UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2021 Tomáš Pilný

UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

ZAŘÍZENÍ PRO SBĚR A ZPRACOVÁNÍ METEOROLOGICKÝCH DAT

Tomáš Pilný

Bakalářská práce

2021

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš Pilný**
Osobní číslo:	**I19036**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Řízení procesů**
Téma práce:	**Zařízení pro sběr a zpracování meteorologických dat**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Cílem práce je navrhnout a realizovat zařízení (meteostanici) pro sběr a zpracování meteorologických dat, která se bude skládat ze dvou jednotek – venkovní a vnitřní. Venkovní jednotka bude umožňovat sběr dat o teplotě a vlhkosti vzduchu, detekci deště a bezdrátový přenos dat do vnitřní jednotky. Vnitřní jednotka budě umožňovat sběr dat o teplotě, vlhkosti a tlaku vzduchu, zpracovávat bezdrátově přijatá data z venkovní jednotky, zobrazovat data a aktuální čas na LCD displeji a pomocí Wi-Fi odesílat údaje na web.

Teoretická část: Rešerše problematiky týkající se návrhu a realizace zařízení pro sběr a zpracovávání meteorologických dat.

Implementační část: Realizace venkovní jednotky na bázi libovolného mikrokontroléru (např. Arduino s využitím programovacího jazyka Wiring) a realizace vnitřní řídicí jednotky s využitím Raspberry Pi a programovacího jazyka Python. Vytvoření webu, který umožní pravidelnou aktualizaci dat z vnitřní jednotky, zobrazení těchto dat v grafech a jejich ukládání do databáze.

Rozsah pracovní zprávy: **cca 50 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

UPTON, E.; HALFACREE, G. 2016. Raspberry Pi User Guide. 4th ed. Chichester, West Sussex (UK): John Wiley & Sons. ISBN 978-1-119-26436-1.
DONAT, W. 2014. Learn Raspberry Pi Programming with Python: Learn to Program on the Worlds Most Popular Tiny Computer. 2nd ed. Berkeley: Apress. ISBN 978-1-4842-3768-7.
BRADBURY, A.; EVERARD, B. 2014. Learning Python with Raspberry Pi. Hoboken (USA): John Wiley & Sons. ISBN 978-1-118-71705-9.
MONK, S. 2019. Raspberry Pi Cookbook: Software and Hardware Problems and Solutions. 3rd ed. Sebastopol (USA): OReilly Media. ISBN 978-1-492-04322-5.

Vedoucí bakalářské práce: **Ing. Libor Kupka, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **27. listopadu 2020**
Termín odevzdání bakalářské práce: **14. května 2021**

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 29. ledna 2021

Prohlášení autora

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 15. 05. 2021

Tomáš Pilný

Poděkování

Rád bych tímto poděkoval vedoucímu bakalářské práce Ing. Liboru Kupkovi, Ph.D. za konzultace a rady týkající se návrhu a realizace práce. Dále bych rád poděkoval Anežce Blažkové za pomoc při korektuře textu.

V Pardubicích dne 15. 05. 2021

Tomáš Pilný

ANOTACE

Cílem této bakalářské práce je vytvořit hardware, který bude zaznamenávat meteorologická data vnitřního a venkovního prostředí a software, který bude ukládat a zpracovávat data. Teoretická část je věnována rozboru využitých mikrokontrolérů, mikroprocesorů, periferií a jednotlivých senzorů. Praktická část popisuje návrhy a realizace systémů, webovou aplikaci a databázi.

KLÍČOVÁ SLOVA

Meteorologická stanice, Raspberry Pi, Arduino, SQLite databáze, Python, lokální server

TITLE

DEVICE FOR OBTAINING AND PROCESSING OF METEOROLOGICAL DATA.

ANNOTATION

The objective of this bachelor's thesis is to create hardware for collecting and processing indoor and outdoor meteorological data and to develop software for further processing of obtained data. The theoretical part of my bachelor's thesis is dedicated to the analysis of used microcontrollers, microprocessors, peripherals, and individual sensors. The practical part is focused on the description of design and realization of implemented systems, web application, and the creation of the database.

KEYWORDS

Meteorological station, Raspberry Pi, Arduino, SQLite database, Python, local server

OBSAH

SEZNAM ZKRATEK

AJAX	Asynchronous JavaScript And XML	Asynchronní JavaScript a XML
APT	Advanced Package Tool	Pokročilý balíčkovací nástroj
ASCII	American Standard Code for Information Interchange	Kódovací tabulka znaků
ASK	Amplitude Shift Keying	Klíčování amplitudovým posuvem
CIDR	Classless Inter Domain Routing	Zkrácený zápis síťové masky
CSI	Camera Serial Interface	Kamerové sériové rozhraní
CSS	Cascading Style Sheets	Kaskádové styly
DHCP	Dynamic Host Configuration Protocol	Protokol nastavující IP adresy
DPS		Deska Plošných Spojů
DSI	Display Serial Interface	Displejové sériové rozhraní
EEPROM	Electrically Erasable Programmable Read Only Memory	Elektricky mazatelná a programovatelná paměť
GIL	Global Interpreter Lock	Zámek globálního tlumočníka
GPIO	General Purpose Input Output	Vstup/výstup pro obecné použití
HD	High Definition	Vysoké (obrazové) rozlišení
HTML	Hypertext Markup Language	Jazyk pro tvorbu webových stránek
I2C	Inter Integrated Circuit	Vzájemně integrovaný obvod
ID	Identification	Identifikace
IDE	Integrated Development Environment	Vývojové prostředí
IP	Internet Protocol	Internetový protokol
IPv4/6	Internet Protocol version 4/6	Internetový protokol verze 4/6
JSON	JavaScript Object Notation	Objektová JavaScript notace
LAN	Local Area Network	Lokální síť
LCD	Liquid Crystal Display	Displej z tekutých krystalů
LED	Light Emitting Diode	Světlo vyřazující dioda
Li-Po	Lithium-Polymer	Lithium-Polymerová baterie
LPDDR4	Low Power Double Data Rate 4	Nízkonapěťová operační paměť
MCU	Microcontroller Unit	Mikrokontrolér

MIPIA	Mobile Industry Processor Interface Alliance	Obchodní aliance vyvíjející rozhraní pro mobilní systémy
NTC	Negative Temperature Coefficient	Záporný teplotní koeficient
OOK	On Off Keying	Dvoustavová modulace (klíčování)
OS	Operating Systém	Operační systém
PNG	Portable Network Graphics	Přenosná síťová grafika
RF	Radio Frequency	Rádiové frekvence
SCL	Synchronous Clock	Synchronní hodinový signál
SD Card	Secure Digital Card	Paměťová karta
SDA	Synchronous Data	Synchronní datový signál
SoC	System on Chip	Systém na čipu
SPI	Serial Peripheral Interface	Sériové periferní rozhraní
SRAM	Static Random Access Memory	Statická paměť s náhodným přístupem
SSH	Secure Shell Protocol	Zabezpečený komunikační protokol
TCP	Transmission Control Protocol	Přenosový řídicí protokol
UART	Universal Asynchronous Receiver Transmitter	Univerzální asynchronní přijímač-vysílač
URL	Uniform Resource Locator	Webová adresa
USB	Universal Serial Bus	Univerzální sériová sběrnice
VNC	Virtual Network Computing	Připojení ke vzdálené ploše
WAL	Write Ahead Log	Zápis v předstihu
Wi-Fi	Wireless Fidelity	Standardy popisující bezdrátovou komunikaci

SEZNAM ZNAČEK (SYMBOLŮ, PROMĚNNÝCH A FUNKCÍ)

I	elektrický proud	A
P	atmosférický tlak	Pa
R	elektrický odpor	Ω
RH	relativní vlhkost	%
T	teplota	°C
t	čas	s
U	elektrické napětí	V

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

ÚVOD

Zařízení určená pro sběr a zpracování meteorologických dat se v dnešní době těší velké oblibě u široké veřejnosti. Většina lidí s přístupem k moderním technologiím vlastní jedno či více zařízení, která umožňují tato data zpracovat, ať už se jedná o údaje v telefonu stažené z příslušného webu nebo koupenou domácí meteorologickou stanici. Důvodem ke zvolení zde prezentovaného tématu je potřeba mít možnost data nejen zobrazovat, ale také je dlouhodobě zaznamenávat a mít možnost je dále zpracovat podle vlastního uvážení.

Cílem práce je tedy navrhnout takový systém, který se bude vlastnostmi blížit komerčně prodávaným meteostanicím, avšak s výhodou vlastního návrhu, a to jak po stránce komponent, tak implementovaného softwaru. Stanice se dělí na dvě části. Venkovní část umožňuje měřit teplotu a vlhkost vzduchu, informovat o výskytu deště a bezdrátově tyto údaje odesílat pomocí radiofrekvenčního vysílače do druhé části. Jednotlivé snímače jsou spolu s využitou vývojovou deskou Arduino Nano napájeny tužkovými bateriemi a uloženy v montážním boxu.

Vnitřní část umožňuje měřit teplotu a vlhkost vzduchu v budově a barometrický tlak. Je dále zodpovědná za příjem venkovních údajů o počasí. Řídicí jednotkou je jednodeskový počítač Raspberry Pi 4 B, který záznamy ukládá do SQLite3 databáze s využitím programovacího jazyka Python. Raspberry Pi pak funguje také jako lokální server, pro který je vytvořeno webové rozhraní, kde je možné sledovat aktuální hodnoty měřených veličin a vykreslovat je v jednotlivých grafech.

1 METEOROLOGIE

Meteorologie obecně je věda zabývající se studiem atmosféry a souvisejících úkazů a dopadů na počasí, což je termín označující okamžitý stav atmosféry na konkrétním místě. Atmosféra je plynný obal Země, ve kterém řečené úkazy probíhají, a je rozdělena na několik částí. Meteorologie se zabývá její nejnižší vrstvou zvanou troposféra (ta končí ve výšce 15 km od povrchu Země). Pro společnost je z mnoha důvodů žádoucí dokázat předpovídat vývoj počasí, aby toho ale mohlo být docíleno pomocí vědeckých metodik, je nutné dokázat změřit veličiny, které se do vývoje počasí promítají. Mezi hlavní ze zmíněných veličin patří teplota, vlhkost, oblačnost, množství srážek, rychlost a směr větru a atmosférický tlak. Aby měření byla co nejpřesnější, využívá meteorologie různých způsobů získávání hodnot. Mezi ně patří například využití meteostanic, meteorologických balónů, radarů a satelitů (Boudreau, 2012).

Tato práce se zaměřuje na tvorbu meteorologické stanice. Ty je obecně možno rozdělit na amatérské a profesionální, které se následně dělí na: synoptické, klimatologické, srážkoměrné a letecké. Následující rozbor se omezuje na způsoby měření těch veličin, které dokáže měřit meteostanice, jejíž návrh a konstrukce je obsahem této práce.

1.1 MĚŘENÍ TEPLOTY

Teplota je základní fyzikální veličina, charakterizující tepelný stav daného objektu. Základní jednotkou je Kelvin a vedlejší jednotkou používanou v evropských zemích je stupeň Celsia . Značí se pomocí velkého *T* (někdy se také značí pomocí malého *t*, ale toto označení je v této práci vymezeno pro čas). Teplota má silný vliv na vývoj počasí a jelikož se jedná o velmi proměnlivou veličinu, vzniká potřeba co možná nepřesnějšího měření. V praxi se využívá několik druhů teploměrů (termometrů), mezi nejčastěji používané patří: kapalinové, odporové a bimetalové (Cox a Bergstresser, 2015).

Kapalinové využívají tepelné roztažnosti použité kapaliny (nejčastěji rtuť). Bimetalové využívají kombinaci dvou různých kovů tvořících pásek, který se vlivem tepla ohýbá a natáčí ručku měřicího přístroje. Odporové teploměry neboli termistory se vyrábí ve dvou variantách PTC – odpor se vzrůstající teplotou narůstá a NTC – odpor se vzrůstající teplotou klesá. Existují také speciální termistory, které mají pevně definovanou hodnotu odporu pro konkrétní teplotu. Příkladem je termistor PT100 vyrobený z platiny s odporem 100 Ω při teplotě 0 °C. Při dosažení teploty 100 °C se hodnota změní skokově na přibližně 138 Ω. Pro všechny typy platí, že měření teploty by mělo probíhat přibližně 2 m nad zemí (nebo podlahou) ve stíněném prostoru (Laurila, 2018).

1.2 MĚŘENÍ VLHKOSTI A SRÁŽEK

Vlhkost je další zásadní veličina, která má velký vliv jak na vývoj počasí, tak na kvalitu života. Při nižší vlhkosti v místnosti může docházet ke svědění kůže, zarudnutí očí a celkovému pocitu únavy. To je způsobeno tím, že čím je vzduch sušší, tím větší má snahu přijímat vlhkost ze svého okolí.

Existují dva typy vlhkosti. Prvním je absolutní vlhkost, která určuje množství vodní páry ve vzduchu bez ohledu na teplotu. Je vyjádřena jako gram vodní páry na krychlový metr. Druhým typem je relativní vlhkost, která vyjadřuje aktuální vlhkost ovzduší. Je vyjádřena jako procentuální hodnota celkového množství vodní páry, které by mohl vzduch pojmout při dané teplotě. Platí přímá úměra, že čím vyšší teplota vzduchu, tím více vlhkosti může pojmout a naopak. Pokud relativní vlhkost dosáhne hodnoty 100 %, vzduch už není schopný pojmout další vlhkost a dochází ke kondenzaci (zkapalnění). Hodnota, při které tento jev nastane, je nazývána teplota rosného bodu. Ideální pokojová vlhkost se pohybuje v rozmezí 50 % až 60 % (Khillar, 2019).

K měření vlhkosti se využívají dva typy přístrojů: hygrometry a psychrometry. Elektronické hygrometry využívají substrát, který v závislosti na vlhkosti mění svůj odpor. Existují také organické verze tohoto přístroje, které měří vlhkost na základě změny délky odmaštěného vlasu. Psychrometry obsahují dva teploměry, z nichž jeden je zvlhčován pomocí kapaliny. Foukáním vzduchu na teploměry pomocí ventilátoru se zvlhčený teploměr ochlazuje více, přičemž výsledný rozdíl teplot je nepřímo úměrný relativní vlhkosti (The Editors of Encyclopaedia Britannica, 1998).

Srážky je pak možné měřit Ombrometrem, který zachytává dešťové kapky do kalibrovaného odměrného válce v přesně definovaných časových intervalech. Lze také využít takzvaný člunkový srážkoměr, který zachytává srážky do žlábku rozděleného na dvě stejně velké části. Když se jedna strana naplní vodou, dojde k překlopení žlábku a vylití vody, díky čemuž se může začít plnit vodou jeho druhá strana. Počet překlopení pak odpovídá celkovému množství srážek (Gires, 2018).

1.3 MĚŘENÍ TLAKU

Tlak je definován jako síla, kterou působí atmosféra na jednotku plochy v určitém místě na Zemi. Někdy se atmosférický tlak označuje také jako barometrický. Měření tlaku je zásadní jak pro předpověď počasí, kde se s jeho pomocí tvoří izobary v synoptických mapách, tak například v letecké dopravě. Izobara označuje křivku, která propojuje vícero míst se stejnou hodnotou atmosférického tlaku. Hodnota tlaku klesá se vzrůstající nadmořskou výškou, tuto změnu popisuje vertikální tlakový gradient. V nižších polohách klesá tlak zhruba o 13 hPa na 100 m převýšení. Tento jev je způsoben tím, že jak vzrůstá nadmořská výška, tak se snižuje atmosférický sloupec nad daným místem. Díky tomu mohou letadla přesně určit svoji aktuální letovou výšku, a to za pomoci referenční hodnoty 1013,25 hPa vztažené k úrovni moře. Tato hodnota se nazývá normální tlak (Smolka, 2015).

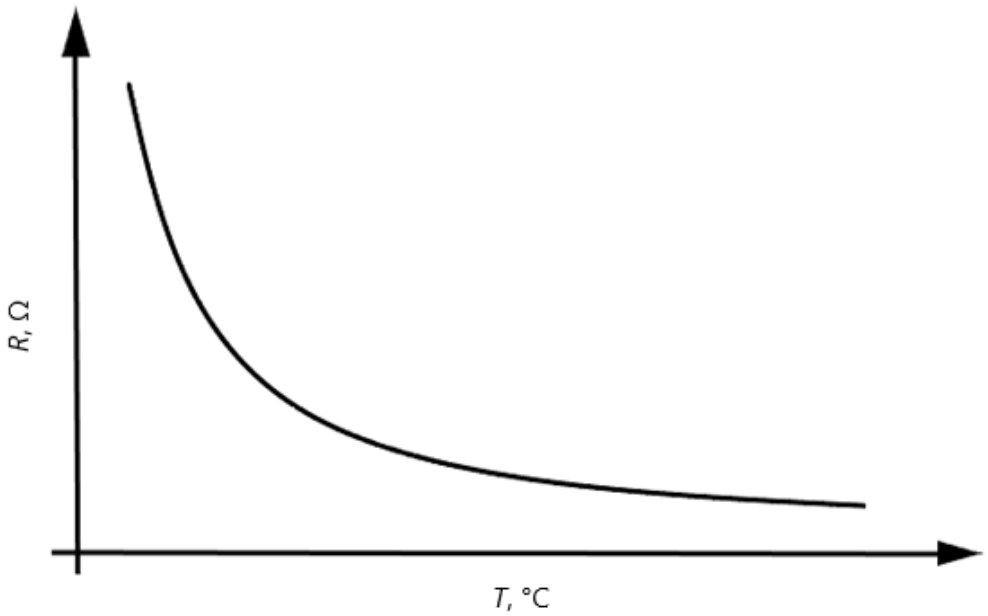
V případě předpovědi počasí lze prudký pokles tlaku brát jako indikaci nadcházející oblačnosti a silného deště. V okamžiku, kdy tlak začne pozvolna narůstat, se dá očekávat výhledově lepší počasí.

K měření tlaku se používají barometry, přičemž v dnešní době převládají jejich elektronické verze využívající piezorezistivního jevu. Tento princip je podrobněji rozveden v kapitole 2.2.

2 MIKROKONTROLÉRY A PERIFERIE

2.1 TEPLOTNÍ A VLHKOSTNÍ SENZOR DHT22

DHT22 je kalibrovaný digitální snímač s operačním rozsahem teploty od -40 °C do +80 °C s přesností ±0,5 °C a měřicím rozsahem vlhkosti od 0 % do 100 % s přesností ±0,5 %. Snímač má 4 vývody (napájení, data, nezapojený pin a uzemnění) a využívá sběrnici single-bus, pomocí které snímá hodnoty s periodou 2 s. Měření teploty zajišťuje NTC termistor a k měření vlhkosti je ve snímači umístěna kapacitní destička. NTC termistor (anglicky Negative Temperature Coefficient) je teplotně závislý odpor, který mění svoji hodnotu v závislosti na teplotě. Označení „Negative“ znamená, že platí nepřímá úměra.



Obr. 1 – Teplotní závislost NTC termistoru

Destička obsahuje 2 elektrody, mezi kterými je vložena vrstva materiálu (substrát) pro zadržení vlhkosti (sůl nebo vodivý plastový polymer). Když substrát pohltí vlhkost, uvolní ionty, což vede ke zvýšení vodivosti mezi elektrodami. Změna odporu mezi elektrodami je poté nepřímo úměrná relativní vlhkosti. Se stoupající hodnotou relativní vlhkosti klesá hodnota odporu mezi elektrodami. Se snižující se hodnotou relativní vlhkosti poté odpor mezi elektrodami stoupá. Snímač obsahuje 8bitový mikročip SOIC-14, který analogové hodnoty převádí na digitální kalibrované hodnoty teploty a relativní vlhkosti. Pro zaručení správného fungování je potřeba mezi napájecí a datový pin připojit pull-up rezistor o hodnotě 10 kΩ. Povolené napájecí napětí se pohybuje v rozsahu 3,3 V až 6 V (Last Minute Engineers, 2018).

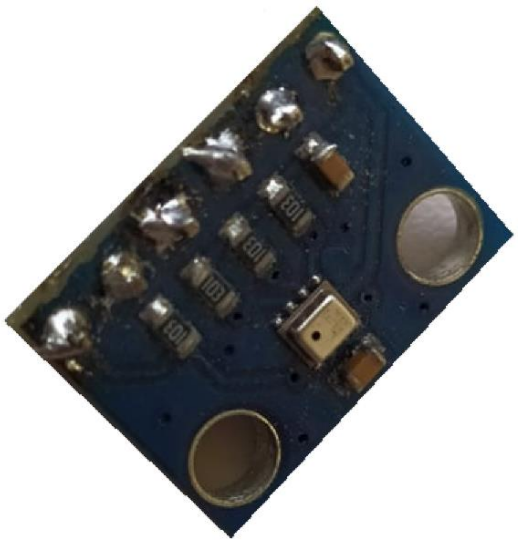
Obr. 2 – Snímač DHT22

2.2 TLAKOVÝ SENZOR BMP280

Snímač BMP280 je určen pro absolutní snímání tlaku v rozsahu 300 hPa až 1100 hPa s relativní přesností ±0,12 hPa (platí pro rozsah 950 hPa až 1050 hPa při teplotě 25 °C). Absolutní znamená, že měřená hodnota barometrického tlaku používá jako referenci tlak naměřený v ideálním vakuu. Snímač také umožňuje měření teploty v rozsahu -40 °C až +85 °C. Na základě naměřené hodnoty barometrického tlaku umožňuje snímač vypočítat nadmořskou výšku s přesností ±1 m. Senzor je schopen snímat hodnoty s frekvencí až 157 Hz (perioda 6,3 ms). K odeslání hodnot do mikrokontroléru využívá I2C nebo SPI sběrnici (*BMP280: Data sheet,* 2015).

Snímač využívá piezorezistivního jevu. Piezorezistivní jev představuje změnu elektrického odporu v závislosti na mechanické deformaci. Jedná se tak o opak piezoelektrického jevu, který generuje napětí při mechanické deformaci. Hodnota tlaku se pak měří pomocí tenzometru zapojeného v můstku na bázi křemíku (Wheatstoneův můstek). V praxi je snímač osazen polovodičovým čipem, který měří tlak pomocí Wheatstoneova můstku vytvořeného pomocí křemíkové membrány a nestlačitelného křemíkového oleje. Výstupní signál v milivoltech je následně tepelně vykompenzován a zesílen na odpovídající hodnotu ve voltech. Nakonec snímač pomocí integrovaného obvodu a A/D převodníku převede analogovou hodnotu na digitální a pomocí zvolené sběrnice odešle hodnotu do mikrokontroléru (Kistler Eastern Europe, neuvedeno).

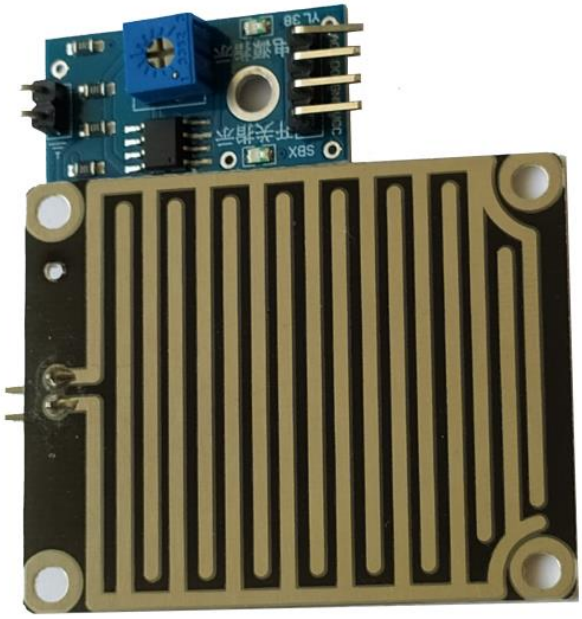
Snímač operuje ve třech režimech: spící, vynucený a normální. Ve spícím módu snímač neprovádí žádná měření a má minimální proudový odběr. Všechny registry jsou dostupné. Ve vynuceném režimu provede snímač jediné měření, uloží hodnoty do registrů a poté se vrací do spícího režimu. V případě potřeby dalšího měření ve vynuceném režimu je nutné tento režim opět vybrat, není tak vhodný pro měření při vysokém množství vzorků. Normální mód pracuje v opakujícím se cyklu, kdy během jeho aktivní části probíhá měření a následuje část neaktivní, které je určena nadefinovaným časem. Čas, po který se snímač nachází v neaktivní části, je určen pomocí registru 0xF5 v rozmezí 0,5 ms až 4000 ms. Výstupní data jsou opět uložena do registrů (*BMP280: Data sheet,* 2015).



Obr. 3 – Snímač BMP280

2.3 DEŠŤOVÝ SENZOR

Dešťový senzor se skládá z porovnávacího modulu a vodivostní podložky. Podložka má na sobě dvě vyleptané vodivé cesty, mezi kterými je malá část nevodivého materiálu. Podložka funguje jako odpor s proměnnou hodnotou, která se mění v závislosti na tom, jak velké množství vody se vyskytuje na povrchu podložky. Čím více je podložka mokrá, tím větší bude její vodivost a tím menší bude její odpor. Výstupní hodnotou čidla je potom napětí naměřené na tomto proměnném odporu. Porovnávací modul pak do mikrokontroléru vrací analogovou hodnotu napětí a pomocí zabudovaného obvodu LM393 ji převádí na digitální. Na modulu je osazen potenciometr, který umožňuje nastavit citlivost převodu z analogové hodnoty na digitální. Modul je také opatřen LED diodou pro indikaci napájení a druhou LED diodou, s jejímž využitím lze potenciometrem přesně nastavit digitální hodnotu. Napájecí napětí senzoru je v rozpětí 3,3 V až 5 V. Nevýhodou tohoto typu senzoru je, že časem dochází vlivem působením deště a venkovních podmínek ke korozi vodivých cest (Last Minute Engineers, 2019).

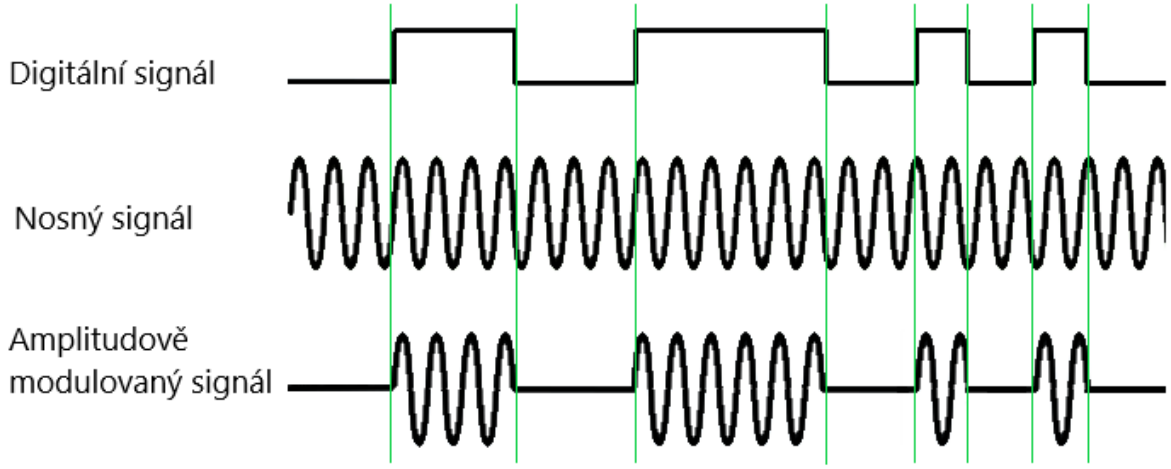


Obr. 4 – Odporový dešťový senzor

2.4 RADIOFREKVENČNÍ OOK VYSÍLAČ A PŘÍJÍMAČ

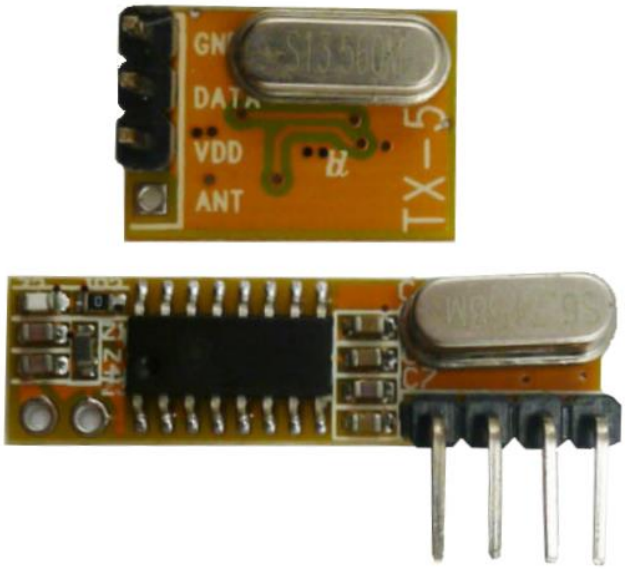
Radiofrekvenční vysílací a příjímací modul (ve zkratce RF vysílač/přijímač) využívá technologii známou jako kmitočtová OOK modulace (on-off keying). Modulace obecně znamená změnu nosného signálu pomocí signálu modulačního. OOK je amplitudová modulace, která reprezentuje digitální hodnoty. Jedná se o zjednodušenou verzi modulace změnou amplitudy nosné vlny (ASK).

Princip OOK modulace spočívá v tom, že se moduluje nosný signál ve formě sinusového průběhu o zvolené frekvenci, v tomto případě 433 MHz. Pokud má být přenášena logická 0, vysílač se vypne a pokud logická 1, tak se pomocí modulátoru vyšle nosný signál s maximálním výkonem. Ke spínání vysílač využívá tranzistor zapojený jako spínač. Přijímač využívá k demodulaci signálu RF tuner, který zpracuje vysokofrekvenční signál a dekóduje z něj data. Signál se následně zesílí pomocí operačních zesilovačů a pošle se na vstup obvodu fázového závěsu. Fázový závěs následně generuje na výstupu signál, jehož frekvence odpovídá vstupnímu signálu, ale již nepodléhá šumu (Rouphael, 2009).



Obr. 5 – Princip OOK modulace

OOK modulace je náchylná na rušení a šum. Pro zlepšení kvality přenosu při delších vzdálenostech je doporučeno připojit k vysílači a přijímači antény. Anténu pro 433 MHz vysílač a přijímač lze realizovat například pomocí vodiče o délce 17 cm.



Obr. 6 – RF vysílač (nahoře) a přijímač

2.5 ARDUINO NANO

Arduino Nano je vývojová deska, která jako řídicí jednotku využívá 8bitový AVR mikrokontrolér (zkráceně MCU) ATmega328p. Deska pracuje s logickou úrovní 5 V na programovatelných výstupních pinech a maximálním výstupním proudem 40 mA. Programování a nahrávání programu do mikrokontroléru je zprostředkováno sběrnicí USB. Desku je možné napájet buď vstupním napájecím konektorem napětím v rozmezí 7 V až 12 V nebo pomocí pinu s označením *VIN*. Tento pin má povolené vstupní napětí v rozmezí 6 V až 20 V a stejně jako vstupní napájecí konektor obsahuje napěťový regulátor, který omezuje připojené napájení na hodnotu 5 V. Arduino obsahuje 8 analogových vstupních pinů a 14 digitálních pinů, které lze definovat buď jako vstupní nebo výstupní. Dále podporuje komunikaci se snímači prostřednictvím sběrnic I2C a SPI (Goel, 2018).

ATmega328p využívá Harvardskou architekturu, která mu umožňuje uchovat informaci uloženou v paměti po odpojení napájení. Označení *p* znamená *pico-power* a značí, že tento konkrétní model má menší proudovou spotřebu oproti klasické verzi ATmega328. Mikrokontrolér obsahuje 32 kB programovatelné paměti typu FLASH (2 kB jsou vyhrazeny pro zavaděč programu – *bootloader*). Dále obsahuje 2 kB operační paměti typu SRAM, 1 kB paměti typu EEPROM a 16Mhz krystalový oscilátor (*ATmega328P [DATASHEET], 2015)*.

Obr. 7 – Arduino Nano

2.6 RASPBERRY PI 4 B

Raspberry Pi 4 model B je jednodeskový počítač s osazeným mikroprocesorem ARM Cortex-A72. Jedná se o plnohodnotný počítač s operačním systémem (primárně Linuxové distribuce). Mikroprocesor má 4 jádra o taktu 1,5 GHz a využívá 64bitovou ARMv8-A SoC architekturu. Zkratka SoC (System on Chip), používá se pro označení integrovaného obvodu, který kromě mikroprocesoru obsahuje i všechny potřebné periferie pro plnohodnotné fungování počítače. V případě modelu 4 B je označení integrovaného obvodu BCM2711. Použitá verze má k dispozici 4 GB operační paměti typu LPDDR4-3200 SDRAM. Jedná se o verzi klasické DDR4-3200 RAM počítačové paměti v odlehčeném provedení s menší spotřebou, na což odkazují první dvě písmena „LP – Low Power“ (Hughes, 2019).

Na desce počítače je také umístěno 40 programovatelných GPIO pinů (General-purpose input/output). Senzory a další periferie je možné připojovat pomocí následujících sběrnic: I2C, SPI, 1-wire a UART. Dále počítač obsahuje Wi-Fi modul (2,4 GHz a 5,0 GHz), Bluetooth modul (verze 5.0), ethernetový konektor, 4 USB porty (2× 3,0 a 2× 2,0) a slot pro mikro SD kartu. Raspberry Pi také obsahuje podporu grafického rozhraní ve formě dvou mikro HDMI portů pro připojení displeje nebo obrazovky a DSI a CSI sběrnice pro připojení oficiálního displeje a kamery. K napájení počítače je potřeba zdroj s výkonem aspoň 15,3 W (*Raspberry Pi 4 Model B Datasheet*, 2019).

Obr. 8 – Raspberry Pi 4 B

2.7 RASPBERRY PI LCD DISPLAY

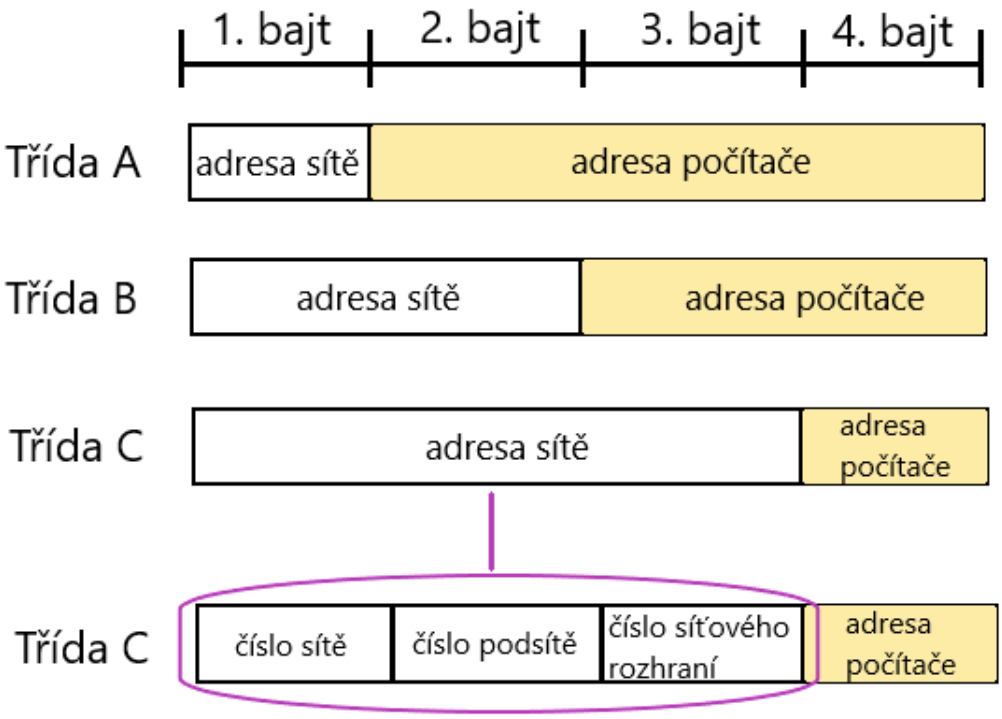
Jedná se o kapacitní dotykový displej využívající technologii LCD (technologie tekutých krystalů). Velikost úhlopříčky použitého displeje je 7 palců (17,78 cm) s rozlišením 800 x 480 pixelů. Displej má na zadní straně připevněnou napájecí DPS (desku plošného spoje). K Raspberry Pi 4 je displej připojen pomocí DSI sběrnice a dvou propojovacích vodičů, které se na jedné straně připojí na napájecí piny DPS a na straně Raspberry Pi na napájecí GPIO piny. Displej podporuje funkci dotyku až v 10 různých bodech najednou. Displej vyžaduje k bezproblémové funkci poslední verzi využitého operačního systému (Raspberry Pi Foundation, 2015).

Obr. 9 – Raspberry Pi display a napájecí DPS

3 KOMUNIKACE

3.1 TCP/IP PROTOKOL

TCP/IP (Transmission Control Protocol/Inthernet Protocol) je označení pro sadu komunikačních protokolů, které pro komunikaci využívají elektronická zařízení připojená k internetu nebo lokální síti LAN (Local Area Network). Aby mohla být data ve formě paketů přenášena mezi dvěma počítači připojenými buď k internetu, nebo lokální síti, musí přenášené pakety obsahovat IP adresu obou zařízení. IP adresa zařízení se skládá ze dvou částí: adresa sítě (net ID) a adresa počítače (host ID). Existují tři použitelné třídy IP adres: A, B a C. V České republice je nejrozšířenější třída C. Konkrétní podobu IP adresy pak udává použitý protokol buď IPv4 nebo IPv6. Protokol IPv4 definuje délku adresy na 32 bitů (Fruhlinger, 2020).



Obr. 10 – Analýza IP adresy

IP adresy v protokolu IPv4 lze tak zapsat buď pomocí CIDR notace: 192.168.1.4/24 nebo ekvivalentně 192.168.1.4/255.255.255.0. Hodnota za lomítkem označuje masku sítě, která slouží jako hranice mezi adresou sítě a adresou počítače. Existují také vyhrazené adresy, které nemůže uživatel využít. Příkladem je takzvaný *loopback*, což je způsob, kterým může posílat počítač packety sám sobě s vyhrazenou adresou 127.x.x.x (Fruhlinger, 2020).

3.2 SSH PROTOKOL

SSH protokol (Secure Shell) je metoda používána k bezpečné komunikaci a přístupu k dalšímu počítači. Bezpečnosti je dosaženo pomocí silného šifrování přenášených dat. Tento protokol je nástupcem nezabezpečeného komunikačního protokolu Telnet. Protokol umožňuje přenos souborů mezi systémy a také vzdálené ovládání počítače, buď pomocí příkazového řádku, nebo vzdálené plochy. Protokol využívá model klient-server. Aby bylo možné se z jednoho počítače (klienta) přihlásit na jiný počítač, musí být na druhém počítači nainstalován SSH server. Autentizace principiálně funguje tak, že se vygeneruje dvojice klíčů – veřejný a soukromý. Veřejný klíč slouží k šifrování dat a je uložen na serveru, zatímco soukromý klíč slouží k dešifrování dat a je uložen v počítači chráněn heslem. Komunikace v případě, že klient již na server uložil veřejný klíč, vypadá následovně. Klient iniciuje komunikaci se serverem na konkrétním portu (nejčastěji č. 22). Server odešle zašifrovanou informaci, kterou klient dokáže dešifrovat pomocí soukromého klíče. Po tomto ověření může začít zabezpečená komunikace mezi oběma účastníky (Ylonen, 1996).

3.3 SBĚRNICE

3.3.1 Sběrnice single bus

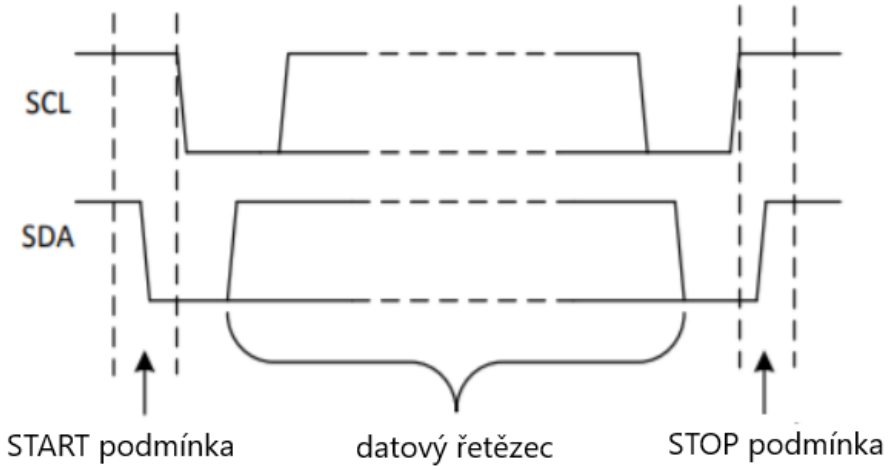
Tuto sběrnici využívá pouze snímač DHT22. Podobá se sběrnici 1-wire tím, že využívá jediný datový vodič, přes který odesílá data do mikrokontroléru. Způsob komunikace je však zcela odlišný, neboť zařízení připojené k 1-wire sběrnici musí obsahovat unikátní adresu, díky které je možné jej při přenosu dat identifikovat. Komunikaci se snímačem zahajuje MCU, tím že sběrnici stáhne do úrovně LOW (logická 0) po dobu aspoň 800 μs. Snímač se poté přepne z výchozího spícího režimu na vysokorychlostní režim. Po uplynutí startovacího signálu odešle snímač odezvu, že počáteční signál obdržel. Poté se vyšle datový řetězec dlouhý 40 bitů, informace o teplotě má rozlišení 16 bitů stejně jako vlhkost, posledních 8 bitů je vyhrazeno pro paritu. Po odeslání řetězce přejde snímač opět do spícího režimu, kde bude vyčkávat do dalšího obdržení startovacího signálu (Waveshare, neuvedeno).

3.3.2 Sběrnice I2C

I2C je poloduplexní sběrnice, která využívá model komunikace master-slave, kdy v pozici master může být více než jedno zařízení, kterých může být připojeno celkem až 128. Jednotlivá zařízení mají unikátní adresy, pomocí kterých, je lze jednoznačně identifikovat při přenosu dat. Zařízení master zahajuje komunikaci a generuje hodinový signál a zařízení slave vykonává případné příkazy, pokud je zařízením master adresováno. I2C využívá dva vodiče: datový vodič SDA (Synchronous Data) a vodič obsahující hodinový signál SCL (Synchronous Clock). Oba signály využívají principu otevřeného kolektoru, to znamená, že mohou sběrnici „stáhnout“ do hodnoty LOW. V opačném případě se uplatní připojený pull-up rezistor a sběrnice nabude hodnoty HIGH (Valdez; Becker, 2015).

Samotná komunikace probíhá ve 3 krocích: master vyšle START podmínku, následně odešle řetězec dat, a nakonec komunikaci ukončí pomocí STOP podmínky. Datový řetězec obsahuje 8bitovou adresu zařízení slave, kde poslední bit uchovává informaci o typu přenosu (čtení-zápis). V tento moment master nastaví sběrnici do stavu HIGH a slave ji stáhne do stavu LOW. Následně slave odešle ACK bit (Acknowedge), který informuje master o připravenosti přijmout data. Následuje 8bitová adresa registru, kam se má informace zapsat a za ní následuje datová osmice bitů. Zařízení slave odesílá ACK bit po každém obdrženém adresovacím a datovém bajtu (Valdez; Becker, 2015).

Master může komunikaci zahájit, jen pokud je sběrnice v klidovém stavu (HIGH). Podmínka START se uplatní v případě, že signál SDA změní hodnotu z HIGH na LOW v okamžiku, kdy signál SCL je ve stavu HIGH. Podmínka STOP je pak definována změnou hodnoty signálu SDA z LOW na HIGH, když je signál SCL ve stavu HIGH. Při odesílání dat se pak může hodnota signálu SDA měnit pouze v případě, že hodinový signál SCL je ve stavu LOW.



Obr. 11 – Princip sběrnice I2C (Valdez; Becker, 2015)

3.3.3 Sběrnice DSI

DSI (Display Serial Interface) je sběrnice, kterou vytvořila MIPI aliance (Mobile Industriy Processor Interface). Jedná se o rychlou sériovou sběrnici, která slouží k připojení displejů (většinou typu LCD) k jednodeskovému počítači (například Raspberry Pi). Výhodou této sběrnice je její nízká náchylnost na elektromagnetická rušení a redukce využitých GPIO pinů na straně jednodeskového počítače (MIPI Alliance, 2015).

Obr. 12 – Sběrnice DSI (Eames, 2015)

4 LINUX

Linux je operační systém s open-source licencí, který v roce 1991 vytvořil Linus Torvalds. Systém je dostupný ve formě jednotlivých distribucí a každá se hodí k jinému druhu operací. Distribuce jsou charakterizovány jádrem (Linux Kernel), balíčkovacím systémem, instalačním programem a podporovaným softwarem. Pro instalaci softwaru na Linuxových distribucích slouží repozitáře, které obsahují balíčky, jenž lze s pomocí balíčkovacího systému jednoduše nainstalovat nebo odinstalovat. Mezi nejznámější distribuce patří Debian (odvozené systémy: Ubuntu, Kali, Knoppix) a Red Hat (odvozené systémy: Fedora, CentOS, Oracle Enterprise). Linux nachází využití v naprosté většině chytré elektroniky a vestavěných systémů (televize, android, chytré hodinky atd.), rovněž je implementován v serverech nebo super-počítačích (The Linux Foundation, 2018).

4.1 RASPBERRY PI OS

Raspberry Pi 4 podporuje širší řadu linuxových distribucí, v této práci je použita distribuce Raspberry Pi OS (dříve nazývaná Raspbian). V podstatě se jedná o Linuxovou distribuci Debian optimalizovanou pro ARM procesory, které jsou osazeny na jednodeskových počítačích. Zmíněný operační systém využívá balíčkovací systém APT (Advanced Packaging Tool). Operační systém je v Raspberry Pi uložen na externí SD kartě, na které musí být nahrán obraz tohoto systému. Na oficiálních webových stránkách raspberry.org je dostupný program s názvem *Raspberry Pi Imager*, který obstará celou instalaci (Thompson, 2012).

5 SOFTWARE

5.1 PYTHON

Jedná se o interpretovaný skriptovací programovací jazyk, který je založen na objektovém programovacím paradigmatu. Interpretovaný znamená, že nevyužívá *překladač*, jako například jazyk C, ale *interpret* (resp. virtuální stroj). Vytvořené programy jsou nazývány skripty, což je obecné označení pro program, který má za úkol automatizovat nějaký nepříliš komplexní úkon. Výhodou Pythonu je jeho široké využití zahrnující například: tvorbu webu, práci s umělou inteligencí nebo vyvíjení softwaru pro vestavěné systémy (Raspberry Pi). Vyvinul jej Guido van Rossum v roce 1991 a aktuálně se používá jeho poslední verze Python 3 (Kothari, 2018).

Důvodem ke zvolení Pythonu v této práci je jmenovitě podpora víceprocesorového systému (multiprocessing), podpora vytváření více vláken v procesu (multithreading) a využití webového mikro frameworku Flask. Dále pak fakt, že se jedná o skriptovací jazyk, který umožňuje automatizovat jednotlivé úkony nezávisle na ostatních.

5.1.1 Multithreading

Python implicitně neumožňuje současné vykonávání více úloh najednou, ale obsahuje modul multihreading, který tento problém částečně řeší. Pojmem vlákno (thread) je v Pythonu myšlena část kódu, která se vykonává nezávisle na zbytku programu uvnitř procesu. Implementace více vláken pak umožňuje paralelní běh více programů s výhodou, že všechna sdílí jmenný prostor (namespace), což ulehčuje vzájemnou komunikaci a sdílení dat. Pokud je vláken více, mohou mezi sebou komunikovat pomocí front. První hodnota, která do fronty vstoupí, jí také na druhé straně jako první opouští. Fronta může mít více vstupů i výstupů.

V Pythonu však existuje omezení ve formě „Zámku globálního tlumočníka“ (Global Interpret Lock – GIL). Ten způsobuje, že vždy pouze jedno vlákno má přístup k interpretu a ostatní musí vyčkat. Nejedná se tak o způsob, jak zajistit opravdový paralelní běh programů. Výhodou oproti modulu multiprocessing jsou menší nároky na využitou paměť (Anderson, 2019).

5.1.2 Multiprocessing

Modul umožňuje vytvoření samostatného procesu, ve kterém může být paralelně spuštěn kód, aniž by byl závislý na ostatních vytvořených procesech. Někdy bývá vytvořený proces z modulu multiprocessing nesprávně zaměňován za subproces. Tímto pojmem je v Pythonu myšlen způsob, jak uvnitř skriptu ovládat a komunikovat s dalšími programy napsanými například v jiných programovacích jazycích. Výhodou oproti modulu multithreading je, že spuštěné procesy skutečně běží paralelně a nejsou omezeny zámkem GIL. Jednotlivé procesy mezi sebou mohou komunikovat buď s využitím front jako v případě modulu multithreading, nebo pomocí rour (pipes). Rozdíl je v tom že rouru si lze představit jako skutečnou trubku, má tak pouze jeden vstup a jeden výstup. Je tak oproti frontě rychlejší (Pieters, 2012).

5.1.3 Mikro framework Flask

Flask je určený pro tvorbu back-end části webové aplikace, ale využívá šablonovací systém Jinja2, který umožňuje vytvoření front-end části. Flask umožňuje vytvoření lokálního serveru, jenž se většinou využívá k testování dané webové aplikace. Označení mikro znamená, že Flask sám o sobě neposkytuje stejné možnosti jako ostatní webové frameworky, ale podporuje rozšíření, díky kterým aplikace může využít funkce, jako kdyby byly implementovány přímo ve Flask. Není součástí základní Python sady a je tak potřeba jej doinstalovat pomocí příkazu: *sudo pip install Flask* (*Flask Documentation, 2010*).

Framework také podporuje knihovnu jQuery. Tato JavaScriptová knihovna umožňuje implementovat dynamické prvky pomocí technologie AJAX (Asynchronous JavaScript and XML). Tato technologie zajišťuje aktualizaci údajů na webové stránce bez nutnosti znovu načíst celou stránku. Toho je dosaženo využitím formátu JSON (JavaScript Object Notation), který předává informace mezi serverem a klientem (Bruijn, 2019).

Flask umožňuje vytvořit cestu mezi URL adresou na webu a funkcí uvnitř skriptu pomocí takzvaného dekoratéru, který má v argumentu webovou adresu. Je možno vytvořit více dekoratérů pro jednu funkci, což znamená, že je možné z webu volat funkci pomocí různých názvů webových adres. Do argumentu je také možno umístit proměnou, která se následně předá funkci jako její parametr nebo takzvané HTTP metody, pomocí kterých generuje žádosti. Každý dekoratér implicitně obsahuje žádost GET, která serveru předává informaci o tom, že má

odeslat data klientovi (webový prohlížeč). Opačným případem je žádost POST, kterou aplikace používá k odeslání dat na server (Nash, 2019).

5.2 SQLITE DATABÁZE

SQLite (verze 3.35.5) patří do skupiny databází, které se označují jako relační. Takto označené systémy ukládají vzájemně propojené datové body a následně k nim poskytují přístup. Jedná se o systém napsaný v jazyce C, který nevyžaduje ke svému fungování server nebo jakoukoliv konfiguraci. Databáze je uložena na pevném disku počítače. Ve vytvořeném souboru (přípona .db) se data ukládají do tabulek, kde má každý řádek své implicitně vytvořené unikátní číselné označení (rowid). Databáze je velmi malá a díky implementaci v jazyce C také rychlá, proto je ideální pro použití ve vestavěných systémech. SQLite lze využít na operačních systémech: Windows, iOS i různých distribucích operačního systému Linux. V případě Linuxu se instalace a nastavení omezuje na následující příkaz: *sudo apt-get install sqlite3*. Mezi podporované datové typy patří: celočíselné hodnoty, hodnoty s plovoucí desetinnou čárkou, text, null a binární hodnoty (blob). Databáze využívá s určitými výjimkami stejné příkazy jako databáze SQL například*: CREATE, DROP, INSERT, SELECT* a *DELETE (Jagtap, 2019).*

Důležitým prvkem SQLite je režim WAL (Write-Ahead Logging). Toto nastavení umožňuje, aby z databáze bylo současně čteno i do ní zapisováno. Toho je docíleno tím, že se mění data ve speciálně vytvořeném souboru wal a nikoli přímo v databázové tabulce (Drake, 2020).

6 NÁVRH A REALIZACE VNITŘNÍHO SYSTÉMU

Pro realizaci vnitřní části bylo zvoleno Raspberry Pi 4 B jako výpočetní jednotka zpracovávající veškeré informace. Důvodem, proč ve vnitřní části není využito Arduino nebo jiný 8bitový kontrolér, je, že Raspberry Pi využívá 64bitový mikroprocesor ARM Cortex-A72 a jedná se tak o plnohodnotný počítač s vlastním operačním systémem. Tím pádem není jen uživatelsky mnohem přívětivější, ale také se více hodí pro náročnější operace vyžadující větší výpočetní výkon a operační paměť. Rychlým příkladem je následující porovnávací tabulka mezi mikrokontrolérem ATmega328p a mikroprocesorem ARM Cortex-A72.

Tabulka 1 – Porovnání parametrů řídicích jednotek

Parametry	AVR ATmega328p	ARM Cortex-A72
Architektura	8 bitů	64 bitů (4 jádra)
Taktovací frekvence	16 MHz	1,5 GHz
Operační paměť	2 KB (SRAM)	4 GB (LPDDR4 SDRAM)

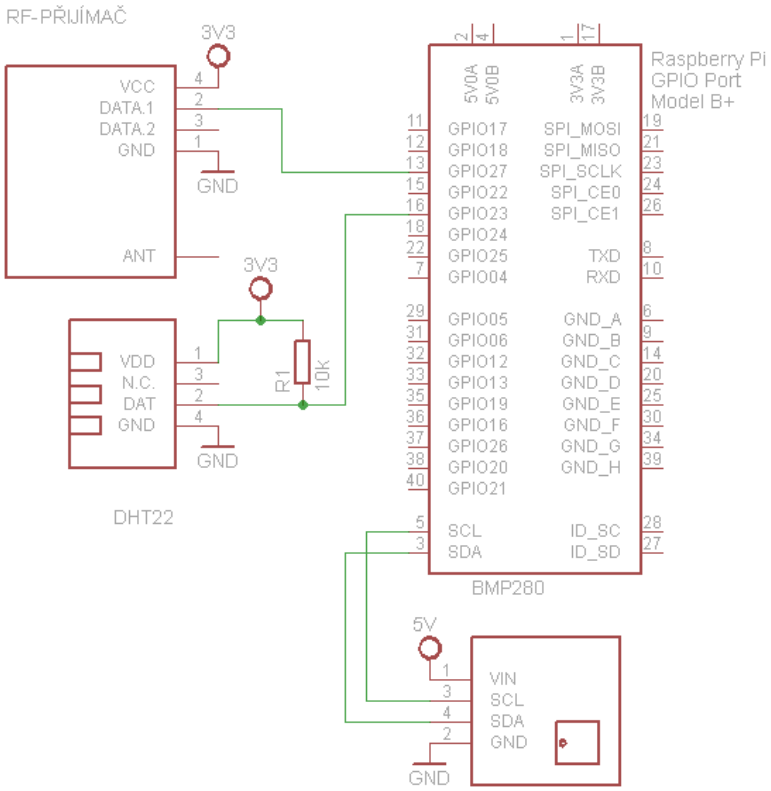
Z porovnání vyplývá, že ARM mikroprocesor je oproti AVR mikročipu víc jak 90× rychlejší a má k dispozici mnohonásobně větší množství operační paměti.

Dalším důvodem ke zvolení Raspberry Pi je podpora skriptovacího programovacího jazyka Python, který je ideální pro realizaci podobných projektů, kde je potřeba provozovat větší množství asynchronních procesů v reálném čase.

Snímače DHT22, BMP280 a RF 433MHz přijímač jsou zvoleny s ohledem na nízkou pořizovací cenu, dobrou dostupnost na trhu a jednoduchou implementaci v projektu díky podpoře softwarových knihoven. V původním návrhu práce se počítalo s použitím senzoru reálného času (DS3231) a použitím 3,5palcového LCD displeje od firmy KeDei (připojení přes SPI sběrnici). Nakonec nedošlo k jejich použití z důvodu špatné softwarové podpory, v případě DS3231 bylo čidlo úplně odstraněno, protože Raspberry Pi disponuje vlastním zdrojem hodin. Displej byl nahrazen oficiálním Raspberry Pi LCD kapacitním dotykovým displejem o větší úhlopříčce (7 palců) připojeným přes DSI sběrnici.

6.1 SCHÉMA ZAPOJENÍ

Schéma zapojení vnitřní části bylo vytvořeno v programu EAGLE 7.0.0. K jeho vytvoření byla využita importovaná knihovna: *raspberrypi_bastelstube_v13.lbr*.



Obr. 13 – Zapojení vnitřního systému

Tabulka 2 – Zapojení vývodů RF přijímače

RF přijímač 433 MHz	Raspberry Pi 4 B
GND	GND (pin č. 9)
VCC	3V3 (pin č. 1)
DATA	GPIO 27 (pin č. 13)

Tabulka 3 – Zapojení vývodů snímače BMP280

BMP280	Raspberry Pi 4 B
GND	GND (pin č. 9)
VCC	5 V (pin č. 2)
SCL	GPIO 3 (pin č. 5)
SDA	GPIO 2 (pin č.3)

Tabulka 4 – Zapojení vývodů snímače DHT22

DHT22	Raspberry Pi 4 B
GND	GND (pin č. 9)
VCC	3V3 (pin č. 1)
DATA	GPIO 23 (pin č. 16)
NC	nezapojeno

6.2 ZAPOUZDŘENÍ VNITŘNÍ ČÁSTI

Z praktických důvodů a s ohledem na případná budoucí vylepšení je Raspberry Pi ponecháno připevněné na zadní straně displeje bez dalšího zapouzdření. Snímače jsou uloženy v montážním boxu, který funguje jako samostatný modul a je možné jej k desce připojit pomocí PATA (Parallel AT Attachment) sběrnice.

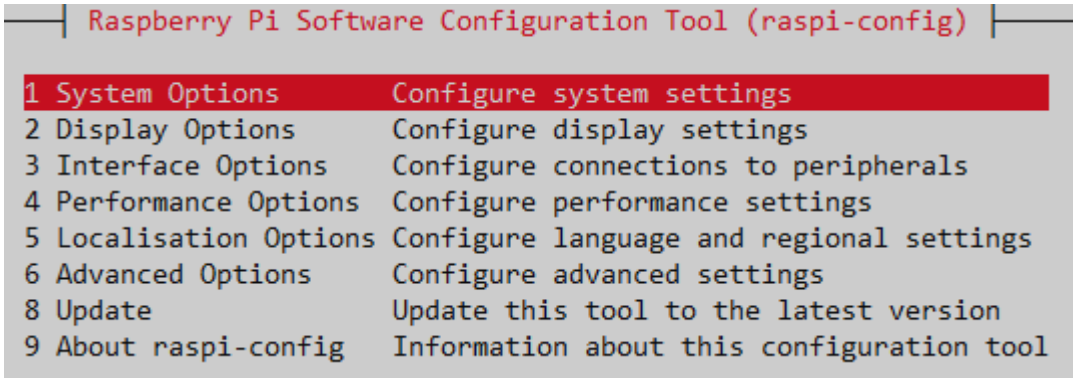
Obr. 14 – Zapouzdření vnitřní části s otevřeným montážním boxem

Obr. 15 – Zapouzdření vnitřní části s uzavřeným montážním boxem

6.3 NASTAVENÍ

6.3.1 Nastavení Raspberry Pi

Před psaním samotného obslužného programu je nutné nejdříve zprovoznit potřebné funkce v Raspberry Pi. Po nainstalování vybraného operačního systému lze povolit používání potřebných sběrnic a komunikačních protokolů pomocí příkazu: *sudo raspi-config*. V části *Interface Options* se aktivují využité sběrnice SPI, I2C a 1-Wire a služby VNC, SSH a Remote GPIO.



Obr. 16 – Konfigurační prostředí v Raspberry Pi Os

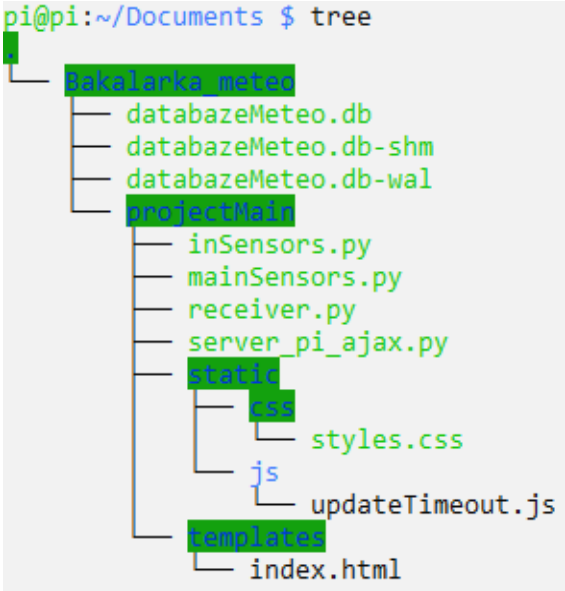
Dále je potřeba v záložce *System Options* nastavit Wi-Fi síť, která se využívá pro přístup k Raspberry pomocí SSH a v záložce *Localisation Options* nastavit Wi-Fi kanály povolené pro Českou republiku. V programu je využita SQLite databáze, kterou je potřeba nejdříve nainstalovat do Raspberry Pi příkazem: *sudo apt install sqlite3*. Dále se musí stáhnout knihovny pro komunikaci s čidly DHT22, BMP280 a RF přijímačem, buď prostřednictvím terminálu, nebo až ve zvoleném vývojovém prostředí PyCharm IDE. Je nutno podotknout, že knihovny k těmto snímačům jsou velice chudě zdokumentované a jejich softwarová podpora nepatří k nejlepším. V programu je tak potřeba ošetřit určité chyby, které vznikají právě vlivem použití některých knihoven.

6.3.2 Nastavení vývojového prostředí PyCharm IDE Professional.

Obslužný program přijímací vnitřní části je napsán v jazyce Python (verze 3.7). Před samotným vývojem kódu je žádoucí přenastavit výchozí verzi (2.7) a instalační nástroj balíků pip. Je nutné vytvořit „alias“ v souboru .bashsrc, který se spustí na místo původních verzí. Tento soubor se nachází v adresáři /home/pi a volá se pokaždé, když se otevře terminál nebo dojde k využití SSH. Na konec souboru je potřeba připsat *alias python = '/usr/bin/python3'*

a *alias pip = pip3*. Když se nyní spustí program z terminálu v podobě: *python soubor.py*, tak je soubor spuštěn pomocí interpretu ve verzi 3.7.

Stěžejním bodem práce byla instalace PyCharm IDE Professional verze na osobní počítač, která umožňuje využití vzdáleného interpretu. To znamená, že program je možné psát na osobním počítači a pomocí SSH protokolu nahrát program s využitím lokální sítě do Raspberry Pi. Pro správnou funkci je ještě potřeba vytvořit vzdálený interpret v samotném PyCharm IDE v záložce nastavení/projekt. Je nutné nakonfigurovat IP adresu, uživatelské jméno, heslo a cestu k interpretu uloženého v Raspberry Pi. Nakonec je nutné nastavit složku, kam se budou vytvořené programy ukládat. Poslední krok je důležitý, protože výchozí nastavení způsobí, že se vytvořené programy uloží do adresáře s názvem /tmp, který se smaže po každém vypnutí systému. V rámci udržení přehlednosti kódu byl program rozdělen do složek se stromovou strukturou, kterou zobrazuje Obr. 17.



Obr. 17 – Stromová struktura programu v Raspberry Pi

Adresář *Bakalarka_meteo* obsahuje hlavní a přidružené soubory, ve kterých je uložena databáze a podadresář *projectMain*. Zde jsou uloženy všechny jednotlivé části programu v sestupném pořadí: obsluha vnitřních snímačů, hlavní proces, proces pro přijímání dat a serverový proces. Ve složce *static* jsou uloženy soubory s příponou css a js, upravující vzhled a funkcionalitu webové aplikace. V poslední složce s názvem *templates* je uložena samotná webová stránka.

6.3.3 Nastavení statické IP adresy a vytvoření databázového systému

IP adresy jsou v Raspberry Pi (a počítačích obecně) nastavovány implicitně jako dynamické pomocí protokolu DHCP (Dynamic Host Configuration Protocol). Typicky se takto nastavené adresy obnovují minimálně jednou denně, jejich využití je proto při vývoji programu přes vzdálený interpret zcela nevhodné. Adresu je tak potřeba nastavit jako statickou, aby nedocházelo k jejímu přepisování. Toho lze dosáhnout pomocí následujících příkazů: *cd /etc* (přepnutí do adresáře etc), *sudo nano dhcpcd.conf* (otevření souboru pro editaci s právy sudo). Na Obr. 18 je ukázáno, co je potřeba dopsat na konec souboru dhcpcd.conf.

```
interface wlan0
static ip_address=192.168.1.4/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Obr. 18 - Nastavení statické IP adresy

Nastavení statické IP adresy je nutné především kvůli využití SSH, jak pro možnost programování pomocí vzdáleného interpretu v PyCharm IDE, tak pro ovládání Raspberry Pi pomocí terminálu. Aby bylo možné v dále popsaném programu využívat databázi, je nutné vytvořit soubor s příponou *.db*, ve kterém budou následně uloženy databázové tabulky. Nejdříve je nutné přepnout se z aktuální pracovní složky do adresáře, kde se má soubor vytvořit s využitím příkazu *cd /home/pi/Documents/Bakalarka_meteo* a následným příkazem *sqlite3 databazeMeteo.db* založit samotný soubor.

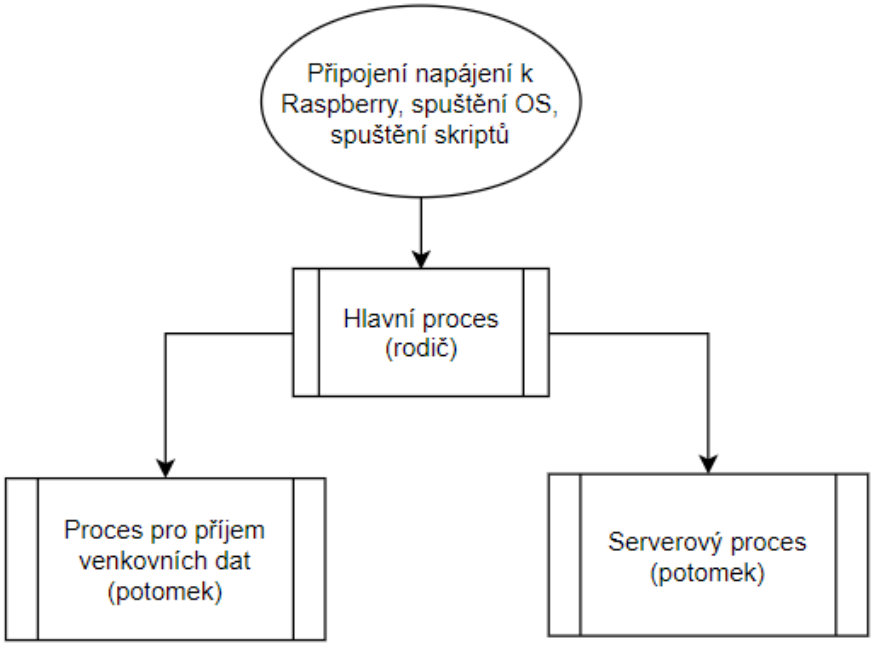
6.3.4 Importované knihovny

Program využívá pro jednodušší komunikaci se snímači importovaných knihoven třetích stran. Tyto knihovny jsou uvedeny v tabulce 5.

Tabulka 5 – Použité knihovny třetích stran

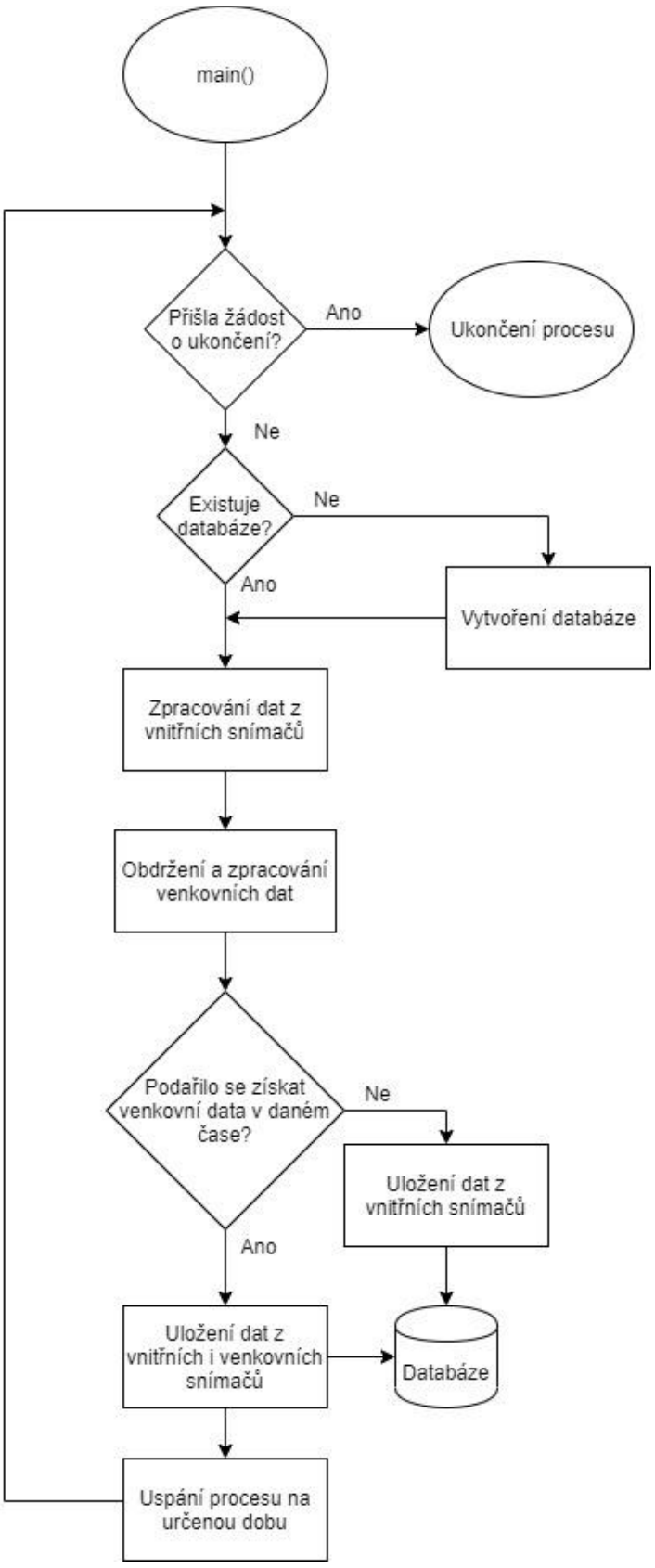
Název	Instalační příkaz	Použití
rpi_rf	*sudo pip install rpi_rf*	obsluha RF přijímače
adafruit_dht	*sudo pip install adafruit-circuitpython-dht*	obsluha DHT22 čidla
adafruit_bmp280	*sudo pip install adafruit-circuitpython-bmp280*	obsluha BMP280 čidla
matplotlib	*sudo apt-get install python3-matplotlib*	vykreslení grafů

6.4 PRINCIP PROGRAMU



Obr. 19 – Vývojový diagram syntézy programu

Cílem programu je přijmout data z venkovní části pomocí RF přijímače, získat data ze snímačů připojených k Raspberry Pi, uložit všechna data do databáze a následně je nahrát na lokální server. Z důvodu optimalizace kódu došlo k jeho rozdělení na 3 separované části pomocí modulu multiprocessing. Pouze tak lze dosáhnout, aby jednotlivé navržené části programu operovaly paralelně a nezávisle na ostatních. V rodičovském procesu je vytvořena funkce *main()*, která dále představuje hlavní proces.



Obr. 20 – Vývojový diagram hlavního procesu

Před vstoupením do samotného cyklu ve funkci *main()* dojde k nastavení SQLite souboru databazeMeteo.db do režimu WAL. To je nutné kvůli tomu, aby v okamžiku, kdy funkce *main()* ukládá data do databáze, mohl serverový proces z této databáze zároveň číst. Následně dojde k vytvoření a spuštění zbylých dvou procesů: *p_receiver – příjem venkovních dat* a *p_server* – odeslání údajů na web. Mezi hlavním a přijímacím procesem je vytvořena roura, která slouží k přenosu přijatých dat do funkce *main()*, kde se následně uloží do databáze.

```
parent_receiver, child_sender = Pipe()
p_receiver = Process(target=start_receiver, args=(child_sender,))
p_server = Process(target=server_ajax) #server_fnc
p_receiver.start()
p_server.start()
main()
```

Obr. 21 – Vytvoření serverového a přijímacího procesu

6.4.1 Ošetření chyb importované knihovny

Ve funkci *main()* se poté hlavní proces zacyklí a nadále obstarává pouze příkazy uvnitř *while* cyklu. Zde se jako první volá funkce *in_sensors()* ze skriptu inSensors.py, která získává hodnoty z vnitřních čidel. Samotné čtení ze snímačů je implementováno pomocí *try-except-finally* bloku z důvodu již zmíněné špatné softwarové podpory čidel. Především se jedná o knihovny adafruit_dht a adafruit_bmp280. Z dostupných informací od tvůrců knihoven (firma Adafruit) je zřejmé, že se jedná o problém s časováním, kdy se snímače snaží změřit hodnoty v moment, kdy nejsou zcela inicializovány. Tento problém lze účinně vyřešit pozastavením programu na krátký čas (2 sekundy) a opětovným využitím zmíněného bloku. Tím se docílí toho, že pokud se objeví chyba, program kvůli ní nebude ukončen, místo toho se pouze vypíše chybová hláška do konzole. Pokud se chyba při snímání hodnot objeví, tak funkce vrací proměnnou *error*, která obsahuje informace o výskytu chyby a znemožní zápis této hodnoty do databáze.

Ve funkci *main()* se následně pomocí roury získají data z přijímacího procesu. Protože může dojít k situaci, že data nedorazí (venkovnímu vysílači například dojdou baterie), je zde implementována funkce *poll(timeout=8).* Ta způsobí, že pokud nedojde k přijetí všech dat do 8 s (pokud se nevyskytne problém, jsou data přijata do 2 s), komunikace mezi procesy se v této iteraci průchodu cyklem vynuceně zastaví a program pokračuje dalším příkazem v cyklu. Pokud nastala chyba, uloží se do databáze pouze hodnoty z vnitřních čidel.

6.5 DATABÁZE

Pokud je program spuštěn poprvé, dojde k vytvoření dvou databázových tabulek, aby se mohla ukládat zvlášť data z venkovní a vnitřní části. Tabulky jsou pojmenovány „web_data_out“ a „web_data_in“, jejich struktura je zobrazena na Obr. 22.

```
("CREATE TABLE IF NOT EXISTS web_data_out (time TEXT, outTemp REAL,outHum REAL, rain TEXT)")
("CREATE TABLE IF NOT EXISTS web_data_in (time TEXT, inTemp REAL, inHum REAL, press REAL)")
```

Obr. 22 - Syntax databázových tabulek

Proměnná *time* typu *TEXT* obsahuje aktuální čas, uložený jako text ve formátu: *“%Y-%m-%d %H:%M:%S“*, což odpovídá zobrazení (náhodné datum pro demonstraci): 2021-03-10 09:50:00. Tento formát je důležitý z hlediska další operace s databází v serverovém procesu.

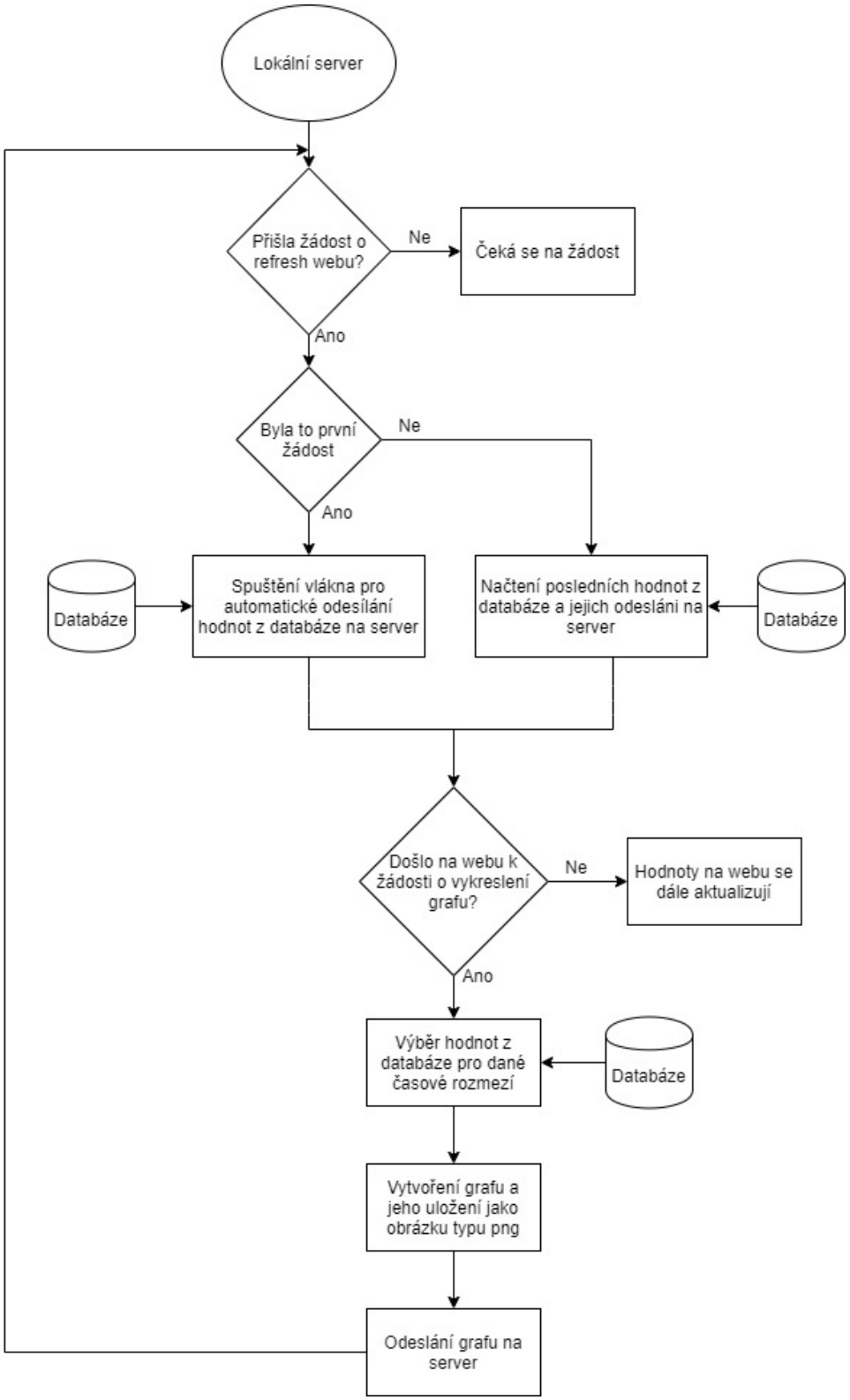
6.5.1 Validace a ošetření přijatých venkovních hodnot

S ohledem na použité komponenty a jejich nepříliš vysokou spolehlivost je také potřeba kontrolovat ukládání dat do databáze. Na straně přijímače totiž ojediněle dochází k přijetí dat, která jsou řádově buď naprosto mimo logický rozsah měřených hodnot nebo se liší například jen v řádech desítek. Odhadem se tato chyba vyskytne 1× za 20 přijatých hodnot.

Řešení je implementováno formou algoritmu, který využívá objekt *Counter* z modulu *collections*. Po každém uložení do tabulky: web_data_out se zavolá vytvořená funkce s názvem *remove_corrupt_out_db()*. Tato funkce vybere posledních 10 záznamů teploty a vlhkosti a vytvoří jejich modus (hodnotu, která se v množině vyskytuje nejčastěji). Následně se vytvoří horní a dolní limit povolených hodnot přičtením definovaného offsetu k modu (±20). V databázi se následně projde posledních 10 záznamů a všechny hodnoty, které neodpovídají povolenému rozmezí se pomocí příkazu *UPDATE,* nahradí hodnotou příslušného modu. Mírně se tak sníží přesnost měření, ale nedojde k narušení integrity a množství dat. Pomocí *try-except* bloku jsou ošetřeny případné statistické chyby vzniklé v případě, že není k dispozici dostatek záznamů (situace při získání prvních 10 měření, pokud je databáze prázdná).

Z důvodu optimalizace rychlosti programu se prochází pouze posledních 10 hodnot. Pokud by se procházela všechna naměřená data a z nich se následně vytvořil modus, při větším počtu dat by tato operace mohla být potenciálně velmi náročná pro mikroprocesor.

6.6 SERVEROVÝ PROCES



Obr. 23 – Vývojový diagram serverového procesu

6.7 WEBOVÁ APLIKACE – BACK END

Back-endová část programu je implementována pomocí webového frameworku jazyka Python nazvaného Flask. Aby mohl být server vytvořen a spuštěn, je nutné vytvořit instanci třídy Flask: *app = Flask(__name__)*. Proměnná *__name__* obsahuje jméno modulu, ve kterém se právě nachází a umožňuje aplikaci přiřadit potřebné systémové soubory. Samotný server se spustí příkazem, jenž je uveden na Obr. 24.

```
app.run(port=80, host='0.0.0.0',debug=True, use_reloader = False)
```

Obr. 24 - Příkaz ke spuštění serveru

Parametr *debug=True* umožňuje změny kódu v reálném čase bez nutnosti zastavování programu, *port=80* nastavuje číslo portu a poslední parametr *host='0.0.0.0'* zajišťuje, že webová stránka bude dostupná pro všechna zařízení nacházející se v lokální síti.

Základním prvkem je funkce *index()*. Tato funkce se prostřednictvím dekoratéru *@app.route('/')* volá pokaždé, když dojde k navštívení lokálního webu na URL adrese odpovídající IP adrese Raspberry Pi (192.168.1.4). Toho se docílí tak, že metoda při zavolání vrátí metodu *render_template('index.html', **pom_template),* která vyhledá ve složce *templates* HTML soubor s názvem „index.html" a předá mu proměnnou typu slovník s názvem *pom_template*. Data předaná do slovníku jsou získána pomocí funkce *data_on_reload()*, která vybere všechny (nejnovější) žádané údaje z databáze.

Protože je data nutno pravidelně aktualizovat (zvolená obnovovací perioda je 1 s), je potřeba zajistit automatizované odesílání na web. Tato implementace je zajištěna pomocí separovaného vlákna, které je vytvořeno taktéž ve funkci *index()* při jejím prvním zavolání. Vlákno následně spustí funkci *periodic_web_data()*, která čte hodnoty z vnitřní i venkovní databáze s periodou 1 s. Tyto údaje získané z databáze jsou ve funkci deklarované jako globální z důvodu, aby k nim měla přístup i další funkce. Primárně jich využívá funkce *update()*, která s pomocí AJAX na front-endové straně serveru data aktualizuje. Aby mohla být data přenesena, je potřeba dekorátor funkce napsat následovně: *@app.route('/update', methods=['POST']).* Funkce *update()* pak pouze při zavolání (které na front-endové straně probíhá každou sekundu) vrátí pomocí funkce *jsonify()* proměnnou typu slovník s požadovanými proměnnými.

6.7.1 VYTVOŘENÍ GRAFU

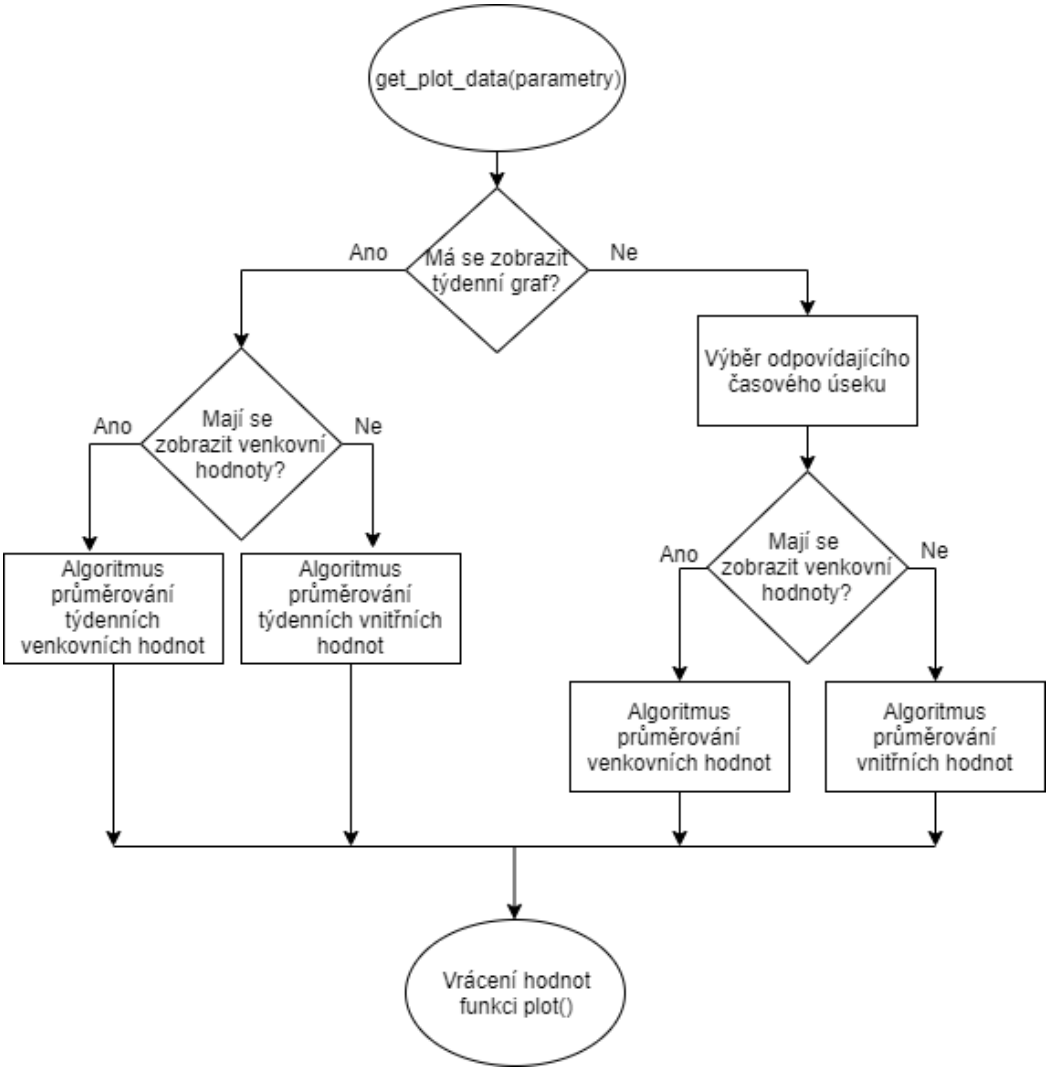
K vytvoření grafu a jeho následnému odeslání na server slouží funkce *plot(location_id, device_id, duration_id).* Funkce má dekorátor, který v URL přijímá tři proměnné, které se zadávají na webu stiskem tlačítka, které generuje žádost o příslušný graf. Proměnná *location_id* nastavuje, jestli se jedná o graf vnitřních nebo venkovních hodnot, *device_id* udává jaký druh hodnot je požadován (tlak, teplota, vlhkost) a *duration_id* definuje časový úsek pro který se má graf vygenerovat.

```
@app.route('/<location_id>/<device_id>/<duration_id>')
def plot(location_id, device_id, duration_id):...
```

Obr. 25 - Dekorátor metody plot

Pro vytvoření grafů je využita knihovna *matplotlib*, která umožňuje využívání podobných funkcí jako jazyk MATLAB. Funkce *plot()* graf uloží jako obrázek typu PNG a následně jej odešle na server jako objekt typu response. Program umožňuje vygenerovat graf pro 5 pevně zvolených uplynulých časových úseků: 1 hodina, 6 hodin, 12 hodin, 1 den a 1 týden. Podle toho se také mění popisy os, například pro časové úseky kratší než 1 týden se zobrazují pouze hodiny a minuty na ose *X* (formát *H:M*), pro záznam hodnot z celého týdne se pak změní popisek na formát (náhodná hodnota pro demonstraci): 20-03 *mo*, kde poslední 2 písmena anglicky označují konkrétní den. Pro získání hodnot se volá metoda *get_plot_data()* s výše zmíněnými parametry. Volání funkce mění tvar na základě parametrů *location_id* a *duration_id*.

Funkce *get_plot_data(_location_id, _device_id, _duration_id)* pak obsahuje dva typy algoritmů, jeden pro výběr hodnot od 1 hodiny do 1 dne a druhý pro týdenní záznam hodnot. V případě obou algoritmů je cílem vybrat rozmezí požadovaných hodnot od – do, kdy v prvním případě jedna z hodnot vždy odpovídá poslednímu vzorku v databázi a druhou je potřeba dohledat a v druhém algoritmu je potřeba vyhledat obě hodnoty.



Obr. 26 – Vývojový diagram funkce get_plot_data()

6.7.2 Algoritmus výběru hodnot pro časové rozmezí 1 hodina až 1 den

Nejdříve se zjistí počet všech vzorků (pro venkovní i vnitřní databázi), to je zajištěno pomocí implicitně vytvořeného rowid sloupce v databázové tabulce, kde nejvyšší hodnota odpovídá nejnovějšímu údaji. Při vzorkovací periodě 10 minut se získá za hodinu 6 záznamů hodnot, pokud by v databázi bylo celkem 1500 jednotlivých záznamů, tak odpovídající rozmezí získané pomocí rowid je 1500 až 1495. Tím se získá odpovídající počet hodnot pro vykreslení grafu na zvolený časový úsek, což by v tomto případě byla 1 hodina. Hodnota rowid nejstaršího požadovaného vzorku, který tvoří druhou část rozmezí (v tomto případě 1495), se získá tak, že se vezme aktuální hodnota času v datovém typu *datetime* a odečte se od ní proměnná typu *datetime.timedelta(hours=_duration_id)* . Tato hodnota se následně převede na text pomocí *datetime.strftime* funkce. Podle této hodnoty se v databázi vyhledá příslušný řádek.

```
if _duration_id == '1h':
    curs.execute("SELECT rowid,* FROM dbInCsv4 WHERE time = ?", [timeHourString])
    pom = curs.fetchall()
    for row in pom:
        hourId = max_samples - row[0]
    number_of_samples = hourId
```

Obr. 27 – Vyhledání hodnoty rowid pro hodinu starý údaj

V proměnné *number_of_samples* je poté uloženo výsledné množství vzorků, které bude potřeba z databáze získat (bráno sestupně podle času a rowid).

```
curs.execute("SELECT rowid,* FROM dbInCsv4 ORDER BY rowid DESC LIMIT ?", [number_of_samples])
pom3 = curs.fetchall()
for row in pom3:
    array_time.append(row[1])
    array_temp.append(float(row[2]))
    array_hum.append(float(row[3]))
    array_press.append(float(row[4]))
```

Obr. 28 – Vlastní získání jednotlivých údajů z databáze cyklem for

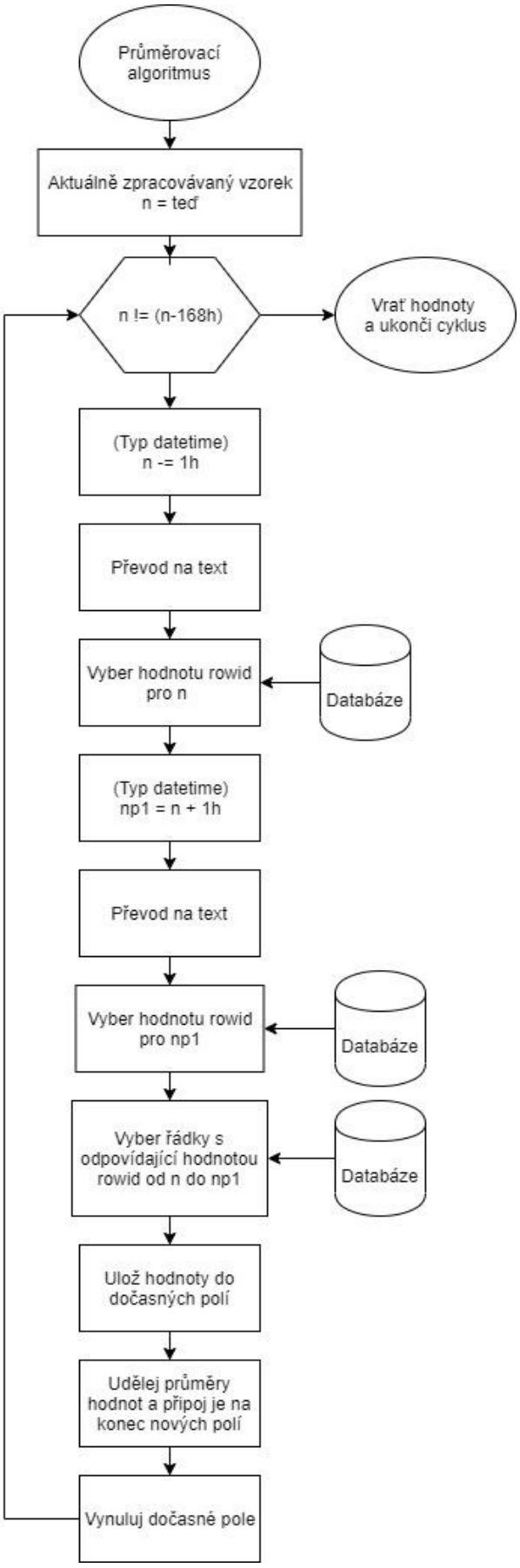
Pomocí funkce *fetchall()* se uloží všech 6 řádků z databáze do smíšeného listu hodnot. Následně se ve for cyklu hodnoty z tohoto listu uloží do dalších separovaných listů. Tyto listy hodnot jsou potom funkcí vráceny do funkce *plot()*, kde ještě dojde k otočení pořadí hodnot v listech, protože vlivem výběru hodnot z databáze jsou seřazena v opačném pořadí, než je potřebné pro tvorbu grafu. Hodnoty na ose *X* odpovídají časům jednotlivých vzorků. Jelikož list vrácený do funkce *plot()* je typu *TEXT*, je nutné jej ještě převézt na datový typ *datetime*,

aby bylo možné hodnoty naformátovat odpovídajícím způsobem. K převedení se využije funkce *datetime.strptime.*

6.7.3 Algoritmus pro výběr hodnot starých 1 týden

Smysl tohoto algoritmu spočívá v tom, že je potřeba snížit výsledný počet hodnot čtených ze zvolené databáze při zachování stejné přesnosti zobrazovaných hodnot jako při původním počtu vzorků. Hodnoty je tedy potřeba zprůměrovat. Konkrétně se vezmou hodnoty za uplynulou hodinu a pomocí *statistic* modulu se udělá jejich aritmetický průměr, který se následně uloží do listu jako záznam odpovídající dané hodině. Ve výsledku se tak místo 1008 hodnot využije 168 hodnot zprůměrovaných.

Opět se vyhledává nejstarší požadovaná hodnota pomocí rozdílu aktuálního času ve tvaru *datetime.now() - datetime.delta(days=7)* s tím, že je opět nutno tuto hodnotu pro vyhledávání v databázi převést na text. Rozdílem však je, že v tomto případě je potřeba projít týdenní časový úsek po jednotlivých hodinových úsecích, aby bylo možné hodnoty průměrovat. To znamená, že rozmezí od – do nebudou fixní hodnoty jako v prvním algoritmu, ale budou se posouvat po časové ose týden zpět po jednotlivých hodinách. Programově je algoritmus implementován jako cyklus, který posouvá čtené hodnoty z databáze o hodinu zpět na základě *datetime* proměnné. Od této proměnné se při každé iteraci cyklu odečte hodina a skončí v okamžiku, kdy se přistoupí v databázi na řádek, kde datum odpovídá zmíněnému rozdílu hodnot. Vždy se tedy získávají dvě hodnoty rowid řádků zároveň, pro *n* hodin starý údaj a *n+1* hodin starý údaj (ve vývojovém diagramu je *n+1* označeno jako *np1*).



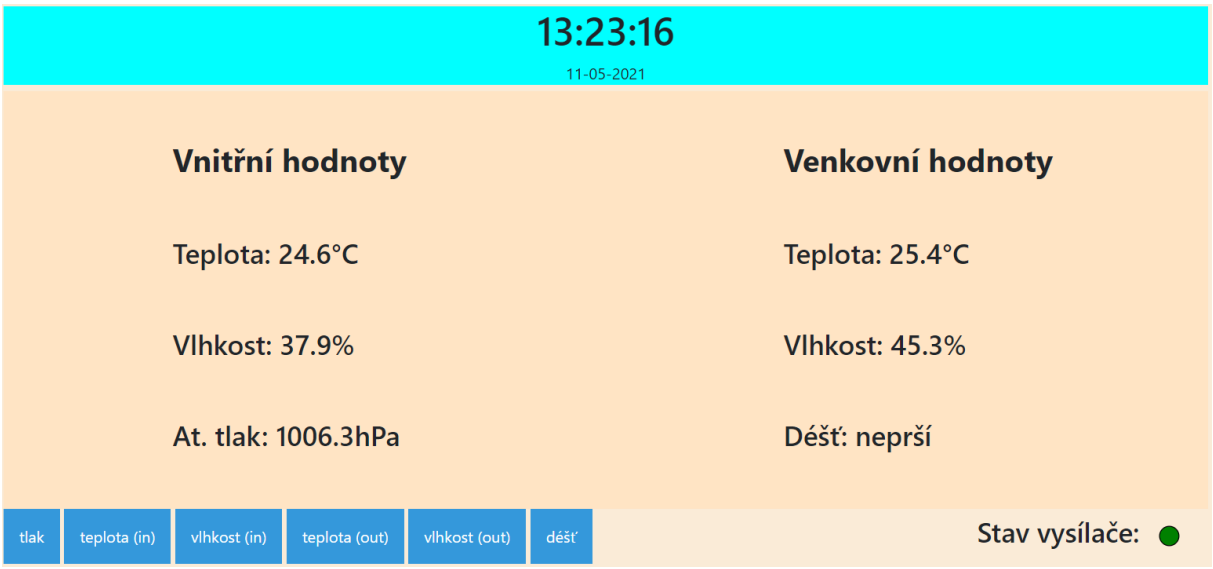
Obr. 29 – Vývojový diagram algoritmu k průměrování týdenních hodnot

6.8 WEBOVÁ APLIKACE – FRONT-END

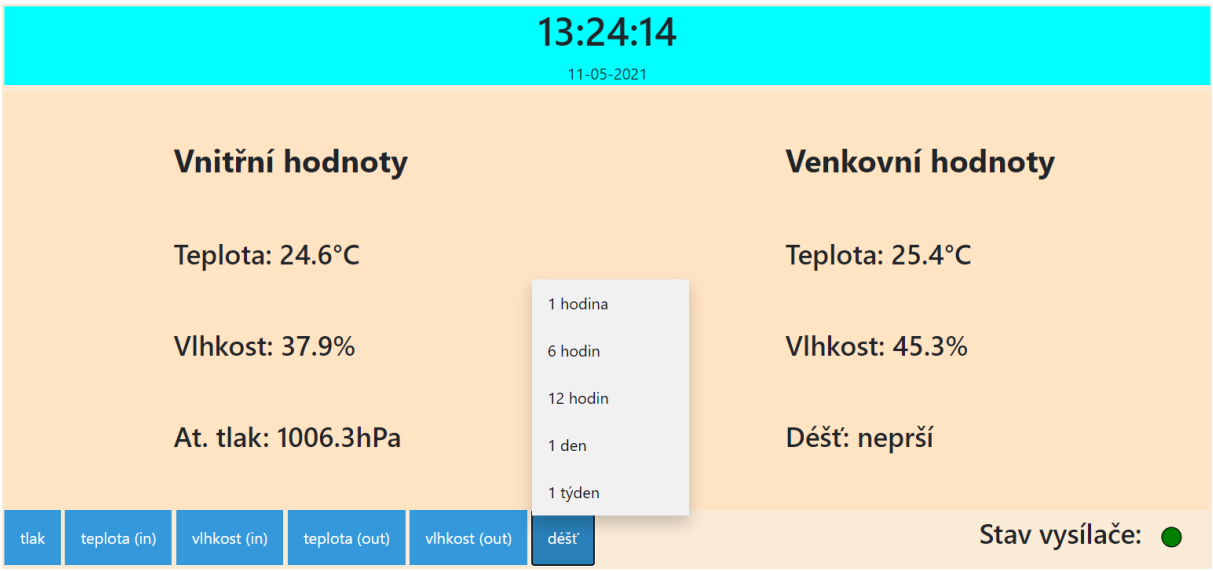
Samotná front-endová část webu je dostupná na IP adrese 192.168.1.4. Je vytvořena pomocí jazyka HTML5 a kaskádových stylů CSS3. Webová aplikace je tvořena jedinou stránku s názvem index.html uloženou ve složce templates a využívá nadefinovaných stylů ze souboru styles.css uloženého ve složce static/css. Ve složce static je dále podsložka s názvem js, ve které je uložený JavaScriptový soubor updateTimeout.js. Jak již bylo zmíněno dříve, tento soubor využívá technologii AJAX pro automatické obnovování dat s periodou jedné sekundy. Soubor styles.css kromě prvků, které nastavují vzhled stránky, také obsahuje atribut *@media*, který nastavuje, jak se stránka bude zobrazovat pro zařízení s různou velikostí displeje. Konkrétně je rozdíl v zobrazení mezi Full HD zobrazením při 1920x1080p a 800x600p rozlišením, na které je nastavený použitý LCD 7palcový displej.

Webová aplikace je rozdělena na záhlaví, tělo a zápatí. V záhlaví je zobrazen aktuální čas a datum. Tělo stránky má formu široké tabulky o dvou sloupcích, ve kterých se nacházejí všechny průběžně aktualizované údaje. Ty jsou zobrazovány s využitím značky <*span*> a atributu *id*, který odkazuje na javascriptový soubor. Poslední řádek informuje o tom, k jakému času a datu jsou informace na webu vztaženy.

V zápatí stránky jsou nadefinovány *drop-up* tlačítka, která generují žádosti o vykreslení grafů. To je zajištěno pomocí odkazů s nadefinovanou syntaxí, které odpovídají vstupním parametrům funkce *plot()*. V pravém kraji zápatí se nachází grafická indikace stavu venkovního vysílače. Zelené kolečko znamená, že je vysílač aktivní, červené znamená, že je neaktivní buď z důvodu vybité baterie nebo jiné poruchy. Tato indikace, která mění barvu v závislosti na stavu proměnné *battery* (vytvořeno v hlavním procesu), je implementována pomocí podmíněných bloků, které lze vepsat přímo do HTML5. To je možné díky integraci šablonovacího systému Jinja2 do frameworku Flask.



Obr. 30 – Vzhled webové aplikace (zobrazení na PC)



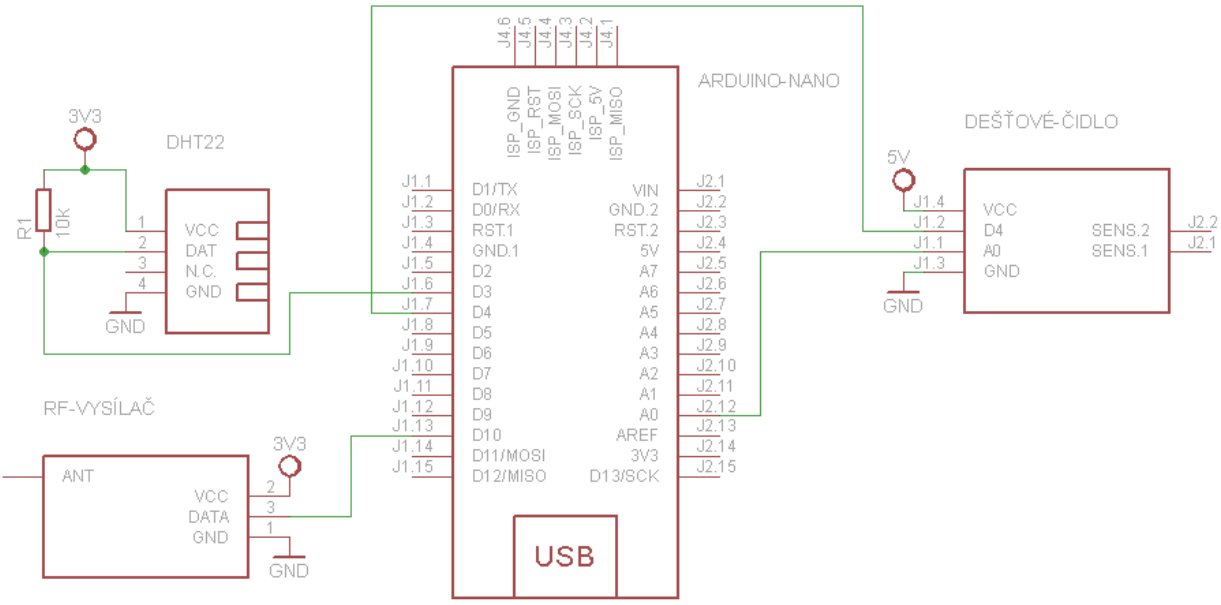
Obr. 31 – Vzhled webové aplikace po stisku drop-up tlačítka (zobrazení na PC)

7 NÁVRH A REALIZACE VENKOVNÍHO SYSTÉMU

Jako výpočetní jednotka v případě venkovní části je využito Arduino Nano pro jeho kompaktní rozměry, relativně malý proudový odběr a dobrou softwarovou podporu. Použité sensory DHT22, RF vysílač a dešťové čidlo jsou využity hlavně pro jejich dobrou dostupnost na trhu, nízkou cenu a jednoduchou implementaci. Pro napájení venkovní části bylo rozhodováno mezi použitím dvou Lithium-polymerových baterií o napětí 3,7 V a čtyřmi tužkovými AA bateriemi o napětí 1,5 V. Z důvodu rychlejší degradace životnosti u Li-Po baterií při vystavení vyšším teplotám (uložení v černém montážním boxu v letním období) byly zvoleny tužkové baterie. V průběhu práce došlo k zvýšení počtu použitých tužkových baterií na 6. To zapříčinil fakt, že pin *VIN*, který Arduino využívá jako způsob napájení, obsahuje napěťový regulátor, který vyžaduje použití napětí v rozsahu 6 V až 20 V. Využití pouze čtyř tužkových baterií s výsledným napětím 6 V tak není vhodné.

7.1 SCHÉMA ZAPOJENÍ

Schéma zapojení venkovní části bylo stejně jako v případě vnitřní části vytvořeno v programu EAGLE 7.0.0. Byla zde využita knihovna *diy-modules* importující nákres vývojové desky Arduino Nano.



Obr. 32 – Schéma zapojení venkovní části

Tabulka 6 – Zapojení vývodů snímače DHT22 (Arduino Nano)

DHT22	Arduino Nano
GND	GND
DAT	D3
NC	nezapojeno
VCC	3V3

Tabulka 7 – Zapojení vývodů RF vysílače

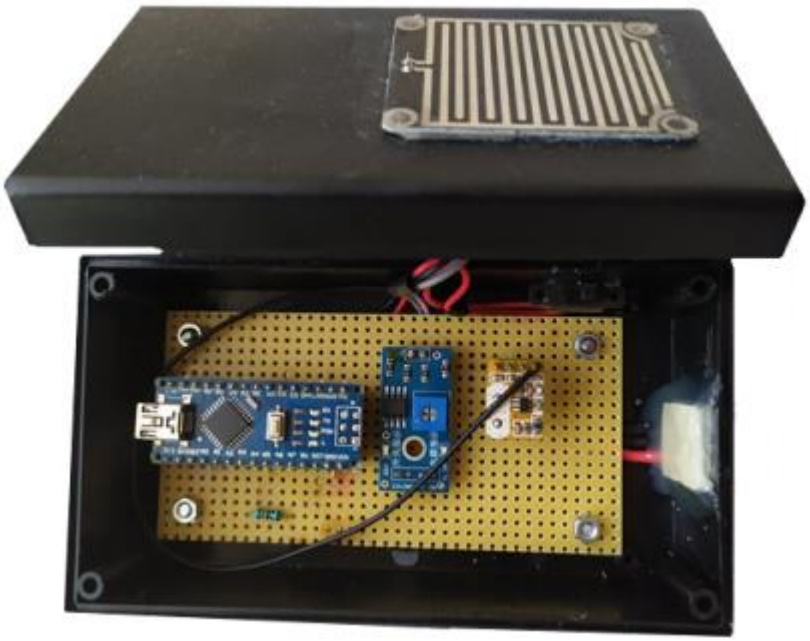
RF vysílač 433 MHz	Arduino Nano
GND	GND
DATA	D10
VCC	3V3
ANT	nezapojeno

Tabulka 8 – Zapojení vývodů dešťového čidla

Dešťové čidlo	Arduino Nano
GND	GND
D4	D4
A0	A0
VCC	5 V

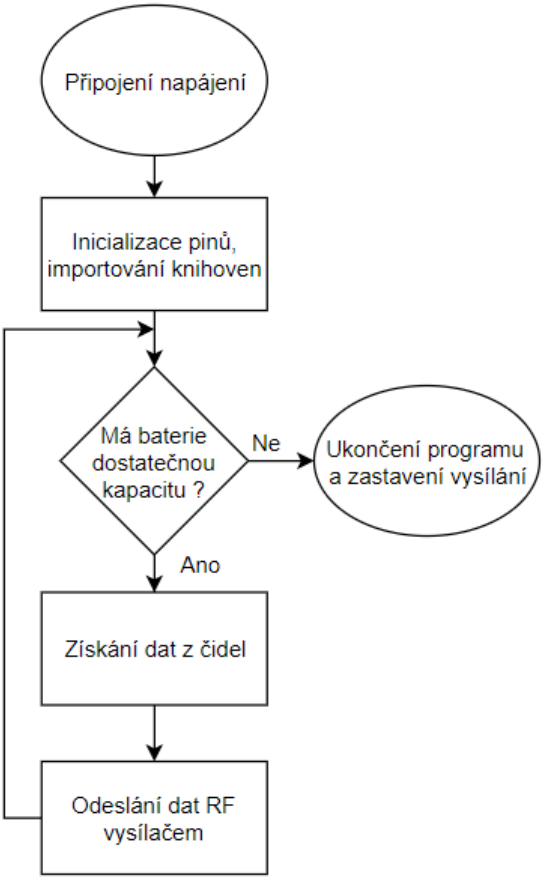
7.2 ZAPOUZDŘENÍ VENKOVNÍ ČÁSTI

Sensor DHT22 je zapuštěn do spodní strany boxu, aby nedocházelo ke zkreslování hodnot a zároveň nedošlo k poškození v případě deště. Sensor sloužící k indikaci deště je umístěn na přední odnímatelnou část boxu. Samotné Arduino Nano je spolu s radiofrekvenčním vysílačem a převodní deskou dešťového senzoru umístěno na univerzální prototypové desce plošných spojů.



Obr. 33 – Uložení venkovní části do montážního boxu

7.3 PRINCIP PROGRAMU



Obr. 34 – Princip programu vysílače

Pro implementaci programu venkovní části meteorologické stanice je využit programovací jazyk Wiring (přesněji se jedná o framework jazyka C++). Nejdříve je potřeba importovat knihovny pro komunikaci s čidlem DHT22 a RF vysílačem. Samotný program je rozdělen na 3 části: získání hodnot ze snímačů, kontrola stavu baterie a odeslání hodnot pomocí RF vysílače. Stěžejní je využití knihovny RCSwitch.h, která umožňuje efektivní využití RF vysílače: nastavení komunikačního protokolu, délky pulsu a počet opakování odeslání hodnot. Pro jednodušší komunikací se snímačem DHT22 Arduino využívá importovanou knihovnu DHT.h.

7.3.1 Kódování odesílaných dat

Snímač DHT22 má relativně dobrou softwarovou podporu pro mikrokontroléry z rodiny AVR, ale pro jistotu je kód ošetřen pomocí několika podmínek. Díky těmto podmínkám se odešlou pouze takové hodnoty teploty, které spadají do pracovního rozsahu čidla DHT22, což je podle oficiální dokumentace -40 °C až +85 °C. Tyto hodnoty byly následně ještě sníženy na polovinu s ohledem na podnebí v našich zeměpisných polohách.

Stejně tak je tomu i u odesílání hodnoty vlhkosti, kterou je možné odeslat jen v případě, že se její hodnota pohybuje v rozsahu 0 % až 100 %. Dále je nutné vyřešit problematiku odesílání a následného dekódování dat v přijímací části, protože vysílač dokáže přenášet pouze binární nebo dekadické hodnoty. Pokud by se měla odeslat proměnná ve formě textu, tak by došlo k odeslání pouze odpovídajících ASCII hodnot jednotlivých znaků. Tento fakt vede k dalšímu problému, kdy od sebe nelze jednoznačně rozlišit přijaté údaje (z pohledu programu). Tento problém je vyřešen pomocí jednoduchého kódování hodnot, které je znázorněno na následující ukázce kódu.

```
if(teplota <= 40 && teplota >= -20)
{
  vysilac.send(255,24);
  delay(100);
  vysilac.send(teplota,24);
  delay(900);
}

if(vlhkost <= 100 && vlhkost > 0)
{
  vysilac.send(254,24);
  delay(100);
  vysilac.send(vlhkost,24);
  delay(900);
}

if(prsi == 1000 || prsi == 2000)
{
  vysilac.send(253,24);
  delay(100);
  vysilac.send(prsi,24);
  delay(900);
}
```

Obr. 35 - Kódování odesílaných dat

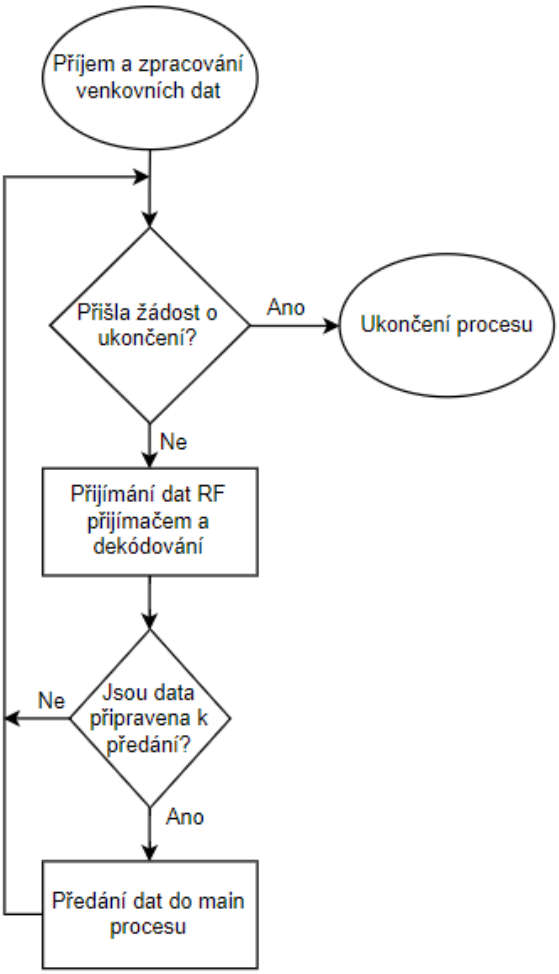
Hodnoty je možno rozlišit pomocí odeslání definované dekadické hodnoty před konkrétní hodnotou naměřenou snímačem. V první podmínce dochází k tomu, že pokud je

splněna teplotní mez a hodnota teploty je v platném rozsahu, tak se odešle hodnota 255 a následně hodnota teploty. Kód poté vyčká necelou sekundu před odesláním další dekadické hodnoty. V poslední podmínce se odešle informace o tom, jestli aktuálně prší, v případě že prší se odešle dekadická hodnota 1000, v opačném případě 2000. V přijímací části meteorologické stanice je poté část kódu, která následně zpracuje přijatá data a uloží je do odpovídajících proměnných.

7.4 PŘÍJEM VENKOVNÍCH DAT

Tato část programu je implementována ve vnitřní části stanice (v Raspberry Pi). Pro lepší vysvětlení a přehlednost je vložena do části práce určené pro realizaci venkovního softwaru.

V této části programu se využívá knihovna rpi_rf, která umožňuje zpracovávat data přijatá RF přijímačem. Důvodem, proč je tuto implementaci nutno udělat formou dalšího procesu je, že ačkoli oba procesy pracují v nekonečném cyklu, tak cyklus přijímače musí být mnohem rychlejší, aby se mohla data přijímat v reálném čase. Princip dekódování je stejný jako bylo popsáno v kapitole 6.3.1. Pokud je přijata dekadická hodnota 255, očekává se, že následující příchozí hodnota bude teplota, pokud je přijatá hodnota 254, tak následující hodnota bude vlhkost a pokud je přijatá hodnota 253, tak bude následovat informace o dešti. Hodnoty se následně nahrubo ošetří podmínkami. Pokud jsou zároveň splněny podmínky pro rozsah teploty, vlhkosti a jedna z očekávaných dekadických hodnot označující, jestli prší, dojde k poslání dat rourou do hlavního procesu.

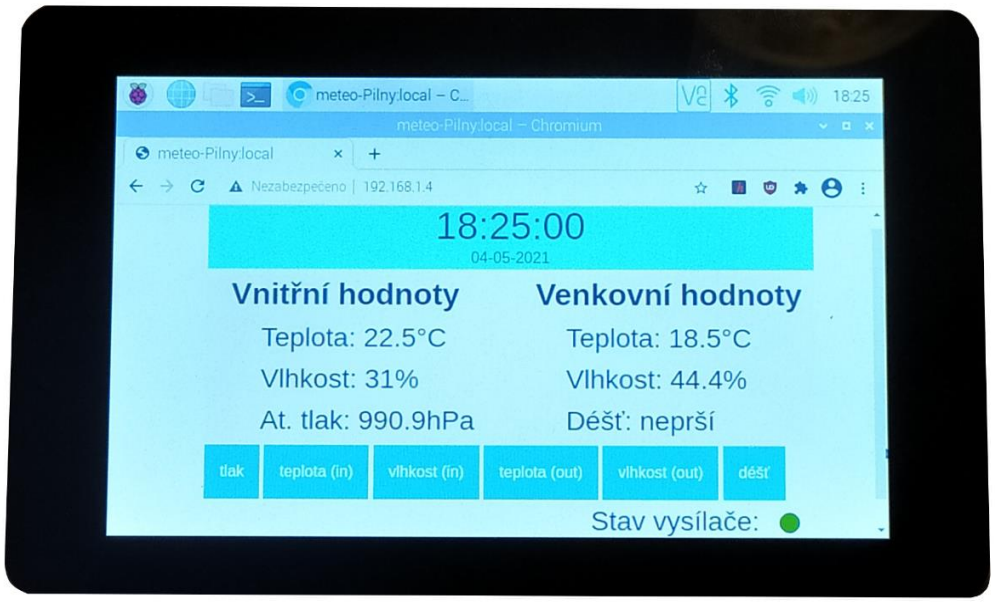


Obr. 36 – Vývojový diagram procesu pro příjem venkovních dat (v Raspberry Pi)

8 ZOBRAZENÍ METEOROLOGICKÝCH DAT

Ukázka naměřených meteorologických dat pokrývá časový horizont 18. 4. – 1. 5. 2021. Veškerá data jsou uložena v databázi, ale vykreslit lze pouze ty hodnoty, které byly naměřeny během posledního týdne, v tomto případě vztažené k datu 1. 5. 2021. Grafy obsahují anglické zkratky označující dny a měsíce, protože jsou vytvořeny pomocí knihovny matplotlib.

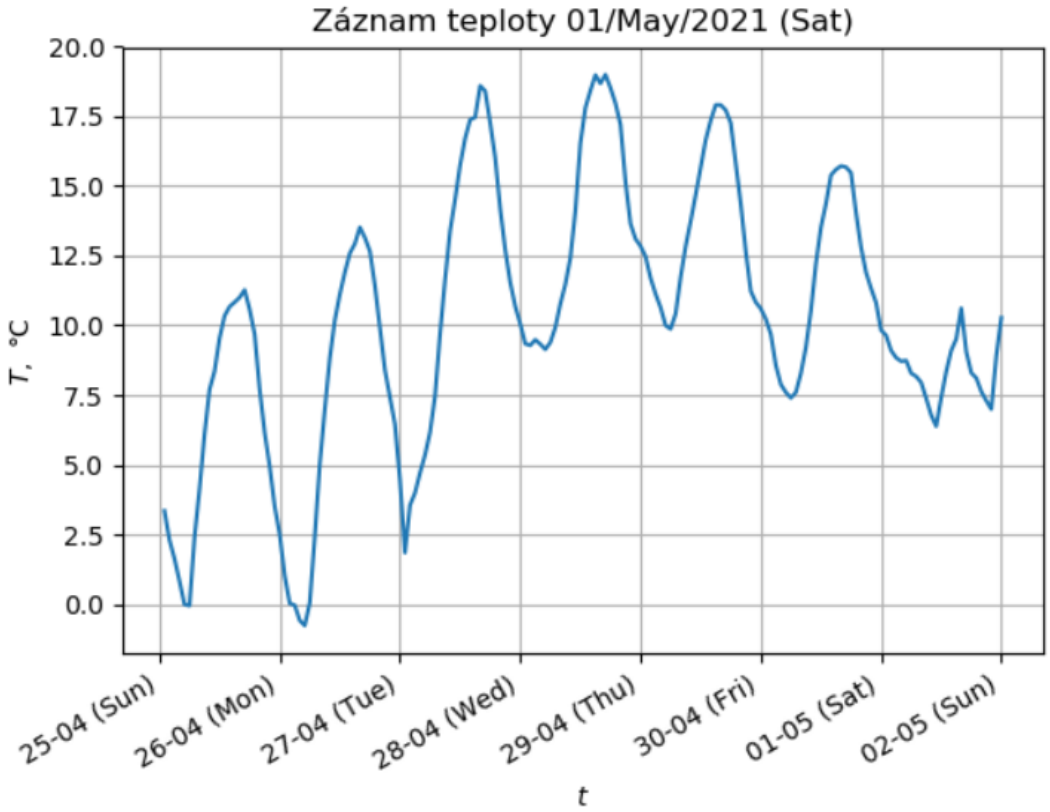
Na obrázku níže je zobrazeno webové rozhraní zobrazené na LCD obrazovce. Při porovnání s Obr. 29, lze vidět, že zobrazení na LCD displeji se liší od zobrazení na PC v šířce sloupců a umístění tlačítek, která jsou zde po celé spodní hraně tabulky.



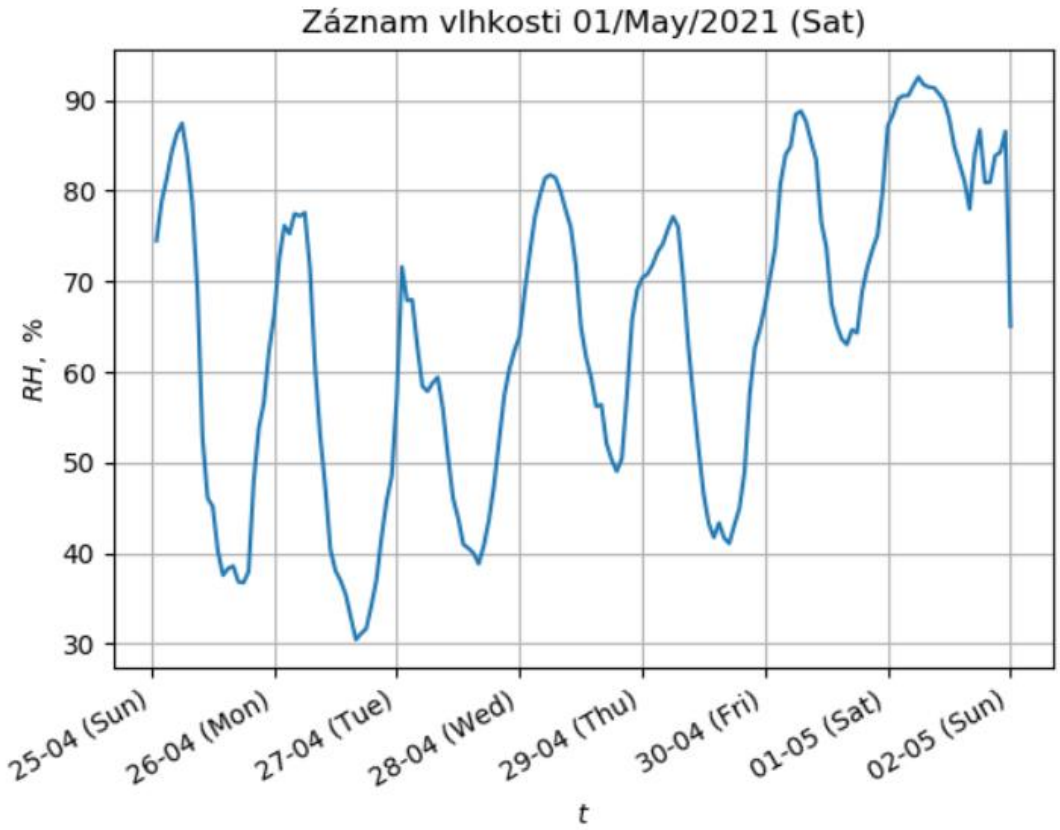
Obr. 37 – Zobrazení webové aplikace na LCD

8.1 GRAFY

Na následujících obrázcích je zobrazen vývoj venkovní teploty a vlhkosti vzduchu v týdnu od 25. 4. do 1. 5. 2021. Obrázky jsou pro praktickou ukázku vytvořeny jako snímky obrazovky PC verze webové stránky. Na displeji meteorologické stanice je jejich zobrazení totožné. V prvním grafu je přehledně vidět, jak se mění venkovní teplota v rámci jednotlivých dní. Propady hodnot v zobrazeném průběhu odpovídají nočním poklesům teplot. Ve stejných časových úsecích lze poté pozorovat nárůst vlhkosti.

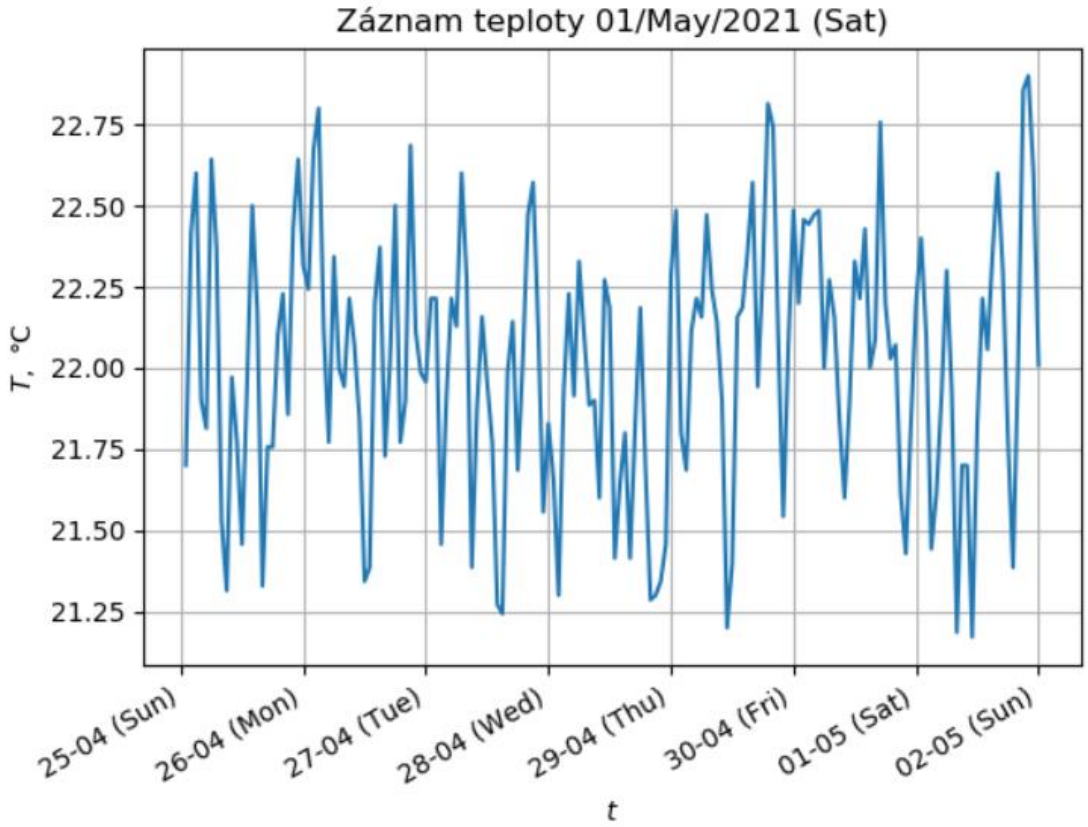


Obr. 38 – Graf průběhu venkovní teploty (25. 4. – 1. 5.)

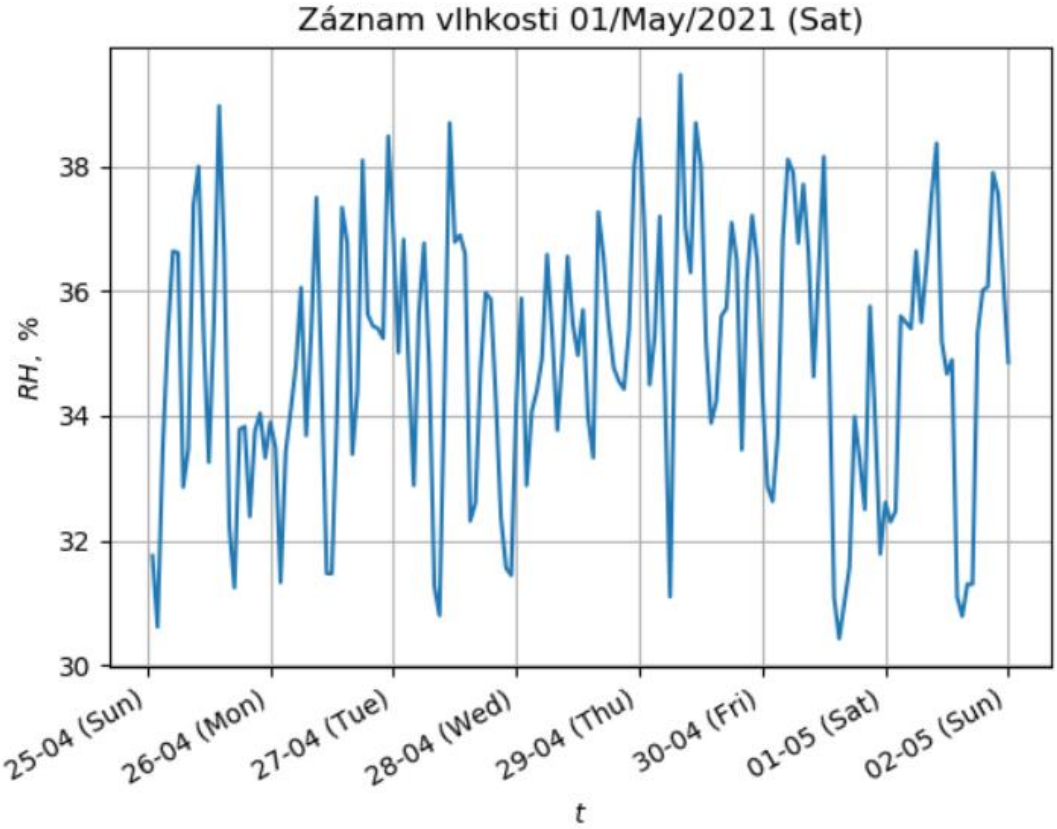


Obr. 39 – Graf průběhu venkovní vlhkosti (25. 4. – 1. 5.)

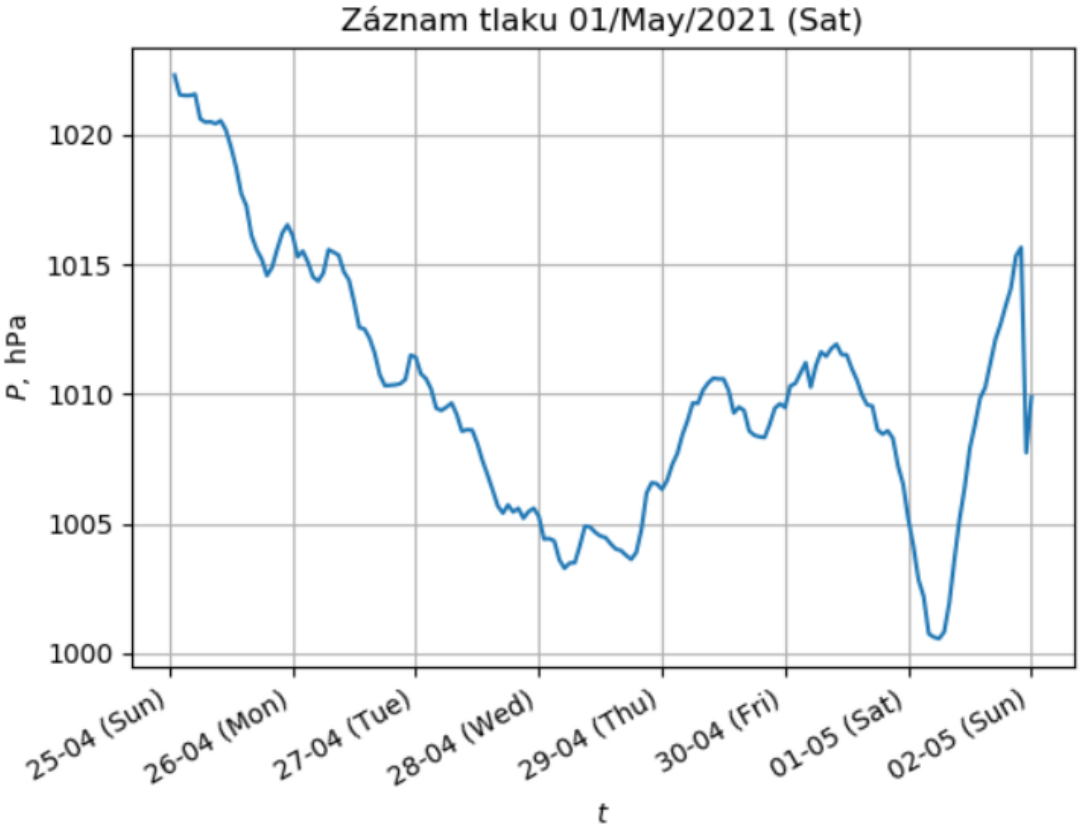
Na dalších třech obrázcích je zobrazen vývoj teploty a vlhkosti v místnosti a průběh barometrického tlaku. V případě vnitřního měření teploty mají hodnoty menší rozptyl než naměřená venkovní data. Důvodem je, že meteostanice byla umístěna v pokoji s topením, které bylo po celou dobu zapnuto a nedocházelo k většímu větrání po čas měření. Z tohoto důvodu nemá smysl uvádět graf hodnot z měření kratšího než týden, jelikož v kratších intervalech nejsou změny prakticky pozorovatelné. Grafy vnitřní vlhkosti a teploty mají velice podobný průběh. Rozdíl je ve vyšším rozsahu hodnot relativní vlhkosti, který je 31 % až 39 %. Ze zaznamenaných hodnot barometrického tlaku lze pozorovat klesající tendenci tlaku od 25. 4. do 28. 4. Podstatný je zde propad zaznamenaný k 1. 5., který odpovídá velice silnému dešti, který následoval.



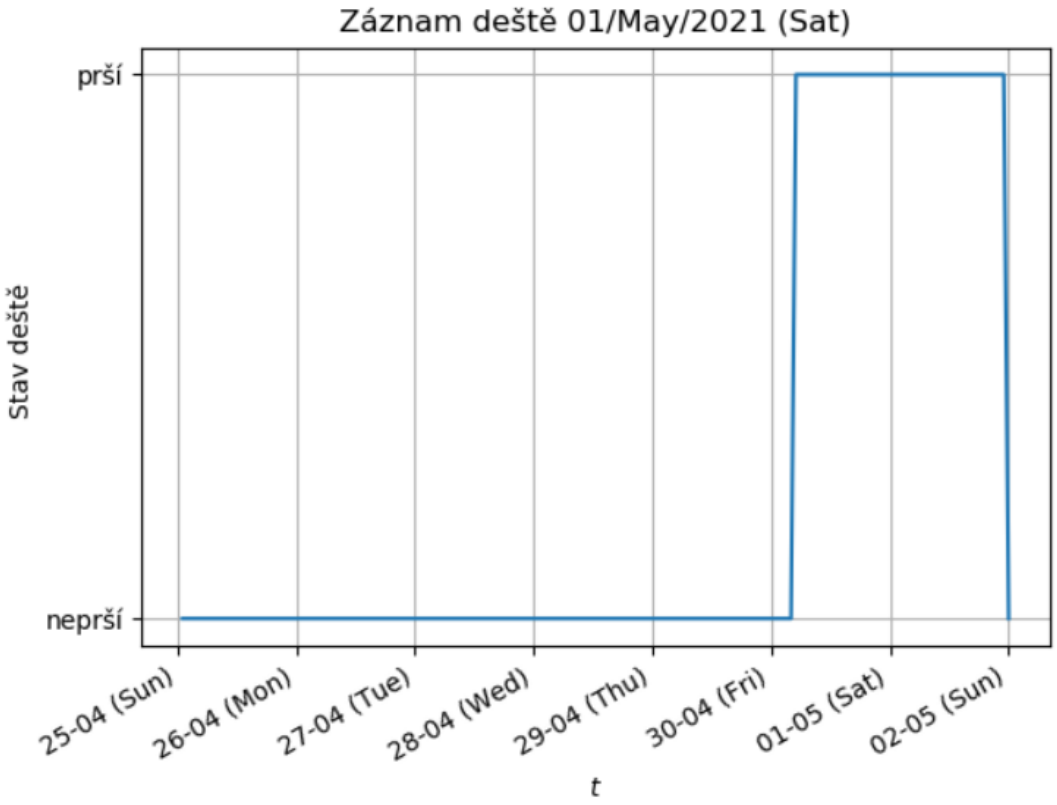
Obr. 40 – Graf průběhu vnitřní teploty (25. 4. – 1. 5.)



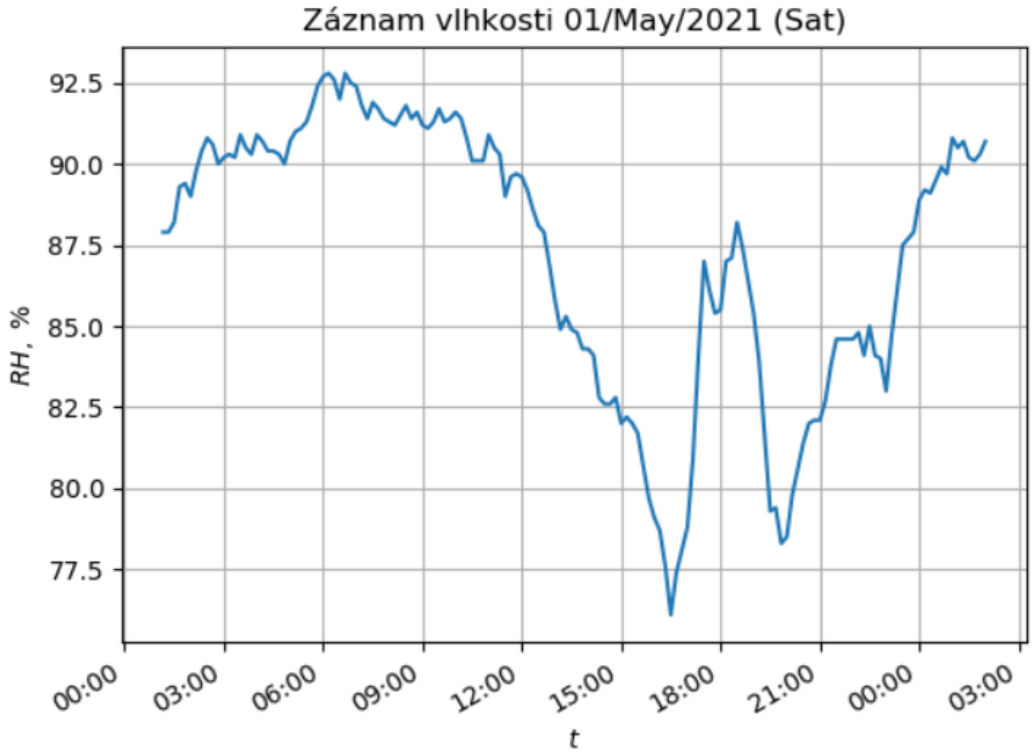
Obr. 42 – Graf průběhu vnitřní vlhkosti (25. 4. – 1. 5.)



Obr. 41 – Graf vývoje barometrického tlaku (25. 4. – 1. 5.)



Obr. 44 – Graf vývoje deště (25. 4. – 1. 5.)



Obr. 43 – Graf průběhu venkovní vlhkosti (1. 5.)

9 ZÁVĚR

Cílem práce bylo vytvoření zařízení pro měření a uchovávání meteorologických dat. Nejdříve byl v práci teoreticky popsán význam získávání a zpracovávání meteorologických dat a rozbor měřených veličin a použitých snímačů, pomocí kterých se data měří.

V praktické části se práce zaměřila na návrh a realizaci vnitřního a venkovního systému. Byly zde popsány důvody ke zvolení patřičných řídicích jednotek, přidružených periferií a použitého softwaru.

Venkovní systém byl zapouzdřen do montážního boxu s otevíracím víkem. Vzhledem k očekávanému působení vnějších podmínek (zejména deště), byly otvory, do kterých byla zapuštěna čidla, zaizolovány pomocí tavné pistole. Vysílač byl poprvé otestován za velmi silného deště a bylo zjištěno pár zásadních problémů v jeho návrhu. Slabinou otvíratelného montážního boxu je existence miniaturní mezery mezi oběma částmi, kterou se dovnitř dostala voda. Dešťové čidlo (vodivá podložka) také začalo korodovat již po první noci, kdy bylo vystaveno silnému dešti. S korozním procesem se počítalo, předpokládalo se ale, že k výraznější korozi dojde v rámci několika týdnů až měsíců. Po vysušení celé venkovní části byla vodivá podložka ošetřena ochranným lakem proti korozi a dovnitř boxu vložen silikagel za účelem odstranění přebytečné vlhkosti. V poslední řadě byla mezera vzniklá mezi oběma díly přelepena izolační páskou. Po této úpravě už nedošlo k dalším komplikacím.

Program pro příjem a zpracování dat byl vyvíjen v době, kdy venkovní část ještě nebyla schopna provozu ve venkovních podmínkách, aniž by nehrozilo její poškození. Hodnoty potřebné pro vytvoření programu zpracovávající údaje z databáze a zobrazujícího grafy byly nasimulovány pomocí MS Excell a importovány do SQLite databáze v podobě csv souborů. V momentě, kdy byla venkovní část schopna plného provozu, byla data v databázi nahrazena skutečnými naměřenými hodnotami. Bylo nutné vyřešit situaci, kdy i s periodou snímání dat 10 min bylo v databázi pro týdenní interval příliš mnoho hodnot a zobrazovaný průběh byl z toho důvodu nepřehledný a nenesl žádnou použitelnou informaci. Nápravy bylo docíleno vytvořením algoritmu pro průměrování hodnot.

V průběhu práce došlo k několika poznatkům, které by mohly sloužit pro potenciální zlepšení a rozšíření funkcí do budoucna. V prvním případě se jedná o snížení proudového odběru vysílací části využitím samotného mikrokontroléru ATmega328p, nikoliv celé Arduino Nano desky. Dále by bylo možné snížit spotřebu energie uspáním mikrokontroléru na definovaný čas a odesíláním hodnot jen po krátkou dobu. Důvodem, proč tento nápad nebyl realizován, byl obslužný program navržený tak, aby se proces pro přijímání a ukládání dat do

databáze uspal po každé iteraci na 10 minut. Vzniklo by tak riziko, že odeslaná data by mohla dorazit v moment, kdy by přijímač už nebyl aktivní.

Dalším budoucím vylepším by pak mohlo být zapouzdření vnitřní části včetně displeje a Raspberry Pi do montážního boxu vytisknutého pomocí 3D tiskárny. Výhoda by v tomto případě byla převážně designového charakteru.

V poslední řadě by mohlo být užitečné rozšířit webovou aplikaci o možnost stažení všech hodnot z databáze do souboru s uživatelsky definovaným formátem a umožnit vykreslení grafu pro jakýkoliv časový úsek z naměřených dat.

LITERATURA A ZDROJE

Atmel. *ATmega328P*. [online]. [cit. 2021-5-1]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

HUGHES, J. 2019. *BCM2711 – Raspberry Pi Dokumentation*. [online]. [cit. 2021-5-2]. Dostupné z: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/README.md

HEIKKI, L. 2018. *Pt100 temperature sensor – useful things to know*. [online]. [cit. 2021-5-2]. Dostupné z: https://blog.beamex.com/pt100-temperature-sensor

Bosch Sensortec. 2015. *BMP280 Digital Pressure Sensor.* [online]. [cit. 2021-5-2]. Dostupné z: https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf

The Editors of Encyclopaedia Britannica. 1998. *Hygrometer meteorological instrument*. [online]. [cit. 2021-5-2]. Dostupné z: https://www.britannica.com/science/hygrometer

Last Minute Engineers. 2018. *How 433MHz RF Tx-Rx Modules Work & Interface with Arduino.* [online]. [cit. 2021-5-1]. Dostupné z: https://lastminuteengineers.com/433mhz-rf-wireless-arduino-tutorial/

GIRES,A. 2018. *How Do We Measure Rainfall?* [online]. [cit. 2021-5-2]. Dostupné z: https://kids.frontiersin.org/article/10.3389/frym.2018.00038

FRUHLINGER, J. 2020. *What is an IP address? And what is your IP address?* [online]. [cit. 2021-5-2]. Dostupné z: https://www.networkworld.com/article/3588315/what-is-an-ip-address-and-what-is-your-ip-address.html

KOTHARI, V. 2018. *Internal working of Python.* [online]. [cit. 2021-4-30]. Dostupné z: https://www.geeksforgeeks.org/internal-working-of-python/

SMOLKA, V. 2015. *Měření a zaznamenávání tlaku vzduchu.* [online]. [cit. 2021-5-2]. Dostupné z: www.in-pocasi.cz/clanky/teorie/tlak-vzduchu-4.8.2015

SMOLKA, V. 2015. *Vlhkost vzduchu a její charakteristiky.* [online]. [cit. 2021-5-2]. Dostupné z: https://www.in-pocasi.cz/clanky/teorie/vlhkost-vzduchu/

DE BRUIJN, L. 2019. *How-to form submissions with Flask and AJAX*. [online]. [cit. 2021-4-30]. Dostupné z: https://javascript.plainenglish.io/how-to-form-submissions-with-flask-and-ajax-dfde9891c620

KISTLER. *Piezoresistive pressure sensor*. [online]. [cit. 2021-5-1]. Dostupné z: https://www.kistler.com/en/glossary/term/piezoresistive-pressure-sensor/

Last Minute ENGINEERS. 2018. *How DH11 DHT22 sensors work & interface with arduino*. [online]. [cit. 2021-4-30]. Dostupné z: https://lastminuteengineers.com/dht11-dht22-arduino-tutorial/

Last Minute ENGINEERS. 2018. *How Rain Sensor Works and Interface it with Arduino*. [online]. [cit. 2021-4-30]. Dostupné z: https://lastminuteengineers.com/dht11-dht22-arduino-tutorial/

Linux.com. 2018. *What is Linux?* [online]. [cit. 2021-5-2]. Dostupné z: https://www.linux.com/what-is-linux/

LIU, T. *Digital-output relative humidity & temperature sensor/module DHT22*. [online]. [cit. 2021-5-1]. Dostupné z: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

Mipi alliance. 2015. *MIPI Display Seriál Interface (MISI DSI)*. [online]. [cit. 2021-5-1]. Dostupné z: https://www.mipi.org/specifications/dsi

MAHESHWARI, M. 2017. *Understanding SSH workflow*. [online]. [cit. 2021-5-1]. Dostupné z: https://medium.com/@Magical_Mudit/understanding-ssh-workflow-66a0e8d4bf65

NASH, J. 2019. *Flask HTTP methods*. [online]. [cit. 2021-5-2]. Dostupné z: https://pythonise.com/series/learning-flask/flask-http-methods

BOUDREAU, D.; MCDANIEL, M. 2012. *Meteorology | National Geographic Society*. [online]. [cit. 2021-5-2]. Dostupné z: https://www.nationalgeographic.org/encyclopedia/meteorology/

Pallets. 2010. *Foreword – Flask documentation (1.1.x).* [online]. [cit. 2021-5-2]. Dostupné z: https://flask.palletsprojects.com/en/1.1.x/foreword/

Python Institute. *What is Python?* [online]. [cit. 2021-4-30]. Dostupné z: https://pythoninstitute.org/what-is-python/

Raspberry Pi Foundation. 2012. *HISTORY: The start of raspbian.* [online]. [cit. 2021-5-2]. Dostupné z: https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=4256.

Raspberry Pi Foundation. *Raspberry Pi Touch Display.* [online]. [cit. 2021-4-30]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-touch-display/

Raspberry Pi Foundation. 2019. *Raspberry Pi 4 Model B*. [online]. [cit. 2021-5-1]. Dostupné z: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi_DATA_2711_1p0_preliminary.pdf

EAMES, A. *DSI-connector_1500*. [online]. [cit. 2021-5-2]. Dostupné z: https://raspi.tv/dsi-connector_1500

ANDERSON, J. 2019. *An Intro to Threading in Python.* [online]. [cit. 2021-4-30]. Dostupné z: https://realpython.com/intro-to-python-threading/#what-is-a-thread

KHILLAR, S. 2019. Difference Between Absolute and Relative Humidity. [online]. [cit. 2021-5-2]. Dostupné z: http://www.differencebetween.net/science/nature/difference-between-absolute-and-relative-humidity/

ROUPHAEL, T. 2009. *Modulation Method.* [online]. [cit. 2021-5-2]. Dostupné z: https://www.sciencedirect.com/topics/computer-science/modulation-method

JAGTAP, M. 2019. *What is SQLite? – SphereGen*. [online]. [cit. 2021-5-2]. Dostupné z: https://www.spheregen.com/sqlite/

SQLite. *Write-Ahead Logging*. [online]. [cit. 2021-5-2]. Dostupné z: https://sqlite.org/wal.html

YLONEN, T. 1996. *SSH Protocol – Secure Remote Login and File Transfer*. [online]. [cit. 2021-5-1]. Dostupné z: https://www.ssh.com/academy/ssh/protocol

PIETERS, M. 2012. *What is the difference between multiprocessing and subprocess?* [online]. [cit. 2021-4-30]. Dostupné z: https://stackoverflow.com/questions/13606867/what-is-the-difference-between-multiprocessing-and-subprocess

COX, E.; BERGSTRESSER, M. 2020. *What is Temperature? – Definition & Measurement*. [online]. [cit. 2021-5-2]. Dostupné z: https://study.com/academy/lesson/what-is-temperature-definition-lesson-quiz.html

VALDEZ, J.; BECKER, J. 2015. *Undestanding the I2C Bus*. [online]. [cit. 2021-5-1]. Dostupné z: https://www.ti.com/lit/an/slva704/slva704.pdf

DRAKE, V. 2020. SQLite in production with WAL. [online]. [cit. 2021-5-2]. Dostupné z: https://victoria.dev/blog/sqlite-in-production-with-wal/

WAVESHARE. *DHT22 Temperature-Humidity Sensor*. [online]. [cit. 2021-5-2]. Dostupné z: https://www.waveshare.com/wiki/DHT22_Temperature-Humidity_Sensor

PŘÍLOHY

A – CD

Příloha A

Příloha k bakalářské práci

ZAŘÍZENÍ PRO SBĚR A ZPRACOVÁNÍ METEOROLOGICKÝCH DAT

Tomáš Pilný

CD

OBSAH

1. Text bakalářské práce ve formátu PDF
2. Zdrojový kód venkovní části
3. Zdrojový kód vnitřní části