VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

PDU S FUNKCÍ MĚŘENÍ SPOTŘEBY PRO JEDNODESKOVÉ POČÍTAČE

PDU WITH CONSUMPTION METERING FUNCTION FOR SINGLE-BOARD COMPUTERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ROMAN ONDRÁČEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VÁCLAV ŠIMEK**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

148220

Ústav: Ústav počítačových systémů (UPSY)
Student: **Ondráček Roman**
Program: Informační technologie
Specializace: Informační technologie

Název: **PDU s funkcí měření spotřeby pro jednodeskové počítače**
Kategorie: Vestavěné systémy
Akademický rok: 2022/23

Zadání:

1. Seznamte se s problematikou napájecích jednotek (PDU) s podporou měření spotřeby. Stručně shrňte klíčové vlastnosti existujících řešení.
2. Prostudujte aktuálně používané IoT protokoly umožňující vzdálený přístup a monitorování činnosti PDU jednotek. Zaměřte se na měření a spínání napájení jednodeskových počítačů, které vyžadují napájení 5 V DC, včetně ošetření mezních stavů (např. překročení mezního proudového odběru).
3. Na základě nastudovaných poznatků navrhněte koncepci vestavěného systému pro monitorovaní a spínání napájení jednodeskových počítačů.
4. Zvolte vhodné komponenty odpovídající vytvořenému návrhu. Proveďte technickou realizaci na obvodové úrovni a základní oživení výsledného řešení.
5. Implementujte obslužný firmware zajišťující požadovanou funkcionalitu systému. Současně implementujte webové uživatelské rozhraní umožňující zobrazování informací o stavu měřených zařízení a konfiguraci napájecích výstupů.
6. Ověřte funkčnost realizovaného řešení. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:
Bez požadavků.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Šimek Václav, Ing.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 31.10.2022

Abstrakt

Cílem této práce je navrhnutí napájecí jednotky pro jednodeskové počítače, která dokáže spínat jednotlivé své výstupy a také na nich měřit elektrické veličiny (napětí, proud a výkon), založené na platformě Espressif ESP32 a navrhnutí webového rozhraní pro správu těchto napájecích jednotek. Toto řešení bude zaměřeno zejména pro využití napájecích jednotek v procesu vývoje vestavěných zařízeních postavených nad jednodeskovými počítači. Navržené řešení se skládá z napájecí jednotky a informačního systému pro správu, komunikace mezi jednotlivými částmi probíhá pomocí protokolu MQTT. Pomocí informačního systému lze ovládat jednotlivé výstupy napájecí jednotky, zobrazit si grafy měřených veličin.

Abstract

The aim of this thesis is to design a power distribution unit for single-board computers that can switch its individual outputs and also measure electrical quantities (voltage, current and power) based on the Espressif ESP32 platform and to design a web interface for the management of these power distribution units. This solution will be mainly aimed for the use of power distribution units in the development process of embedded devices built on top of single board computers. The proposed solution consists of a power distribution unit and a management information system, the communication between the different parts is done using the MQTT protocol. The information system can be used to control the individual outputs of the power distribution unit, to view graphs of the measured quantities.

Klíčová slova

napájecí jednotka, PDU, ESP32, MQTT, jednodeskový počítač, IoT, Wi-Fi

Keywords

power distribution unit, PDU, ESP32, MQTT, single-board computer, IoT, Wi-Fi

Citace

ONDRÁČEK, Roman. *PDU s funkcí měření spotřeby pro jednodeskové počítače*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Václav Šimek

PDU s funkcí měření spotřeby pro jednodeskové počítače

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana inženýra Václava Šimka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Roman Ondráček
10. května 2023

Poděkování

Děkuji mému vedoucímu panu inženýru Václavu Šimkovi, které mi poskytl technické rady a podporu při vypracování této bakalářské práce.

Obsah

1 Úvod **10**

2 Úvod do problematiky napájecích jednotek **11**
2.1 Úvod do napájecích jednotek . . . 11
2.2 Existující řešení . . . 12
2.2.1 NETIO PowerPDU 4C . . . 12
2.2.2 Ubiquiti UniFi Switch USW-Lite-8-PoE . . . 14
2.2.3 Nordic Semiconductor Power Profiler Kit II . . . 16
2.2.4 RuiDeng UM25C . . . 17
2.2.5 Porovnání existujících řešení . . . 19
2.3 Měření elektrických veličin . . . 20
2.3.1 Princip fungování analogově-digitálního převodníku . . . 20
2.3.2 Měření elektrického napětí . . . 22
2.3.3 Měření elektrického proudu . . . 22
2.3.4 Měření pomocí senzorů s komunikačním rozhraním . . . 22
2.4 Možnosti spínání napájení . . . 22

3 Úvod do komunikačních technologií pro napájecí jednotky **23**
3.1 Základní komunikační vzory . . . 23
3.1.1 Model Požadavek-Odpověď *(Request-Response)* . . . 23
3.1.2 Model Producent-Odběratel *(Publish-Subscribe)* . . . 23
3.1.3 Model *Push-Pull* . . . 24
3.2 Technologie pro drátovou komunikaci . . . 24
3.2.1 UART . . . 24
3.2.2 SPI . . . 25
3.2.3 I²C . . . 25
3.2.4 Ethernet . . . 26
3.3 Technologie pro bezdrátovou komunikaci . . . 27
3.3.1 LoRa . . . 27
3.3.2 IQRF . . . 27
3.3.3 WiFi . . . 29
3.4 Internetové komunikační protokoly . . . 30
3.4.1 SNMP . . . 30
3.4.2 MQTT . . . 30
3.4.3 HTTP . . . 32
3.4.4 CoAP . . . 34

4 Návrh architektury řešení **35**

4.1 Požadavky na napájecí jednotku . . . 36
4.2 Požadavky na komunikační protokol . . . 36
4.3 Požadavky na centrální správu napájecích jednotek . . . 36

5 Technická realizace hardwaru napájecí jednotky 38
5.1 Blokový diagram napájecí jednotky . . . 38
5.2 Bezdrátový modul Espressif ESP32-WROOM-32D . . . 39
5.3 Napájecí vstup . . . 40
5.4 Snižující DC-DC měnič Maxim MAX17634BATP+ . . . 40
5.5 USB-UART převodník Silicon Labs CP2102N . . . 40
5.6 Senzor Texas Instruments INA3221 . . . 41
5.7 Hodiny reálného času Microchip MCP7940N . . . 42
5.8 USB spínač Texas Instruments TPS2000C . . . 42
5.9 Napájecí výstupy . . . 43

6 Implementace obslužného firmwaru napájecí jednotky 44
6.1 ESP IoT Development Framework . . . 44
6.2 Operační systém reálného času FreeRTOS . . . 44
6.3 Funkcionalita obslužného firmwaru . . . 45
6.4 Konfigurace napájecí jednotky . . . 45
6.5 MQTT . . . 46
6.6 HTTP server . . . 47
6.7 Frontend . . . 47
6.7.1 Vue.js . . . 48

7 Implementace softwarového řešení pro centrální správu 50
7.1 Backend . . . 50
7.2 Frontend . . . 50
7.3 Odběratel a zpracovatel MQTT zpráv . . . 50
7.4 Práce s relační databází . . . 51
7.5 Práce s databází časových řad . . . 51
7.5.1 InfluxDB 2.x . . . 52
7.6 Správa uživatelů . . . 53
7.7 Autentizace uživatelů . . . 53
7.7.1 JSON Web Token . . . 53
7.7.2 Dvoufaktorová autentizace pomocí TOTP . . . 54
7.8 Správa napájecích jednotek . . . 54
7.9 HTTP REST API . . . 55

8 Testování 57
8.1 Základní pojmy v testování softwaru . . . 57
8.1.1 Specifikace . . . 57
8.1.2 Softwarová chyba . . . 57
8.1.3 Testování . . . 58
8.1.4 Mockování . . . 58
8.1.5 CI/CD . . . 58
8.2 Stupně testování . . . 58
8.2.1 Vodopádový model . . . 59
8.2.2 V-model . . . 59

8.2.3 Jednotkové testování . . . 59
8.2.4 Integrační testování . . . 59
8.2.5 Systémové testování . . . 60
8.2.6 Akceptační testování . . . 60
8.2.7 End-to-end testování . . . 60
8.3 Použité nástroje pro testování . . . 60
8.3.1 Nette Tester . . . 60
8.3.2 Postman . . . 60
8.3.3 GitLab CI . . . 60
8.4 Testování napájecí jednotky . . . 61
8.4.1 Oživení hardwaru napájecí jednotku . . . 61
8.4.2 Testování obslužného firmwaru . . . 62
8.4.3 Ověření přesnosti měření elektrických veličin . . . 62
8.5 Testování informačního systému pro centrální správu . . . 63
8.5.1 Testování backendu . . . 64
8.5.2 Testování frontendu . . . 64
8.6 Možnosti dalšího rozvoje projektu . . . 64

9 Závěr **65**

Literatura **66**

Přílohy **69**
Seznam příloh . . . 70

A Obsah přiloženého paměťového média **71**

B Schéma napájecí jednotky **72**

C Návrh plošného spoje **77**

D Soupiska součástek **79**
D.1 Cena výroby jedné napájecí jednotky . . . 81

E Fotografie napájecí jednotky **82**

Seznam obrázků

2.1 Fotografie PDU NETIO PowerPDU 4C převzatá ze stránek produktu[21] . 12
2.2 Webové rozhraní PDU NETIO PowerPDU 4C 13
2.3 Webové rozhraní cloudu pro napájecí jednotku NETIO PowerPDU 4C . . . 14
2.4 Fotografie přepínače Ubiquiti UniFi Switch USW-Lite-8-PoE převzatá ze stránek produktu[33] 14
2.5 Zobrazení odebraného výkonu z PoE portů v UniFi Network Application . . 15
2.6 Fotografie analyzátoru napájení Nordic Semiconductor Power Profiler Kit II, převzatá ze stránek produktu[22] 16
2.7 Snímek obrazovky z nRF Connect Power Profileru 17
2.8 Fotografie USB testeru RuiDeng UM25C 17
2.9 Snímek obrazovky z webové aplikace rd-usb 18
2.10 Schéma paralelního analogově-digitálního převodníku 21
2.11 Schéma analogově-digitálního převodníku s dvojitou integrací 21
2.12 Schéma napěťového děliče 22

3.1 Diagram komunikačního modelu Požadavek-Odpověď 23
3.2 Diagram komunikačního modelu Producent-Odběratel 24
3.3 Schéma zapojení zařízení komunikujících pomocí UART sběrnice bez hardwarového řízení toku dat 24
3.4 Schéma zapojení zařízení komunikujících pomocí SPI sběrnice 25
3.5 Schéma zapojení zařízení komunikujících pomocí I²C sběrnice 26
3.6 Architektura SNMP protokolu 30
3.7 Diagram komunikace při odeslání MQTT zprávy s QoS 0 31
3.8 Diagram komunikace při odeslání MQTT zprávy s QoS 1 32
3.9 Diagram komunikace při odeslání MQTT zprávy s QoS 2 32

4.1 Blokový diagram architektury navrženého systému 35
4.2 Diagram případu užití systému pro centrální správu napájecích jednotek . . 37

5.1 Blokový diagram napájecí jednotky 38
5.2 Blokové schéma bezdrátového modulu Espressif ESP32 převzaté z katalogového listu[4] 39
5.3 Část schématu s napájecím vstupem 40
5.4 Schéma logického obvodu pro automatické přepnutí do programovacího módu 40
5.5 Blokové schéma senzoru TI INA3221 převzaté z katalogového listu[31] . . . 41
5.6 Část schématu s obvodem hodin reálného času 42
5.7 Blokové schéma USB spínače TI TPS2000C, převzato z katalogového listu[32] 42

6.1 MVVM architektura frameworku Vue.js 48

6.2 Přehled napájecích výstupů v administraci napájecí jednotky 49

7.1 Diagram schématu MySQL databáze 51
7.2 Detail napájecí jednotky 55

8.1 Metody verifikace softwaru 57
8.2 Schéma vodopádového modelu 59
8.3 Schéma V-modelu 59
8.4 Nestabilní výstup DC-DC snižujícího měniče při proudovém odběru 1,7 A u 2. revize 61
8.5 Zvlnění na výstupu DC-DC snižujícího měniče při proudovém odběru 4,1 A 62
8.6 Graf procházejícího proudu napájecí jednotkou a analyzátorem napájení . . 63
8.7 Graf rozdílu naměřených hodnot protékajících proudů napájecí jednotkou a analyzátorem napájení 63

C.1 Vrchní vrstva desky plošných spojů 77
C.2 Spodní vrstva desky plošných spojů 78

E.1 Fotografie vrchní vrstvy desky plošných spojů napájecí jednotky 82
E.2 Fotografie spodní vrstvy desky plošných spojů napájecí jednotky 83

Seznam tabulek

2.1 Přehled standardů pro Power over Ethernet 12
2.2 Porovnání existujících řešení 19

3.1 UART rámec 24
3.2 Vodiče používané SPI sběrnicí 25
3.3 I2C datový rámec 26
3.4 I2C rámec 26
3.5 Ethernetový rámec 27
3.6 Schéma paketu požadavku IQRF DPA 28
3.7 Schéma paketu odpovědi IQRF DPA 28
3.8 Revize standardu IEEE 802.11 29
3.9 Popis jednotlivých částí URL 33

5.1 Pravdivostní tabulka obvodu pro automatické přepnutí do programovacího módu 40
5.2 Konfigurační registr senzoru TI INA3221 41
5.3 Registr pro naměřenou hodnotu napětí senzoru TI INA3221 42
5.4 Vodiče používané sběrnicí USB ve verzi 2.0 43
5.5 Typy portů dle specifikace Battery Charging Specification v1.2 43
5.6 Typy portů dle specifikace Apple Fast Charge 43

6.1 Konfigurace uložená v NVS 46
6.2 MQTT témata publikovaná napájecí jednotkou 46
6.3 MQTT témata odebíraná napájecí jednotkou 47
6.4 Implementované REST API endpointy napájecí jednotkou 47

7.1 Měření ukládaná do InfluxDB 52
7.2 Popis JWT tokenu používaného informačním systémem 54
7.3 Parametry `otpauth` URI pro TOTP 54
7.4 Implementované REST API endpointy pro systém centrální správy 56

D.1 Soupiska součástek 79
D.2 Pokračování soupisky součástek 80
D.3 Pokračování soupisky součástek 81
D.4 Ceny jednotlivých položek potřebných k výrobě jedné napájecí jednotky . . 81

Seznam zkratek

ADC Analog-to-digital converter, Analogově-digitální převodník. 20

API Application Programming Interface. 12, 15, 19

CD Continuous delivery. 58

CI Continuous integration. 58

CLI Command Line Interface. 50

CoAP Constrained Application Protocol. 34

CORS Cross-Origin Resource Sharing. 34

CRC Cyclic redundancy check. 27

CSV Comma-separated values. 16

DOM Document Object Model. 48

DPA Direct Peripheral Access. 28

DSSS Direct-sequence spread spectrum. 29

DTLS Datagram Transport Layer Security. 34

ESP-IDF ESP IoT Development Framework. 44

FIDO Fast IDentity Online. 54

GPIO General-purpose input/output. 45, 62

HTTP Hypertext Transfer Protocol. 10, 23, 32

IEEE Institute of Electrical and Electronics Engineers. 26, 29

IoT Internet of Things, Internet věcí. 30

IP Internet Protocol, Internetový protokol. 13

ISM Industrial, Scientific and Medical. 27, 28

ISTQB International Software Testing Qualifications Board. 58

I²C Inter-Integrated Circuit. 25

JSON JavaScript Object Notation. 12, 15

JWT JSON Web Token. 53

LDO Low Dropout. 61

LED Light-Emitting Diode. 61

M2M Machine-to-Machine. 30

MAC Media Access Control. 26

MIBs Management Information Bases. 30

MII Media-independent interface. 26

MIME Multipurpose Internet Mail Extensions. 33

MIMO Multiple-Input and Multiple-Output. 8, 29

MOSFET Metal Oxide Semiconductor Field Effect Transistor. 40

MQTT MQ Telemetry Transport. 10, 12, 13, 19, 24, 30, 35

MU-MIMO Multi-user MIMO. 29

NMS Network Management System. 30

NTP Network Time Protocol. 54

NVS Non-volatile Storage. 45

OFDM Orthogonal frequency-division multiplexing. 29

OFDMA Orthogonal frequency-division multiple access. 29

PDU Power Distribution Unit. 4, 11, 13

PHP PHP: Hypertext Preprocessor. 15, 50

PoE Power over Ethernet. 4, 12, 14, 15

PSRAM Pseudostatic RAM. 39

PXE Preboot Execution Environment. 10

QAM Quadrature amplitude modulation. 29

QoS Quality of Service. 31

RAM Random Access Memory. 8

REST Representational State Transfer. 12, 15, 19, 34

RISC Reduced Instruction Set Computer. 39

RMII Reduced MII. 26

RSSI Received signal strength indicator. 28

SD Secure Digital. 10

SHA Secure Hash Algorithm. 54

SNMP Simple Network Management Protocol. 12, 19, 30

SNTP Simple Network Time Protocol. 42, 45

SoC System on Chip. 16

SPI Serial Peripheral Interface. 25, 39

SRAM Static RAM. 39

TCP Transmission Control Protocol. 30, 33

TOTP Time-based one-time password. 54

TSDB Time series database. 51

UART Universal asynchronous receiver-transmitter. 39

UDP User Datagram protocol. 30

URI Uniform Resource Identifier. 6, 54

URL Uniform Resource Locator. 33

USB Universal Serial Bus. 4, 12, 16, 17, 19

USB-IF USB Implementers Forum. 43

XML Extensible Markup Language. 12

Kapitola 1

Úvod

Dříve elektronická zařízení byla založena na diskrétních logických obvodech a jednoduchých mikrokontrolérech. Nyní díky rozvoji tzv. Internetu věcí vzniká stále více zařízení založených na pokročilejších mikrokontrolérech, které mají různá rozhraní pro komunikaci s nadřazeným systémem (například Bluetooth Low-Energy, Wi-Fi či Ethernet) a používají různé komunikační protokoly (například MQTT nebo HTTP), a na jednodeskových počítačích. A tak vzniká potřeba při vývoji a testování jejich spolehlivosti monitorovat jejich napájení a v případě překročení provozních limitů informovat vývojáře, že došlo k překročení v daný časový okamžik, aby šlo jednodušeji v ladících záznamech dohledat důvod proč k překročení došlo.

V posledních letech se také čím dál častěji setkáváme s vestavěnými zařízeními založenými nad některým z jednodeskových počítačů, na kterých běží vlastní software, který řídí další připojené systémy, nebo dokonce vlastní operační systémy založené nad Yocto Projectem[1]. Při procesu vývoje takového řešení je časté testování výchozího stavu diskového obrazu SD karty s operačním systémem a dalšími programy, což může obnášet časté přepisování SD karty a manuální vypínání a zapínání jednodeskového počítače. Toto lze zautomatizovat pomocí bootu z počítačové sítě (např. pomocí technologie PXE a napájecí jednotky.

Kapitola 2 uvádí čtenáře do problematiky napájecích jednotek a podrobně se zabývá možnostmi spínání a měření výkonu a dalších elektrických veličin.

Kapitola 3 uvádí čtenáře do základních komunikačních vzorů používaných pro Internet věcí a dále jej seznamuje s nejpoužívanějšími komunikačními protokoly pro vzdáleny přístup a monitorování napájecích jednotek.

Kapitola 4 seznamuje čtenáře s architekturou navrženého systému a se všemi jeho částmi. A požadavky na jednotlivé části systému.

Kapitola 5 seznamuje čtenáře s technickou realizací hardwaru napájecí jednotky, popisuje jednotlivé použité komponenty a vysvětluje, proč byly zvoleny.

Kapitola 6 seznamuje čtenářem s implementací obslužného firmwaru napájecí jednotky a popisuje všechny její části.

Kapitola 7 seznamuje čtenáře s implementací softwarového řešení pro vzdálenou centrální správu napájecích jednotek.

Kapitola 8 nejdříve čtenáře uvádí do problematiky testování a dále jej seznamuje s principy a technologiemi, které byly pro testování navrženého řešení použity. Nakonec čtenáře seznamuje s možnostmi dalšího rozvoje projektu.

[1] https://www.yoctoproject.org/

Kapitola 2

Úvod do problematiky napájecích jednotek

V této kapitole je popsán základ problematiky napájecích jednotek. Dále jsou popsány jednotlivé typy a existující řešení napájecích jednotek a jejich výhody a nevýhody. A na konci kapitoly je čtenář seznámen s problematikou číslicového měření elektrických veličin.

2.1 Úvod do napájecích jednotek

Napájecí jednotkou PDU (*Power Distribution Unit*) myslíme zařízení pro rozvod napájecího napětí. Typicky je určené pro horizontální nebo vertikální montáž do 19" rozvaděče (tzv. racku) pro IT techniku, kde je nutné mít pod kontrolou napájení zařízení umístěných v těchto rozvaděčích.

Jednotky lze rozdělit podle vlastností na následující kategorie (toto členění je běžně používáno v průmyslu např. výrobci NETIO[20] a EATON[2]):

- **základní PDU (*Basic PDU*)** - zajišťují pouze distribuci napájecího napětí do několika zásuvek (typicky IEC-320 C13), jedná se pouze o klasické prodlužovací přívody, které jsou určeny pro montáž do 19" rozvaděčů,
- **PDU s měřením spotřeby na vstupu (*Metered Input PDU*)** - distribuují napájecí napětí a měří celkovou spotřebu na vstupu, typicky naměřené hodnoty jsou zobrazovány na displeji a nejsou připojeny k nadřazenému systému prostřednictvím Internetu,
- **PDU s měřením spotřeby na jednotlivých výstupních zásuvkách (*Monitored PDU* nebo *Metered outlet PDU*)** - distribuují napájecí napětí, měří spotřebu na jednotlivých výstupních zásuvkách, naměřené hodnoty jsou zobrazovány na displeji a typicky jsou připojeny k nadřazenému systému prostřednictvím Internetu,
- **PDU s možností spínat jednotlivé výstupní zásuvky (*Managed PDU* nebo *Switched PDU*)** - jedná se o rozšíření jednotek PDU s měřením spotřeby na jednotlivých výstupních zásuvkách o možnost jednotlivé zásuvky spínat.

2.2 Existující řešení

V tomto odstavci jsou popsány vlastnosti již existujících řešení, které lze použít jako napájecí jednotky pro jednodeskové počítače.

Aktuálně na trhu není dostupná žádná napájecí jednotka, která by mohla přímo napájet jednodeskové počítače. Proto je nutné použít napájecí jednotku, která pracuje se síťovým napětím a spínaný zdroj pro napájení jednodeskového počítače. To ovlivňuje přesnost měření, protože spínaný zdroj má svou účinnost, která není konstantní a dle jeho zatížení se může i výrazně měnit.

Dále je možné použít síťový přepínač s podporou napájení připojených zařízení pomocí PoE (*Power over Ethernet*), takto napájené zařízení však musí tento způsob napájení podporovat nebo musí být doplněno o externí zařízení tzv. PoE Splitter, které tuto funkcionalitu doplní. U PoE přepínačů je dále nutné rozlišovat jakou verzi standardu používají, protože v jednotlivých verzích se výrazně liší maximální odebíraný proud z jednoho portu, což je popsáno v tabulce 2.1. Danou verzi standardu musí pak podporovat jak přepínač, tak i napájené zařízení. V případě, že jedno ze zařízení podporuje pouze nižší verzi standardu, tak je použita nejvyšší verze standardu, kterou podporují obě zařízení.

Standard	**IEEE 802.af**	**IEEE 802.3at**	**IEEE 802.3bt**
Název	PoE	PoE+	PoE++
Maximální odebíraný výkon	12,95 W	25,5 W	51 W

Tabulka 2.1: Přehled standardů pro Power over Ethernet

Dále lze pro napájení použít běžný spínaný zdroj a spínat a měřit napájení jednodeskového počítače pomocí analyzátoru napájení nebo USB testeru, který je vybaven komunikačním rozhraním (např. Bluetooth či USB).

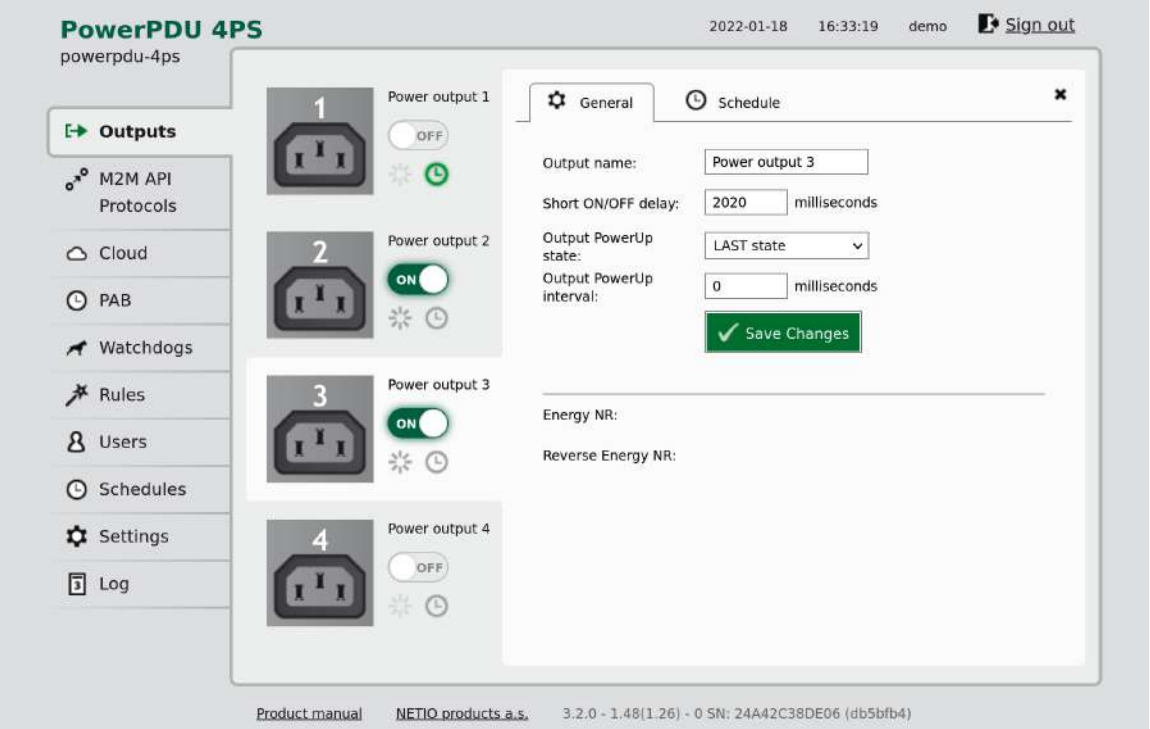
2.2.1 NETIO PowerPDU 4C

Jedná se napájecí jednotku vyráběnou českou firmou NETIO Products a.s. se čtyřmi měřenými a spínanými výstupy se zásuvkou IEC-320 C13 na zadní straně. Jednotka je určená pro horizontální montáž do rozvaděče, na výšku zabírá 1 pozici a má poloviční šířku a tak lze do jedné pozice umístit tyto jednotky dvě. Jednotku lze ovládat pomocí lokálního webového rozhraní, vzdáleného webového rozhraní prostřednictvím cloudového řešení výrobce, dále pomocí protokolů telnet, SNMP, Modbus/TCP, MQTT a REST API (JSON i XML) a také tlačítky, které jsou umístěny na čelní straně napájecí jednotky.

Obrázek 2.1: Fotografie PDU NETIO PowerPDU 4C převzatá ze stránek produktu[21]

Lokální webové rozhraní

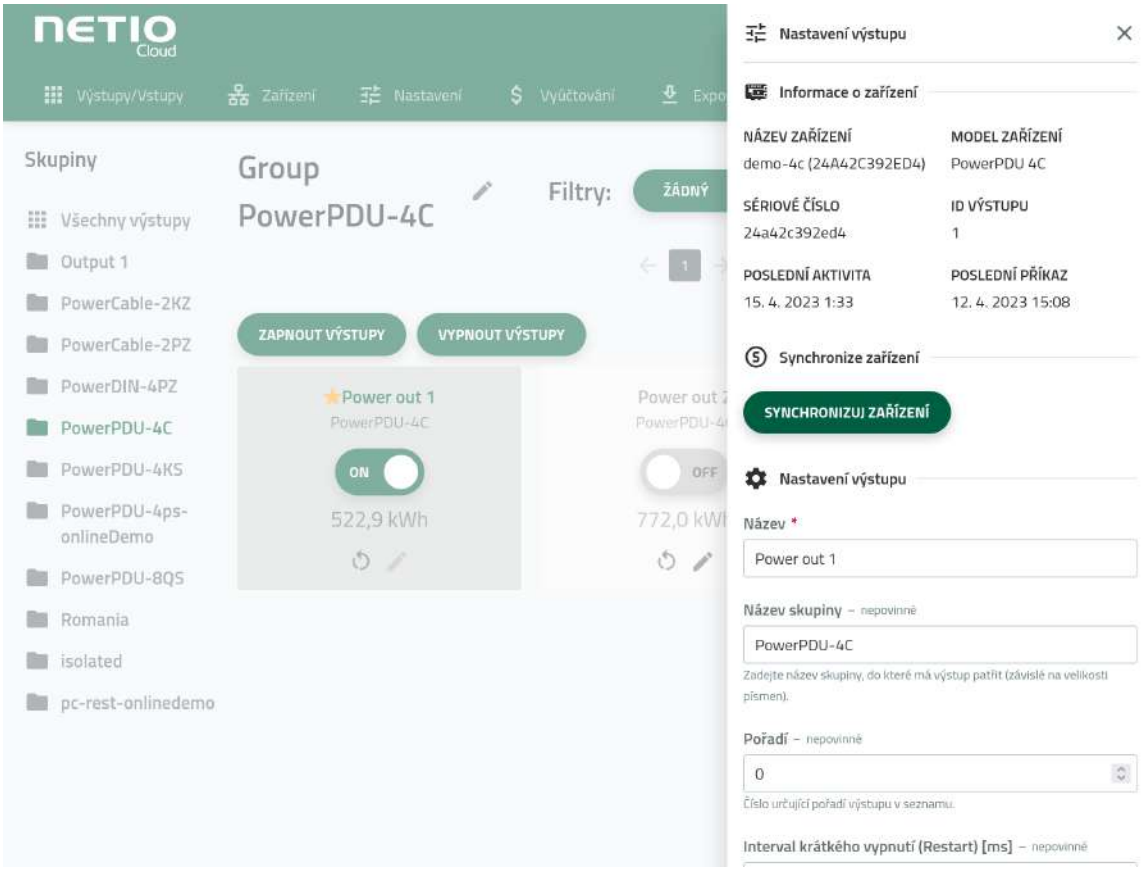
Lokální webové rozhraní umožňuje konfigurovat a spínat jednotlivé výstupy, zobrazit naměřené hodnoty, konfigurovat komunikační protokoly a připojení do cloudové služby výrobce. Dále také spravovat uživatelské účty a nastavovat watchdogy a plány. Lze například nastavit watchdog, který se při nedostupnosti dané IP adresy pokusí zařízení připojené k danému výstupu restartovat a odešle o tom e-mail. Dále lze nastavit odesílání denních zpráv o funkčnosti napájecí jednotky na e-mailovou adresu. Pomocí skriptovacího jazyku Lua si lze nadefinovat různé akce, například dle aktuálního odběru na jednom výstupu zapnout nebo vypnout jiný výstup a informovat o tom uživatele e-mailem. Dále lze prohlížet protokol událostí, který obsahuje záznamy generované samotnou jednotkou (například start jednotky, přihlášení do webového rozhraní, připojení do cloudu), tak i záznamy z uživatelských Lua skriptů.



Obrázek 2.2: Webové rozhraní PDU NETIO PowerPDU 4C

Cloudové řešení výrobce

Cloudové řešení umožňuje hromadnou správu napájecích jednotek jak z webového uživatelského rozhraní, tak také pomocí protokolu MQTT. V bezplatné verzi je však funkcionalita značně limitována, například nelze zobrazit aktuálně odebíraný elektrický výkon, ale pouze celkovou spotřebu na daném napájecím výstupu. Pomocí MQTT lze jednotlivé výstupy zapnout, vypnout a restartovat, odebírat aktuální stav výstupu, celkovou spotřebu výstupu v kilowatthodinách a aktuálně odebíraný výkon (pouze v placené variantě).



Obrázek 2.3: Webové rozhraní cloudu pro napájecí jednotku NETIO PowerPDU 4C

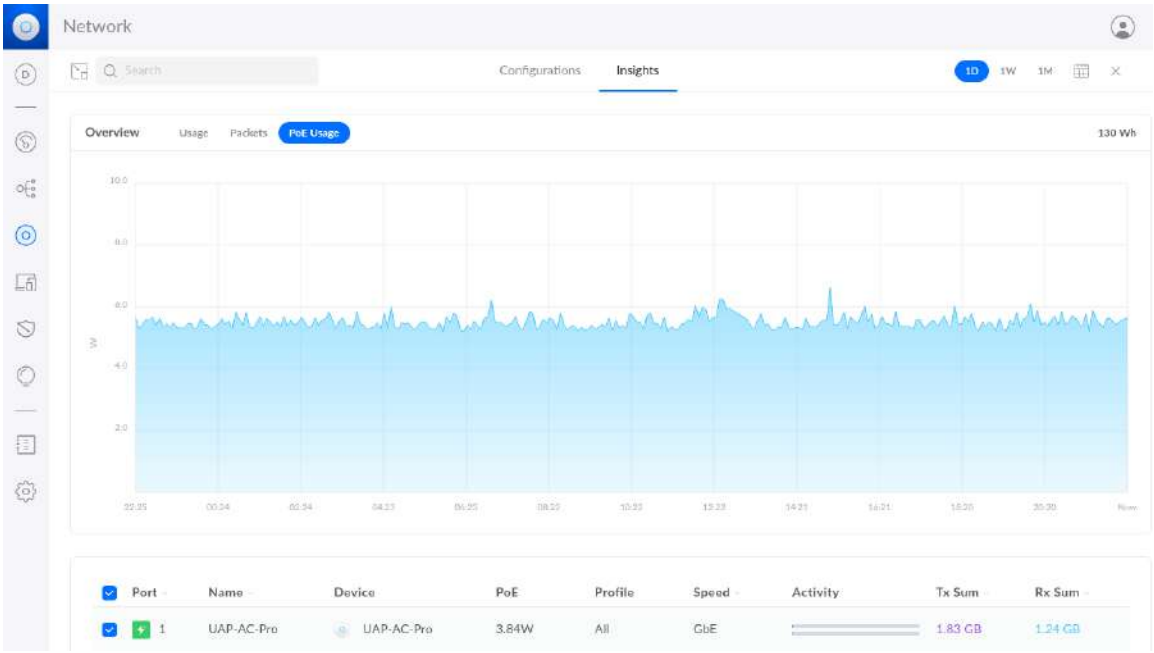
2.2.2 Ubiquiti UniFi Switch USW-Lite-8-PoE

Jedná se o 8-portový gigabitový L2 přepínač vyráběný americkou společností Ubiquiti Inc., který na prvních čtyřech portech podporuje napájení připojených zařízení pomocí PoE+ (*IEEE 802.3at*). Přepínač je spravován pomocí softwaru pro centrální správu, který se nazývá UniFi Network Application. S přepínačem lze komunikovat pomocí REST API UniFi Network Application nebo pomocí SNMP, které neumožňuje získat všechny metriky (např. odebíraný výkon z PoE portů).

Obrázek 2.4: Fotografie přepínače Ubiquiti UniFi Switch USW-Lite-8-PoE převzatá ze stránek produktu[33]

UniFi Network Application

UniFi Network Application (někdy také UniFi Controller) je multiplatformní software napsaný v programovacím jazyce Java pro centrální správu síťových prvků (přepínačů, směrovačů a Wi-Fi přístupových bodů) řady UniFi společnosti Ubiquiti Inc. Pro ukládání dat se používá open-source dokumentovou NoSQL databázi MongoDB. REST API není veřejně dokumentované a je primárně určeno pro frontend, který je napsán v skriptovacím jazyce JavaScript a používá knihovnu React, která slouží pro tvorbu uživatelských rozhraní. Existují komunitou spravované knihovny pro práci s tímto API - například Art-of-WiFi/UniFi-API-client[1] pro skriptovací jazyk PHP.



Obrázek 2.5: Zobrazení odebraného výkonu z PoE portů v UniFi Network Application

```
{
        "port_table": [
                {
                        "port_idx": 1,
                        "poe_current": "89.00", // Odebíraný proud z portu [mA]
                        "poe_power": "4.68", // Odebíraný výkon z portu [W]
                        "poe_voltage": "52.57" // Elektrické napětí na portu [V]
                }
        ],
        "name": "USW-Lite-8-PoE",
        "total_max_power": 52, // Výkon, který lze maximálně odebrat [W]
        "total_used_power": 6.05, // Celkový odebíraný výkon [W]
}
```

Výpis 2.1: Část JSON odpovědi na dotaz pro získání informací o zařízeních s údaji o PoE portu z REST API UniFi Network Application

[1] https://github.com/Art-of-WiFi/UniFi-API-client

2.2.3 Nordic Semiconductor Power Profiler Kit II

Jedná se o analyzátor napájení vyráběný norskou společností Nordic Semiconductor s USB komunikačním rozhraním. Analyzátor může fungovat pouze jako ampérmetr s rozsahem 200 nA až 1 A nebo také jako napájecí zdroj s napětím 0,8-5 V, analyzátor je napájen pomocí microUSB konektoru (při odběru větším než 500 mA je nutné připojit také druhý microUSB kabel, který pak slouží pouze pro napájení).

Analyzátor dokáže pouze měřit elektrický proud a rozlišení měřeného proudu je 100 nA nebo 1 mA dle nastaveném rozsahu. Vzorkovací frekvence je až 100 tisíc vzorků za sekundu, maximální doba vzorkování je 5 minut. Dále analyzátor obsahuje 8 digitálních vstupů, které lze použít jako logický analyzátor.

Výrobce pro práci s analyzátorem dodává aplikaci nRF Connect for Desktop[2], která je podrobněji popsaná níže.

Obrázek 2.6: Fotografie analyzátoru napájení Nordic Semiconductor Power Profiler Kit II, převzatá ze stránek produktu[22]

nRF Connect for Desktop

Jedná se o multiplatformní open-source aplikaci s vývojovými nástroji pro produkty společností Nordic Semiconductor. V aplikaci je několik různých vývojových nástrojů.

Nástroj Power Profiler je určen pro práci s analyzátory napájení Power Profiler Kit a Power Profiler Kit II. Tento nástroj umožňuje naměřená data exportovat do vlastního formátu nebo jako CSV či snímek obrazovky. Nástroj může protékající proud vzorkovat určitou vzorkovací frekvencí po uživatelem nastavený čas nebo začít vzorkovat proud po překročení určité meze proudu po dobu až 52 ms.

Nástroj Bluetooth Low Energy, který je určen k testování zařízení, které pro komunikaci používají tento standard. Dále tato aplikace obsahuje nástroj pro programování nRF SoC.

[2] https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-desktop

Obrázek 2.7: Snímek obrazovky z nRF Connect Power Profileru

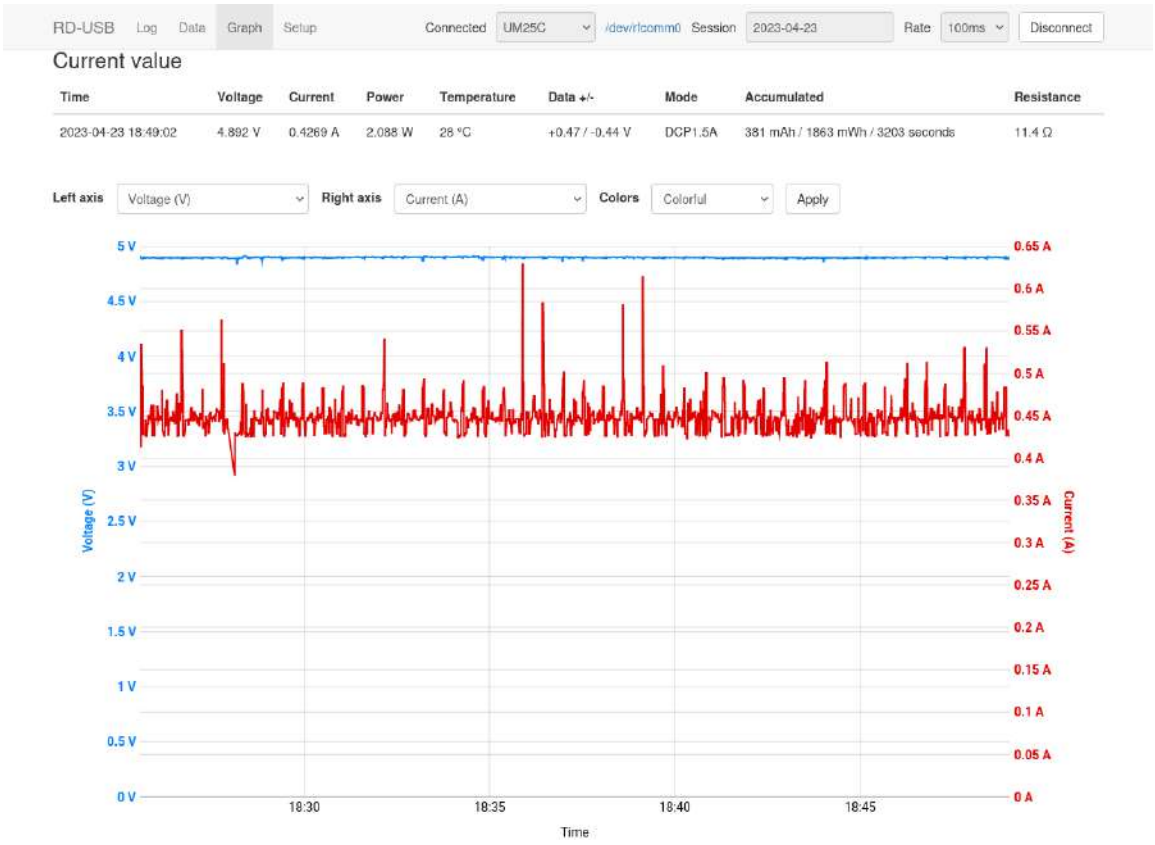
2.2.4 RuiDeng UM25C

Jedná se o USB tester vyráběný čínskou firmou Hangzhou Ruideng Technology Co., Ltd. s Bluetooth bezdrátovou komunikací, které využívá profilu pro emulovanou sériovou linku (*SPP - Serial Port Profile*) s přenosovou rychlostí 9600 baudů. Díky tomu lze se zařízením komunikovat jak pomocí výrobcem dodávaných aplikací, tak i pomocí další nástrojů - například pomocí open-source nástroje pro analýzu signálů sigrok[3][29] nebo pomocí webové aplikace kolinger/rd-usb[4].

Tester dokáže měřit napětí 4-24 V s přesností 1 mV a proud až 5 A s přesností 0,1 mA. Vzorkovací frekvence je dva vzorky za sekundu. Tester dokáže identifikovat protokol použitý pro vyjednání napájecího napětí a maximálního proudu, které může připojené zařízení odebírat, podporuje standardy Qualcomm Quick Charge 2.0 a 3.0, Apple Fast Charging a USB BC 1.2 DCP. Jako vstup lze použít jeden ze tří typů vstupních konektorů (USB A, microUSB B a USB-C). Jako napájecí výstup lze použít buď USB A nebo USB-C konektor.

Obrázek 2.8: Fotografie USB testeru RuiDeng UM25C

[3] https://sigrok.org/wiki/Main_Page
[4] https://github.com/kolinger/rd-usb



Obrázek 2.9: Snímek obrazovky z webové aplikace rd-usb

2.2.5 Porovnání existujících řešení

Vlastnost	NETIO PowerPDU 4C	Ubiquiti UniFi Switch USW-Lite-8-PoE	Nordic Semiconductor Power Profiler Kit II	RuiDeng UM25C
Typ	Napájecí jednotka	PoE přepínač	Analyzátor napájení	USB tester
Napájecí vstup	IEC-320 C14	DC jack	pinová lišta nebo microUSB	USB A, microUSB nebo USB-C
Napájecí výstupy	4 (IEC-320 C13)	4 (RJ45)	1 (pinová lišta)	1 (USB A nebo USB-C)
Napětí na výstupu	230 V AC	50-57 V DC	0.8-5,0 V DC	dle připojeného zdroje (4-24 V DC)
Maximální odebíraný příkon z výstupu	2300 W	25,5 W	5 W	záleží na připojeném zdroji a zařízení, max. 100 W
Maximální odebíraný příkon z výstupů	2300 W	52 W	5 W	záleží na připojeném zdroji a zařízení, max. 100 W
Lze přímo připojit jednodeskový počítač?	Ne, je nutný napájecí zdroj	Možná, pokud zařízení nepodporuje PoE, tak je nutné použít PoE Splitter	Ano	Ne, je nutný napájecí zdroj
Měřené veličiny	Proud, napětí, výkon, energie, účiník a frekvence	Proud, napětí a výkon	Proud	Proud, napětí, výkon a energie
Umožňuje přímé měření napájení jednodeskového počítače?	Ne	Ne	Ano, pouze protékající elektrický proud	Ano
Komunikační protokoly	telnet, SNMP, MQTT, Modbus/TCP, REST API	SNMP a REST API	USB	Bluetooth SPP 9600 Bd
Cena	8 337 Kč	2 736 Kč	2 170,32 Kč	644,94 Kč
Cena za 1 výstup	2 084,25 Kč	684 Kč	2 170,32 Kč	644,94 Kč

Tabulka 2.2: Porovnání existujících řešení

2.3 Měření elektrických veličin

V tomto odstavci jsou popsány principy měření elektrických veličin ve stejnosměrných obvodech - elektrického napětí, proudu a příkonu. Protože elektrické veličiny jsou spojité, tak je musíme pomocí nějakého zařízení zdiskretizovat, abychom jsme mohli s naměřenou hodnotou pracovat v mikroprocesoru, který rozumí pouze diskrétním signálům, a tímto zařízením je analogově-digitální převodník.

2.3.1 Princip fungování analogově-digitálního převodníku

Analogově-digitální převodník (zkráceně ADC nebo A/D převodník) je zařízení, které slouží pro převod spojitého (analogového) signálu na signál diskrétní (digitální). Umožňuje zpracování analogových signálů na číslicových počítačích. Princip převodu spojitého signálu na diskrétní probíhá ve dvou fázích, nejdříve se provede vzorkování vstupního signálu a poté následuje kvantování.

Vzorkování

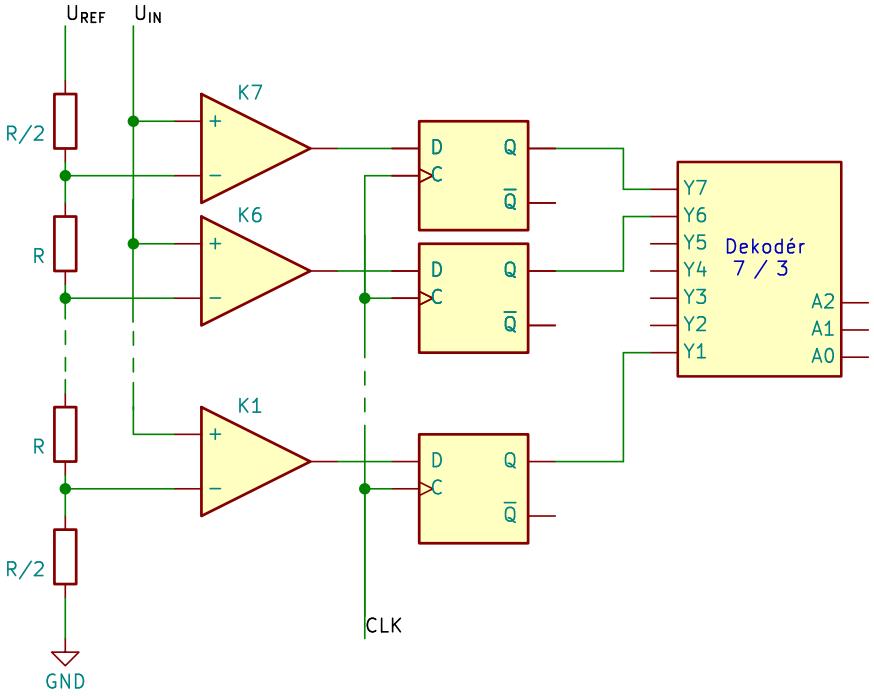
Vzorkování je proces, při kterém se v pravidelných intervalech zjišťuje okamžitá hodnota vstupního signálu. Počet vzorků za sekundu vyjadřuje vzorkovací frekvence, která musí být vhodně zvolena. Dle Shannonova teorému měla by být minimálně dvojnásobná oproti nejvyšší harmonické frekvenci vzorkovaného signálu. Příliš vysoká vzorkovací frekvence klade vysoké nároky na paměť pro uchovávání vzorků a také na rychlost daného převodníku. Příliš nízká vzorkovací frekvence nenávratně zkresluje průběh vzorkovaného signálu kvůli jevu, který se nazývá aliasing.

Kvantování

Kvantování je proces, při kterém se převádí vzorek na výsledný binární kód. Rozlišení je dáno počtem bitů výsledného výstupu a vyjadřuje citlivost daného převodníku, přičemž n-bitový převodník dokáže rozlišit až 2^n hodnot. Čím vyšší je rozlišení, tím nižší je tzv. kvantizační krok, který označuje rozdíl mezi dvěma po sobě jdoucími úrovněmi. Rozdíl mezi skutečnou hodnotou a kvantizovanou hodnotou se nazývá chyba kvantování, která je vyjádřena v absolutní hodnotě a může nabývat nejvýše poloviny kvantizačního kroku.

Paralelní analogově-digitální převodník

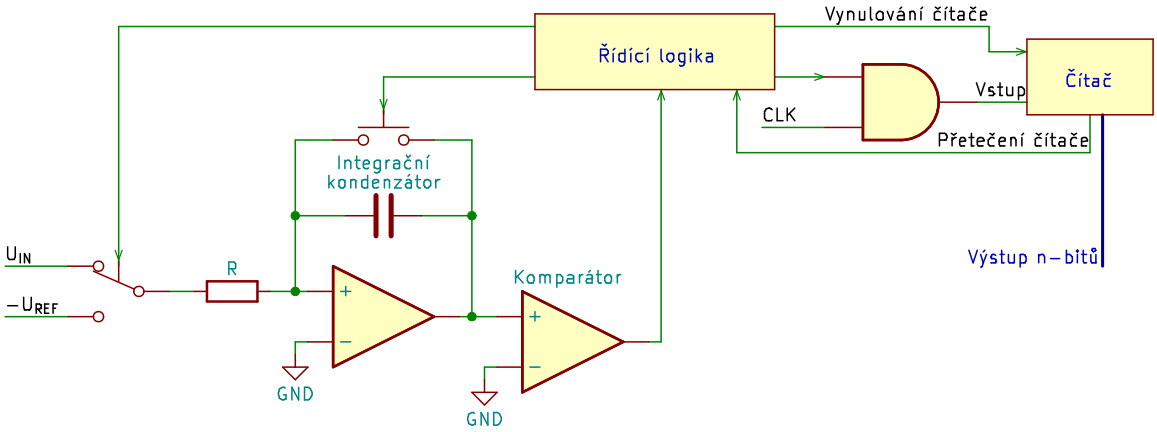
Paralelní analogově-digitální převodník je zástupcem komparačních převodníků a je nejrychlejším typem převodníku, protože převod trvá přesně jeden hodinový takt. Kvantování vstupního signálu probíhá v komparátorech, kterých musí být použito 2^{n-1}, kde n je rozlišení převodníku. Tyto komparátory porovnávají vstupní napětí (U_{IN}) s odstupňovaným referenčním napětím (U_{REF}), toto odstupňování je zajištěno pomocí odporové sítě. Výstup komparátorů je zapsán do D-klopných obvodů s vzestupnou hranou hodinového signálu. Výstupy jednotlivých klopných obvodů jsou zapojeny do dekodéru, který z hodnot jednotlivých vstupů převede informaci na výsledný kód pro další zpracování. Je vhodný pro zpracování rychlých signálů.



Obrázek 2.10: Schéma paralelního analogově-digitálního převodníku

Analogově-digitální převodník s dvojitou integrací

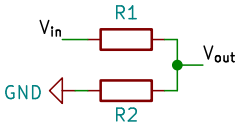
Analogově-digitální převodník s dvojitou integrací je zástupcem integračních převodníků, je vhodný pro měření konstantních a pomalu se měnících signálů, proto se používá v multimetrech pro měření elektrického napětí a v teplotních (RTD) senzorech. Převod probíhá ve dvou krocích. Na začátku převodu je čítač vynulovaný a integrační kondenzátor vybitý. V prvním kroku je na vstup integrátoru přivedeno vstupní napětí (U_{IN}) a čítač čítá impulzy hodinového signálu. Dohází k nabíjení integračního kondenzátoru do přetečení čítače. Po přetečení čítače následuje druhý krok, při kterém se přepne vstup integrátoru na referenční napětí, které má opačnou polaritu než vstupní napětí. Tím se integrační kondenzátor začne vybíjet a až do jeho vybití čítač čítá hodinový signál. Vybitím integračního kondenzátoru se ukončuje převod a hodnota čítače vyjadřuje velikost měřeného signálu.



Obrázek 2.11: Schéma analogově-digitálního převodníku s dvojitou integrací

2.3.2 Měření elektrického napětí

Stejnosměrné elektrické napětí je měřeno pomocí analogově digitálního převodníku, který měří napětí pouze do nějaké stanovené referenční hodnoty (např. 1,8 V, 3,3 V), a proto měřené napětí je potřeba snížit pomocí napěťového děliče. Hodnota rezistoru, na kterém měříme může být uvedena v katalogovém listu použitého analogově-digitálního převodníku. Hodnota U_{out} je dána vzorcem $U_{out} = U_{in} \cdot \frac{R_2}{R_1+R_2}$.



Obrázek 2.12: Schéma napěťového děliče

2.3.3 Měření elektrického proudu

Stejnosměrný elektrický proud lze měřit buď pomocí měření úbytku napětí na bočníku nebo pomocí Hallovy sondy. Měření pomocí Hallovy sondy není vhodné pro měření malých proudů, typicky se tohoto principu využívá pro měření proudů v řádu desítek ampér. Měření pomocí Hallovy sondy využívají například některé klešťové multimetry nebo senzor Allegro Microsystems ASC712[5].

2.3.4 Měření pomocí senzorů s komunikačním rozhraním

Pro měření elektrických veličin můžeme využít také specializovaných integrovaných obvodů, které obsahují analogovou část pro úpravu vstupního signálu (například PGA (*Programmable-gain amplifier*)), analogově-digitální převodník, registry pro uchování změřených hodnot a komunikační rozhraní (typicky I²C nebo SPI).

Nejčastěji se můžeme setkat se senzory Texas Instruments INA219[6] (pro měření jednoho kanálu) nebo INA3221[7] (pro měření až tří kanálů), které mají I²C rozhraní. Senzor Texas Instruments INA3221 je podrobněji popsán v sekci 5.6.

Tyto obvody mohou mít registr, do kterého lze nastavit mezní přípustnou hodnotu. Překročení je typicky signalizováno na specifickém výstupu obvodu, který může řídící mikrokontrolér vyčítat pomocí přerušení. Případně tato funkcionalita může být doplněna externím integrovaným obvodem - tzv. eFuse (*elektronickou pojistkou*) nebo USB spínačem.

2.4 Možnosti spínání napájení

Napájení jednodeskového počítače lze spínat pomocí elektromagnetického relé, které může mít jeden nebo více spínacích kontaktů (například pro spínání napájecího napětí i země), P-channel MOSFET tranzistoru (pro spínání napájecího napětí), N-channel MOSFET tranzistoru (pro spínání země) nebo pomocí specializovaných obvodů - tzv. USB spínačů. Příkladem takového obvodu Texas Instruments TPS2000C[8], který podrobněji popsán v sekci 5.8.

[5] https://www.allegromicro.com/en/products/sense/current-sensor-ics/zero-to-fifty-amp-integrated-conductor-sensor-ics/acs712

[6] https://www.ti.com/product/INA219

[7] https://www.ti.com/product/INA3221

[8] https://www.ti.com/product/TPS2000C

Kapitola 3

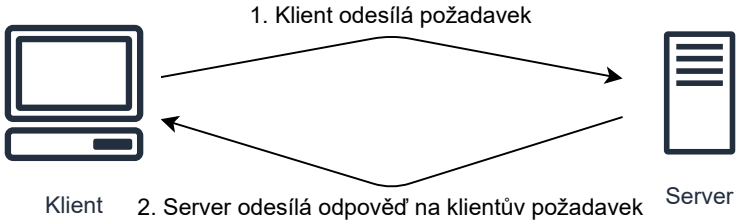
Úvod do komunikačních technologií pro napájecí jednotky

V této kapitole jsou nejdříve popsány základní komunikační vzory a poté podrobněji popsány jednotlivé technologie pro komunikaci a protokoly používané pro vzdálený přístup a monitorování činnosti napájecích jednotek.

3.1 Základní komunikační vzory

3.1.1 Model Požadavek-Odpověď *(Request-Response)*

Komunikační model Požadavek-Odpověď se využívá při komunikaci mezi dvěma zařízeními, kde první zařízení (tzv. klient) pošle požadavek na zařízení druhé (tzv. server) a server následně klientovi odpoví. V tomto modelu se klient snaží od serveru získat nějakou informaci nebo na serveru provést nějakou akci. Server tento požadavek přijme, provede potřebné akce pro vytvoření odpovědi klientovi a poté tuto odpověď klientovi pošle. Tento komunikační model využívá například protokol HTTP, který je podrobněji popsán níže v podkapitole 3.4.3.

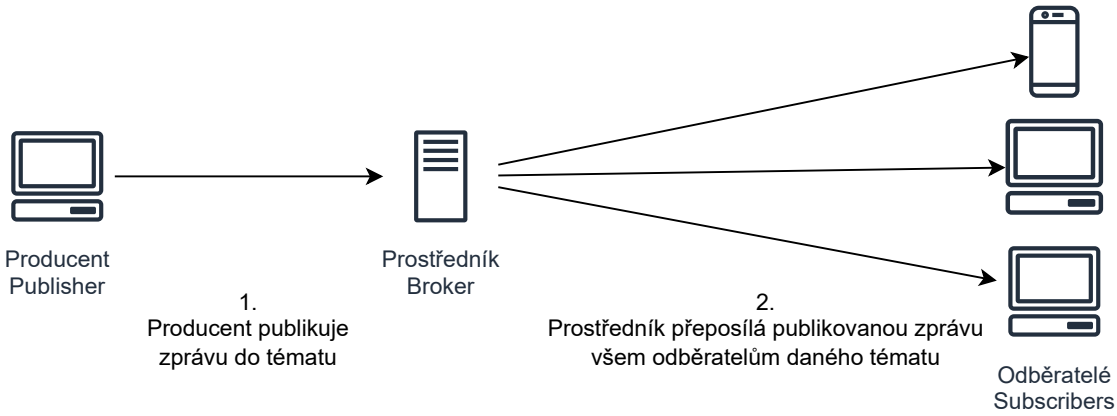


Obrázek 3.1: Diagram komunikačního modelu Požadavek-Odpověď

3.1.2 Model Producent-Odběratel *(Publish-Subscribe)*

Komunikační model Producent-Odběratel umožňuje různým zařízením publikovat a přihlašovat se k odběru zpráv z určitých témat (topic). Tento model se využívá zejména u distribuovaných systémů, kde různé zařízení či aplikace chtějí komunikovat mezi sebou, ale nechtějí či nemohou být přímo propojeny, proto jsou propojeny přes prostředníka - tzv. broker. Broker zajišťuje autentizaci, autorizaci připojených producentů a odběratelů a přijímá od producentů zprávy, které pak přeposílá všem odběratelům tématu, do kterého byla

zpráva odeslána. Tento komunikační model využívá například protokol MQTT, který je podrobněji popsán níže v podkapitole 3.4.2.



Obrázek 3.2: Diagram komunikačního modelu Producent-Odběratel

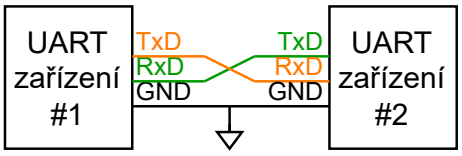
3.1.3 Model *Push-Pull*

Jedná se o modifikaci modelu Producent-Odběratel, kde producenti ví o všech svých odběratelích a naopak, producenti komunikují přímo s odběrateli. Producenti vkládají (push) nové zprávy do fronty, ze které si je pak odběratelé postupně vytahují (pull).

3.2 Technologie pro drátovou komunikaci

3.2.1 UART

UART (*Universal asynchronous receiver-transmitter*) je asynchronní sériová sběrnice. Přenos dat probíhá mezi 2 zařízeními po dvou vodičích `TxD` a `RxD`, které se mezi komunikujícími zařízeními kříží.



Obrázek 3.3: Schéma zapojení zařízení komunikujících pomocí UART sběrnice bez hardwarového řízení toku dat

Prvním vodičem je `TxD` (*Transmit Data*), přes který vysílač odesílá data přijímači. Druhým vodičem je `RxD`, pomocí kterého přijímač přijímá data od vysílače. Hodiny vysílače a přijímače jsou synchronizovány přes start (log. 0) a stop bity (log. 1) v jednotlivých rámcích. Nejčastěji se setkáváme s rychlostí (tzv. *Baud rate*) 9 600 Bd nebo 115 200 Bd a konfigurací `8N1`, která znamená 8 datových bitů, žádný paritní bit a jeden stop bit.

Délka v bitech	1	5-9	0-1	1-2
Popis	Start bit	Datový rámec	Parita	Stop bit

Tabulka 3.1: UART rámec

Dále obě zařízení mohou být propojeny dalšími dvěma vodiči (`RTS` a `CTS` nebo `DTR` a `DSR`), které slouží pro hardwarové řízení toku dat.

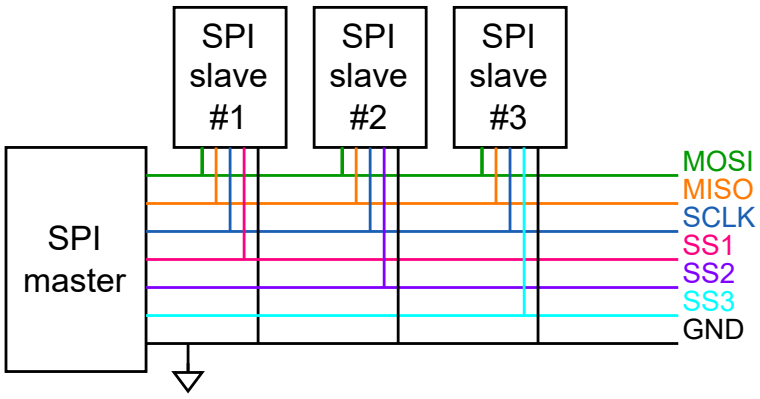
3.2.2 SPI

SPI (*Serial Peripheral Interface*) je čtyřvodičová synchronní sériová sběrnice, kterou řídí tzv. master, který generuje hodinový signál a pomocí vodiče `SS` (*Slave Select*) určuje s jakým připojeným zařízením (tzv. slave) bude komunikovat.

Sběrnice používá čtyři vodiče, které jsou popsány v tabulce 3.2. Komunikace je zahájena masterem přechodem do log. 0 na vodiči `SS` pro dané slave zařízení. Tímto se slave zařízení připojí ke sběrnici a master může se slave komunikovat (například vyčítat hodnoty z registrů), komunikace je ukončena přechodem do log. 1. na vodiči `SS`.

Zkratka	**Název**	**Popis**
`MOSI`	Master Out, Slave In	Data odeslaná masterem pro slave
`MISO`	Master In, Slave Out	Data odeslaná slavem pro master
`SCKL`	Serial Clock	Hodinový signál generovaný masterem
`SS` nebo `CS`	Slave/Chip Select	Signál pro výběr slave zařízení

Tabulka 3.2: Vodiče používané SPI sběrnicí

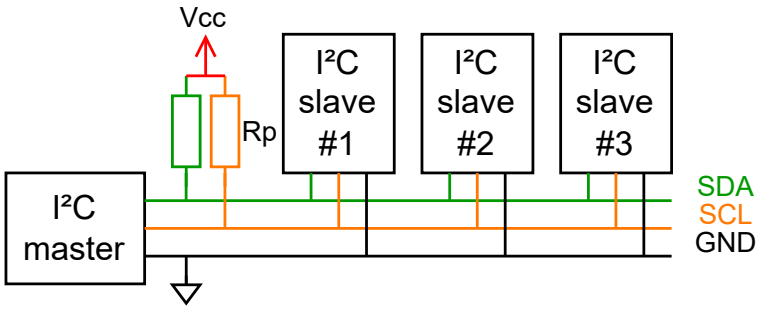


Obrázek 3.4: Schéma zapojení zařízení komunikujících pomocí SPI sběrnice

3.2.3 I²C

I²C (*Inter-Integrated Circuit*) je synchronní sériová směrnice, kterou řídí tzv. master a který může komunikovat s dalšími připojenými zařízeními - tzv. slaves. Slave zařízení je adresováno pomocí adresy o délce 7 bitů, takže master může komunikovat až s 128 slave zařízeními.

Sběrnice používá dva vodiče - SDA (*Serial Data Line*) a SCL (*Serial Clock Line*). SDA i SCL používají výstup typu otevřený kolektor, takže je nutné na oba vodiče umístit tzv. pull-up rezistory, aby komunikace byla vůbec možná. Hodinový signál SCL je generován masterem a typicky se jedná o obdélníkový signál o frekvencí 100 kHz nebo 400 kHz.



Obrázek 3.5: Schéma zapojení zařízení komunikujících pomocí I²C sběrnice

Komunikace je zahájena přechodem z log. 1 na log. 0 na signálu SDA, zatímco je signál SCL v log. 1. Během komunikace je dovoleno měnit hodnotu signálu SDA pouze když se nachází signál SCL v log. 0. Master pomocí R/W bitu sděluje slave zařízení, zda-li bude z něj číst (log. 1) nebo bude do něj zapisovat (log. 0). Slave zařízení pomocí tzv. ACK/NACK bitu potvrzuje příjem dat, při log. 0 byl bajt úspěšně přijat, při log. 1 bajt nebyl přijat žádným zařízením. Komunikace je ukončena přechodem z log. 0 na log. 1 na signálu SDA, zatímco je signál SCL v log. 1.

Délka v bitech	8	1
Popis	Data	ACK/NACK

Tabulka 3.3: I2C datový rámec

Délka v bitech	1	7	1	1	9·N	1
Popis	Start	Adresa	R/W	ACK/NACK	N datových rámců	Stop

Tabulka 3.4: I2C rámec

3.2.4 Ethernet

Ethernet je nejpoužívanějším protokolem fyzické a linkové vrstvy pro drátové počítačové sítě, je definován ve standardech IEEE 802.3. V dnešní době je fyzická vrstva realizována nad kroucenou dvojlinkou nebo optickým kabelem. Zařízení jsou v síti Ethernet adresována pomocí tzv. MAC (*Media Access Control*) adresy o délce 48 bitů. Data jsou přenášena na linkové vrstvě v tzv. Ethernetových rámcích, které jsou podrobněji popsány v tabulce 3.5. A na fyzické vrstvě v tzv. Ethernetových paketech.

Mikrokontroléry se do sítě Ethernet mohou připojit následujícími dvěma způsoby:

- Pomocí řadiče, který v sobě obsahuje implementaci fyzické i linkové vrstvy, se kterým typicky komunikují pomocí sběrnice SPI. Příkladem takového řadiče je WIZnet W5100.
- Pomocí řadiče fyzické vrstvy, se kterým komunikuje linková vrstva implementovaná v mikrokontroléru pomocí sběrnice MII (*Media-independent interface*) nebo RMII (*Reduced MII*). Příkladem takového řadiče je Microchip LAN8720A a příkladem takového mikrokontroléru je Espressif Systems ESP32-D0WD-V3, který je obsažen v bezdrátovém modulu Espressif Systems ESP32-WROOM-32D.

Název	MAC adresa cíle	MAC adresa zdroje	802.1Q tag (volitelný)	EtherType	Data	Kontrolní součet CRC32
Velikost	6 oktetů	6 oktetů	4 oktety	2 oktety	46-1500 oktetů	4 oktety

Tabulka 3.5: Ethernetový rámec

3.3 Technologie pro bezdrátovou komunikaci

V této sekci si přestavíme několik technologií pro bezdrátovou komunikaci, které jsou používané zařízeními Internetu věcí.

3.3.1 LoRa

LoRa (z anglického *Long Range*) je proprietární protokol fyzické vrstvy pro zařízení Internetu věcí, který původně vyvinula francouzská společnost Cycleo, která později byla odkoupena společností Semtech. Protokol používá spread spectrum modulaci na ISM (*Industrial, Scientific, and Medical*) radiofrekvenčních pásmech 868 MHz (v Evropě), 915 MHz (v Severní Americe a Asii) a 2,4 GHz (celosvětově). Díky použité modulaci je možné odesílat odesílat malé množství dat s malou spotřebou a malou rychlostí (do 50 kb/s) na velkou vzdálenost.

LoRaWAN

LoRaWAN je standardizovaný protokol linkové vrstvy, který se používá nad fyzickou vrstvou poskytovoanou protokolem LoRa. Protokol je definován ve standardu ITU-T Y.4480[1], o vývoj protokolu se stará nezisková organizace LoRa Alliance. Protokol je navržen tak, aby zpracování dat probíhalo v cloudu.

The Things Network[2] je komunitní projekt, který implementuje cloudové jádro pro zpracování dat a umožňuje uživatelům provozovat brány, které přijímají zprávy ze zařízení a které je pak přeposílají do cloudu pro další zpracování.

Pokud bychom chtěli data zpracovávat sami, tak lze použít open-source projekt ChirpStack[3], který implementuje jádro sítě a umožňuje spravovat připojená zařízení a brány, zpracovávat přijatá data a přeposílat je do cloudů třetích stran.

3.3.2 IQRF

Technologie IQRF byla původně v roce 2004 vyvinuta českou společností MICRORISC s.r.o., od roku 2017 vývoj má na starosti dceřiná společnost IQRF Tech s.r.o. Jedná se technologii, která využívá mesh topologii, kde některé uzly jsou propojeny s více než uzly v síti. Tato topologie poskytuje redundanci propojení, takže komunikace je nadále možná i v případě výpadku některých uzlů či linek.

Síť je řízena tzv. koordinátorem a v jedné síti je možné mít až dalších 239 uzlů, které mohou směrovat provoz (typicky zařízení napájené ze sítě). Jedná se o technologii pro odesílání malých množství dat (v jednom paketu lze odeslat až 64 bajtů), malou rychlostí

[1] https://handle.itu.int/11.1002/1000/14818-en?locatt=format:pdf&auth
[2] https://www.thethingsnetwork.org
[3] https://www.chirpstack.io/

(19,2 kbit/s) a velmi nízkou spotřebou. V síti se používají protokoly IQMESH a DPA, který je podrobněji popsán níže. Pro komunikaci se zařízeními v síti Internet se využívají tzv. brány, které jsou typicky postaveny nad jednodeskovým počítačem, na kterém běží linuxová distribuce a software IQRF Gateway Daemon, který slouží pro překlad DPA na JSON API, skrze které s ním lze komunikovat pomocí MQ, MQTT nebo WebSocketu.

Využívá se radiofrekvenční pásmo ISM, konkrétně se jedná o frekvence 868 MHz pro použití v Evropě, 916 MHz pro použití v Americe a 433 Mhz, které lze použít všude na světě. Bezdrátové moduly používají 16-bitové mikrokontroléry PIC16, které byly původně vyvinuté společností Microchip, kterou v roce 2016 odkoupila konkurenční společnost Atmel. Na bezdrátových modulech běží velice jednoduchý operační systém IQRF OS, který poskytuje funkcionalitu jak pro protokoly IQMESH a DPA, tak i pro uživatelské aplikace.

IQRF DPA

Direct Peripheral Access (DPA)[19] je jednoduchý binární protokol pro správu periferií zařízení, formát odesílaných zpráv je znázorněn v tabulkách 3.6 a 3.7. Každé zařízení implementuje tzv. vestavěné periferie (např. OS pro práci s IQRF OS), některá zařízení mohou implementovat některou ze standardních periferií (pro binární výstupy, osvětlení, senzory nebo pro protokol DALI) nebo mohou implementovat své vlastní periferie.

Název	`NADR`	`PNUM`	`PCMD`	`HWPID`	`PDATA`
Velikost	2 B	1 B	1 B	2 B	0-58 B

Tabulka 3.6: Schéma paketu požadavku IQRF DPA

Název	`NADR`	`PNUM`	`PCMD`	`HWPID`	`ERRN`	`DPAVAL`	`PDATA`
Velikost	2 B	1 B	1 B	2 B	1 B	1 B	0-56 B

Tabulka 3.7: Schéma paketu odpovědi IQRF DPA

- `NADR` - Přidělená adresa zařízení v IQMESH síti, ačkoliv se jedná o dvoubajtovou hodnotu, tak se nižší bajt nevyužívá. Například `0x0100` pro zařízení na adrese 1.
- `PNUM` - Číslo periferie, se kterou budeme pracovat. Například `0x5E` pro standardní senzor.
- `PCMD` - Číslo příkazu periferie, který chceme vykonat. V požadavcích je nejvyšší bit nastaven vždy nastaven na `0` a odpovědích je nastaven nejvyšší bit na `1`. Například `0x01` pro vyčtení všech senzorů na zařízení včetně jejich typů.
- `HWPID` - Identifikátor hardwarového profilu, který označuje typ zařízení, čehož se využívá pro získání implementovaných periferií a pro filtrování zpráv pro specifický typ zařízení. V případě použití `0xFFFF` zprávu zpracují všechna zařízení.
- `ERRN` - Číslo chyby, která nastala. Pokud žádná chyba nenastala, tak hodnota je nulová.
- `DPAVAL` - DPA hodnota, která například může vracet RSSI, lze ji pomocí speciálního požadavku nastavit.
- `PDATA` - Volitelná data periferie.

IQRF Standard Binary Output

Jedná se o standardní periferii, která umožňuje ovládat binární výstupy. Číslo periferie je `0x4B`. Zařízení může implementovat až 32 výstupů. Periferie implementuje příkazy pro nastavení a získání stavu výstupů a enumeraci výstupů.

IQRF Standard Sensor

Jedná se o standardní periferii pro sběr senzorických dat. Číslo periferie je `0x5E`. Zařízení může implementovat až 32 senzorů. Periferie implementuje příkazy pro vyčtení naměřených hodnot senzory, volitelně včetně jejich typů a pro enumeraci senzorů.

3.3.3 WiFi

WiFi je nejrozšířenějším standardem fyzické a linkové versty pro bezdrátové počítačové sítě, definována pomocí standardů IEEE 802.11, využívá bezlicenční radiofrekvenční pásma ISM 2,4, 5 nebo 6 GHz. Ve WiFi sítích se objevují 2 typy zařízení - přístupové body (AP, *Access Point*), které poskytují přístup do počítačové sítě, a koncová zařízení (STA, *Station*), která se připojují k přístupovým bodům.

První verze standardu vznikla v roce 1997 a od té doby vzniklo několik revizí tohoto standardu. Vestavěná zařízení typicky podporují revizi Wi-Fi 4 (IEEE 802.11n), které využívá frekvenční pásmo 2,4 GHz a poskytuje více než dostatečnou šířku pásma pro přenos dat. Nejpoužívanější řešení pro připojení vestavěných zařízení k Wi-Fi síti je pomocí bezdrátových modulů Espressif Systems ESP8266 nebo ESP32.

Název	**Revize**	**Rok**	**Frekvence**	**Šířka pásma**	**Maximální rychlost**	**Modulace**
	802.11	1997	2,4 GHz	22 MHz	2 Mb/s	DSSS
	802.11a	1999	5 GHz	20 MHz	54 Mb/s	OFDM
	802.11b	1999	2,4 GHz	22 MHz	11 Mb/s	DSSS
	802.11g	2003	2,4 GHz	20 MHz	54 Mb/s	OFDM
Wi-Fi 4	802.11n	2009	2,4/5 GHz	20/40 MHz	600 Mb/s	MIMO OFDM (64-QAM)
Wi-Fi 5	802.11ac	2013	5 GHz	20/40/ 80/160 MHz	3466,8 Mb/s	MU-MIMO OFDMA (256-QAM)
Wi-Fi 6	802.11ax	2021	2,4/5 GHz	20/40/ 80/160 MHz	9608 Mb/s	MU-MIMO OFDMA (1024-QAM)
Wi-Fi 6E	802.11ax	2021	6 GHz	20/40/ 80/160 MHz	9608 Mb/s	MU-MIMO OFDMA (1024-QAM)

Tabulka 3.8: Revize standardu IEEE 802.11

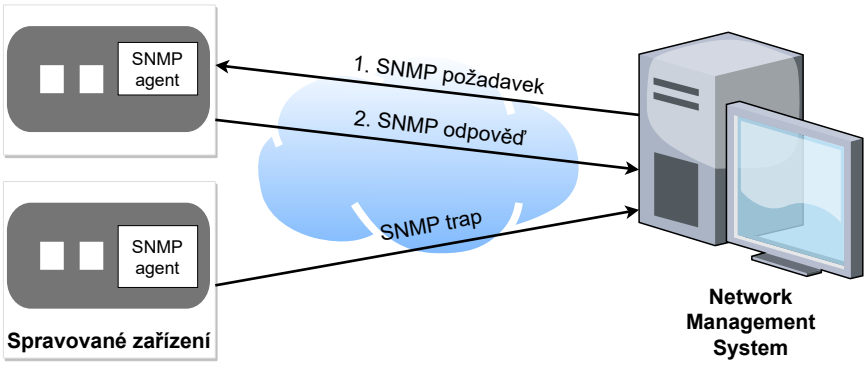
3.4 Internetové komunikační protokoly

3.4.1 SNMP

Simple Network Management Protocol (SNMP) je protokol pro monitoring a správu zařízení (např. směrovačů, přepínačů a serverů) počítačových sítí. Jako transportní vrstvu používá protokol UDP, standardně běží na portu 161. Umožňuje průběžné získávání dat ze síťových zařízeních pro potřeby správy sítě a jejich správu. Existují 3 verze tohoto protokolu:

- SNMPv1 - zajišťuje základní funkcionalitu
- SNMPv2 - přidává autentizaci
- SNMPv3 - přidává šifrování

Na vyčítaných zařízeních běží tzv. SNMP agent, který zpřístupňuje informace o zařízení pomocí tohoto protokolu. Informace ze zařízení pak vyčítá a ukládá do databáze tzv. SNMP manager (známý jako NMS - *Network Management System*), příklady open-source implementací jsou Zabbix[4] či LibreNMS[5]. Informace nemusí být pouze vyčítány pomocí manageru, v případě splnění předem definované podmínky (např. odpojení připojeného zařízení z definovaného portu) může agent o této situaci informovat manager pomocí tzv. pasti na událost (Trap). Informace jsou uspořádány hierarchicky, tato hierarchie a přidružená metadata jsou popsána pomocí tzv. MIBs (*Management Information Bases*).



Obrázek 3.6: Architektura SNMP protokolu

3.4.2 MQTT

MQ Telemetry Transport (MQTT, dříve Message Queuing Telemetry prokotol) je jednoduchý internetový protokol pro IoT (Internet of Things, internet věcí) a M2M (Machine-to-Machine) komunikaci. Původně byl vyvinutý firmou IBM v roce 1999 a v roce 2013 byl standardizován organizací OASIS a stal se z něj otevřený standard.

Používá komunikační model Producent-Odběratel, jako transportní vrstvu používá TCP a standardně běží na portu 1883 a jeho šifrovaná verze pak na portu 8883. Připojená zařízení se připojují k centrálnímu bodu (tzv. MQTT brokeru). Pomocí protokolu lze odesílat zprávy až do velikosti necelých 256 MB (268 435 455 bajtů), ale broker může tento limit snížit.

[4] https://www.zabbix.com/
[5] https://www.librenms.org/

Jako broker se nejčastěji používá open-source implementace mosquitto[6], která je spravována neziskovou organizací Eclipse Foundation[7].

MQTT témata

Zprávy jsou uspořádány do hierarchických témat, hierarchie je dělena pomocí lomítek. Hierarchii témat je nutné si dopředu promyslet, aby případní odběratelé se mohli přihlásit k odběru jedné či více podúrovním témat pomocí tzv. žolíků `+` a `#`.

Žolík `+` nahrazuje jednu úroveň hierarchie, například `sbc_pdu/+/status` nám umožní získat stav všech napájecích jednotek.

Žolík `#` nahrazuje jednu čí více úrovní hierarchie a musí být uveden vždy jako poslední. Například pomocí tématu `sbc_pdu/#` můžeme odebírat všechna témata, které publikují připojené napájecí jednotky.

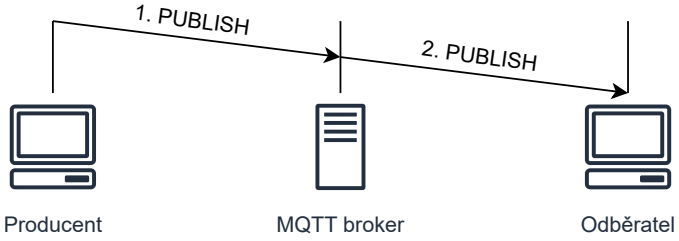
Pokud jméno tématu začíná znakem `$`, tak se jedná o tzv. speciální téma, které je určeno pro zprávy publikované samotným MQTT brokerem. Nejčastěji taková témata začínají na `$SYS/` a nalezeme zde různá metadata.

Stupně kvality služby

Jelikož je protokol je navržen aby fungoval i v nestabilních sítích (například s vyšší ztrátovostí paketů), tak obsahuje mechanismus, pomocí kterého lze určit kvalitu služby (QoS - *Quality of Service*) odesílaných zpráv. Protokol definuje následující 3 stupně kvality služby:

- 0 - At most once - Maximálně jednou,
- 1 - At least once - Minimálně jednou,
- 2 - Exactly once - Právě jednou.

Při odeslání zprávy se stupněm kvality služby 0, odesílatel zprávu odesílá brokeru pomocí zprávy `PUBLISH`, že broker má zprávu přeposlat (také pomocí zprávy `PUBLISH`) odběratelům daného tématu a nekontrolovat, zda-li ji přijali. Jedná se o nejefektivnější způsob doručení, ale doručení není garantováno.

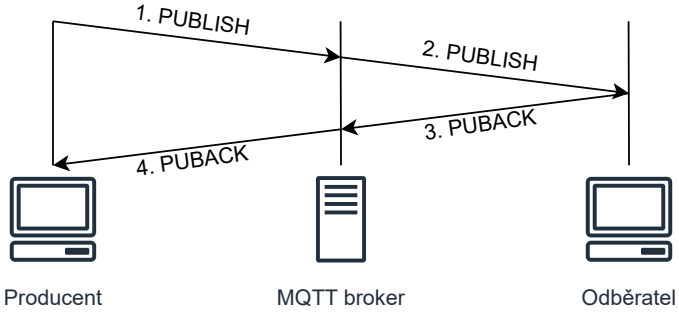


Obrázek 3.7: Diagram komunikace při odeslání MQTT zprávy s QoS 0

Při odeslání zprávy se stupněm kvality služby 1, odesílatel pošle brokeru zprávu a čeká až broker odeslání potvrdí. Broker zprávu přijme a přepošle ji odběratelům daného tématu, přeposlání proběhne s QoS 1 (případně s QoS 0, když odběratel požaduje QoS 0). Po

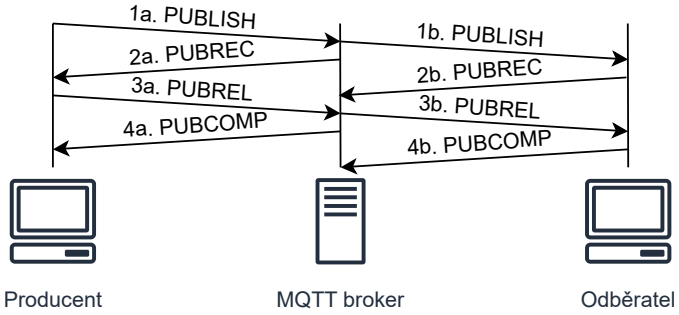
[6] https://mosquitto.org/
[7] https://www.eclipse.org/

obdržení zprávy odběratelé příjem potvrdí pomocí `PUBACK` a poté broker pomocí `PUBACK` odeslání odesílateli potvrdí. Potvrzení pomocí `PUBACK` broker může poslat dříve než odběratelé potvrdí příjem zprávy, záleží jaké chování používaný broker implementuje.



Obrázek 3.8: Diagram komunikace při odeslání MQTT zprávy s QoS 1

Při odeslání zprávy se stupněm kvality služby 2, producent pošle brokeru zprávu a čeká až broker odeslání potvrdí. Broker zprávu přijme a přepošle ji odběratelům daného tématu, přeposlání proběhne s QoS 2 (případně nejvyšším společným QoS s odběratelem). Po obdržení zprávy příjemci příjem potvrdí pomocí `PUBREC`. Pokud odesílatel nedostane od příjemce potvrzení, odesílá zprávu znovu s příznakem `DUP`, který označuje, že se jedná o opakované odeslání zprávy. Příjemce pomocí `PUBREL` potvrdí odesílateli, že potvrzení dostal a odesílatel komunikaci uzavře pomocí `PUBCOMP`. Je garantováno, že zpráva bude doručena právě jednou, ale za cenu rychlosti a komplexnosti.



Obrázek 3.9: Diagram komunikace při odeslání MQTT zprávy s QoS 2

Poslední vůle a závěť

Protokol MQTT obsahuje mechanizmus, pomocí kterého může zařízení, po odpojení od MQTT brokeru dát vědět pomocí odeslání určité zprávy, že již dané zařízení není připojeno. Tento mechanismus se nazývá Poslední vůle a závěť (z anglického *Last Will and Testament*).

3.4.3 HTTP

Hypertext Transfer Protocol (HTTP) je internetový protokol primárně určený pro přenos hypertextových dokumentů ve značkovacím jazyce HTML (*Hypertext Markup Language*) a dalších souborových formátů. Jiné souborové formáty jsou přenášeny pomocí rozšíření

MIME (*Multipurpose Internet Mail Extensions*), které původně vzniklo pro potřeby elektronické pošty. Toto rozšíření umožňuje pomocí HTTP hlavičky `Content-Type` definovat o jaký souborový formát se jedná. Využívá komunikační model Požadavek-Odpověď. Existuje několik verzí tohoto protokolu:

- HTTP/1.0 - pro každý požadavek se vytváří nové TCP spojení, textový přenos přes TCP,
- HTTP/1.1 - rozšiřuje verzi 1.0 o znovupoužití ustanoveného TCP spojení pro více požadavků,
- HTTP/2 - používá komprimovaný binární přenos, používá více TCP spojení naráz pro rychlejší načítání dat,
- HTTP/3 - používá přenos pomocí protokolu QUIC (RFC 9000[8]), který funguje nad protokolem UDP a zajišťuje spolehlivé doručování dat, které protokol UDP sám o sobě negarantuje.

URL

Jednotlivé zdroje, které HTTP server poskytuje jsou identifikovány pomocí URL (*Uniform Resource Locator*), které je definováno v RFC 1738[9]. Syntax URL vypadá následovně: `schéma`://[`uživatel`@]hostitel[:`port`]/`cesta`[?`dotaz`][#`fragment`], volitelné položky jsou v hranatých závorkách, jednotlivé části jsou podrobněji popsány v tabulce 3.9.

Část	Popis
`schéma`	Schéma označuje účel URL, v případě protokolu HTTP se jedná o `http` nebo `https` při použití šifrovaného přenosu.
`uživatel`	Volitelné uživatelské jméno, případně dvojice uživatelské jméno a heslo oddělena dvojtečkou.
`hostitel`	Adresa hostitele, může se jednat o doménové jméno, IPv4 nebo IPv6 adresu, která je pak zaobalena v hranatých závorkách
`port`	Volitelný port, na kterém se nachází server na daném hostiteli, u schématu `http` se implicitně jedná o port 80 a u schématu `https` o port 443
`cesta`	Sekvence segmentů oddělena lomítky, které identifikují daný zdroj v hierarchické struktuře
`dotaz`	Volitelný dotaz, který je typicky používán pro filtrování kolekce, při použití více filtrů jsou navzájem odděleny ampersandem (`&`)
`fragment`	Volitelný odkaz na část HTML dokumentu s daným `id`.

Tabulka 3.9: Popis jednotlivých částí URL

HTTP metody

HTTP definuje několik metod, které lze nad jednotlivými zdroji provést, sémantika jednotlivých metod je popsána v RFC 9110[10]. Definovány jsou následující metody:

- **GET** - slouží k získání informací o zdroji identifikovaného příslušnou URL,

[8] https://datatracker.ietf.org/doc/html/rfc9000
[9] https://datatracker.ietf.org/doc/html/rfc1738
[10] https://datatracker.ietf.org/doc/html/rfc9110

- **HEAD** - slouží k získání metadat o zdroji identifikovaného příslušnou URL,
- **POST** - slouží k vytváření nového zdroje a k vykonávání akcí nad zdroji,
- **PUT** - slouží k úpravě zdroje identifikovaného příslušnou URL,
- **DELETE** - slouží k odstranění zdroje identifikovaného příslušnou URL,
- **CONNECT** - slouží k tzv. HTTP Tunnelingu, kdy přes protokol HTTP je tunelovaný TCP/IP provoz,
- **OPTIONS** - slouží pro získání metod, které lze nad zdrojem provést, využívá se pro CORS (*Cross-Origin Resource Sharing*),
- **TRACE** - slouží k ladění nastavení webového serveru, v odpovědi je kopie požadavku, který server přijal,
- **PATCH** - slouží pro částečnou úpravu zdroje, můžeme tím ušetřit šířku pásma potřebnou pro úpravu.

REST

Representational state transfer (REST) je architektonický styl pro vývoj webových aplikací, které v roce 2000 Roy Fielding představil ve své dizertační práci[5]. Aby aplikace splňovala používala tento architektonický styl, tak musí splňovat následující požadavky:

- **Klient-Server** (*Client-Server*) — musí používat architekturu Klient-Server, klient a server jsou navzájem nezávislí,
- **Bezstavovost** (*Stateless*) — server nesmí zaznamenávat stav klienta,
- **Mezipamět** (*Cache*) — server musí používat mezipaměť a měl by označovat, která data klient si může ukládat do své mezipaměti,
- **Jednotné rozhraní** (*Uniform Interface*) - server vystavuje klientovi požadavky jednotným a předvídatelným způsobem,
- **Vícevrstvý systém** (*Layered System*) - prostřednici mezi klientem a serverem neovlivňují chování.

Tento styl plně využívá metody, které mu poskytuje protokol HTTP pro popis akcí prováděnými nad danými zdroji, které jsou identifikovány pomocí URL.

3.4.4 CoAP

Constrained Application Protocol (CoAP) je internetový protokol pro komunikaci mezi zařízeními s omezenými prostředky (jako jsou senzory, aktuátory a další zařízení Internetu věcí) v omezené síti (například s vysokou ztrátovostí paketů). Je definován v RFC 7252[11], vychází z protokolu HTTP, jako transportní vrstvu používá protokol UDP a volitelné šifrování je zajištěno pomocí DTLS (*Datagram Transport Layer Security*). Standardně používá port 5683 pro základní verzi a port 5684 pro šifrovanou verzi.

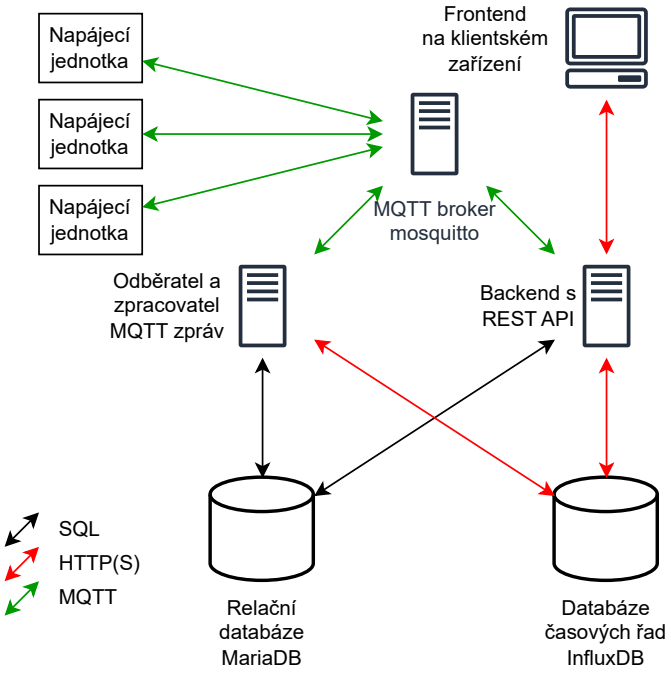
[11] https://datatracker.ietf.org/doc/html/rfc7252

Kapitola 4

Návrh architektury řešení

Úkolem systému, jak je zmíněno v úvodu této práce je vzdálené monitorování a řízení napájení pro jednodeskové počítače, které jsou napájeny pomocí stejnosměrného zdroje s elektrickým napětím 5 V. Navržená napájecí jednotka je vestavěné zařízení Internetu věcí, která komunikuje s informačním systémem, který umožňuje provádět hromadnou správu napájecích jednotek a sbírá periodicky odesílané naměřené hodnoty. Proto se práce skládá z několika fází: návrhu a realizace hardwaru, návrhu a implementace obslužného firmwaru řídícího mikrokontroléru, výběru komunikačního protokolu, návrhu a implementace informačního systému pro centrální správu a testování.

Řešení lze rozdělit na dva základní bloky - samotnou napájecí jednotku a informační systém pro centrální správu napájecích jednotek. Tyto dva bloky jsou navzájem propojeny pomocí protokolu MQTT a zpráv, které jsou přes tento protokol posílány. MQTT zprávy slouží pro periodické odesílání naměřených hodnot elektrických veličin z napájecích výstupů a pro vzdálené ovládání jednotlivých výstupů.



Obrázek 4.1: Blokový diagram architektury navrženého systému

4.1 Požadavky na napájecí jednotku

Napájecí jednotka musí splňovat následující požadavky:

- musí obsahovat alespoň dva napájecí výstupy pro jednodeskové počítače, které jsou napájeny pomocí stejnosměrného napětí o hodnotě 5 V,
- musí být schopna jednotlivé napájecí výstupy zapnout, vypnout a měřit na nich elektrické veličiny (elektrické napětí a protékající elektrický proud),
- musí být schopna komunikovat s nadřazeným systémem, do kterého bude periodicky posílat naměřené data a pomocí kterého bude možné ovládat jednotlivé výstupy.

Technická realizace hardwaru napájecí jednotky je podrobněji popsána v kapitole 5 a implementace obslužného firmwaru je podrobněji popsána v kapitole 6.

4.2 Požadavky na komunikační protokol

Protokol pro komunikaci mezi napájecí jednotkou a centrální správou musí splňovat následující požadavky:

- musí to být bezpečný, lehký, robustní, standardizovaný protokol,
- musí používat komunikační model Producent-Odběratel, protože se jedná o nejvhodnější model pro hromadnou správu,
- musí umožňovat odesílat z každé jednotky minimálně 2 zprávy za minutu,
- musí zajišťovat autorizaci připojených klientů.

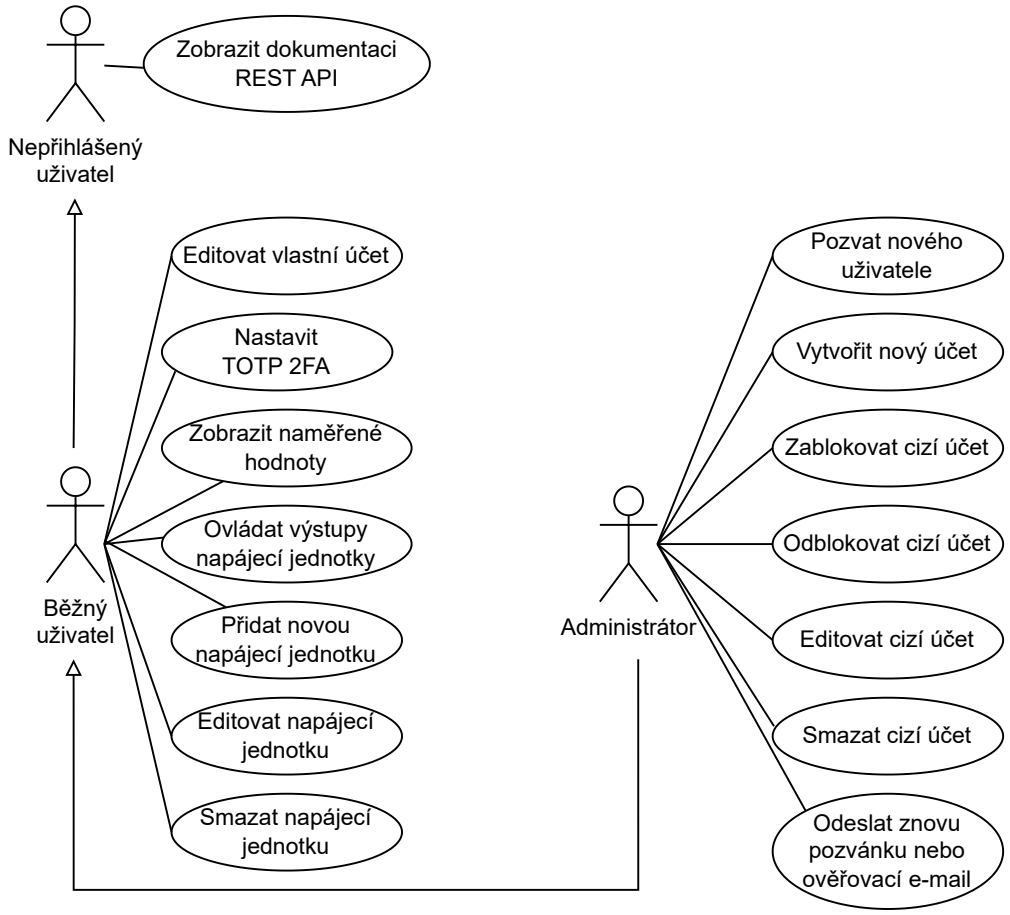
Na základě těchto požadavků jsem vybral komunikační protokoly WiFi a MQTT. Jelikož se jedná o zařízení, které bude napájeno ze sítě, tak energetická náročnost komunikačního rozhraní nemusí být řešena.

4.3 Požadavky na centrální správu napájecích jednotek

Softwarové řešení pro centrální správu napájecích jednotek musí splňovat následující požadavky:

- musí umožňovat spravovat více napájecích jednotek z jednoho místa,
- musí sbírat naměřené hodnoty z napájecích jednotek do databáze,
- musí umožňovat tyto hodnoty poté uživateli zobrazit v grafu a umožnit uživateli zvolit časové okno,
- musí umožňovat více uživatelům pracovat se systémem a administrátorovi umožňovat jejich správu,
- musí umožňovat lokalizaci uživatelského rozhraní,
- musí umožňovat uživatelům si nastavit dvoufaktorovou autentizaci (zejména administrátorům).

Implementace softwarového řešení pro centrální správu napájecích jednotek je podrobněji popsána v kapitole 7.



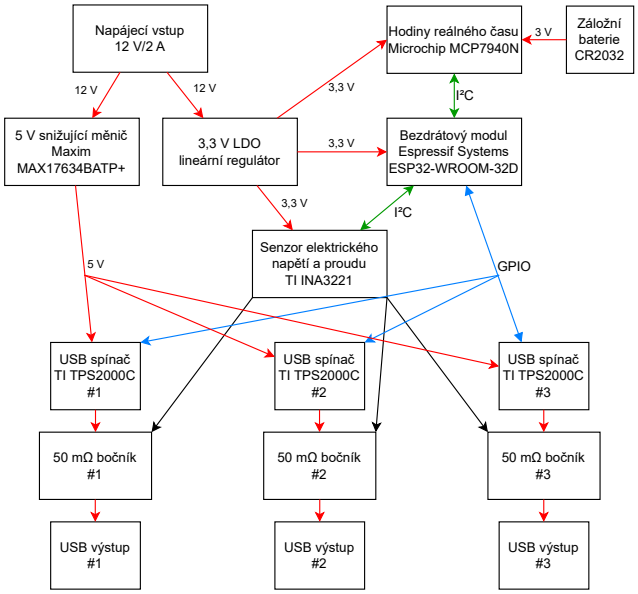
Obrázek 4.2: Diagram případu užití systému pro centrální správu napájecích jednotek

Kapitola 5

Technická realizace hardwaru napájecí jednotky

Napájecí jednotka je realizována na navržené desce plošných spojů. Pro tvorbu obvodového schématu a návrhu desky plošných spojů byl využit open-source nástroj KiCAD[1]. Jednotka je napájena přes 5,5/2,1 mm DC jack pomocí externího spínaného zdroje s výstupním napětím 12 V a výstupním elektrickým proudem alespoň 2 A. Vstupní napájecí napětí je sníženo na 5 V pomocí synchronního snižovacího DC-DC měniče Maxim MAX17634BATP+. Napájecí jednotka je řízena pomocí mikrokontroléru ve WiFi bezdrátovém modulu Espressif Systems ESP32-WROOM-32D. Elektrické veličiny na jednotlivých výstupech jsou měřeny pomocí senzoru Texas Instruments INA3221. Jednotlivé napájecí výstupy realizovány jako USB A konektor a jsou spínány pomocí USB spínače Texas Instruments TPS2000C.

5.1 Blokový diagram napájecí jednotky



Obrázek 5.1: Blokový diagram napájecí jednotky

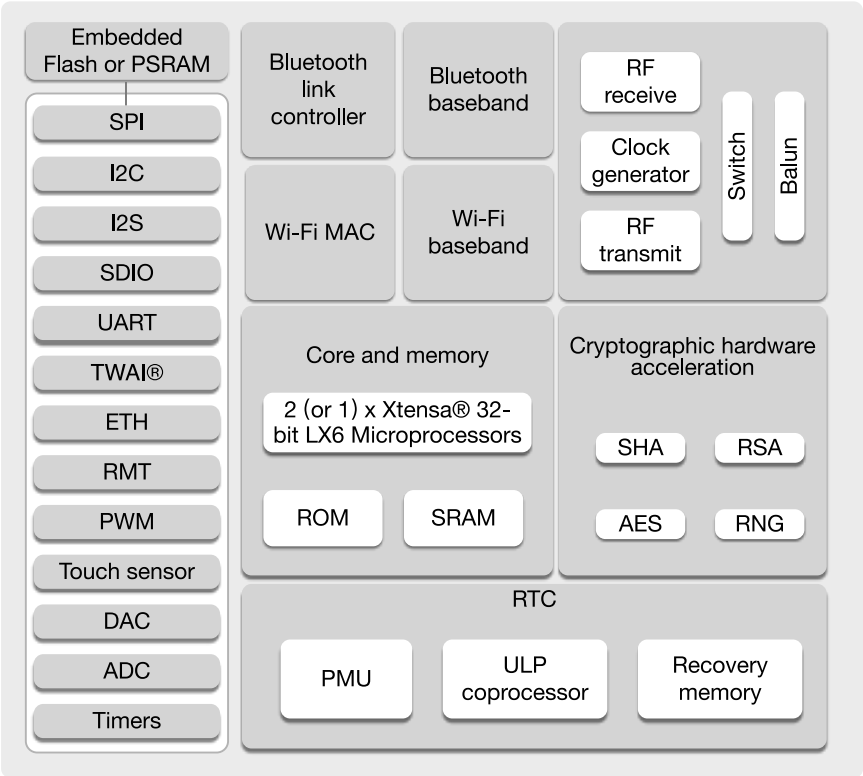
[1] https://www.kicad.org/

5.2 Bezdrátový modul Espressif ESP32-WROOM-32D

Napájecí jednotka je řízena pomocí WiFi bezdrátového modulu ESP32-WROOM-32D od firmy Espressif Systems, který obsahuje dvoujádrový mikrokontrolér ESP32-D0WD od stejné firmy a 16 MB SPI flash paměti, kde je uložen program. Bezdrátový modul byl zvolen, protože se jedná o levné a populární řešení pro zařízení Internetu věcí, které obsahuje relativně výkonný mikrokontrolér, základní HW akceleraci pro kryptografii a široké spektrum komunikačních protokolů.

Mikrokontrolér ESP32-D0WD obsahuje dvě 32-bitová RISC jádra Tensilica Xtensa LX6, které běží na frekvenci 160 nebo 240 MHz a ULP (*Ultra Low Power*) koprocesor, který se využívá když je mikrokontrolér uspán. Mikrokontrolér obsahuje několik periferií, které jsou vyobrazeny na blokovém diagramu 5.2, pro komunikační sběrnice SPI, I²C, I²S, UART, RMII (pro připojení integrovaného obvodu pro řízení fyzické vrstvy Ethernetu), JTAG (pro ladění firmwaru). Dále obsahuje dva analogově-digitální převodníky a jeden digitálně-analogový převodník, WiFi a Bluetooth.

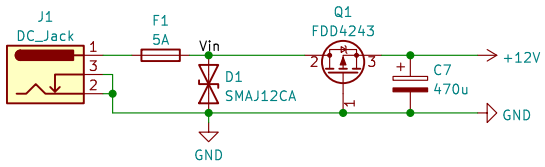
Mikrokontrolér neobsahuje žádnou flash paměť pro uložení vykonávaného programu, takže pomocí sběrnice SPI je připojena externí flash paměť (podporované velikosti jsou 2, 4, 8 a 16 MB). Mikrokontrolér obsahuje 520 kB SRAM (*Statické RAM*) paměti, kterou lze dále rozšířit až o 4 MB PSRAM (*Pseudostatické RAM*, jedná se o dynamickou RAM, která obsahuje interní obvod pro opětovné nabíjení parazitní kapacity - tzv. refresh, zvenku se tváří jako statická RAM) připojené přes SPI. Mikrokontrolér může být naprogramován přes sběrnici UART nebo JTAG.



Obrázek 5.2: Blokové schéma bezdrátového modulu Espressif ESP32 převzaté z katalogového listu[4]

5.3 Napájecí vstup

Napájecí vstup je řešen pomocí standardního 5,5/2,1 mm DC jack konektoru, pomocí kterého je připojen externí 12 V spínaný zdroj, který musí umožňovat napájecí jednotce odebrat proud alespoň 2 A. Hned za vstupním konektorem se nachází pojistka a transil SMAJ12CA, který slouží jako přepěťová ochrana - v případě přepětí vyšším než 13,3 V vytvoří zkrat, který přepálí předřazenou pojistku a tak ochrání jednotku. Ochrana proti přepólování je řešena pomocí výkonového P-channel MOSFET tranzistoru, který propustí napájení pouze při správné polaritě, oproti řešení pomocí diody má toto řešení výhodu v mnohem nižším úbytku napětí.



Obrázek 5.3: Část schématu s napájecím vstupem

5.4 Snižující DC-DC měnič Maxim MAX17634BATP+

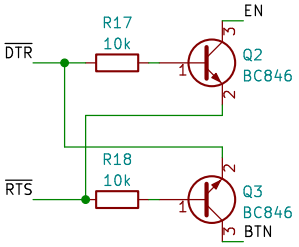
Jelikož je napájecí jednotka napájena pomocí stejnosměrného 12 V zdroje, tak je nutné napájecí napětí snížit, aby bylo možné napájet připojená zařízení pomocí stejnosměrného napětí 5 V. K tomuto účelu slouží synchronní snižující (tzv. buck nebo step-down) DC-DC měnič Maxim MAX17634BATP+, který vstupní napětí 6,5-36 V sníží na 5 V, maximální proud, který lze z něj odebírat je 4,25 A. Návrh je odvozen od referenčního návrhu vývojové sady Maxim MAX17634BEVKIT#[16][2].

5.5 USB-UART převodník Silicon Labs CP2102N

Pro nahrání obslužného firmware do napájecí jednotky je použit USB-UART převodník CP2102N od firmy Silicon Labs. Převodník obsahuje fyzickou vrstvu pro USB, generátor hodinového signálu a 3,3 V napěťový regulátor, podporuje USB ve verzi 2.0. Pro funkční automatické nahrávání a reset mikrokontroléru je potřeba přidat externí obvod, který se skládá ze dvou NPN bipolárních tranzistorů a dvou 10 kΩ rezistorů.

DTR	RTS	EN	GPIO0
0	0	1	1
0	1	1	0
1	0	0	1
1	1	1	1

Tabulka 5.1: Pravdivostní tabulka obvodu pro automatické přepnutí do programovacího módu

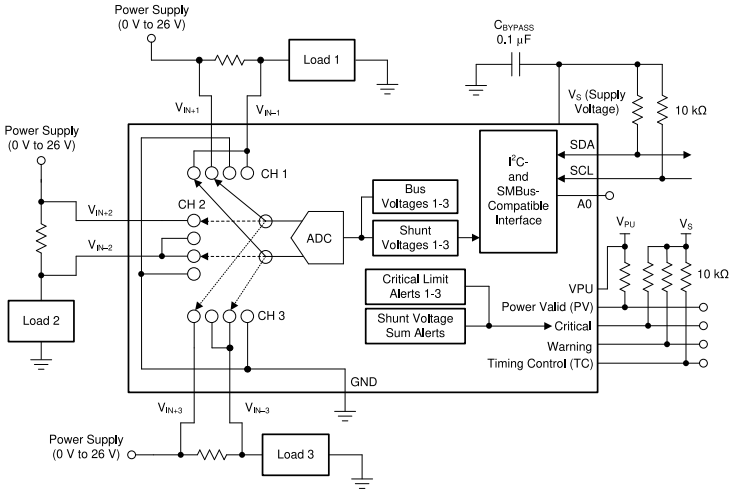


Obrázek 5.4: Schéma logického obvodu pro automatické přepnutí do programovacího módu

[2] https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/max17634bevkit.html

5.6 Senzor Texas Instruments INA3221

Pro měření protékajícího elektrického napětí a proudu na jednotlivých výstupech napájecí jednotky jsem zvolil senzor Texas Instruments INA3221, protože se jedná o rozšířený tříkanálový senzor elektrického proudu a napětí se sériovým rozhraním I²C nebo SMBUS. Elektrický proud je měřen pomocí měření úbytku elektrického napětí na bočníku s odporem 50 mΩ. Senzor obsahuje jeden analogově digitální převodník, na který jsou postupně připojovány jednotlivé vstupy pro měření elektrického napětí a úbytku elektrického napětí na bočníku, který slouží pro měření elektrického proudu. Vyčtené hodnoty z tohoto převodníku jsou ukládány do jednotlivých registrů, které lze vyčíst z mikrokontroléru po sériovém rozhraní.



Obrázek 5.5: Blokové schéma senzoru TI INA3221 převzaté z katalogového listu[31]

Konfigurace senzoru je uložena v registru na adrese 0, ze kterého ji lze přečíst a do kterého ji lze nahrát, jednotlivé konfigurační bity jsou popsány v tabulce 5.2.

Bit	**Název**	**Popis**	**Výchozí hodnota**	**Nastavená hodnota**
15	RST	Reset bit	0	
14	$CH1_{en}$	Povolení 1. kanálu	1 - výstup povolen	
13	$CH2_{en}$	Povolení 2. kanálu		
12	$CH3_{en}$	Povolení 3. kanálu		
9-11	AVG	Režim průměrování	000 - 1 vzorek	111 - 1024 vzorků
6-8	$V_{BUS}CT$	Doba převodu napětí	100 - 1,1 ms	111 - 8,244 ms
3-5	$V_{SH}CT$	Doba převodu úbytku napětí na bočníku	100 - 1,1 ms	111 - 8,244 ms
0-2	MODE	Operační mód	111 - Průběžné měření napětí a úbytku napětí na bočníku	

Tabulka 5.2: Konfigurační registr senzoru TI INA3221

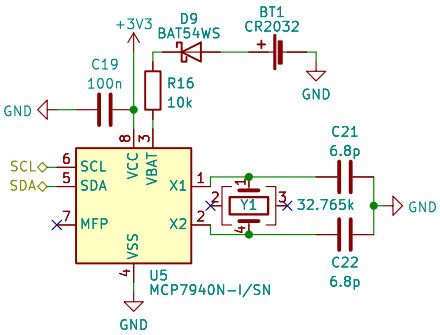
Naměřený úbytek napětí na bočníku na kanále n je uloženo na adrese $2n-1$, naměřené napětí na je pak uloženo na adrese $2n$, jednotlivé bity v daných registrech jsou popsány v tabulce 5.3.

Bit	Název	Popis
15	SIGN	Znaménkový bit, 1 indikuje negativní hodnotu
14	$DATA_{11}$	Nejvýznamnější bit naměřené hodnoty
⋮	⋮	
3	$DATA_0$	Nejméně významný bit bit naměřené hodnoty
0-2		Rezervováno

Tabulka 5.3: Registr pro naměřenou hodnotu napětí senzoru TI INA3221

5.7 Hodiny reálného času Microchip MCP7940N

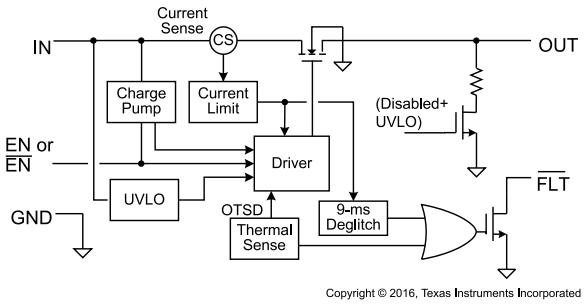
Aby při každém startu napájecí jednotky se nemuselo několik sekund čekat na synchronizaci času pomocí protokolu SNTP, tak napájecí jednotka obsahuje hodiny reálného času. Zvolil jsem Microchip MCP7940N, protože se jedná o levné a rozšířené hodiny reálného času se sériovým rozhraním I²C. Datum a čas je uložen v registru čipu, který je zálohován pomocí CR2032 3 V baterie. Čip ke své práci potřebuje externí 32,768 kHz krystal.



Obrázek 5.6: Část schématu s obvodem hodin reálného času

5.8 USB spínač Texas Instruments TPS2000C

Pro spínání jednotlivých výstupů napájecí jednotky je použit USB spínač Texas Instruments TPS2000C, který také funguje jako nadproudová ochrana. Při překročení jmenovitého elektrického proudu 2 A zvýší svůj vnitřní odpor a tak limituje proud a při vyšším překročení (typicky nad 2,75 A) nebo zkratu na výstupu vypne daný výstup a na svém příslušném výstupním pinu signalizuje chybový stav.



Obrázek 5.7: Blokové schéma USB spínače TI TPS2000C, převzato z katalogového listu[32]

5.9 Napájecí výstupy

Napájecí výstupy jsou realizovány pomocí USB A konektorů, které obsahují 4 vodiče, které jsou popsány v tabulce 5.4. Podle standardu USB může ve verzi 2.0 výstup dodávat 500 mA a ve verzi 3.0 může dodávat 900 mA, tak je nutné pro vyšší proudové odběry indikovat napájenému zařízení, že může odebrat větší proud než mu dovoluje standard USB. Jeden z takových způsobů je definován ve specifikaci USB-IF Battery Charging Specification v1.2[34], která umožňuje z určitých typů portu odebírat až 1,5 A. Specifikace definuje 3 typy USB portů, které jsou popsány v tabulce 5.5. Jelikož napájecí jednotka slouží pouze pro napájení připojených zařízení, tak je použit typ portu DCP, který umožňuje z portu odebírat až 1,5 A. Ale existují i další řešení pro signalizaci maximálního proudového odběru, například Apple Fast Charge, který je popsán v tabulce 5.6.

Jméno	**Číslo pinu na konektoru**	**Barva vodiče**	**Popis**
VBUS	1	červená	Napájecí napětí sběrnice 5 V ± 5 %
D-	2	zelená	Diferenciální pár pro přenos dat
D+	3	bílá	
GND	4	černá	Země sběrnice

Tabulka 5.4: Vodiče používané sběrnicí USB ve verzi 2.0

Zkratka	**Název**	**Popis**
SDP	Standard Downstream Port	Port obsahuje 15 kΩ pull-down rezistory na D+ a D-, maximálně lze odebírat 500 mA
DCP	Dedicated Charging Port	Port slouží pouze pro napájení, D+ a D- jsou navzájem propojeny, maximálně lze odebírat 1,5 A
CDP	Charging Downstream Port	Port slouží pro přenos dat i pro napájení vyšším proudem, obsahuje 15 kΩ pull-down rezistory na D+ a D- a obvod, který je připojen při detekci zařízení, maximálně lze odebírat 1,5 A

Tabulka 5.5: Typy portů dle specifikace Battery Charging Specification v1.2

Proudový limit	**Napětí na D+**	**Napětí na D-**
2,4 A	2,7 V	2,7 V
2,1 A	2 V	2,7 V
1,0 A	2,7 V	2 V
0,5 A	2 V	2 V

Tabulka 5.6: Typy portů dle specifikace Apple Fast Charge

Kapitola 6

Implementace obslužného firmwaru napájecí jednotky

Obslužný firmware napájecí jednotky je napsán v programovacím jazyce C++ a využívá framework ESP-IDF (*ESP IoT Development Framework*).

6.1 ESP IoT Development Framework

ESP-IDF[3][1] je oficiální open-source framework společnosti Espressif pro vývoj aplikací pro jejich bezdrátové moduly a SoC v programovacím jazyce C nebo C++.

Framework je postaven nad open-source operačním systémem reálného času FreeRTOS. A nabízí řadu funkcí pro práci se zabudovanými komponentami a také zaobaluje některé další knihovny (například Mbed TLS[2] pro kryptografii).

Překlad je zajišťován pomocí CMake[3] nebo Ninja[4]. Pro nahrání výsledného sestaveného binárního souboru se používá nástroj `esptool`[5], který je napsán ve skriptovacím jazyce Python.

Pokud bychom nechtěli pracovat přímo s těmito nástroji, tak lze použít nástroj `idf.py`, který je napsán ve skriptovacím jazyce Python a které tyto nástroje zaobaluje.

Repozitář se zdrojovými soubory obsahuje celou řadu ukázek, které jsou určeny jako výchozí bod pro psaní vlastního firmwaru a při používání jednotlivých komponent, které bezdrátový modul poskytuje.

6.2 Operační systém reálného času FreeRTOS

FreeRTOS[6] je nejpoužívanější open-source operační systém reálného času pro vestavěné zařízení založené nad mikrokontroléry. Aktuálně podporuje více než 40 různých architektur mikrokontrolérů. Původně byl vyvinut Richardem Barrym v roce 2003, od roku 2017 se o vývoj stará tým z Amazon Web Services.

Poskytuje mnoho prostředků pro komunikaci a synchronizaci úloh a nástroje pro plánování těchto úloh. Mezi tyto prostředky patří například časovače, fronty, mutexy a semafory.

[1] https://github.com/espressif/esp-idf
[2] https://www.trustedfirmware.org/projects/mbed-tls/
[3] https://cmake.org/
[4] https://ninja-build.org/
[5] https://github.com/espressif/esptool
[6] https://www.freertos.org/

6.3 Funkcionalita obslužného firmwaru

Při každém startu napájecí jednotky se provedou následující kroky:

- Inicializuje se TCP/IP zásobník, smyčka událostí a úložiště certifikátů kořenových certifikačních autorit.
- Inicializuje se NVS úložiště, které obsahuje konfiguraci jednotky a které je popsáno v sekci 6.4. Při prvním spuštěni jsou do tohoto úložiště uloženy výchozí hodnoty pro jednotlivé klíče.
- Inicializuje se ovladač pro I²C sběrnici, pro hodiny reálného času a senzor elektrických veličin. Dále se nastaví jednotlivé GPIO piny, včetně přerušení pro ovládací tlačítka a vstupy pro indikaci poruchy z USB spínačů.
- Inicializují se síťová rozhraní, mDNS (*Multicast DNS*) klient, SNTP klient, HTTP server, který je podrobněji popsán v sekci 6.6, a MQTT klient, který je podrobněji popsán v sekci 6.5.
- Nakonec se v nekonečné smyčce čtou naměřené elektrické veličiny ze senzoru a každých 500 ms se publikují přes MQTT. V případě, že z výstupů je odebírán elektrický proud větší než 4,2 A (což je limitace daná použitým DC-DC snižujícím měničem), tak je vypnut výstup s nejvyšším proudovým odběrem.
- Při stisknutí tlačítka se daný výstup zapne či vypne v závislosti na aktuálním stavu výstupu. V případě vypnutého výstupu se zapne a vice versa.

6.4 Konfigurace napájecí jednotky

Konfigurace napájecí jednotky je uložena v nevolatilním úložišti NVS (*Non-volilate Storege*), které je umístěno ve samostatném oddílu na SPI flash. NVS pracuje s páry klíč-hodnota. Klíče jsou ASCII řetězce s maximální délkou 15 znaků. Aby byly zabráněno kolizi jmen klíčů mezi komponentami, tak pár klíč-hodnota je uložen jmenném prostoru, který by měl být unikátní pro každou komponentu. Jméno jmenného prostoru je ASCII řetězec s maximální délkou 15 znaků. Uložené hodnoty mohou mít jeden z těchto datových typů:

- celé číslo (`uint8_t`, `int8_t`, `uint16_t`, `int16_t`, `uint32_t`, `int32_t`, `uint64_t` a `int64_t`),
- textový řetězec ukončený nulovým bajtem,
- binární data s proměnnou délkou - tzv. blob.

Jednotlivé konfigurační volby uložené v nevolatilním úložišti jsou podrobněji popsány v tabulce 6.1.

Jmenný prostor „`httpCredentials`“ obsahující přihlašovací údaje k REST API		
Klíč	**Datový typ**	**Popis**
`username`	string	Uživatelské jméno
`password`	string	Heslo
Jmenný prostor „`mqtt`“ obsahující konfiguraci MQTT klienta		
`uri`	string	Adresa MQTT brokeru ve formátu (`mqtt(s)://host:port`)
`username`	string	Uživatelské jméno pro MQTT klienta
`password`	string	Heslo pro MQTT klienta
Jmenný prostor „`ntp`“ obsahující konfiguraci SNTP klienta		
`servers`	uint8_t	Počet NTP serverů
`server{N}`	string	Adresa {N}. NTP serveru
`timezone`	string	Časová zóna
Jmenný prostor „`wifi`“ obsahující konfiguraci WiFi		
`authmode`	uint8_t	Zabezpečení AP (hodnota z výčtu `wifi_auth_mode_t`)
`ssid`	string	SSID přístupového bodu, ke kterému se připojujeme
`psk`	string	Předsdílený klíč (uplatňuje se u zabezpečení z rodiny WPA-Personal)

Tabulka 6.1: Konfigurace uložená v NVS

6.5 MQTT

Pro napojení napájecí jednotky do nadřazeného systému jsem zvolil protokol MQTT, protože se jedná o jednoduchý, robustní protokol s komunikačním modelem Producent-Odběratel, který je výhodný pro komunikaci několika jednotek s nadřazeným systémem.

Po připojení k MQTT brokeru je odeslána zpráva „`online`“ do tématu „`sbc_pdu/ID/status`“ a dále je nastavena poslední vůle se zprávou „`offline`“ do stejného tématu. Tímto je zajištěno reportování stavu napájecí jednotky, protože při odpojení od MQTT brokeru se publikuje zpráva, která indikuje, že jednotka je offline.

Každých 500 ms se odesílají naměřené hodnoty z jednotlivých napájecích výstupů a také jejich stav - zda-li jsou zapnuty a zda-li na nich je porucha.

MQTT téma*	**Datový typ**	**Popis**
`/status`	string	Stav napájecí jednotky (`online` nebo `offline`)
`/outputs/{ID}/alert`	int (0,1)	Závada (například zkrat) na výstupu s ID `{ID}`
`/outputs/{ID}/enabled`	int (0,1)	Stav (vypnutí/zapnutí) výstupu s ID `{ID}`
`/outputs/{ID}/current`	float	Protékající el. proud v mA výstupem s ID `{ID}`
`/outputs/{ID}/voltage`	float	Elektrické napětí ve V na výstupu s ID `{ID}`

* všechna MQTT témata mají prefix „`sbc_pdu/ID`“, kde ID je MAC adresa WiFi bez separátorů

Tabulka 6.2: MQTT témata publikovaná napájecí jednotkou

MQTT téma*	Datový typ	Popis
`/outputs/{ID}/enable`	int (0,1)	Vypnutí nebo zapnutí výstupu s ID `{ID}`

* všechna MQTT témata mají prefix „`sbc_pdu/ID`“, kde ID je MAC adresa WiFi bez separátorů

Tabulka 6.3: MQTT témata odebíraná napájecí jednotkou

6.6 HTTP server

Napájecí jednotka obsahuje HTTP server, který slouží pro lokální ovládání napájecích výstupů, zobrazení aktuálně změřených elektrických veličin na jednotlivých výstupech a úpravu nastavení jednotlivých naimplementovaných komponent. Implementace HTTP serveru byla odvozena z ukázky v ESP-IDF repozitáři[7], která je licencovaná pod licencí CC0.

HTTP server obsahuje REST API endpointy, které jsou popsány v tabulce 6.4, a metodu pro získání souborů (například pro frontend) ze SPIFFS. Autentizace uživatele je řešena pomocí Basic autentizačního schématu, jednotka obsahuje jeden uživatelský účet. Ve výchozím nastavení uživatelské jméno je `admin` a heslo `sbc-pdu`. Přihlašovací údaje si uživatel může pomocí REST API endpointu změnit. Po úpravě nastavení je nutné napájecí jednotku restartovat buď pomocí tlačítka nebo pomocí REST API endpointu.

Cesta*	HTTP Metoda	Popis
`/auth`	GET	Zkontroluje přihlašovací údaje
`/auth`	PUT	Upraví přihlašovací údaje
`/hostname`	GET	Vrátí aktuální jméno hostitele jednotky
`/hostname`	PUT	Upraví jméno hostitele jednotky
`/mqtt`	GET	Vrátí aktuální konfiguraci MQTT klienta
`/mqtt`	PUT	Upraví konfiguraci MQTT klienta
`/ntp`	GET	Vrátí aktuální konfiguraci SNTP klienta
`/ntp`	PUT	Upraví konfiguraci SNTP klienta
`/outputs`	GET	Vrátí naměřené veličiny na výstupech a jejich stav
`/outputs/switch`	POST	Zapne/vypne výstup napájecí jednotky
`/system/info`	POST	Vrátí systémové informace
`/system/restart`	POST	Restartuje jednotku
`/wifi`	GET	Vrátí aktuální konfiguraci WiFi
`/wifi`	PUT	Upraví konfiguraci WiFi
`/wifi/scan`	POST	Provede sken WiFi přístupových bodů

* všechny cesty mají prefix „`/api/v1`“

Tabulka 6.4: Implementované REST API endpointy napájecí jednotkou

6.7 Frontend

Jelikož vykreslovat uživatelsky přívětivé prostředí ve firmwaru by bylo velice pracné a výsledné řešení by nebylo moc přehledné, tak jsem se rozhodl, že uživatelské rozhraní bude vykreslováno až v prohlížeči uživatele pomocí JavaScriptu. Konkrétně jsem zvolil skriptovací

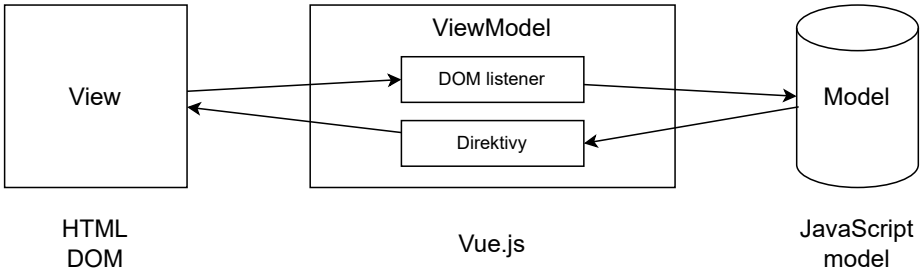
[7] https://github.com/espressif/esp-idf/blob/release/v5.1/examples/protocols/http_server/restful_server/main/rest_server.c

jazyk TypeScript, který je nadstavbou nad JavaScriptem, která přidává typovou kontrolu, při sestavení aplikace se transpiluje do JavaScriptu.

Existuje mnoho technologií pro vytváření uživatelských rozhraní tímto způsobem, nejznámější jsou knihovny a frameworky Angular, React, Svelte a Vue.je. Pro vytvoření frontendu pro napájecí jednotku jsem zvolil posledně jmenovaný framework, protože práce s ním mi byla nejpříjemnější a sestavená aplikace je relativně malá a tak se vejde i do SPIFFS oddílu na SPI flash v použitím bezdrátovém modulu.

6.7.1 Vue.js

Progresivní webový aplikační framework Vue.js[39] původně vyvinul Evan You, když pracoval pro Google s konkurenčním frameworkem AngularJS, protože s některými částmi frameworku se mu špatně pracovalo, nyní framework vyvíjí komunita. Framework je v jádru zaměřen pouze na zobrazovací vrstvu, je inspirován MVVM (*Model-View-ViewModel*) architekturu, pohled a model jsou navzájem propojeny obousměrnou vazbou, kterou zajišťuje ViewModel reprezentovaný frameworkem Vue.js.



Obrázek 6.1: MVVM architektura frameworku Vue.js

Ostatní funkcionalita (např. router nebo správa vnitřního stavu) může být doplněna přidruženými knihovnami. Jako router je použita knihovna vue-router[8], protože se jedná o oficiální implementaci routeru pro Vue.js. Pro správu vnitřního stavu aplikace je použita knihovna pinia[9], protože se jedná o lehkou, moderní, oficiální knihovnu pro správu vnitřního stavu ve Vue.js. Alternativní knihovnou pro správu vnitřního stavu aplikace je Vuex[10], která poskytuje pokročilejší funkcionalitu za cenu složitějšího API a nemá vestavěnou podporu pro TypeScript.

Základní jednotkou aplikace vytvořená v tomto frameworku je komponenta. Komponenta se může skládat ze 3 částí - z šablony, která využívá rozšířené HTML, z kaskádových stylů a z logiky napsané ve skriptovacím jazyce JavaScript nebo TypeScript. Jednotlivé komponenty a HTML značky v aplikaci tvoří stromovou strukturu - tzv. virtuální DOM. Framework od verze 3.0 umožňuje dva přístupy tvorby komponent - použití Options nebo Composition API. Zvolil jsem práci s Composition API, protože výsledný kód je kratší a přehlednější a jedná se o preferovaný způsob ve Vue.js verze 3.

[8] https://router.vuejs.org/
[9] https://pinia.vuejs.org/
[10] https://vuex.vuejs.org/

```
<template>
  <h1>Hello {{ name }}!</h1>
  <label for='name'>Name</label>:
  <input id='name' v-model='name'>
  <h2>Counter: {{ counter }}</h2>
  <button @click='increment()'>
    Increment counter
  </button>
</template>

<script>
export default {
  data: function() {
    return {
      /// Counter
      counter: 0,
      /// Name
      name: 'World',
    };
  },
  methods: {
    increment: function() {
      this.counter++;
    },
  },
};
</script>
```

Výpis 6.1: Ukázka použití Options API

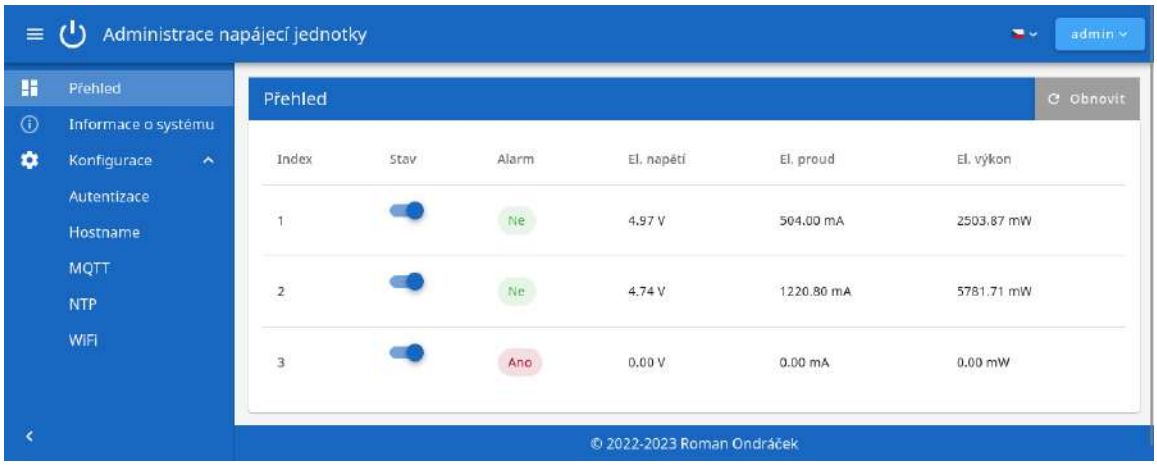
```
<template>
  <h1>Hello {{ name }}!</h1>
  <label for='name'>Name</label>:
  <input id='name' v-model='name'>
  <h2>Counter: {{ counter }}</h2>
  <button @click='increment()'>
    Increment counter
  </button>
</template>

<script setup>
import { ref } from 'vue';

/// Name
const name = ref('World');
/// Counter
const counter = ref(0);

/// Increment counter
function increment() {
  counter.value++;
}
</script>
```

Výpis 6.2: Ukázka použití Composition API



Obrázek 6.2: Přehled napájecích výstupů v administraci napájecí jednotky

Kapitola 7

Implementace softwarového řešení pro centrální správu

Softwarové řešení pro centrální správu napájecích jednotek se skládá ze 3 částí - klientské části (tzv. frontendu), serverové části (tzv. backendu) a odběratele a zpracovatele MQTT zpráv, relační databáze a databáze časových řad. Uživatelské rozhraní je lokalizováno do anglického a českého jazyka.

7.1 Backend

Backend informačního systému je napsán ve skriptovacím jazyce PHP, využívá prostředky poskytované frameworkem Nette[7]. Dále jsou využity některé komponenty z frameworku Symfony, které jsou do aplikace integrovány pomocí balíčků z Contributte[1], například Console pro CLI rozhraní nebo Translation pro překlady. Pro připojení k MQTT brokeru se používá knihovna php-mqtt/client[2].

7.2 Frontend

Frontend informačního systému je napsán ve skriptovacím jazyce TypeScript a používá progresivní webový aplikační framework Vue.js. Pro tvorbu uživatelského prostředí jsem využil knihovnu Vuetify[35], která obsahuje Vue.js komponenty, které vychází z designového jazyku Material Design, který původně vyvivul Google. Frontend komunikuje s REST API backendu pomocí knihovny axios[3]. Pro vykreslování grafů měřených veličin je použita knihovna Apache ECharts[4].

7.3 Odběratel a zpracovatel MQTT zpráv

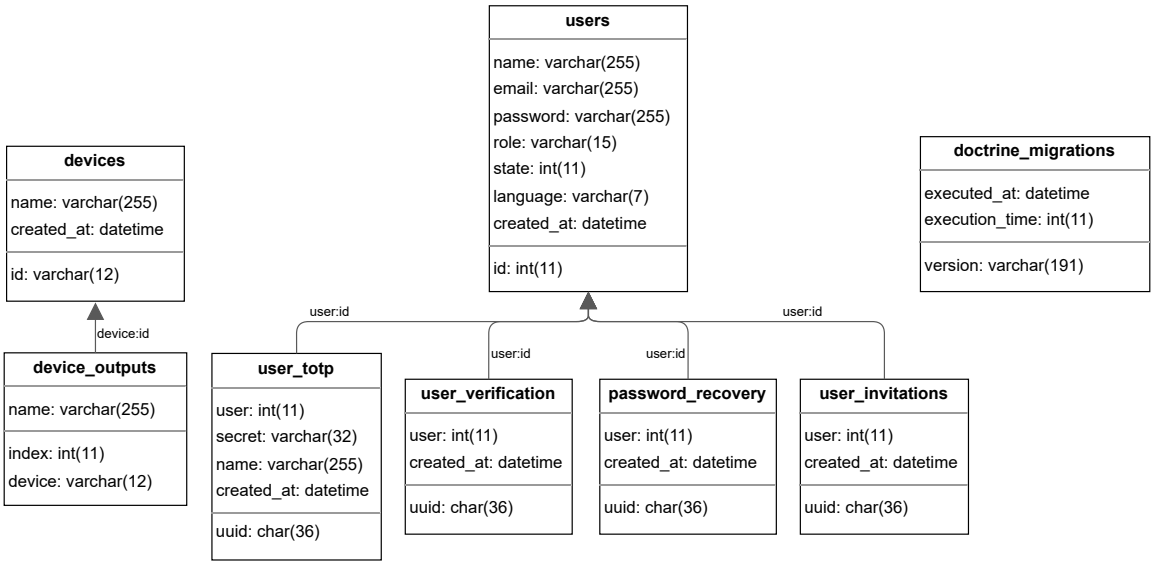
Odběratel a zpracovatel příchozích MQTT zpráv je napsán jako část backendu, která se spouští přes CLI (*Command Line Interface*), tato funkcionalita je poskytována knihovnou symfony/console. Při příchodu nové MQTT zprávy je zkontrolováno, zda-li napájecí jed-

[1] https://contributte.org/
[2] https://github.com/php-mqtt/client
[3] https://axios-http.com/
[4] https://echarts.apache.org/

notka, která zprávu odeslala, je v systému. Pokud není, tak je zpráva ignorována. Jinak je uložena do databáze časových řad.

7.4 Práce s relační databází

Pro práci s relační databází MySQL/MariaDB jsem zvolil framework Doctrine[5], který je určen pro objektově relační mapování jednotlivých databázových entit ve skriptovacím jazyku PHP, a knihovny Nettrine[6] z balíčku Contributte, které tento framework integrují do frameworku Nette. Výhodou tohoto přístupu je větší míra testovatelnosti kódu a lepší validace dat. Framework umožňuje vytvářet tzv. migrace, kdy jednotlivé změny v entitách se promítají do kódu v PHP, který se provádí při aktualizaci aplikace, a tak je jednoduše zajištěna konzistence databázového schématu.



Obrázek 7.1: Diagram schématu MySQL databáze

7.5 Práce s databází časových řad

Databáze časových řad (z anglického Time series Database, TSDB) jsou databázové systémy, které jsou optimalizované pro ukládání hodnot v časových řadách. Časové řady jsou série událostí (například změření hodnoty nebo aktivace alarmu), které jsou sledovány a zpracovávány v čase. Základní vlastností těchto databází je schopnost přijímat velké množství dat, které generují různá zařízení Internetu věcí, a efektivně s nimi pracovat a udržovat historická data pouze po určitou dobu (mohou je mazat nebo do jiné databáze uložit zprůměrované hodnoty). Mezi nejvýznamnější implementace těchto databázových systému patří TimescaleDB a InfluxDB. TimescaleDB je založena nad relační databází PosgreSQL a tak pracuje s dotazovacím jazykem SQL. Tento systém však pracuje s databázovým systémem InfluxDB ve verzi 2.6, který pracuje s dotazovacím jazykem Flux. Zvolil jsem InfluxDB, protože se jedná o nejpoužívanější databázový systém časových řad.

[5] https://www.doctrine-project.org/
[6] https://contributte.org/nettrine/

7.5.1 InfluxDB 2.x

InfluxDB[10] je nejpoužívanější open-source databázový systém časových řad vyvíjený společností InfluxData v programovacím jazyce Go. Využívá se pro sběr, ukládání, zpracování a vizualizaci časových řad. Data jsou v databázovém systému uloženy v následující hierarchii:

- **Kbelík** (*Bucket*) - pojmenované umístění pro uložení časových řad, může obsahovat více měření.
  - **Měření** (*Measurement*) - Logické seskupení dat časové řady. Všechny body v daném měření by měly obsahovat stejné značky. Měření obsahuje více značek a polí.
    - **Značka** (*Tag*) - Páry klíč-hodnota, které jsou určeny pro ukládání metadat pro každý bod (například index napájecího výstupu a ID napájecí jednotky).
    - **Pole** (*Fields*) - Páry klíč-hodnota s hodnotami, které se v čase mění (například protékající proud napájecím výstupem).
    - **Časové razítko** (*Timestamp*) - Časové označení spojené s ukládanými daty. Při ukládání na disk a dotazování jsou všechna data seřazena podle času.

Dalšími důležitými pojmi v InfluxDB jsou:

- **Body** (*Points*) - jednotlivé datové záznamy identifikované měřením, klíči a hodnotami značek, hodnotou pole a časovým razítkem.
- **Řady** (*Series*) - skupiny bodů se stejným měřením, klíči a hodnotami značek.

S databázovým systéme lze interagovat pomocí následujících rozhraní:

- **Webové uživatelské rozhraní** - jedná se o část databázového systému, která uživateli umožňuje vizualizovat uložená data, interagovat s InfluxDB a také ji spravovat.
- **Konzolový program** `influx` - jedná se o samostatnou konzolovou aplikaci, která umožňuje interagovat s InfluxDB a také ji spravovat z příkazové řádky.
- **REST API** - jedná se o část databázového systému, která umožňuje dalším aplikacím interagovat s InfluxDB, autentizace je řešena pomocí API tokenů. Každý token může mít udělená specifická oprávnění pro práci se zdroji, které InfluxDB poskytuje.

Informační systém s InfluxDB komunikuje pomocí REST API přes oficiální open-source knihovnu `influxdb-client-php`[7] pro skriptovací jazyk PHP. Data jsou ukládány do kbelíku `sbc_pdu`, ukládané body obsahují pole `value` s naměřenou hodnotou, značky `device` s ID napájecí jednotky a `output` s indexem napájecího výstupu. Jednotlivá měření včetně datového typu naměřené hodnoty jsou popsány v tabulce 7.1.

Měření	Datový typ	Popis
`alert`	boolean	Porucha na napájecím výstupu
`current`	float	Protékající elektrický proud napájecím výstupem
`enabled`	boolean	Zapnutí napájecího výstupu
`voltage`	float	Elektrické napětí na napájecím výstupu

Tabulka 7.1: Měření ukládaná do InfluxDB

[7] https://github.com/influxdata/influxdb-client-php

7.6 Správa uživatelů

Informační systém umožňuje administrátorovi přidat, pozvat, upravit, smazat a zablokovat či odblokovat jednotlivé uživatele systému. Jednotliví uživatelé si mohou upravit informace o svém uživatelském účtu, změnit si své heslo a požádat o obnovu zapomenutého hesla.

Při pozvání uživatele do systému je na uživatelovu e-mailovou adresu poslána zpráva s odkazem pro vytvoření hesla, odkaz je platný po dobu 7 dní.

Při změně e-mailové adresy uživatele je odeslána zpráva s verifikačním odkazem, který je platný 1 den.

Při zapomenutí hesla uživatel vyplní do patřičného formuláře svou e-mailovou adresu, na kterou je poté odeslána zpráva s odkazem pro nastavení nového hesla, platnost tohoto odkazu je 1 den.

7.7 Autentizace uživatelů

Autentizace uživatelů je řešena pomocí JWT (*JSON Web Token*), který je popsán v podsekci 7.7.1. Token je vygenerován přes REST API endpoint, kterému uživatel předá uživatelské jméno, heslo a kód z TOTP (*Time-Based One-Time Password*) autentikátoru, pokud má uživatel nastavenou dvoufaktorovou autentizaci. TOTP dvoufaktorová autentizace je podrobněji popsána v podsekci 7.7.2.

7.7.1 JSON Web Token

JSON Web Token je standardizovanou metodou pro bezpečnou výměnu informací potřebných pro autorizaci, s volitelným elektronickým podpisem a šifrováním, je definován v RFC 7519[8]. Token se skládá ze 3 částí, které jsou zakódovány pomocí algoritmu Base64 a které jsou navzájem odděleny tečkami:

- **Hlavička** - jedná se o JSON objekt, který obsahuje vlastnost „`typ`“ s hodnotou „`JWT"` pro identifikaci formátu tokenu a vlastnost „`alg`“, která definuje algoritmus použitý pro elektronický podpis.
- **Tělíčko** - jedná se o JSON objekt, který obsahuje informace potřebné pro autentizaci a autorizaci a metadata potřebná pro validaci tokenu (například časové ohraničení platnosti).
- **Elektronický podpis** - obsahuje binární výstup s algoritmu pro elektronický podpis specifikovaného v hlavičce. Do tohoto algoritmu přichází jako vstup hlavička zakódovaná v Base64, oddělovací tečka a tělíčko zakódované v Base64.

[8] https://datatracker.ietf.org/doc/html/rfc7519

Hlavička	Base64	`eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzM4NCJ9`
	JSON	`{"typ":"JWT","alg":"ES384"}`
	Popis	JWT token s ECDSA SHA-384 podpisovým algoritmem
Tělíčko	Base64	`eyJpYXQiOjE2ODMzMjA1MDcsImV4cCI6MTY4MzMyNTkwNywidWlkIjozNH0`
	JSON	`{"iat":1683320507,"exp":1683325907,"uid":34}`
	Popis	Token pro uživatele s ID `34`, platný od 5. 5. 2023 21:01:47 UTC do 22:31:47 UTC

Tabulka 7.2: Popis JWT tokenu používaného informačním systémem

7.7.2 Dvoufaktorová autentizace pomocí TOTP

TOTP je jeden ze dvou nejpoužívanějších způsobů řešení dvoufaktorové autentizace, druhý způsob je pomocí technologie Webauthn, která vyžaduje aby uživatel měl FIDO2 bezpečnostní token (například YubiKey). TOTP je definováno v RFC 6238[9]. Autentikátorem se myslí aplikace, která má v sobě databázi tajných klíčů a generuje TOTP kódy, příkladem takové aplikace je Google Authenticator[10].

TOTP funguje na principu, kdy server a autentikátor se dohodnou na následujících parametrech - tajném klíči (tzv. secret), hashovací funkci (typicky SHA1), periodě (typicky 30 s) a délce výsledného kódu (typicky 6 číslic). Kód je vytvořen pomocí algoritmu, který obsahuje danou hashovací funkci, která má jako vstup tajný klíč a časovou značku zaokrouhlenou dolů podle periody. Proto je důležité, aby server a autentikátor měli sesynchronizovaný čas (například pomocí protokolu NTP).

Parametry autentikátor může získat z QR kódu, který obsahuje textový řetězec s URI `otpauth://totp/NÁZEV?PARAMETRY`. Případně uživatel může do autentikátoru zadat tajný klíč jako textový řetězec zakódovaný v Base32 a nastavit další parametry.

Parametr	**Povinnost**	**Výchozí hodnota**	**Popis**
`issuer`	Silně doporučený		Název vydavatele
`secret`	Povinný		Tajný klíč zakódovaný v Base32
`algorithm`	Volitelný	SHA1	Hashovací algoritmus
`digits`	Volitelný	6	Délka TOTP kódu
`period`	Volitelný	30	Perioda v sekundách

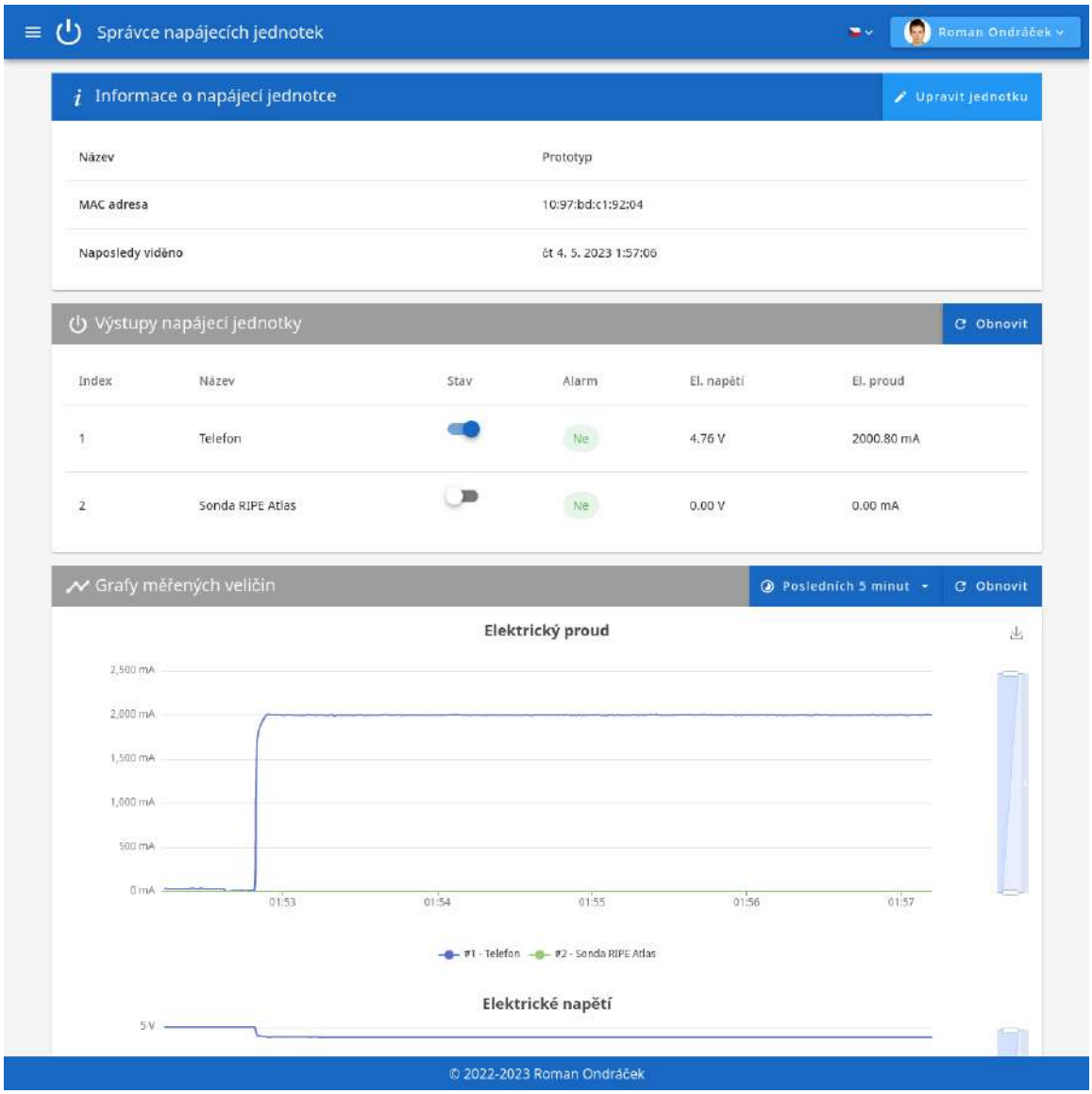
Tabulka 7.3: Parametry `otpauth` URI pro TOTP

7.8 Správa napájecích jednotek

Informační systém umožňuje uživatelům přidávat, upravovat a odebírat jednotlivé napájecí jednotky. Jednotlivé napájecí jednotky jsou v systému identifikovaný pomocí MAC adresy formátované jako řetězec hexadecimálních hodnot bez oddělovačů mezi jednotlivými bajty. V detailu napájecí jednotky jsou zobrazeny informace o jednotce, aktuální stav jednotlivých napájecích výstupů, grafy měřených elektrických veličin a uživatel může ovládat jednotlivé

[9] https://datatracker.ietf.org/doc/html/rfc6238
[10] https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2

výstupy. Při odstranění napájecí jednotky jsou také odstraněna naměřená data z databáze časových řad.



Obrázek 7.2: Detail napájecí jednotky

7.9 HTTP REST API

Pro tvorbu REST API jsem zvolil knihovnu Apitte z balíčku Contributte, protože se jedná o nejpoužívanější knihovnu pro tvorbu REST API, který má integraci do frameworku Nette. Jednotlivá JSON těla požadavků a odpovědí jsou validovány pomocí JSON schémat. Jednotlivé REST API endpointy jsou popsány v tabulce 7.4.

Specifikace REST API je zapsána ve formátu OpenAPI Specification verze 3. Pro zobrazení specifikace REST API byl použit open-source projekt Swagger UI. Specifikaci REST API si lze prohlédnout na `https://sbc-pdu.romanondracek.cz/apiDocs`.

Cesta*	**HTTP Metoda**	**Popis**
`/account`	GET	Získá informace o přihlášeném uživateli
`/account`	PUT	Upraví profil přihlášeného uživatele
`/account/totp`	GET	Vrátí seznam TOTP autentikátorů
`/account/totp`	POST	Zaregistruje nový TOTP autentikátor
`/account/totp/{UUID}`	GET	Získá informace o autentikátoru s UUID `{UUID}`
`/account/totp/{UUID}`	DELETE	Odstraní autentikátor s UUID `{UUID}`
`/account/verification/{UUID}`	POST	Ověří uživatelský účet s UUID ověřovacího požadavku `{UUID}`
`/account/verification/resend`	POST	Odešle e-mailem přihlášenému uživateli nový ověřovací požadavek
`/auth/password/recovery`	POST	Odešle e-mailem uživateli nový požadavek pro obnovu hesla
`/auth/password/reset/{UUID}`	POST	Nastaví nové heslo uživateli s UUID požadavku pro obnovu hesla `{UUID}`
`/auth/password/set/{UUID}`	POST	Nastaví heslo uživateli s UUID pozvánky `{UUID}`
`/auth/sign/in`	POST	Přihlásí uživatele, vrátí informace o uživatelském účtu a vytvoří JWT
`/auth/token/refresh`	POST	Prodlouží platnost JWT tokenu
`/devices`	GET	Vrátí seznam napájecích jednotek
`/devices`	POST	Přidá novou napájecí jednotku
`/devices/{DID}`	GET	Vrátí informace o napájecí jednotce s ID `{DID}`
`/devices/{DID}`	PUT	Upraví informace o napájecí jednotce s ID `{DID}`
`/devices/{DID}`	DELETE	Odstraní napájecí jednotku s ID `{DID}`
`/devices/{DID}/measurements`	GET	Vrátí naměřené hodnoty napájecí jednotkou s ID `{DID}`
`/devices/{DID}/outputs/{OID}`	POST	Zapne/vypne výstup s ID `{OID}` napájecí jednotky s ID `{DID}`
`/openapi`	GET	Vrátí OpenAPI specifikaci
`/openapi/schemas/requests/{N}`	GET	Vrátí JSON schéma požadavku `{N}`
`/openapi/schemas/responses/{N}`	GET	Vrátí JSON schéma odpovědi `{N}`
`/users`	GET	Vrátí seznam uživatelů
`/users`	POST	Vytvoří nebo pozve nového uživatele
`/users/{ID}`	GET	Vrátí informace o uživateli s ID `{ID}`
`/users/{ID}`	PUT	Upraví informace o uživateli s ID `{ID}`
`/users/{ID}`	DELETE	Odstraní uživatele s ID `{ID}`
`/users/{ID}/block`	POST	Zablokuje uživatele s ID `{ID}`
`/users/{ID}/resend`	POST	Znovu odešle ověřovací e-mail nebo e-mail s pozvánkou uživateli s ID `{ID}`
`/users/{ID}/unblock`	POST	Odblokuje uživatele s ID `{ID}`

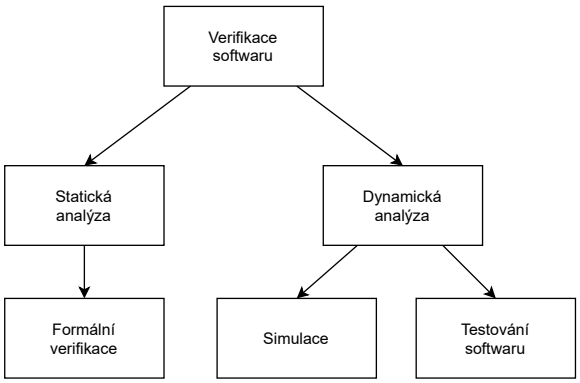
* všechny cesty mají prefix „`/v1`“

Tabulka 7.4: Implementované REST API endpointy pro systém centrální správy

Kapitola 8

Testování

Kontrola kvality software je v dnešní době nedílnou součástí vývoje každého úspěšného softwaru. Kvalitu softwaru lze zajistit pomocí statické a dynamické analýzy. Statická analýza zkoumá vlastnosti softwaru bez jeho provádění. Dynamická analýza ověřuje vlastnosti softwaru na základě jeho provádění. Testování je jedna z metod dynamické analýzy.



Obrázek 8.1: Metody verifikace softwaru

8.1 Základní pojmy v testování softwaru

8.1.1 Specifikace

Specifikace je popis korektního chování, činnosti, reakcí softwaru, včetně popisu vstupních a výstupních dat.

8.1.2 Softwarová chyba

Nejobecnější definici pojmu *softwarová chyba* nalezneme v knize *Software Testing*[23, str. 15], kterou napsal Ron Patton. K softwarové chybě dojde, když platí jedna z následujících podmínek.

- *Software nedělá něco, co by podle specifikace dělat měl.*
- *Software dělá něco, co by podle specifikace dělat neměl.*
- *Software dělá něco, co není uvedeno ve specifikaci.*

- *Software nedělá něco, co není uvedeno ve specifikaci, ale měla by to být ve specifikaci uvedeno.*
- *Software je obtížně srozumitelný, obtížně se s ním pracuje, je pomalý nebo podle názoru testera jej koncový uživatel nebude považovat za správný.*

Z podmínek lze vyčíst, že testování neslouží pouze k ověření toho, že software dělá to, co je zadáno ve specifikaci, ale slouží také k ověření, že software nedělá něco, co ve specifikaci není nebo dokonce, co by mohlo negativně ovlivnit uživatelkou zkušenost s daným software.

8.1.3 Testování

Dle slovníku *Standard Glossary of Terms Used in Software Testing*[11], který je vydáván asociací ISTQB (*International Software Testing Qualifications Board*), se jedná o proces, který se skládá ze všech aktivit životního cyklu, který se týká plánování, přípravy a hodnocení komponenty nebo systému a souvisejících pracovních produktů s cílem určit, zda splňují specifikované požadavky a ukázat, že vyhovují účelu a najít defekty.

8.1.4 Mockování

Mockování znamená nahrazení původní funkce či metody za její testovací imitaci, která neprovádí žádnou akci, jen předstírá chování původní implementace, které potřebujeme během testování.

8.1.5 CI/CD

Pomocí zkratky CI označujeme tzv. *kontinuální integraci* (zkratka pochází z anglického Continuous Integration). Pomocí zkratky CD můžeme označovat tzv. *průběžné nasazení* (zkratka pochází z anglického Continuous Deployment) nebo tzv. *průběžnou dodávku* (zkratka pochází z anglického Continuous Delivery).

Průběžná integrace

Průběžná integrace je první krok automatizace používané při vývoji softwaru. Pokud na daném softwarovém projektu pracuje více lidí, tak umožňuje zefektivnit vývoj daného softwarového projektu. Díky automatizovanému spouštění testů (nejčastěji jednotkových a integračních) se na případné chyby přichází již během vývoje, což výrazně zlevní opravu takovéto chyby.

Průběžné nasazení

Průběžné nasazení je posledním krokem automatizace používané při vývoji softwaru. Pomocí tohoto kroku je zautomatizováno nasazení softwaru do produkce.

8.2 Stupně testování

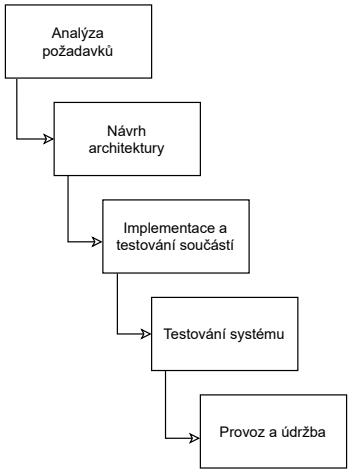
Stupně testování souvisí s jednotlivými etapami životního cyklu vývoje softwaru. Životní cyklus vývoje softwaru je postup vývoje softwaru od počátečního nápadu, specifikaci produktu, implementaci až po jeho nasazení a údržbu. Těchto postupů je několik a definují různé modely životního cyklu vývoje softwaru. Jeden ze základních modelů je tzv. *vodopádový model*.

8.2.1 Vodopádový model

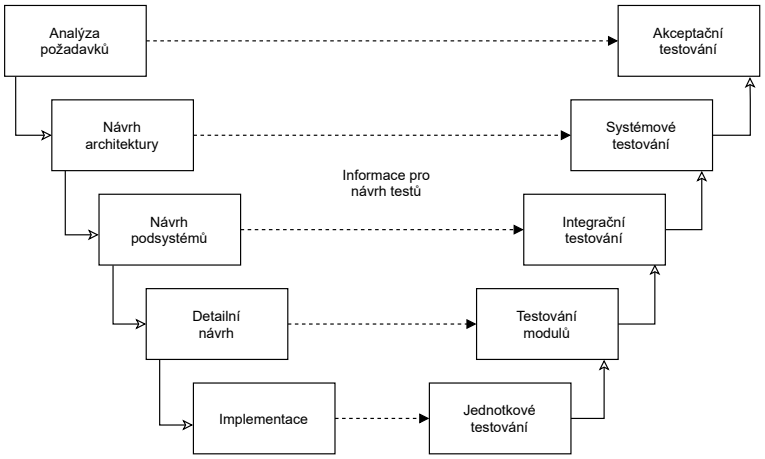
Vodopádový model je základní model životního cyklu softwaru, ve kterém se na vývoj nahlíží jako na svažující se tok jednotlivými fázemi - analýzou požadavků, návrhem, implementací, testováním a provozem a údržbou. Následující etapa začíná až po ukončení předcházející. Schéma vodopádového modelu je ukázáno na diagramu 8.2.

8.2.2 V-model

V-model je modifikací vodopádového modelu, která jej rozšiřuje o fáze testování, které náleží odpovídající etapě vývojového cyklu. V praxi to znamená, že testování se provádí během vývojového cyklu na různých úrovních a návrh testů probíhá během celého vývojového cyklu, ačkoliv software bude spustitelný až v implementační etapě. Schéma V-modelu je ukázáno na diagramu 8.3.



Obrázek 8.2: Schéma vodopádového modelu



Obrázek 8.3: Schéma V-modelu

8.2.3 Jednotkové testování

U testy řízeného vývoje se jednotkové testy píší zásadně před implementací, aby bylo otestováno, že testy správně selžou. Testují se tzv. *jednotky*, což jsou fragmenty kódu popisující chování software a související datové struktury zaobalené v dále nedělitelném logickém celku. Příkladem takového nedělitelného logického celku jsou funkce v procedurálních jazycích, jednoduché třídy nebo složitější metody v objektově orientovaných jazycích. Účelem je zmenšení rozsahu testování na vyšších úrovních. Dokáže odhalit pouze chyby na nejnižší úrovni (například špatnou návratovou hodnotu, vedlejší účinky). Testeři jsou téměř vždy samotní programátoři jednotek. Předpokladem je, že funkce a datové struktury jsou navrženy pro jednotkové testování.

8.2.4 Integrační testování

Účelem je ověření rozhraní modulů v podsystému a jejich vzájemnou komunikaci. Dokáže odhalit chyby v rozhraní a předpokládaných stavech systému. Testeři jsou většinou součástí vývojového týmu. Předpokladem je, že jednotlivé moduly pracují správně.

8.2.5 Systémové testování

Ověřuje, zda-li software jako celek splňuje specifikaci požadavků. Dokáže odhalit chyby v návrhu a specifikaci. Testeři jsou obvykle oddělená skupina od programátorů. Předpokladem je, že jednotlivé části systému pracují samostatně.

8.2.6 Akceptační testování

Ověřuje, zda-li software splňuje požadavky zákazníka. Dokáže odhalit, že software neplní svůj primární účel. Testeři musí mít znalosti dané domény. Předpokladem jsou případy užití domluvené se zákazníkem.

8.2.7 End-to-end testování

Ověřuje, zda-li průchod softwarem souhlasí s návrhem pomocí průchodu procesů od jejich počátku až do konce za podmínek blízkých podmínkám produkčním.

8.3 Použité nástroje pro testování

8.3.1 Nette Tester

Nette Tester[8] je minimalistický open source testovací framework pro jazyk PHP, který vznikl pro testování frameworku *Nette*. Velká výhoda je, že každý test je PHP skript, který lze samostatně spustit, což umožňuje jednoduché krokování ve vývojovém prostředí. Převážně se používá pro jednotkové a integrační testování. Nabízí generování přehledů pokrytí kódu testy a jednoduchou integraci do CI systémů. Pro mockování závislostí je nutné použít externí knihovnu např. Mockery.

8.3.2 Postman

Postman[26] je primárně platforma pro spolupráci při vývoji API, která také o programové vybavení pro testování API. Základní funkcionalita (API klient, testování API[25]) je poskytována zdarma. Postman umožňuje vytvářet testovací scénáře a kolekce testovacích jednotek pro testování REST API, které lze všechny naráz spustit nebo je spouštět manuálně po jednom.

Pro průběžnou integraci se používá open source nástroj Newman, což je konzolová varianta tohoto nástroje, která pouze spouští daný testovací scénář a umožňuje pomocí JSON souboru předat parametry pro testy.

8.3.3 GitLab CI

GitLab CI[6] je nativní platforma pro kontinuální integraci a nasazení pro kolaborační platformu GitLab. O provádění jednotlivých fází pipeline se stará GitLab CI runner, který zpravidla pro testovací prostředí používá Docker. Konfigurace pipeline se nachází ve souboru `.gitlab-ci.yml`, který je ve formátu YAML a obsahuje definici jednotlivých fází. Konfigurace fáze pipeline obsahuje definici Docker obrazu a jednotlivých příkazů, které se mají vykonat.

8.4 Testování napájecí jednotky

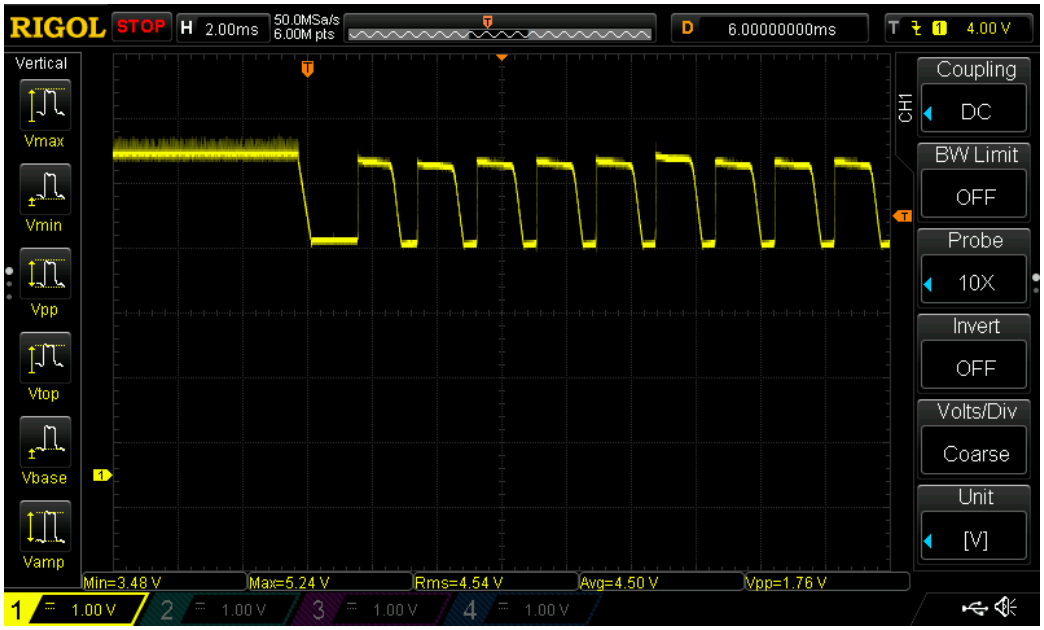
Testování napájecí jednotky se skládalo z několika kroků: oživení hardwaru, nahrání obslužného firmwaru, manuálního ověření funkcionality, která se nedala zautomatizovat, spuštění automatických testů REST API a ověření přesnosti měřených elektrických veličin.

8.4.1 Oživení hardwaru napájecí jednotku

Během vývoje hardwaru napájecí jednotky byly navrženy a sestaveny tři revize desek plošných spojů, které byly vyrobeny firmou Gatema PCB a.s.[1] Každá z desek plošných spojů byla z velké části osázena komponentami pomocí horkovzdušné pájecí stanice a pájecí pasty. Během oživování jsem všechny napájecí větve a důležité signály sledoval pomocí osciloskopu Rigol DS1054Z. Obslužný firmware byl nahráván pomocí ladícího nástroje Espressif Systems ESP-Prog[2].

První prototyp sloužil na ověření konceptu a obsahoval 2 napájecí výstupy, nefunkční hodiny reálného času a USB-UART převodník, chyběla tlačítka pro přímé ovládání uživatelem a signalizační LED.

U druhé revize prototypu byl opraven obvod hodin reálného času, přibyl Ethernetový řadič Microchip LAN8720A, ovládací tlačítka a signalizační LED. U této revize jsem narazil na limit v počtu GPIO, které poskytuje ESP32 a tak část funkcionality (tlačítka a třetí výstup nejsou funkční, když je pomocí propojky nastaveno, že je povolen JTAG pro ladění). U LED pro signalizaci závady na výstupu došlo k invertování logiky. Ethernetový řadič nefungoval a přetěžoval 3,3 V LDO lineární regulátor LM1117. Snižující DC-DC měnič v této revizi byl z neznámého důvodu nestabilní při odběru vyšším než 1,7 A, na výstupu se objevilo obdélníkové zvlnění, které je vidět na snímku obrazovky osciloskopu 8.4.

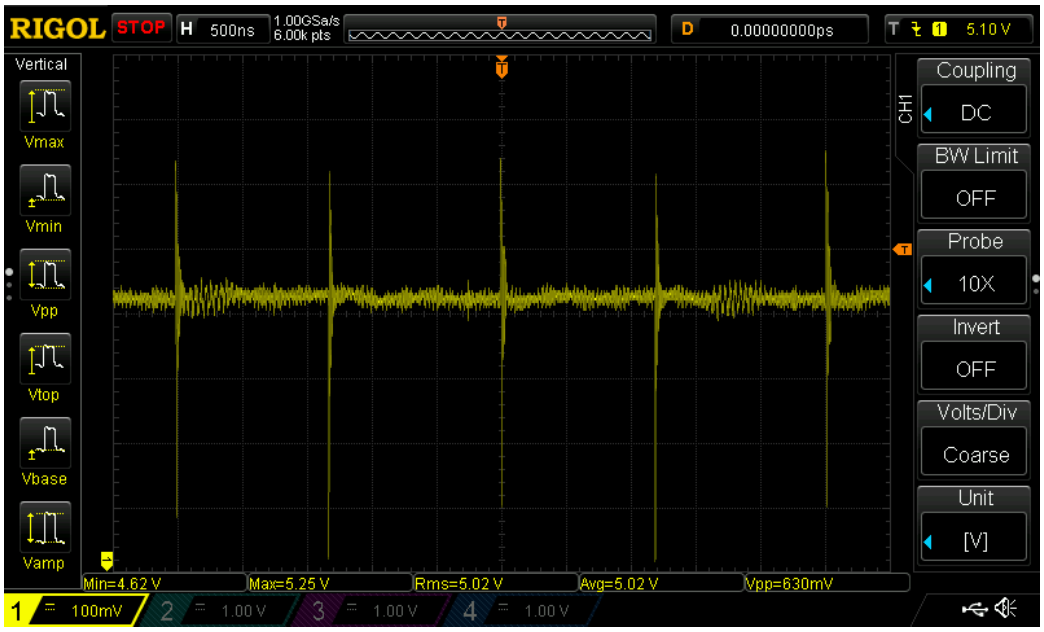


Obrázek 8.4: Nestabilní výstup DC-DC snižujícího měniče při proudovém odběru 1,7 A u 2. revize

[1] https://www.gatemapcb.cz/

[2] https://espressif-docs.readthedocs-hosted.com/projects/espressif-esp-iot-solution/en/latest/hw-reference/ESP-Prog_guide.html

U třetí revize prototypu byl odstraněn Ethernetový řadič a uvolněné GPIO piny byly použity pro část funkcionality, která v předchozím prototypu sdílela piny s JTAGem pro ladění. Dále byl opraven USB-UART převodník a DC-DC snižující měnič je nyní stabilní i při plném proudovém vytížení, což je ilustrováno snímkem obrazovky osciloskopu 8.5.



Obrázek 8.5: Zvlnění na výstupu DC-DC snižujícího měniče při proudovém odběru 4,1 A

8.4.2 Testování obslužného firmwaru

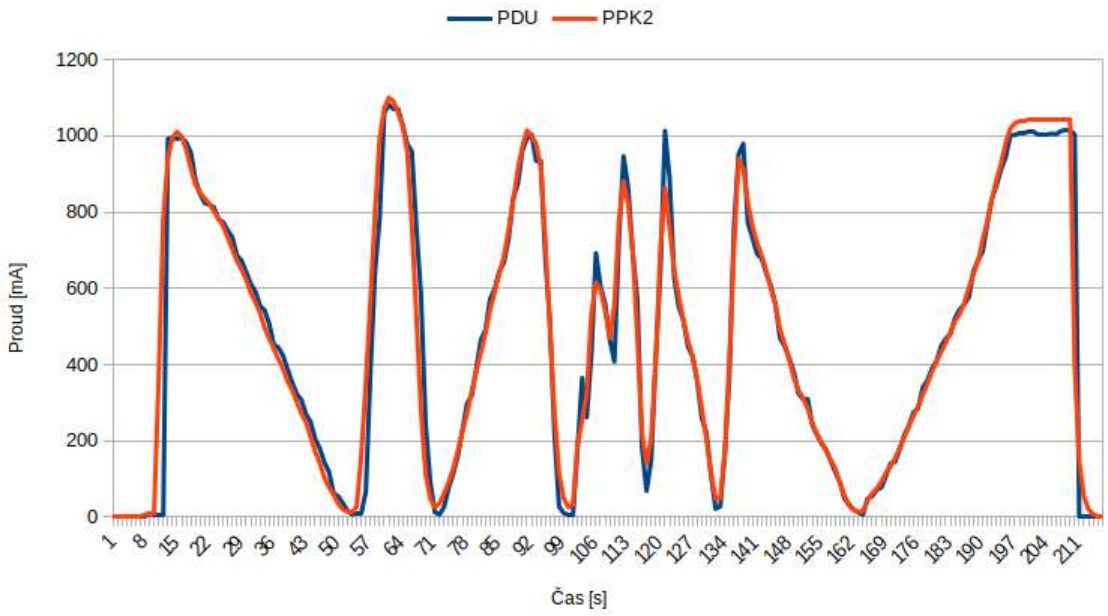
V GitLab repozitáři je nastavena základní průběžná integrace v GitLab CI, která při každém nahrání kódu do repozitáře ověří, že lze firmware přeložit.

Pro testování REST API byl použit nástroj Postman, který je popsán v podsekci 8.3.2. Díky testování jsem objevil několik chyb, které jsem následně opravil.

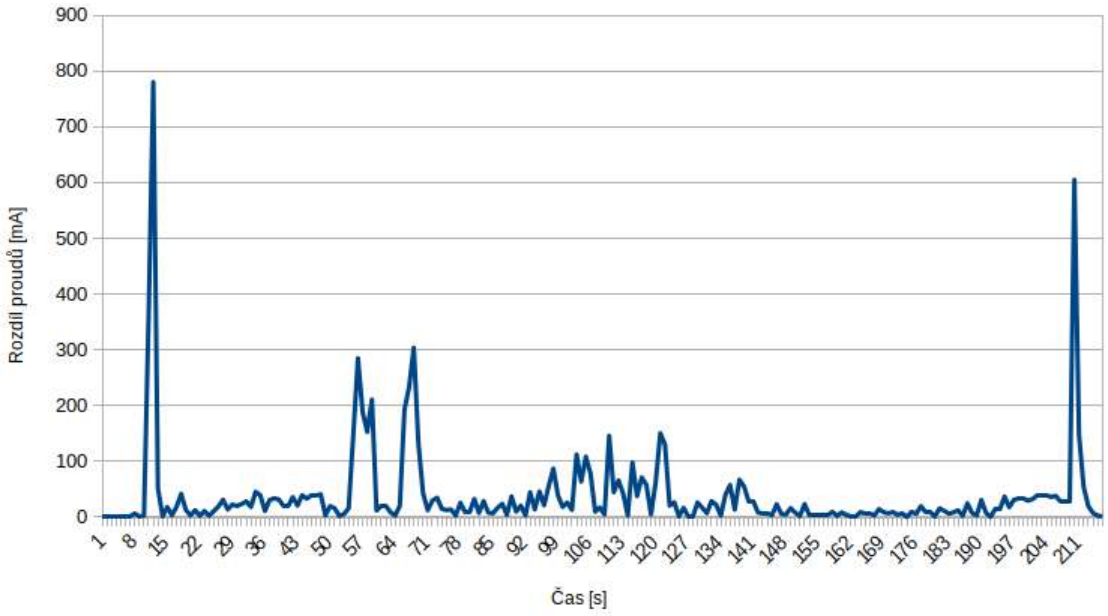
8.4.3 Ověření přesnosti měření elektrických veličin

Přesnost měření elektrických veličin byla zjištěna pomocí analyzátoru napájení Nordic Semiconductor Power Profiler Kit II, který je podrobněji popsán v podsekci 2.2.3. Nejdříve jsem otestoval přesnost měření pomocí elektronické zátěže, kdy jsem měnil odebíraný proud zátěží. Naměřené hodnoty byly odečítány jednou za sekundu.

Ve grafu 8.6 můžeme vidět porovnání naměřených hodnot a ve grafu 8.7 rozdíl naměřených hodnot. Při velkých změnách protékajícího proudu můžeme vidět relativně velké rozdíly. Tyto rozdíly jsou dány tím, že použitý senzor v napájecí jednotce měří nepřetržitě a průměruje naměřené hodnoty. Dále tyto rozdíly mohly vzniknout tím, že proud nebyl snímán přesně v jeden okamžik, ale byla zde nějaká prodleva v řádu desítek milisekund. Při pomalých změnách byly rozdíly mezi naměřenými hodnotami v řádu jednotek miliampér.



Obrázek 8.6: Graf procházejícího proudu napájecí jednotkou a analyzátorem napájení



Obrázek 8.7: Graf rozdílu naměřených hodnot protékajících proudů napájecí jednotkou a analyzátorem napájení

8.5 Testování informačního systému pro centrální správu

Obě části informačního systému mají ve svých GitLab repozitářích nastavenou průběžnou integraci pomocí nástroje GitLab CI, která při každém nahrání kódu do repozitáře nejprve

sestaví Docker image pro testování a pak spustí jednotlivé kroky potřebné pro kontrolu kvality kódu u daných repozitářů.

8.5.1 Testování backendu

Backend je částečně pokryt jednotkovými testy, pro které je použit nástroj Nette Tester a testy REST API, pro které je použit nástroj Postman.

V GitLab repozitáři[3] je nastavena průběžná integrace, která spouští jednotkové testy a statickou analýzu pomocí nástrojů PHPStan[4], PHP_CodeSniffer[5] a Nette Code Checker[6].

8.5.2 Testování frontendu

Aktuálně frontend není pokryt žádnými automatickými testy, ale do budoucna bych chtěl přidat jednotkové a integrační testy, pro které bych použil nástroj Vitest[7], a End-to-End testy, pro které bych použil nástroj Cypress[8]. Proto testování bylo manuální.

V GitLab repozitáři[9] je nastavena průběžná integrace, která spouští statickou analýzu pomocí nástrojů ESLint[10] (pro Vue.js komponenty, TypeScript skripty), Stylelint[11] (pro kaskádové styly) a vue-i18n-extract[12] (pro překlady).

8.6 Možnosti dalšího rozvoje projektu

Jednotlivé části systému je nadále možné rozšiřovat v několika různých směrech. Aktuálně se jedná o minimální použitelný produkt. Pokud bychom chtěli systém komerčně prodávat, tak by bylo nutné některé věci navrhnout lépe a doimplementovat je.

Největší slabina systému je v nutnosti zadání konfigurace do zdrojového souboru obslužného firmwaru. Lepší řešení by bylo vytvoření WiFi přístupového bodu s tzv. captive portálem na napájecí jednotce, když jednotka není nakonfigurována. Původně to bylo v plánu, ale bohužel jsem to nestihl. Dále jednotka by mohla obsahovat Ethernetový řadič pro připojení k drátové počítačové síti. Musela by se přizpůsobit deska plošných spojů pro montáž do komerčně dostupné krabičky.

Poté by bylo potřeba vyřešit jednoduší připojení jednotek do systému pro centrální správu. To by se dalo vyřešit pomocí speciálních autentizačních tokenů, pomocí kterých by si napájecí jednotka přes HTTPS stáhla ze systému konfiguraci připojení k MQTT brokeru a přidala se do systému.

Pak by se do systému musely také přidat oprávnění k jednotlivým napájecím jednotkám, aby si je mohli zobrazit a s nimi manipulovat pouze specifikovaní uživatelé.

Také by bylo dobré přidat API klíče pro uživatelské aplikace, pomocí kterých by se autentizovali pro práci REST API poskytovaném systémem.

[3] https://gitlab.com/sbc-pdu/central-management/backend
[4] https://phpstan.org/
[5] https://github.com/squizlabs/PHP_CodeSniffer
[6] https://doc.nette.org/en/code-checker
[7] https://vitest.dev
[8] https://www.cypress.io/
[9] https://gitlab.com/sbc-pdu/central-management/frontend
[10] https://eslint.org/
[11] https://stylelint.io/
[12] https://github.com/Spittal/vue-i18n-extract

Kapitola 9

Závěr

Cílem této práce bylo vytvořit napájecí jednotku pro jednodeskové počítače s funkcí měření spotřeby. Tento cíl byl splněn realizací funkčního vzorku. V rámci této práce se mi dále podařilo naimplementovat informační systém pro centrální správu napájecích jednotek s uživatelsky přívětivým webovým rozhraním a REST API pro integraci do dalších systémů. Komunikace napájecí jednotky s informačním systémem probíhá pomocí protokolu MQTT.

V práci je shrnuto pomocí jakých komerčně dostupných alternativ lze měřit a spínat napájení jednodeskových počítačů. Dále byl vypracován přehled běžných komunikačních protokolů pro zařízení Internetu věcí a protokolů, které se běžně používají pro správu běžných napájecích jednotek, které pracují se síťovým napětím.

Na základě získaných poznatků a provedené rešerše stávajících řešení byla navrhnuta a sestavena napájecí jednotka založená na bezdrátovém modulu Espressif ESP32-WROOM-32D. Poté byl naimplementován obslužný firmware napájecí jednotky v jazyce C++ s využitím frameworku ESP-IDF a klientská webová aplikace pro správu napájecí jednotky v jazyce TypeScript s využitím frameworku Vue.js.

Jednotlivé jednotky je poté možné ovládat z informačního systému pro centrální správu. Tato aplikace umožňuje spravovat více napájecích jednotek, zobrazit si historická naměřená data, ovládat jednotlivé výstupy napájecí jednotky. Informační systém se skládá ze dvou částí. První je serverová část implementující REST API server v programovacím jazyce PHP s využitím frameworku Nette. A druhá je klientská část implementující webové uživatelské rozhraní v jazyce TypeScript s využitím frameworku Vue.js.

V závěru této práce je krátký úvod do testování a poté je popsán proces testování celého řešení. Tímto testováním byla prokázána funkčnost řešení a díky němu byly odstraněny některé chyby vzniklé při vývoji. A nakonec jsou popsány možnosti dalšího vývoje projektu.

Celé řešení je dostupné jako open-source na GitLabu[1], softwarové komponenty jsou pod licencí Apache License 2.0. Během vývoje informačního systému pro centrální správu se mi podařilo objevit pár chyb v použitých knihovnách, které jsem opravil a mé opravy jsem poslal správcům daných open-source projektů, které je poté začlenili. Dále jsem vytvořil několik nových schématických značek a modely pouzder pro KiCAD, které jsem poslal k začlenění do projektu.

[1] https://gitlab.com/SBC-PDU

Literatura

[1] Digi-Key Electronics. *An Introduction to Power-over-Ethernet* [online]. Květen 2020 [cit. 2023-03-10]. Dostupné z: https://www.digikey.cz/en/articles/an-introduction-to-power-over-ethernet2.

[2] Eaton. *Rack PDU | Power distribution units for server racks* [online]. 2022 [cit. 2022-01-18]. Dostupné z: https://www.eaton.com/us/en-us/products/backup-power-ups-surge-it-power-distribution/power-distribution-for-it-equipment/power-distribution-units-for-server-racks.html.

[3] Espressif Systems. *ESP-IDF Programming Guide* [online]. 2022 [cit. 2022-11-20]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html.

[4] Espressif Systems. *ESP32 Series Datasheet v4.2* [online]. Leden 2023 [cit. 2023-02-20]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[5] Fielding, R. T. *REST: Architectural Styles and the Design of Network-based Software Architectures*. 2000. [cit. 2023-03-08]. Doctoral dissertation. University of California, Irvine. Dostupné z: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

[6] GitLab B.V.. *GitLab CI/CD | GitLab* [online]. GitLab B.V., duben 2023 [cit. 2023-05-01]. Dostupné z: https://docs.gitlab.com/ce/ci/.

[7] Grudl, D. *Nette – Pohodlný a bezpečný vývoj webových aplikací v PHP* [online]. 2022 [cit. 2022-11-05]. Dostupné z: https://nette.org/cs/.

[8] Grudl, D. *Nette Tester – pohodové testování v PHP* [online]. 2023 [cit. 2023-05-01]. Dostupné z: https://tester.nette.org/.

[9] HiveMQ GmbH. *MQTT & MQTT 5 Essentials* [online]. 1. vyd. 2020 [cit. 2023-04-22]. ISBN 978-3-00-067913-1. Dostupné z: https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf.

[10] InfluxData. *Get started with InfluxDB | InfluxDB OSS 2.7 Documentation* [online]. 2023 [cit. 2023-05-06]. Dostupné z: https://docs.influxdata.com/influxdb/v2.7/get-started/.

[11] International Software Testing Qualifications Board. *Standard Glossary of Terms Used in Software Testing* [online]. International Software Testing Qualifications Board, březen 2016 [cit. 2023-01-16]. Dostupné z: https://glossary.istqb.org/en.

[12] IQRF Alliance. *IQRF Standard Binary Output* [online]. Prosinec 2017 [cit. 2023-04-24]. Dostupné z: https://www.iqrfalliance.org/techdoc_files/IQRF-StandardBinaryOutput_V004.pdf.

[13] IQRF Alliance. *IQRF Standard Sensor* [online]. únor 2023 [cit. 2023-04-24]. Dostupné z: https://www.iqrfalliance.org/techdoc_files/IQRF-StandardSensor_V015.pdf.

[14] Kurose, J. *Computer Networking: A Top-Down Approach*. 7. vyd. 2017 [cit. 2023-04-20]. ISBN 978-0-13-359414-0.

[15] Maxim Integrated. *MAX17634A/MAX17634B/ MAX17634C — 4.5V to 36V, 4.25A, High-Efficiency, Synchronous Step-Down DC-DC Converter* [online]. Září 2019 [cit. 2023-02-24]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/max17634a-max17634c.pdf.

[16] Maxim Integrated. *MAX17634BEVKIT# Evaluation Kit — Evaluates: MAX17634B 5V Output-Voltage Application* [online]. Září 2019 [cit. 2023-02-24]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX17634BEVKIT.pdf.

[17] Microchip Technology Inc.. *LAN8720A/LAN8720AI Data Sheet* [online]. červenec 2016 [cit. 2023-04-20]. Dostupné z: https://ww1.microchip.com/downloads/en/devicedoc/00002165b.pdf.

[18] Microchip Technology Inc.. *MCP7940N Data Sheet* [online]. Duben 2018 [cit. 2023-02-24]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP7940N-Battery-Backed-I2C-RTCC-with-SRAM-20005010G.pdf.

[19] MICRORISC s.r.o.. *IQRF DPA v4.30 Framework Technical Guide* [online]. Březen 2023 [cit. 2023-04-24]. Dostupné z: https://static.iqrf.org/Tech_Guide_DPA-Framework-430_230307.pdf.

[20] NETIO Products a.s.. *PDU (Power Distribution Unit)* [online]. 2022 [cit. 2022-01-18]. Dostupné z: https://www.netio-products.com/cs/slovnik/pdu-power-distribution-unit.

[21] NETIO Products a.s.. *PowerPDU 4C* [online]. 2022 [cit. 2022-01-18]. Dostupné z: https://www.netio-products.com/cs/zarizeni/powerpdu-4.

[22] Nordic Semiconductor. *Power Profiler Kit II — nordicsemi.com* [online]. 2023 [cit. 2023-04-13]. Dostupné z: https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2.

[23] Patton, R. *Softwate Testing*. 1. vyd. Sams Publishing, 2001. ISBN 0-672-31983-7.

[24] Plugable Technologies. *USB Power Delivery and Charging Battery-Powered Devices — Plugable Knowledge Base* [online]. Listopad 2020 [cit. 2022-12-03]. Dostupné z: https://kb.plugable.com/usb-hubs-cables-switches/usb-charging-battery-powered-devices.

[25] Postman, Inc.. *Automated API testing* [online]. Postman, Inc., 2023 [cit. 2023-05-01]. Dostupné z: https://www.postman.com/automated-testing/.

[26] Postman, Inc.. *Postman | The Collaboration Platform for API Development* [online]. Postman, Inc., 2023 [cit. 2023-05-01]. Dostupné z: https://www.postman.com/.

[27] Seeed Technology. *UART vs I2C vs SPI – Communication Protocols and Uses* [online]. Září 2019 [cit. 2023-03-14]. Dostupné z: https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/.

[28] Seeed Technology. *What are LoRa and LoRaWAN?* [online]. Prosinec 2021 [cit. 2023-05-04]. Dostupné z: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/.

[29] Sigrok contributors. *RDTech UM series* [online]. 2023 [cit. 2023-01-12]. Dostupné z: https://sigrok.org/wiki/RDTech_UM_series.

[30] Silicon Laboratories, Inc.. *USBXpress™ Family CP2102N Data Sheet* [online]. Březen 2016 [cit. 2023-02-20]. Dostupné z: https://www.silabs.com/documents/public/data-sheets/cp2102n-datasheet.pdf.

[31] Texas Instruments. *INA3221 Triple-Channel, High-Side Measurement, Shunt and Bus Voltage Monitor with I2C- and SMBUS-Compatible Interface* [online]. Březen 2016 [cit. 2023-02-16]. Dostupné z: https://www.ti.com/lit/ds/symlink/ina3221.pdf.

[32] Texas Instruments. *TPS20xxC and TPS20xxC-2 Current Limited, Power-Distribution Switches* [online]. Duben 2016 [cit. 2023-02-22]. Dostupné z: https://www.ti.com/lit/ds/symlink/tps2000c.pdf.

[33] Ubiquiti Inc.. *Switch Lite 8 PoE* [online]. 2023 [cit. 2023-04-12]. Dostupné z: https://eu.store.ui.com/collections/unifi-network-routing-switching/products/unifi-switch-lite-8-poe.

[34] USB Implementers Forum Inc.. *USB Battery Charging 1.2 Compliance Plan Revision 1.0* [online]. říjen 2011 [cit. 2022-12-02]. Dostupné z: https://usb.org/sites/default/files/USB_Battery_Charging_1.2.pdf.

[35] Vuetify. *Vuetify — A Vue Component Framework* [online]. 2022 [cit. 2022-11-24]. Dostupné z: https://vuetifyjs.com/en/.

[36] Wikipedia. *Analog-to-digital converter — Wikipedia, The Free Encyclopedia* [online]. 2023 [cit. 2023-05-02]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Analog-to-digital%20converter&oldid=1152147100.

[37] Wikipedia. *IEEE 802.11 — Wikipedia, The Free Encyclopedia* [online]. 2023 [cit. 2023-04-28]. Dostupné z: https://en.wikipedia.org/w/index.php?title=IEEE_802.11&oldid=1150555088.

[38] Wikipedia. *Representational State Transfer — Wikipedie,* [online]. 2023 [cit. 2023-05-03]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Representational_State_Transfer&oldid=22353842.

[39] You, E. *Vue.js - The Progressive JavaScript Framework | Vue.js* [online]. 2022 [cit. 2022-11-15]. Dostupné z: https://vuejs.org/.

Přílohy

Seznam příloh

A Obsah přiloženého paměťového média **71**

B Schéma napájecí jednotky **72**

C Návrh plošného spoje **77**

D Soupiska součástek **79**
D.1 Cena výroby jedné napájecí jednotky 81

E Fotografie napájecí jednotky **82**

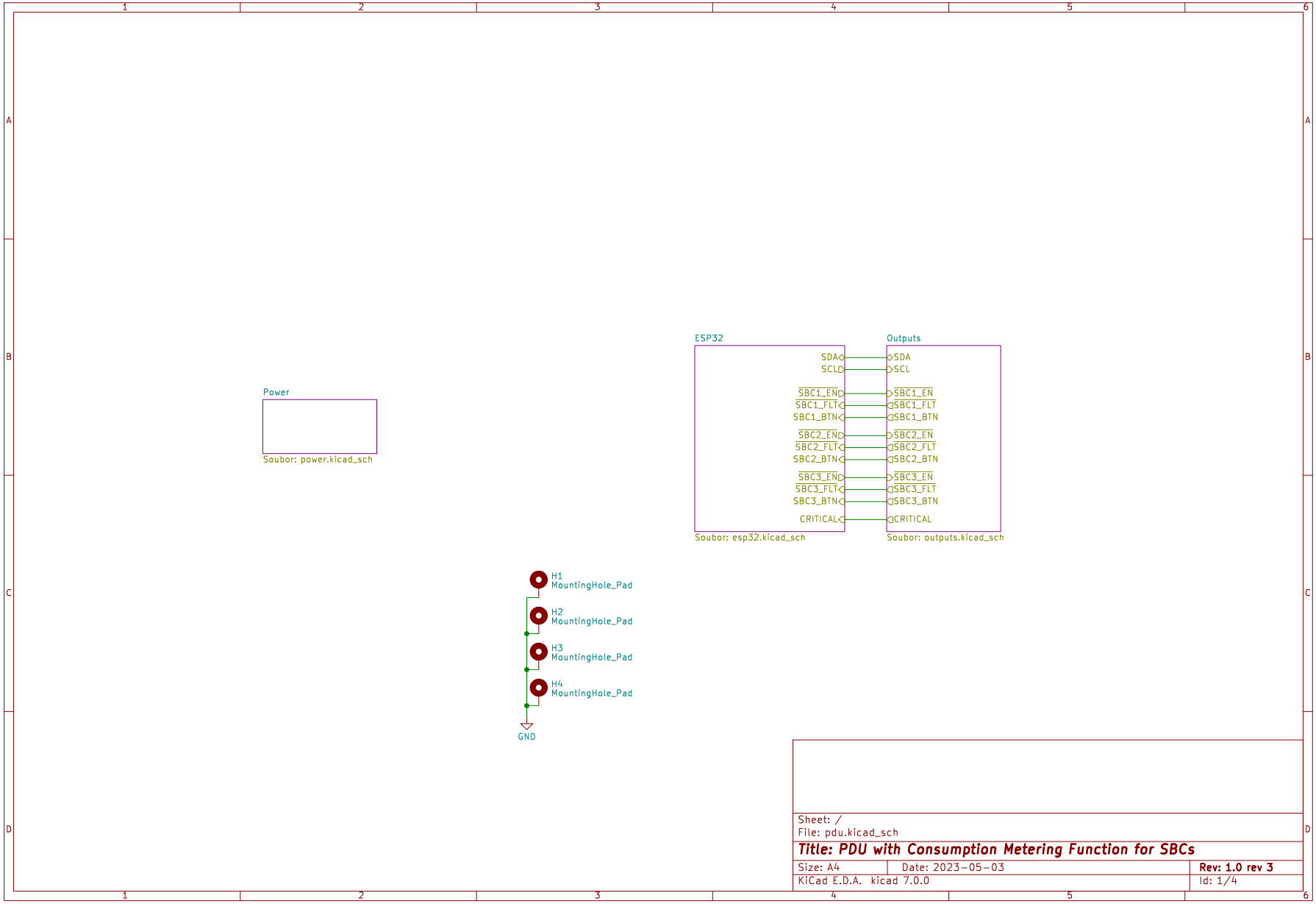
Příloha A

Obsah přiloženého paměťového média

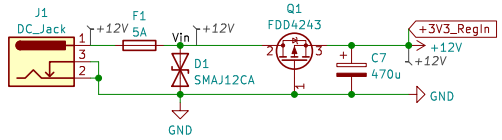
- `/` — Kořenová složka přiloženého média
  - `centralni-sprava/`
    - `backend/` — Zdrojové soubory serverové části systému pro centrální správu
    - `frontend/` — Zdrojové soubory klientské části systému pro centrální správu
  - `mereni/` — Soubory s provedenými měřeními
  - `napajeci-jednotka/`
    - `firmware/` — Zdrojové soubory obslužného firmwaru
      - `webapp/` — Zdrojové soubory klientské části administrace jednotky
    - `hardware/` — Obvodové schéma a návrh plošného spoje v KiCAD
  - `technicka-zprava/` — Zdrojové soubory této práce v LaTeX
  - `technicka-zprava.pdf` — PDF verze této práce

Příloha B

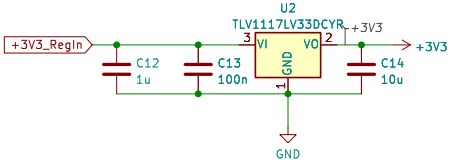
Schéma napájecí jednotky


Power
Soubor: power.kicad_sch
ESP32
Outputs
SDA
SCL
SBC1_EN
SBC1_FLT
SBC1_BTN
SBC2_EN
SBC2_FLT
SBC2_BTN
SBC3_EN
SBC3_FLT
SBC3_BTN
CRITICAL
Soubor: esp32.kicad_sch
Soubor: outputs.kicad_sch
H1
MountingHole_Pad
H2
MountingHole_Pad
H3
MountingHole_Pad
H4
MountingHole_Pad
GND
Sheet: /
File: pdu.kicad_sch
Title: PDU with Consumption Metering Function for SBCs
Size: A4
Date: 2023-05-03
Rev: 1.0 rev 3
KiCad E.D.A. kicad 7.0.0
Id: 1/4

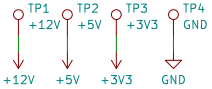
Power input



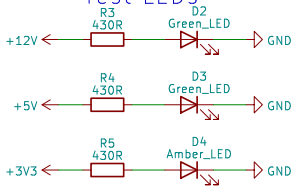
+3.3V LDO regulator



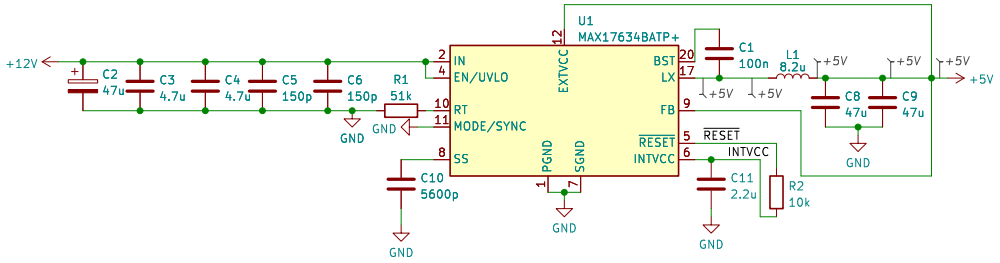
Test points



Test LEDs

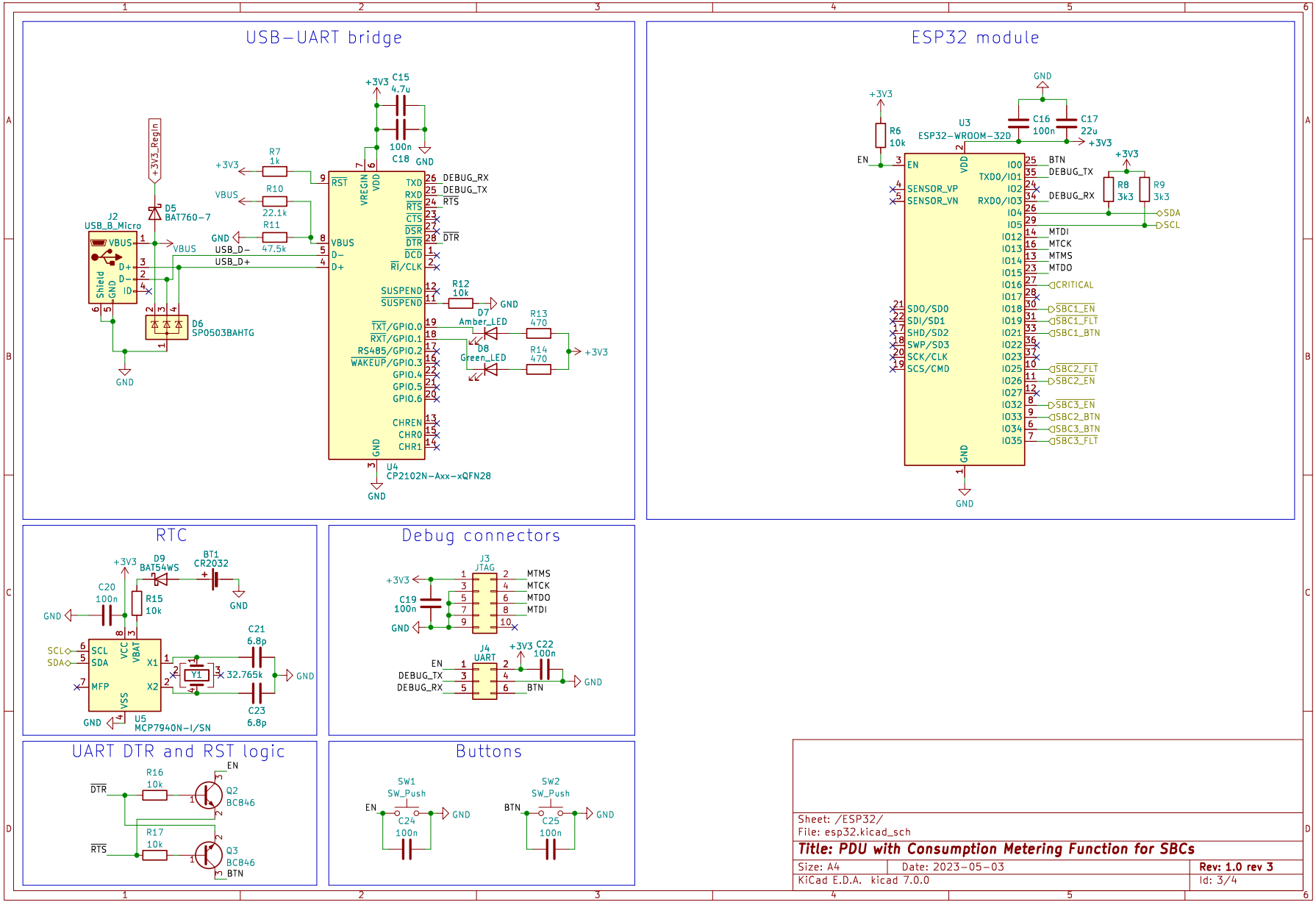


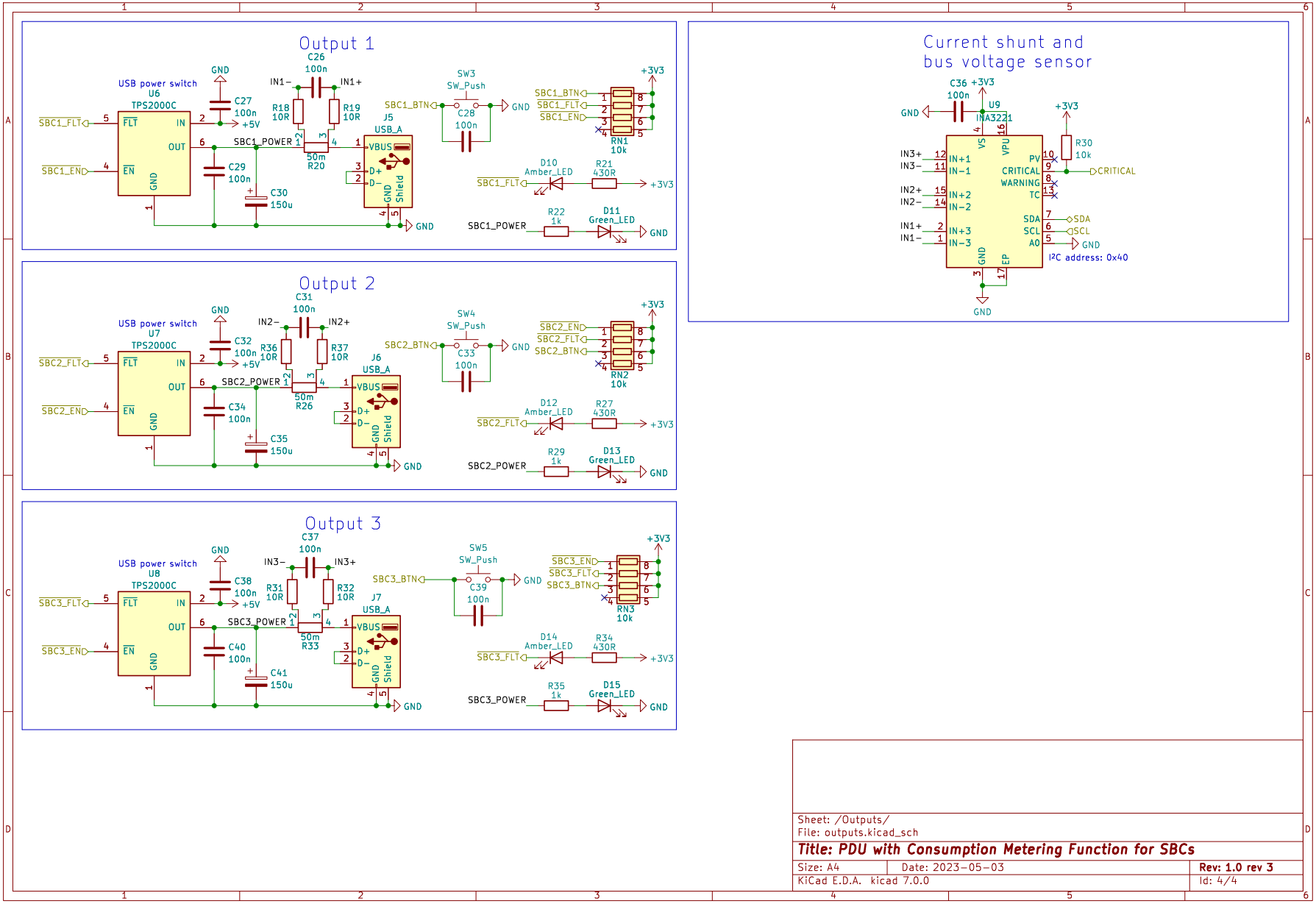
5V switching regulator



Sheet: /Power/	
File: power.kicad_sch	
Title: PDU with Consumption Metering Function for SBCs	
Size: A4	Date: 2023-05-03
KiCad E.D.A. kicad 7.0.0	Id: 2/4

Rev: 1.0 rev 3

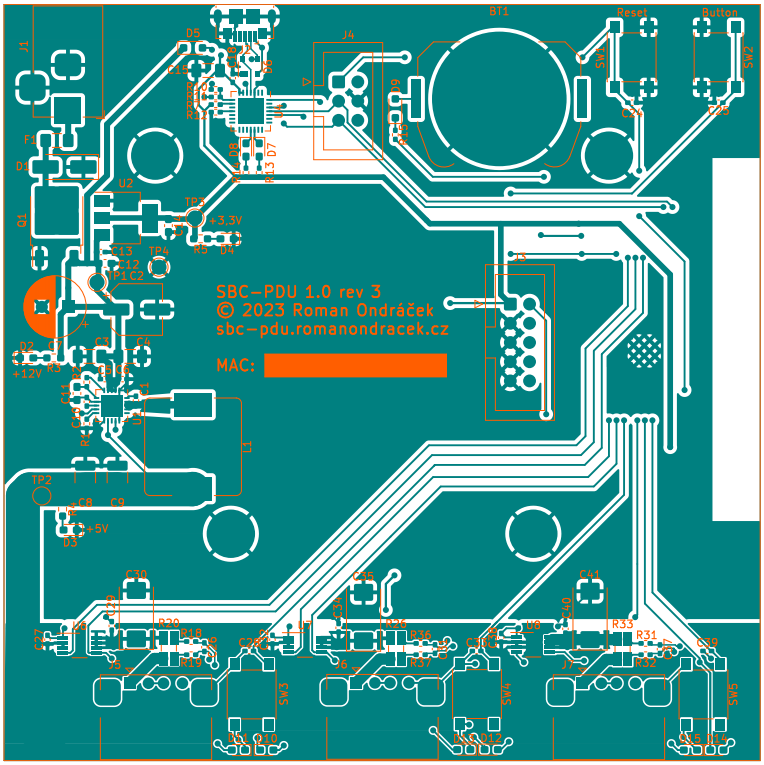

USB–UART bridge
ESP32 module
RTC
Debug connectors
UART DTR and RST logic
Buttons
Sheet: /ESP32/
File: esp32.kicad_sch
Title: PDU with Consumption Metering Function for SBCs
Size: A4
Date: 2023-05-03
Rev: 1.0 rev 3
KiCad E.D.A. kicad 7.0.0
Id: 3/4



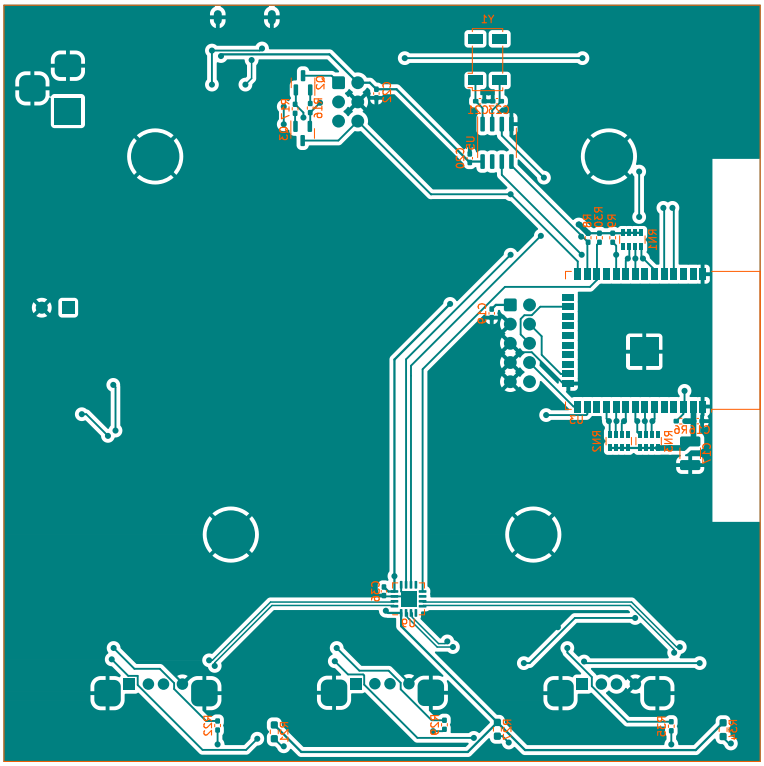
Sheet: /Outputs/		
File: outputs.kicad_sch		
Title: PDU with Consumption Metering Function for SBCs		
Size: A4	Date: 2023-05-03	**Rev: 1.0 rev 3**
KiCad E.D.A. kicad 7.0.0		Id: 4/4

Příloha C

Návrh plošného spoje



Obrázek C.1: Vrchní vrstva desky plošných spojů



Obrázek C.2: Spodní vrstva desky plošných spojů

Příloha D

Soupiska součástek

Reference	Kusů	Popis	Výrobce a model
BT1	1	Držák baterie CR2032	Keystone Electronics 3034
C1, C13, C16, C18-C20, C22, C24-C29, C31-C34, C36-C40	22	MLCC kondenzátor 0,1 µF 100 V, SMD 0402, X5R	Murata Electronics GRM155R62A104KE14J
C2	1	Elektrolytický kondenzátor 47 µF 50 V	Nichicon UCM1H470MCL1GS
C3, C4, C15	3	MLCC kondenzátor 4,7 µF 50 V, SMD 1206, X7R	Murata Electronics GRM31CR71H475KA12K
C5, C6	2	MLCC kondenzátor 150 pF 100 V, SMD 0402, C0G	TDK C1005C0G2A151J050BA
C7	1	Elektrolytický kondenzátor 470 µF 16 V, 8x11,5 mm	Nichicon UVR1C471MPD
C8, C9	2	MLCC kondenzátor 47 µF 10 V, SMD 1210, X7R	Murata Electronics GRM32ER71A476ME15L
C10	1	MLCC kondenzátor 5 600 pF 50 V, SMD 0402, C0G	Murata Electronics GRM1555C1H562GE01D
C11	1	MLCC kondenzátor 2,2 µF 10 V, SMD 0603, X7R	Murata Electronics GRM188R71A225KE15J
C12	1	MLCC kondenzátor 1 µF 25 V, SMD 0603, X5R	Samsung CL10A105KA8NNNC
C14	1	MLCC kondenzátor 10 µF 10 V, SMD 0603, X5R	Murata Electronics GRM188R61A106ME69D
C17	1	MLCC kondenzátor 22 µF 16 V, SMD 1206, X5R	Samsung CL31A226MOCLNNC
C21, C23	2	MLCC kondenzátor 6,8 pF 50 V, SMD 0402, C0G	Murata Electronics GRM1555C1H6R8DA01D
C30, C35, C41	3	Tantalový kondenzátor 150 µF 10 V	KYOCERA AVX TPSD157K010S0100
D1	1	12 V transil	Vishay SMAJ12CAHE3_A/H

Tabulka D.1: Soupiska součástek

Reference	Kusů	Popis	Výrobce a model
D2, D3, D8, D11, D13, D15	6	Zelená LED, SMD 0603	Wurth Elektronik 150060GS75000
D4, D7, D10, D12, D14	5	Oranžová LED, SMD 0603	Wurth Elektronik 150060AS75000
D5	1	Schottkyho dioda 0,5 A 30 V, SOD-232-2	Diodes Incorporated BAT760-7
D6	1	ESD ochrana pro USB	Littelfuse SP0503BAHTG
D9	1	Schottkyho dioda 0,2 A 30 V, SOD-232-2	Vishay BAT54WS-G3-08
F1	1	5 A pojistka, SMD 1206	Bel Fuse C1T5
J1	1	5,5/2,1 mm DC jack	CLIFF FC68148
J2	1	microUSB konektor	GCT USB3076-30-A
J3	1	JTAG konektor	Wurth Elektronik 61201021621
J4	1	UART konektor	Wurth Elektronik 61200621621
J5-J7	3	USB-A 2.0 konektor	GCT USB1125-GF-B
L1	1	Cívka 8,2 µH	Bourns SRP1245A-8R2M
Q1	1	P-MOSFET	onsemi FDD4243
Q2, Q3	2	NPN tranzistor, SOT-232-3	Nexperia BC846BW,135
R1	1	Rezistor 51 kΩ, SMD 0402	Panasonic ERJ-2RKF5102X
R2, R6, R12, R15-R17, R30	7	Rezistor 10 kΩ, SMD 0402	Bourns CR0402-FX-1002GLF
R3-R5, R21, R27, R34	6	Rezistor 430 Ω, SMD 0603	Panasonic ERJ-UP3J431V
R7, R13, R14, R22, R29, R35	6	Rezistor 1 kΩ, SMD 0402	YAGEO RC0402FR-071KL
R8, R9	2	Rezistor 3,3 kΩ, SMD 0402	YAGEO RC0402FR-073K3L
R10	1	Rezistor 22,1 kΩ, SMD 0402	YAGEO RC0402FR-1322K1L
R11	1	Rezistor 47,5 kΩ, SMD 0402	YAGEO RC0402FR-1047K5L
R18, R19, R31, R32, R36, R37	6	Rezistor 10 Ω, SMD 0402	Panasonic ERJ-2RKF10R0X
R20, R26, R33	3	Bočník 50 mΩ	Ohmite LVK12R050DER
RN1-RN3	3	Pole 4 rezistorů 10 kΩ	Bourns CAY16-1002F4LF
SW1-SW5	5	Tlačítko 6x6 mm	Omron B3FS-1000
U1	1	DC-DC snižující měnič s výstupem 5 V	Maxim Integrated MAX17634BATP+
U2	1	3,3 V LDO lineární regulátor, SOT-223-3	Texas Instruments TLV1117LV33DCYR

Tabulka D.2: Pokračování soupisky součástek

Reference	Kusů	Popis	Výrobce a model
U3	1	Wi-Fi bezdrátový modul	Espressif Systems ESP32-WROOM-32E-N16
U4	1	USB-UART převodník	Silicon Labs CP2102N-A02-GQFN28
U5	1	Hodiny reálného času	Microchip Technology MCP7940N-I/SN
U6-U8	3	USB spínač	Texas Instruments TPS2000CDGNR
U9	1	Senzor el. veličin	Texas Instruments INA3221AIRGVR
Y1	1	Krystal 32,768 kHz, 12,5 pF, SMD	ABRACON ABS25-32.768KHZ-T

Tabulka D.3: Pokračování soupisky součástek

Součástky pro osazení desky plošných spojů lze objednat přes následující odkazy: https://cz.mouser.com/ProjectManager/ProjectDetail.aspx?AccessID=e9668a2ef5 a https://www.tme.eu/cz/details/fc68148/konektory-dc/cliff/dc-10a-fc68148/.

D.1 Cena výroby jedné napájecí jednotky

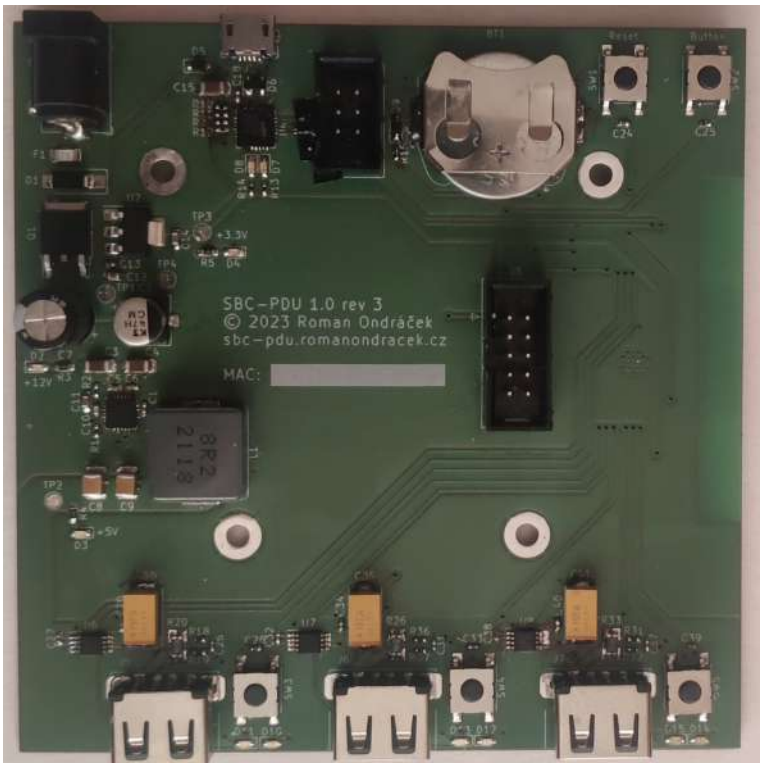
Celkové náklady na výrobu jedné napájecí jednotky jsou necelých 2 500 Kč, ceny jednotlivých položek jsou popsány v tabulce D.4.

Popis	Cena
Součástky z obchodu Mouser.com	1 363,14 Kč
Součástky z obchodu TME.eu	33,29 Kč
Plošný spoj vyrobený firmou Gatema PCB a.s.	1 089 Kč
Celkem	2 485,43 Kč

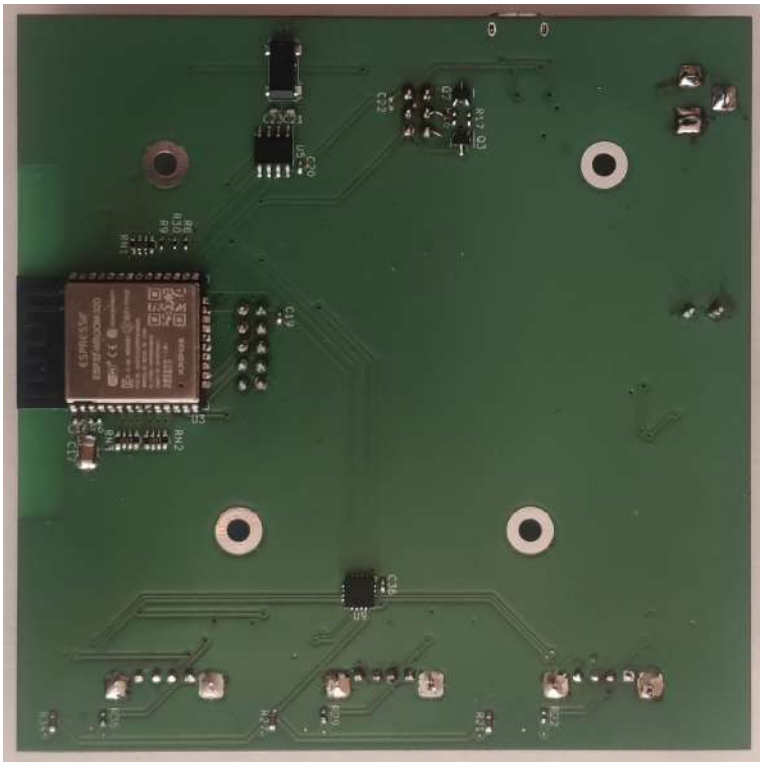
Tabulka D.4: Ceny jednotlivých položek potřebných k výrobě jedné napájecí jednotky

Příloha E

Fotografie napájecí jednotky



Obrázek E.1: Fotografie vrchní vrstvy desky plošných spojů napájecí jednotky



Obrázek E.2: Fotografie spodní vrstvy desky plošných spojů napájecí jednotky