Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Venkovní meteostanice s využitím platformy Arduino

Jiří Pavelka

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jiří Pavelka

Systémové inženýrství a informatika
Informatika

Název práce

Venkovní meteostanice s využitím platformy Arduino

Název anglicky

Outdoor weather station with using Arduino platform

Cíle práce

Cílem bakalářské práce je vytvoření venkovní meteostanice se zobrazením dat na webové stránce.

Dílčí cíle práce jsou:
-Vytvoření senzorických jednotek.
-Vytvoření hlavní jednotky.
-Vytvoření webové aplikace pro zobrazení naměřených dat.

Metodika

Prostřednictvím platformy Arduino a vybraných senzorů bude realizována senzorická část zařízení, která bude síťově propojena s hlavní částí. Pomocí jednodeskového počítače od firmy Espressif, dalších komponent a implementace vhodného zdrojového kódu bude navržena hlavní část pro zpracování, posílání a zálohování meteorologických dat. Pro ukládání meteorologických dat bude proveden výběr vhodného síťového úložiště v rámci internetu, a realizován software pro jeho obsluhu. Následně budou vytvořeny jednoduché webové stránky s vhodnými funkcemi pro zobrazení aktuálních a historických dat z měření. Vzhledem k vlastnostem použitých jednodeskových počítačů bude využit režim snížené spotřeby. Závěr práce zhodnotí teoretické poznatky a praktickou část.

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

Meteostanice, Arduino, ESP32, RS485, MySql

Doporučené zdroje informací

Kitchen, Z. V. Průvodce světem Arduina. Nakladatelství Martin Stříž, Bučovice (2017). ISBN 978-80-87106-93-8

Martin Malý. HRADLA, VOLTY, JEDNOČIPY Úvod do bastlení. CZ.NIC, z. s. p. o. (2017). ISBN 978-80-88168-23-2

Miroslav Virius. Programování v C++: od základů k profesionálnímu použití. Grada (2018). ISBN 978-80-271-0502-1

Pinker, J. Mikroprocesory a mikropočítače. BEN – technická literatura (2011). ISBN 80-7300-110-1

Vladimír Váňa. ARM pro začátečníky. BEN – technická literatura (2009). ISBN 978-80-7300-246-6

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 9. 3. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 9. 3. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 10. 03. 2020

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Venkovní meteostanice s využitím platformy Arduino" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 12. 3. 2020

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce Ing. Markovi Píckovi, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce.

Venkovní meteostanice s využitím platformy Arduino

Abstrakt

Tato práce se zabývá možným návrhem, sestavením a naprogramováním zařízení meteorologické stanice. Zařízení využívá mikrokontrolery a periférie platformy Arduino pro měření teploty, vlhkosti, tlaku, rychlosti větru, směru větru a množství srážek. Cílem je navržení a vytvoření funkčního prototypu meteostanice, která přenáší naměřená data na internetové úložiště, odkud jsou následně načítána a prezentována jako webové stránky. Jsou vysvětleny principy fungování jednotlivých komponent modulů použitých při tvorbě zařízení s důrazem na mikrokontrolery a jejich komunikační rozhraní.

Měření rychlosti větru a množství srážek je realizováno vyhodnocováním pulsů anemometru a srážkoměru. Směr větru je reprezentován velikostí napětí. V bezprostřední blízkosti čidel se nachází MCU AtMega328p, který zpracovává signály a následně získaná data odesílá po RS485. Tyto části tvoří první senzorovou jednotku. Druhá pro měření teploty, vlhkosti a tlaku obsahuje stejný MCU, který odesílá data stejným způsobem jako první jednotka, ale data z čidel jsou přenášena přes rozhraní I^2C. Data z obou jednotek jsou přenesena na vzdálenost až 33 m do hlavní jednotky. Ta je zpracovává a odesílá na internetové úložiště za pomocí dvoujádrového mikrokontroleru ESP32 přes síť WiFi. Stránky webového rozhraní načítají a zobrazují uložená data.

Vytvořené zařízení umožňuje měřit všechny uvedené veličiny a zároveň disponuje vlastnostmi jako jsou záloha napájení, nebo záloha dat, což zvyšuje spolehlivost. Veškeré uvedené funkce a vlastnosti byly otestovány čtyřměsíčním spolehlivým provozem bez jediné ztráty záznamu.

Klíčová slova

meteostanice, ESP32, AtMega328p, I2C, SPI, MySQL, PHP, javaScript, HTML, RS485

Outdoor weather station with using Arduino platform

Abstract

This work deals with possible design, assembly and programming of weather station equipment. The device uses Arduino platform microcontrollers and peripherals to measure temperature, humidity, pressure, wind speed, wind direction and rainfall. The aim is to design and create a functional prototype of the weather station, which transmits the measured data to the Internet repository, from where it is subsequently downloaded and presented as a website. Principles of operation of individual components of modules used in the creation of equipment with emphasis on microcontrollers and their communication interface are explained.

Measurement of wind speed and amount of precipitation is realized by evaluation of pulses of anemometer and rain gauge. The wind direction is represented by the magnitude of the voltage. The AtMega328p MCU is located in the immediate vicinity of the sensors, which processes the signals and sends the acquired data via the RS485. These parts forming the first sensor unit. The second for temperature, humidity and pressure measurement contains the same MCU, which sends data in the same way as the first unit, but the sensor data is transmitted via the I2C interface. Data from both units is transferred up to 33 m to the main unit. It processes and sends them to the Internet storage using the dual-core microcontroller ESP32 over WiFi network. Web interface pages retrieve and display stored data.

The device makes it possible to measure all of the above quantities and at the same time it has features such as power backup or data backup, which increases reliability. All of these features have been tested for four months of reliable operation without any loss of record.

Keywords

weather station, ESP32, AtMega328p, I2C, SPI, MySQL, PHP, javaScript, HTML, RS485

Obsah

Seznam obrázků

Seznam použitých zkratek

ADC	Analog-to-Digital Converter
BOD	Brown Out Detection
DAC	Digital-to-Analog Converter
EEC	Error Checking and Correcting
EEPROM	Electrically Erasable Programmable ROM
FTP	File Transfer Protocol
GPIO	General Purpose Input Output
HTML	Hypertext markup language
I^2C	Inter-Integrated Circuit
I2S	Inter-IC Sound
JSON	JavaScript Object Notation
LSB	Least Significant Bit
MCU	Microcontroler
MSB	Most Significant Bit
NTP	Network Time Protocol
PHP	Personal Home Page
RAM	Random-Access Memory
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
RS232	Recommended Standard 232
RS485	Recommended Standard 485
RTC	Real Time Clock
SD	Secure Digital Memory Card
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SRAM	Static Random Access Memory
UART	Universal Asynchronous Reciver Transmiter
UPS	Uninterruptible Power Supply
URL	Uniform Resource Locator
USB	Universal Serial Bus

1 Úvod

Meteostanice je zařízení, které slouží k měření meteorologických veličin, jako jsou např. teplota, vlhkost, tlak, rychlost větru, směr větru, úhrn dešťových srážek, nebo sluneční svit. Většina dnešních moderních domácností je vybavena meteostanicí, umožňující měřit alespoň teplotu, vlhkost a barometrický tlak. Lépe vybavené meteostanice mohou obsahovat další senzory, jako např. senzor rychlosti větru, směru větru, srážkoměr a senzor slunečního svitu. Pomocí přepočtu některých veličin je možné získávat další data, jako pocitovou teplotu, nebo rosný bod. Venkovní meteostanice může obsahovat všechny uvedené senzory, které jsou součástí senzorových jednotek umístěných ve venkovním prostředí s hlavní jednotkou umístěnou ve vnitřním prostoru. Vnitřní senzory jsou zpravidla zabudované v hlavní jednotce, jejíž součástí může být displej pro zobrazení naměřených dat. Senzorové jednotky jsou spojeny s hlavní jednotkou pevným, nebo bezdrátovým spojením. Naměřená data ze senzorů jsou aktualizována v pravidelných intervalech a poté zobrazena na displej jednotky meteostanice, nebo mohou být předávána na internetové úložiště a zobrazena ve formě webových stránek.

Tato práce se zabývá vytvořením zařízení venkovní meteostanice na platformě Arduino, které měří teplotu, vlhkost, tlak, rychlost větru, směr větru, úhrn dešťových srážek a zobrazuje tyto data na webových stránkách. Práce se dělí na teoretickou a praktickou část.

Teoretická část práce se zaměřuje na popis jednotlivých komunikačních rozhraní, která jsou použita při návrhu a realizaci zařízení. Dále popisuje hardware jednodeskových počítačů platformy Arduino a hardware modulů periférií využitých pro meteostanici. Poslední část popisuje software platformy a software na straně serveru pro ukládání a zobrazování naměřených dat pro webové stránky.

Praktická část popisuje jednotlivá řešení použitá při výrobě zařízení meteostanice jako jsou optimální umístění, konstrukce senzorových jednotek včetně zpracování naměřených meteorologických dat ze senzorů. Následuje konstrukce hlavní jednotky, zařízení pevného úložiště, použití samostatné jednotky času RTC s řešením pro úsporu a zálohu energie. Kapitola sériová komunikace řeší vzájemné propojení jednotek meteostanice včetně protokolu komunikace. Poslední část je věnována návrhu zdrojového kódu zařízení, výměně dat s databází a popisu řešení webového rozhraní.

2 CÍL PRÁCE A METODIKA

2.1 Cíl práce

Cílem této bakalářské práce je navrhnout a vytvořit venkovní meteostanici s využitím platformy Arduino. Naměřená data jako jsou teplota, vlhkost, tlak, rychlost větru, směr větru a množství srážek by měla být předávána na internetové úložiště a zobrazována na webových stránkách. Poslední naměřené meteorologické hodnoty budou aktualizovány v intervalu jedné minuty a v intervalu 30 minut ukládány pro dlouhodobé statistiky zobrazované ve formě grafů. Z důvodu optimálního umístění jednotlivých senzorů jsou nutné dvě senzorové jednotky, první na otevřeném prostranství pro měření rychlosti větru, směru větru a srážek a druhá ve stinném místě pro měření teploty, vlhkosti a tlaku. Pro případ výpadku síťového napájení musí meteostanice obsahovat záložní zdroj napájení, který vydrží poskytovat energii minimálně 3 dny. Při ztrátě spojení mezi meteostanicí a internetovým uložištěm by mělo být navrženo a vytvořeno dočasné úložiště naměřených dat.

2.2 Metodika

Pro realizaci meteostanice budou použity běžně dostupné komponenty ve formě hotových modulů určených pro platformu Arduino. Všechny vybrané moduly pro meteostanici budou vybrány s ohledem na dostupnost dokumentace a knihoven poskytujících potřebné funkce. Hlavní část bude vzhledem k požadavkům na konektivitu a výkon tvořena modulem s mikrokontrolerem ESP32. Propojení mezi jednotlivými moduly a částmi meteostanice bude vytvořeno pomocí sériových komunikačních rozhraní. Samozřejmě je nutné brát ohled na nejrůznější elektrotechnické standarty, které jsou dány výrobcem použitého hardwaru a podle nich provést výběr vhodného rozhraní, které ovlivňuje vlastnosti jako jsou rychlost, spolehlivost a vzdálenost.

Na všechny použité moduly existují knihovny napsané v jazyce c++, které poskytují jednotlivé funkce pomocí zápisu a čtení registrů čipu modulu. Navíc je k dispozici velké množství informací ať už v podobě příkladů hotových řešení, nebo diskusních fór zabývajících se touto problematikou. V rámci vytváření kódu pro takové zařízení je nutné začínat od menšího rozsahu kódu a před přidáním dalších funkcionalit nejprve vše řádně otestovat. Pokud vznikne chyba, lze ji většinou tímto postupem rychle nalézt.

3 PLATFORMA ARDUINO

„Vývoj prvního Arduina započal v roce 2005, když se lidé z italského Interaction Design Institute ve městě Ivrea rozhodli vytvořit jednoduchý a levný vývojový set pro studenty, kteří si nechtěli pořizovat, v té době rozšířené a drahé desky BASIC Stamp. Mezi studenty se Arduino uchytilo, a tak se tvůrci rozhodli poskytnout ho celému světu. A to nejenom prodejem vlastních desek, ale i sdílením všech schémat a návodů (jedná se o Open Source projekt). Programová část Arduina byla založena na Processing, což je programovací jazyk s vlastním editorem, určený k výuce programování. V dnešní době se prodalo již několik stotisíc desek Arduino. Důkazem, že tato platforma není mrtvá, může být i to, že nedávno byl ohlášen vývoj nové a výkonné desky Arduino Galileo, která vzniká ve spolupráci s Intelem. Za osm let vývoje již vzniklo spoustu různých typů Arduina. Jelikož se jedná o opensource projekt, vznikalo společně s hlavní linií projektu i spoustu dalších, neoficiálních typů, takzvaných klonů.“ [1]

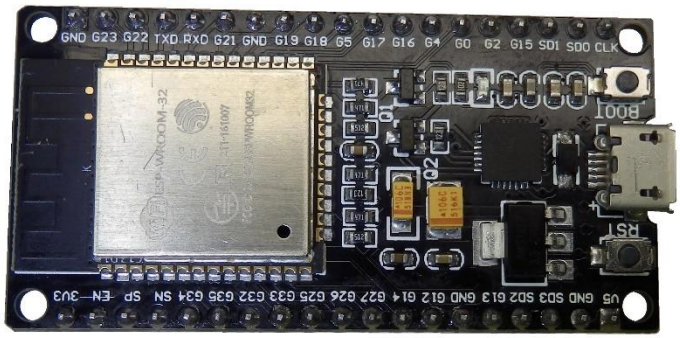
4 HARDWARE

Základem každého zařízení pro platformu Arduino je vývojová deska. K té se připojují jednotlivé periférie přes nejrůznější typy rozhraní jako I^2C, SPI, UART a další. Periférie mohou být vstupní, výstupní, nebo obojího charakteru, jako např. externí paměti. Na trhu je nepřeberné množství různých typů senzorů, displejů, řízení motorů, bezdrátových zařízení, vývojových desek a dalších. Vývojové desky a periférie lze zakoupit v originálním provedení, nebo jako klony jiných výrobců, které nemají takovou kvalitu jako originál, ale jsou mnohonásobně levnější.

Hlavní částí každé vývojové desky je mikrokontroler. Všechny potřebné komponenty pro jeho činnost jsou integrovány uvnitř jednoho malého křemíkového čipu, takže oproti klasickému procesoru nevyžaduje připojení externích obvodů jako např. pamětí, nebo komunikačních rozhraní. Proto se nazývá jednočipový počítač. Nachází velké uplatnění v nejrůznějších systémech měření, regulace a automatizace. Tyto systémy jsou většinou určené pro specifický účel, což umožňuje jejich jednodušší konstrukci a tím i vysokou spolehlivost s příznivou cenou.

4.1 **ESP32**

Pro hlavní jednotku meteostanice byla vybrána deska s mikrokontrolerem ESP32 (viz obrázek 1), od firmy Espressif. ESP32 je dvoujádrový, 32bitový procesor, na frekvenci až 320MHz, s instrukční sadou typu RISC. Konektivitu zajišťuje bezdrátové připojení WiFi v pásmu 2.4GHz s rychlostí 150 Mbps, případně Bluetooth 4.0 se zpětnou kompatibilitou pro starší rozhraní. Oproti předchůdci s čipem ESP8266 je jedno jádro vyhrazeno pouze pro provoz sítě WiFi, a tak je druhé jádro plně k dispozici programátorovi. Počet GPIO pinů byl zvýšen na 48, přičemž většina z nich je multiplexována, takže je lze využít jako univerzální. Nalézá se zde několik typů rozhraní jako jsou UART, SPI, I^2C, zvukové rozhraní I2S, samozřejmostí jsou i DAC/ADC převodníky. Piny, které jsou konfigurované jako výstupní, mohou být použity pro řízení koncových obvodů akčních členů využívajících pulzně-šířkové-modulace, tzv. PWM. V případě nečinnosti je možný přechod do některého z režimu spánku a ušetřit energii, což je vhodné zejména při napájení z baterií, nebo akumulátorů. Kapacita paměti ROM pro uložení programu je 448 kB a operační paměť SRAM má velikost 520 kB. Pro uchováni dat v režimu spánku slouží paměť 16 kB v jednotce RTC. [2]

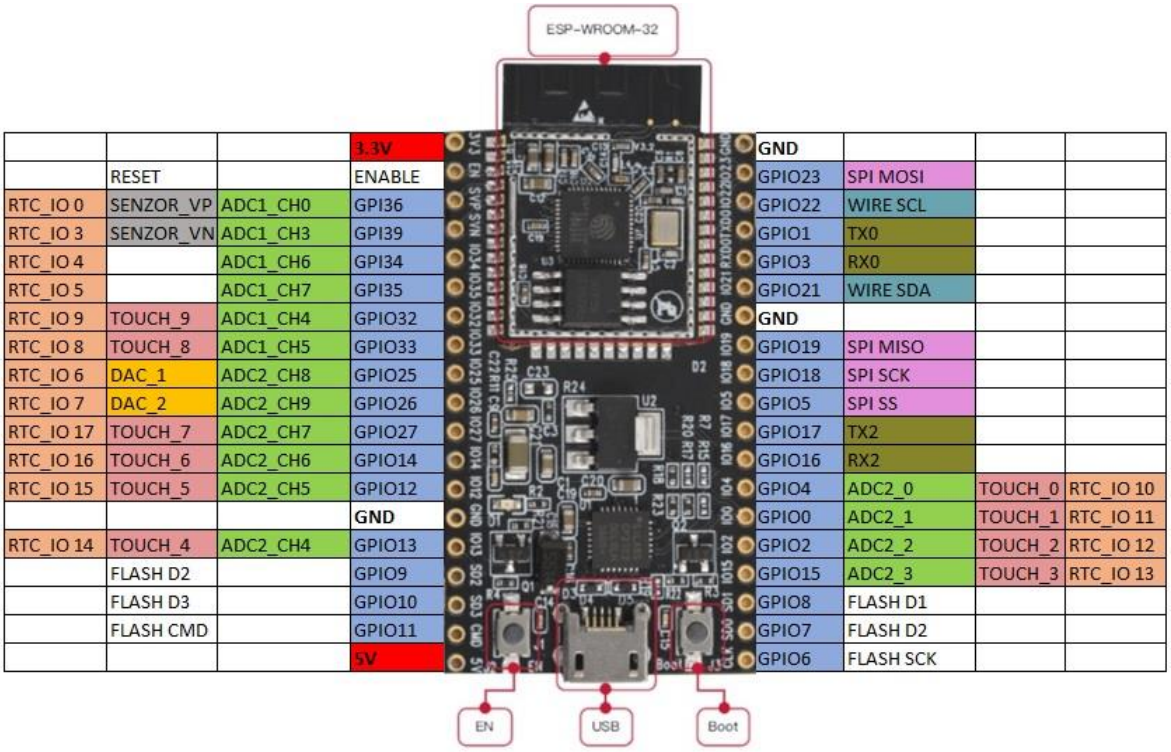


Obrázek 1: *Vývojová deska ESP32 od Espressif*

Zdroj: vlastní zpracování

4.2 Rozhraní GPIO

Hlavní modul pro meteostanici bude obsahovat mikrokontroler ESP32, který má 48 pinů. Některé z těchto pinů jsou vyhrazeny pro specifické funkce, ty ostatní jsou označeny jako vstupně-výstupní piny pro všeobecný účel, zkráceně GPIO a jsou vyvedeny na piny modulu. Tyto GPIO se dají pomocí funkce multiplexování použit jako digitální vstup, výstup, převodník DAC, převodník ADC, dotykový senzor, pin rozhraní SPI, pin rozhraní I^2C, pin rozhraní UART, napěťový výstup Hallova senzoru a další. Možnosti konfigurace jednotlivých pinů jsou na obrázku 2. Změna funkce pinu se provádí úpravou odpovídajících registrů z prostředí programu. Piny 34, 35, 36 a 39 jsou pouze vstupní. Piny 6, 7, 8, 9, 10 a 11 jsou určené pro komunikaci s vnitřní pamětí flash a měli by zůstat volné. To samé platí pro piny TX0 a RX0, které jsou připojeny k rozhraní pro komunikaci s počítačem. Pokud je využíváno rozhraní SPI, nebo I^2C, nelze tyto piny používat k jinému účelu. V případě, že je používána WiFi, piny ADC2 nelze použít. Digitální piny, konfigurované jako vstupní, umožňují softwarové vřazení rezistoru (pull-down) k nulovému potenciálu,



Obrázek 2: *Rozmístění jednotlivých pinů*

Zdroj: **[3]**

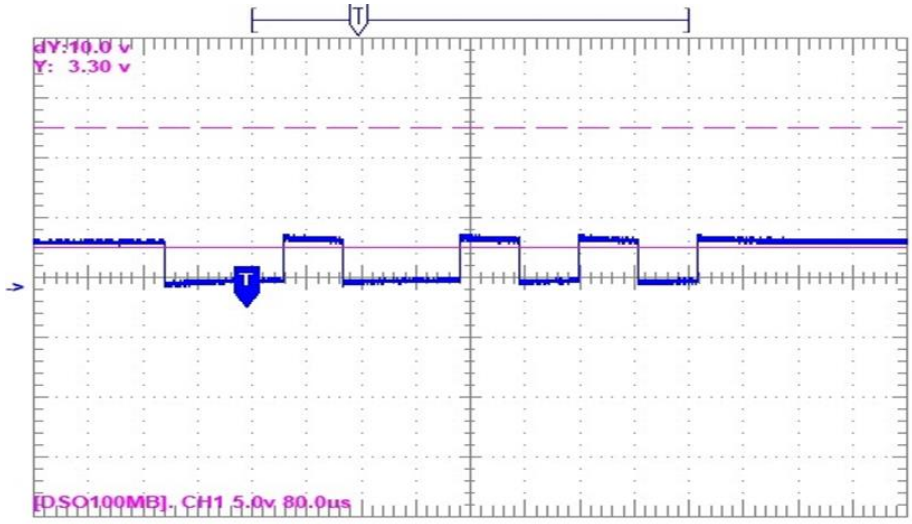
nebo k potenciálu napájecího napětí (pull-up) a lze je využít pro zpracování externího přerušení. U digitálních pinů, které jsou nastaveny jako výstupní, lze využít funkce PWM pro řízení výkonu koncových stupňů akčních členů. Mikrokontroler ESP32 má maximální napětí pinu 3.3 V a zatížitelnost 12 mA. Připojování zařízení s jinou napěťovou logikou vyžaduje použití převodníku napěťových úrovní (viz kapitola 4.9). [3] [2]

4.3 Sběrnice UART

Rozhraní je určené pro přenos dat mezi dvěma body se stejnou napěťovou logikou na vzdálenost několika desítek centimetrů. Při rozdílné napěťové logice je nutné použít převodník napěťových úrovní (viz kapitola 4.9). Komunikace probíhá pomocí asynchronního přenosu jednotlivých znaků v plně duplexním režimu. Znaky jsou přenášeny pomocí dvoustavového signálu, který představuje jednotlivé datové bity. Počet datových bitů pro každý znak je volitelný v rozmezí 7–9 bitů, standartně je nastaveno 8 datových bitů. Dále lze zvolit parametry jako počet stop bitů, rychlost přenosu a kontrolu paritou, ta ale nebývá příliš spolehlivá vzhledem k možnosti výskytu sudého počtu chyb. Pro spolehlivější přenos dat lze kontrolovat parametry jako jsou chyba bitu, chyba rámce, nebo ztráta dat. Přidáním převodníku k tomuto rozhraní je možné používat standarty jako jsou RS232, RS485, USB a další. [4] [5]

Aby mohl být přenos uskutečněn je důležité, aby přijímač začal přijímat ve stejné chvíli, kdy vysílač začne odesílat. To je provedeno synchronizačním pulsem na začátku přenosu pomocí tzv. „start-bitu“. Data jsou na jedné straně odesílána po vodiči označeném jako Tx metodou LSB a na druhé straně jsou po vodiči Rx přijímána a ukládána do přijímacího bufferu. Zapojení je tedy provedeno v konfiguraci Rx1 -> Tx2 a Tx1 -> Rx2. Dále je nezbytný 3.vodič s referenčním nulovým napětím pro určení logického stavu na lince. [5]

U ESP32 definuje logickou 0 napětí v rozsahu 0–0,8 V a logickou 1 napětí 2,0-3.3 V. U Arduino Nano definuje logickou 0 napětí v rozsahu 0-1,5 V a logickou 1 napětí 3-5 V. V klidovém stavu udržuje vysílač stav 1. Ve chvíli zahájení přenosu je vyslán „start bit“ s logickou hodnotou 0 a poté již následuje samotný přenos znaku, který je ukončen jedním, nebo dvěma stavy logické 1 – tzv. „stop bit“. Vysílač je tedy uveden zpět do klidového stavu



Obrázek 3: *Průběh signálu při posílání znaku*

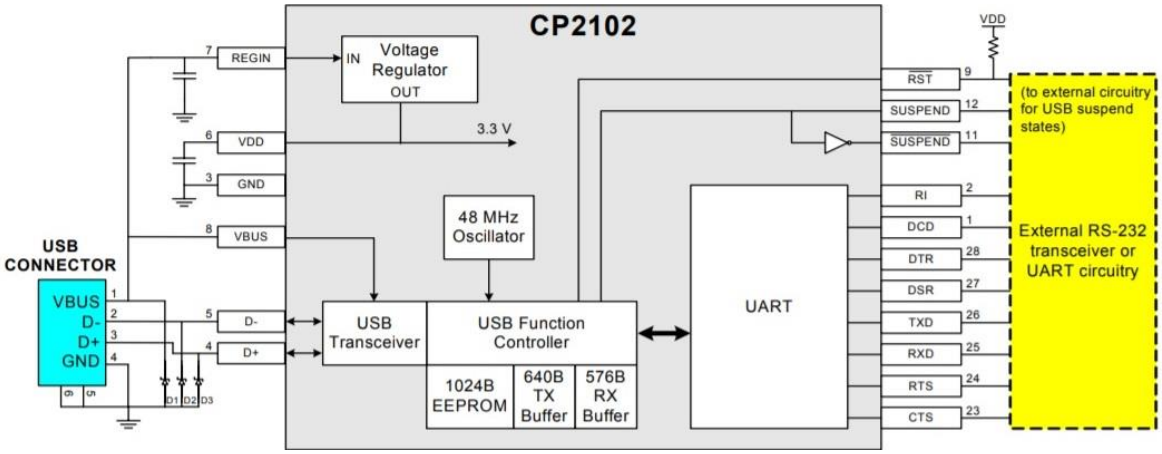
Zdroj: vlastní zpracování

do doby, než bude posílán další znak. Průběh signálu na lince při odesílání znaku „R“ (metodou LSB, bez kontroly parity, binárně 01001010) je zachycen na obrázku 3.

Možnost využít síťového zapojení je omezena pouze na konfiguraci jeden vysílač – více přijímačů, kdy je využit 9. datový bit pro identifikaci přijímače na lince, přičemž tuto funkci umožňují jenom některé mikrokontrolery. [4]

4.4 Převodník CP2102

Jedná se o převodník pro komunikaci mezi rozhraním UART a USB. Využívá se při komunikaci mikrokontroleru s počítačem, ať už pro zápis, nebo čtení, ale také pro nahrání programu do paměti ROM. Vyrábí se ve formě malého mikročipu, obsahujícího vlastní regulátor napětí, oscilační obvod, přijímací a vysílací buffer s celkovou velikostí paměti 1024 kB a řadičem USB2.0 (viz obrázek 4). Může být i součástí některých vývojových desek. Lze ho připojit k logice 3.3 V i 5 V, přičemž jsou jeho výstupy dimenzovány na maximální proud 100 mA. [6]



Obrázek 4: *Vnitřní schéma obvodu CP2102*

Zdroj: **[6]**

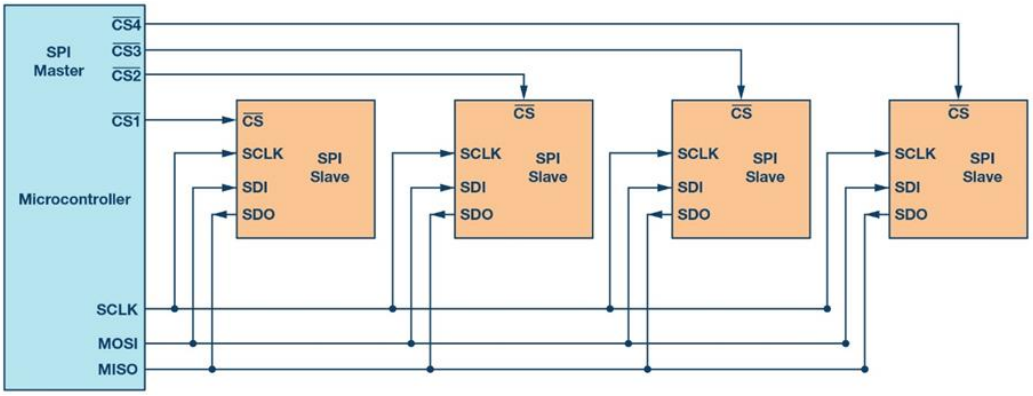
4.5 Sběrnice SPI

Další sériové rozhraní se označuje zkratkou SPI a je obvykle nedílnou součástí mikrokontrolerů. Bylo definováno firmou Motorola na konci 80.let a stalo se standardem, který se používá dodnes, někdy se také označuje jako „four-wire serial bus“. Bylo vyvinuto pro rychlou komunikaci mezi jednotlivými obvody na velmi malou vzdálenost, obvykle v rámci jednoho plošného spoje. Mezi tyto obvody patří sériové paměti RAM, EEPROM, FLASH, paměťové karty SD, ovladače displejů, nebo různé senzory, také se dost často používá pro komunikaci mezi mikrokontrolery. Přenos dat probíhá synchronně, v plně duplexním režimu, při rychlosti až několik megabajtů za sekundu. Rozhraní vždy obsahuje pouze jeden hlavní řídící obvod typu master a k němu je připojen jeden, nebo více podřízených obvodů typu slave (viz obrázek 5). [4] [5]

Komunikace je řízena z masteru výběrovým signálem, který určuje, pro které zařízení slave je linka vyhrazena. Tento výběr je uskutečněn přepnutím pinu CS (Chip Select) na obvodu slave do stavu log. 0. Přepnutí je realizováno prostřednictvím výstupního pinu obvodu master. Pro adresování každého zařízení slave je tedy nutný jeden výstupní pin mikrokontroleru. Výhodou je rychlejší přenos, protože jsou přenášena pouze data, nikoliv adresa. Jakmile je uskutečněn výběr, začne po vodiči SCLK (Clock synchronization) od obvodu master přicházet hodinový signál a zároveň je zahájen přenos jednotlivých

datových bitů po vodiči MOSI (Master Out Slave In) z jednotky master do jednotky slave. Datové bity jsou odesílány metodou LSB a jsou postupně zpracovány posuvnými registry, které jsou obsaženy jak na výstupu, tak na vstupu obvodů master i slave. Komunikace ve směru od zařízení slave probíhá po vodiči MISO (Master In Slave Out). Pro tento typ komunikace je tak zapotřebí čtyř vodičů vč. výběrového. Pokud nejsou vyžadována data ze zařízení slave, lze vynechat vodič MISO. Pomocí programu je možné konfigurovat, jestli budou data čtena na vzestupné, nebo sestupné hraně signálu a logickou hodnotu, při které je vodič SCLK v klidovém stavu. [4] [5]

V moderních zařízeních sériových pamětí je instalován dvojnásobný, nebo čtyřnásobný počet pinů MOSI/MISO, a tak lze přenášet dva, nebo čtyři datové bity v jednom hodinovém cyklu. Tato provedení se označují jako Dual SPI, nebo Quad SPI. [5]



Obrázek 5: *Zapojení rozhraní SPI*

Zdroj: **[31]**

4.6 SD karta

Jedno z paměťových zařízení, které je možné připojit ke sběrnici SPI. Karta je dostupná ve standartní velikosti, nebo ve zmenšené variantě microSD. Pro zálohu naměřených dat bude použita karta microSD s paměťovým čipem NAND flash (viz obrázek 6). Tato technologie byla představena společností Toshiba v roce 1987 jako další generace

paměti EEPROM a používá se dodnes. Jde o nevolatilní paměť, což znamená, že data jsou uchovány i po odpojení zdroje elektrického proudu. [7]

Informace je ukládána v jednotlivých binárních buňkách, které jsou tvořeny unipolárními tranzistory s plovoucími hradly, izolovaných od okolí vrstvou oxidu křemíku. Při ukládání informace do binární buňky se vlivem zachytávání přiváděných elektronů do plovoucího hradla, mění elektrický náboj tranzistoru. Tento náboj poté reprezentuje uloženou hodnotu binární informace. Nevýhodou je, že při opakovaném používání stejné buňky dochází k opotřebování vrstvy oxidu, což má za následek omezený počet zápisů. U technologie SLC (Single-level-cell), která ukládá jeden bit informace na buňku, to může být až 100 tisíc cyklů. U technologií, které používají např. hustotu 2 bity na buňku, označenou jako MLC (Multi-level-cell), nebo 3 bity na buňku, označenou jako TLC(Triple-level-cell), je počet cyklů mnohonásobně nižší. [8]

Paměť je organizována do bloků o velikosti 256 kB. Zápis a mazání dat probíhá po celých blocích. Každý blok tvoří 64 stránek s velikostí 4 kB, přičemž jsou tyto stránky nejmenší nedělitelnou částí paměti. Jednotlivé stránky obsahují dalších 64 bajtů pomocných informací, jako jsou opravný kód chyb ECC, nebo údaje o počtu mazání, což je využito pro řízení paměti, které tak může hlídat rovnoměrnost opotřebení a v případě potřeby přesměrovat zápis na méně používanou stránku. Mezi výhody patří cena, ta se v době psaní této práce pohybovala okolo 12 Kč za 1 GB. Dále rychlost čtení a zápisu až 95/90 MBps a nízká spotřeba energie, průměrně 35 mA. [7] [9]

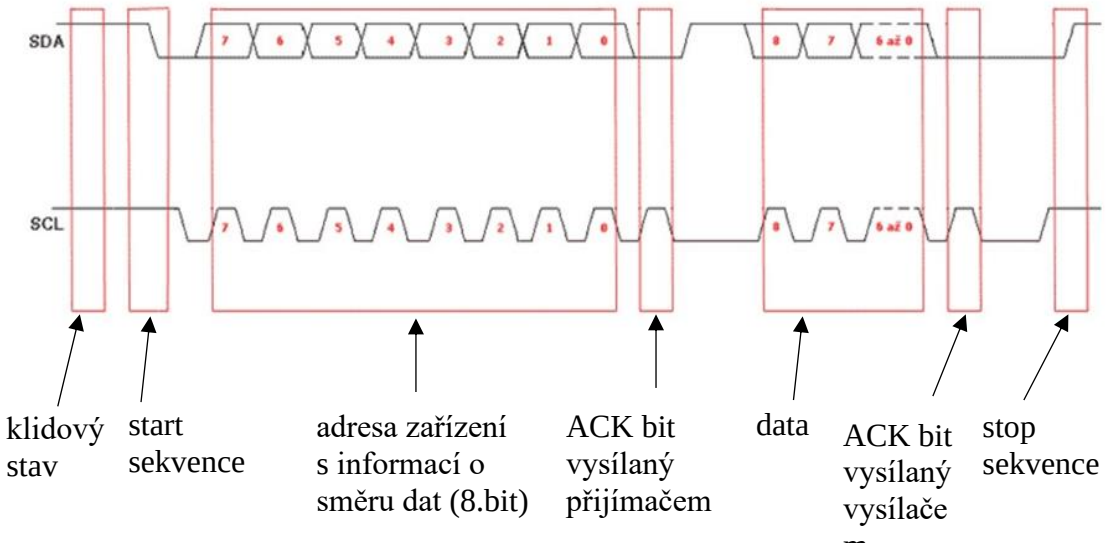
Obrázek 6: *Paměťová karta microSD se slotem*

Zdroj: vlastní zpracování

4.7 Sběrnice I²C

Další typ sériového rozhraní, které se využívá ke komunikaci s periferiemi, jako jsou např. hodiny reálného času RTC, senzory teploty, vlhkosti a tlaku. Vyvinuto firmou Philips pod označením I²C, což je zároveň ochranná známka této společnosti, a proto se u ostatních výrobců používá jiné označení, jako např. TWI (Two-Wire-Interface). Rozhraní je určeno pro rychlou vzájemnou komunikaci jednotlivých obvodů, na velmi malou vzdálenost, řádově desítky cm.

Přenos dat probíhá synchronně pomocí dvou vodičů, SDA pro přenos dat a SCL pro hodinový signál. Na rozdíl od SPI může být na lince zapojeno více než jedno zařízení master, potom hovoříme o sběrnici typu multimaster a jedno, nebo více zařízení slave. Každé zařízení má sedmibitovou adresu, která musí být v rámci jedné linky jedinečná, aby nedocházelo ke kolizi. Většinou jsou adresy pevně dány výrobcem, ale u některých zařízení je možnost tuto adresu měnit, především tam, kde se počítá s vyšším počtem zařízení stejného druhu na jedné sběrnici, jako jsou paměti EEPROM, expandéry, převodníky atd. Rychlost přenosu dat se pohybuje od 100 kbit/s až po 3.4 Mbit/s. Data jsou přenášena po osmi bitech, od nejvyššího (MSB). Průběh komunikace na fyzické vrstvě je znázorněn na obrázku 7. [5]



Obrázek 7: *Průběh logických úrovní při přenosu dat*

Zdroj: **[33]**

Vodiče SDA a SCL jsou pull-up rezistory udržovány v klidovém stavu, na logické 1. Pomocí principu otevřeného kolektoru jsou realizovány změny z log 1 na log 0. Zahájení přenosu (start sekvence), nebo dalšího znaku je provedeno změnou úrovně linky SDA z logické 1 na logickou 0, zatímco je linka SCL v logické 1. K přenosu jednotlivých bitů dochází při načítání logického stavu linky SDA v okamžiku náběžné hrany SCL. Změna na lince SDA musí probíhat pouze v okamžiku, kdy je SCL na logické 0, v opačném případě by to znamenalo signalizaci start, nebo stop sekvence. Za start sekvenci je odeslána sedmibitová adresa s osmým bitem, který signalizuje, jestli bude probíhat čtení, nebo zápis. Poté master čeká na potvrzení (ACK), jestli je zařízení s danou adresou připraveno přijímat data, což je signalizováno devátým bitem. Pokud master příjme tento bit, jako logickou 0, provede sekvenci start a začne odesílat data. [5] [10]

4.8 Jednotka času RTC

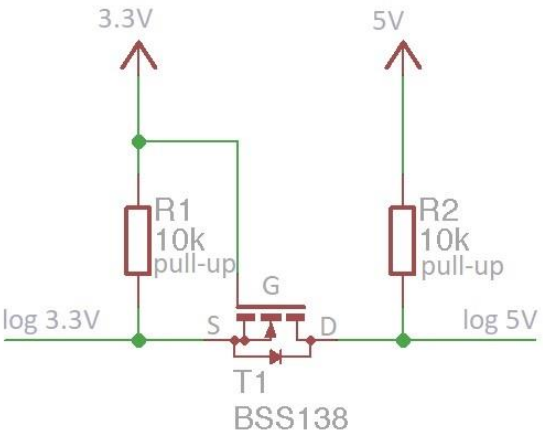
Jednotka Real-Time-Clock (RTC) se využívá pro práci s reálným časem a je určena pro výše uvedené rozhraní I^2C. Pro meteostanici bude zvolen modul s velmi přesným obvodem DS3231, který má řízený oscilátor s kompenzací kolísání frekvence krystalu vlivem teploty. Obsahuje funkci pro korekci přestupného roku, dva alarmy s funkcí přerušení, přepínání mezi 12 a 24hodinovým režimem a kalendář. Výpočty jednotlivých údajů času jsou prováděny samotným obvodem DS3231 a ukládány do registrů času a datumu. Mikrokontroler pomocí rozhraní I^2C a adres registrů RTC získá požadované údaje. Tyto údaje jsou sekundy, minuty, hodiny, dny v týdnu, dny v měsíci, měsíce a roky. Nastavení RTC na aktuální datum a čas probíhá obráceně, data jsou přes I^2C zapsána mikrokontrolerem do registrů času a datumu. Dále jsou tyto registry aktualizovány již samotným zařízením RTC. Mikrokontroler tak může po obnově ztráty napájení či restartu načíst aktuální čas i bez připojení k internetu, samozřejmě za předpokladu, že jednotka RTC byla již nastavena a má instalovanou záložní baterii. Na trhu je možné sehnat hotový modul s tímto obvodem, 32 kB EEPROM pamětí, paticí na záložní baterii pro případ výpadku napájení a pull-up rezistory pro komunikaci přes I^2C za cenu 40 Kč (viz obrázek 8). [11]

Obrázek 8: *Modul RTC*

Zdroj: vlastní zpracování

4.9 Převodník napěťových úrovní

Zařízení (angl. level-shifter) pro případ, kdy je nutné propojit dva, nebo více modulů s rozdílnou napěťovou logikou. V případě platformy Arduino jde většinou o napětí 3.3 V a 5 V. Tento převodník lze sehnat ve formě hotového modulu, který může být jednosměrný, nebo obousměrný a obsahuje obvykle čtyři, nebo osm linek. Jednosměrný převodník využívá při převodu vyššího napětí na nižší napěťový dělič, který je tvořen dvěma rezistory. V opačném případě se používá obousměrný převodník, který obsahuje n-mosfet tranzistor a dva pull-up rezistory (viz obrázek 9). Ve stavu logické 1 je rozdíl napětí mezi hradlem G a emitorem S nulový, tranzistor je uzavřený a na logických linkách je velikost napětí udržována pull-up rezistory. Po připojení linky s nižším logickým napětím k zemi, tzn. k logické 0, je vlivem rozdílu napětí mezi hradlem G a emitorem S, otevřen tranzistor a uzemněna linka na straně s vyšším logickým napětím. V případě uzemnění linky ze strany vyššího logického napětí, je pomocí diody v tranzistoru uzemněna i linka s nižším logickým napětím. [12]



Obrázek 9: *Schéma zapojení oboustranného převodníku logických úrovní*

Zdroj **[12]**

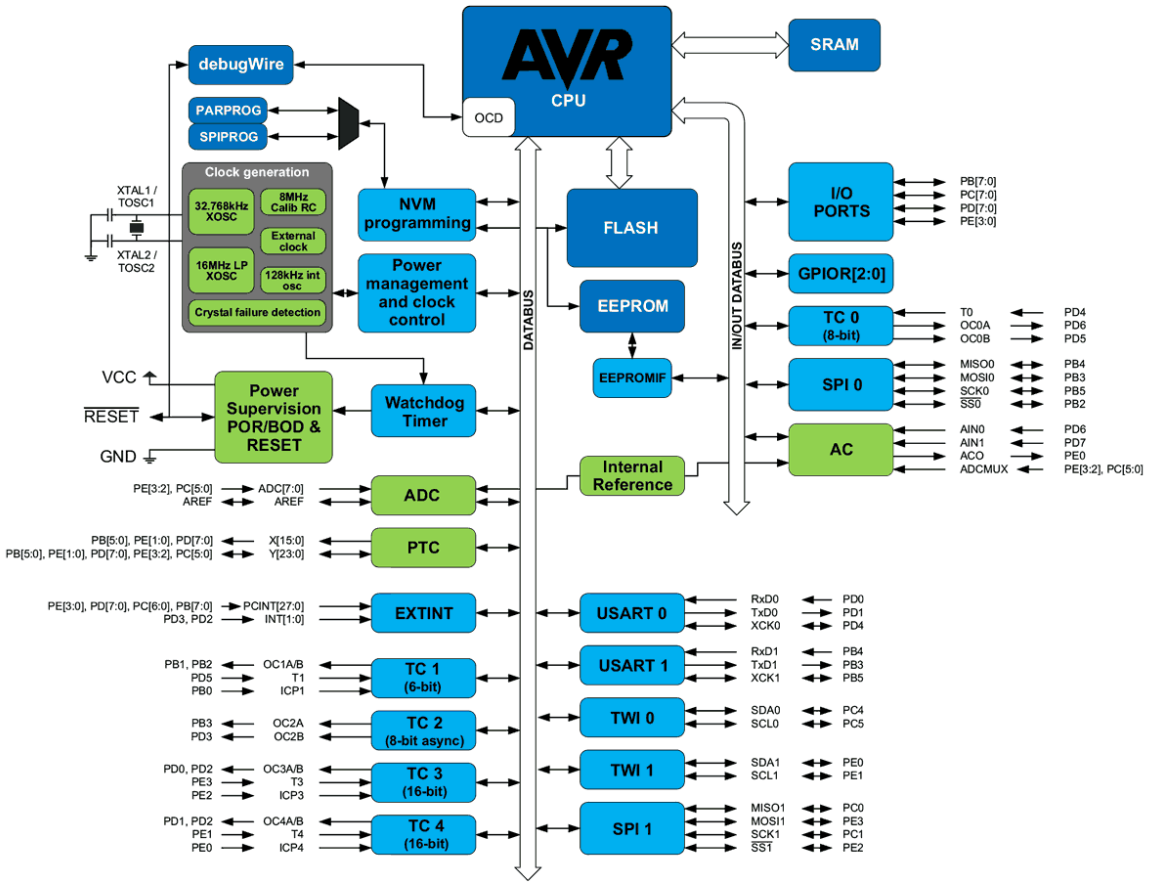
4.10 Arduino Nano

Modul o rozměrech 23x43mm platformy Arduino s mikrokontrolerem ATmega328p, obvodem FT232R pro komunikaci s rozhraním USB a lineárním stabilizátorem AMS1117 s výstupním napětím 3.3 V. Na tomto modulu je zabudován mikrokontroler generace ATmega, od firmy Atmel, která byla v roce 2016 odkoupena firmou Microchip. [13]

Mikrokontroler ATmega328 obsahuje osmibitový procesor, s harvardskou architekturou, což znamená, že má oddělenou paměť pro program a pro data. V případě mikrokontrolerů určených pro jedinou úlohu po celou dobu fungování zařízení, je tato architektura vhodnější, protože nahraný program se již nemění. Procesor obsahuje 32 osmibitových registrů, které pracují s 16bitovou instrukční sadou RISC. Sada je zúžena na jednoduché a často používané instrukce, což vede k jednoduššímu návrhu obvodu s cílem zrychlit jeho činnost. Procesory s výše uvedenými vlastnostmi se označují AVR. Vnitřní uspořádání je zachyceno na obrázku 10. [5]

Pro program je určena paměť flash o velikostí 32 kB a pro data paměť SRAM o velikosti 2 kB. MCU obsahuje periferie jako jsou USART pro synchronní sériovou komunikaci, SPI, TWI (Philips I^2C), čítače, 1 kB EEPROM pro uchování údajů při výpadku napájení. Dále obvod Watchdog pro ochranu proti zacyklení, obvod BOD pro reset po zapnutí

zařízení, nebo při poklesu napětí pod kritickou úroveň a bootloader, který je určen k nahrání programu do flash paměti mikrokontroleru. [5] [4]



Obrázek 10: *Vnitřní uspořádání mikrokontroleru AVR*

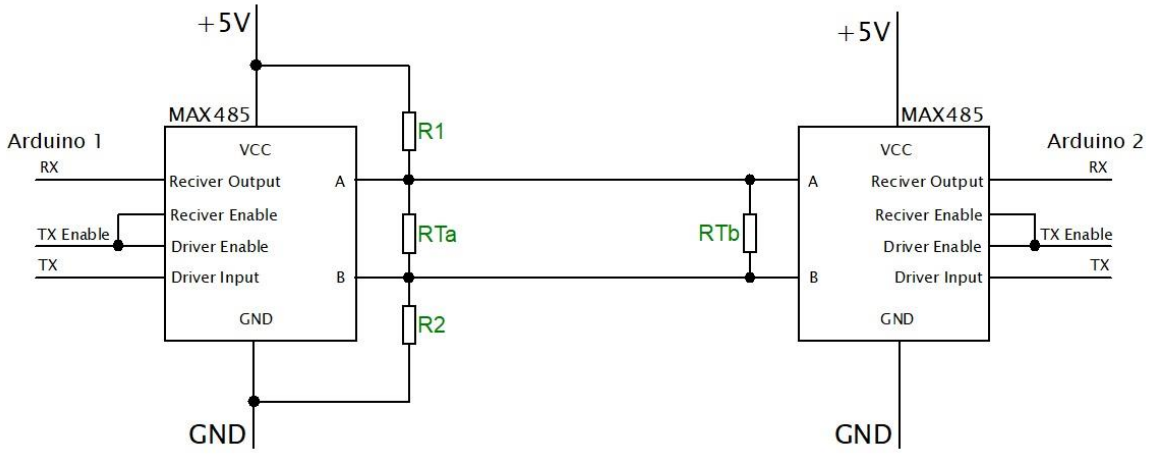
Zdroj: **[32]**

4.11 **Rozhraní RS485**

Rozhraní asynchronní sériové komunikace RS485, které bylo definováno v roce 1983. Nachází uplatnění především v průmyslových aplikacích. Pro vytvoření fungující sítě stačí jeden pár vodičů, obvykle označených jako A a B. Pro vodiče s kapacitou do 65pF/m může být celková vzdálenost linky až 1600 m. Při takové vzdálenosti klesá rychlost přenosu dat pod 100kBps, zatímco při vzdálenosti 10 metrů je možné přenášet data rychlostí 10 MBps.

Dosah lze zvýšit použitím opakovačů. Při správném provedení vyniká velkou spolehlivostí přenosu dat. [14] [15]

Komunikace probíhá asynchronně v polo-duplexním režimu. Na fyzické vrstvě jsou jednotlivé znaky přenášeny stejně jako u rozhraní UART, tzn. nulový start bit, pak jednotlivé datové bity a na konci stop bit reprezentovaný logickou 1. Datové bity jsou identifikovány podle vzájemné polarity mezi vodiči A a B při minimálním napětí 200mV. Pro rozdíl A–B > 200mV je definován logický stav 0 a pro B-A > 200mV je definován logický stav 1. Přidáním další komunikační linky lze dosáhnout plně-duplexního režimu, protože jedna linka odesílá a druhá přijímá. V režimu polovičního-duplexu je nutné pomocí softwaru zajistit korektní přepínání mezi přijímáním a odesíláním dat. Na jednu linku lze standartně připojit až 32 zařízení, přičemž komunikace musí být softwarově ošetřena, aby nedocházelo ke konfliktům. Oba konce linky musí být chráněny zakončovacími rezistory, tzv. „terminátory“. Ty zabraňují rušení, které vzniká odrazem signálu od konce vedení. Definování klidového stavu je zajištěno připojením vodičů A přes rezistor k zemi a B přes rezistor k napájení, což zvyšuje odolnost linky proti rušení indukovaným napětím. Moduly s obvodem MAX485, určené pro mikrokontrolery platformy Arduino, mají všechny tyto rezistory již zabudované. Pokud nebude modul použit na konci linky je třeba odstranit terminační rezistor.



Obrázek 11: *Zapojení dvou uzlů v síti RS485*

Zdroj: vlastní zpracování

Zapojení mezi dvěma uzly je vytvořeno pomocí dvou převodníků s obvodem MAX485 připojených k rozhraní UART mikrokontroleru Arduino (viz obrázek 11). Vysílání, nebo přijímání je řízeno pinem mikrokontroleru, připojeným k TX Enable. [14] [15]

Při větších vzdálenostech je nutné použití třetího vodiče k propojení zemního potenciálu, ale je třeba brát ohled na velikost napětí, které vzniká mezi komunikujícími uzly. U rozdílu 7 V by docházelo k nespolehlivé funkci a při více než 14 V k poškození zařízení komunikace. Aby nedocházelo k zemním smyčkám vlivem propojení potenciálů s velkým rozdílem je třeba použít galvanické oddělení. [16]

4.12 **Senzory**

Následující senzory umožňují měřit meteorologické veličiny a předávat je k dalšímu zpracování. Komunikace je umožněna přes rozhraní I^2C, digitální, nebo analogový vstup mikrokontroleru.

4.12.1 **Senzor HTU21D**

Velmi přesný senzor pro měření teploty a vlhkosti. Rozsah měřené teploty je od -40 °C až po 120°C. Nejmenšího zkreslení 0,3 °C při měření teploty je dosaženo mezi 5 a 60 stupni celsia. Pro měření vlhkosti platí optimální rozsah mezi 20 % a 80 % při odchylce dvou procent. Standartní rozlišení pro měření teploty je 2^{12} (0,04 °C) a pro vlhkost 2^8 (0,4 %). Rozlišení lze měnit, stejně jako dobu měření pomocí zápisu do řídících registrů. Dále je možné využít funkce vyhřívání pro rychlejší vysušení čidla vlivem dlouhodobé vlhkosti. Napájecí napětí je 1,5-3,6 V. Maximální odběr proudu při měření může dosáhnout

Obrázek 12: *Modul se senzorem HTU21D*

Zdroj: vlastní zpracování

až 500 µA, ale v klidovém stavu odebírá pouze 140 nA. Komunikace s mikrokontrolerem je zprostředkována pomocí rozhraní I^2C. Modul s HTU21D je na obrázku 12. [17]

4.12.2 **Senzor Bosch BMP180**

Senzor pro měření barometrického tlaku od firmy Bosch Sensortec. Obsahuje také teplotní čidlo, které ale nedosahuje takové přesnosti jako v případě htu21d. Jedná se o výkonný senzor, který je svými vlastnostmi vhodný do zařízení malých rozměrů s minimální spotřebou elektrické energie. Je to nástupce modelu BMP085. Rozsah měření tlaku se pohybuje od 300 do 1100 hPa. Komunikace probíhá přes rozhraní I^2C. Měření je nezávislé na kolísání napájecího napětí, které se může pohybovat mezi 1,62 V a 3,6 V. Spotřeba energie v klidovém stavu se pohybuje okolo 3 µA, ve špičce při měření a posílání může být odběr až 650 µA.

Obrázek 13: *Modul se senzorem BMP180*

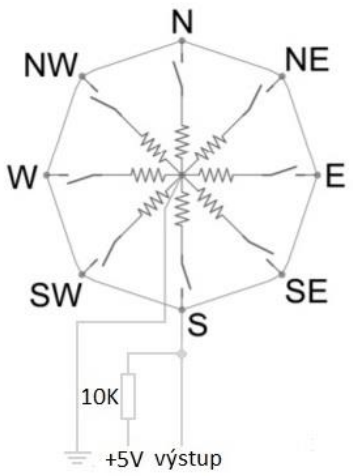
Zdroj: vlastní zpracování

Senzor mění svůj elektrický odpor v závislosti na tlaku, získaná analogová hodnota je převedena pomocí vestavěného ADC převodníku, a poté je zpracována pomocí řídící jednotky s pamětí E^2PROM a odeslána přes rozhraní I^2C. V paměti E^2PROM je uloženo 22 bajtů kalibračních dat, která jsou využívána pro kompenzaci negativních vlivů při měření. Přenos dat probíhá rychlostí 3,4 MHz. V řídících registrech je možné nastavit počet měření za sekundu, respektive přesnost. [18]

4.12.3 **Anemometr**

Zařízení pro měření rychlosti a směru větru. Měření rychlosti větru probíhá pomocí jazýčkového spínače a magnetu, který způsobí jedno sepnutí každou otáčku. Pro rychlost větru 2,4 km/h je sepnutí provedeno jedenkrát za sekundu.

Pro měření směru větru je využito osmi jazýčkových kontaktů, pravidelně rozmístěných do kruhu a připojených přes rozdílné hodnoty rezistorů. Při sepnutí dvou kontaktů zároveň vzniká další kombinace, to znamená, že je k dispozici celkem 16 pozic s intervalem 22,5° s rozdílnou hodnotou elektrického odporu. Po připojení pull-up rezistoru jsou jednotlivé hodnoty reprezentovány šestnácti rozdílnými hodnotami napětí, které jsou zpracovány analogovým vstupem mikrokontroleru. Zapojení a jednotlivé hodnoty jsou na obrázku 14.



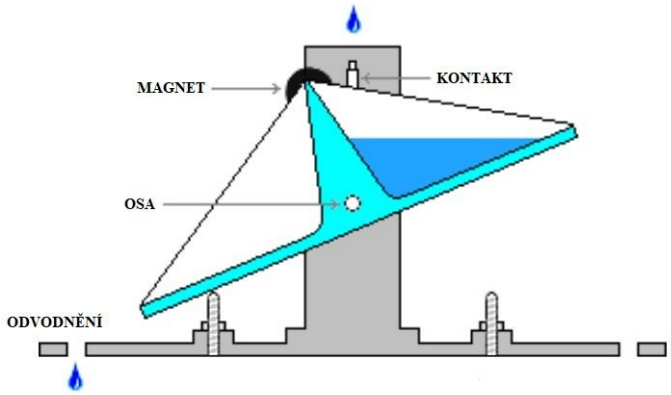
směr (°)	odpor (kΩ)	výstup (V)
0	33k	3.84v
22.5	6.57k	1.98v
45	8.2k	2.25v
67.5	891	0.41v
90	1k	0.45v
112.5	688	0.32v
135	2.2k	0.90v
157.5	1.41k	0.62v
180	3.9k	1.40v
202.5	3.14k	1.19v
225	16k	3.08v
247.5	14.12k	2.93v
270	120k	4.62v
292.5	42.12k	4.04v
315	64.9k	4.78v
337.5	21.88k	3.43v

Obrázek 14: *Zapojení kontaktů vč tabulky hodnot výstupu měření směru větru*

Zdroj: **[34]**

4.12.4 **Srážkoměr**

Měření probíhá pomocí děleného překlápěcího člunku, namontovaného na osu. Na člunku jsou umístěné dvě malé misky o objemu, který reprezentuje úhrn 0,3 mm/m^2. Tato sestava je umístěna pod sběračem deště, který převádí zachycené srážky do misek. Když je miska zaplněna, vlivem váhy kapaliny dojde k jejímu překlopení a vyprázdnění. Překlopením dojde k natočení druhé misky pod sběrač srážek a cyklus se opakuje. Na člunku je umístěn magnet, který při každém překlopení sepne jazýčkový kontakt, takže tento impuls odpovídá ekvivalentu úhrnu srážek 0,3 mm /m^2 (viz obrázek 15).



Obrázek 15: *Člunkový srážkoměr*

Zdroj: **[35]**

4.13 **Záložní zdroj UPS**

Hlavní část záložního zdroje pro případ výpadku napájení bude tvořit modul s obvodem TP4056 pro bezpečné nabíjení akumulátoru a FS8205A pro ochranu proti přepětí, podpětí a nadproudu. Zároveň tento modul umožňuje v případě ztráty vstupního napětí plynulý přechod na akumulátor bez jakéhokoliv zakolísání napětí. Dále je doplněn o step-up měnič, který upravuje nižší napětí z akumulátoru na požadovaných výstupních 5 V. Vstupní napětí pro obvod TP4056 je 2.8-4.2 V. Pro zdroj záložní energie bude použit akumulátor li-ion s napětím 3,6 V kapacitou 3600 mAh.

5 SOFTWARE

K vytváření zdrojového kódu mikrokontrolerů platformy Arduino je určeno vývojové prostředí IDE (Integrated Development Enviroment), které je napsané v jazyce Java. Jde o mírně upravený software z výukového prostředí Processing s přidanými funkcemi a podporou jazyka Wiring.

Pomocí programových prostředků jazyků PHP, JavaScript a HTML je možné prezentovat data, uložená v databázovém systému, který komunikuje prostřednictvím jazyka SQL.

5.1 Jazyk Wiring

Arduino je možné programovat přímo v jazyce C, nebo C++, ale pro snadnější programování se využívá knihovny Wiring. Pro její rozsah a složitost se o ní hovoří jako o samostatném programovacím jazyku. Vzhledem k tomu, že vychází z jazyka C++, lze využívat výhod objektově orientovaného programování. [1]

Základní program v jazyce Wiring se skládá ze dvou klíčových bloků. První z nich, pojmenovaný „setup“, obsahuje kód, který se výkonná pouze jednou, a to po spuštění zařízení, nebo jeho restartu. Druhá část, pojmenovaná „loop“, obsahuje kód, který se neustále opakuje až do vypnutí zařízení. Tyto bloky jsou povinné a musí být vloženy i v případě, že je jejich obsah prázdný. Zápis jednotlivých příkazů probíhá vložením mezi složené závorky, přičemž jsou jednotlivé příkazy odděleny středníkem. Jednoduchá ukázka

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  delay(1000);
  Serial.print("Arduino");
}
```

Obrázek 16: *Struktura jednoduchého kódu v jazyce Wiring*

Zdroj: vlastní zpracování

základního kódu v jazyce Wiring, který provádí výpis do konzole prostředí Arduino IDE je na obrázku 16. Po spuštění zařízení je nejprve nastavena sériová komunikace, která bude probíhat rychlostí 9600 bitů za sekundu, a poté je spuštěna smyčka, která bude vykonávat každou sekundu výpis textu „Arduino“. Syntaxe je velmi podobná jako v jazyce C, bloky jsou uzavřeny složenými závorkami, jednotlivé příkazy odděleny středníkem, stejný zápis funkcí, deklarací proměnných, práce s ukazatelem, nebo referencí, podmínky, cykly a další. [1]

Pro úspěšné nahrání kódu je třeba propojit port USB počítače se vstupem převodníku vývojové desky mikrokontroleru. V prostředí IDE je nutné nastavit v záložce „Nástroje“ správný typ vývojové desky mikrokontroleru a číslo COM portu, a poté zmáčknout tlačítko nahrát, nebo klávesovou zkratku Ctrl+U. Následně je kód zkontrolován a v případě, že nejsou nalezeny syntaktické chyby je zkompilován a nahrán přes sériový převodník FT232R (kapitola 4.10), nebo CP2102 (kapitola 4.4) do paměti flash. [19]

5.2 Nastavení rozhraní v jazyce Wiring

Nastavení jednotlivých pinů pro vstup, nebo výstup je definováno v prvním bloku „setup“. Provádí se zápisem funkce „pinMode“, která má první parametr číslo pinu a druhý parametr reprezentuje směr dat. Pro vstup je určen parametr „INPUT“ a pro výstup parametr „OUTPUT“, k tomu je možné softwarově zapnout vnitřní pull-up, nebo pull-down rezistor. Například „pinMode(12, INPUT_PULLUP)“ znamená, že pin č.12 je nastaven jako vstup s rezistorem připojeným ke kladnému potenciálu napájení mikrokontroleru. K načítání stavu digitálního pinu slouží příkaz „digitalRead“ s parametrem určujícím číslo pinu a pro zápis příkaz „digitalWrite“ s prvním parametrem pro číslo pinu a druhým pro binární hodnotou 1, nebo 0. Téměř stejně funguje příkaz pro analogový pin „analogRead“, respektive „analogWrite“, ale místo binární hodnoty je hodnota určená dle rozlišení ADC, nebo DAC převodníku, např. u Arduino Nano to je 2^{10} (0-1023). Pro sériová rozhraní jsou určeny specifické piny (viz kapitola 4.2). Tyto piny jsou obvykle definovány v tzv. knihovnách. [1]

5.3 Knihovny pro Arduino

Jedná se o předpřipravené balíčky kódu v jazyce C, nebo C++, které rozšiřují základní program o další funkcionality. Tyto balíčky mohou přidávat například funkce pro jednotku času RTC3231, kartu SD, senzory připojené přes I^2C, nebo pro práci s časem v Arduinu. Jde o soubory, které vytvářejí třídu obsahující metody a vlastnosti. Jedná se o hlavičkový soubor s koncovkou *.h, který je následně připojen v souboru hlavního kódu knihovny s koncovkou *.cpp. Oba soubory jsou uloženy ve stejné složce, která má stejné jméno a je obvykle umístěná v adresáři s názvem „libraries“. V hlavičkovém souboru jsou deklarovány proměnné, popis funkcí a úrovně přístupu. V souboru *.cpp jsou definovány vlastnosti, funkce a konstruktor. Soubory *.h jsou zahrnuty do hlavního souboru Arduina

hlavičkový soubor mat.h

```
class Funkce
{
private: //deklarace privatních proměnných a funkcí
public:  //deklarace veřejných proměnných, funkcí a konstruktoru (povinný)
    Funkce();
    int secti(int prvni, int druhe);
};
```

kód knihovny mat.cpp

```
#include <mat.h>
//konstruktor
Funkce::Funkce() {
}
//funkce pro sčítání
int Funkce::secti(int prvni, int druhe) {
    return prvni + druhe;
}
```

hlavní soubor *.ino

```
#include <mat.h>
Funkce mat; //zavoláním konstruktoru je vytvořena instance
void setup() {
  Serial.begin(9600);
  Serial.print(mat.secti(2, 3)); //vypíše do konzole IDE součet 5 (2+3)
}

void loop() {
}
```

Obrázek 17: *Příklad knihovny pro sčítání dvou čísel*

Zdroj: vlastní zpracování

s koncovkou *.ino pomocí příkazu #include. Některé základní funkce jsou součástí knihoven prostředí Arduino IDE, a proto nemusí být dodatečně zahrnovány, jedná se například o funkce třídy „Serial“. V hlavním kódu je zavoláním konstruktoru vytvořena instance třídy, na které je možné volat všechny její veřejné funkce a vlastnosti. Velké množství knihoven je již vytvořeno a jsou volně ke stažení. Příklad jednoduché knihovny pro sčítání dvou celých čísel je na obrázku 17. Samozřejmě je možné si knihovnu pro určité zařízení vytvořit, nebo upravit, ale je nutné mít k dispozici specifikaci výrobce hardwaru, většinou ve formě datového listu, tzv. datasheetu, kde jsou uvedeny jednotlivé adresy řídících a stavových registrů, případně ostatní možnosti nastavení. [20] [21]

5.4 **Webhosting**

Jedná se o pronájem prostoru na disku serveru včetně souvisejících služeb pro prezentaci webových stránek. Jde o sdíleny webhosting, tzn., že hardware serveru je sdílený společně s dalšími zákazníky. Zprostředkovatelem webhostingu je zaregistrováno doménové jméno neboli URL (Uniform Resource Locator), např. arduinopj01.cz, které slouží pro jednoznačnou identifikaci umístění v síti internet. Správa souborů webových stránek na serveru je zajištěna přes aplikaci FTP server s komunikací přes protokol FTP. Velikost přiděleného prostoru pro soubory je 700 MB. Samotná data a jejich správa probíhá pomocí databázového systému MySQL, který komunikuje přes rozhraní vytvořeném v kódu PHP. Maximální velikost jedné databáze může být 50 MB, přičemž k dispozici je dohromady 100 MB. Mezi další služby patří např. Anti-DDoS ochrana. AntiMalware ochrana, HTTPS, nebo SSL certifikát. [22]

5.4.1 **FTP**

Je to protokol pro přenos a změny v souborech mezi dvěma počítači, které nejsou v jedné lokální sítí, ale jsou připojeny k síti internet kdekoliv na světě. Pro práci s jednotlivými soubory na serveru je k dispozici webová aplikace od poskytovatele webhostingu. Při hromadné práci se soubory je vhodnější nainstalovat do počítače vlastního FTP klienta, např. program FileZilla. [23]

FTP protokol používá dva samostatné kanály. Řídící kanál na serverovém portu 21 je určen k přijímání klientských připojení a jednoduchých příkazů, jako jsou např. autentizace, nebo zahájení komunikace pro přenos. Řídící kanál zůstává otevřen až do odpojení klienta,

nebo ukončení činnosti ze strany serveru. Pokud je autentizace úspěšná, je otevřen nový společný port pro datový kanál, po kterém jsou soubory, nebo jejich změny přenášeny. Jakmile je přenos dokončen, řídící kanál odešle hlášení o úspěchu, nebo selhání. Z důvodu vyšší bezpečnosti se používá zabezpečený přenos pomocí vrstvy TLS (Transport Layer Security), označovaný jako FTPS. Ten šifruje jak datový, tak řídící kanál. [23]

5.4.2 **Databázový systém MySQL**

Nejrozšířenější open-source databázový systém na světě. Vyniká pro svou výkonost, spolehlivost a snadnost používání, o čemž svědčí jeho nasazení ve společnostech jako jsou Facebook, Twiter, nebo YouTube. Byl vytvořen švédskou firmou MySQL AB, později prodán firmě Sun Microsystems, která je od roku 2010 vlastněna společností Oracle Corporation. Velmi často se používá označení této databáze ve zkratce LAMP, která znamená spojení operačního systému Linux, webové služby Apache, databázového systému MySQL a programovacího jazyka PHP, Perlu, nebo Phytonu. [24]

Bohužel při použití levnějšího webhostingu nejsou k dispozici pokročilejší funkce databázového systému jako vytváření automatických úloh pomocí triggerů. Navíc možnost kontroly doménové integrity pomocí příkazu „check“ je až do verze 8.0.16 ignorována, takže rozsah hodnot je nutné kontrolovat v rámci aplikační vrstvy, např. PHP skriptem. [25]

Samotná data jsou fyzicky uložena v databázi a přístup k nim je umožněn pouze přes systém řízení báze dat, dohromady tvoří databázový systém. Každá databáze může mít jednu, nebo více tabulek, přičemž každá tabulka by měla obsahovat položky jednoho typu. Každý řádek v tabulce reprezentuje jeden záznam a sloupce představují vlastnosti těchto záznamů. Jednotlivé záznamy (entity) v tabulce by měli být odlišeny pomocí unikátního klíče. Komunikace s databázovým systémem probíhá pomocí strukturovaného dotazovacího jazyka SQL, který umožňuje z prostředí příkazové řádky např. vytvářet databáze, tabulky, nebo manipulovat s daty (vyhledávání, vkládání, mazání, aktualizace). [26]

5.4.3 **PHP**

Velmi populární skriptovací multiplatformní objektový jazyk, který umožňuje programovat webové aplikace, nebo přidávat do webových stránek interaktivní prvky. Na rozdíl od JavaScript jde o tzv server-side jazyk, který se provádí na straně serveru, kde dochází po spuštění skriptu k vygenerování HTML stránky a ta je poté odeslána

klientovi. Jedna z nejvýznamnějších funkci tohoto jazyka je podpora nejrůznějších databázových systémů včetně MySQL. Komunikace s databázovým systémem probíhá pomocí třídy mysqli, přičemž k předání dat do JavaScript je možné použít formát JSON. Z bezpečnostních důvodů je důležité se vyvarovat vkládání proměnných do dotazu sql. V opačném případě hrozí, že útočník může přímým vložením části sql dotazu do pole formuláře, nebo adresy url, manipulovat s databází. Příklad výměny dat pomocí PHP skriptu z MySQL do objektu JSON je uveden v následujícím kódu na obrázku 18. [27] [28]

```php
<?php
header('Content-Type: application/json'); //nastavení HTTP hlavičky pro JSON
$mysqli = mysqli_connect("127.0.0.1","adresa.cz","heslo","jménoDB");//připojení
if($mysqli and isset($_POST['from']) and isset($_POST['to']))
{
      $from = $_POST["from"];$to = $_POST["to");
       $query = "SELECT date_time,pressure FROM table_weather WHERE date_time
BETWEEN ? AND ? ORDER by date_time asc";
       $stmt = $mysqli->prepare($query);
       $stmt ->bind_param('ss',$from,$to);
       $stmt->execute();
       $result = $stmt->get_result();
}
else
{
  die("Chyba připojení: " . $mysqli->error);
}
$data = array(); //smyčka pro uložení dat do pole
foreach ($result as $row)
{
  $data[] = $row;
}
$result->close(); //uvolnění paměti
$mysqli->close(); //uzavření spojení
print json_encode($data); //předání do objektu JSON
```

Obrázek 18: *Předání dat z databáze do JSON pomocí PHP*

Zdroj: vlastní zpracování

5.5 JavaScript

Multiplatformní objektově orientovaný skriptovací jazyk, který je jedním ze základních prvků současných webových aplikací. Umožňuje webovým stránkám dodat prvky dynamického chování, jako jsou např. vyskakovací okna, vykreslování grafů, nebo interakce při vyplňování polí ve formulářích (např. automatické doplňování výrazu při vyhledávání v google). K tomu se používá aplikační programové rozhraní DOM (Document Object

Model), které definuje logickou strukturu dokumentu, pomocí které lze přistupovat k jednotlivým prvkům a pracovat s nimi. Velká výhoda spočívá v tom, že javaScript běží v prohlížeči klienta, využívá výkon jeho hardwaru a data nemusí být opakovaně stahována ze serveru. Bohužel je to i nevýhoda, jelikož se zdrojový kód nachází u klienta, tak ho lze bez problému získat, upravovat a předávat dál. Javascript umožňuje tzv. AJAX (Asynchronous JavaScript and XML). Jde o schopnost asynchronního načítání kódu. Pokud je při zpracování scriptu odeslán nějaký požadavek na server, script nemusí bezprostředně čekat na odpověď a pokračuje dál v provádění kódu. Ve chvíli získání odpovědi od serveru je provádění kódu pozastaveno (dříve, či později podle priority prováděné operace) a je spuštěna funkce, tzv. callback. Taková funkce se nazývá neblokující. Vzhledem k této vlastnosti je možné vyměňovat data se serverem a aktualizovat je v rámci webové stránky, aniž by bylo nutné znovu načítat celou stránku. Lze také použít knihovny pro přidání dalších funkcí, např. jquery, bootstrap, nebo highcharts pro tvorbu interaktivních grafů. [29] [30]

5.6 HTML A CSS

HTML je značkovací jazyk pro vytváření webových prezentací. Propojuje jednotlivé elementy (texty, obrázky, tabulky, grafy atd.), které jsou definovány pomocí značek, do logické struktury dokumentu. Tyto elementy jsou mezi sebou propojené odkazy, tzv. hypertexty, čímž vzniká uspořádaný celek existujících informací z různých zdrojů. Jazyk byl vytvořil Tim Berners-Lee v roce 1989, v laboratořích CERN pro potřeby prezentace a sdílení výsledků výzkumu. V současné době je spravován konsorciem W3C. Pro způsob zobrazení jednotlivých elementů v rámci dokumentu HTML se využívá kaskádových stylů, tzv CSS (Cascading Style Sheets). Ty specifikují např. barvu písma, velikost písma, umístění prvků atd.

6 VÝROBA METEOSTANICE

Meteostanice by měla měřit meteorologické veličiny pomocí senzorů, dostupných v rámci platformy Arduino a předávat je k dalšímu zpracování. Senzory přenášejí data prostřednictvím signálů analogového, digitálního, nebo sériového rozhraní. Data lze posílat na maximální možnou vzdálenost, při které nedochází ke zkreslení signálu. Pokud vzdálenost převyšuje přípustnou hodnotu, je nutné nejprve pomocí mikrokontroleru načíst data ze senzoru, upravit pro odeslání, a pak přenést přes rozhraní, které splňuje požadované vlastnosti. Přenesená data musí být dále zpracována tak, aby nad nimi bylo možné provádět operace umožňující zobrazování aktuálních meteorologických veličin včetně jejich dlouhodobých statistik. Provádění operací, jako jsou vykreslování grafů dlouhodobých měření, vyžaduje zařízení, které disponuje vhodným výpočetním výkonem, např. běžný osobní počítač, nebo smartphone. Vzhledem k požadavkům a maximálnímu dostupnému výkonu, může zařízení platformy Arduino zprostředkovávat pouze funkce pro spolehlivé zajištění přenosu kompletních dat do databáze. Z databáze jsou data dostupná v rámci sítě internet a zpracovatelná zařízením, které obsahuje webový prohlížeč.

Senzory měření rychlosti větru a srážkoměru budou připojeny k digitálním vstupům mikrokontroleru první senzorové jednotky. Měřič směru větru využívající odporový dělič, který generuje různou velikost napětí, bude připojen k analogovému vstupu. Pulsy a analogový signál od těchto senzorů zpracuje modul Arduino Nano s MCU ATmega328p a následně pošle po sériové RS485 na vzdálenost cca 33 m. Druhá senzorová jednotka bude obsahovat stejný MCU a data ze senzorů pro měření teploty, vlhkosti a tlaku načítat přes rozhraní I^2C a poté posílat po RS485 na vzdálenost cca 7 m. Přijatá data upraví a odešle do databáze hlavní jednotka s MCU ESP32, vybavena zařízením UPS s čipem TP4056 pro zálohu napájení, paměťovým úložištěm s kapacitou 512 Mb s rozhraním SPI a jednotkou pro zajištění správné hodnoty reálného času, komunikující po standartu I^2C. Databázový systém MySQL s podporou jazyka PHP bude sloužit pouze jako úložiště pro naměřená data, která budou načítána ve formě struktury JSON, zpracována kódem javaScript s frameworky pro vykreslení grafů a nakonec v rámci webové prezentace zobrazena uživateli.

6.1 Senzorové jednotky

Vytvořená meteostanice obsahuje dvě senzorové jednotky. Umístění senzorových jednotek je provedeno s ohledem na dosažení nejmenšího možného zkreslení. První, která měří srážky, rychlost větru a směr větru je umístěná na střeše obytného domu ve výšce 14 m od země, tak aby ji nestínily žádné překážky jako jsou vzrostlé stromy, nebo okolní zástavba. Druhá, která měří teplotu, vlhkost a tlak je umístěna na severní zdi domu, kde není ovlivňována přímým slunečním svitem.

Každá senzorová jednotka má svou řídící jednotku ve formě jednodeskového počítače Arduino Nano, který zpracovává naměřená data. Ty jsou na základě požadavku hlavní jednotky odesílány po sériovém rozhraní RS485. Pro tuto komunikaci byl vytvořen protokol, pomocí kterého jsou přenášeny požadovaná data po rámcích, tvořených jednotlivými znaky včetně kontrolního součtu (více v kapitole 6.4). Výše zmíněné vlastnosti umožňují rozmístění senzorových jednotek na velkou vzdálenost, aniž by docházelo ke zkreslování naměřených údajů. V rámci technických prostředků, které byly k dispozici byla odzkoušena vzdálenost 300 m. Jednotka pro měření větru a srážek je vzdálena cca 35 m a jednotka pro teplotu, vlhkost a tlak cca 7 m od hlavní jednotky. V blízkosti každé jednotky je umístěna diagnostická zásuvka, určena pro úpravy zdrojového kódu, nebo pro diagnostický výpis dané senzorové jednotky. Propojení je realizováno párem vodičů síťového kabelu FTP cat 5e, přičemž další pár slouží k napájení jednotek. Venkovní část kabeláže je vyrobena z UV odolného materiálu. Vzhledem k maximální vzdálenosti 35 m a maximálnímu naměřenému odběru 25 mA se úbytek napětí pohybuje mezi 40 a 80 mV.

6.1.1 Jednotka pro anemometr a srážky

Hardware jednotky je tvořen jednodeskovým počítačem Arduino Nano, modulem pro komunikaci s hlavní jednotkou meteostanice, anemometrem pro snímání rychlosti a směru větru a člunkovým srážkoměrem. Kompletní hotová senzorová jednotka pro měření rychlosti, směru větru a množství srážek je na obrázku 19.

Jednotka umožňuje měření průměrné a nárazové rychlosti větru. Základem pro výpočet hodnot rychlosti větru jsou jednotlivé časové intervaly, které jsou generovány pomocí magnetického kontaktu měřiče rychlosti větru připojeného na vstup mikrokontroleru ATmega328p. Intervaly jsou ukládány v poli hodnot a poté jsou provedeny výpočty maximální a průměrné rychlosti. Výpočet je proveden po nasbírání 24 vzorků, nebo po překročení časové kvóty 24 sekund. Jelikož není nikdy známa počáteční hodnota pro první naměřený interval, je tento interval při výpočtech vynechán, aby nedocházelo ke zkreslení. Další nezbytností je ošetření chyb, které jsou způsobeny zákmity kontaktů, pomocí vložení časového zpoždění (5ms) a pomocné proměnné udržující informaci o posledním stavu kontaktů. Samotný výpočet je proveden metodou pro získání průměrné hodnoty a nalezení nejmenšího časového intervalu v poli hodnot. Získané hodnoty jsou poté pomocí konstanty přepočítány na rychlost v m/s.

Obrázek 19: *Senzorová jednotka anemometru a srážkoměru*

Měření směru větru je realizováno čtením analogových hodnot z výstupu větrné korouhvičky. Okamžitá získaná hodnota je porovnána v rámci definovaných rozsahů, které reprezentují šestnáct možných směrů větru. Po porovnání je hodnota zařazena jako celé číslo, jehož velikost odpovídá pozici v poli hodnot, např. hodnota 1 bude uložena na pozici 1 (index 0), atd. Pro spolehlivé měření není třeba opakovaný zápis již získané pozice, takže je v rámci časové kvóty pěti sekund každá získaná hodnota zapsána pouze jednou. Vzhledem k tomu, že při měření směru větru dochází k výchylkám korouhvičky, a tudíž k uložení více rozdílných hodnot, je třeba získat střední hodnotu kolem které oscilace probíhá. K tomu byla vytvořena funkce zaměřená na výpočet střední hodnoty v kruhovém poli. Pomocí získané hodnoty a konstanty 22,5 je následně určen směr větru ve stupních. Odeslané hodnoty směru i rychlosti větru jsou vypočítány z údajů získaných za posledních 30 minut.

Poslední částí senzorové jednotky je člunkový srážkoměr. Při dosažení množství, které způsobí překlopení člunku, respektive sepnutí jazýčkového kontaktu, je spuštěno přerušení provádění hlavního kódu senzorové jednotky. Přerušení je aktivováno na vzestupné hraně signálu, poté je vykonána jednoduchá funkce, která iteruje proměnnou pro pulsy srážkoměru. Následně je řízení vráceno a provádění kódu pokračuje od místa, kde byl přerušen. Ošetření zákmitu kontaktů je provedeno kontrolou minimálních časových intervalů mezi překlopením člunku do opačné polohy. Počet pulsů srážkoměru vynásobený konstantou 0,3 je každých 30 minut odeslán hlavní jednotce jako úhrn srážek v mm. Po potvrzení o doručení je úhrn vynulován a počítání začíná od nuly. Pokud by nedošlo k odeslání srážkových dat je tato informace uchována až do potvrzení o úspěšném doručení, nebo odpojení napájení, popř. restartu jednotky.

6.1.2 **Jednotka pro teplotu, vlhkost a tlak**

Zařízení je sestaveno z jednodeskového počítače Arduino Nano, senzoru pro měření teploty, vlhkosti, tlaku a modulu, který umožňuje komunikaci s hlavní jednotkou pomocí rozhraní RS485. Vše je umístěno v PVC tubusu, zakrytém radiačním krytem. Kompletní funkční jednotka je na následujícím obrázku.

Obrázek 20: *Senzorová jednotka teploty, tlaku a vlhkosti*

Měření teploty a vlhkosti zprostředkovává senzor HTU21D, který se vyznačuje velmi vysokou přesností. Data ze senzoru jsou načítána prostřednictvím sériové linky standartu I^2C a uložena do proměnných pro další zpracování. Komunikace mezi senzorem a mikrokontrolerem ATmega328p, včetně výpočtu hodnot teploty a vlhkosti, je řízena kódem knihovny SparkFunHTU21D.h, která byla napsána výrobcem senzoru SparkFun Electronics.

Hodnota tlaku je získána pomocí senzoru BMP180. Naměřený tlak je vypočítán a předán po sériové lince I^2C mikrokontroleru ATmega328p pro další zpracování za použití knihovny Adafruit_BMP085. Následně je třeba naměřený absolutní tlak přepočítat na tlak u hladiny moře, přičemž je třeba zohlednit i teplotu, která může mít vliv až 5 mbar při rozdílu 40 °C a nadmořskou výšku, která dosahuje v místě instalace zařízení 231 m. Pro výpočet tlaku u hladiny moře slouží následující vzorec.

$$tlak\ u\ hladiny\ moře\ = \frac{abs.tlak\ *\ 9.80665\ *\ nadm.\ výška}{\left[287 * \left(273 + teplota + \frac{nadm.výška}{400}\right)\right]} + abs.tlak$$

Ze všech získaných hodnot je sestaven textový řetězec, ke kterému je přidán kontrolní součet včetně id jednotky a dohromady tak tvoří základ pro datový rámec předávaný hlavní jednotce k dalšímu zpracování. Zdrojový kód hlavního programu senzorové jednotky je na následujícím obrázku.

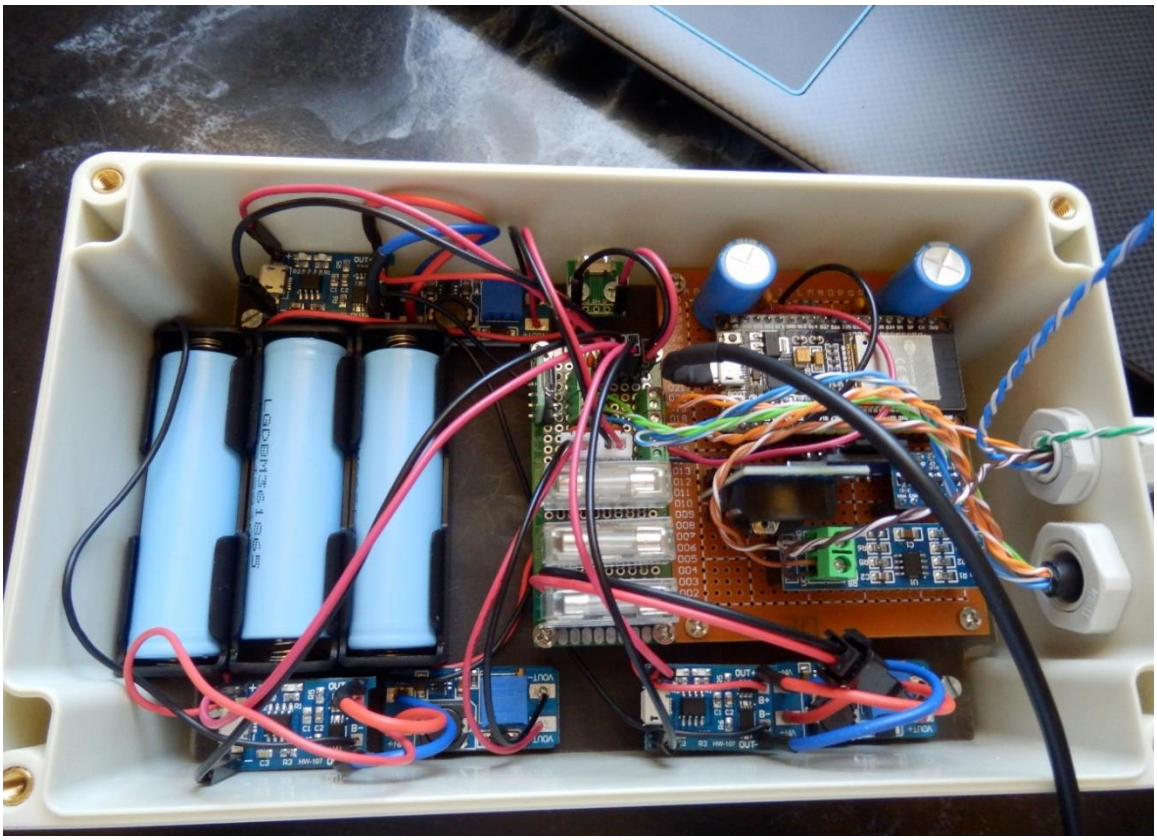
```
void loop()
{
  //volání jednotlivých metod instance htu21d a bmp180 pomocí
  //kterých jsou získány požadované hodnoty přes I2C
  temperature = htu21d.readTemperature();
  humidity = htu21d.readHumidity();
  absPressure = bmp180.readPressure() / 100;
  //přepočet absolutního tlaku a zaokrouhlení na 2 desetinná místa
  pressure = round(((absPressure * 9.80665 * 231) / (287 * (273 +
          temperature + (231 / 400))) + absPressure) * 100) / 100;
  //metoda pro sestavení zprávy a výpočet crc (data1,data2,data3)
  slaveid2.compile_message(temperature, humidity, pressure, 0.00);
  //hlídá požadavky z jednotky master a podle nich řídí odesílání
  slaveid2.check_send();
}
```

Obrázek 21: *Hlavní část zdrojového kódu senzorové jednotky*

6.2 Hlavní jednotka

Zařízení hlavní jednotky je vzhledem ke složitějším operacím nad daty konfigurováno jako klientské zařízení, které předává data na webový server k dalšímu zpracování. Srdcem hlavní jednotky meteostanice je vývojová deska s mikrokontrolerem ESP32.

Pro komunikaci se senzorovými jednotkami je přidán modul RS485 napojený na rozhraní UART. Přes rozhraní SPI je připojená microSD karta, na kterou jsou průběžně ukládána naměřená meteorologická data. Zároveň poskytuje prostor pro dočasné úložiště dat při výpadku spojení se serverem. Po obnovení spojení jsou data nahrány zpět na server. Pokud by došlo k jakékoliv chybě při nahrávání, je možné microSD kartu vyjmout a data nahrát manuálně. O přesný čas naměřených údajů se stará časová jednotka RTC, připojená k rozhraní I^2C, která je pravidelně synchronizována časovým serverem pomocí protokolu NTP. Komunikaci pro přenos dat mezi hlavní jednotkou a sítí internet zprostředkovává zařízení mikrokontroleru ESP32 pro bezdrátové připojení WiFi. Veškerý hardware je řešený modulově, aby byla v případě poruchy možná rychlá výměna. Patice pro moduly jsou zaletovány v univerzálním plošném spoji, který je spolu se zařízením záložního zdrojem a pojistkovým modulem zabudován v elektroinstalační krabici o rozměrech 200x120x75 (viz obrázek 22).



Obrázek 22: *Kompletní zařízení hlavní jednotky*

6.2.1 **Pevné úložiště meteostanice**

Pevné úložiště meteostanice je tvořeno kartou microSD a čtečkou připojenou přes rozhraní SPI mikrokontroleru ESP32. Záloha naměřených dat pro dlouhodobé statistiky je ukládána do souboru, pojmenovaném MAINW.CSV. Soubor je uložen v souborovém systému FAT32, na kartě microSD. Při vkládání dlouhodobých dat do databáze, které probíhá v intervalu třiceti minut, jsou zároveň tytéž data zapsána do souboru. V případě chybějících dlouhodobých dat v databázi je možné vyjmout kartu microSD a pomocí souboru MAIN.CSV provést import, přičemž jsou zapsány pouze data, která neobsahují duplicitní klíč. Než začne přenos dat, je mikrokontrolerem ESP32 ověřena dostupnost spojení mezi serverem a databází. V rámci dostupných prostředků bylo odzkoušeno, že chyby způsobující absenci některých záznamů v databázi, vznikají nahodile maximálně jednou za 24 hodin. Další chybějící záznamy mohou vznikat například výpadkem routeru, nebo chybami na straně poskytovatele internetového připojení. Pro tyto chyby byla vytvořena dočasná záloha v podobě souboru TEMPW.TXT uložena na kartě microSD. Pokud dojde k výpadku komunikace s databází a není přijata zpráva o doručení, je záznam uložen jako textový řetězec jednoho řádku v souboru TEMPW.TXT. Po obnovení komunikace jsou jednotlivé záznamy ze souboru načítány a nahrávány do databáze. Zároveň je opět prováděna kontrola doručení těchto záznamů a pokud není záznam doručen, je uchován na kartě microSD a při další příležitosti znovu odeslán do databáze.

6.2.2 **Jednotka pro práci s časem**

Databáze umožňuje opatřit každý vložený záznam časovým razítkem, ale v případě tohoto řešení by vznikl problém s identifikací neúspěšně nahraných záznamu při zápisu na kartu microSD. Pokud by byly záznamy dodatečně nahrávány do databáze, probíhalo by přidělování jiného časového razítka oproti skutečnosti, kdy byl záznam pořízen. Navíc by bylo nutné nějakým způsobem řešit přesnost intervalu jednotlivých záznamů v rámci zařízení hlavní jednotky.

Proto byla vyzkoušena funkce mikrokontroleru ESP32 pro měření a práci s časem. ESP32 po připojení k internetu pomocí protokolu NTP načte údaj o aktuálním čase a ten pak běží v samotném mikrokontroleru až do odpojení napájení, nebo restartu. Aktuální hodnota je vrácena zavoláním funkce pro výpis aktuálního času. Pak už stačí v určitých intervalech synchronizovat čas pomocí protokolu NTP.

Problém u předchozího řešení nastal ve chvíli, kdy byl použit režim hlubokého spánku pro snížení spotřeby. Režim ovlivňuje chod jednotky RTC obsažené v mikrokontroleru ESP32 a dochází k předcházení času. To by znamenalo po každém probuzení synchronizovat čas, což v situaci, kdy není dostupná síť internet není možné.

Řešení výše uvedených problémů spočívalo v přidání samostatného modulu RTC DS3231, který udržuje aktuální čas, který je přes sériové rozhraní I^2C načten mikrokontrolerem ESP32 z registru DS3231. Vzhledem ke spotřebě 500nA na záložní baterii v režimu nečinnosti, byla zvolena klasická baterie CR2032, za předpokladu vyřazení funkce dobíjecího obvodu. Životnost baterie je při tomto použití minimálně 4 roky. Modul je velmi přesný, nepřesnost je podle datového listu výrobce ± 2ppm, což je cca 63 sec za rok. Synchronizace probíhá jednou denně, takže i při neúspěšných pokusech několik dní po sobě, dojde k rozdílu maximálně jedné sekundy. Každý záznam tak může být opatřen časovým razítkem, které slouží pro jednoznačnou identifikaci v rámci pevného úložiště hlavní jednotky a zároveň funguje jako primární klíč záznamu v databázi. Nakonec byla přidána vlastní metoda, která ověří platnost letního, respektive zimního času a provede požadovaný posun hodin.

6.3 Záloha napájení

Každá část meteostanice má svou vlastní jednotku UPS pro zálohu napájení v případě přerušení hlavního zdroje energie, který je tvořen síťovým adaptérem 5V/4,5A. Jednotka UPS zajišťuje plynulý přechod na rezervní zdroj, tak aby nedošlo k restartování zařízení vlivem poklesu napětí. Napětí na výstupu z jednotky UPS je 3,3 V, proto je za každou jednotkou přidán modul pro zvýšení napětí na požadovanou úroveň 5 V.

Průměrný naměřený odběr každé senzorové jednotky je 100 mW. Kapacita pro instalovaný akumulátor 3600mAh LG INR18650-M36 je dle datového listu výrobce 12500 mWh. To znamená, že vypočítaná výdrž záložního zdroje by měla být přibližně 125 hodin.

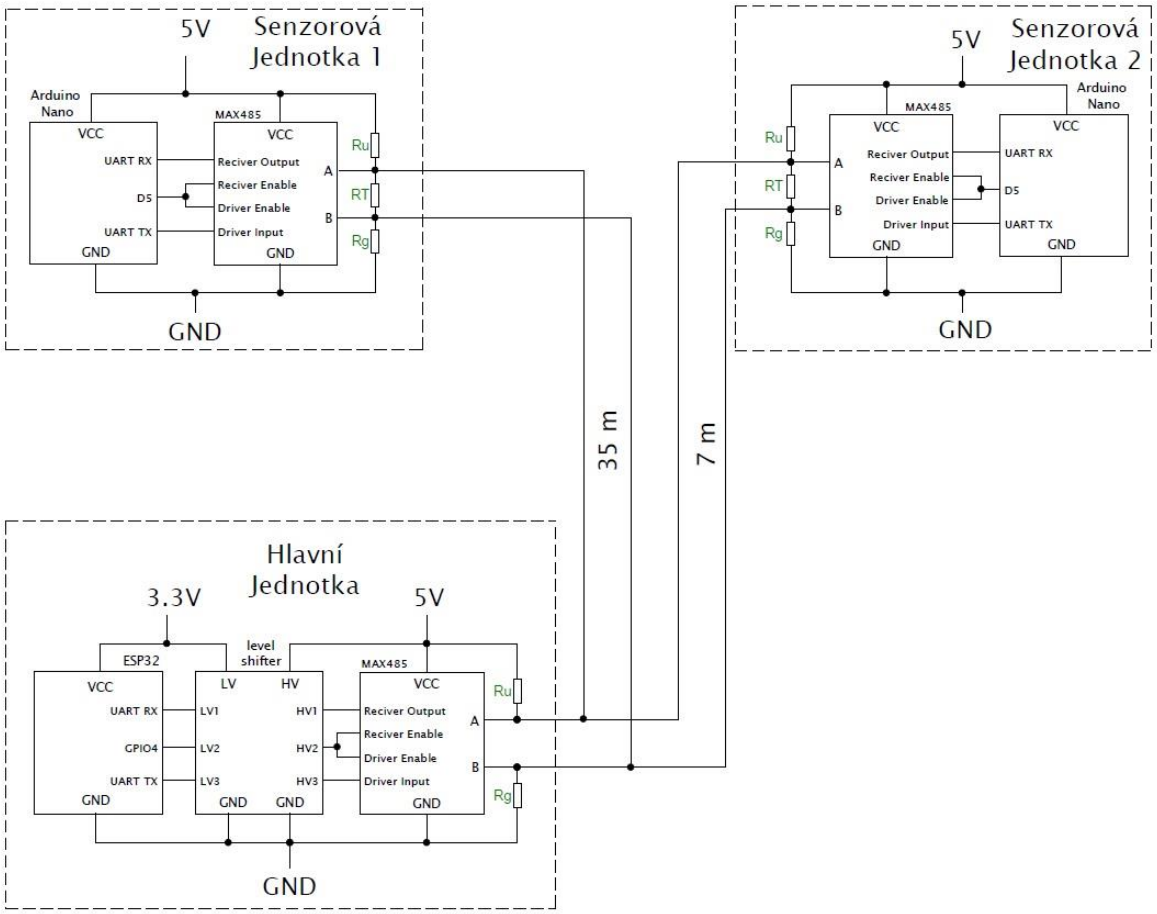
Pro hlavní jednotku byl naměřen a vypočítán průměrný odběr proudu při běžném režimu s připojením k WiFi cca 120 mA, což by při napájení 5 V znamenalo výdrž pouze 6 hodin. Z těchto důvodů byl využit režim snížené spotřeby, který umožní mikrokontroleru ESP32 přejít při nečinnosti do režimu hlubokého spánku s odběrem v řádech jednotek miliampérů. Vypínáním napájení převodníku RS485 a jednotky RTC (obsahuje záložní baterii) při režimu hlubokého spánku bylo dosaženo další úspory energie. Všechny uvedené úpravy

vedly ke snížení spotřeby v režimu spánku na 4 mA. Průměrná spotřeba hlavní jednotky byla vypočítána na 23,3 mA, což je přibližně 107 hodin provozu.

Při praktické zkoušce provozu celého zařízení meteostanice byla ověřena výdrž akumulátoru hlavní jednotky na 112 hodin. Akumulátory senzorových jednotek po této době stále vykazovaly zbylou kapacitu pro udržení zařízení v provozu.

6.4 Sériová komunikace

Fyzické propojení je realizováno krouceným párem vodičů ethernetového kabelu kategorie 5e. Kabel je vzhledem k požadavkům a maximální použité vzdálenosti naprosto dostačující, což bylo ověřeno čtyřměsíčním zkušebním provozem. Koncové uzly RS485 jsou z důvodu odrazů zakončeny terminačním rezistorem RT o velikosti odporu 120 ohmů. Klidový stav linky je zajištěn rezistory připojenými ke kladnému a zápornému potenciálu napájení. Tyto



Obrázek 23: *Celkové schéma zapojení komunikace pro RS485*

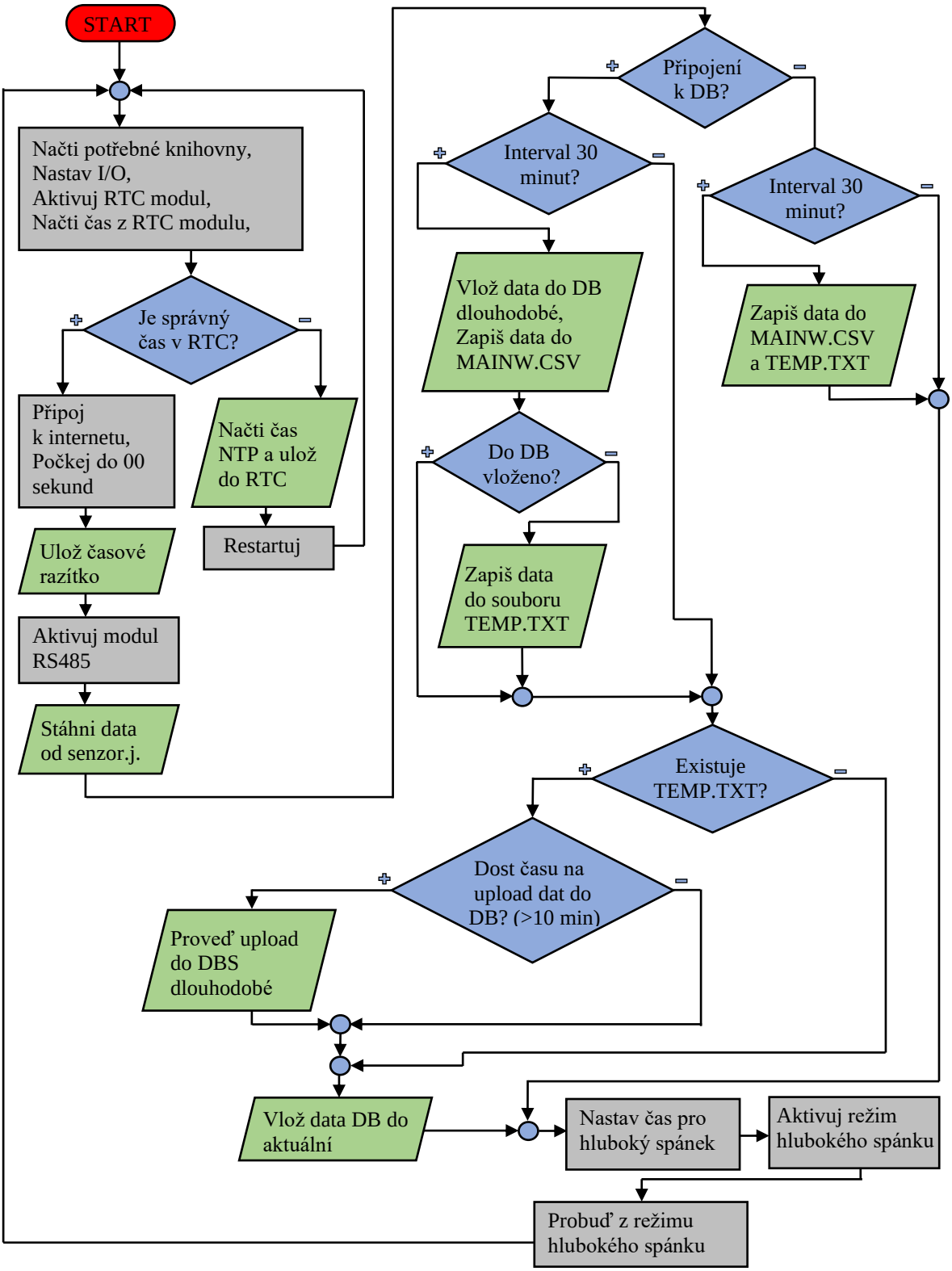
a terminační rezistory jsou součástí zakoupeného modulu RS485. Pokud není modul použit v koncovém uzlu, je nutné terminační rezistor odstranit. Celkové schéma zapojení komunikace je znázorněno na obrázku 23.

Pro přenos dat je využit vlastní komunikační protokol, uložený ve dvou knihovnách. Jedna je určena pro hlavní jednotku a druhá pro senzorové jednotky. Komunikační protokol byl naprogramován s ohledem na následující požadavky, jako jsou ověření správnosti doručených dat kontrolním součtem, odeslání dat ze senzorových jednotek dle požadavku hlavní jednotky, adresace senzorových jednotek připojených na společné lince dle id, schopnost komunikovat pouze s jednou senzorovou jednotkou při výpadku druhé, potvrzení od hlavní jednotky o úspěšném doručení dat, opakovaná žádost hlavní jednotky při neúspěšném získání dat ze senzorové jednotky, nejmenší možné zpoždění při komunikaci jednotlivých uzlů, vytvoření chybových hlášení pro diagnostické účely, ošetření neznámých adres a další funkce, které chrání komunikaci před zacyklením.

Požadavky od hlavní jednotky jsou každou minutu odesílány senzorovým jednotkám. Data získaná ze senzorů jsou nejprve zpracována MCU Arduino Nano a ve chvíli obdržení odpovídajícího požadavku, odeslána hlavní jednotce. Jestliže senzorová jednotka obdrží potvrzení o úspěšném doručení, začne načítat a zpracovávat data pro další požadavek. Aby v případě neúspěšného doručení nedocházelo k držení zastaralých dat, je maximální čekací doba senzorové jednotky nastavena na 2 minuty. Po uplynutí této doby jsou stávající data zahozena a dochází k novému zpracování naměřených hodnot. Výjimku tvoří pouze srážkový úhrn, který je držen až do skutečného doručení. Hardwarové rozhraní komunikace včetně komunikačního protokolu vykazuje vysokou spolehlivost, což bylo ověřeno nepřetržitým fungováním po dobu jednoho měsíce, aniž by došlo k jediné chybě.

6.5 Návrh zdrojového kódu

Návrh probíhal nejprve od části pro připojení k internetu, poté bylo třeba vyřešit komunikaci jednotlivých jednotek mezi sebou a vyzkoušet přenos dat. Dále byl vytvořen a odzkoušen kód pro vkládání záznamů do databáze, přičemž pro vyzkoušení musely být naprogramovány a připraveny součásti na serverové straně (viz 6.6 a 6.7). Následovala část pro zálohu dat na microSD kartu a kód pro použití režimu hlubokého spánku. Poslední část vznikla po přidání samostatného modulu RTC (viz kap. 6.2.2) a obsahuje kód pro práci s časem. Návrh kódu pro hlavní jednotku je na následujícím obrázku.

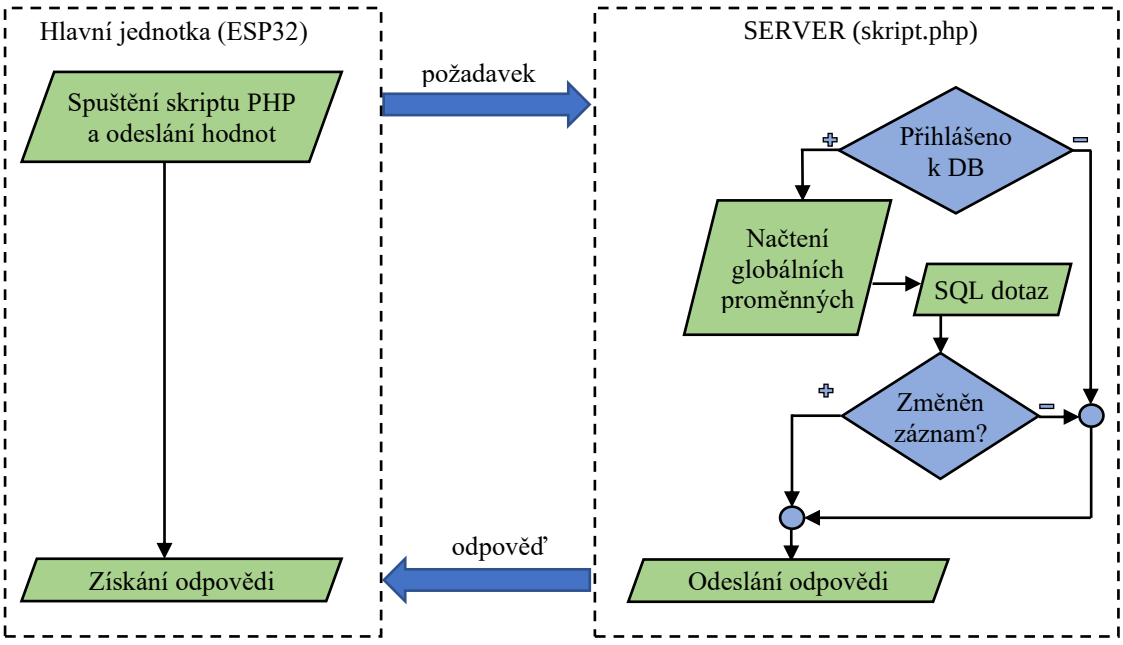


Obrázek 24: *Diagram zdrojového kódu hlavní jednotky*

6.6 Komunikace s databází

Vkládání dat do databáze MySQL je z bezpečnostních důvodů zprostředkováno výhradně přes PHP skript, který je uložen na webovém serveru. Skript obsahuje přihlašovací údaje do databáze a kód pro požadované operace. Pro meteostanici jsou vytvořeny dva skripty, jeden pro ukládání do tabulky dlouhodobých meteorologických dat a druhý, který ukládá záznamy v intervalu jedné minuty, přičemž jeho kód udržuje v tabulce aktuálních meteorologických dat pouze posledních třicet záznamů.

Kód mikrokontroleru ESP32 pomocí metod begin a POST třídy „HTTPClient“, spustí skript „insert.php“, nebo „insert_act.php“ a předá hodnoty proměnných. Skript na serveru nejprve ověří přihlašovací údaje a pokud jsou platné, provede připojení k databázi. Následuje provedení SQL dotazu pro vložení záznamu s naměřenými daty. Hodnoty proměnných jsou předány hlavičkou http protokolu do skriptu PHP jako globální proměnné, které jsou vloženy do dotazu nepřímo, pomocí metod „prepare“ a „bind_param“. Přímé vložení proměnných do dotazu je z důvodu možného útoku „sql injection“ nežádoucí. Dotaz SQL je proveden metodou „execute“, a pak následuje metoda „mysqli_affected_rows“, která vrátí počet změněných řádků v databázi.



Obrázek 25: *Komunikace mezi serverem a MCU*

Vrácený počet je porovnán v podmínce vypisující do webové stránky text o úspěchu, nebo neúspěchu. Metoda „getString“ třídy „HTTPClient“ v kódu MCU ESP32 tento text získá, následně porovná a vyhodnotí jako výsledek akce zápisu záznamu do databáze. Vzhledem k výše uvedenému je vždy zaručeno spolehlivé předání dat a podle odezvy může mikrokontroler řídit činnost při ukládání záznamů do databáze, nebo do dočasného pevného úložiště zařízení hlavní jednotky. Princip komunikace zachycuje obrázek 25.

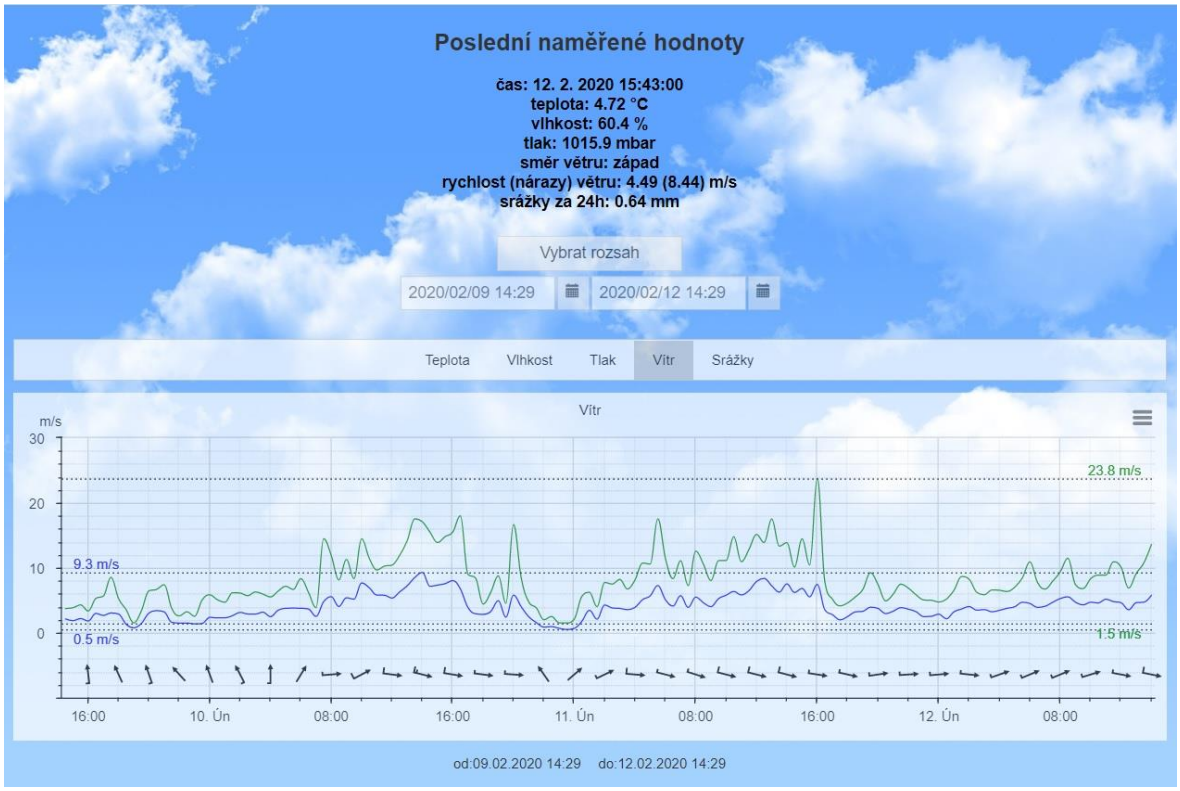
Získání dat z databáze je provedeno obdobně. Skript PHP po ověření platnosti přihlašovacích údajů načte hodnoty proměnných, které jsou předány metodou POST z webové stránky meteostanice. Proměnné představují meze hledaného časového intervalu. Následně je sestaven a spuštěn dotaz SQL, který získá požadovaná data a předá je pomocí struktury JSON.

6.7 Webové rozhraní

Pro zpracování dat a jejich prezentaci je vybrána základní varianta webhostingu od firmy Onebit.cz s databázovým systémem MySQL s podporou jazyka PHP, úložištěm pro soubory webu a nástroji pro správu. Naměřená data jsou vizualizována prostřednictvím webové stránky http://arduinopj01.cz. Pro každou veličinu je k dispozici graf zachycující její vývoj v půlhodinových intervalech. Aby nedocházelo k neustálému obnovování celé stránky je poslední zaznamenaný stav měřených meteorologických hodnot aktualizován asynchronní metodou AJAX. Výběr požadovaného rozsahu je možné realizovat pomocí formulářového pole kalendáře pro počáteční a koncové datum a tlačítka potvrzující výběr. Jednotlivé grafy jsou umístěny v záložkách, které jsou sdruženy do jednoho kontejneru. Vybraný rozsah je možné přiblížit v rámci časové osy, vyexportovat do grafického souboru *.png, *.jpeg, *.pdf a *.svg, nebo zobrazit na celou obrazovku a vytisknout. Graf srážek zobrazuje srážkový úhrn v rámci konkrétního dne, od 00:00 do 23:59. Záložka grafu „Vítr“ obsahuje graf průměrné rychlosti, nárazové rychlosti a směru větru, který je znázorněn natočenými šipkami. U každého grafu dochází k zobrazení maxima a minima ve vybraném časovém rozsahu. Zobrazení maxima a minima se aktualizuje i po přiblížení grafu.

Pro vykreslování grafů a většinu funkcí jsou použity frameworky javascriptu. Grafy jsou vykreslovány pomocí frameworků od highcharts, stejně tak i funkce s nimi související. Převody časových formátů mezi webovou stránkou a databází jsou realizovány pomocí knihovny timeClass. Formulářové pole pro datum a čas v HTML5 bohužel nefungovalo

v prohlížeči Safari (v době psaní této práce) a pro jeho vytvoření bylo nutné použít frameworky bootstrap. Snímek obrazovky webového rozhraní meteostanice je na následujícím obrázku.



Obrázek 26: *Webová stránka meteostanice*

7 ZÁVĚR

Stanovené cíle byly naplněny. Výsledkem bakalářské práce je teoretická charakteristika jednotlivých částí meteorologické stanice. Jsou představeny jednotlivé komponenty, které jsou nezbytné k vytvoření tohoto zařízení. V práci je vysvětlena teorie fungování mikrokontrolerů použitých při vytváření zařízení meteostanice a navazujících periférií nezbytných pro sběr dat. Použité moduly jsou pomyslně rozebrány na jednotlivé části, které jsou dále podrobně charakterizovány. Součástí teoretické části je popis vrstev webového rozhraní v rámci sdíleného webhostingu a jejich funkcionalita. Pro názornost popisované technologie zařízení je textová část doplněna množstvím obrázků.

Přínosem pro danou problematiku je praktický příklad návrhu meteorologické stanice, sběru, analýzy a vizualizace nasbíraných dat, společně s vytvořením vlastního komunikačního protokolu pro RS485. Tato část představuje tvorbu projektu zařízení, využívajícího mikrokontrolerů platformy Arduino, kterou lze specifikovat do obecných bodů postupu. Pro zařízení jsou vybrány součásti, které svými vlastnostmi odpovídají daným požadavkům. Sestavování vybraných součástí je doplněno o schémata zapojení a fotodokumentaci. Na základě hardwarové sestavy je navržen a implementován zdrojový kód. V návaznosti je založena databáze pro ukládání jednotlivých záznamů. Následně je celý systém vizualizace dat vytvořen pomocí programových prostředků jazyku PHP, HTML a javaScript. Požadované funkce v navrženém systému jsou odzkoušeny dlouhodobým testováním a spolehlivým provozem.

8 SEZNAM POUŽITÝCH ZDROJŮ

[1] Z. V. &. T. H. Kitchen, Průvodce světem Arduina, Bučovice: Nakladatelství Martin Stříž, 2017.

[2] E. Systems, „the INTERNET of THIHGS with ESP32," [Online]. Available: http://esp32.net/. [Přístup získán 10 11 2019].

[3] Random Nerd Tutorials, „ESP32 Pinout Reference: Which GPIO pins should you use?," 2 10 2019. [Online]. Available: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/. [Přístup získán 10 11 2019].

[4] J. Pinker, Mikroprocesory a mikropočítače, Praha: BEN - technická literatura, 2011.

[5] M. Malý, Hradla, volty, jednočipy, Praha: CZ.NIC, z.s.p.o., 2017, pp. 329-332.

[6] S. Laboratories, „Sparkfun Start Something," [Online]. Available: https://learn.sparkfun.com/tutorials/cp2102-usb-to-serial-converter-hook-up-guide/all. [Přístup získán 10 11 2019].

[7] K. Javůrek, „Historie flash paměti s bleskovým čtením i zápisem," 16 04 2012. [Online]. Available: https://www.zive.cz/clanky/historie-flash-pameti-s-bleskovym-ctenim-i-zapisem/sc-3-a-163269/default.aspx. [Přístup získán 12 11 2019].

[8] Patrick Twele, Rutronik, „Není NAND jako NAND," 06 2017. [Online]. Available: https://www.dps-az.cz/soucastky/id:53616/neni-nand-jako-nand. [Přístup získán 12 11 2019].

[9] Redakce, Digi-Key, „Výběr a použití správné technologie paměti Flash pro aplikace IoT," 11 06 2018. [Online]. Available: https://vyvoj.hw.cz/vyber-a-pouziti-spravne-technologie-pameti-flash-pro-aplikace-iot.html. [Přístup získán 12 11 2019].

[10] M. Dudka, „I2C – Relativně jednoduše," 12 11 2016. [Online]. Available: http://www.tajned.cz/2016/10/i2c-relativne-jednoduse/. [Přístup získán 13 11 2019].

[11] Maxim Integrated, „Extremely Accurate I²C-Integrated RTC/TCXO/Crystal," [Online]. Available: https://www.maximintegrated.com/en/products/analog/real-time-clocks/DS3231.html. [Přístup získán 15 11 2019].

[12] V. Slinták, „Konverze mezi 5V a 3,3V logikou," 5 10 2011. [Online]. Available: https://uart.cz/253/konverze-mezi-5v-a-3v-logikou/. [Přístup získán 15 11 2019].

[13] B. Benchoff, „MICROCHIP TO ACQUIRE ATMEL FOR $3.56 BILLION," 20 01 2016. [Online]. Available: https://hackaday.com/2016/01/20/microchip-to-acquire-atmel-for-3-56-billion/. [Přístup získán 19 11 2019].

[14] P. Poucha, „Komunikace pro průmyslových linkách RS485 a RS422," [Online]. Available: https://papouch.com/komunikace-pro-prumyslovych-linkach-rs485-a-rs422-p3735/. [Přístup získán 18 11 2019].

[15] J. Ř. Jan Staněk, „RS 485 & 422," 15 01 1998. [Online]. Available: https://vyvoj.hw.cz/teorie-a-praxe/dokumentace/rs-485-422.html. [Přístup získán 18 11 2019].

[16] SOS electronic, „Jak zabezpečit stabilní RS485 komunikaci?," SOS electronic, 07 07 2016. [Online]. Available: https://www.soselectronic.cz/articles/linear-technology/linear-technology-jak-si-zabazpecit-stabilni-rs485-komunikaci-1769. [Přístup získán 19 11 2019].

[17] TE Conectivity, „DIGITAL HIGH ACCURACY RH/T SENSOR,“ [Online]. Available: https://www.te.com/global-en/product-CAT-HSC0004.html. [Přístup získán 21 11 2019].

[18] Bosch Sensortec, „BMP180,“ [Online]. Available: https://www.bosch-sensortec.com/bst/products/all_products/bmp180. [Přístup získán 21 11 2019].

[19] K. Ježková, „SEZNAMTE SE S ARDUINO VÝVOJOVÝM PROSTŘEDÍM,“ 14 08 2014. [Online]. Available: https://arduino.cz/seznamte-s-arduino-vyvojovym-prostredim/. [Přístup získán 24 11 2019].

[20] Arduino, „Libraries,“ [Online]. Available: https://www.arduino.cc/en/reference/libraries. [Přístup získán 30 11 2019].

[21] M. Virius, Programování v C++, Praha: Grada Publishing, 2018.

[22] Onebit, „Webhosting,“ [Online]. Available: https://www.onebit.cz/cz/webhosting/. [Přístup získán 30 11 2019].

[23] M. Chan, „FTP, FTPS, and SFTP – what are the differences?,“ 16 07 2019. [Online]. Available: https://www.thorntech.com/2019/07/ftp-ftps-sftp-differences/. [Přístup získán 27 11 2019].

[24] M. Kozák, „MySQL pro zelenáče,“ 13 02 2008. [Online]. Available: https://www.linuxexpres.cz/praxe/mysql-pro-zelenace. [Přístup získán 27 11 2019].

[25] MySQL, „CHECK Constraints,“ [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/create-table-check-constraints.html. [Přístup získán 27 11 2019].

[26] P. Lutus, „MySQL Tutorial 1: Overview, Tables, Queries,“ 2012. [Online]. Available: https://arachnoid.com/MySQL/. [Přístup získán 30 11 2019].

[27] M. Š. Zdeněk Moravec, „PHP a MySQL – MySQLi – 1. díl,“ 14 01 2010. [Online]. Available: http://programujte.com/clanek/2009103100-php-a-mysql-mysqli-1-dil/. [Přístup získán 30 11 2019].

[28] Z. L. Zdeněk Moravec, „PHP a MySQL – MySQLi - 2. díl,“ 01 06 2010. [Online]. Available: http://programujte.com/clanek/2010030700-php-a-mysql-mysqli-2-dil/. [Přístup získán 30 11 2019].

[29] S. Morris, „AJAX—What it id, how it works, and what it’s used for,“ skillcrush, 26 04 2018. [Online]. Available: https://skillcrush.com/2018/04/26/what-is-ajax/. [Přístup získán 30 11 2019].

[30] J. Sridhar, „What Is JavaScript and How Does It Work?,“ 2 10 2017. [Online]. Available: https://www.makeuseof.com/tag/what-is-javascript/. [Přístup získán 30 11 2019].

[31] P. Dhaker, „analog-dialogue,“ [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html#. [Přístup získán 11 11 2019].

[32] E. Williams, „Bye-bye ATmega328P, Hello 328PB!,“ radiolocman, 15 09 2016. [Online]. Available: https://www.radiolocman.com/review/article.html?di=162667. [Přístup získán 19 11 2019].

[33] M. Olejár, „Stručný popis sběrnice I2C a její praktické využití k připojení externí eeprom 24LC256 k mikrokontroléru PIC16F877,“ vyvoj.hw.cz, 20 05 2000. [Online]. Available: https://vyvoj.hw.cz/navrh-obvodu/strucny-popis-sbernice-i2c-a-jeji-prakticke-vyuziti-k-pripojeni-externi-eeprom-24lc256. [Přístup získán 13 11 2019].

[34] Sparkfun, „Weather Meters,“ [Online]. Available: https://www.bc-robotics.com/shop/anemometer-wind-speed-sensor-2/. [Přístup získán 21 11 2019].

[35] WeatherShack.com, „Tipping Bucket Rain Gauge,“ [Online]. Available: https://www.weathershack.com/static/ed-tipping-bucket-rain-gauge.html. [Přístup získán 22 11 2019].

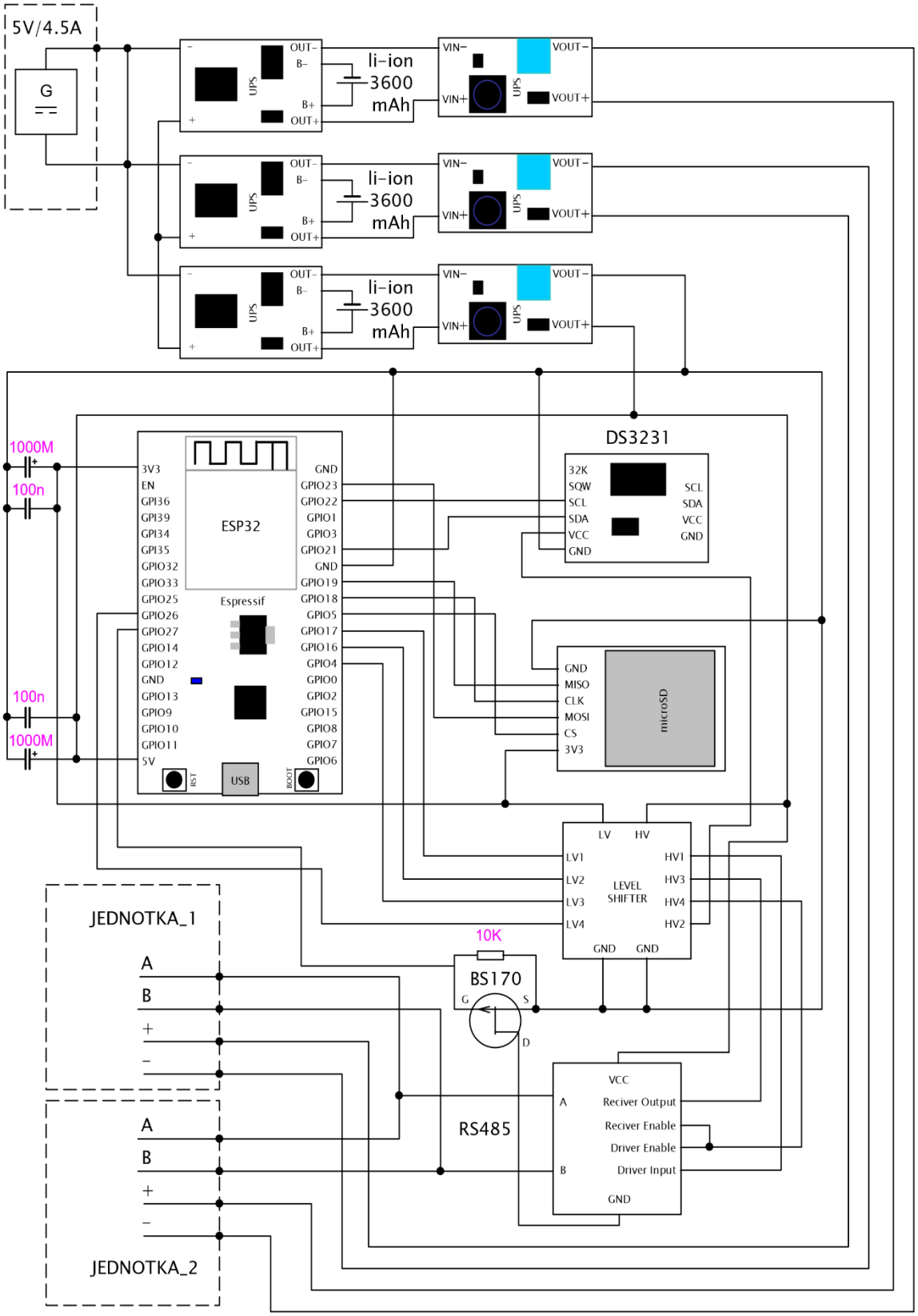
9 PŘÍLOHY

A SCHÉMATA ZAPOJENÍ

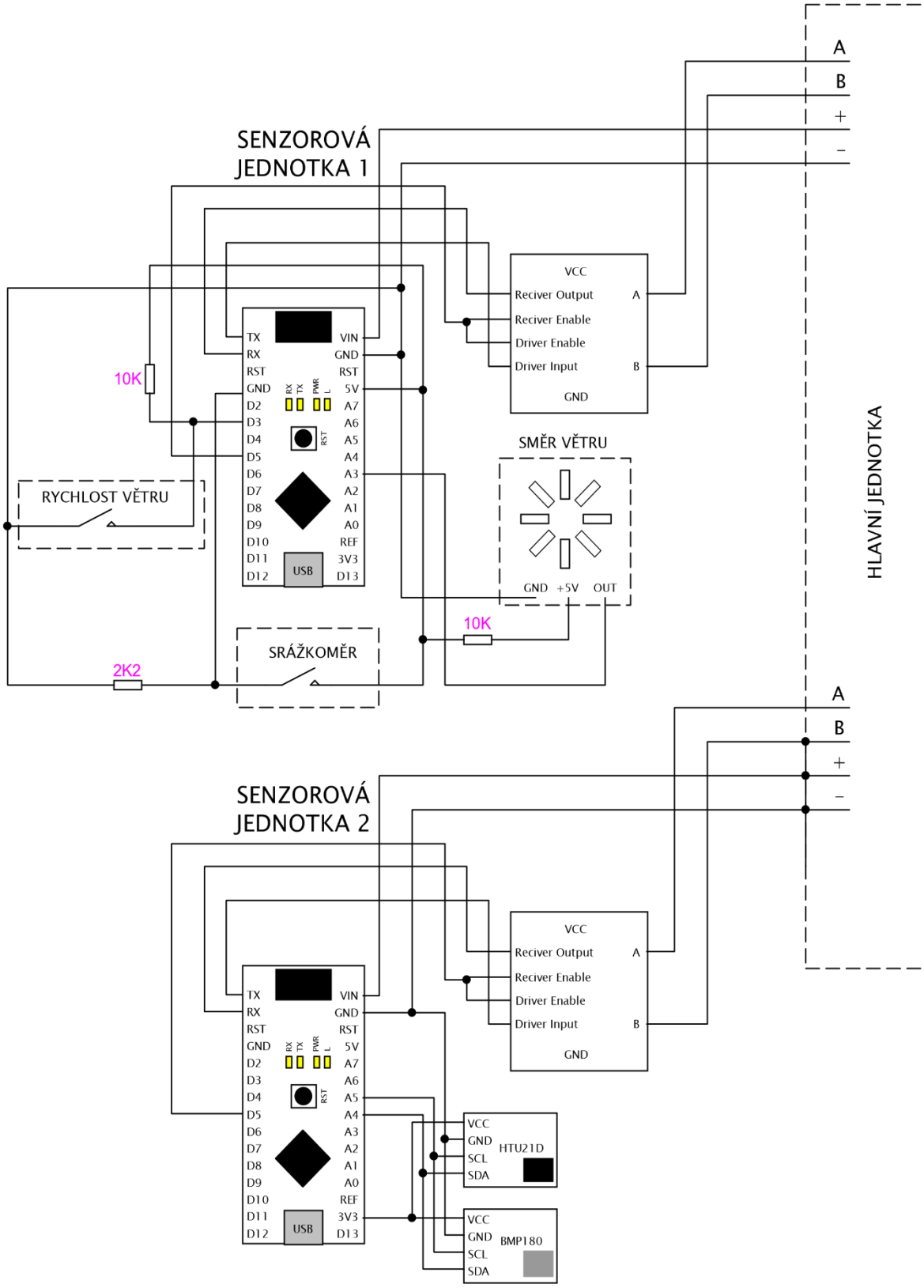
B UKÁZKY ZDROJOVÝCH KÓDŮ

C SNÍMKY OBRAZOVEK

A SCHÉMATA ZAPOJENÍ



Obrázek A.1: *Schéma zapojení hlavní jednotky meteostanice.*



Obrázek A.2: *Schéma zapojení senzorových jednotek meteostanice.*

B UKÁZKY ZDROJOVÝCH KÓDŮ

```
10  ////////////////////////////////////////////////////////// PRO ZJIŠTĚNÍ ČETNOSTI ////////////////////////////////////////////////////////////
11  //parametry (pole, počet položek)
12  float Stat::search_frequency(float* arrayValues, int *countValues)
13  {
14      //počet předchozích nalezených hodnot
15      int previousCountFound = 0;
16      //počet aktuálně nalezených hodnot
17      int countFound = 0;
18      //pro dočasně nalezenou hodnotu
19      float floatTempFound = 0;
20      //pro nalezenou hodnotu
21      float floatFound = 0;
22      //veme aktuální hodnotu a porovnává s ostatními v poli
23      for (int i = 0; i <= *countValues; i++)
24      {
25          //pro uložení aktuálně vybraného čísla, které je určeno k porovnávání s ostatními hodnotami
26          float actualValue = arrayValues[i];
27          //vynuluje počet nalezených opakujících se čísel
28          countFound = 0;
29          //porovná aktuálně vybranou hodnotu s ostatními
30          for (int j = 0; j <= *countValues; j++)
31          {
32              //jestliže se hodnota opakuje
33              if (actualValue == arrayValues[j])
34              {
35                  //je zvýšen počet opakování aktuálně vybrané hodnoty o 1
36                  countFound++;
37                  //a tato hodnota je uložena do dočasné proměnné
38                  floatTempFound = arrayValues[j];
39              }
40          }
41
42          //jesliže je počet nalezených hodnot vyšší než předchozí max. počet nalezených hodnot, pak je
43          if (countFound > previousCountFound)
44          {
45              //do proměnné "floatFound" uložena nalezená hodnota
46              floatFound = floatTempFound;
47              //a počet opakování této hodnoty je uložen do proměnné "previousCountFound"
48              previousCountFound = countFound;
49          }
50      }
51      return floatFound;
52  }
53
```

Obrázek B.1: *Zdrojový kód funkce, která vrací hodnotu dle četnosti.*

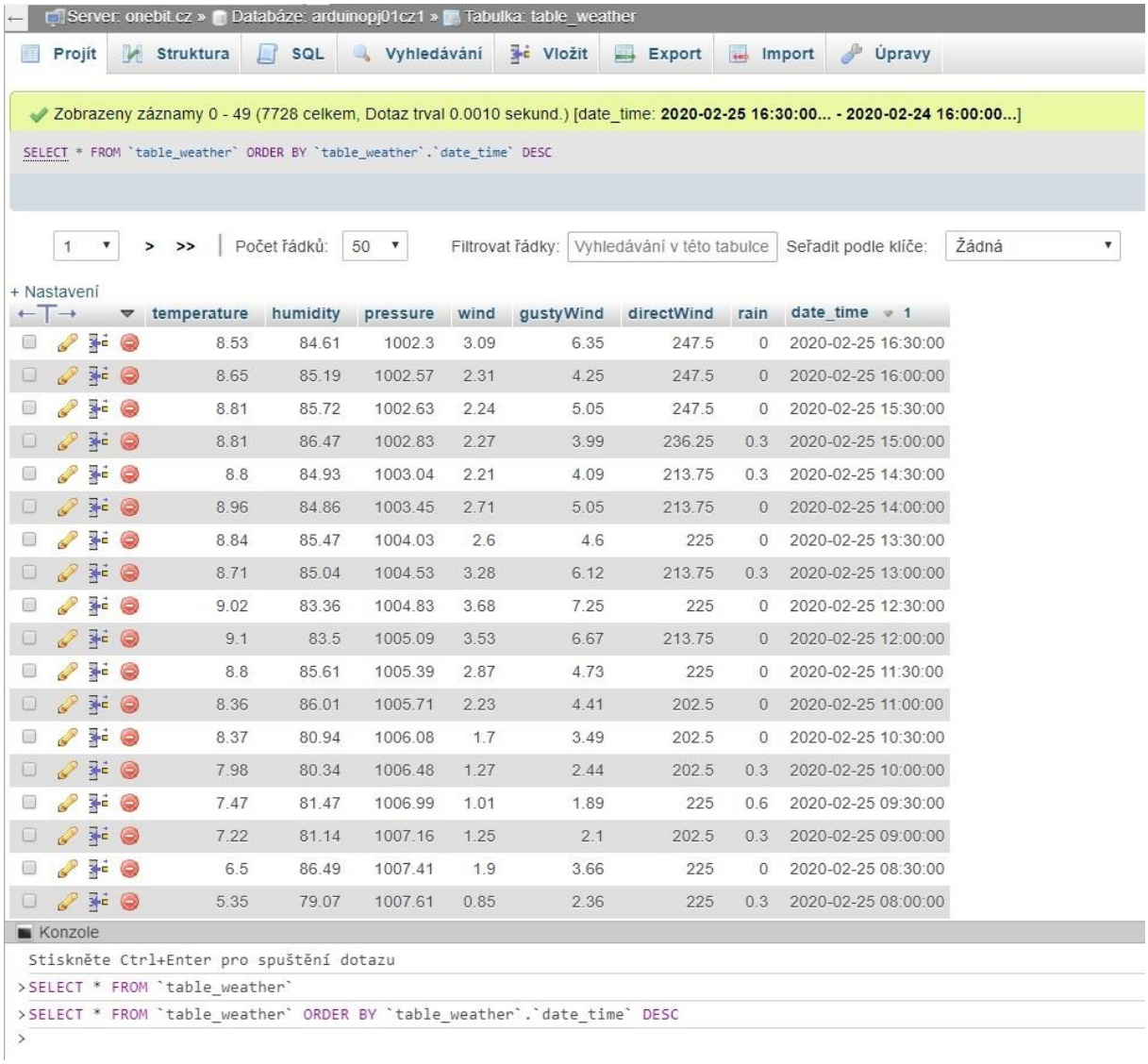
```
55  ////////////////////////////////////////////////////////////// PRO ZJIŠTĚNÍ PRŮMĚRU ////////////////////////////////////////////////////////////
56  float Stat::average_value(float* arrayVal, int *countVal)
57  {
58      float sum = 0;
59      for (int i = 0; i <= *countVal; i++)
60      {
61          sum += arrayVal[i];
62      }
63      float result = sum / (*countVal + 1);
64      return round(result * 1000) / 1000.0;
65  }
66
67
68  ////////////////////////////////////////// PRO NALEZENÍ NEJVYŠŠÍ HODNOTY Z POSLEDNÍCH NAMĚŘENÝCH ////////////////////////////////////////////
69  float Stat::highest_value(float* arrayVal, int *countVal)
70  {
71      float maxValue = arrayVal[0];
72      for (int i = 0; i <= *countVal; i++)
73      {
74          if (arrayVal[i] > maxValue)
75          {
76              maxValue = arrayVal[i];
77          }
78      }
79      return round(maxValue * 1000) / 1000.0;
80  }
```

Obrázek B.2: *Zdrojový kód funkcí pro průměr a nalezení maxima.*

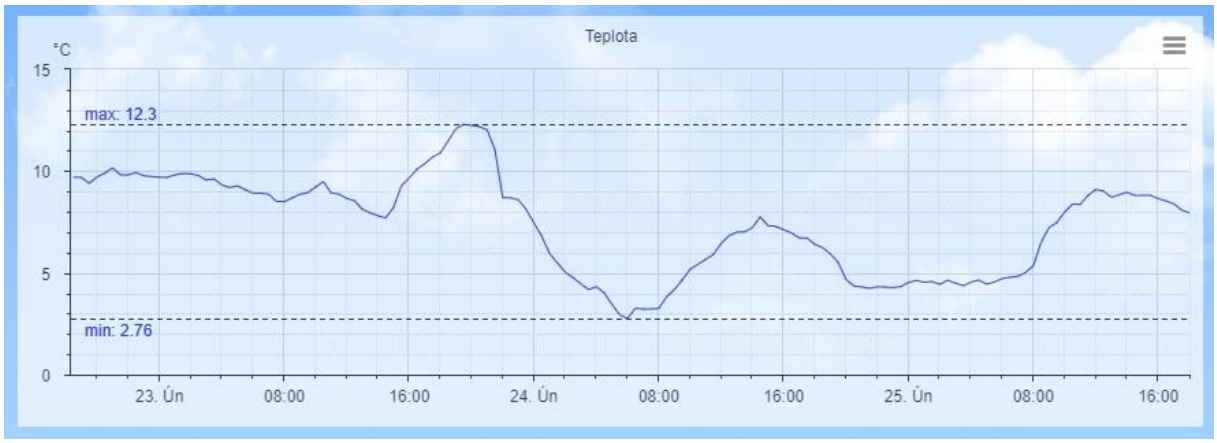
```
104   //hlídá požadavky a spouští odesílání
105   void MySlave::check_send()
106   {
107       bool recivedOK = false;
108       bool sendOK = false;
109       //vymazání případných fragmentů v příjmacím bufferu
110       clear_buffer();
111       unsigned long check_send_timer = millis();
112       //následuje čekání na úspěšné odeslání dat do jednotky master
113       while (!sendOK)
114       {
115           //naměřená data můžou být stará max 120s
116           if (millis() - check_send_timer > 120000)
117           {
118               sent = false;
119               break;
120           }
121           //jestliže se něco načte
122           if (Serial.available() > 0)
123           {
124               //při správném požadavku odešle na jednotku master
125               if (Serial.peek() == slaveAdd)
126               {
127                   report->println("received correct request\n");
128                   //spustí odesílání
129                   send_data();
130                   //čekání na zpracování dat v J.M. (poté následuje potvrzení)
131                   delay(15);
132               }
133               //jestliže přišlo potvrzení
134               if (Serial.peek() == slaveConf)
135               {
136                   report->println("confirmation OK\n");
137                   //ukončí tuto smyčku
138                   sendOK = true;
139                   sent = true;
140               }
141               //jestliže je adresa určena pro jinou jednotku slave
142               byte foreignAdr = Serial.peek();
143               for (byte i = 0; i < strlen(legalArrayAddress); i++)
144               {
145                   if (foreignAdr == legalArrayAddress[i])
146                   {
147                       report->println("request for other device");
148                       delay(10);
149                   }
150               }
151               //jestliže je znak z bufferu potvrzující značka pro jinou jednotku slave
152               byte foreignConf = Serial.peek();
153               for (byte i = 0; i < strlen(legalArrayConfirm); i++)
154               {
155                   if (foreignConf == legalArrayConfirm[i])
156                   {
157                       report->println("confirmation for other device");
158                       delay(4);
159                   }
160               }
161               //pro vymazání bufferu -> po splnění jakékoliv podmínky v tomto cyklu
162               clear_buffer();
163           }
164       }
165   }
166
167
168   //pro inicializaci pole povolených adres a potvrzení
169   void MySlave::legal_address(byte* items, byte itemsLength)
170   {
171       byte maxLength = 32;
172       //ochrana proti přetečení
173       if (itemsLength > maxLength)itemsLength = maxLength;
174       for (byte i = 0; i < itemsLength; i++)
175       {
176           legalArrayAddress[i] = items[i];
177       }
178   }
```

Obrázek B.3: *Zdrojový kód pro odesílání datových zpráv ze senzorové jednotky*

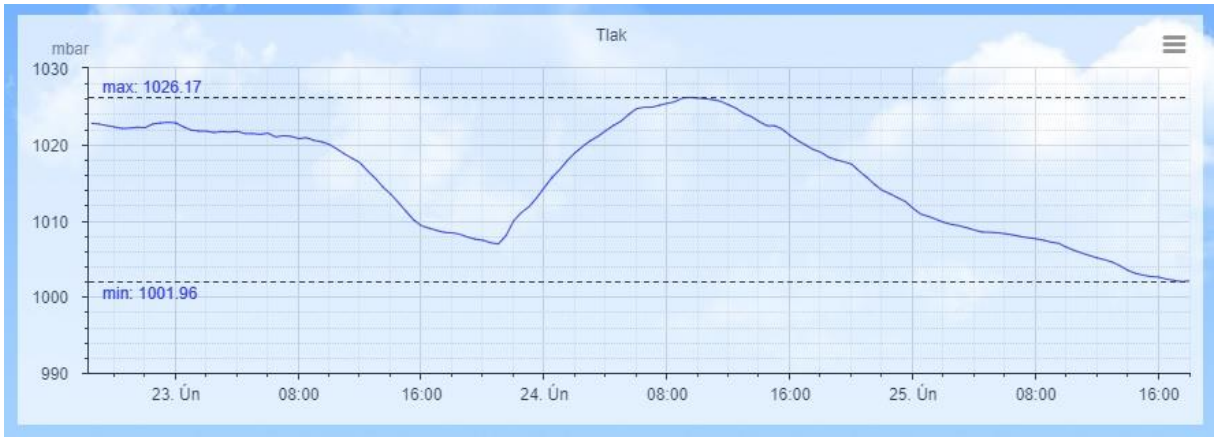
C SNÍMKY OBRAZOVEK



Obrázek C.1: *Jednotlivé záznamy uložené v databázovém systému MySQL.*



Obrázek C.2: *Vývoj teploty za 72 hodin s maximální a minimální hodnotou.*



Obrázek C.3: *Vývoj tlaku za 72 hodin s maximální a minimální hodnotou.*

Obrázek C.4: *Vývoj směru, rychlosti a nárazů větru s maximální a minimální hodnotou.*

Obrázek C.5: *Vývoj vlhkosti za 72 hodin s maximální a minimální hodnotou.*

Obrázek C.6: *Přírůstky srážkového úhrnu za 72 hodin s maximální a minimální hodnotou.*