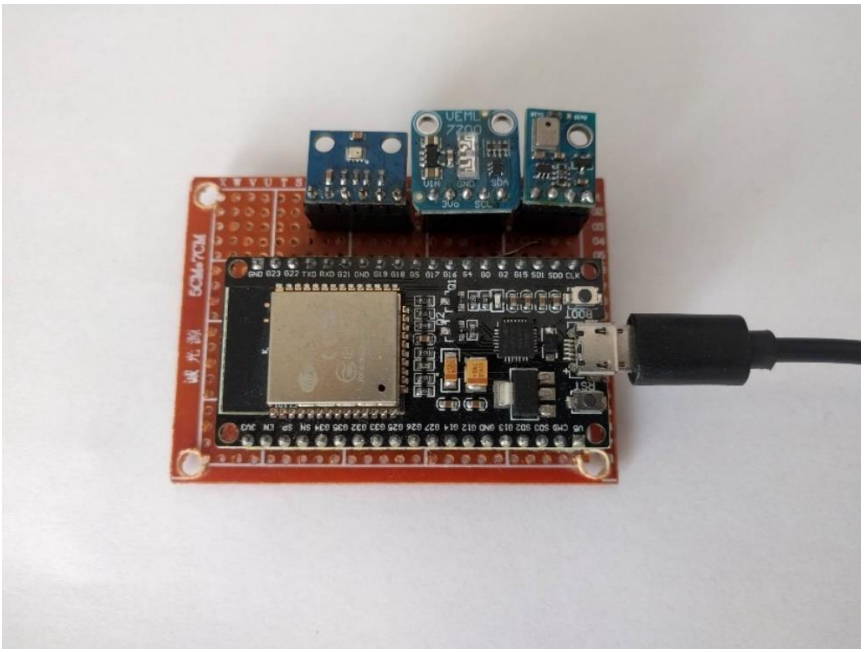
Volání REST zdrojů je bezstavové. Ověření uživatele by nemělo záviset na session nebo cookies. Spolu s každým požadavkem by měly být odeslány ověřovací údaje. Nejběžněji se k autentizaci využívá tokenů. [31]

4 REALIZACE

Celá aplikace má za cíl robustnost, škálovatelnost a spolehlivost. Toho bylo dosaženo koncipováním aplikace jako dílčích mikroslužeb. Úlohou každé mikroslužby je poskytovat omezené množství funkcí, které spolu logicky nebo implementačně souvisí. Jedná se o samostatné aplikace, které po spuštění na serveru vystupují jako samostatný proces. Navzájem mezi sebou komunikují přes standardní protokoly. Při realizaci bylo k tomuto účelu využito REST API, zasílání zpráv pomocí protokolu MQTT, popsaným výše, kde jako broker slouží Mosquitto, případně mezi sebou aplikace sdílejí stejnou databázi. Architektura mikroslužeb se perfektně snoubí s využitím Dockeru.

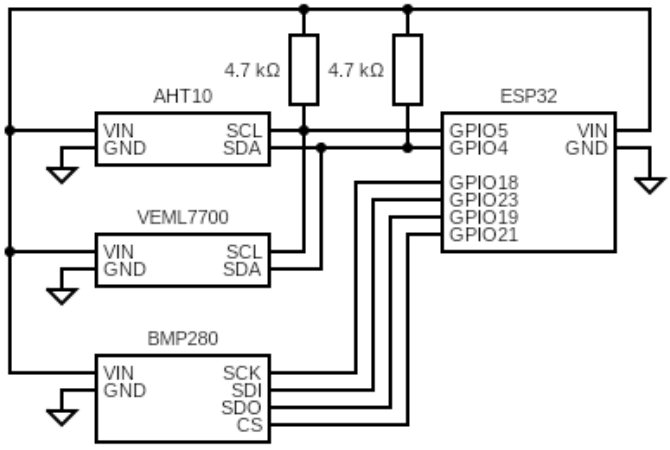
4.1 Hardwarové řešení

Zařízení bylo z důvodu počtu komponent a nízkým množství propojovacích tras mezi nimi sestaveno na univerzální PCB. Zařízení obsahuje modul ESP-WROOM-32, senzor AHT10, senzor BMP280 a senzor VEML7700.



Obrázek 5: Realizace měřícího zařízení na PCB

Ze schématu níže je patrné, že komunikace mezi senzory AHT10 a VEML7700 je zajištěna pomocí sběrnice I^2C. BMP280 pro svou komunikaci využívá sběrnici SPI.



Obrázek 6: Schéma zapojení

4.2 Obslužný program zařízení

Program v zařízení má za cíl připojit se k WiFi síti, navázat spojení s MQTT brokerem, načíst hodnoty ze senzorů a ty vypublikovat. Poté zařízení čeká stanovenou dobu a kroky opakuje. Kritické je správně zvolit prodlevu mezi měřeními. Musí být zajištěno, že uživatel má k dispozici dostatečné množství historicky naměřených dat a že nejnovější naměřená hodnota není tak stará, aby ztratila na relevanci. Pokud by doba mezi měření byla příliš dlouhá, aktuální hodnota by přestala být relevantní a pro zpětnou analýzu dat by neexistovalo dostatečné množství hodnot. Naopak při příliš častém měření dochází k vyšší spotřebě zařízení, vytěžování sítě a rychlému plnění databáze. Jako optimální se ukázalo odesílat data jednou za minutu.

4.3 Gateway

Brána zajišťuje klíčovou funkci v aplikaci. Z toho důvodu jsou její zodpovědnosti omezeny na nejnutnější minimum. Tím je zajištěna stabilita a vysoká dostupnost. Jedinou její funkcí je připojit se k brokeru jako subscriber na všech topicích, kde jsou očekávaná data ze zařízení. Brána asynchronně zpracovává data na těchto jednotlivých topicích, provede jejich validaci a spolu s aktuálním časem je uloží do databáze do příslušné kolekce podle měřené veličiny.

```
List<String> topics =
  Arrays.asList("temperature", "humidity", "pressure", "lightLevel");
for(String topic : topics) {
  Persistence persistence =
    new Persistence(DB_HOST, prepareCredentials(), DB_NAME, topic);
  new Client(BROKER_HOST, TOPIC_PREFIX + "/" + topic,
    persistence::save);
}
```

Zdrojový kód 4: Provázání dat z odposlouchávaných topiců s kolekcí v databázi

Instance třídy Persistence vytvoří spojení s databázovým serverem na specifikované adrese v prvním parametru. Jelikož je server zabezpečený, v dalším parametru obdrží přístupové údaje. Třetí parametr udává, ke které konkrétní databázi se připojit a posledním parametrem je název kolekce, která koresponduje s názvem obsluhovaného topicu.

Client se stará o příjem dat z brokeru. Z toho důvodu vyžaduje adresu MQTT serveru. Dále na kterém topicu datům naslouchat a takzvanou callback funkci. Callbackem může být libovolný Java consumer. V tomto konkrétním případě při přijetí dat instance třídy Client vyvolá metodu save z předem vytvořené instance třídy Persistence. Jejím úkolem je ověřit, že subscriber neobdržel prázdná data, připravit dokument a ten vložit do databáze.

```
ByteBuffer payload = value.getPayload()
  .orElseThrow(() -> new PersistenceException(Error.MISSING_PAYLOAD));
String decodedPayload = StandardCharsets.UTF_8.decode(payload)
  .toString();
Document document = new Document();
try {
  document.append("value", decodedPayload);
  document.append("time", new Date());
  collection.insertOne(document);
} catch (RuntimeException e) {
  logger.error(Error.DOCUMENT_NOT_CREATED, e);
  throw new PersistenceException(Error.DOCUMENT_NOT_CREATED, e);
}
logger.info("Document was stored into database with no problems.
  Document: {}.", document);
```

Zdrojový kód 5: Ukládání odposlechnutých dat

4.4 Správa dat

Aplikace obsahuje webový server pro zpřístupnění komunikace skrz REST volání. Nahlíží a pracuje se stejnou databází jako gateway, tudíž má přístup ke všem datům, které poskytlo měřící zařízení. Hlavní funkcionalitou je práci s těmito daty. Především poskytování poslední

uložené hodnoty, listování určitého množství posledních záznamů a výpočet trendu. Také jsou k dispozici servisní operace pro korekci, mazání a přidávání dat.

Je využíváno návrhového vzoru MVC. Základem je controller, který zpřístupňuje jednotlivé vybrané metody z modelu skrze REST. Model v sobě udržuje veškerou logiku programu. Využitím tohoto návrhového vzoru dochází k výraznému zpřehlednění kódu. Poslední nevysvětlené písmeno z výše zmíněné zkratky zastupuje view. Tedy zobrazování dat. V tomto případě je celý frontend oddělený do samostatné sub aplikace. Díky využití Spring Framework stačí třídu controlleru anotovat jako @RestController a v této třídě vytvořit metody, které volají metody z modelu, s anotací @PostMapping případně @GetMapping. Tím je specifikován typ metody HTTP požadavku. Do této anotace je zapotřebí doplnit, skrz který koncový bod bude metoda zpřístupněna. O konverzi přenášených dat na Java objekt se postará anotace @RequestBody.

```
@RestController
public class TemperatureController {

  @PostMapping("/temperature/create")
  public TemperatureCreateOutput temperatureCreate(
    @RequestBody TemperatureCreateInput dtoIn) {
    return temperatureModel.create(dtoIn);
  }

  @PostMapping("/temperature/update")
  public TemperatureUpdateOutput temperatureUpdate(
    @RequestBody TemperatureUpdateInput dtoIn) {
    return temperatureModel.update(dtoIn);
  }

  @GetMapping("/temperature/get")
  public TemperatureGetOutput temperatureGet(
    @RequestBody TemperatureGetInput dtoIn) {
    return temperatureModel.get(dtoIn);
  }

  @PostMapping("/temperature/list")
  public TemperatureListOutput temperatureTrend(
    @RequestBody TemperatureListInput dtoIn) {
    return temperatureModel.list(dtoIn);
  }

  @PostMapping("/temperature/delete")
  public TemperatureDeleteOutput temperatureDelete(
    @RequestBody TemperatureDeleteInput dtoIn) {
    return temperatureModel.delete(dtoIn);
  }

  @PostMapping("/temperature/getTrend")
  public double temperatureTrend(
    @RequestBody TemperatureTrendInput dtoIn) {
    return temperatureModel.getTrend(dtoIn);
  }

  @GetMapping("/temperature/getLatest")
  public Temperature temperatureGetLatest() {
    return temperatureModel.getLatest();
  }
}
```

Zdrojový kód 6: Ukázka anotací využívaných při vytváření REST controlleru pomocí Spring Framework

S konverzí dat umí Spring Framework pracovat nejen v případě REST dotazů, ale zvládne zpříjemnit i práci s databází. Objekt, který chceme načítat nebo naopak vkládat, můžeme nadepsat anotací @Document spolu s názvem kolekce z databáze. Mapování id zajistí anotace @Id. V případě, že by se lišil název atributu ve třídě oproti názvu klíče v databázi, pomůže nám

anotace @Field, do které doplníme název klíče z databáze a umístíme jí nad požadovaný atribut.

```
@Document("temperature")
public class Temperature {

@Id
private String id;
private double value;
private Instant time;
```

Zdrojový kód 7: Anotace objektu využívaného k práci s databází

Kromě standardních CRUD operací, tedy vytváření, čtení, aktualizace a mazání, poskytuje každý model metodu k výpočtu trendu. Ten je realizován lineární regresí. Vstupem metody je objekt obsahující počáteční a koncový čas, tedy období, za které chceme trend vypočítat. Tyto hodnoty využijeme k načtení dat z databáze, které leží ve sledovaném intervalu. Připravíme dva listy hodnot. Jeden pro naměřená data a druhý pro časy. Ty využijeme k sestavení matice, ze které bude výsledná regrese vypočtena. K tomu poslouží knihovna SimpleRegression, která skrze metodu getSlope zpřístupňuje sklon přímky vypočtené regrese.

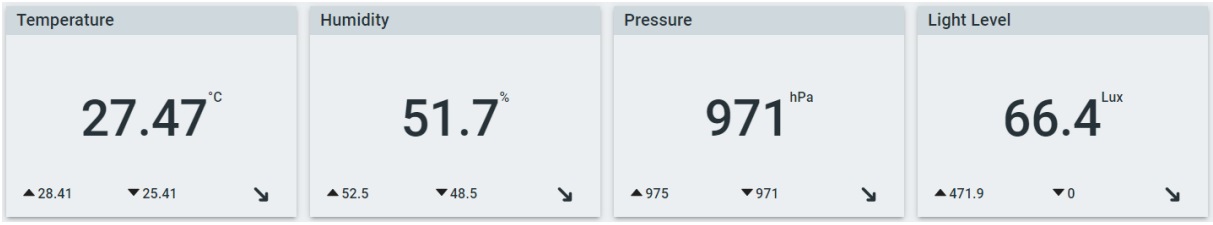
```
List<Temperature> temperatureList = getTemperatures(dtoIn);
if (temperatureList.isEmpty()) {
  return 0;
}
List<Double> temperatures = temperatureList.stream()
  .map(Temperature::getValue)
  .collect(Collectors.toList());
List<Double> times = temperatureList.stream()
  .map(Temperature::getTime)
  .map(Instant::getEpochSecond)
  .map(Long::doubleValue)
  .collect(Collectors.toList());
SimpleRegression regression = new SimpleRegression();
for (int i = 0; i < temperatures.size(); i++) {
  regression.addData(temperatures.get(i), times.get(i));
}
return regression.getSlope();
```

Zdrojový kód 8: Výpočet trendu

4.5 Zobrazování dat

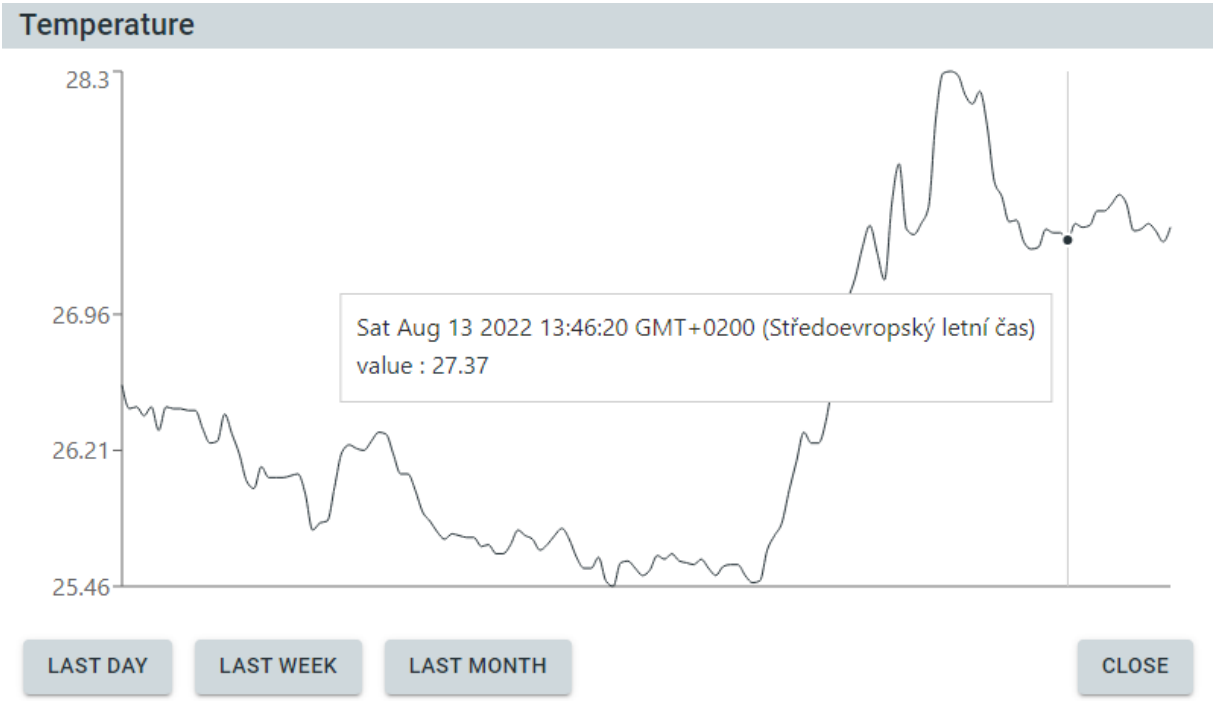
Aplikace skrz REST dotazy získává informace o naměřených datech, které ve srozumitelné a přehledné podobě reprezentuje uživateli. Celá webová aplikace je koncipovaná formou jediné stránky. Nachází se zde 4 dlaždice, pro každou sledovanou hodnotu jedna. Uprostřed každé

dlaždice je zobrazena nejaktuálnější hodnota a jednotka sledované veličiny. V dolní části se zobrazuje nejvyšší denní maximum, nejnižší denní minimum a trend vypočítaný za poslední dvě hodiny. V případě neaktuálních dat, které jsou starší než 6 hodin, nás dlaždice upozorní formou vykřičníku v pravém horním rohu. Aby dlaždice zobrazovala vždy nejaktuálnější možnou hodnotu, dochází k periodickému provolávání dotazu a v případě novější hodnoty je dlaždice překreslena.



Obrázek 7: Dlaždice s aktuálními hodnotami

Po kliknutí na dlaždici dojde ke zobrazení modálního okna s vykresleným grafem za posledních 24 hodin. K dispozici jsou tlačítka pro zobrazení mimo posledního dne i poslední týden a poslední měsíc. Při najetí myší na graf dostaneme informace o bodě, nad kterým se myš nachází. Dojde ke zvýraznění a vypsání času spolu s naměřenou hodnotou. Aby bodů v grafu nebylo příliš a nesnižovala se jeho čitelnost, data jsou vždy zprůměrována za určitý časový úsek a reprezentována jako jeden bod. Čím delší je sledovaný časový úsek v grafu, tím více hodnot je spolu zprůměrováno.



Obrázek 8: Detail grafu hodnot

K vykreslení dlaždice slouží komponenta ValueDisplayer. Přijímá 6 parametrů, v Reactu zvaných propsy. Prvním z nich je zobrazovaný popisek, druhým měrná jednotka veličiny. Následuje funkce zodpovědná za formátování. Tedy zaokrouhlování na určitý počet míst. Dalšími v pořadí jsou koncové body REST volání do aplikace správy dat. Vždy je poskytnut koncový bod pro aktuální hodnotu, trend a list.

```
<ValueDisplayer
  label={"Temperature"}
  unit={"°C"}
  correction={temperatureFormat}
  valuePayload={"/temperature/getLatest"}
  trendPayload={"/temperature/getTrend"}
  listPayload={"/temperature/list"}
/>
```

Zdrojový kód 9: Komponenta ValueDisplayer

List se skrz komponentu přeposílá hlouběji do komponenty modálního okna grafu, kde slouží k načítání vykreslovaných dat. V komponentě ValueDisplayer je využit pro hledání extrému za sledované období. Po tom, co jsou data načtena, dojde k jejich konverzi na JSON objekt a dále k jejich korekci, tedy zaokrouhlení na počet desetinných míst, pomocí funkce předané do komponenty. Využitím funkce min a max z Nodejs knihovny Math dojde k vybrání nejnižší a nejvyšší hodnoty z pole. Nakonec se nastaví stav a tím dojde k překreslení části komponenty.

```
let listInput = {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    "from": moment().subtract(1, 'days').endOf('day'),
    "to": moment()
    })
  };

fetch(listPayload, listInput)
  .then(response => response.json())
  .then(
    (data) => {
      let max = "N/A";
      let min = "N/A";
      if (data.dataList && data.dataList.length !== 0) {
        let values = data.dataList
          .map(val => val.value)
          .map(val => correction(val));
        max = Math.max(...values);
        min = Math.min(...values);
      }

    setMinValue(min);
    setMaxValue(max);
  }
);
```

Zdrojový kód 10: Získání extrémů z naměřených hodnot

K vykreslování a zobrazení grafu slouží komponenta GraphModal, která využívá knihovny Recharts. Při měření hodnot každou minutu a maximálním intervalem pro vykreslení jeden měsíc by v extrémním případě mohlo dojít k požadavku vykreslit 30240 bodů do grafu. Jelikož jsou tyto požadavky zpracovávány na straně klienta, docházelo by k jeho nadměrnému zatěžování a v případě slabších zařízení možná i k nestabilitě prohlížeče. Také by se mohla snížit celková čitelnost grafu. Proto je před samotným vykreslením provedeno sloučení hodnot, aby se výsledný počet bodů pohyboval okolo 150. Z celkového počtu dat je zjištěno, kolik bodů se vždy musí sloučit do jednoho. Ty jsou následně načteny do speciálního pole, sečteny a zprůměrovány. Pro zprůměrovanou hodnotu je vždy použit čas poslední hodnoty ze skupiny.

```
let modifiedData = [];
let countPerPoint = Math.ceil(data.length / numberOfPoints);
for (let i = 0; i < numberOfPoints; i++) {
  let tempArray = data.slice(
    countPerPoint * i, countPerPoint * (i + 1));
  if (!tempArray.length) {
    continue;
  }
  let avg = tempArray
    .map(val => val.value)
    .reduce((a, b) => a + b, 0) / tempArray.length;
  modifiedData[i] = {
    value: correction(avg),
    time: tempArray.pop().time
  };
}
return modifiedData;
```

Zdrojový kód 11: Slučování bodů

4.6 Server

Z důvodu, že řešení má být dostupné napříč internetem z libovolného místa, je zapotřebí aplikaci zprovoznit na serveru s veřejnou IP adresou. Jako nejrozumnější řešení se ukázalo využít služeb některého z poskytovatelů virtuálních serverů. Díky tomu, že virtuální server využívá pouze část výkonu skutečného hardwarového serveru a na jednom takovém stroji může virtuálních serverů být provozováno paralelně desítky ve stejný moment, je cena velice příznivá.

Server musí být schopen obsloužit běh dvou Java aplikací a jedné React/Node.js aplikace. Také je potřeba dostatečná velikost diskového úložiště pro naměřená data. Vybraná konfigurace virtuálního serveru poskytuje 2 GB RAM a 40 GB diskového prostoru. Ve smluvích podmínkách je zmínka o omezení přenosu dat na 5 TB měsíčně. Na tuto část nebylo třeba brát zřetel, jelikož aplikace není na objem přenášených dat nikterak náročná. Na výběr mezi operačními systémy zpravidla bývá několik Linuxových distribucí. Pro tento server byl vybrán Debian 11. Velikost operační paměti se na první pohled zdá být předimenzovaná. Pro běh Nodejs aplikace je ale před jejím prvním spuštěním zapotřebí doinstalovat závislosti externích knihovech, kterých je pro React velké množsví. S nižší kapacitou RAM se ukázalo jako velmi obtížné, skoro až nereálné, tyto závislosti doinstalovat.

Služba byla objednána od společnosti https://forpsi.com/ se smluvenou měsíční cenou 150 Kč bez DPH.

Prvním a hlavním krokem pro nazasení aplikace byla instalace Dockeru do systému. Ten se nenachází v oficiálním repozitáři a ještě před samotnou instalací je zapotřebí jeho vlastní repozitář přidat. Tím se k instalaci přes apt-get zpřístupní potřebné balíčky docker-ce, docker-ce-cli, containerd.io a docker-compose-plugin. [32]

Dalším krokem je instalace MongoDB a Eclipse Mosquitto. Obě dvě služby jsou dostupné na DockerHub. Tím je instalace ulehčena a stačí tyto dva image zakomponovat do souboru docker-compose.yaml

```
services:
  mosquitto:
    image: eclipse-mosquitto
    network_mode: host
    volumes:
            - ./conf:/mosquitto/config
            - ./data:/mosquitto/data
            - ./log:/mosquitto/log
    restart: always
  mongodb:
    image: mongo:5.0
    ports:
      - 27017:27017
    volumes:
      - ~/apps/mongo:/data/db
    command: [--auth]
    restart: always
```

Zdrojový kód 12: Konfigurační soubor docker-compose

Mosquitto ke svému běhu vyžaduje konfigurační soubor s názvem mosquitto.conf uložený v adresáři specifikovaném v docker-compose. V tomto souboru je zapotřebí poskytnout port, na kterém server naslouchá. Jelikož zařízení nevyužívá zabezpečenou komunikaci skrz MQTT protokol, v konfiguračním souboru musí být poskytnuta informace o tom, že může obsluhovat i anonymně připojené zařízení.

```
# Listen on a port/ip address combination. By using this variable
# multiple times, mosquitto can listen on more than one port. If
# this variable is used and neither bind_address nor port given,
# then the default listener will not be started.
# The port number to listen on must be given. Optionally, an ip
# address or host name may be supplied as a second argument. In
# this case, mosquitto will attempt to bind the listener to that
# address and so restrict access to the associated network and
# interface. By default, mosquitto will listen on all interfaces.
# listener port-number [ip address/host name/unix socket path]
listener 1883

# Boolean value that determines whether clients that connect
# without providing a username are allowed to connect. If set to
# false then a password file should be created (see the
# password_file option) to control authenticated client access.
# Defaults to false, unless there are no listeners defined in the
# configuration file, in which case it is set to true, but
# connections are only allowed from the local machine.
allow_anonymous true
```

Zdrojový kód 13: Příklad konfiguračního souboru pro Mosquitto

Z důvodu bezpečnosti jsou v operačním systému Debian defaultně zablokovány porty. Je tedy nutné komunikaci na všech využívaných portech povolit. V tomto konkrétním případě se jedná o port 27017 pro MongoDB, 1883 pro Mosquitto a 8080 pro backendovou část aplikace pro zpřístupnění REST volání. React pro svůj běh využívá port 3000. Pro větší uživatelskou přívětivost je vhodnější tento port přesměrovat na 80, aby adresa, přes kterou se přistupuje ve webovém prohlížeči nemusela být doplněna i o zmíněný port.

Poslední náležitostí před samotným spuštěním aplikace je nainstalovat běhové prostředí pro Javu a Nodejs. Ani jeden ze zmíněných potřebných balíčků není dostupný v oficiálním repozitáři a je proto nutné tyto repozitáře dle preferencí dohledat a do systému přidat.

Samotné aplikace je možné provozovat spuštěné napřímo v systému. Poté je ale vhodné vyřešit automatický start po spuštění systému, případně restart při pádu. Jednodušší a příhodnější možností je tuto práci přenechat Dockeru. Ten nám mimo zmíněných situací usnadní i návaznost spouštění jednotlivých aplikací. V souboru docker-compose.yaml můžeme specifikovat, že Gateway závisí na MongoDB i Mosquitto, zatímco aplikace pro správu dat si vystačí pouze s MongoDB stejně tak jako mikroslužba zodpovědná za zobrazování dat.

4.7 Cenový rozpočet

Při konstrukci zařízení byla kromě kvality komponent důležitá i jejich cena. Moduly byly zakoupeny prostřednictvím internetového obchodu https://aliexpress.com/.

Komponenta	**Cena**
ESP-WROOM-32	102 Kč
AHT10	26 Kč
BMP280	14 Kč
VEML7700	130 Kč
Součet	272 Kč

Tabulka 1: Soupis cen komponent

5 TESTOVÁNÍ WEBU

Několik uživatelů s různými úrovněmi technických znalostí a s různými znalostmi informačních technologií byli požádání o přirozené použití webové aplikace. Hlavním hodnotícím kritériem byla funkčnost webu jako celku. Ve zpětné vazbě dostali prostor pro vyjádření se i k uživatelské přívětivosti. Uživatelé měli možnost zhodnotit rozložení komponent na obrazovce, snadnost pochopení významu ikon a míru složitosti orientace v grafech.

Rozložení webové aplikace bylo navrhováno s myšlenkou maximální jednoduchosti a přehlednosti. Tento cíl byl dle ohlasů oslovených uživatelů splněn. Všichni respondenti reagovali na webovou aplikaci kladně.

6 POROVNÁNÍ S KOMERČNĚ DOSTUPNÝMI ŘEŠENÍMI

Cílem práce bylo poskytnout vzdálený monitoring venkovního prostředí, proto je vzdálený přístup hlavní porovnávané kritérium. Všechna porovnávaná komerční řešení používají vlastní mobilní aplikaci a neposkytují informace o protokolu, přes který jsou data přenášena. Za nevýhodu komerčních řešení považuji nutnost použití serverů výrobce meteostanice. Pokud by výrobci přistoupili k použití otevřených protokolů jako je například MQTT, bylo by možné integrovat zařízení bez větší námahy do rozšířených platforem chytrých domácností jako je Openhab nebo Home assistant.

Většina komerčních meteostanic obsahuje kromě venkovní jednotky i vnitřní jednotku s displejem. Není tedy zapotřebí ke každé kontrole aktuálních hodnot otevírat mobilní nebo webovou aplikaci. To považuji za vhodné a uživatelsky přívětivé řešení.

Zařízení bylo umístěno poblíž komerčně dostupné meteostanice značky TFA. Každou půl hodinu po dobu 12 hodin byly odečteny hodnoty z obou zařízení. Výrobce bohužel neposkytuje informace o odchylce měření senzorů použitých v jeho řešení. Rozdíl teplot byl v průběhu testování vždy nižší než 1 °C a rozdíl vlhkostí nižší než 2 %. Výsledek testu je tedy kladný a výstupy ze zařízení je možné považovat za přesné.

ZÁVĚR

Zadáním této bakalářské práce bylo navrhnout meteostanici, která autonomně a v pravidelných intervalech odesílá zaznamenané hodnoty na databázový server. K vyřešení tohoto problému bylo zapotřebí nastudovat problematiku meteostanic. S tím souvisí pochopení možností a principů senzorů určených k měření meteorologických veličin.

V druhé části práce následoval průzkum trhu komponent a vhodného mikrokontroleru, který bude splňovat podmínku přenosu dat na vzdálený server. Na základě toho byly vybrány hardwarové prvky. Z důvodu studijního oboru, který se soustředí převážně na informační technologie, a ne primárně na elektrotechniku, byly senzory i vývojová deska vybrány jako hotové moduly. Ze stejného důvodu bylo zařízení sestaveno pouze na prototypové desce, nikoli na míru vytvořené desce plošného spoje.

Třetím krokem bylo vybrat technologie vhodné k ukládání naměřených dat, zobrazování hodnot na webové stránce a výběr komunikačních protokolů. Přihlíželo se k popularitě jednotlivých programovacích jazyků, jejich možnostem využití protokolů a přívětivosti vzájemné integrace jednotlivých služeb napříč různými jazyky. Minoritní roli při výběru hrála roli i informace, jestli je dané řešení dostupné pod licencí open source.

Po dokončení realizace bylo řešení spuštěno na vzdáleném serveru a ověřena funkčnost. Zařízení i server byl několik týdnů ponechán bez vnějších zásahů a tím byla ověřena stabilita celého systému. Porovnáním s komerčně dostupnou meteostanicí a sledováním hodnot, které poskytovala byla ověřena podobnost dat a tím potvrzená správnost implementace zařízení.

POUŽITÁ LITERATURA

[1] KRÁLOVÁ, Magda. Meteorologie - Eduportál Techmania [online]. [cit. 04.08.2022]. Dostupné z: https://edu.techmania.cz/cs/encyklopedie/fyzika/meteorologie

[2] KRÁLOVÁ, Magda. Meteorologická stanice - Eduportál Techmania [online]. [cit. 07.08.2022]. Dostupné z: https://edu.techmania.cz/cs/encyklopedie/fyzika/meteorologie/meteorologicka-stanice

[3] KRÁLOVÁ, Magda. Teplota vzduchu - Eduportál Techmania. Eduportál [online]. [cit. 04.08.2022]. Dostupné z: http://edu.techmania.cz/cs/encyklopedie/fyzika/meteorologie/teplota-vzduchu

[4] KRÁLOVÁ, Magda. Vlhkost vzduchu - Eduportál Techmania [online]. [cit. 04.08.2022]. Dostupné z: http://edu.techmania.cz/cs/encyklopedie/fyzika/meteorologie/vlhkost-vzduchu

[5] KRÁLOVÁ, Magda. Tlak vzduchu - Eduportál Techmania [online]. [cit. 05.08.2022]. Dostupné z: http://edu.techmania.cz/cs/encyklopedie/fyzika/meteorologie/tlak-vzduchu

[6] CIRI, Patrick. How to Measure Light Intensity: Understanding & Using a Lux Meter [online]. [cit. 06.08.2022]. Dostupné z: https://bioslighting.com/how-to-measure-light-intensity/architectural-lighting/

[7] Široký výběr modulů ESP32 [online]. [cit. 06.08.2022]. Dostupné z: https://www.tme.eu/cz/news/library-articles/page/21733/siroky-vyber-modulu-esp32/

[8] The Internet of Things with ESP32 [online]. [cit. 06.08.2022]. Dostupné z: http://esp32.net/

[9] AHT10 Technical Manul [online]. [cit. 06.08.2022]. Dostupné z: https://server4.eca.ir/eshop/AHT10/Aosong_AHT10_en_draft_0c.pdf

[10] LIU, Thomas. Digital-output relative humidity & temperature senror/module DHT22 (DHT22 also named as AM2302) [online]. [cit. 07.08.2022]. Dostupné z: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[11] Data sheet BMP 280 Digital Pressure Sensor [online]. [cit. 06.08.2022]. Dostupné z: https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf

[12] VEML7700. High Accuracy Ambient Light Sensor With I^2C Interface [online]. [cit. 06.08.2022]. Dostupné z: https://cz.mouser.com/datasheet/2/427/veml7700-1767629.pdf

[13] Ambient Light Sensor Module Light Measuring Board I2C Bus Interface for Raspberry Pi VEML7700. [online]. [cit. 14.08.2022]. Dostupné z: https://www.joom.com/cs/products/623d3b78ca075701299c4c96

[14] MAX440009. Industry's Lowest-Power Ambient Light Sensor with ADC [online]. [cit. 06.08.2022].
Dostupné z: https://datasheets.maximintegrated.com/en/ds/MAX44009.pdf

[15] I2C Info – I2C Bus, Interface and Protocol [online]. [cit. 06.08.2022]. Dostupné z: https://i2c.info/

[16] I2C (TWI) – sériová komunikace po dvou vodičích [online]. [cit. 14.08.2022]. Dostupné z: http://www.zavavov.cz/cz/elektrotechnika/komunikacni-sbernice/68-i2c-twi-seriova-komunikace-po-dvou-vodicich-s-adresaci/

[17] CAMPBELL, Scott. Basics of the SPI Communication Protocol [online]. [cit. 06.08.2022] Dostupné z: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/

[18] Learn about WiFi standards and the latest WiFi 6 (802.11 ax) [online]. [cit. 06.08.2022]. Dostupné z: https://www.netspotapp.com/blog/wifi-security/explaining-wifi-standards.html

[19] AQEEL, Adnan. Introduction to Arduino IDE [online]. [cit. 06.08.2022]. Dostupné z: https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html

[20] IntelliJ IDEA [online]. [cit. 06.08.2022]. Dostupné z: https://www.jetbrains.com/help/idea/discover-intellij-idea.html

[21] Robo 3T | Free, open-source MongoDB GUI (formerly Robomongo) [online]. [cit. 06.08.2022]. Dostupné z: https://robomongo.org/

[22] The API Design Platform and API Client - Insomnia. [online]. [cit. 06.08.2022]. Dostupné z: https://insomnia.rest/

[23] Docker [online]. [cit. 10.08.2022]. Dostupné z: https://www.docker.com/

[24] HERTMAN, James. What is Java? Definition, Meaning & Features of Java Platforms [online]. [cit. 07.08.2022] Dostupné z: https://www.guru99.com/java-platform.html

[25] Spring [online]. [cit. 06.08.2022]. Dostupné z: https://spring.io/

[26] React – A JavaScript library for building user interfaces [online]. [cit. 06.08.2022]. Dostupné z: https://reactjs.org/

[27] MQTT – The Standard for IoT Messaging [online]. [cit. 06.08.2022]. Dostupné z: https://mqtt.org/

[28] Tutorial: Using MQTT on SwitchDoc Labs OurWeather [online]. [cit. 14.08.2022]. Dostupné z: https://www.switchdoc.com/2020/01/tutorial-using-mqtt-on-switchdoc-labs-ourweather-station/

[29] MŮČKA, Jan. Relační databáze vs. nerelační databáze: Čím se liší? [online]. [cit. 13.08.2022]. Dostupné z: https://www.master.cz/blog/relacni-databaze-nerelacni-databaze-jake-jsou-rozdily/

[30] MongoDB: The Developer Data [online]. [cit. 06.08.2022]. Dostupné z: https://www.mongodb.com/

[31] GUPTA, Lokesh. What is REST – REST API Tutorial [online]. [cit. 10.08.2022]. Dostupné z: https://restfulapi.net/

[32] Install Docker Engine on Debian [online]. [cit. 16.08.2022]. Dostupné z: https://docs.docker.com/engine/install/debian/