Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Měření spotřeby a doby používání strojů s využitím platformy Arduino

Bc. Jiří Pavelka

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jiří Pavelka

Systémové inženýrství a informatika
Informatika

Název práce

Měření spotřeby a doby používání strojů s využitím platformy Arduino

Název anglicky

Measure consumtion and time of use of machines with using the Arduino platform

---

Cíle práce

Cílem diplomové práce je vytvoření zařízení pro měření spotřeby a doby používání strojů se zobrazením dat na webové stránce.
Dílčí cíle jsou:
- vytvoření jednotek pro měření spotřeby,
- vytvoření webové stránky pro zobrazování statistik,
- vytvoření webového uživatelského rozhraní.

Metodika

Jednotky pro měření spotřeby budou realizovány za pomoci mikrokontroleru, vybraných měřících transformátorů a dalších elektronických komponent, které jsou dostupné ve formě hotových modulů. V programovacím jazyce Wiring bude vytvořen zdrojový kód pro mikrokontroler, ten bude prostřednictvím bezdrátové sítě WiFi odesílat naměřená data s číslem stroje a časem záznamu do relační databáze. Dále bude vytvořena webová aplikace s rozhraním pro vkládání uživatelských dat (číslo zakázky, zákazník, obsluha stroje, výrobek, časový rozsah práce, aj.) a zobrazování důležitých statistik, jako např. vytíženost jednotlivých strojů, jejich spotřeba, přehledy o zakázkách, výpočty nákladů. Amortizace a statistiky budou odvozovány z nasbíraných dat a zobrazeny ve formě grafů. Pro spolehlivé měření, bez ztráty dat bude v jednotkách použita zálohovaná paměť a zdroj napájení. Celý systém měření bude odzkoušen a dále provozován v praktickém použití. Závěr práce zhodnotí teoretické poznatky a praktickou část.

Doporučený rozsah práce

50-60 stran

Klíčová slova

ESP32, analyzátor sítě, MySQL, Arduino, měření spotřeby, amortizace

---

Doporučené zdroje informací

David Sklar, PHP 7 – Praktický průvodce nejrozšířenějším skriptovacím jazykem pro web. Zoner Press (2018). ISBN: 978-80-7413-363-3

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

Kitchen, Z. V. Průvodce světem Arduina. Nakladatelství Martinn 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

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 3. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 7. 3. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 28. 03. 2022

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Měření spotřeby a doby používání strojů s využitím platformy Arduino" jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 28.3.2022

Poděkování

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

Měření spotřeby a doby používání strojů s využitím platformy Arduino

Abstrakt

Tato práce se zabývá návrhem, sestavením a naprogramováním zařízení pro měření spotřeby elektrické energie a doby chodu strojních zařízení v průmyslové výrobě. Zařízení je navrženo a sestaveno za pomoci mikrokontroleru ESP32 a dalších modulů platformy Arduino. Měření veličin, jako jsou napětí, proud, činný výkon, spotřeba el. energie a účiník probíhá pomocí modulů PZEM004T v obvodu zátěže třífázové soustavy na strojích Mazak Integrex 100Y, Brother TC-225 a Boge VL-11.

Měření veličin elektrické sítě zajišťují 3 moduly PZEM004T s měřícími proudovými transformátory. Uvedené moduly disponují vlastním MCU založeném na čipu Intel 8052, který řeší veškeré dílčí výpočty a jejich následného uložení do bufferu. Moduly PZEM004T jsou propojeny rozhraním UART s hlavním MCU ESP32, který načítá data z bufferu v intervalech 1sekundy. V minutových intervalech jsou MCU ESP32 prováděny sumarizace a další odvozené výpočty, jejichž výsledky jsou přes WiFi odesílány do databázového systému MySQL. Data jsou zobrazována ve formě grafů a textových údajů pomocí aplikace využívající třívrstvé architektury. Aplikace také umožňuje zadat další uživatelská výrobní data (zahájení a ukončení výroby daného stroje, č. výrobku, č. zakázky a počet kusů), která v kombinaci s naměřenými veličinami poskytují ucelené reporty o nákladech na výrobu. Přístup do aplikace je zabezpečen pomocí jména a hesla s dvěma úrovněmi oprávnění.

Vytvořené zařízení umožňuje měřit všechny výše uvedené veličiny. Pomocí aplikace lze navíc zadávat uživatelská data, která umožňují výpočet nákladů na výrobu. Výsledky získané prostřednictvím uvedeného zařízení mohou sloužit jako podkladová data pro diagnostiku samotného stroje, nebo při rozhodování budoucího rozšiřování, či specializace výroby. Požadované funkce v navrženém systému byly odzkoušeny půlročním testováním a spolehlivým provozem tří zařízení ve výrobní dílně.

Klíčová slova: ESP32, analyzátor sítě, MySQL, PZEM004T, Arduino, měření spotřeby, amortizace, DS3231, LR7843, multidrop

Measuring the consumption and time of use of machines with using the Arduino platform

Abstract

This work deals with the design, assembly and programming of equipment for measuring the electricity consumption and running time of machinery in industrial production. The device is designed and assembled using the ESP32 microcontroller and other Arduino platform modules. Measurements of quantities such as voltage, current, active power, power consumption and power factor are made using PZEM004T modules in the load circumference of the three-phase system on Mazak Integrex 100Y, Brother TC-225 and Boge VL-11 machines.

Measuring electrical grid quantities are provided by 3 PZEM004T modules with measuring current transformers. These modules have their own MCU based on the Intel 8052 chip, which handles all sub-calculations and their subsequent buffer storage. The PZEM004T modules are connected by the UART interface to the main MCU ESP32, which reads data from the buffer at intervals of 1 second. The MCU ESP32 is summarized and other derived calculations are performed at minute intervals, the results of which are sent via WiFi to the MySQL database system. The data is displayed in the form of graphs and text data using an application using three-layer architectures. The application also allows you to enter additional user production data (start and end of production of the given machine, product No., order No., and number of pieces) which, combined with the measured quantities, provide comprehensive reports on production costs. Access to the app is secured using a name and password with two permission levels.

The device created allows you to measure all of the above variables. In addition, you can use the application to enter user data to calculate the cost of production. The results obtained by means of the equipment may serve as background data for the diagnosis of the machine itself, or in the decision of future expansion, or the specialisation of production. The required functions in the designed system have been tested by a half-yearly testing and reliable operation of the three plants in the production workshop.

Keywords: ESP32, network analyzer, MySQL, PZEM004T, Arduino, power measurement, amortization, DS3231, LR7843, multidrop

Obsah

1 Úvod 11

2 Cíl práce a metodika 12

2.1 Cíl práce 12

2.2 Metodika 13

3 Měření činného výkonu elektrické energie 14

4 Platforma Arduino 15

5 Hardware 16

5.1 ESP32 17

5.2 Rozhraní GPIO 18

5.3 Rozhraní UART 19

5.4 Převodník CP2102 20

5.5 Rozhraní SPI 21

5.6 Karta microSD 22

5.7 Rozhraní I^2C 23

5.8 RTC3231 25

5.9 Level shifter 26

5.10 Modul MOSFET LR7843 27

5.11 Záložní zdroj UPS 28

5.12 Modul PZEM-004T 29

5.13 Měřící transformátor 31

6 Software 32

6.1 Jazyk Wiring 33

6.2 Nastavení rozhraní v jazyku Wiring 34

6.3 Knihovny pro Arduino 34

6.4 MySQL 36

6.5 Jazyk PHP 37

6.6 JavaScript 38

6.7 HTML a CSS 38

7 Měřič spotřeby 39

7.1 Návrh měřiče spotřeby 40

7.2 Zapojení a odzkoušení modulu PZEM-004 41

7.3 Multidrop UART 42

7.4 UPS jednotka s řízeným vypínáním zátěže 43

7.5 Úprava jednotky RTC 45

7.6 Odzkoušení celkového obvodu měřiče ... 46
7.7 Hardwarové uspořádání ... 47
7.8 Cenová kalkulace materiálu ... 48
7.9 Návrh a implementace zdrojového kódu měřiče spotřeby ... 49
7.10 Zkušební provoz a odladění kódu ... 51
7.11 Ukázka části zdrojového kódu zařízení ... 53

8 Webová aplikace ... 54
8.1 Návrh aplikace ... 55
8.2 Struktura webové aplikace ... 56
8.3 Návrh a vytvoření databáze MySQL ... 57
8.4 Aplikační vrstva ... 58
8.5 Klientská část v JavaScript ... 61
8.6 Struktura složek serveru FTP ... 63
8.7 Vytváření grafů pomocí knihovny Highcharts ... 64
8.8 Statistiky jednotlivých strojů ... 66
8.9 Elektronický deník ... 67
8.10 Nastavení parametrů výroby ... 68
8.11 Přehledy o výrobě ... 69
8.12 Náklady výroby ... 70

9 Závěr ... 71

10 Reference ... 72

11 Přílohy ... 76

Seznam obrázků

Obrázek 1: *průběh střídavého výkonu v obvodu s induktivní zátěží* 14
Obrázek 2: *Vývojová deska ESP32 od Espressif* 17
Obrázek 3: *Rozmístění jednotlivých pinů* 18
Obrázek 4: *Vnitřní schéma obvodu CP2102* 20
Obrázek 5: *Příklad zapojení rozhraní SPI* 21
Obrázek 6: *Paměťová karta microSD s paticí* 23
Obrázek 7: *Přenos datových bitů v rozhraní I^2C* 24
Obrázek 8: *Modul RTC* 25
Obrázek 9: *Schéma zapojení oboustranného převodníku* 26
Obrázek 10: *Schéma zapojení modulu s LR7843* 27
Obrázek 11: *Modul s MT3608 (vlevo) a TP4056 (vpravo)* 28
Obrázek 12: *Schéma zapojení modulu PZEM-004T* 29
Obrázek 13: *Měřící transformátor, včetně výpočtu zátěžového odporu* 31
Obrázek 14: *Struktura jednoduchého kódu v jazyku Wiring* 33
Obrázek 15: *Příklad knihovny mat.h pro sčítání dvou čísel* 35
Obrázek 16: *Historický vývoj MySQL* 36
Obrázek 17: *Předání dat z databáze do JSON pomocí PHP* 37
Obrázek 18: *Schéma měření a předávání ze zařízení* 39
Obrázek 19: *Schéma zapojení modulu PZEM -004T a ESP32* 41
Obrázek 20: *Schéma zapojení sběrnice UART multidrop* 42
Obrázek 21: *Schéma zapojení UPS s řízeným vypínáním* 44
Obrázek 22: *Úprava DS3231 pro baterii CR2032* 45
Obrázek 23: *Zapojení měřících obvodu modulu PZEM-004T* 46
Obrázek 24: *Krabička zařízení měřiče spotřeby* 47
Obrázek 25: *Vývojový diagram měřiče spotřeby* 50
Obrázek 26: *Diagnostický graf volné paměti a síly signálu* 52
Obrázek 27: *Ukázka části zdrojového kódu měřiče* 53
Obrázek 28: *Doménový model struktury webové aplikace* 56
Obrázek 29: *Relační datový model* 57
Obrázek 30: *Funkce pro získání dat v jazyku PHP* 59

Obrázek 31: *Algoritmus přípravy dat Ganttova diagramu* 60
Obrázek 32: *Ukázka kódu v jazyku JavaScript* 61
Obrázek 33: *Kód JavaScript pro přidání řádku tabulky* 62
Obrázek 34: *Stromová struktura složek aplikace* 63
Obrázek 35: *Ukázka definování grafu v Highcharts* 64
Obrázek 36: *Graf Highcharts pro statistiku stroje* 65
Obrázek 37: *Statistika stroje Mazak Integrex 100Y* 66
Obrázek 38: *Snímek záložky elektronického deníku* 67
Obrázek 39: *Záložka nastavení parametrů výroby* 68
Obrázek 40: *Celkové statistiky výroby* 69
Obrázek 41: *Záložka s přehledem nákladů* 70

Seznam tabulek

Tabulka 1: *Specifikace modulu Peacefair PZEM-004T* 30
Tabulka 2: *Rozpis položek včetně jejich ceny* 48
Tabulka 3: *Zahrnuté knihovny zdrojového kódu měřiče* 49
Tabulka 4: *Parametry pro záznamy elektronického deníku* 54

Seznam použitých zkratek

NTP	Network Time Protocol
UX	User Experience
JSON	JavaScript Object Notation
LED	Light-Emitting Diode
ADC	Analog to Digital Converter
AJAX	Asynchronous JavaScript and XML
CMS	Content Management System
CNC	Computer Numerical Control
DAC	Digital to Analog Converter
EEPROM	Electrically Erasable Programmable Read-Only Memory
GPIO	General-purpose input/output
I^2C	Inter Integrated Circuit
I^2S	Inter IC Sound
IDE	Integrated Development Environment
LSB	Least Significant Bit
MCU	Microcontroler
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MSB	Most Significant Bit
PHP	Personal Home Page
PWM	Pulse Width Modulation
RAM	Random Access Memory
RISC	Reduced Instruction. Set Computer
RTC	Real Time Clock
SELV	Safety Extra Low Voltage
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SRAM	Static Random Access Memory
SSID	Service Set Identifier
UART	Universal Asynchronous Receiver-Transmitter
UPS	Uninterruptible Power Supply
USB	Universal Serial Bus

1 Úvod

Elektroměr je zařízení pro sledování množství odebrané elektrické energie v čase. Je připojen na rozhraní distribuční sítě a místa odběru, tím může být např. malá domácnost, rodinný dům, menší firma, nebo velký průmyslový podnik. Pro distribuční síť v České republice je využívána třífázová soustava, která je následně transformována na napětí 3x400 V. Spotřebitel je do distribuční sítě připojen přes třífázový, nebo jednofázový elektroměr. Jednofázový elektroměr nachází uplatnění v malých domácnostech, většinou bytech, které nevyužívají elektrickou energii pro vytápění, či napájení třífázových spotřebičů. Na základě odečtu z elektroměru provádí dodavatel pravidelné vyúčtování.

Elektroměry umožňují přímé, nebo nepřímé měření, mohou být vybaveny systémem pro měření ve dvou tarifních třídách. V situacích, kdy je spotřebitel zároveň výrobcem, např. vlastní fotovoltaickou elektrárnu, je instalován elektroměr umožňující měřit spotřebu i výrobu zároveň, přičemž měření spotřeby a výroby probíhá po fázích, nikoliv součtově. Přímé měření je realizováno přes elektroměr zapojený přímo mezi vstupní a výstupní vodiče místa spotřeby. Proti tomu nepřímé měření využívá principu elektromagnetické indukce, kdy jsou přívodní vodiče protaženy skrz cívky měřících transformátorů. Druhý způsob měření je obvykle instalován v místech, kde měřený proud překračuje 100 A.

Elektroměr může být také zapojen jako podružný měřič, ať už pro určitý celek, nebo jednotlivý spotřebič. Ve výrobních podnicích, kde může spotřeba energie představovat značnou část nákladů na výrobu, má měření spotřeby jednotlivých strojů, či výrobních celků, významnou roli při analýze nákladů. Dále může obsahovat funkce pro záznam elektrických veličin daného stroje včetně doby chodu, která poskytuje základní měřítko pro výpočet amortizace.

Tato diplomová práce se zabývá vytvořením zařízení pro měření spotřeby a doby chodu strojů v průmyslovém prostředí. Zařízení vychází z platformy Arduino a umožňuje měření dalších elektrických veličin v třífázové soustavě, jako jsou elektrické napětí, činný výkon a protékající proud. Naměřená data jsou zaznamenávána a zobrazována prostřednictvím webové aplikace, která obsahuje další funkcionality, jako jsou nastavení parametrů výroby a jejich prostředků pro automatický výpočet nákladů.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem této diplomové práce je návrh a vytvoření zařízení pro měření spotřeby elektrického proudu a jeho dalších veličin ve strojích průmyslové výroby při využití platformy Arduino. Vytvořené zařízení musí umožňovat nepřímé měření elektrických veličin v třífázové soustavě při maximální chybě měření 0,5 %. Získaná data, jako jsou spotřeba elektrického proudu, činný výkon, proud a doba chodu, budou ukládána v minutových intervalech ze všech instalovaných zařízení do tabulky databáze. Prostřednictvím webové aplikace bude docházet k zobrazování statistik elektrických veličin ve formě grafů a textových informací. Kromě zobrazení nejrůznějších statistik bude webová aplikace poskytovat rozhraní pro nastavení parametrů výroby (amortizace, cena obsluhy, sazby, cena energie, cena materiálu, podíl podpůrných strojů, aj.) a ukládání výrobních záznamů do elektronického deníku. Podle parametrů a zaznamenaných dat budou probíhat automatické výpočty nákladů. Pro pilotní provoz aplikace budou vytvořeny 2 uživatelské role, první s oprávněním administrátor, druhá pro uživatele s omezeným oprávněním (náhled do statistik strojů a zadávání do elektronického deníku). Dále bude zařízení obsahovat záložní datové úložiště pro případ ztráty spojení s internetovým úložištěm a zmenšenou jednotku UPS s baterií li-ion pro dokončení přesunu, či uložení dat po vypnutí stroje. Elektronické součásti budou uloženy v plastové krabičce umožňující snadnou montáž do rozvaděče měřeného stroje. Měření spotřeby bude odzkoušeno na dvou obráběcích strojích CNC a šroubovém kompresoru s celkovým příkonem do 30 kVA. Výsledky měření budou porovnány s kalibrovaným měřičem spotřeby elektrického proudu.

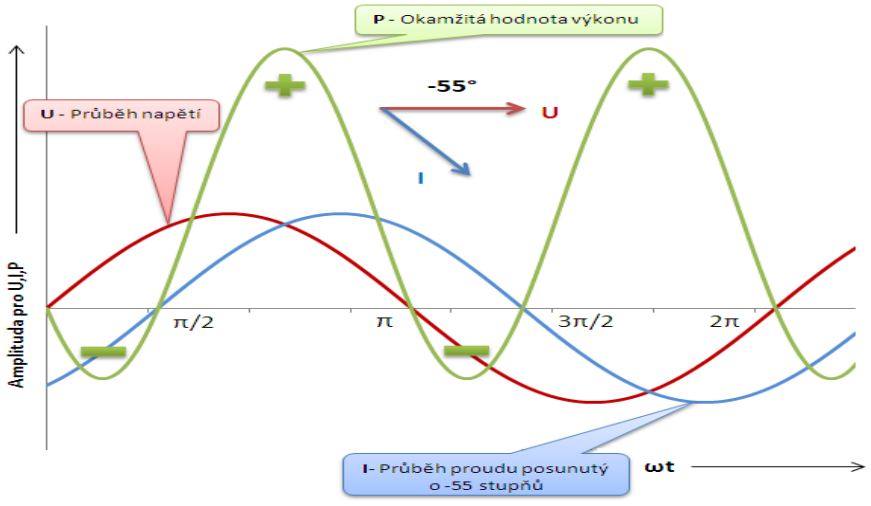
2.2 Metodika

Zařízení pro měření spotřeby elektrické energie bude vytvořeno pomocí běžně dostupných elektronických komponent, většinou v podobě hotových modulů. Výběr komponent bude zohledňovat dané požadavky, elektrické parametry, dostupnost dokumentace a možnosti využití již existující báze znalostí (diskusní fóra, tutoriály). Centrální část měřiče bude tvořena mikrokontrolerem ESP32, který bude komunikovat přes sériové rozhraní se třemi měřícími moduly PZEM-004T, každý pro měření jedné fáze. Přenos dat ze zařízení do sítě internet bude používat bezdrátové připojení WiFi. Měřič bude osazen kartou microSD obsahující soubor dočasné zálohy a konfigurace (SSID, heslo, číslo stroje, url). Vypínání zařízení po uložení veškerých naměřených dat bude zajišťovat elektronické relé typu MOSFET. Pro snadnou a rychlou výměnu modulů budou slaboproudé elektrické spoje vybaveny konektory. Všechny moduly budou umístěny v tištěné plastové krabičce navržené pro montáž na rozvaděčovou lištu DIN (35 mm). Měření proudu bude provedeno měřícími neinvazivními transformátory a měření napětí přímým připojením na vstupní silové svorky stroje. Z okamžitých hodnot napětí a proudu bude mikrokontroler modulu PZEM-004T provádět výpočet požadovaných elektrických veličin.

Moduly budou naprogramovány v jazyku Wiring za pomoci existujících knihoven, které jsou naprogramovány v jazyku C++. Knihovny jsou dostupné např. na stránkách výrobce, úložištích stránek Arduino, nebo Github. Nejprve budou naprogramovány a odzkoušeny jednotlivé části hardwaru. Následně bude navrženo a realizováno elektrické propojení všech částí. Vývoj zdrojového kódu bude postupovat od menších částí, které budou po úspěšném odzkoušení skládány do většího celku. Uvedený postup umožňuje vyšší efektivitu při hledání a odstraňování nově vzniklé chyby. Dále bude vytvořena databázová tabulka pro nasbíraná data. Po získání dostatečného množství dat bude v kódu jazyka HTML a JavaScript naprogramována webová aplikace, ta bude následně propojena s databázovým systémem prostřednictvím aplikační vrstvy naprogramované v jazyku PHP.

3 Měření činného výkonu elektrické energie

Pro přesný výpočet skutečně odebrané energie dle vzorce $W = P \,.\, t$ je nutné znát hodnotu činného výkonu, který je závislý na velikosti proudu, efektivního napětí a účiníku. Účiník vyjadřuje poměr mezi činným a zdánlivým elektrickým výkonem v obvodu střídavého proudu a napětí. Zdánlivý výkon je součin proudu a efektivního napětí měřený ve voltampérech (VA). Činný výkon představuje elektrickou energii přeměněnou na mechanickou, tepelnou, světelnou, apod. a je dán součinem efektivní hodnoty napětí a činné složky proudu, měří se ve wattech (W). [1] Induktivní spotřebiče, jako jsou např. motory, odebírají ze sítě navíc výkon potřebný k vytvoření magnetického pole, který následně odevzdávají zpět. Pokud je spotřebič induktivního (cívka), nebo kapacitního (kondenzátor) charakteru připojen do obvodu střídavého proudu a napětí, dochází k fázovému posunu mezi napětím a proudem. [2] Na základě vzorce pro výpočet činného výkonu $P = U * I * \cos\varphi$, kde $\cos\varphi$ vyjadřuje hodnotu účiníku, vyplývá přímá úměrnost mezi účiníkem a činným výkonem. Hodnota φ představuje fázový posun, tzn. že čím vyšší je fázový posun, tím nižší je skutečně odevzdaný výkon, resp. nižší účiník a skutečně odebraná hodnota elektrické energie. Následující graf představuje spotřebič (induktivní) v obvodu střídavého proudu a napětí s fázovým posunem.



Obrázek 1: *průběh střídavého výkonu v obvodu s induktivní zátěží*

Zdroj: **[1]**

4 Platforma Arduino

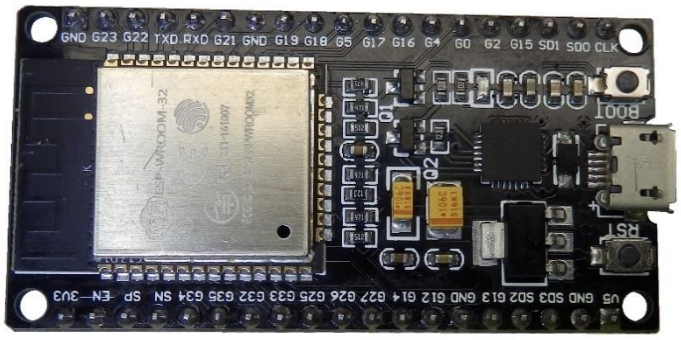
„Vývoj prvního Arduina započal v roce 2005, když se lidé z italského Interaction Design Institute ve městě Ivrea rozhodli vytvořit jednoduchý a levný vývojový set pro studenty, kteří si nechtěli, nebo nemohli pořídit tehdy velmi drahé desky, jako například BASIC Stamp. Mezi studenty se Arduino uchytilo, a tak se tvůrci rozhodli poskytnout ho celému světu. Velké rozšíření však nebylo způsobeno prodejem desek, ale hlavně sdílením všech schémat a návodů celému světu. Programová část Arduina byla založena na jazyku Processing, což je knihovna pro jazyk Java, ke které je přidán i vlastní editor. Vše má za cíl zjednodušit výuku programování. V dnešní době se prodalo několik stovek tisíc desek Arduino. Důkazem, že tato platforma není mrtvá, může být i to, že vzniklo několik desek ve spolupráci s velkými společnostmi, například Intelem. Od roku 2005 bylo vytvořeno spoustu různých typů Arduina. Jelikož se jedná o Open source projekt, vznikalo společně s hlavní linií projektu i spoustu dalších, neoficiálních typů, takzvaných klonů.“ [3]

5 Hardware

Klíčovým modulem pro platformu Arduino je vývojová deska, která obsahuje tzv. jednočipový počítač. K vývojové desce se pomocí dostupných rozhraní (UART, SPI, I^2C, I^2S, ad.) připojují jednotlivé moduly periférií. Periférie jsou buď vstupní, výstupní, nebo vstupně-výstupní jako např. přídavné paměti EEPROM**.** [3] V době psaní této práce bylo na trhu nepřeberné množství těchto modulů, ať už ve formě nejrůznějších senzorů, paměťových zařízení, rozšiřujících rozhraní, displejů, koncových obvodů akčních členů a mnoho dalších. Vybírat lze mezi originálním výrobkem, nebo jeho klonem, což je výrobek s nižší kvalitou, ale mnohonásobně nižší cenou. Moduly je také možné zakoupit na nákupních portálech asijských výrobců za nejnižší možnou cenu. Většina modulů pracuje s logickým napětím 5 V, nebo 3,3 V, přičemž velikost napájecího napětí je závislá na výrobci a obvykle se pohybuje od 3,3 V až do 12 V. Následující kapitoly popisují nejdůležitější charakteristiky a principy fungování součástek v modulech použitých pro stavbu zařízení měření elektrické energie.

5.1 ESP32

Jedná se o jednočipový počítač (označován jako mikrokontroler, zkráceně MCU) od firmy Espressif. Všechny klíčové obvody pro jeho fungování, jako např. operační paměť, nebo komunikační rozhraní, jsou integrovány v pouzdře jednoho čipu. V ESP32 jsou zabudovány dva 32bitové procesory Xtensa LX7, s instrukční sadou typu RISC, taktované na frekvenci až 240 MHz. Bezdrátová konektivita je zajištěna pomocí WiFi standardu 802.11bgn s rychlostí až 150 Mbps a rozhraním Bluetooth ve verzi 4.2 umožňující zpětnou kompatibilitu. Fungování WiFi zprostředkovává jedno jádro čipu, druhé je využíváno pro zpracovávání programu. Fyzické propojení dostupných rozhraní je realizováno vyvedenými vodiči z pouzdra čipu označených jako GPIO piny. Dále mikrokontroler obsahuje rozhraní UART, SPI, I^2C, zvukové rozhraní I^2S a DAC/ADC převodníky. Pokud jsou GPIO piny nastaveny jako výstupní, je možné využít pulzně šířkové modulace PWM k regulaci silových spínacích obvodů akčních členů. V situacích kdy je MCU napájen baterií, nebo akumulátorem, je možné přejít do některého z režimu spánku a ušetřit energii. Pro uložení programu je k dispozici paměť flash s kapacitou 4 MB. Zpracování programu zajišťuje operační paměť SRAM s kapacitou 520 kB. Při aktivaci režimu spánku, je možné použit 8 KiB paměti v jednotce RTC. [4]

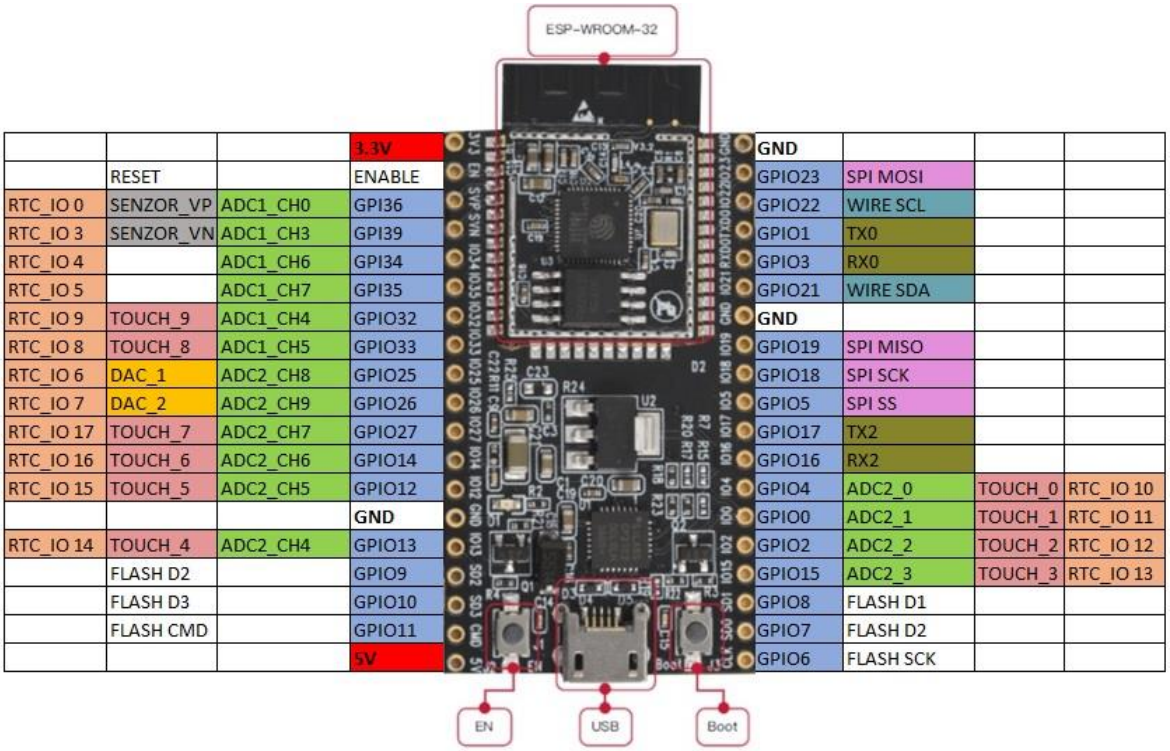


Obrázek 2: *Vývojová deska ESP32 od Espressif*

Zdroj: vlastní zpracování

5.2 Rozhraní GPIO

Srdcem měřiče spotřeby bude vývojová deska s MCU ESP32, která má 48 pinů. Většina z těchto pinů umožňuje universální použití pro vstupně-výstupní účely (anglicky General-purpose input/output, odtud zkratka GPIO), ostatní piny jsou rezervovány pro specifické funkce. Pomocí principu multiplexování lze nastavit pin jako digitální vstup, výstup, ADC/DAC převodník, pin pro rozhraní SPI, I^2C, UART, nebo jako dotykový senzor, či napěťový výstup Hallova senzoru. [4] Možnosti nastavení funkce pinů GPIO jsou na následujícím obrázku. Změna funkce pinu se provádí pomocí zápisu binárních hodnot odpovídajících registrů. Piny 34-36 a 39 lze nastavit pouze jako vstupní, piny 6-11 jsou vyhrazeny pro výměnu dat s pamětí flash, a proto by měly zůstat volné, stejně tak piny TX0 a RX0, které jsou určeny pro komunikaci s počítačem. Při aktivním rozhraní SPI, nebo I2C, nelze používat piny 5, 18, 19, 23, nebo 21 a 22, to samé platí pro piny ADC2 v režimu aktivní WiFi. Vstupní digitální piny mohou mít softwarově vřazený pull-up (ke kladnému potenciálu), nebo pull-down (k nulovém potenciálu) a lze je použít pro zpracování externího



Obrázek 3: *Rozmístění jednotlivých pinů*

Zdroj: **[5]**

přerušení. U výstupních digitálních pinů je možné použít pulzně šířkovou modulaci PWM (Pulse Width Modulation), která umožňuje plynulou regulaci výkonu koncových stupňů akčních členů, jako jsou např. řízení rychlosti otáčení motoru, průtoku pneumatického ventilů, nebo intenzity osvětlení. Mikrokontroler ESP32 má definovanou pozitivní logiku v rámci napětí 3,3 V při proudovém zatížení až 40 mA. Komunikace se zařízením s jinou napěťovou logikou vyžaduje použití převodníku napěťových úrovní (viz kapitola 5.9). [5]

5.3 Rozhraní UART

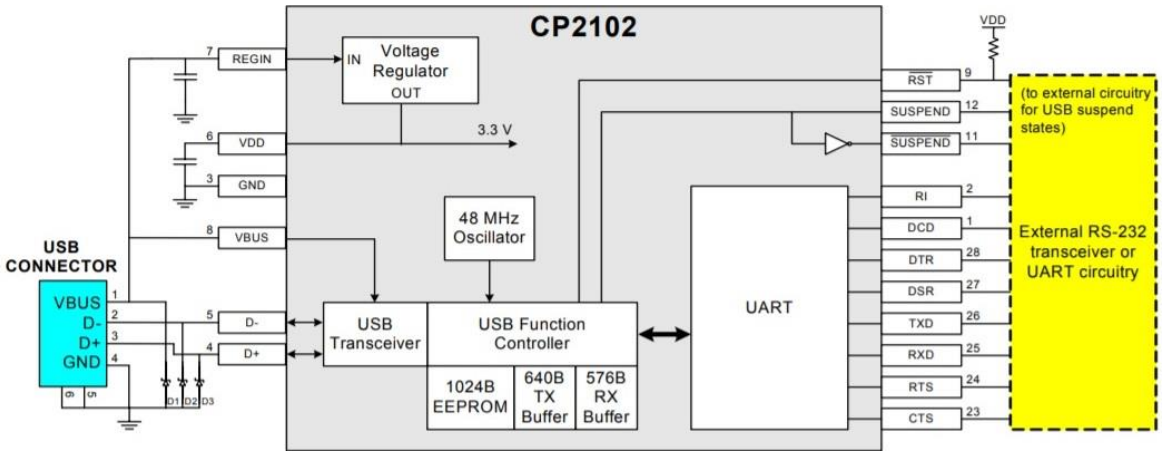
Univerzální rozhraní, které umožňuje vzájemnou výměnu dat mezi dvěma zařízeními při stejné napěťové logice. Vzdálenost těchto zařízení dosahuje maximálně desítek centimetrů. Pokud mají zařízení rozdílnou napěťovou logiku, je nutné použít převodník napěťových úrovní (viz kapitola 5.9). Vzájemná výměna dat probíhá asynchronně, pomocí přenosu jednotlivých znaků ASCII v plně duplexním režimu. Na fyzické vrstvě je přenos realizován pomocí dvoustavového signálu reprezentujícího logickou 0 a 1. Určitá sekvence bitů představuje přenášený znak ASCII. Délku sekvence lze programově volit v rozmezí 7-9 bitů, obvykle je nastaveno 8 bitů. [6] Mezi další volitelné parametry patří rychlost přenosu, počet stop bitů a kontrola parity. Vzhledem k možnosti výskytu sudého počtu chyb je spolehlivost kontroly parity značně omezená, a proto je nutné použít další kontrolu, jako je chyba bitu, rámce, nebo detekce ztráty dat. K uvedenému rozhraní je možné přidat převodníky standardů RS232, RS485, USB a dalších. [7]

Pro úspěšné předání dat je nutné, aby jedno zařízení začalo „naslouchat“ ve stejnou chvíli, kdy jsou data z druhého zařízení odesílána. To je provedeno synchronizačním pulsem, tzv. „start-bitem“ na začátku přenosu. Na jedné straně jsou data odesílána po vodiči označeném jako TX a na druhé straně načítána do přijímajícího bufferu pomocí vodiče RX metodou LSB (nejméně významným bitem napřed). Funkční zapojení musí být provedeno v konfiguraci Tx1 => Rx2 a Rx1 => Tx2 včetně nezbytného nulového vodiče označeného jako GND. [6]

Dvoustavový signál ESP32 je definován pro logickou 0 napětím v rozsahu 0-0,8 V a pro logickou 1 napětím 2-3,3 V. Pokud je linka v klidovém stavu, je vysílačem udržována ve stavu logické 1. Při zahájení přenosu je vyslán start bit (log 0), a pak následuje přenos samotného znaku ASCII, který je ukončen jedním, nebo dvěma stop bity (log 1). Následně přechází linka zpět do klidového stavu, až do chvíle odeslání dalšího znaku. [6]

5.4 Převodník CP2102

Integrovaný obvod obsahující prvky pro převod signálu mezi rozhraním USB a UART, který je součástí vybrané vývojové desky s MCU ESP32. Převodník je využíván pro komunikaci mezi MCU a počítačem při čtení a zápisu dat, nebo nahrávání programu do paměti flash. Na následujícím obrázku jsou znázorněny všechny klíčové součásti převodníku včetně jejich zapojení. V pouzdře integrovaného obvodu se nachází regulátor napětí, oscilátor, buffer pro příjem (573 B), buffer pro vysílání (640 B) a řadič USB 2.0. Umožňuje provoz v rámci napěťové logiky 3,3 V, nebo 5 V při max proudu 100 mA. Pro zprovoznění komunikace je nutné nainstalovat ovladače od společnosti Silicon Labs [8] (https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers).



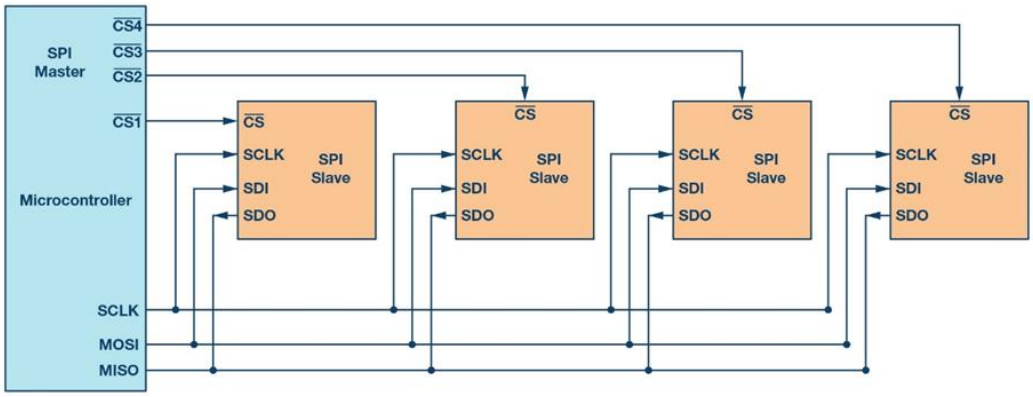
Obrázek 4: *Vnitřní schéma obvodu CP2102*

Zdroj: **[8]**

5.5 Rozhraní SPI

SPI (Seriál Peripheral Interface) je zabudováno ve všech mikrokontrolerech platformy Arduino. Bylo vynalezeno a uvedeno firmou Motorola ke konci 80.let jako nový standard, který je také někdy označován jako „four-wire serial bus“. Rozhraní umožňuje rychlou komunikaci na velmi malé vzdálenosti, které jsou obvykle zahrnuty v oblasti jednoho plošného spoje. Zařízení využívající rozhraní SPI mohou představovat např. sériové paměti RAM, EEPROM, paměťové karty SD, displeje, různé senzory, nebo MCU vzájemně propojené mezi sebou. Přenos dat probíhá synchronně, v plném duplexu, při rychlostech v řádu jednotek megabajtů za sekundu. U rozhraní SPI je obsaženo právě jedno zařízení typu master a jedno, nebo více zařízení typu slave. [6]

Řízení komunikace probíhá pomocí výběrového signálu generovaného zařízením master. Při výměně dat je CS (Chip Select) pin vybraného zařízení slave uveden zařízením master do stavu logické 0. Adresace tedy probíhá pomocí vodiče CS, nikoliv odesíláním adresy v datech. Uvedený způsob adresace přispívá k vyšší rychlosti, protože s daty není nutné přenášet adresu jako u jiných komunikačních rozhraní. Nevýhodou tohoto řešení může být omezený počet výstupních pinů MCU. [6] Na následujícím obrázku je znázorněno schéma zapojení rozhraní SPI.



Obrázek 5: *Příklad zapojení rozhraní SPI*

Zdroj: **[33]**

Samotná komunikace probíhá následujícím způsobem. Po uskutečnění výběru je od zařízení master po vodiči SCLK posílán synchronizační hodinový signál a po vodiči

MOSI (Master Out Slave In) tok datových bitů do zařízení slave metodou LSB. Datové bity jsou sekvenčně zpracovány pomocí posuvných registrů, které jsou obsaženy na vstupech a výstupech zařízení obou typů (master i slave). V opačném směru, tj. od zařízení slave do zařízení master jsou data přenášena po vodiči MISO (Master In Slave Out). [6]

Rozhraní SPI vyžaduje zapojení čtyř vodičů, včetně výběrového CS, ale v situaci kdy není požadován přenos dat ve směru od zařízení slave, je možné vynechat vodič MISO. Rozhraní SPI umožňuje programově konfigurovat čtení dat podle vzestupné, či sestupné hrany signálu a logickou úroveň pro klidový stav vodiče SCLK. [6] V některých zařízeních, obvykle paměťových, může být nainstalován dvojnásobný, nebo dokonce čtyřnásobný počet vodičů MISO/MOSI, což zvyšuje propustnost na 2, nebo 4 datové bity v jeden okamžik (jeden hodinový cyklus). Taková řešení bývají označena jako Dual SPI, nebo Quad SPI. [6]

5.6 Karta microSD

Jedná se nevolatilní paměťové zařízení, u kterého jsou data uchována i po odpojení zdroje elektrické energie. Pomocí patice je možné kartu připojit přes výše uvedené rozhraní SPI. Patice i karta je dostupná v běžné velikosti SD, nebo její zmenšené variantě microSD. Základním stavebním prvkem je paměťový čip NAND flash. Tato technologie byla vyvinuta společností Toshiba v roce 1987 jako nová generace pamětí EEPROM. [9]

V čipu jsou obsaženy binární buňky tvořené unipolárními tranzistory s plovoucími hradly izolovanými od okolí vrstvou oxidu křemíku. Zápis informace do paměťové buňky je realizován tokem elektronů přivedených na plovoucí hradlo unipolárního tranzistoru, který následně udržuje elektrický náboj reprezentující hodnotu zapsané informace. Při vícenásobném přepisování stejné buňky dochází k degradaci vrstvy oxidu křemíku, což má za následek omezenou životnost. Technologie SLC (Single Level Cell) ukládá do buňky pouze 1 bit informace při životnosti cca 100 000 zápisů. Technologie MLC (Multi Level Cell) umožňuje zvýšit kapacitu na dvojnásobek při stejném počtu tranzistorů a technologie TLC (Triple Level Cell) zvyšuje kapacitu na trojnásobek proti SLC. [10] Nevýhodou MLC a TLC proti SLC je pomalejší zápis a nižší životnost, ta je u MLC cca 10 000 a TLC cca 5000 zápisů. [11]

Nejmenší nedělitelnou částí paměti microSD karty je stránka o velikosti 4 kB. Každá stránka obsahuje 64 B metadat využívaných např. pro automatickou opravu chyb ECC, nebo pro uchování informace o počtu přepisů dané stránky. Stránky jsou poskládány do bloků

o velikosti 256 kB, přičemž zápis probíhá po celých blocích. Počet přepisů uložených v metadatech stránky umožňuje kontrolu a řízení rovnoměrného opotřebení všech stránek celé paměti.

V době psaní této práce bylo možné koupit paměťovou kartu v provedení microSD s kapacitou začínající na 4 GB, rychlostí zápisu 10 MB/s a rychlostí čtení 15 MB/s, za cenu cca 130 Kč. Průměrná spotřeba energie se pohybuje okolo 24 mA. [12]

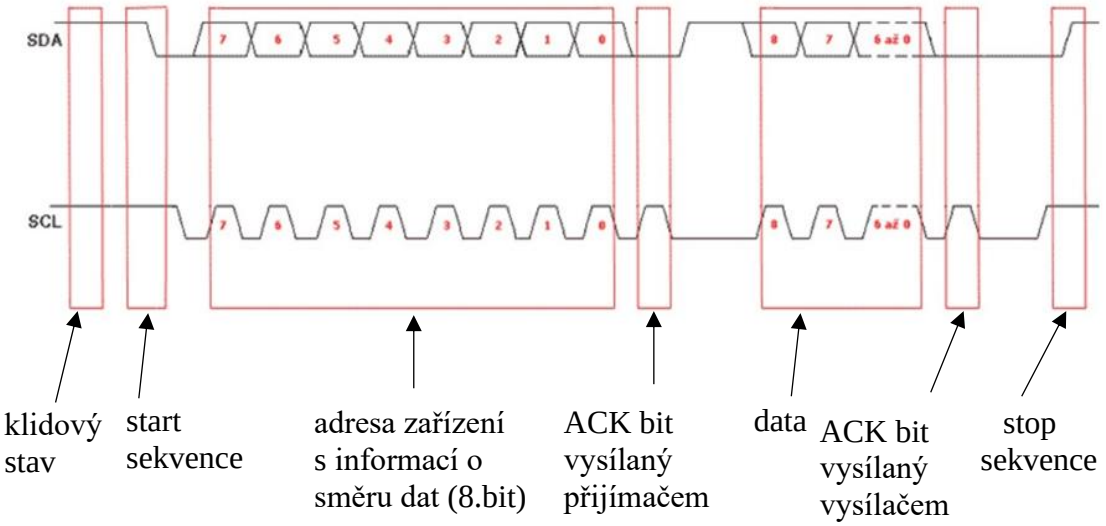
Obrázek 6: *Paměťová karta microSD s paticí*

Zdroj: vlastní zpracování

5.7 Rozhraní I²C

Jedná se o sériové rozhraní, které vyvinula společnost Philips. Zkratka I2C spadá pod chráněnou značku, a tak ostatní výrobci používají značení jako např TWI (Two Wire Interface). Používá se pro rychlou výměnu dat (až 400 kbit/s) mezi uzly sítě na vzdálenost několika desítek centimetrů. Zařízení standardu High Speed může přenášet data rychlostí až 3,4 Mbit/s. [6] U platformy Arduino se rozhraní využívá při komunikaci mezi MCU a periférií, jako jsou např. hodiny reálného času RTC, senzory tlaku, ohybu a magnetického pole.

Data jsou přenášena synchronně v polovičním duplexu, což znamená, že nelze odesílat a přijímat data v jeden okamžik. Fyzické propojení je realizováno dvojicí vodičů s označením SDA a SCL. Na rozdíl od rozhraní SPI umožňuje I^2C zapojit více zařízení typu master, takové uspořádání je označováno jako multi-master. Zařízení jsou adresována pomocí 7bitové adresy (v některých případech až 10bitová), která je uvnitř sítě jedinečná. Většinou jsou adresy pevně definovány výrobcem konkrétního zařízení. Změnu adresy obvykle umožňují obvody, kterých je zapojeno větší množství stejného druhu v jednom zařízení, např. paměti EEPROM, expandéry a převodníky. [6]



Obrázek 7: *Přenos datových bitů v rozhraní I^2C*

Zdroj: **[13]**

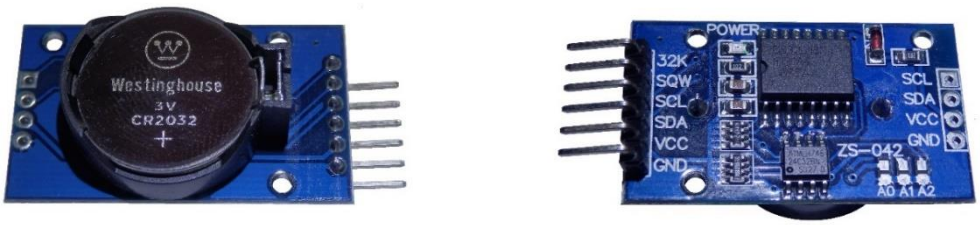
Přenos dat představuje sekvenci 8 bitů přenášenou metodou MSB (od nejvýznamnějšího bitu). Změny logických stavů jsou realizovány prostřednictvím obvodů s tzv. otevřeným kolektorem, obsahujícím tranzistor NPN, který při přechodu z logické 1 do logické 0 spíná výstup (otevřený kolektor) ke společnému potenciálu. [6]

Příklad přenosu dat je znázorněn na předcházejícím obrázku. Linky SCL a SDA jsou pomocí pull-up rezistorů udržovány v klidovém stavu (logická 1). Začátek přenosu znaku je zahájen přechodem z logické 1 na logickou 0 u vodiče SDA, přičemž vodič SCL zůstává ve stavu logické 1. Přenos samotných bitů je realizován načítáním logického stavu vodiče SDA při náběžné hraně vodiče SCL. Dle obrázku je patrné, že změna logického stavu vodiče

SDA při přenášení dat může probíhat pouze ve chvíli, kdy je vodič SCL ve stavu logické 0, v opačném případě dojde k ukončení, nebo zahájení nového přenosu (dle logického stavu linky SDA). První část za startovací sekvencí obsahuje sedmibitovou adresu s osmým bitem pro signalizaci směru (logická 1 pro čtení a 0 pro zápis). Pokud je nalezeno zařízení s odpovídající adresou, zařízení master uvolní datovou sběrnici a čeká na potvrzení (ACK - Acknowledge) připravenosti jednotky slave. Potvrzení je reprezentováno logickým stavem v 9. hodinovém cyklu na vodiči SDA. Pokud master přečte hodnotu 0, je zahájen přenos užitečných dat, v opačném případě, nebylo nalezeno zařízení s odpovídající adresou. [13]

5.8 RTC3231

Modul zařízení RTC (Real-Time-Clock) pro uchování reálného času, který komunikuje přes výše zmíněné rozhraní I^2C. Obsahuje velmi přesný obvod DS3231, jehož součástí je řízený oscilátor. Oscilátor dosahuje téměř konstantního kmitočtu po celou dobu fungování pomocí kompenzace vlivu okolní teploty. [14] Mezi hlavní funkce patří korekce přestupného roku, možnost volby mezi 12ti a 24hodinovým rozsahem, kalendář, nebo alarm s možností vyvolání přerušení. DS3231 provádí veškeré výpočty data a času a následně je ukládá do svých registrů, odkud jsou načítány mikrokontrolerem přes rozhraní I^2C. Přenášené údaje obsahují aktuální počet sekund, minut, hodin, pořadí dne v týdnu, v měsíci, měsíce a číslo roku. Nastavení aktuálního data a času probíhá zápisem do registrů DS3231 směrem od MCU, opět přes rozhraní I^2C, dále je již datum a čas udržován samotným obvodem DS3231. Pro uchování aktuálního času je nutné napájení záložní baterií, nebo akumulátorem. [15]



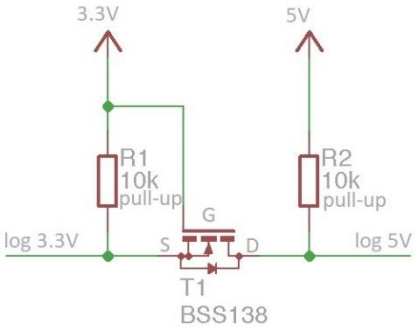
Obrázek 8: *Modul RTC*

Zdroj: vlastní zpracování

5.9 Level shifter

Level shifter je převodník napěťových úrovní využívaný pro situace, kdy je nutné propojit dvě zařízení s různou napěťovou logikou. U modulů platformy Arduino se jedná o logickou úroveň 3,3 V a 5 V. Převodník je dostupný ve formě hotového modulu jako jednosměrná, nebo obousměrná varianta pro 4, nebo 8 datových vodičů. Jednosměrný převodník nachází uplatnění při přechodu z vyšší napěťové logiky do nižší a jeho provedení je realizováno pomocí napěťových děličů. Obousměrný převodník vyžaduje složitější konstrukci obvykle řešenou tranzistory MOSFET. [16] Na následujícím obrázku je znázorněno schéma převodníku pro jeden datový vodič.

Rozdíl napětí mezi hradlem G a emitorem S je při logické 1 (vodič log 3,3 V) nulový, tranzistor BSS138 je uzavřený a napětí na obou koncích obvodu je udržováno pouze



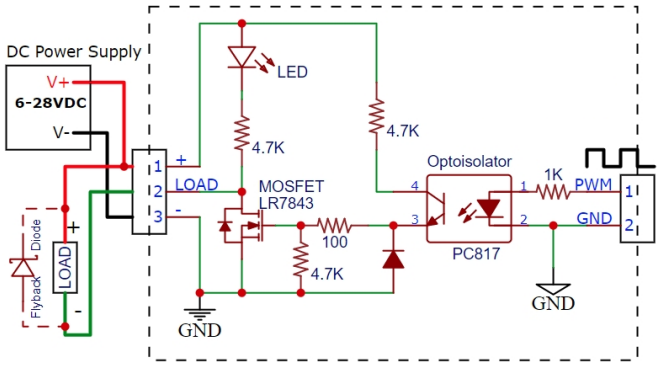
Obrázek 9: *Schéma zapojení oboustranného převodníku*

Zdroj: **[16]**

rezistory pull-up (R1, R2). Po přivedení logické 0 (vodič log 3,3 V) začné vlivem rozdílných potenciálů mezi hradlem G a emitorem S protékat proud, který otevře tranzistor BSS138. Přes otevřený tranzistor (přechod D-S) protéká daleko vyšší proud, než přes pull-up R2, čímž dochází k uzemnění vodiče (log 5 V) a jeho přechodu do stavu logické 0. Pokud je nejprve uzeměn vodič ze strany (log 5 V), je vodič (log 3,3 V) uzemněn pomocí diody v tranzistoru BSS138 a přechází do stavu logické 0. [16]

5.10 Modul MOSFET LR7843

Spínací proudy výstupních pinu MCU se pohybují v řádech desítek mA, u zmiňovaného ESP32 je maximální povolená zátěž na pin 40 mA. V aplikacích vyžadující daleko vyšší proudy je nutné použít výkonnější spínací obvod, jako např. LR7843. Jedná se tranzistor typu MOSFET, jehož proudové zatížení může dle datového listu dosahovat až 161 A, samozřejmě za podmínek dostatečně dimenzovaného chlazení a dodržení předepsaných specifikací výrobce. [17] Za cenu méně než 40 Kč (v době psaní této práce) lze sehnat osazený modul s galvanickým oddělením (optočlen PC817), rezistory, kontrolní LED a svorkami. Schéma zapojení uvedeného modulu je znázorněno na následujícím obrázku.



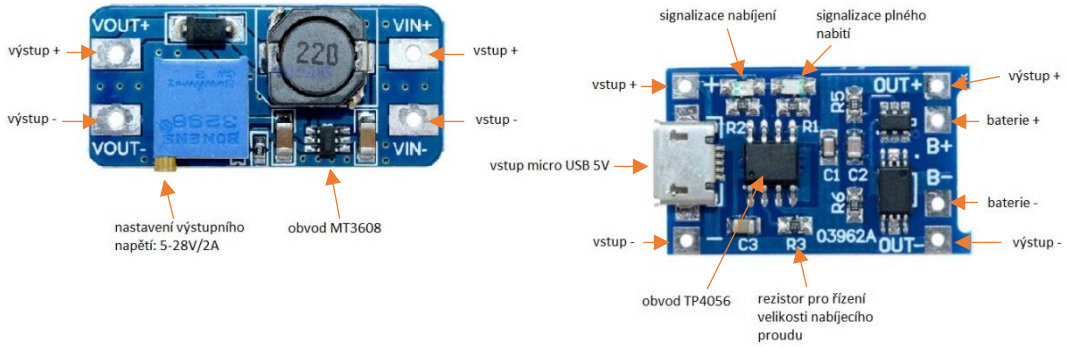
Obrázek 10: *Schéma zapojení modulu s LR7843*

Zdroj: **[18]**

Ovládací signál je galvanicky oddělen pomocí optočlenu, který chrání MCU při případném poškození tranzistoru LR7843. Vřazením proudového omezovacího rezistoru 1k je možné používat řídící napětí až 20 V [18], což umožňuje spolehlivé použití jakéhokoliv MCU platformy Arduino. Pokud dojde k sepnutí fototranzistoru uvnitř optočlenu, je přes napěťový dělič (2 x rezistor 4,7k) přivedeno cca 50 % velikosti napájecího napětí na hradlo tranzistoru LR7843, který následně přes přechod kolektor-emitor připojí zátěž k zemi. Při vypnutém fototranzistoru je hradlo v LR7843 přitaženo rezistorem 4,7k k zemi, LR7843 je vypnutý a zátěž odpojena.

5.11 Záložní zdroj UPS

Aby nedošlo k poškození, nebo ztrátě dat vlivem výpadku elektrické energie, je nutností použít záložní zdroj s plynulým přechodem na napájení z akumulátoru, takový zdroj se označuje zkratkou UPS (Uninterruptible Power Supply). UPS v sobě zahrnuje základní části, jako jsou měnič napětí, akumulátor, jeho nabíjení a ochrany proti zkratu, přetížení, přepětí, podpětí, nebo poškození akumulátoru. Pro konstrukci zdroje UPS nízkého napětí jsou na trhu k dispozici moduly s obvodem TP4056 v několika různých modifikacích. Variantu s ochranným obvodem baterie DW01A proti přebíjení, hlubokému vybití (< 2,9 V), přetížení, či zkratu lze pořídit za cenu cca 20 Kč. Uvedený modul je navržen pro záložní akumulátor typu li-ion 18650 s nabíjecím proudem do 1 A. Velikost výstupního napětí modulu se mění v závislosti na stavu množství energie v akumulátoru (2,9-4,2 V). [19] Pokud MCU vyžaduje napájení 3,3 V, nebo 5 V je nutné za uvedený modul UPS připojit měnič, který reguluje výstupní napětí na požadovanou konstantní hodnotu. Pro uvedený modul s ESP32 vyžadující vstupní napětí o velikosti 5 V a minimální proud 1 A je vhodný např. modul step-up s obvodem MT3608, který je schopný dodávat při požadovaném napětí až 2 A. Uvedené moduly jsou zobrazeny následujícím obrázku.

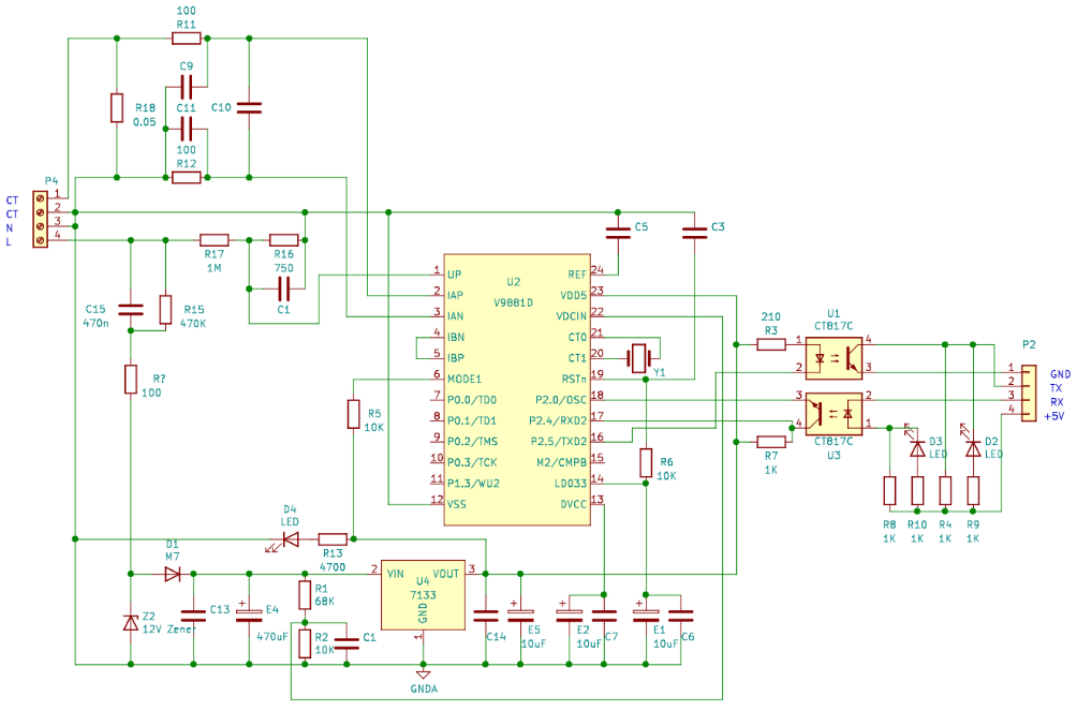


Obrázek 11: *Modul s MT3608 (vlevo) a TP4056 (vpravo)*

Zdroj: vlastní zpracování

5.12 Modul PZEM-004T

Modul pro měření spotřeby, příp. dalších veličin síťového napětí a proudu od výrobce Peacefair Electronic Technology. V době psaní této práce byl k dispozici ve verzi 3, která umožňuje měření síťového napětí, proudu, účiníku, činného výkonu, frekvence a spotřeby v jedné fázi. Na výběr je varianta s rozsahem 0-10 A s přímým měřením, nebo varianta pro vyšší rozsah 0-100 A s nepřímým měřením. Srdcem modulu je čip V9881D od Vango Technologies. V čipu je obsažen 8bitový MCU 8052, původně vyvinutý firmou Intel (taktovaný na 26 MHz), dále jednotka RTC pro práci s reálným časem, obvod watchdog timer proti zacyklení, 128 kB paměti flash a 4 kB operační paměti typu SRAM. V paměti flash je výrobcem vytvořený a vložený program včetně kalibračních dat. Komunikace probíhá prostřednictvím sériového rozhraní UART, které je galvanicky odděleno pomocí dvou optočlenů PC817, jeden pro RX a druhý pro TX. Napájení čipu V9881D je realizováno děličem síťového napětí a následnou lineární stabilizací obvodem 7133-1 na výsledných 3,3 V. [20] Celé schéma zapojení je uvedeno na následujícím obrázku.



Obrázek 12: *Schéma zapojení modulu PZEM-004T*

Zdroj: **[35]**

Pomocí odporového děliče je nejprve snížena velikost měřeného síťového napětí na povolený rozsah vstupu čipu VD9881D. Pro měření proudu je nutné nejprve provést převod na odpovídající napětí pomocí měřících transformátorů (viz. následující kapitola). Takto upravené hodnoty jsou přes filtry přivedeny na vstupy 20bitových ADC (Analog Digital Converter) převodníků. Výstupem převodníků jsou hodnoty s rozsahem 2^{20}, které umožňují v rámci následujících výpočtů získat velmi přesné výsledky (dle výrobce je třída přesnosti 0,5 %) požadovaných veličin obvodu elektrické sítě. Aby bylo možné vypočítat hodnotu účiníku, je nutné vztáhnout získané vzorky napětí a proudu ke konkrétnímu časovému okamžiku, ten poskytuje vnitřní jednotka RTC. [20] Vypočtené hodnoty jsou uloženy do registrů bufferu čipu VD9881D, odkud jsou načítány přes rozhraní UART. Pokud je třeba aktualizovat hodnoty v bufferu, je poslán nový požadavek a měření se opakuje. Měření akumulované energie je možné resetovat, pokud není resetováno do hodnoty 9999,99 kWh, dojde k jeho vynulování automaticky. V následující tabulce jsou uvedeny základní specifikace modulu PZEM-004T-100A. [21]

měřená veličina	**rozsah**	**rozlišení**	**přesnost**
napětí	80 ~ 260 V	0,1 V	0,5 %
proud	0 ~ 100 A	0,001 A	0,5 %
činný výkon	0 ~ 23 kW	0,1 W	0,5 %
účiník	0,00 ~ 1,00	0,01	1 %
frekvence	45 ~ 65 Hz	0,1 Hz	0,5 %
spotřebovaná en.	0 ~ 9999,99 kWh	1 Wh	0,5 %
Pro verzi PZEM-004T-10A je rozsah měření proudu 0 ~ 10 A a činného výkonu 0 ~ 2,3 kW.			

Tabulka 1: *Specifikace modulu Peacefair PZEM-004T*

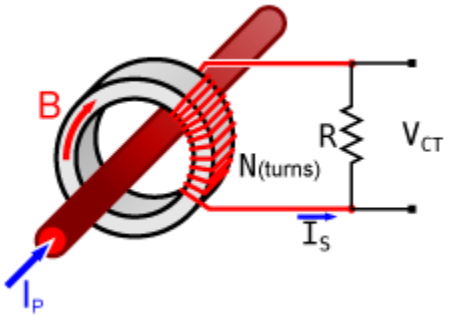
Zdroj: **[21]**

Z uvedeného schématu je patrné, že zařízení vyžaduje připojení k síťovému napětí, které dodává energii čipu V9881D, a také připojení nízkého napětí pro napájení optočlenů PC817 rozhraní UART s napěťovou logiku 5 V. Optočleny oddělují zónu s nebezpečným síťovým napětím od zóny s bezpečným nízkým napětím 5 V, v tomto případě se jedná o způsob ochrany SELV (Safety Extra Low Voltage) dle normy ČSN EN 62368-1. Dále je nutné poznamenat, že při práci s tímto modulem je nezbytné dodržovat všechny zásady

pro ochranu před úrazem elektrickým proudem včetně dodržení vyhlášky 50/1978 o odborné způsobilosti v elektrotechnice, minimálně §6.

5.13 Měřící transformátor

Zařízení pro nepřímé měření proudu protékajícího v obvodu zátěže elektrické sítě. Vysokonapěťové proudy jsou transformovány na nižší hodnoty pro bezpečné monitorování skutečného proudu. Nepřímé měření je založeno na elektromagnetické indukci na rozdíl od přímého měření, které měří úbytek napětí na rezistoru vloženého do obvodu zátěže. Měřící transformátor se skládá z jednoho, nebo velmi malého počtu závitů primárního vinutí tvořeného silovým vodičem provlečeného skrz jádro sekundární cívky. Sekundární vinutí má mnohem vyšší počet závitů (určených dle převodového poměru) navinutých na laminovaném jádře z magneticky měkké oceli a je připojeno k odporové zátěži. [22] Velikost odporu zátěžového rezistoru pro měřící transformátor je vypočítána dle následujícího vzorce.



$$R = \frac{V_{CT} * N}{I_P}$$

Obrázek 13: *Měřící transformátor, včetně výpočtu zátěžového odporu*

Zdroj: **[34]**

6 Software

Zdrojové kódy pro MCU platformy Arduino je možné vytvářet pomocí vývojového prostředí IDE (Integrated Development Enviroment), jak ve verzi pro desktop, tak ve verzi pro cloud. Celé vývojové prostředí bylo napsané v jazyku Java. Jedná se o upravený software původně používaný pro výuku v prostředí Processing s podporou knihovny Wiring, která je doplněna o některé další funkce.

Vizualizace dat může být realizována prostřednictvím MCU v kombinaci se zobrazovacím zařízením, jako je např. monochromatický led displej, nebo barevný dotykový displej. Pro výpočetně náročnější aplikace jsou data předávána výkonnějšímu stroji, např. mikropočítači Raspberry, Orange PI, nebo „klasickému“ počítači. Data jsou do uvedených strojů přenášena buď bezdrátově, nebo po pevné síti a jejich zpracování musí být zajištěno dalším programovým vybavením, jako jsou operační systém, webový server a databázový systém. Další možností může být použití hotového řešení ve formě cloudu.

Vzhledem k požadavkům a budoucímu nasazení na firemní server Apache s podporou MySQL a PHP, bude zpracování a prezentace dat provedena pomocí třívrstvé architektury. Následující kapitoly se budou zabývat uvedeným databázovým systémem, programovacím jazykem PHP a programovacími jazyky HTML a JavaScript pro zobrazování a dynamiku webových stránek.

6.1 Jazyk Wiring

Zařízení pro platformu Arduino lze programovat přímo pomocí jazyka C, nebo C++, ale pro rychlý a snadnější vývoj zdrojového kódu je upřednostňována knihovna Wiring. Vzhledem k jejímu rozsahu a komplexnosti se o ní někdy hovoří jako o samostatném programovacím jazyku. Protože je základem jazyk C++, je možné vytvářet zdrojový kód pomocí objektově orientovaného přístupu. [3]

Základní struktura zdrojového kódu, vytvořená pomocí knihovny Wiring v prostředí Arduino IDE, je složena ze dvou hlavních funkcí. Kód umístěný v první funkci, pojmenované `setup()`, je zpracován pouze jedenkrát po spuštění zařízení. Kód v těle druhé funkce, pojmenované `loop()`, je cyklicky prováděn až do restartu, nebo vypnutí zařízení. Pro úspěšnou kompilaci kódu musí být obě funkce obsaženy, i když jedna z nich bude prázdná (většinou funkce `loop()`). Bloky kódu jsou tvořeny složenými závorkami a příkazy uvnitř bloku jsou odděleny středníkem. Ukázka velmi zjednodušeného kódu, provádějícího výpis do konzole je znázorněna na následujícím obrázku.

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

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

Obrázek 14: *Struktura jednoduchého kódu v jazyku Wiring*

Zdroj: vlastní zpracování

Program je zpracováván sekvenčně, tzn. že je nejprve nastavena komunikace sériového rozhraní UART (9600 baud/s) pro výpis znaku do konzole prostředí Arduino IDE, pak následuje cyklické vypisování textu „Arduino“ v intervalu jedné sekundy. Samozřejmě lze využívat podobné techniky jako při programování v C++, ať už v podobě složených datových typů, či funkcí využívajících např. reference, nebo ukazatele. [3]

Pro úspěšné nahrání programu do MCU je nutné propojit vývojovou desku přes převodník, např. CP2102 (viz. kapitola 5.4), s počítačem pomocí kabelu USB. Dále nastavit správný typ vývojové desky, číslo portu, popř. další nastavení a stisknout tlačítko „nahrát“, nebo použít klávesovou zkratku „ctrl+U“. Pokud nejsou nalezeny chyby, je provedeno sestavení (převod do strojového hexadecimálního kódu) a nahrání do flash paměti MCU. [23]

6.2 Nastavení rozhraní v jazyku Wiring

Pomocí funkce `pinMode()` deklarované v prvním bloku `setup()` se nastavují piny rozhraní GPIO. Funkce `pinMode()` přijímá dva argumenty, první určuje číslo pinu a druhý směr dat, přičemž vstup je definován klíčovým výrazem „`INPUT`“ a výstup klíčovým výrazem „`OUTPUT`“. Na vstup je možné softwarově přidat pull-up rezistor zapsáním argumentu „`INPUT_PULLUP`“. Z výše uvedeného vyplývá, že zápis `pinMode(10,OUTPUT)` provede nastavení pinu č. 10 na digitální výstup. Pro čtení stavu digitálních vstupů se používá funkce `digitalRead()` s argumentem určujícím číslo pinu čteného vstupu. Dále jsou k dispozici funkce `analogRead()` a `analogWrite()` pro načítání a zápis hodnoty analogového vstupu a výstupu, funkce přijímají jako argument číslo pinu. [3] Rozsah hodnot analogových vstupů, nebo výstupů je definován parametry ADC, nebo DAC převodníku. Na MCU ESP32 se nacházejí dva 8bitové výstupy DAC a až osmnáct 10bitových ADC vstupů. [4] Sériová rozhraní jsou definovány specifickými piny (viz kapitola 5.2) a pro své funkce používají předpřipravené knihovny.

6.3 Knihovny pro Arduino

Knihovny jsou naprogramované části kódu v jazyku C++, které poskytují rozšířené funkcionality, ať už pro stávající MCU, nebo připojené periférie, jako jsou např. modul RTC s DS3231, karta microSD, nebo již zmíněný modul PZEM-004T. Knihovny jsou nejčastěji tvořeny dvěma soubory, jedním hlavičkovým s koncovkou *.h, který obsahuje deklarace proměnných a funkce s úrovněmi přístupu (public, private) a druhý soubor s koncovkou *.cpp, který obsahuje kód deklarovaných funkcí v prvním souboru *.h. Soubory *.h a *.cpp jsou uloženy v jedné složce se stejným názvem pod adresářem „libraries“. Pomocí klíčového výrazu `#include` s připojeným názvem hlavičkového souboru je do zdrojového kódu

(obvykle hlavní soubor *.ino) provedeno zahrnutí dané knihovny. Následně je zavoláním konstruktoru v hlavním kódu vytvořena instance třídy, na které je možné volat všechny její veřejné funkce a vlastnosti.

V základních knihovnách, které jsou součástí instalace prostředí IDE se nacházejí některé standardní funkce jako např. `Serial.println()`, což je funkce třídy `Serial` umožňující výpis řádku textu do ladící konzole vývojového prostředí. Ostatní knihovny musejí být dodatečně nainstalovány a zahrnuty do zdrojového kódu. Na internetu je volně k dispozici značné množství hotových knihoven pro nepřeberné množství funkcí. Knihovny lze upravovat, nebo vytvářet vlastní, příklad jednoduché knihovny, která sčítá celočíselné datové typy je uveden na následujícím obrázku. Pokud je nutné vytvářet vlastní knihovnu pro určitý hardware, je nutná znalost adres stavových a řídících registrů, přičemž jejich hodnoty lze dohledat v dokumentaci výrobce příslušného zařízení.

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

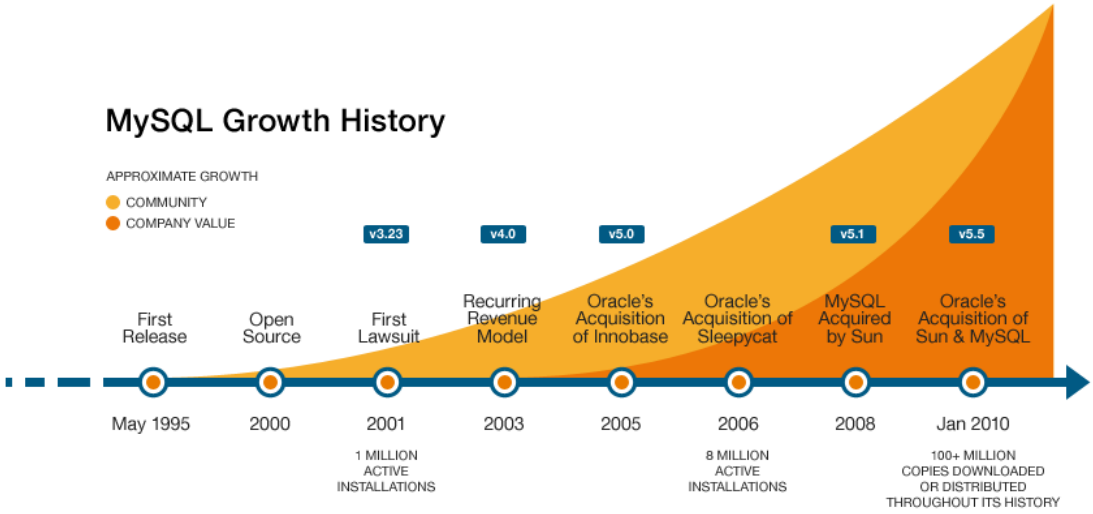
```
#include <mat.h>
Funkce mat; //zavoláním konstruktoru je vytvořena instance
void setup() {
  Serial.begin(9600);
  Serial.print(mat.secti(2, 3)); //vypíše do konzole IDE součet 5 (2+3)
}

void loop() {
}
```

Obrázek 15: *Příklad knihovny mat.h pro sčítání dvou čísel*

Zdroj: vlastní zpracování

6.4 MySQL

Jedná se o databázový systém s otevřeným kódem (open-source) pro správu dat v relačních databázích. MySQL je možné provozovat téměř ve všech operačních systémech založených na UNIX, nebo Windows. Díky své spolehlivosti, výkonosti a relativně snadnému používání patří v kategorii open source mezi nejrozšířenější databázové systémy. [24] MySQL používají společnosti jako např. Facebook, Netflix, Tesla, Nokia, Apple a mnoho dalších, jejich aktuální seznam lze nalézt na stránkách mysql.com. Původně byl navržen a vytvořen firmou MySQL AB v roce 1995, poté odkoupen firmou Sun Microsystems, která od roku 2010 spadá pod společnost Oracle. [25] Následující obrázek zobrazuje historický vývoj MySQL.



Obrázek 16: *Historický vývoj MySQL*

Zdroj: **[25]**

Databázový systém se skládá z databáze a systému řízení báze dat, přes který je umožněna manipulace s daty. Data jsou reprezentována hodnotami uloženými v tabulkách. Řádky tabulky představují jednotlivé záznamy a sloupce jejich atributy. Tabulka obsahuje položky jednoho typu, které mohou být provázány s položkami jiné tabulky přes cizí klíč, přičemž každý záznam v dané tabulce by měl být jednoznačně identifikovatelný přes primární, náhradní, nebo složený klíč. [26] Komunikace s databázovým systémem je realizována

pomocí strukturovaného dotazovacího jazyka SQL z prostředí příkazové řádky, nebo grafického rozhraní (aplikace Workbench).

6.5 Jazyk PHP

Skriptovací multiplatformní programovací jazyk s objektově orientovaným přístupem, v době psaní této práce dostupný ve verzi 8.0. Jedná se o jazyk interpretovaný na straně serveru, tzv. server-side, používaný při tvorbě interaktivních webových stránek, či aplikací, např. je v něm naprogramován CMS Wordpress a v jeho odvozené podobě také podstatná část Facebooku. [27] Pro jazyk PHP vzniklo mnoho frameworků, mezi nejznámější patří např. Laravel, Symfony, CacePHP, Phalcon [28], nebo český Nette, na kterém bylo vytvořeno mnoho soukromých a komerčních projektů. [27] Lze ho nalézt u většiny poskytovatelů webhostingu jako podporu k databázovým systémům MySQL, nebo PostgreSQL. [29] Příklad naprogramované funkce v jazyku PHP, umožňující získat data z databáze MySQL a kódovat je do formátu JSON, je zobrazen na následujícím obrázku.

```php
<?php
function getRecord($dateTime, $numMach){
    $mysqli = mysqli_connect(SERVER, USER, PASSWORD, DB);
        if($mysqli){
        $query = "select numMach, dateTime from recordrs where  numMach = ? and dateTime > ?;";
        $stmt = $mysqli->prepare($query);
        $stmt->bind_param('is',$numMach,$dateTime);
        if(!$stmt->execute()){
            echo json_encode(array("error" => $mysqli->errno));
            exit();
        }
        $result = $stmt->get_result();
        $success = mysqli_affected_rows($mysqli);
        $mysqli->close();
        if($success > 0){
            $data = array();
            while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
                $data[] = $row;
            }
            $result->close();
            print json_encode($data);
            return true;
        }
        else return false;
    }
    else return false;
}
?>
```

Obrázek 17: *Předání dat z databáze do JSON pomocí PHP*

Zdroj: vlastní zpracování

6.6 JavaScript

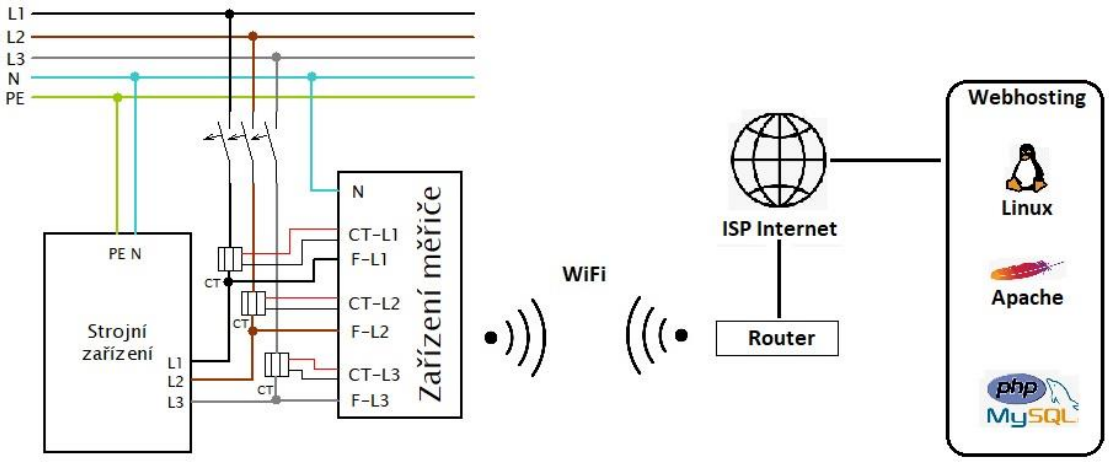
Objektové orientovaný skriptovací jazyk, který umožňuje dodat webovým stránkám prvky dynamiky, jako jsou např. automatické doplňování polí formulářů (např. při zadávání vyhledávacího řetězce na stránkách Google), vyskakovací okna, dynamické přidávání, nebo odebírání HTML prvků za běhu, či vykreslování grafů. Používá tzv. DOM (Document Object Model), což je aplikační programové rozhraní umožňující přistupovat k jednotlivým prvkům a uzlům webové stránky za účelem jejich čtení, nebo změny. JavaScript je prováděn na straně klienta, využívá výkon stroje, na kterém běží a nemusí být znovu načítán při každém obnovení stránky. Jazyk poskytuje funkcionalitu pro asynchronní načítání, zkráceně nazvanou AJAX (Asynchronous JavaScript and XML). Pokud je při zpracování skriptu odeslán požadavek na server, či zařízení, které má delší odezvu, zpracování skriptu nečeká na odpověď a pokračuje dál. Ve chvíli kdy je požadavek zpracován, je spuštěna funkce zpětného volání, která dokončí požadované operace s daty odpovědi. AJAX v kombinaci s DOM umožňuje aktualizovat získaná data ze serveru bez nutnosti opětovného načítání celé webové stránky. [30] [31]

6.7 HTML a CSS

HTML (Hypertext Markup Language) je značkovací popisný jazyk, který umožňuje vytvářet webové prezentace. Jazyk byl původně vytvořen pro sdílení a prezentace výzkumu v laboratořích CERN v roce 1989 a jeho zakladatelem je Tim Berners-Lee. V době psaní této práce byl k dispozici ve verzi 5. Jazyk definuje strukturu a význam obsahu webové prezentace, přičemž vzhled je formátován pomocí kaskádových stylů CSS (Cascading Style Sheets). Základem jsou elementy poskládané do logické struktury webového dokumentu. Elementy jsou tvořené značkami (tagy), uvnitř kterých jsou zapsány hodnoty a atributy. Pro vývoj zdrojového kódu postačuje libovolný textový editor. Jazyk je spravován a normován konsorciem W3C. [32] Na stránkách https://www.w3schools.com/ je možné nalézt pravidla a ukázky použití všech značek HTML včetně tutoriálů na programování jazyků jako např. PHP, Python, JavaScript, Java, SQL, a dalších.

7 Měřič spotřeby

Navrhovaný měřič spotřeby elektrické energie by měl navíc měřit činný výkon, elektrický proud jednotlivých fází a dobu chodu stroje. Pro budoucí rozšiřování je nutné ukládat další veličiny, jako jsou účiník (odděleně pro každou fázi), nebo elektrické napětí. Požadovaná odchylka měření by měla dosahovat maximální hodnoty 0,5 % ve srovnání s běžně instalovanými elektroměry (třída přesnosti 0,5). Zařízení by mělo mít kompaktní rozměry pro zabudování do rozvaděče strojního zařízení, nejlépe na lištu DIN 35. Dílčí výpočty je nutné provádět samotným zařízením a výsledky odesílat bezdrátově do úložiště sítě internet, popř. intranet. Bezdrátový přenos musí být umožněn i v situaci, kdy je zařízení stíněno ocelovou konstrukcí stroje, či jeho rozvaděče. Pro případ nečinnosti musí být zařízení vybaveno hardwarem pro samočinné odpojení od zdroje elektrické energie. Pokud by došlo k výpadku spojení s databázovým systémem, je nutné naměřená data dočasně uložit a při obnovení komunikace odeslat. Aktuální stav každého zařízení musí být možné posoudit vizuálně v místě provozu. Nezbytnou podmínkou je umožnění snadné montáže a konfigurace zařízení bez nutnosti složitějších operací, jako je např. přehrávání softwaru. Měřiče spotřeby by měly být namontovány na stroje CNC Mazak Integrex 100Y (16,2 kVA), Brother TC225 (9 kVA) a kompresor Boge (max. 15,5 kVA) a následně odzkoušeny všechny jejich uvažované funkcionality. Schéma měření a předávání dat do databáze je znázorněno na následujícím obrázku.



Obrázek 18: *Schéma měření a předávání ze zařízení*

Zdroj: vlastní zpracování

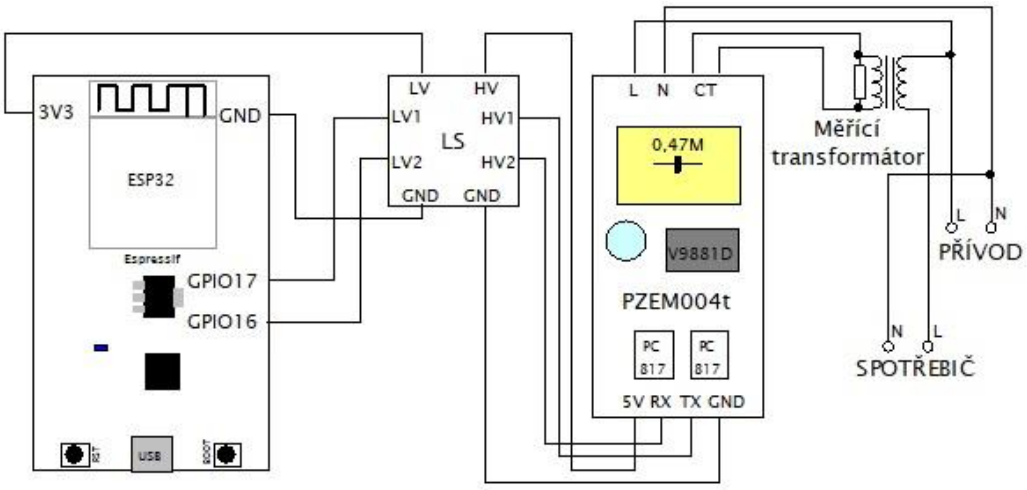
7.1 Návrh měřiče spotřeby

Měření v třífázové soustavě je provedeno třemi samostatnými moduly PZEM-004. Jako hlavní modul je vybrán ESP32, který vzhledem k ceně a zabudovanému hardwaru odpovídá požadavkům na vytvářené zařízení. Komunikace mezi měřícími moduly a mikrokontrolerem je realizována přes rozhraní UART. Zapojení rozhraní bylo upraveno tak, aby bylo možné použít pouze jedno UART MCU pro připojení všech třech modulů PZEM-004 (viz kapitola multidrop UART). Pro síťové napájení zařízení byl vybrán spínaný zdroj s výstupním napětím 5 V a minimálním proudem 1 A. Vzhledem k možným vibracím stroje je samočinné odpínání provedeno bezkontaktně za pomoci elektronického relé LR7843 s tranzistorem MOSFET. Záložní napájení zařízení po vypnutí stroje je zajištěno UPS, tvořenou modulem TP4056, step-up měničem MT3608 a akumulátorem li-ion 18650. Pro dočasné datové zálohy a konfigurační data byla do zařízení přidána karta microSD se čtečkou připojenou přes rozhraní SPI. Vzhledem k možnosti dodatečné synchronizace dat bylo nutné ke každému naměřenému záznamu vložit údaj o čase a datu měření, proto zařízení měřiče obsahuje modul pro uchování reálného času DS3231 propojený přes rozhraní I^2C. Signalizace stavu zařízení je realizována dvěma LED připojenými na výstupní piny MCU, zelená pro stav zařízení v provozu (spuštěný program) a oranžová signalizující připojení k databázovému systému. Bezdrátový přenos dat zajišťuje rozhraní WiFi posílené externí anténou. Zařízení je zabudováno do plastové krabičky vytisknuté na 3D tiskárně za pomoci materiálu PETG. Měřič spotřeby posílá v minutových intervalech data, jako jsou číslo stroje, čas a datum záznamu, napětí, celkový činný výkon, maximální proud, průměrný účiník a spotřebu v každé fázi. Naměřená data jsou ukládána do tabulky databáze MySQL.

7.2 Zapojení a odzkoušení modulu PZEM-004

Vzhledem k uvedeným požadavkům a ceně byl vybrán modul PZEM-004, který je navržen pro použití s měřícími transformátory a umožňuje měřit elektrický proud až 100 A. Modul byl odzkoušen na jedné fázi v rámci běžně dostupných spotřebičů elektrické energie, v tomto případě se jednalo o nabíječku notebooku (cca 130 W), která disponuje spínaným zdrojem el. energie způsobující fázový posun, resp. nízký účiník (viz kapitola o měření činného výkonu el. energie). Rozdílné logické napětí mezi měřícím modulem a mikrokontrolerem ESP32 vyžadovalo použití převodníku napěťových úrovní. Modul PZEM-004 byl připojen na piny HV (log. napětí 5 V) a MCU ESP32 na piny LV (log. napětí 3,3 V). Z důvodu odzkoušení měřících transformátorů bylo nutné provést provizorní silové připojení s rozděleným fázovým a nulovým vodičem.

Pro odzkoušení modulu byl vytvořen zdrojový kód, do kterého byla vložena knihovna ze stránky https://www.arduinolibraries.info/libraries/pzem004-tv30, ta obsahuje funkce, jako jsou zahájení a inicializace spojení po sériové lince, načítání požadovaných elektrických veličin, ověření správnosti dat (kontrolní součet CRC), nastavení a získání adresy zařízení, či vynulování měřiče. Nejprve bylo provedeno zapojení modulu podle blokového schématu (viz následující obrázek) a následně probíhalo zkušební měření s periodickým výpisem klíčových veličin (každých 10 sekund) do konzole vývojového prostředí Arduino IDE. Uvedené funkce byly odzkoušeny a ověřeny jednodenním provozem bez zjevných chyb.

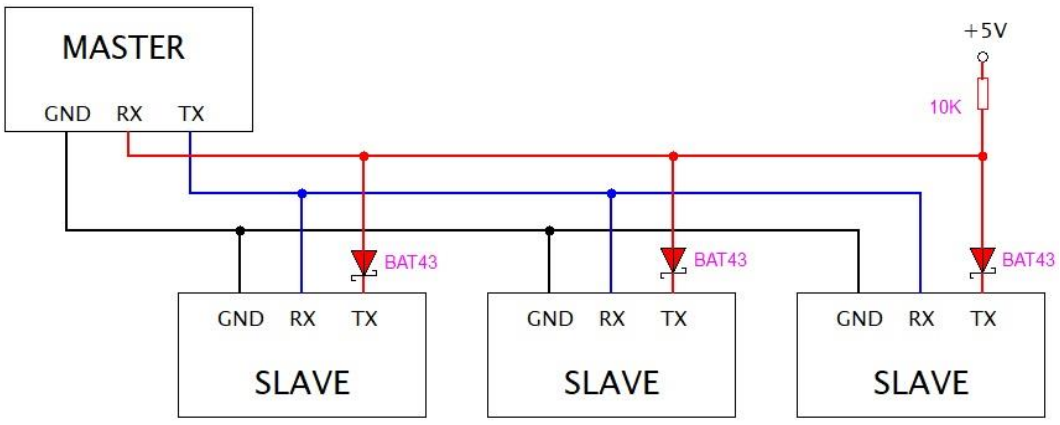


Obrázek 19: *Schéma zapojení modulu PZEM -004T a ESP32*

Zdroj: vlastní zpracování

7.3 Multidrop UART

Mikrokontroler ESP32 obsahuje 3 sériové porty rozhraní UART, přičemž jeden z těchto portů je pro komunikaci mezi PC a MCU namapován na převodník CP2102. Vzhledem k zachování funkce pro nahrání programu do MCU, či výpisu do ladící konzole přes USB, bylo nutné vyřešit komunikační propojení měřících modulů jednotlivých fází s jedním portem UART. V úvahu přicházely dvě možnosti, buď pomocí přídavného obvodu provádět multiplexování, nebo využít sdružené zapojení, tzv. multidrop. Nejprve byla vyzkoušena první varianta pomocí rychlého multiplexeru CMOS HC4052, která vykazovala značnou nestabilitu, kdy docházelo k časté ztrátě dat z některých modulů PZEM-004T. Druhá varianta s Schottkyho diodami a pull-up rezistorem byla sestavena a odzkoušena několikahodinovým testováním bez sebemenších poruch. Vzhledem k úspěšnému testu, jednodušší konstrukci a nižší ceně, byla vybrána druhá varianta, tzn. multidrop. Blokové zapojení sběrnice v této konfiguraci je znázorněno na následujícím obrázku.



Obrázek 20: *Schéma zapojení sběrnice UART multidrop*

Zdroj: vlastní zpracování

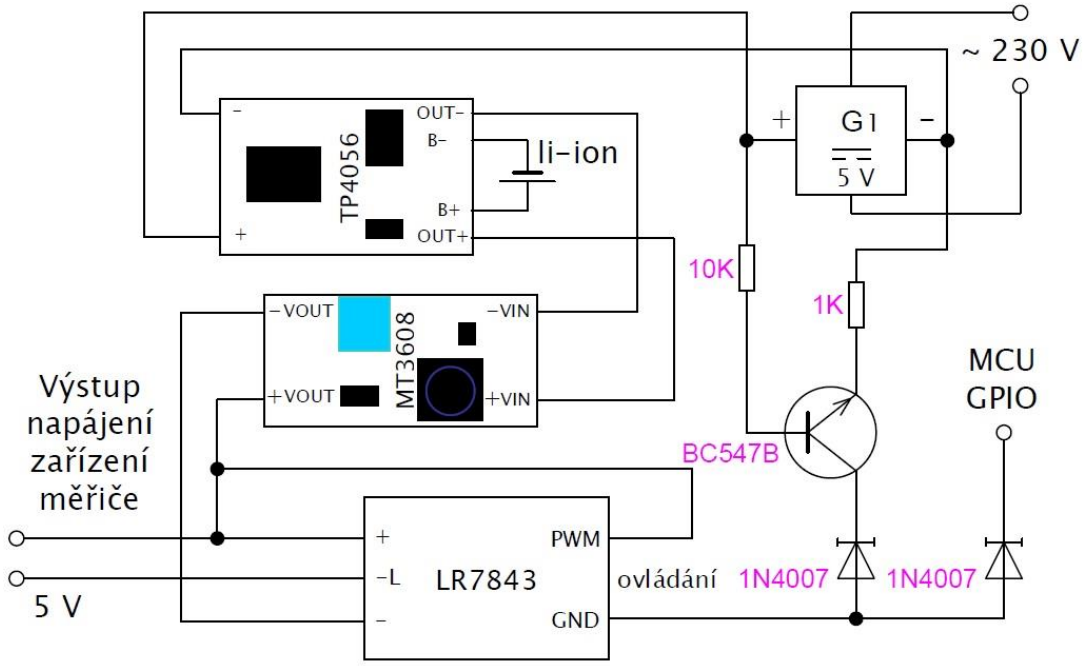
Aby nedocházelo ke vzájemným kolizím zařízení slave při odesílání dat do zařízení master, je nutné oddělit vývody TX pomocí Schottkyho diod. Při požadavku zařízení master je komunikace řízena nastavením odlišných adres v zařízeních slave. Nejprve je jednotkou master odeslán požadavek s adresou zařízení slave pro získání dat z 1. fáze. Jakmile je adresa

rozeznána daným zařízením slave, začne přes Shottkyho diodu probíhat přenos jednotlivých datových bitů (po lince TX od slave) do zařízení master. V okamžiku ukončení přenosu je linka uvolněna a probíhá požadavek na zařízení pro 2. fázi a nakonec 3. fázi. Mezi jednotlivými požadavky je nastaveno cca 100 milisekund zpoždění. Nastavení adresy bylo provedeno funkcí `setAddress()`, která je obsažena v knihovně PZEM-004Tv30 (viz odkaz v minulé kapitole). Pro úspěšné nastavení adresy je nutné připojit jednotku PZEM-004T na silové straně (napájení MCU VD9881) a samozřejmě na straně UART s logickou úrovní 5 V.

7.4 UPS jednotka s řízeným vypínáním zátěže

Pro zařízení měřiče spotřeby energie byla navržena jednotka UPS, která se také osvědčila v projektu meteostanice (dodnes funkční na stránkách arduinopj01.cz). Propojením modulu TP4056 s akumulátorem byl vytvořen základ UPS poskytující výstupní napětí odpovídající aktuálnímu napětí akumulátoru. Jelikož napětí vybraného li-ion akumulátoru kolísá v rozmezí 2,8-4,2 V, byla UPS doplněna o modul step-up měniče MT3608, který umožňuje zvýšit napětí na vyšší konstantní hodnotu. Během nastavování a oživování modulu nesmí nastat situace, kdy je na vstupu vyšší napětí, než na výstupu, jinak dojde k okamžitému zničení obvodu MT3608. Před uvedením do provozu je doporučeno nastavit trimr naplno doleva a teprve potom připojit přívod, dále provést nastavení na požadovanou hodnotu a nakonec připojit zátěž.

UPS jednotka poskytuje napájení po dobu, kdy jsou zařízením měřiče ještě dokončovány výpočty a přenosy dat do databáze. Po dokončení těchto operací je signálem výstupního pinu provedeno vypnutí celého měřiče. Výstupní pin bylo nutné posílit na požadovanou proudovou zátěž, která může při náběhu přesahovat hodnotu 1 A. Vzhledem k uvedeným požadavkům, proudové zátěži a ceně, byl zvolen obvod s tranzistorem MOSFET LR7843. Tranzistor potřebuje pro spolehlivou funkci řídící napětí min. 5 V, proto je ovládací výstup MCU připojen navíc přes převodník napěťových úrovní. Při zkoušení buzení tranzistoru pouze napětím 3,3 V nebylo možné přes jeho silový přechod přenést požadovanou zátěž. Pro nastavení nižšího budícího napětí by bylo nutné upravit napěťový dělič hradla tranzistoru. Nevýhodou úpravy napěťového děliče by byla vyšší pracnost a časová náročnost proti pouhé výměně (odpojení a připojení modulu na konektor). Blokové schéma UPS jednotky, včetně řízení vypínání zátěže, je zobrazeno na následujícím obrázku.



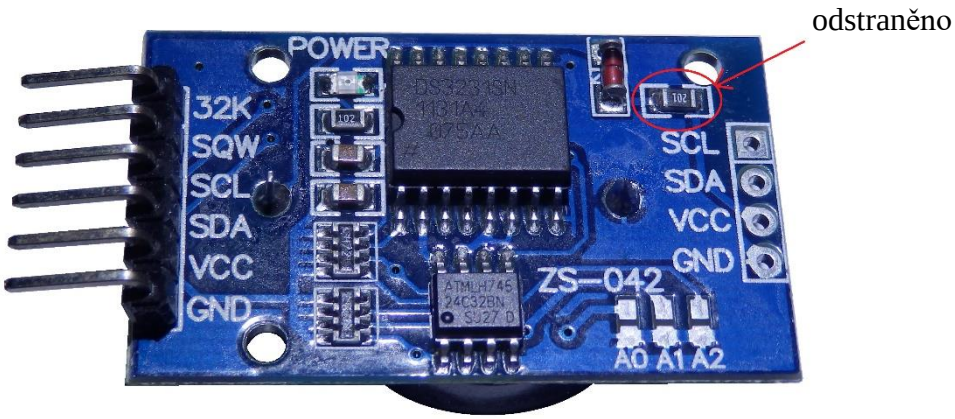
Obrázek 21: *Schéma zapojení UPS s řízeným vypínáním*

Zdroj: vlastní zpracování

Po přivedení síťového elektrického napětí (zapnutí stroje) je na výstupních svorkách zdroje G1 vytvářeno stejnosměrné napětí 5 V, které je přiváděno přes rezistor 10 K na bázi tranzistoru BC547B. Mezi bází a emitorem (připojen k nulovému potenciálu) tranzistoru začne protékat proud, který způsobí otevření přechodu kolektor-emitor. Moduly TP4056 a MT3608 mají průchozí kladný potenciál, proto otevřením přechodu kolektor-emitor vznikne napětí na ovládacích svorkách (PWM, GND) modulu LR7843, který následně spustí celé zařízení měřiče. Pokud dojde k vypnutí stroje, je ovládací napětí udržováno výstupním pinem MCU až do požadované chvíle, kdy je vyslán signál k odepnutí. Z důvodu oddělení dvou odlišných signálů bylo nutné doplnit obvod o diody 1N4007.

7.5 Úprava jednotky RTC

Pokud nebude dostupná síť, nebo databáze, je nutné naměřená data dočasně uložit v souboru paměťové karty zařízení měřiče. Po úspěšném spojení jsou data z paměťové karty odesílána do tabulky databáze, tzn. že pokud by byly jednotlivé záznamy opatřeny časovým razítkem až v databázi, docházelo by k jejich špatnému označení, protože byly pořízeny dříve, než v době zápisu do databáze. Z výše uvedených skutečností vyplývá, že je nutné udržovat reálný čas a datum v zařízení měniče, proto byl instalován modul DS3231. Pro uchování správných hodnot data a času i po odpojení zdroje elektrické energie je modul vybaven záložní baterií CR2032. Původní zapojení modulu je navrženo pro nabíjecí akumulátor LIR2032. Navržené nabíjení je zcela nevhodné pro daný typ akumulátoru, neobsahuje žádné ochrany, ale pouze rezistor, který při rostoucím napětí akumulátoru snižuje nabíjecí proud. Bez jakýchkoliv úprav by mohlo dojít k poškození akumulátoru a následně dalším materiálním škodám, proto byl odstraněn rezistor nabíjení (viz uvedený obrázek) a vložena klasická baterie CR2032. Při odběrů zařízení 4µA vychází životnost klasické baterie CR2032 na cca 5 let.

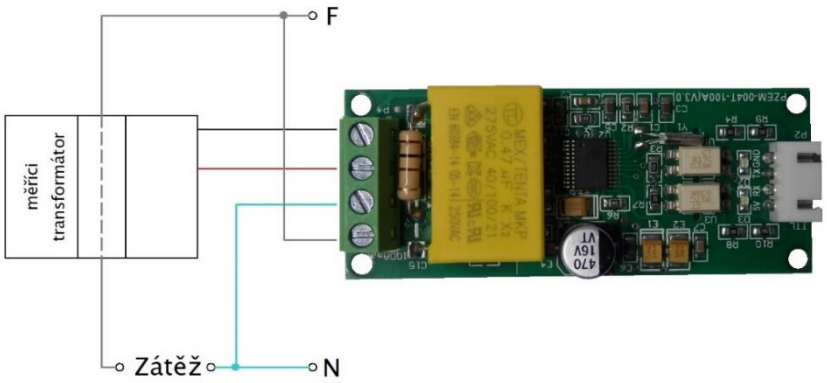


Obrázek 22: *Úprava DS3231 pro baterii CR2032*

Zdroj: vlastní zpracování

7.6 Odzkoušení celkového obvodu měřiče

Po otestování a úpravách dílčích částí zařízení měřiče bylo nutné provést celkové zapojení a jeho odzkoušení. Použitý převodník napěťových úrovní obsahuje 4 kanály, 2 byly použity pro komunikaci s moduly PZEM-004T (RX, TX) a jeden pro zvýšení úrovně ovládacího napětí automatického vypínání zařízení. Napájení zdroje měřiče bylo připojeno k 1. fázi za měření tak, aby bylo možné zahrnout i vlastní spotřebu. Signalizace provozu, resp. úspěšná inicializace všech klíčových částí po spuštění měřiče je napojena na pin GPIO26 (zelená LED) a signalizace připojení k databázovému systému na pin GPIO27 (oranžová LED). Stejnou svítivost signalizačních led bylo nutné odladit velikostí předřadných rezistorů, 220 ohm pro zelenou LED a 2,2 k ohm pro oranžovou LED. Modul DS3231 a karta microSD vyžadují dle datového listu výrobce napětí 3,3 V, a proto byly napojeny na vývod lineárního stabilizátoru AMS1117 (součást modulu ESP32), který poskytuje požadované napětí. Zkušební zapojení modulu a dalších elektronických součástek pro měření všech tří fází bylo provedeno pomocí nepájivého pole a propojovacích vodičů s konektory DuPont. Dle dostupných návodů zapojení modulu PZEM-004T je možné použít dvě varianty. Pro zmenšení oblasti s výskytem nebezpečného napětí byla použita varianta, která je uvedena na následujícím obrázku. Druhá varianta má prohozený nulový a fázový vodič a vzhledem k navrženému plošnému spoji rozšiřuje oblast s nebezpečným napětím. Takto sestavený obvod byl připojen k měřené zátěži asynchronního třífázového elektromotoru o výkonu 1,5 kW a úspěšně otestován po dobu jedné hodiny.



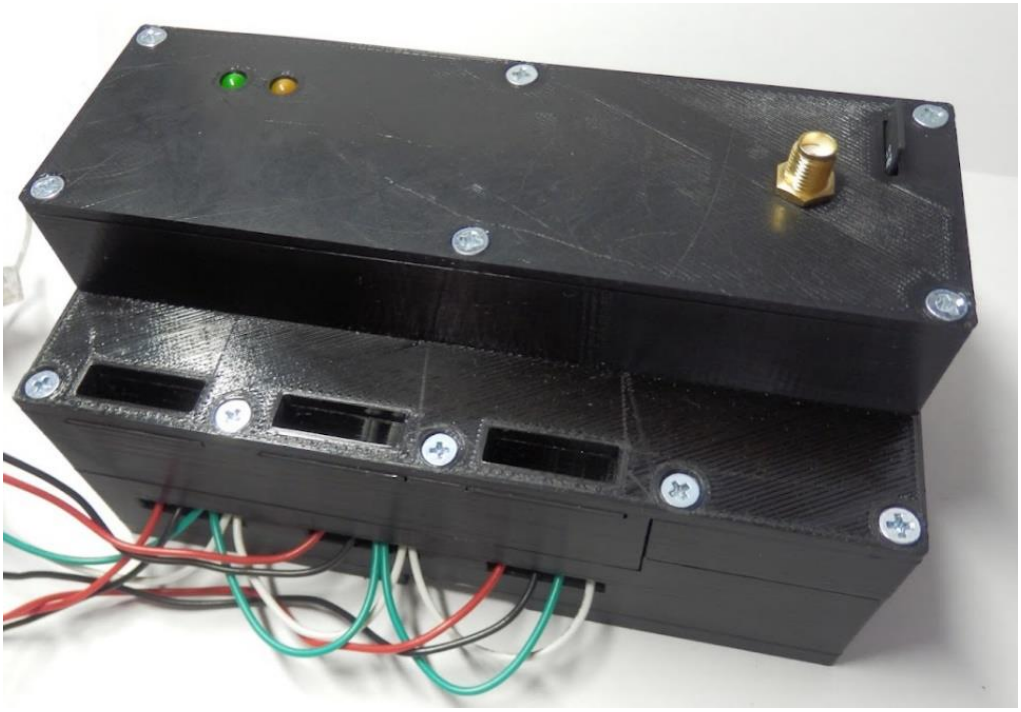
Obrázek 23: *Zapojení měřících obvodu modulu PZEM-*

Zdroj: vlastní zpracování

7.7 Hardwarové uspořádání

Veškerou elektroniku zařízení bylo nutné zabudovat do krabičky, která bude plnit několik funkcí, jako jsou ochrana před nebezpečným dotykem živých částí, ochrana elektronických komponent a snadná montáž do stroje na lištu DIN 35. Dostupné elektroinstalační krabičky neumožňovaly uspořádat jednotlivé komponenty tak, aby bylo dosaženo vhodné velikosti pro umístění do rozvaděče. Také bylo nutné vyřešit dispozici svorek měřících modulů, budoucí přístup k paměťové kartě a konektoru microUSB k nahrávání softwaru.

Všechny uvedené požadavky byly vyřešeny vlastním návrhem krabičky vytištěné na 3D tiskárně za pomoci materiálu PETG. Krabička byla rozdělena do tří sekcí, spodní sekce obsahuje silové měřící obvody a napájecí zdroj, v prostřední je zabudován obvod UPS s akumulátorem, modul RL7843, převodník napěťových úrovní, RTC a plošný spoj se součástkami pro multidrop sběrnici a řízení napájení. Mikrokontroler a paměťová karta jsou z důvodu snadného přístupu umístěny ve vrchní části zařízení. Krabička byla navržena v softwaru Autodesk Fusion 360 a vytištěna na tiskárně Creality CR10.



Obrázek 24: *Krabička zařízení měřiče spotřeby*

Zdroj: vlastní zpracování

7.8 Cenová kalkulace materiálu

V následující tabulce je uveden soupis materiálu pro stavbu měřiče spotřeby včetně jeho ceny. Součástky a moduly byly zakoupeny v létě 2021 na tuzemském trhu. Od asijských prodejců byly dodány pouze moduly PZEM-004T. Množství PETG je odhadnuto na základě počtu vyrobených kusů na 1 kg materiálu. Drobný materiál jako vodiče, elektronické součástky, konektory, aj., je zahrnut v položce elektronika. Celková cena materiálu činila 1867 Kč vč. DPH.

položka	počet	cena/ks	prodejce
ESP32	1	180	hadex.cz
DS3231	1	65	laskarduino.cz
LR7843	1	28	laskarduino.cz
logický převodník	1	14	laskarduino.cz
TP4056	1	24	laskarduino.cz
MT3608	1	24	laskarduino.cz
li-ion 18650	1	198	laskarduino.cz
držák 18650	1	18	laskarduino.cz
zdroj 5 V/2 A	1	78	laskarduino.cz
PZEM-004t-100A + CT	3	267	ebay.com
čtečka microSD	1	26	laskarduino.cz
karta microSD (64 GB)	1	156	alza.cz
elektronika	-	50	tme.eu
konektory	-	80	laskarduino.cz
PETG	-	100	plastymladec.cz

Tabulka 2: *Rozpis položek včetně jejich ceny*

Zdroj: vlastní zpracování

7.9 Návrh a implementace zdrojového kódu měřiče spotřeby

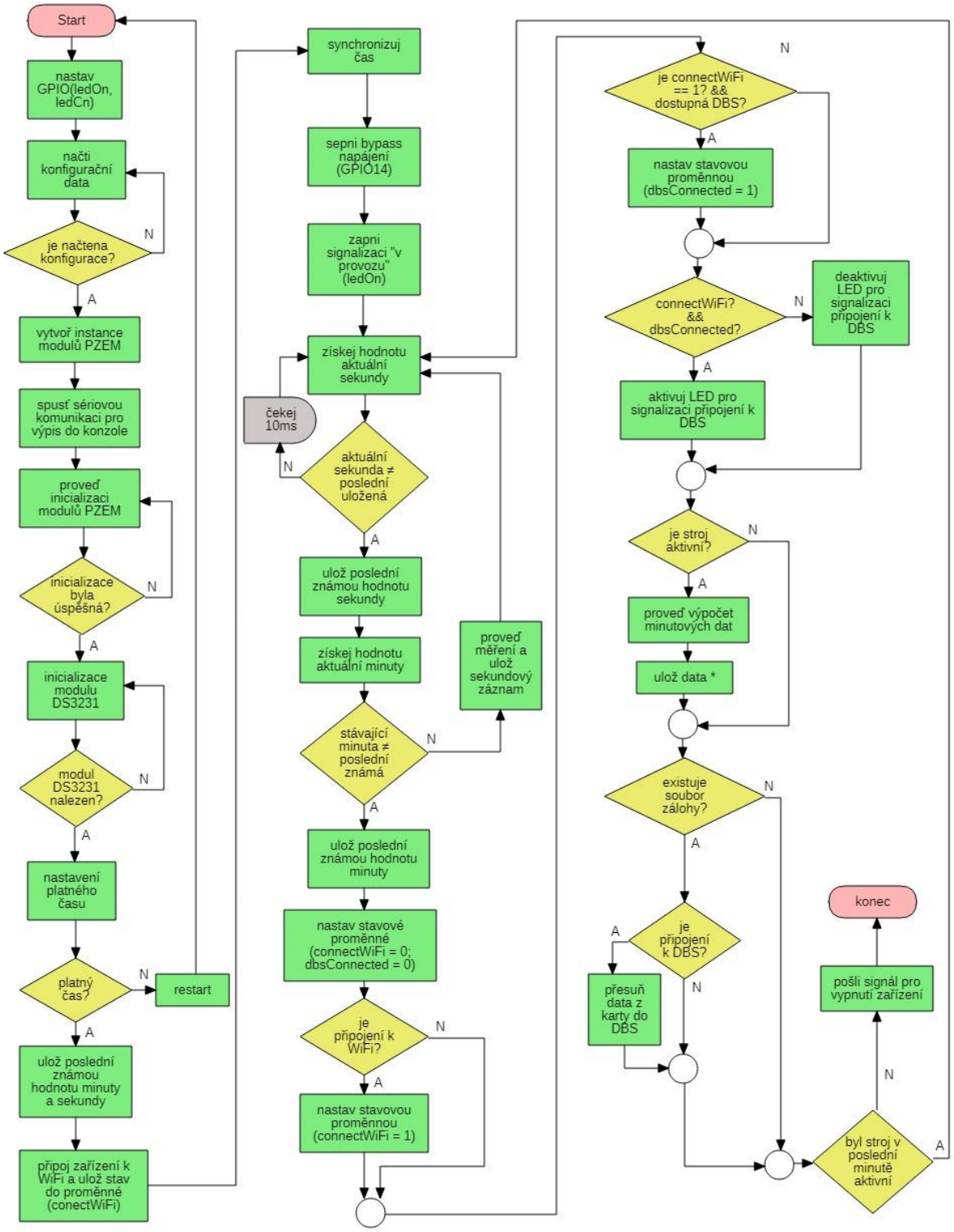
Pro lepší přehlednost a budoucí rozšiřování byl zdrojový kód rozdělen do čtyř dílčích částí. První část poskytuje funkcionality, jako jsou připojení k WiFi, kontrola spojení s databázovým systémem a odesílání dat na server. Druhá část obsahuje funkce umožňující pracovat s modulem DS3231 včetně detekce letního času, nebo synchronizace času přes server NTP. Třetí část zajišťuje veškeré operace s jednotkami PZEM-004T, jako jsou získání měřených veličin, počítání minutových průměrů a maxim, kontrola aktivity měřícího modulu, nebo kontrola platného rozsahu získaných dat. Poslední čtvrtá část obsahuje funkce čtení a zápisu paměťové karty. Uvedené funkce jsou volány z kódu hlavního souboru energy_meter.ino, ve kterém jsou zahrnuty knihovny uvedené v následující tabulce.

pořadí	název knihovny	autor	popis
1.	PZEM004Tv30.h	Jakub Mandula	funkce jednotky PZEM-004T*
2.	WiFi.h	Ivan Grokhotkov	bezdrátový přenos dat
3.	NTPClient.h	Fabrice Weinberg	komunikace s NTP serverem*
4.	WiFiUdp.h	Bjoern Hartmann	přenos dat protokolem UDP
5.	SPI.h	Hristo Gochkov	komunikace přes rozhraní SPI
6.	SD.h	Espressif Systems	čtení a zápis paměťové karty
7.	HTTPClient.h	Markus Sattler	přenos dat protokolem http
8.	RTClib.h	by JeeLabs	komunikace s DS3231*
* knihovny nejsou součástí standardního balíčku instalace pro MCU ESP32			

Tabulka 3: *Zahrnuté knihovny zdrojového kódu měřiče*

Zdroj: vlastní zpracování

Nejprve byla vytvořena část pro práci s jednotkami PZEM-004T, poté zprovozněna komunikace s databázovým systémem a nakonec přidán kód pro modul RTC a paměťovou kartu. Do datové struktury `dataPzemx` jsou v intervalu jedné sekundy ukládány naměřené hodnoty. Každou minutu jsou uložená data zpracována (průměry, maxima, odečty spotřeby) a odeslána na server, nebo uložena do souboru paměťové karty. Na následující stránce je zobrazen vývojový diagram zařízení měřiče spotřeby.



Obrázek 25: *Vývojový diagram měřiče spotřeby*

Zdroj: vlastní zpracování

Proces „ulož data“ je řízen parametrem, který určuje aktuální uložiště (paměťová karta, nebo tabulka databáze). Data jsou odesílány metodou POST s nastaveným časovým limitem 5 sekund. Pokud jsou data úspěšně vložena do databáze, server odešle do zařízení měřiče kladnou odpověď. Pro komunikaci mezi zařízením a databázovým systémem bylo naprogramováno rozhraní v jazyku PHP, které umožňuje vkládání záznamů, upload z paměťové karty a test připojení k databázovému systému.

Upload dat z paměťové karty do databáze probíhá po blocích (max. 20 záznamů), přičemž je hlídán stanovený časový limit (max. 20 sekund). Oba časové limity lze volit jako parametry funkce `readFile()`. Jestliže vyprší časový limit, je uložena pozice rozečteného souboru a předáno řízení hlavní smyčce programu, upload pokračuje od uložené pozice až po provedení klíčových procesů (měření a uložení).

Konfigurační data jsou uložena na paměťové kartě v souboru CONFIG.TXT, první řádek představuje číslo stroje, druhý a třetí řádek identifikátor sítě WiFi včetně jejího hesla, čtvrtý řádek adresu serveru (v tomto případě arduinopj01.cz) a poslední řádek určuje, zdali budou data přenášena po zabezpečeném kanálu (http, nebo https). Pro úspěšné načtení konfiguračních dat je nutné převést textové řetězce (datový typ String) na statické pole znaků (char[x]) tak, aby byl odstraněn ukončovací znak řetězce „/n“.

Vyhodnocení stavu zařízení (připojení k WiFi, připojení k databázovému systému, existence dočasné zálohy) je prováděno v rámci minutového intervalu a ukládáno do stavových proměnných tak, aby nedocházelo k opakovanému volání některých funkcí.

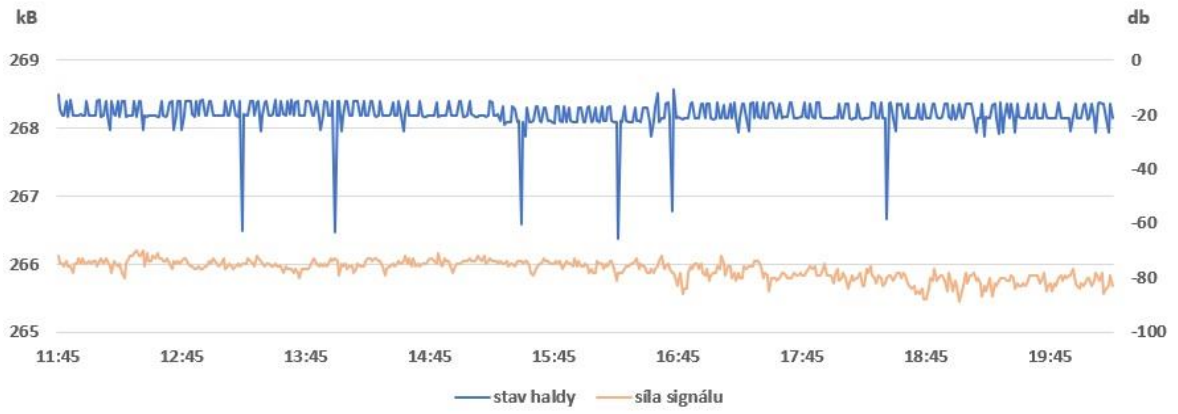
7.10 Zkušební provoz a odladění kódu

První měřič spotřeby byl nainstalován do strojního zařízení Mazak Integrex 100Y na začátku srpna 2021, kdy databáze obsahovala pouze tabulku pro sběr dat.

V prvních dnech docházelo k posílání zkreslených hodnot, např. naměřený proud na 1. fázi přesahoval 200 A, přičemž nepomohla ani funkce zdrojového kódu pro kontrolu validity dat. Revizí kódu byla odhalena příčina v podobě přepisování paměti jinými daty. Původní detekce nové minuty spočívala v kontrole nulového stavu sekund, ale v situaci menšího zpoždění (cca 1 sekunda) docházelo k občasnému přeskočení tohoto údaje. Pokud nebyla detekována nová minuta, program pokračoval v zápisu dalších dat, což způsobilo překročení velikosti definovaného pole pro naměřené hodnoty a jejich přetečení do jiné části paměti. Náprava zahrnovala implementaci ochrany proti přetečení datového pole a vylepšení detekce

nové minuty, která porovnává aktuální minutu s poslední uloženou. Kromě dvou užití byl z kódu vyřazen složený datový typ String, který způsoboval zbytečnou fragmentaci paměti.

Další závada se projevovala formou samovolných restartů, nebo úplným zastavením zařízení v nepravidelných intervalech, v řádech několika hodin až dnů. Vzhledem k charakteru závady byla dodatečně navržena a implementována funkce diagnostiky, která ukládá hodnoty klíčových parametrů (datum a čas, připojení, síla signálu WiFi, volná paměť a trvání uploadu) do souboru LOG. Data ze souborů byla vyhodnocována pomocí softwaru Excel ve formě grafů, na kterých bylo patrné, že dochází k úbytku volné paměti, tzv. „haldy". Postupným zjednodušováním původního kódu byla chyba identifikována v knihovně SD.h, která si po připojení karty vytvořila novou instanci, ale již neprovedla její odebrání při ukončení komunikace s kartou. Uvedená chyba je velmi dobře popsána na adrese https://github.com/espressif/arduino-esp32/commit/4a0305a0 a její oprava spočívá v přidání jednoho řádku kódu. Po této opravě se již volná paměť stabilizovala na hodnotě cca 268 kB a nedocházelo k uvedeným poruchám, jak dokazuje graf na následujícím obrázku. Takto odladěný kód byl použit v dalších dvou zařízeních instalovaných během měsíce září 2021.



Obrázek 26: *Diagnostický graf volné paměti a síly signálu*

Zdroj: vlastní zpracování

7.11 Ukázka části zdrojového kódu zařízení

Na následujícím obrázku je uvedena část zdrojového kódů zařízení měřiče. Umožňuje načíst konfigurační data ze souboru CONFIG.TXT uloženém na paměťové kartě. Metodou `SD.begin()` je nejprve zahájena komunikace s SD kartou, další funkce `SD.open()` otevře konfigurační soubor pro čtení. Pokud jsou obě předchozí funkce úspěšně provedeny (vrátí logickou hodnotu true), následuje cyklus pro načítání řádků souboru. Funkce `readStringUntil()` třídy `File` přiřadí řetězec celého řádku do proměnné `configItem` a předá ji do funkce `conv_char()`, která provede převod na pole znaků čitelné funkcemi pro připojení k WiFi a databázovému systému. Funkce `atoi()` slouží k převodu čísla reprezentovaného polem znaků, např. [‘1‘,‘2‘] je převedeno do celočíselné hodnoty 12. Při úspěšném načtení konfiguračních dat vrátí funkce `load_config()` číslo 0.

```
int load_config() {
  //zahájí komunikaci s SD kartou
  if (!SD.begin()) {
    SD.end();
    return 2;
  }
  //poté otevře soubor s configurací "CONFIG.TXT"
  File conf = SD.open("/CONFIG.TXT");
  if (!conf) {
    conf.close();
    SD.end();
    return 3;
  }
  String configItem;
  configItem.reserve(32);
  for (int i = 0; i < 5; i++) {
    //zde začíná číst konfigurační soubor....
    configItem.remove(0);
    configItem = conf.readStringUntil('\n');
    conv_cha(configItem, confData[i]);
  }
  conf.close();
  SD.end();
  numMach = atoi(nmch);
  return 0;
}
```

Obrázek 27: *Ukázka části zdrojového kódu měřiče*

Zdroj: vlastní zpracování

8 Webová aplikace

Webová aplikace by měla zobrazovat podrobné statistiky každého stroje a jejich celkové souhrny ve formě grafů a textových údajů, příp. dalších výpočtu dle časového intervalu a stroje. Získaná data by měla být využívána pro vedení elektronického deníku, který bude evidovat veškeré důležité informace o dané výrobě. Záznamy elektronického deníku by měly využívat klíčových parametrů nastavení výroby, uvedených v následující tabulce.

parametr	jednotky
amortizace stroje	Kč/h
režim standby	W
práh sazby	W
míra využití podpůrného stroje	%
cena elektrické energie	kWh
obsluha	login
cena seřizování(nastavování)	Kč/h
cena obrábění	Kč/h
váha materiálu	kg
cena materiálu	Kč/kg
číslo zakázky	-

Tabulka 4: *Parametry pro záznamy elektronického deníku*

Číslo zakázky by mělo umožňovat dohledání podrobných údajů o zákazníkovi (IČO, DIČ, název, jméno, příjmení a kontaktní informace). Na základě záznamů elektronického deníku a zadání vstupů, jako jsou č. zakázky, stroje, výrobku, nebo zadání časového rozsahu, by aplikace měla zobrazovat podrobné přehledy o nákladech. Pro každého uživatele by mělo být vytvořeno rozhraní umožňující změna hesla. V pilotní fázi aplikace se předpokládají dvě uživatelské role, administrátor a uživatel s omezeným přístupem (statistiky jednotlivých strojů, zadávání záznamů do elektronického deníku). Přístup do aplikace bude zabezpečen pomocí hesla a uživatelského jména.

8.1 Návrh aplikace

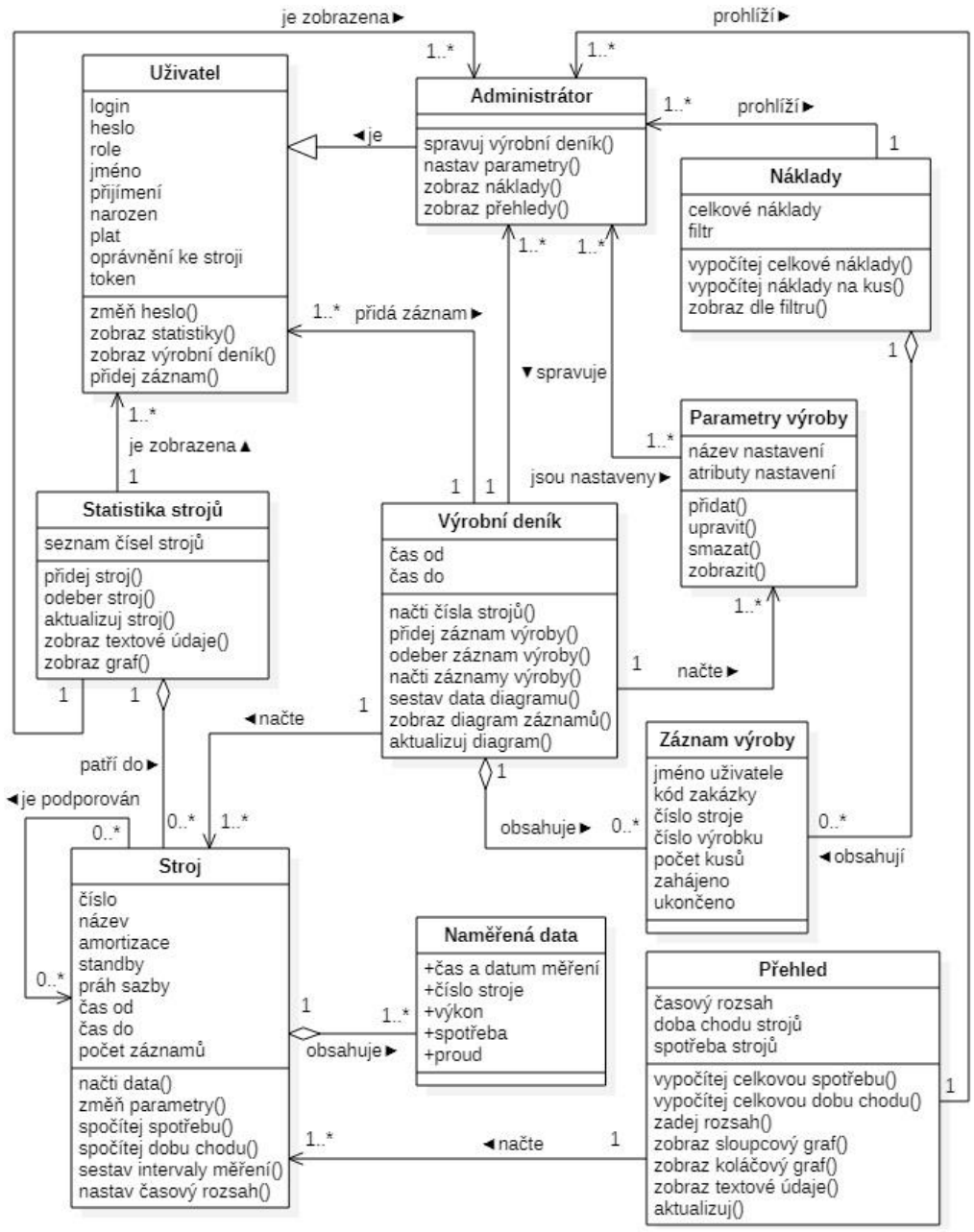
Pro práci s naměřenými daty byla vytvořena webová aplikace využívající třívrstvé architektury. Datovou vrstvu představuje databázový systém MySQL s databází, ve které jsou uložená data aplikace. Prostřední aplikační vrstva zajišťuje komunikaci s databázovým systémem MySQL pomocí API mysqli, které je voláno z kódu jazyka PHP. V tomto jazyku jsou také naprogramovány metody umožňující většinu výpočtů a operací s daty tak, aby nejvyšší vrstva zprostředkovávala pouze prezentaci a zadávání vstupů od uživatele. Výše uvedené vrstvy představují serverovou část aplikace. Klientská část aplikace tvoří nejvyšší prezentační vrstvu a je naprogramována v kódu jazyka HTML doplněného o styly CSS. Dynamiku klientské vrstvy poskytuje kód jazyka JavaScript a jeho frameworky. V době psaní této práce byla webová aplikace provozována na webhostingu firmy Onebit.cz.

Celé uživatelské rozhraní bylo navrženo a vytvořeno jako jednostránková webová aplikace, tzv. SPA (Single Page Application), která je rozdělena do několika záložek. Pro aktualizaci dat je využívána technologie asynchronního načítání AJAX v kombinaci s přenosovým formátem JSON, tzn. že jsou aktualizovány pouze relevantní položky, nikoliv celá webová stránka. Záložky webové stránky představují jednotlivé sekce, jako jsou statistiky výrobních strojů, záznamy výroby, nastavení parametrů, přehledy o výrobě, náklady na výrobu a informace o aktuálním uživateli. Uvedená struktura a technologie umožňují snadný pohyb v rámci celé aplikace a přispívají k lepšímu UX. Pro snadnější programování klientské části v JavaScript byly použity frameworky jako např. jQuery, nebo Highcharts. Vzhledem k dostupnému hardwaru a požadavkům byla aplikace odladěna v prohlížečích Google Chrome a Microsoft Edge, další rozšiřování počítá s prohlížečem Safari.

Přístup do aplikace je zabezpečen uživatelským jménem a heslem. Byly implementovány dvě úrovně přístupu. Úroveň s nižším oprávněním umožňuje zobrazení statistik jednotlivých strojů a zadání záznamu výroby s ohledem na oprávnění k danému stroji. Vyšší úroveň rozšiřuje přístup do sekce přehledů, nákladů výroby a nastavení, navíc umožňuje úpravu již zadaného záznamu výroby. Každá operace s databázovým systémem je autorizována ověřením hodnoty odpovídajícího parametru cookies s hodnotou atributu v tabulce uživatelů, to samé platí pro zobrazování jednotlivých sekcí.

8.2 Struktura webové aplikace

Diagram struktury je zachycen na následujícím obrázku. Nejedná se přímo o implementační variantu, ale o doménový model s doplněnými metodami a dalšími podrobnostmi.

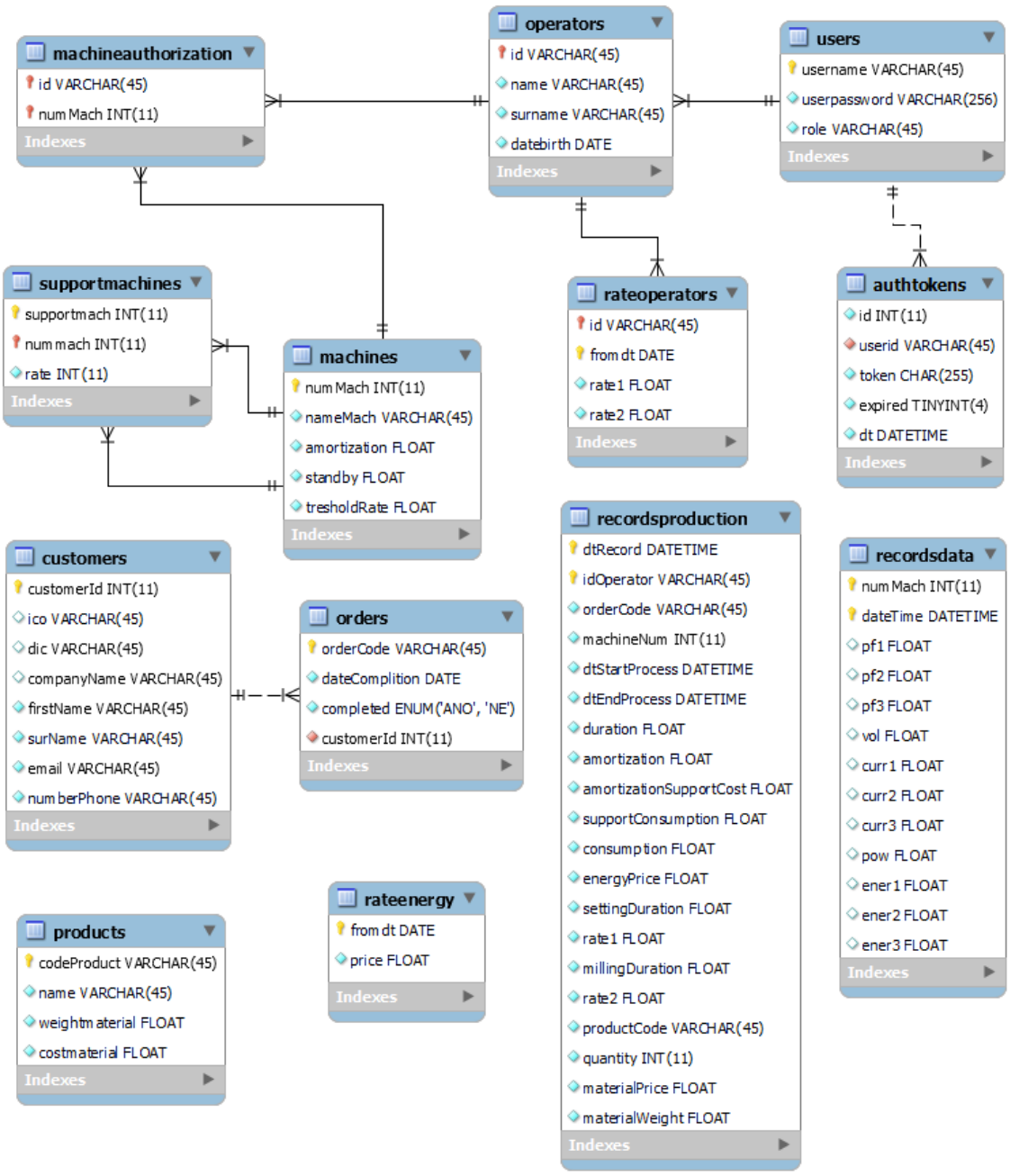


Obrázek 28: *Doménový model struktury webové aplikace*

Zdroj: vlastní zpracování

8.3 Návrh a vytvoření databáze MySQL

Návrh databáze byl proveden pomocí software MySQL Workbench 8.0. Na následujícím obrázku je zobrazen relační datový model, podle kterého byla databáze vytvořena.



Obrázek 29: *Relační datový model*

Zdroj: vlastní zpracování

Některé tabulky databáze nejsou vzájemně provázány a slouží pouze jako úložiště dat, jedná se o naměřená data (recordsdata), záznamy výroby (recordsproduction), výrobky (products) a cenu elektrické energie (rateenergy). Zachování referenční integrity je provedeno pomocí cizích klíčů, např. není možné odstranit uživatele, který existuje jako obsluha, má přidělené oprávnění a plat. Mnohonásobná vazba ve vztahu stroje a obsluhy (stroj může obsluhovat více pracovníků a naopak) je doplněna o vazební tabulku, která přiřazuje oprávnění obsluhy k danému stroji. Vzhledem k tomu, že stroj může být také podpůrným strojem (kompresor, separátor emulze, atd.) pro jeden, nebo více strojů a naopak, byla k tabulce strojů připojena vlastní vazební tabulka podpůrných strojů (supportmachines). Vazba mezi zákazníkem a objednávkou není povinná, protože v okamžiku přidání nového zákazníka ještě nemusí být vytvořena zakázka, podobně jako ve vztahu uživatele a autorizačního tokenu, který vzniká až po prvním úspěšném přihlášení. Pro pilotní provoz aplikace byly data o zákaznících zahrnuty do jedné tabulky, další vývoj počítá s dekompozicí a doplněním tabulky kontaktů a adres zákazníků.

8.4 Aplikační vrstva

Aplikační vrstva byla naprogramována v kódu jazyka PHP a jeho rozhraní API mysqli. Rozhraní zpracovává řetězce s odpovídajícími SQL dotazy pro operace, jako jsou vložení, čtení, aktualizace, nebo odstranění dat. Dotazy SQL jsou napsány tak, aby byly získány pouze podstatná data a nedocházelo k navyšování objemu přenášených dat mezi serverem a klientem, např. pro nalezení minima a maxima v daném časovém období jsou využívány funkce databázového systému `min()`, nebo `max()`, které vrátí výsledek. Takto získané výsledky jsou kódovány do formátu JSON a odeslány klientské části aplikace. V záhlaví souboru PHP je vždy vložen kód, který nastaví typ obsahu, parametry hlavičky pro odpověď, údaje pro připojení k databázovému systému a zahrne script umožňující autorizaci konkrétního požadavku. Parametry jsou z klienta do skriptu PHP odesílány metodou POST a následně ve skriptu vloženy do dotazu metodou `bind_param()`. Na následujícím obrázku je ukázka funkce `getChartData()` obsahující metody třídy rozhraní `mysqli` k získání dat pro graf statistiky stroje.

```php
//načte data pro grafické zobrazení statistik daného stroje
function getChartData($numMach, $from, $to){
    $mysqli = mysqli_connect(SERVER, USER, PASSWORD, DB);
        if($mysqli){
        $query = "select
                        round(pow,2) pow,
                        round(curr1,2) curr1,
                        round(curr2,2) curr2,
                        round(curr3,2) curr3,
                        dateTime
                    from
                        namerenadata,
                        (select standby from machines where numMach = ? limit 1) mch
                    where
                        numMach = ? and pow > mch.standby
                    and dateTime between ? and ?;";
        $stmt = $mysqli->prepare($query);
        $stmt->bind_param('iiss',$numMach,$numMach,$from,$to);
        if(!$stmt->execute()){
            echo json_encode(array("error" => $mysqli->errno));
            exit();
        }
        $result = $stmt->get_result();
        $success = mysqli_affected_rows($mysqli);
        $mysqli->close();
        if($success > 0){
            $data = array();
            //načíta jednotlivé záznamy jako asociativní pole
            while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)){
                $data[] = $row;
            }
            $result->close();
            print json_encode($data);
            return true;
        }
        else return false;
    }
    else return false;
}
```

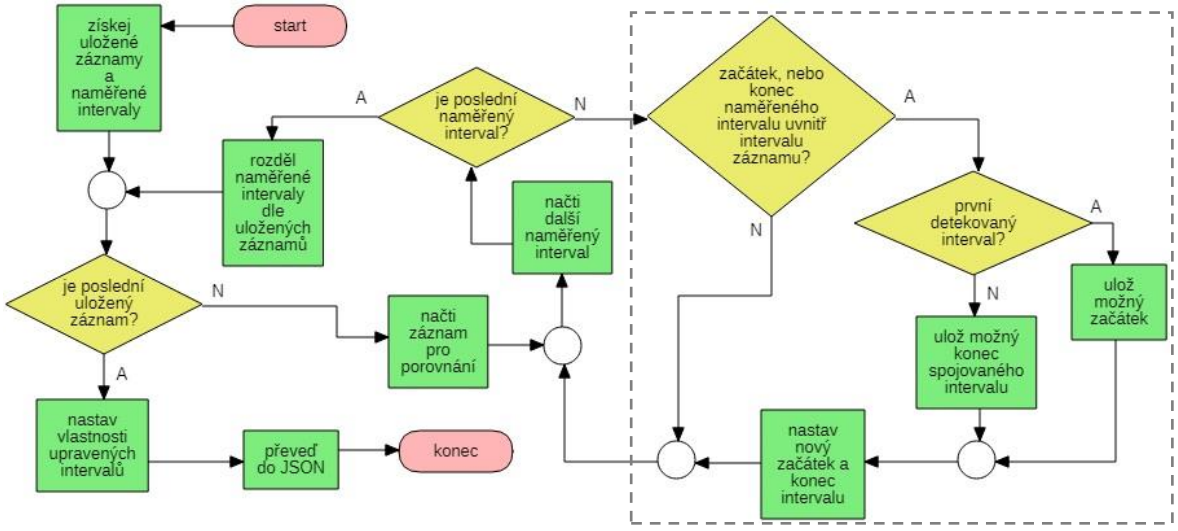
Obrázek 30: *Funkce pro získání dat v jazyku PHP*

Zdroj: vlastní zpracování

Funkce je volána s parametry, které identifikují číslo stroje a požadovaný časový rozsah. Nejprve je vytvořena reference (`$mysqli`) na objekt zprostředkovávající připojení k databázovému systému. Pokud je připojení úspěšné, objekt vrátí logickou hodnotu true a kód pokračuje sestavením dotazu SQL. Sestavení je provedeno metodou `prepare()` a `bind_param()`, která nahradí zástupné symboly „?“ hodnotami předaných proměnných. Poslední uvedená metoda zajišťuje, že jakýkoliv předaný řetězec v parametru dotazu je interpretován jako hodnota parametru a nikoliv jako přímá součást řetězce dotazu,

což zvyšuje odolnost proti útoku SQL injection. Takto sestavený dotaz je proveden metodou `execute()` a pokud je úspěšně vykonán, vrací objekt s požadovanými daty do proměnné `$result`, v opačném případě vrací číslo chyby. Metodou `close()` je spojení ukončeno a pokud byla získána data (`$success > 0`) , následuje jejich převod do formátu JSON. Převod do stejného formátu je proveden i v případě neúspěchu (ř. 147), protože funkce v klientské části vždy očekává datový typ JSON, jiný datový typ by byl vyhodnocen jako obecná chyba komunikace (viz. kapitola Aplikační vrstva).

Uvedené funkce umožňují získat převážnou většinu výsledků pro prezentační vrstvu, další přidané funkce umožňují např. export dat pro zobrazení Ganttova diagramu v elektronickém deníku. Tyto funkce nejprve načtou a sestaví intervaly naměřených dat požadovaného stroje, poté získají již uložené výrobní záznamy a následně provedou jejich porovnání a sestavení. Pokud je nalezeno více naměřených intervalů v rámci uloženého výrobního záznamu, pak dochází k jejích sloučení, v opačném případě k rozdělení. Poslední částí je přidělení patřičných atributů a převod do formátu JSON. Celý proces je znázorněn na následujícím schématu, podrobněji je dekomponován algoritmus pro rozdělování naměřených intervalů (ohraničeno přerušovanou čarou).



Obrázek 31: *Algoritmus přípravy dat Ganttova diagramu*

Zdroj: vlastní zpracování

8.5 Klientská část v JavaScript

Jedná se o část naprogramovanou v kódu jazyka JavaScript a jeho frameworku jQuery, který přidává dynamiku chování uživatelského rozhraní a některé další funkcionality jako např. vytváření tabulek, nebo nastavení vstupů. Na následujícím obrázku je ukázka funkce k získání dat pro grafy a tabulky celkových statistik v rámci zvoleného časového období.

```
167   //získání celkových statistik strojů dle čísla a rozsahu datumu a času
168   function getTotalData(from, to, init){
169       var params = [];
170       params.push({ name: 'select', value: 'totalData' }, { name: 'from', value: from },
171           { name: 'to', value: to }, { name: 'csrftoken', value: csrftoken });
172       $.ajax({
173           type: 'POST',
174           url: './statistics/php/totalStat.php',
175           data: params,
176           async: true,
177           dataType:"json",
178           success: function (data) {
179               if (Object.keys(data)[0] === 'error') {
180                   viewSqlError(response.error);
181                   return;
182               }
183               viewTableTotalStat(data,init);
184               viewChartTotalStat(data);
185           },
186           error: function (err){
187               console.log("Error by read total statistics machines! ", err.responseText);
188           }
189       });
190   }
```

Obrázek 32: *Ukázka kódu v jazyku JavaScript*

Zdroj: vlastní zpracování

Uvedená funkce využívá asynchronního volání AJAX. Předávané argumenty jsou číslo stroje a meze pro zvolený časový rozsah. Nejprve je vytvořeno pole objektů, do kterého jsou uloženy hodnoty předaných argumentů následně odesílaných na server metodou POST. Parametry funkce `$.ajax()` definují způsob předání dat požadavku, adresu skriptu PHP, data požadavku, asynchronní režim, obsah vrácených dat a funkce, které jsou volány v případě úspěchu, či neúspěchu. Pokud databázový systém vyvolá chybu, je uvnitř podmínky (ř. 180) spuštěn kód pro zavolání funkce `viewSqlError(response.error)`, která dle předaného čísla chyby (`response.error`) informuje uživatele. Aby bylo možné získat číslo chyby vyvolané databázovým systémem, je nutné vracet chybu jako datový

objekt JSON, proto je ve skriptu PHP (viz ukázka kódu v předchozí kapitole na ř. 148) kód pro převod dat do požadovaného formátu.

Další funkce klientské části aplikační vrstvy poskytují např. dynamické vytváření tabulek. Na následujícím obrázku je kód využívající framework jQuery, který přidává řádky do tabulek v sekci pro celkové přehledy strojů.

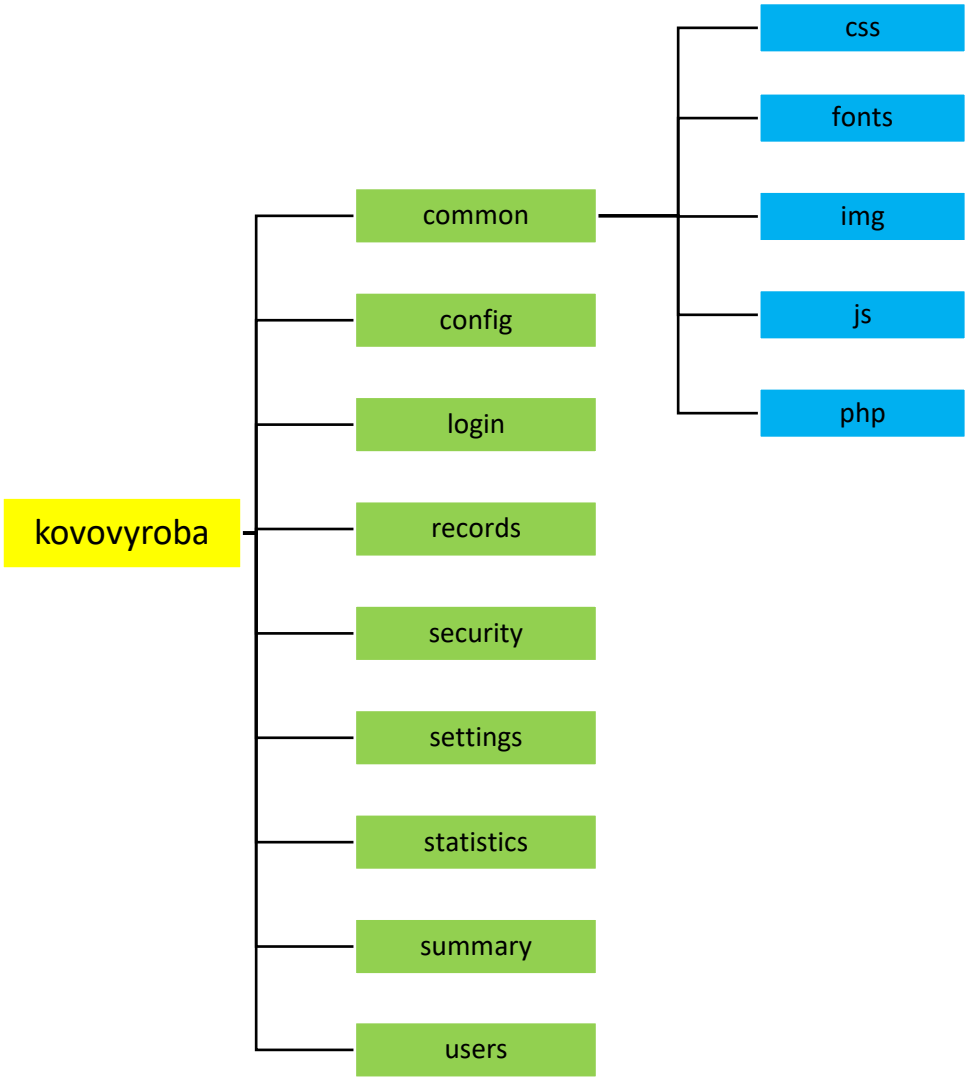
```
22    //přidává řádky tabulky
23    function addRowTableTotalStat() {
24        var table = $('.totalTextStat > div > table');
25        //najde poslední číslo id
26        var temp = table.find('tr:last > td > span').attr('id').split('-');
27        var numId = parseInt(temp[1], 10) + 1;
28        //přidá řádek to tabulky
29        table.find('tr:last').clone().appendTo(table);
30        //přiřadí správná id
31        var newElm = table.find('tr:last');
32        newElm.find("[id^='nm-']").attr('id', 'nm-' + numId);
33        newElm.find("[id^='from-']").attr('id', 'from-' + numId);
34        newElm.find("[id^='to-']").attr('id', 'to-' + numId);
35        newElm.find("[id^='tc-']").attr('id', 'tc-' + numId);
36        newElm.find("[id^='tt-']").attr('id', 'tt-' + numId);
37    }
```

Obrázek 33: *Kód JavaScript pro přidání řádku tabulky*

Zdroj: vlastní zpracování

8.6 Struktura složek serveru FTP

Vzhledem k rozsahu aplikace (cca 7000 řádků bez frameworků), budoucímu rozšiřování a opravám, bylo nutné rozdělit kód do několika logických celků, které se dále dělí na menší části obsahující soubory s kódem, nebo další doplňky jako např. ikony, či obrázky. Na následujícím obrázku je zobrazena stromová struktura složek celé aplikace uložená na serveru FTP.



Obrázek 34: *Stromová struktura složek aplikace*

Zdroj: vlastní zpracování

8.7 Vytváření grafů pomocí knihovny Highcharts

Zobrazení statistik jednotlivých strojů je provedeno přes spojnicové grafy. Celkové statistiky jsou zobrazeny pomocí sloupcového a koláčového grafu. Vložení záznamu do elektronického deníku umožňuje Ganttův diagram, který je upraven tak, aby obsahoval pouze jednu sérii s body definovanými časovým rámcem, přičemž po kliknutí na příslušný bod automaticky vloží do formuláře datum a čas zahájení a ukončení dané výroby. Všechny uvedené grafy jsou generovány prostřednictvím frameworků Highcharts, které jsou bezplatně poskytovány pro osobní, či školní projekty.

V sekci statistiky jsou grafy s textovými údaji daného stroje odebírány, nebo přidávány dynamicky, proto byla vytvořena samostatná funkce generující spojnicové grafy. Funkce přijímá 3 argumenty, první představuje data, druhý číslo stroje a třetí jednoznačný identifikátor elementu webové stránky pro zobrazení vytvořeného grafu. Prvky datového pole se skládají z času měření a hodnot elektrických veličin naměřených v tomto čase. Ukázka použití frameworku Highcharts pro vykreslení grafu je na následujícím obrázku.

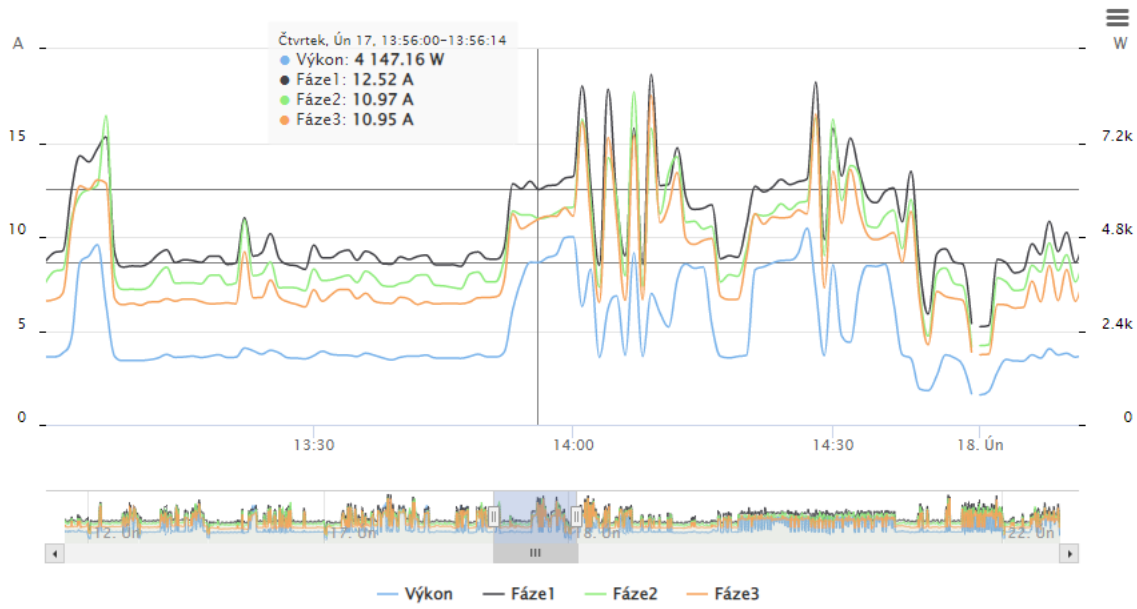
```
684     //vytvoří objekt grafu a vloží ho do elementu dle id
685     function createObjectChart(data, numMach, idElement) {
686         Highcharts.stockChart(idElement, {
687             chart: {
688                 type: 'spline',
689             },
690             series: [{
691                 name: 'Výkon',
692                 yAxis: 0,
693                 data: data[0],
694                 tooltip: {
695                     valueSuffix: ' W',
696                     valueDecimals: 2,
697                 }
698             }]
699         })
700     }
```

Obrázek 35: *Ukázka definování grafu v Highcharts*

Zdroj: vlastní zpracování

Jedná se o velmi zjednodušenou ukázku grafu vykreslujícího pouze křivku výkonu, v reálné webové aplikaci obsahuje 240 řádků kódu, který přidává další nastavení, jako jsou např. vlastnosti hlavních a vedlejších os, popisky, vyšší počet sérií, nastavení hlavní a vedlejší mřížky, přidání navigace, nebo akce pro obsluhu událostí (změna rozsahu hlavní

osy, kliknutí na data grafu a nahrání grafu). Celkový výsledek kompletně nastaveného grafu vytvořené webové aplikace je zobrazen na následujícím obrázku.



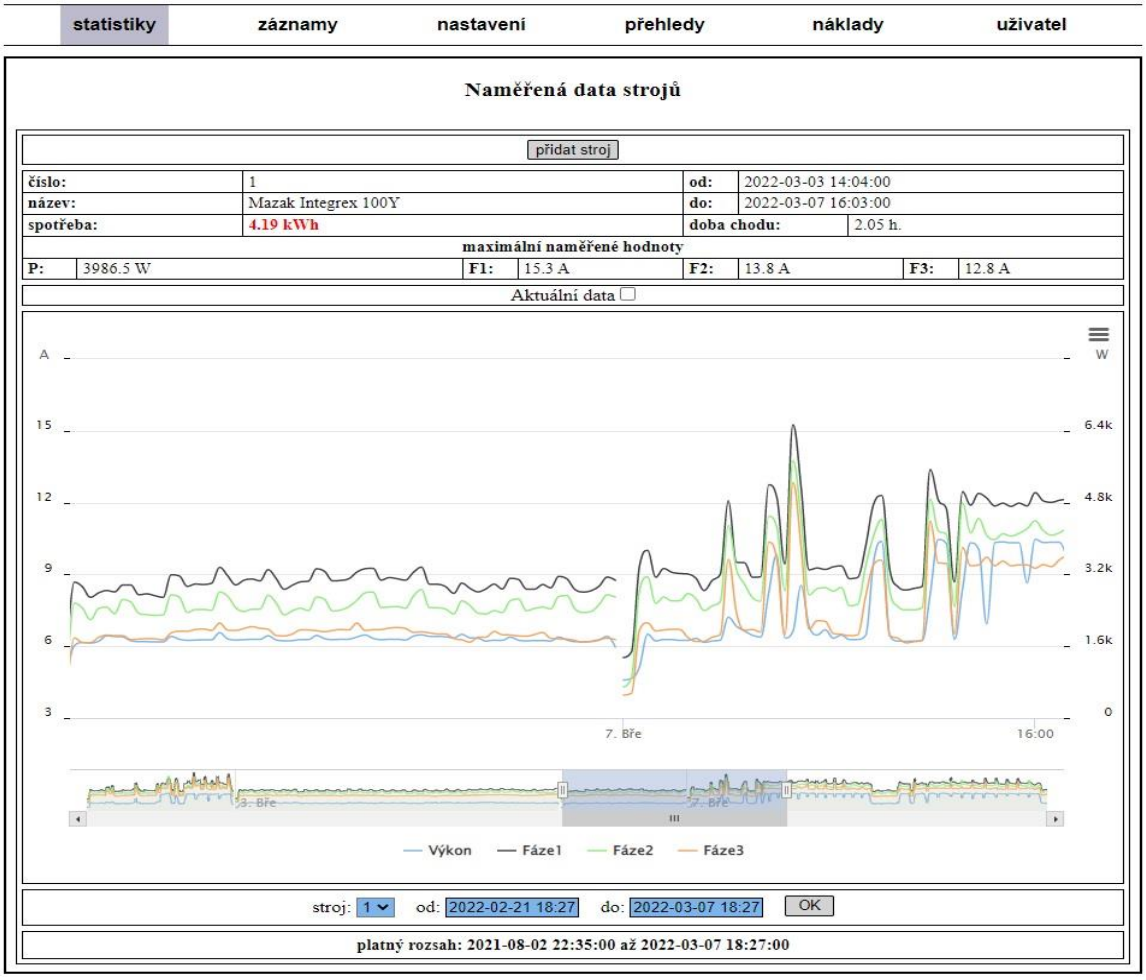
Obrázek 36: *Graf Highcharts pro statistiku stroje*

Zdroj: vlastní zpracování

Na frameworku Highcharts je také založeno grafické uživatelské prostředí pro uložení záznamu výroby. Uživatel pouze klikne na příslušný bod, který přenese rozsah data a času do zadávacího formuláře, v případě kliknutí na již uložený záznam navíc zobrazí kód zakázky, výrobku, id obsluhy a počet kusů. Naprogramování uvedeného způsobu ukládání bylo proti klasickému zadávání (formulář pro datum a čas) složitější, ale jeho přínos spočívá v minimalizaci chyb při zadávání a lepší UX.

8.8 Statistiky jednotlivých strojů

Umožňuje graficky zobrazit průběh činného výkonu a proudu v jednotlivých fázích elektrické sítě. Dále vypočítává a zobrazuje spotřebu a dobu chodu podle aktuálně vybraného období a čísla stroje. Výpočet spotřeby a doby chodu je vázán na nastavený rozsah navigátoru grafu, pokud dojde k jeho změně, jsou obě hodnoty automaticky přepočítány. Přepočet je uskutečněn po dvou sobě následujících událostech, změně rozsahu navigátoru a uvolnění tlačítka myši. Bez vazby na druhou událost by vlivem pohybu táhla neustále docházelo k opakovaným požadavkům na server a zbytečnému nárůstu velikosti přenášených dat. V záložce je také možné zapnout aktualizaci, která při aktivním stroji přidává nové body grafu a aktualizuje hodnoty údajů tabulky každou minutu.

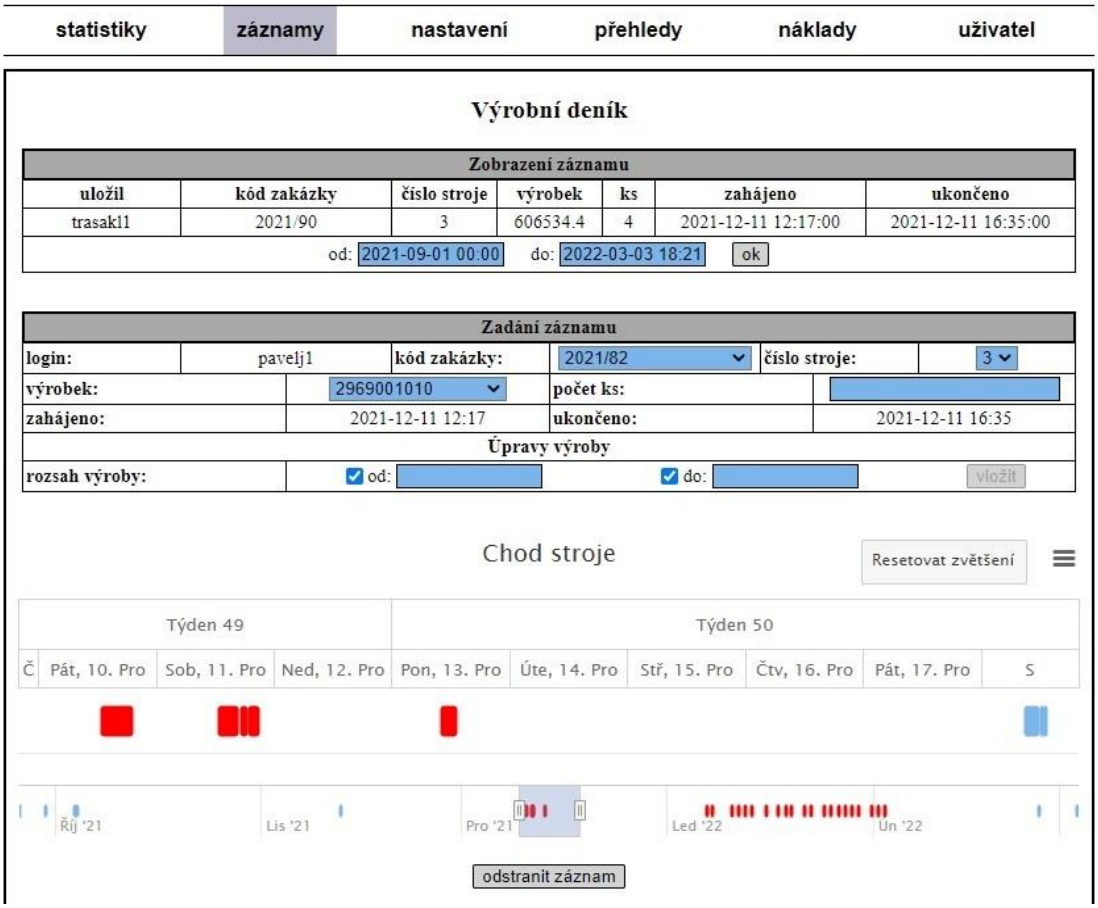


Obrázek 37: *Statistika stroje Mazak Integrex 100Y*

Zdroj: vlastní zpracování

8.9 Elektronický deník

Na této záložce je možné zobrazovat, zadávat a odstraňovat záznamy výroby. Standardní časový rozsah je nastaven na 14 dní zpětně od posledního záznamu vybraného čísla stroje. Kliknutím na bod diagramu je vyplněn časový rozsah výroby do formuláře „Zadání záznamu“, další parametry jako kód zakázky, výrobek, nebo číslo stroje jsou zadávány přes výběrová vstupní pole (modře podbarvené) tak, aby uživatel nemohl vložit neplatné údaje. Po kliknutí na uložený záznam výroby (označen červeně) jsou zobrazeny detaily v horní tabulce s názvem „Zobrazení záznamu“ a pod grafem aktivováno tlačítko pro možné odstranění (pouze role administrátora). Pokud je nutné naměřený záznam před uložením rozdělit, nebo sloučit, je k dispozici sekce „Úpravy výroby“.

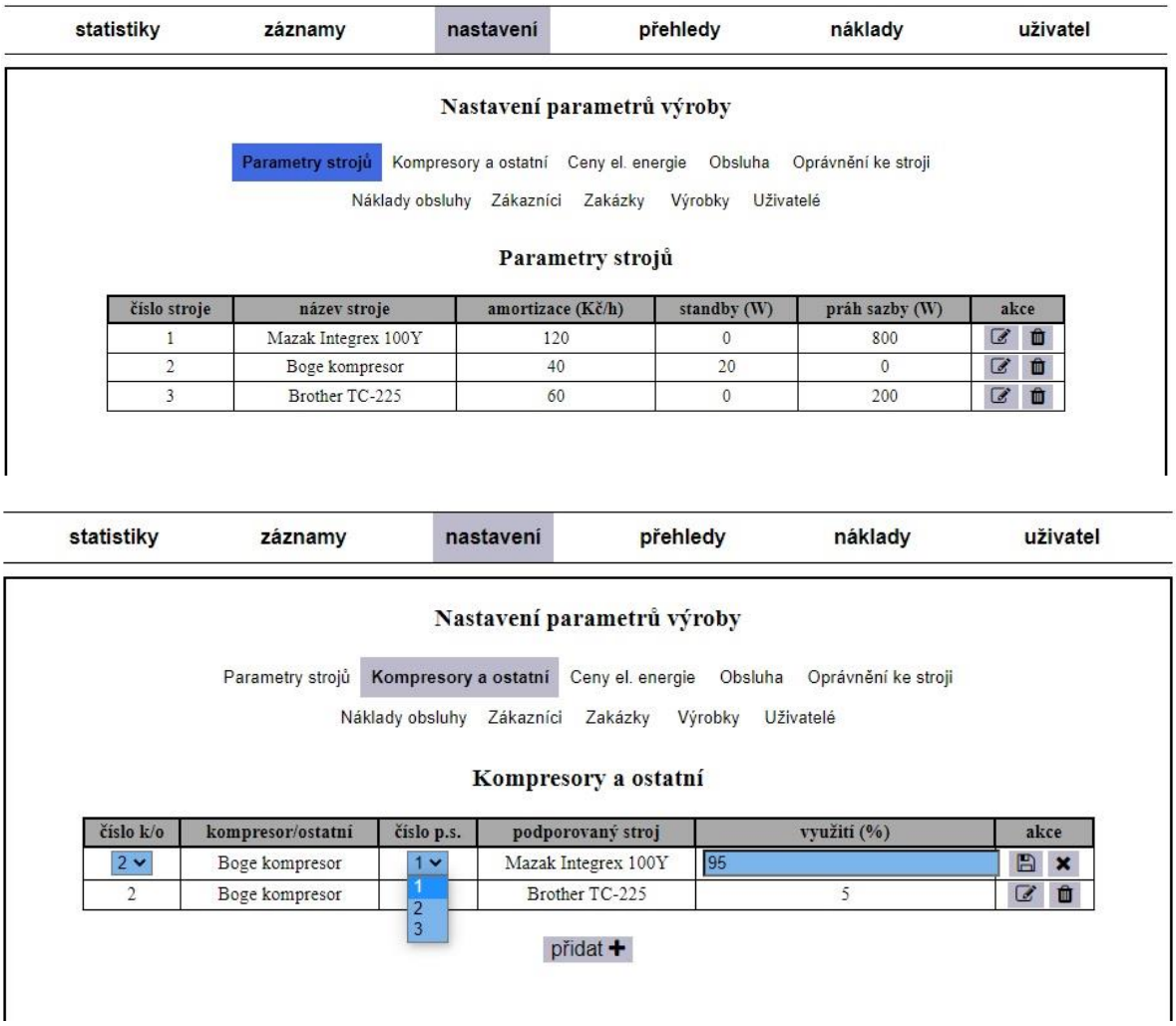


Obrázek 38: *Snímek záložky elektronického deníku*

Zdroj: vlastní zpracování

8.10 Nastavení parametrů výroby

Na následujících ukázkách jsou zobrazeny dvě různé sekce nastavení parametrů výroby. Výčet dalších sekcí nastavení je patrný z uvedeného snímku obrazovky. Všechny tabulky nastavení mají funkční tlačítka pro úpravu, nebo smazání. Tlačítko editace je při stisku změněno na tlačítko zavření úprav. Pro minimalizaci rizika zadání neplatných hodnot je při úpravách, nebo přidání, nabízen pouze výčet platných hodnot (viz 2. sekce nastavení na následujícím obrázku). Parametry strojů nemají tlačítko přidat, protože jejich počet a číslo jsou vázány na tabulku naměřených dat v databázi MySQL.



Obrázek 39: *Záložka nastavení parametrů výroby*

Zdroj: vlastní zpracování

8.11 Přehledy o výrobě

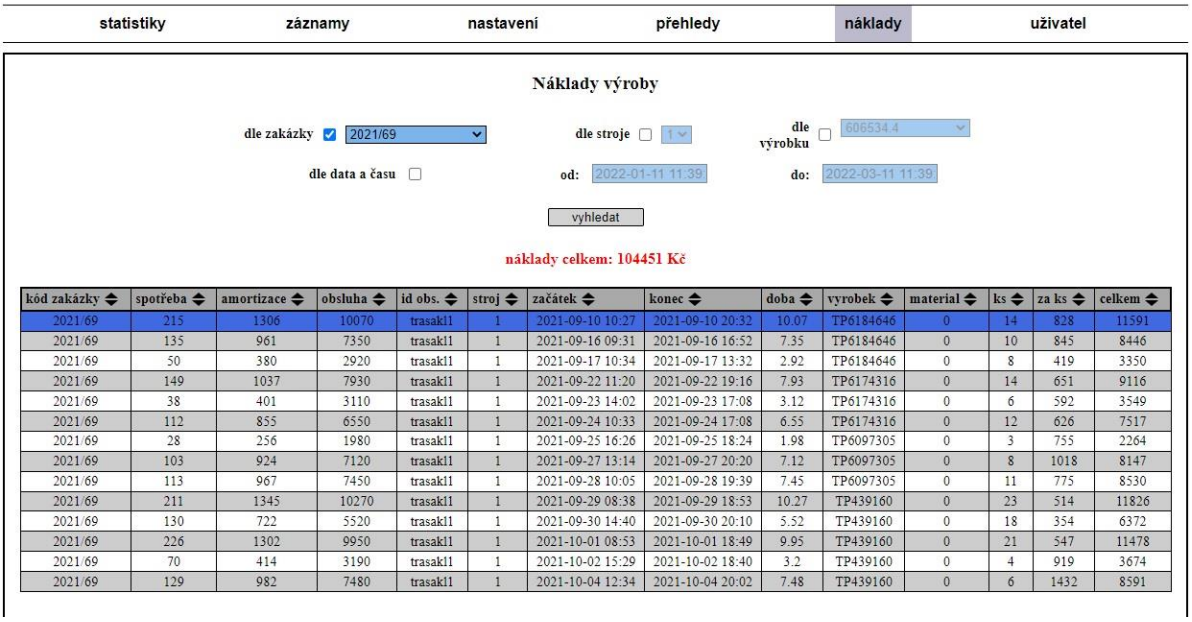
Poskytují podrobný přehled o spotřebě a vytížení strojů ve výrobě. Z následujících koláčových grafů je patrné, že stroj č. 1 spotřebuje daleko více energie proti stroji č.3 při stejné době používání. Podle pravého koláčového grafu vychází největší využití na stroj č. 1, v tomto případě se jedná o Mazak Integrex 100 Y. Samozřejmě je třeba vzít v úvahu další výrobní faktory, jako je např. kapacita stroje a jeho další možnosti.

Obrázek 40: *Celkové statistiky výroby*

Zdroj: vlastní zpracování

8.12 Náklady výroby

Zobrazuje tabulku s řádky představující jednotlivé záznamy výroby s detailním rozpisem nákladů. Nad tabulkou je uvedena celková částka nákladů (červeně zvýrazněná) v závislosti na nastaveném filtru. Náklady lze filtrovat podle čísla zakázky, čísla stroje, čísla výrobku, časového období, nebo kombinací uvedených filtrů. Tabulka umožňuje řazení podle sloupce v rostoucím, nebo klesajícím pořadí. Z prvního řádku tabulky lze např. vyčíst, že výrobek „TP6184646“ byl vyroben v počtu 14 kusů, dne 10.9.2021, s náklady 828 Kč/ks, výroba trvala cca 10 hodin, energie činila 215 Kč a amortizace činila 1306 Kč. Uvedená tabulka zobrazuje výrobní záznamy s náklady na zakázku č. 2021/69.



kód zakázky	spotřeba	amortizace	obsluha	id obs.	stroj	začátek	konec	doba	vyrobek	material	ks	za ks	celkem
2021/69	215	1306	10070	trasakl1	1	2021-09-10 10:27	2021-09-10 20:32	10.07	TP6184646	0	14	828	11591
2021/69	135	961	7350	trasakl1	1	2021-09-16 09:31	2021-09-16 16:52	7.35	TP6184646	0	10	845	8446
2021/69	50	380	2920	trasakl1	1	2021-09-17 10:34	2021-09-17 13:32	2.92	TP6184646	0	8	419	3350
2021/69	149	1037	7930	trasakl1	1	2021-09-22 11:20	2021-09-22 19:16	7.93	TP6174316	0	14	651	9116
2021/69	38	401	3110	trasakl1	1	2021-09-23 14:02	2021-09-23 17:08	3.12	TP6174316	0	6	592	3549
2021/69	112	855	6550	trasakl1	1	2021-09-24 10:33	2021-09-24 17:08	6.55	TP6174316	0	12	626	7517
2021/69	28	256	1980	trasakl1	1	2021-09-25 16:26	2021-09-25 18:24	1.98	TP6097305	0	3	755	2264
2021/69	103	924	7120	trasakl1	1	2021-09-27 13:14	2021-09-27 20:20	7.12	TP6097305	0	8	1018	8147
2021/69	113	967	7450	trasakl1	1	2021-09-28 10:05	2021-09-28 19:39	7.45	TP6097305	0	11	775	8530
2021/69	211	1345	10270	trasakl1	1	2021-09-29 08:38	2021-09-29 18:53	10.27	TP439160	0	23	514	11826
2021/69	130	722	5520	trasakl1	1	2021-09-30 14:40	2021-09-30 20:10	5.52	TP439160	0	18	354	6372
2021/69	226	1302	9950	trasakl1	1	2021-10-01 08:53	2021-10-01 18:49	9.95	TP439160	0	21	547	11478
2021/69	70	414	3190	trasakl1	1	2021-10-02 15:29	2021-10-02 18:40	3.2	TP439160	0	4	919	3674
2021/69	129	982	7480	trasakl1	1	2021-10-04 12:34	2021-10-04 20:02	7.48	TP439160	0	6	1432	8591

Obrázek 41: *Záložka s přehledem nákladů*

Zdroj: vlastní zpracování

9 Závěr

Přínosem diplomové práce je praktický návrh a vytvoření zařízení pro měření spotřeby a doby chodu ve strojích průmyslové výroby s využitím platformy Arduino. Zařízení umožňuje nepřímé měření spotřeby, proudu a činného výkonu v obvodu zátěže elektrické třífázové sítě. Naměřené veličiny jsou bezdrátově odesílány do databáze, odkud jsou předávány webové aplikaci, která umožňuje jejich zobrazování formou grafů, tabulek a textových údajů. Aplikace je dostupná na stránkách https://arduinopj01.cz/kovovyroba.

V teoretické části jsou vysvětleny principy fungování mikrokontrolerů a periférií, které byly použity při stavbě zařízení měřiče. Dále jsou podrobně popsány standardy komunikačních rozhraní, které umožňují výměnu dat mezi mikrokontrolerem a přídavnými moduly zařízení. Přídavné moduly jsou pomyslně rozebrány na jednotlivé částí, které jsou podrobně charakterizovány. Technologie hardwaru jsou pro názornost doplněny obrázky a schématy. V kapitole Software jsou uvedeny základní charakteristiky programovacích jazyků použitých v rámci třívrstvé architektury včetně jazyku Wiring platformy Arduino. Uvedená kapitola je doplněna o ukázky zdrojových kódů včetně jejich popisu.

Praktická část představuje tvorbu projektu zařízení, využívajícího platformy Arduino, kterou lze specifikovat do obecných bodů postupu. Jsou vybrány komponenty a další prvky, které svými vlastnostmi odpovídají daným požadavkům a je provedeno jejich ekonomické vyhodnocení formou tabulky s podrobným rozpisem materiálu. Na základě sestaveného hardwaru je navržen a implementován zdrojový kód. Následně je proveden návrh a založení databáze. Pomocí softwarových prostředků uvedených v teoretické části je vytvořena webová aplikace umožňující vizualizaci a správu dat. Kromě samotné vizualizace jsou navíc přidány funkce pro celkové přehledy spotřeby, doby chodu a pro výpočet nákladů odvozených kombinací uživatelských dat elektronického deníku a naměřených dat konkrétního stroje.

Měřič byl uveden do provozu v létě 2021 a webová aplikace spuštěna v listopadu 2021. Zdrojové kódy webové aplikace obsahují cca 7000 řádků vlastního kódu. Požadované funkce v navrženém systému byly odzkoušeny půlročním testováním a spolehlivým provozem tří zařízení ve výrobní dílně. Výsledky získané prostřednictvím uvedeného zařízení mohou sloužit jako podkladová data pro diagnostiku samotného stroje, nebo při rozhodování budoucího rozšiřování, či specializace výroby.

10 Reference

[1] Z. Obermaier, „PCtuning - Úvod do měření počítačových zdrojů,“ 28 7 2011. [Online]. Available: https://pctuning.cz/article/uvod-do-mereni-pocitacovych-zdroju-teorie-a-funkce?chapter=2. [Přístup získán 22 12 2021].

[2] z německého originálu časopisu de, 8/2008, upravil Ing. Josef Košťál, redakce Elektro, „Elektro - Kompenzace elektrického jalového výkonu,“ 3 2009. [Online]. Available: http://www.odbornecasopisy.cz/elektro/casopis/tema/kompenzace-elektrickeho-jaloveho-vykonu--11073. [Přístup získán 23 12 2021].

[3] Z. V. &. T. H. Kitchen, Průvodce světem Arduina, Bučovice: Nakladatelství Martin Stříž, 2017, p. 12.

[4] E. Systems, „the INTERNET of THIHGS with ESP32,“ [Online]. Available: http://esp32.net/. [Přístup získán 10 11 2021].

[5] 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 2021].

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

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

[8] 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 2021].

[9] 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 2021].

[10] S. Larrivee, „Solid State Drive Primer # 2 - SLC, MLC and TLC NAND Flash,“ Cactus Technologies, 2 3 2015. [Online]. Available: https://www.cactus-tech.com/resources/blog/details/solid-state-drive-primer-2-slc-mlc-and-tlc-nand-flash/. [Přístup získán 18 12 2021].

[11] Digital Discount, „EVERYTHING YOU NEED TO KNOW ABOUT SLC, MLC, & TLC NAND FLASH,“ 27 7 2015. [Online]. Available:

https://www.mydigitaldiscount.com/everything-you-need-to-know-about-slc-mlc-and-tlc-nand-flash.html. [Přístup získán 18 12 2021].

[12] lui_gough, „Experiment: microSD Card Power Consumption & SPI Performance,“ TECHZONE, 27 02 2021. [Online]. Available: https://goughlui.com/2021/02/27/experiment-microsd-card-power-consumption-spi-performance/. [Přístup získán 20 11 2021].

[13] 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].

[14] L. M. Engineers, „Interface DS3231 Precision RTC Module with Arduino,“ 2021. [Online]. Available: https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/. [Přístup získán 25 11 2021].

[15] 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 2021].

[16] 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].

[17] Internation IOR Rectifier, „IRLR7843PbF,“ 04 2008. [Online]. Available: https://www.infineon.com/dgdl/Infineon-IRLR7843-DataSheet-v01_01-EN.pdf?fileId=5546d462533600a40153566de53526d8. [Přístup získán 26 11 2021].

[18] ProtoSupplies, „LR7843 MOSFET Control Module,“ 2021. [Online]. Available: https://protosupplies.com/product/lr7843-mosfet-control-module/. [Přístup získán 27 11 2021].

[19] NanJing Top Power ASIC Corp, „TP4056,“ [Online]. Available: http://www.tp4056.com/d/tp4056.pdf. [Přístup získán 27 12 2021].

[20] V. Technologies, „V98XX Datasheet,“ 2016.

[21] InovatorsGuru, „PZEM-004T V3,“ 28 06 2019. [Online]. Available: https://innovatorsguru.com/pzem-004t-v3/. [Přístup získán 2 12 2021].

[22] Electronics Tutorials, „The Current Transformer,“ [Online]. Available: https://www.electronics-tutorials.ws/transformer/current-transformer.html. [Přístup získán 1 12 2021].

[23] K. Ježková, „SEZNAMTE SE S ARDUINO VÝVOJOVÝM PROSTŘEDÍM,“ 14 08 2014. [Online]. Available: https://arduino.cz/seznamte-s-arduino-vyvojovym-prostredim/. [Přístup získán 27 11 2021].

[24] L. Moore, „MySQL,“ 7 2018. [Online]. Available: https://searchoracle.techtarget.com/definition/MySQL. [Přístup získán 28 12 2021].

[25] EmmanuelleRieuf, „History of MySQL,“ 16 12 2016. [Online]. Available: https://www.datasciencecentral.com/history-of-mysql/. [Přístup získán 28 12 2021].

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

[27] J. Čížek, „Mnozí mu předpovídali stagnaci a smrt. Jazyk PHP ale žije a nově už ve verzi 8.0,“ 26 11 2020. [Online]. Available: https://www.zive.cz/clanky/mnozi-mu-predpovidali-stagnaci-a-smrt-jazyk-php-ale-zije-a-nove-uz-ve-verzi-80/sc-3-a-207201/default.aspx. [Přístup získán 28 12 2021].

[28] C. Brotherton, „The Most Popular PHP Frameworks to Use in 2022,“ 6 07 2021. [Online]. Available: https://kinsta.com/blog/php-frameworks/. [Přístup získán 29 12 2021].

[29] R. Kejduš, „Programovací jazyk PHP slaví 25 let. Používá jej 80 % webových stránek i Facebook,“ 10 6 2020. [Online]. Available: https://www.cnews.cz/php-programovaci-jazyk-25-let-80-procent-webu/. [Přístup získán 29 12 2021].

[30] 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].

[31] 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].

[32] B. Koďousková, „HTML PRO ZAČÁTEČNÍKY ANEB JAK ZAČÍT PSÁT WEB,“ 15 07 2021. [Online]. Available: https://www.rascasone.com/cs/blog/html-pro-zacatecniky-jak-psat-web. [Přístup získán 30 1 2022].

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

[34] Z3Controls, „CT Calculation Helper,“ [Online]. Available: https://z3controls.com/ct-helper.php. [Přístup získán 29 11 2021].

[35] TheHWcave, „Peacefair-PZEM-004T,“ 17 04 2021. [Online]. Available: https://github.com/TheHWcave/Peacefair-PZEM-004T-/blob/main/PZEM004T-orig.pdf. [Přístup získán 1 12 2021].

11 Přílohy

A SCHÉMA ZAPOJENÍ

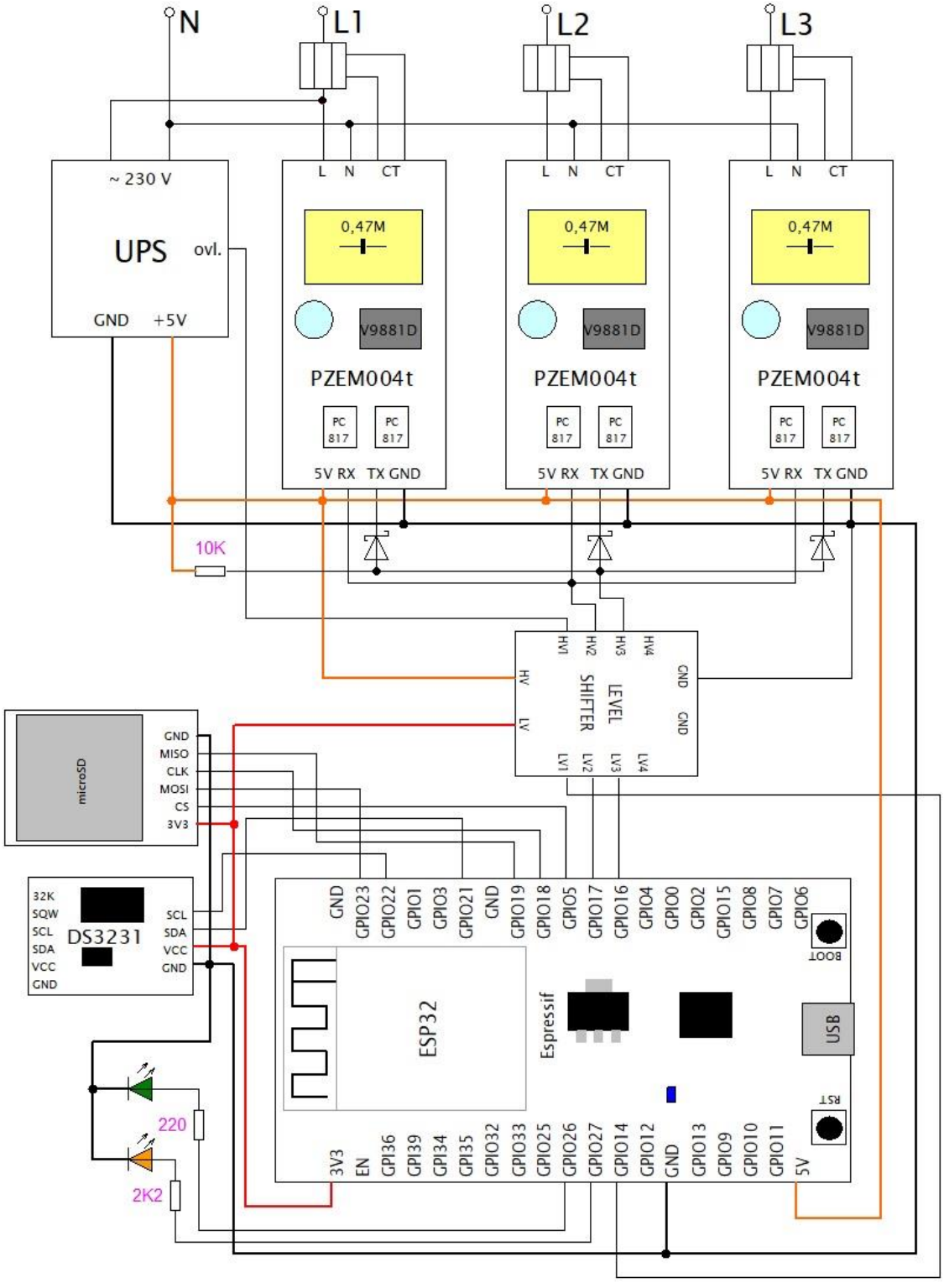
B UKÁZKY ZDROJOVÝCH KÓDŮ

C OBRÁZEK TABULKY NAMĚŘENÝCH DAT

D FOTODOKUMENTACE

E MODEL KRABIČKY ZAŘÍZENÍ

A SCHÉMA ZAPOJENÍ



Obrázek A.1: *Celkové schéma zapojení zařízení měřiče*

B UKÁZKY ZDROJOVÝCH KÓDŮ

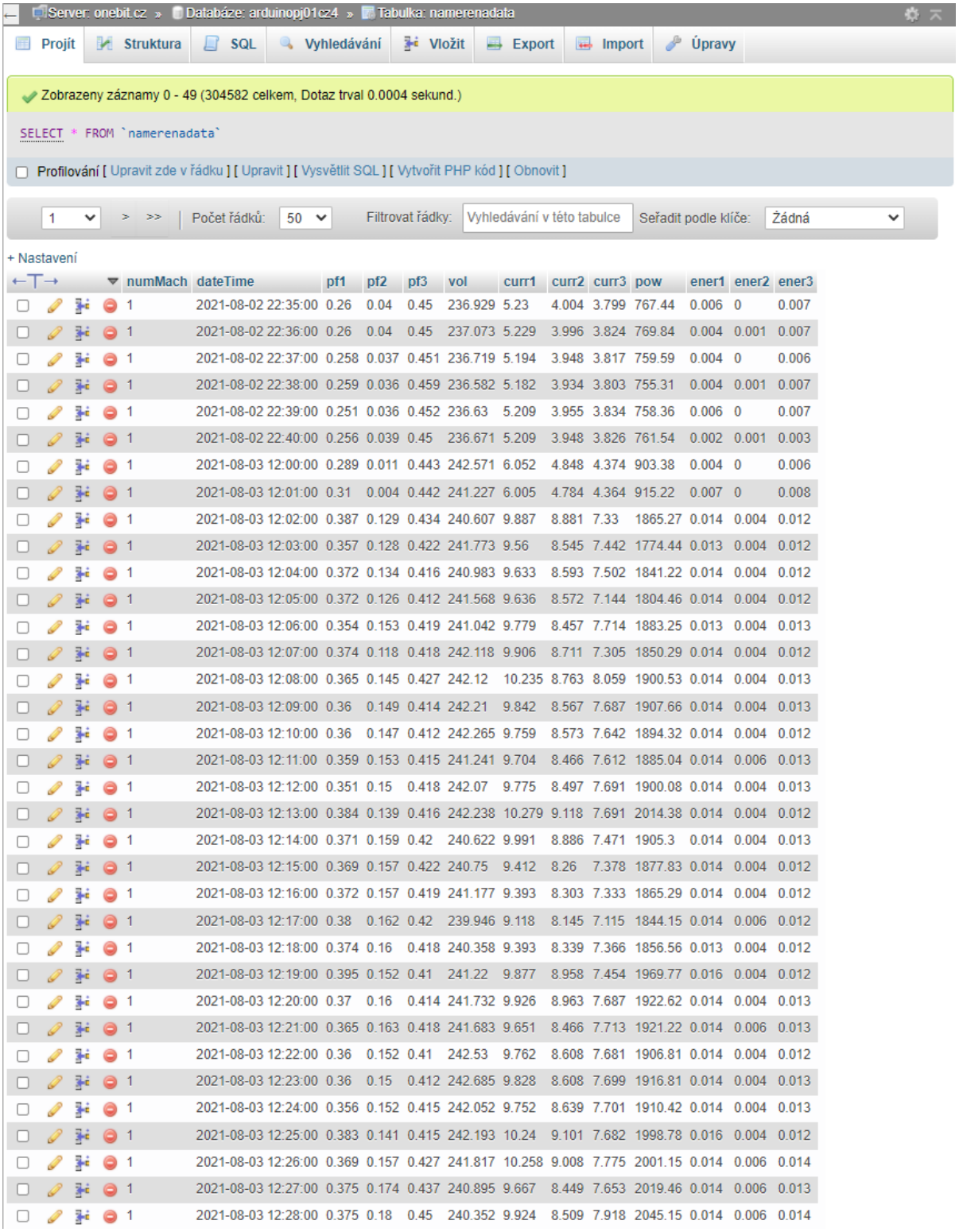
```
void loop() {
  //čekání na další vteřinu
  while (previousSec == timeRTC.second()) {
    delay(10);
    timeRTC = DS3231.now();
  }

  //výpočet v celé minutě
  if (previousMin != timeRTC.minute() && firstMsmrt) {
    previousMin = timeRTC.minute();
    save_time_values();
    Serial.println("celá minuta, provede se výpočet");
    //vypočítá minutové údaje
    for (int i = 0; i < NUM_PZEMS; i++) calc_measured_data(i, dpZem[i], pzems[i]);

    //kontrolní výpis do konzole
    for (int i = 1; i <= 3; i++) {
      printData(i);
    }
    delay(100);
    Serial.println(String() + "Čas: " + realDateTime + "\n*********************************************************************************************

    //POSTUPNÉ NASTAVENÍ STAVOVÝCH PROMĚNNÝCH (FLAGS)
    conectWiFi = false;
    dbsConected = false;

    //nejprve otestuje připojení k wifi, pokud není, zkusí se připojit
    if (WiFi.status() == WL_CONNECTED)conectWiFi = true;
    else if (connect_to_Wifi())conectWiFi = true;

    //pokud je připojení k WiFi a dostupná databáze, je nastavena stavová proměnná "dbsConnected"
    if (conectWiFi) {
      if (test_connect_dbs())dbsConected = true;
    }
    //pokud byl stroj v poslední minutě pod napětím, nastaví stavovou proměnou "activeMachine"
    if (checkPzem() == NUM_PZEMS)activeMachine = true;
    else activeMachine = false;

    //VYHODNOCENÍ STAVOVÝCH PROMĚNNÝCH
    if (conectWiFi && dbsConected)digitalWrite(ledCn, HIGH);
    else digitalWrite(ledCn, LOW);
    //pokud byl stroj v předcházející minutě pod napětím, zapíše data a vyčistí dočasné datové pole
    unsigned long startTm = millis();
    if (activeMachine)send_to_dbs(&calcData[0], dbsConected);
    memset(calcData, 0, sizeof(calcData));
    unsigned int uploadTime = millis() - startTm;
    //pro diagnostiku
    Serial.println(String() + "Doba uložení dat: " + uploadTime);
    bool synchro = false;

    //pokud existuje záloha (a je připojení, nebo je záloha nedočtená do konce)
    if (exists_file(backupFile)) {
      if (dbsConected || bkpCompl == 1) {
        synchro = true;
        bkpCompl = read_file(backupFile, 20000, 20);
      }
    }

    //------------------------------------ diagnostika - ukládá na kartu SD ---------------------------//
    int freeHeap = ESP.getFreeHeap();
    long rssi;
    if (conectWiFi) rssi = WiFi.RSSI();
    else rssi = 0;
    //připojení k wifi, síla signálu, připojení k dbs, volná paměť, čas uložení do dbs/sd karty, synchronizace
    save_time_values();
    if (exists_file("/DIAG")) save_log(realDateTime, conectWiFi, rssi, dbsConected, freeHeap, uploadTime, synchro);
    //-----------------------------------------------------------------------------------------------//

    //pokud je stroj neaktivní a není rozečtený záložní soubor, vypne celou jednotku
    if (!activeMachine && bkpCompl != 1) {
      Serial.println("VYPÍNÁM...");
      delay(1000);
      digitalWrite(14, HIGH);
    }
    Serial.println("-------------------------------------------------");
    Serial.println(String() + "Volná paměť po celé smyčce: " + freeHeap);
    Serial.println("-------------------------------------------------");
    //ochrana proti přetečení haldy
    if (bkpCompl != 1 && freeHeap < 140000)ESP.restart();
  }
  //načte data z jednotek PZEM (parametry - reference data z jednotky pzem a objekt PZEM)
  if (previousSec != timeRTC.second()) {
    previousSec = timeRTC.second();
    for (int i = 0; i < NUM_PZEMS; i++) {
      save_from_pzem(dpZem[i], pzems[i]);
      delay(100);
    }
  }
  if (!firstMsmrt)firstMsmrt = true;
}
```

Obrázek B.1: *Zdrojový kód hlavního souboru měřiče*

```php
//vytvoří instanci dle následující třídy
$records = new Records(5);

//$records->getDataForGantt(1,'2022-01-07 00:00', '2022-03-07 23:59');

//třída, určená pro složení záznamu do Ganttova grafu
class Records {
    private $gap;

    //konstruktor nastaví velikost přijatelné mezery mezi minutovými záznamy
    public function __construct($gap){
        if(!is_numeric($gap)) $gap = 1;
        else if($gap < 0 || $gap > 30) $gap = 1;
        $timeFormat = sprintf('00:%02d:00', $gap);
        $this->gap = $timeFormat;
    }

    //hlavní funkce nastavení dat pro Ganttův diagram
    public function getDataForGantt($numMach, $from, $to){
        //do proměnných načte pole pole naměřených hodnot a změřených hodnot
        $measured = $this->getMeasuredData($numMach,$from,$to);
        $recorded = $this->getProducedData($numMach,$from,$to);
        if(count($measured) > 0) $this->checkRecordsInMeasured($measured, $recorded);
    }

    //kontroluje záznamy výroby proti naměřeným záznamům a předává parametry funkcím pro sloučení, nebo rozdělení
    private function checkRecordsInMeasured(&$measured, $recorded){
        //vezme existující záznam výroby a porovná ho s celou řadou naměřených
        foreach($recorded as $recordedItem){
            $this->joinRecordedItem($recordedItem['fromDt'], $recordedItem['toDt'], $measured);
            $this->searchInsideMeasuredInterval($recordedItem['fromDt'], $recordedItem['toDt'], $measured);
        }
        //nastaví vlastnosti jako kód zakázky, id operátora, atd, samozřejmě za předpokladu, že existují, jinak jsou klíče pole nastaveny na prázdné
        $this->setPropData($measured, $recorded);
        print json_encode($measured);
    }

    //třídí multidimenzionální pole
    private function sortByStartDate(&$measured){...
    }

    //spojení - v rámci předaného bodu (záznam výroby) odstraní nadbytečné body záznamů stroje - parametr je předáván odkazem, nikoliv hodnotou
    private function joinRecordedItem($fromRecordedItem, $toRecordedItem, &$measured){...
    }

    //hledá začátky a konce datových záznamů výroby v intervalech záznamů strojů
    private function searchInsideMeasuredInterval($fromRecordedItem, $toRecordedItem, &$measured){
        $counter = 0;
        //foreach($measured as $itemMeasured){
        while($counter < count($measured)){
            //kdykoliv je koncový, či počáteční bod uvnitř intervalu, roztrhne ho a vytvoří dva nové, přičemž rozšíří stávající pole
            if($measured[$counter]['fromDt'] < $fromRecordedItem && $measured[$counter]['toDt'] > $fromRecordedItem){
                $this->SplitMeasuredItem($fromRecordedItem, $counter, $measured);
                continue;
            }
            if($measured[$counter]['fromDt'] < $toRecordedItem && $measured[$counter]['toDt'] > $toRecordedItem){
                $this->SplitMeasuredItem($toRecordedItem, $counter, $measured);
                continue;
            }
            $counter++;
        }
    }

    //rozděli naměřený datový záznam podle výrobního záznamu (parametry: bod rozdělení, pozice v poli, pole)
    private function SplitMeasuredItem($point, $index, &$measured){...
    }

    //hromadně nastavuje vlastnosti datovým záznamům
    private function setPropData(&$measured, $recorded){
        $index = 0;
        foreach($measured as $measuredItem){
            $this->setPropDataItem($measured, $recorded, $index);
            $index++;
        }
    }

    //nastaví vlastnosti datové položky, pokud nude shoda s záznamem výroby, budou doplněny attributy výroby, jinak bude nastaveno na prázdné
    private function setPropDataItem(&$measuredData, $recordedData, $index){...
    }

    //získá naměřená data, která jsou složena z datových záznamů
    private function getMeasuredData($numMach,$fromDt,$toDt){...
    }

    private function getProducedData($numMach,$fromDt,$toDt){
    $mysqli = mysqli_connect(SERVER, USER, PASSWORD, DB);...
    }


//konec deklarace třídy
}
```

Obrázek B.2: *Zdrojový kód PHP pro získání podkladových dat Ganttova diagramu*

C OBRÁZEK TABULKY NAMĚŘENÝCH DAT

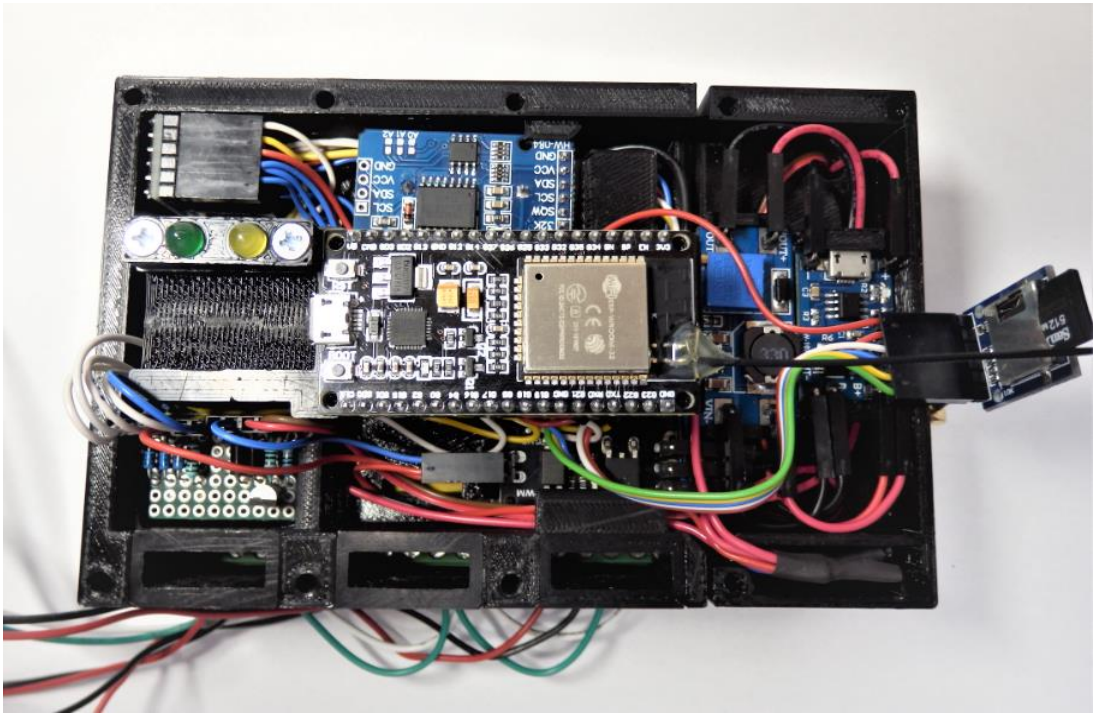


numMach	dateTime	pf1	pf2	pf3	vol	curr1	curr2	curr3	pow	ener1	ener2	ener3
1	2021-08-02 22:35:00	0.26	0.04	0.45	236.929	5.23	4.004	3.799	767.44	0.006	0	0.007
1	2021-08-02 22:36:00	0.26	0.04	0.45	237.073	5.229	3.996	3.824	769.84	0.004	0.001	0.007
1	2021-08-02 22:37:00	0.258	0.037	0.451	236.719	5.194	3.948	3.817	759.59	0.004	0	0.006
1	2021-08-02 22:38:00	0.259	0.036	0.459	236.582	5.182	3.934	3.803	755.31	0.004	0.001	0.007
1	2021-08-02 22:39:00	0.251	0.036	0.452	236.63	5.209	3.955	3.834	758.36	0.006	0	0.007
1	2021-08-02 22:40:00	0.256	0.039	0.45	236.671	5.209	3.948	3.826	761.54	0.002	0.001	0.003
1	2021-08-03 12:00:00	0.289	0.011	0.443	242.571	6.052	4.848	4.374	903.38	0.004	0	0.006
1	2021-08-03 12:01:00	0.31	0.004	0.442	241.227	6.005	4.784	4.364	915.22	0.007	0	0.008
1	2021-08-03 12:02:00	0.387	0.129	0.434	240.607	9.887	8.881	7.33	1865.27	0.014	0.004	0.012
1	2021-08-03 12:03:00	0.357	0.128	0.422	241.773	9.56	8.545	7.442	1774.44	0.013	0.004	0.012
1	2021-08-03 12:04:00	0.372	0.134	0.416	240.983	9.633	8.593	7.502	1841.22	0.014	0.004	0.012
1	2021-08-03 12:05:00	0.372	0.126	0.412	241.568	9.636	8.572	7.144	1804.46	0.014	0.004	0.012
1	2021-08-03 12:06:00	0.354	0.153	0.419	241.042	9.779	8.457	7.714	1883.25	0.013	0.004	0.013
1	2021-08-03 12:07:00	0.374	0.118	0.418	242.118	9.906	8.711	7.305	1850.29	0.014	0.004	0.012
1	2021-08-03 12:08:00	0.365	0.145	0.427	242.12	10.235	8.763	8.059	1900.53	0.014	0.004	0.013
1	2021-08-03 12:09:00	0.36	0.149	0.414	242.21	9.842	8.567	7.687	1907.66	0.014	0.004	0.013
1	2021-08-03 12:10:00	0.36	0.147	0.412	242.265	9.759	8.573	7.642	1894.32	0.014	0.004	0.012
1	2021-08-03 12:11:00	0.359	0.153	0.415	241.241	9.704	8.466	7.612	1885.04	0.014	0.006	0.013
1	2021-08-03 12:12:00	0.351	0.15	0.418	242.07	9.775	8.497	7.691	1900.08	0.014	0.004	0.013
1	2021-08-03 12:13:00	0.384	0.139	0.416	242.238	10.279	9.118	7.691	2014.38	0.014	0.004	0.012
1	2021-08-03 12:14:00	0.371	0.159	0.42	240.622	9.991	8.886	7.471	1905.3	0.014	0.004	0.013
1	2021-08-03 12:15:00	0.369	0.157	0.422	240.75	9.412	8.26	7.378	1877.83	0.014	0.004	0.012
1	2021-08-03 12:16:00	0.372	0.157	0.419	241.177	9.393	8.303	7.333	1865.29	0.014	0.004	0.012
1	2021-08-03 12:17:00	0.38	0.162	0.42	239.946	9.118	8.145	7.115	1844.15	0.014	0.006	0.012
1	2021-08-03 12:18:00	0.374	0.16	0.418	240.358	9.393	8.339	7.366	1856.56	0.013	0.004	0.012
1	2021-08-03 12:19:00	0.395	0.152	0.41	241.22	9.877	8.958	7.454	1969.77	0.016	0.004	0.012
1	2021-08-03 12:20:00	0.37	0.16	0.414	241.732	9.926	8.963	7.687	1922.62	0.014	0.004	0.013
1	2021-08-03 12:21:00	0.365	0.163	0.418	241.683	9.651	8.466	7.713	1921.22	0.014	0.006	0.013
1	2021-08-03 12:22:00	0.36	0.152	0.41	242.53	9.762	8.608	7.681	1906.81	0.014	0.004	0.012
1	2021-08-03 12:23:00	0.36	0.15	0.412	242.685	9.828	8.608	7.699	1916.81	0.014	0.004	0.013
1	2021-08-03 12:24:00	0.356	0.152	0.415	242.052	9.752	8.639	7.701	1910.42	0.014	0.004	0.013
1	2021-08-03 12:25:00	0.383	0.141	0.415	242.193	10.24	9.101	7.682	1998.78	0.016	0.004	0.012
1	2021-08-03 12:26:00	0.369	0.157	0.427	241.817	10.258	9.008	7.775	2001.15	0.014	0.006	0.014
1	2021-08-03 12:27:00	0.375	0.174	0.437	240.895	9.667	8.449	7.653	2019.46	0.014	0.006	0.013
1	2021-08-03 12:28:00	0.375	0.18	0.45	240.352	9.924	8.509	7.918	2045.15	0.014	0.006	0.014

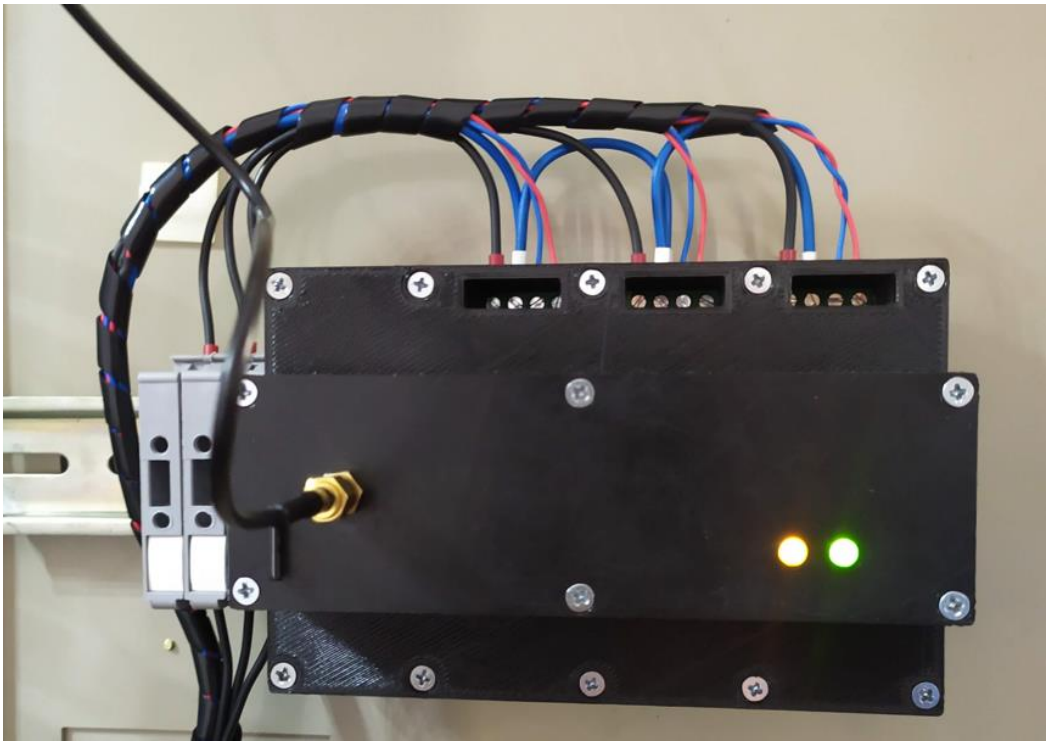
Obrázek C.1: *Tabulka naměřených dat databáze MySQL*

D FOTODOKUMENTACE

Obrázek D.1: *Uspořádání silové části měřiče*

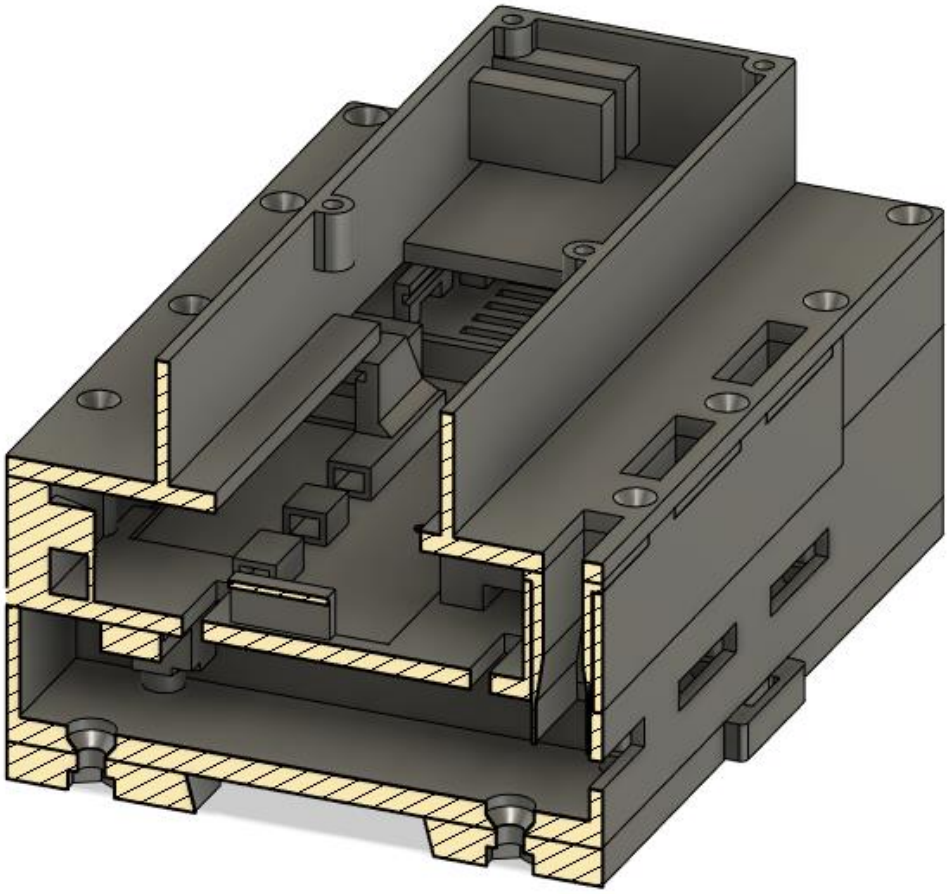


Obrázek D.2: *Uspořádání slaboproudé části měřiče*



Obrázek D.3: *Nainstalované zařízení měřiče*

E MODEL KRABIČKY ZAŘÍZENÍ



Obrázek E.1: *Řez modelem krabičky zařízení (SW Fusion 360)*