VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Petr Ondráček

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ZAŘÍZENÍ PRO AUTOMATICKÉ MĚŘENÍ VOLTAMPÉROVÝCH CHARAKTERISTIK
DEVICE FOR AUTOMATIC MEASUREMENT OF VOLT-AMPERE CHARACTERISTICS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Petr Ondráček

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Ivo Lattenberg, Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Petr Ondráček ***ID:*** 186153

Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Zařízení pro automatické měření voltampérových charakteristik

POKYNY PRO VYPRACOVÁNÍ:

Realizujte elektronický měřič voltampérových charakteristik s využitím modulu ESP32. Zařízení bude měřit voltampérovou charakteristiku připojené součástky a ukládat do paměti. Přes webové rozhraní bude možno prohlížet grafy a odečítat hodnoty. Zařízení bude napájeno z externího zdroje a napěťový rozsah bude alespoň +/- 20 V a proudový +/- 200 mA. Rozsahy budou konfigurovatelné, bude možno nastavit i výkonové omezení.

DOPORUČENÁ LITERATURA:

[1] Elektrotechnická měření. Praha: BEN - technická literatura, 2002. ISBN 80-7300-022-9.

[2] BRTNÍK, Bohumil a David MATOUŠEK. Mikroprocesorová technika: [práce s mikrokontroléry řady ATMEL AVR ATXmega A4]. Praha: BEN - technická literatura, 2011. ISBN 978-80-7300-406-4.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: doc. Ing. Ivo Lattenberg, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Diplomová práce se zabývá návrhem a realizací elektronického měřiče voltampérových charakteristik pomocí Wifi modulu ESP32. Zařízení je navrhnuto takovým způsobem, aby bylo schopno proměřit charakteristiku v napěťovém rozsahu +/- 20 V a proudovém rozsahu +/- 200 mA. Zařízení je ovládáno pomocí webového rozhraní, ke kterému se uživatel může připojit například pomocí svého chytrého telefonu. Toto rozhraní umožňuje uživateli konfigurovat různé parametry, včetně rozsahu měření a výkonového omezení. Dále je možné graficky zobrazit naměřenou charakteristiku a tyto data exportovat. Pro nezávislost na okolní síti je ESP32 provozováno jako access point (přístupový bod), čímž vytváří novou jednoduchou síť bez připojení k internetu. V diplomové práci je teoreticky vysvětlen princip měření napětí a proudu, je navrhnuto zapojení zařízení, deska plošných spojů a model krytu zařízení. Rovněž je i popsán proces realizace zařízení podle vytvořeného návrhu a kryt zařízení je vytisknut pomocí 3D tiskárny. Poté je v práci navrhnut a popsán řídící program, který byl do výsledného zařízení nahrán. Na závěr je funkčnost zařízení otestována proměřením voltampérových charakteristik několika součástek.

KLÍČOVÁ SLOVA

Voltampérová charakteristika, Wifi modul, ESP32, webové rozhraní, přístupový bod, síť, deska plošných spojů

ABSTRACT

The Master's thesis discusses the design and realization of electronic meter of volt-ampere characteristics using Wifi module ESP32. The device is designed to be able to measure the characteristic in the voltage range of +/- 20 V and current range of +/- 200 mA. The device is controlled by web interface. The user is able to connect to it with for example his or her smartphone. This interface enables the user to configure various parameters, including the range of measurement and the power restriction. Furthermore, it is possible to graphically display the measured characteristic and export the data. To be independent on the outer network, the ESP32 is operated as an access point, which is creating a new simple network without the internet connection. In the Master's thesis, the principle of voltage and current measurement is explained theoretically, the circuit connection of device and the printed circuit board are designed and the cover of the device is modeled. Also the process of creating the device according to the design is described and the cover of the device is printed on 3D printer. After that, the control program is designed, described and uploaded into the resulting device. Finally, the functionality of the device is tested by measuring the volt-ampere characteristics of a few components.

KEYWORDS

Volt-ampere characteristic, Wifi module, ESP32, web interface, access point, network, printed circuit board

ONDRÁČEK, Petr. *Zařízení pro automatické měření voltampérových charakteristik*. Brno, 2020, 73 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Ivo Lattenberg, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Zařízení pro automatické měření voltampérových charakteristik“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Ivovi Lattenbergovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Cílem diplomové práce je navrhnout a realizovat elektronický měřič voltampérových charakteristik s využitím modulu ESP32. Zařízení bude schopné změřit voltampérovou charakteristiku uživatelem připojené součástky a ukládat naměřené hodnoty do RAM paměti mikrokontroleru. Zařízení bude fungovat jako Access point (přístupový bod), ke kterému se bude moci uživatel připojit pomocí sítě Wi-fi využitím svého počítače nebo chytrého telefonu. Dále zařízení umožní uživateli připojení k webovému rozhraní, na kterém bude možné konfigurovat různé parametry včetně konfigurace napěťového rozsahu a výkonového omezení měření. Pomocí webového rozhraní bude také umožněno samotné měření zahájit a zobrazit naměřené hodnoty v podobě grafu z něhož bude uživateli umožněno odečítat hodnoty napětí a odpovídající hodnoty proudu. Naměřené hodnoty bude také možné exportovat v podobě textového dokumentu. Zařízení bude napájeno z externího zdroje a napěťový rozsah bude alespoň +/- 20 V a proudový +/- 200 mA.

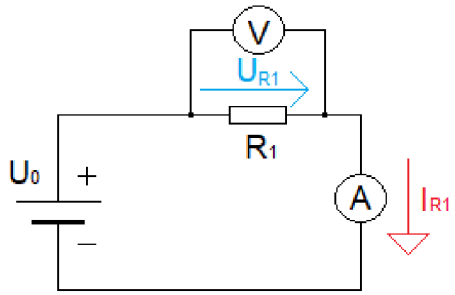
Samotná práce je logicky strukturovaná od teoretického základu, pres návrh zařízení až po konečnou realizaci. První kapitola se zabývá problematikou měření voltampérových charakteristik a převodu analogových signálu na digitální. Další kapitola je zaměřena na čip ESP32, který je nejdůležitější částí celého zařízení. Dále již následuje návrhová část práce. Nejdříve je vytvořeno blokové schéma zařízení, kde jsou vysvětleny základní části a funkce celého zařízení. V další kapitole je z tohoto blokového schématu vytvořeno schéma zapojení reálných součástek, které budou v zařízení použity. V následující kapitole je již ze schématu zapojení vytvořen návrh desky plošných spojů a zařízení je sestaveno včetně ochranného krytu, který byl vytisknut na 3D tiskárně. V předposlední kapitole je navrhnut řídící program, který je do desky následně nahrán. V poslední kapitole je proměřeno několik součástek a je tak otestována funkčnost vytvořeného zařízení.

1 Měření voltampérových charakteristik

Práce se zabývá návrhem a realizací zařízení, které je schopno proměřit voltampérovou charakteristiku uživatelem připojené součástky. Voltampérová charakteristika je grafická reprezentace závislosti mezi napětím a proudem, který prochází mezi dvěma měřenými uzly obvodu. Pokud se mezi těmito uzly nachází pouze jediná součástka, výsledná voltampérová charakteristika vyjadřuje vlastnost právě této součástky. Zařízení, které je výstupem této práce musí být navrhnuto takovým způsobem, aby bylo schopno naměřit hodnoty napětí a proudu, tyto hodnoty zpracovat a náležitě je uživateli interpretovat. Pro úplnost dokumentu je v další kapitole popsán způsob, jakým lze obecně napětí a proud převést na informaci, s kterou lze dále jednodušeji pracovat, například pomocí mikrokontroléru.

1.1 Měření napětí a proudu

Měření napětí se provádí pomocí voltmetru a měření proudu pomocí ampérmetru v zapojení, které je zachyceno na obrázku obr.1.1.



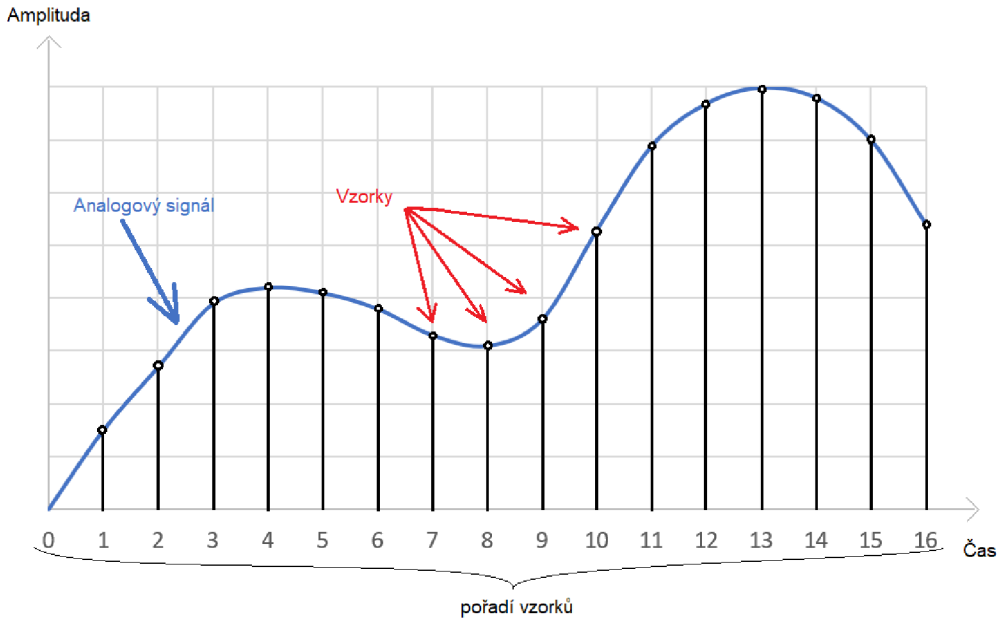
Obr. 1.1: Měření napětí a proudu

Svorky voltmetru jsou zapojené paralelně k měřené součástce. Vnitřní odpor voltmetru by měl mít co největší hodnotu, aby zanesl do měřených hodnot co nejmenší chybu (průchod proudu přes voltmetr je nežádoucí). Svorky ampérmetru jsou zapojeny sériově do obvodu. Vnitřní odpor ampérmetru by měl být co nejnižší, aby na něm vznikal co nejmenší úbytek napětí (nejlépe žádný, což je ale v reálném světě nemožné). Voltmetry a ampérmetry převádí elektrickou veličinu buďto na mechanický pohyb, který typicky vychýlí ručičku přístroje ukazující na naměřenou hodnotu (označovány jako analogové nebo elektromechanické) nebo převádí elektrickou veličinu na posloupnost jedniček a nul (digitální), jejichž výsledek je zobrazen například pomocí čísel na displeji. Pro účely této práce bude podrobněji popsán druhý způsob,

kdy dochází k převodu naměřené (analogové) veličiny na číslo, respektive posloupnost jedniček a nul (digitální signál). Tento převod se provádí pomocí analogově digitálního převodníku (dále označován jako A/D převodník). Princip funkce tohoto převodníku lze rozdělit do tří částí (vzorkování, kvantování a kódování), které jsou vysvětleny v následující kapitole.

1.1.1 Převod analogového signálu na digitální

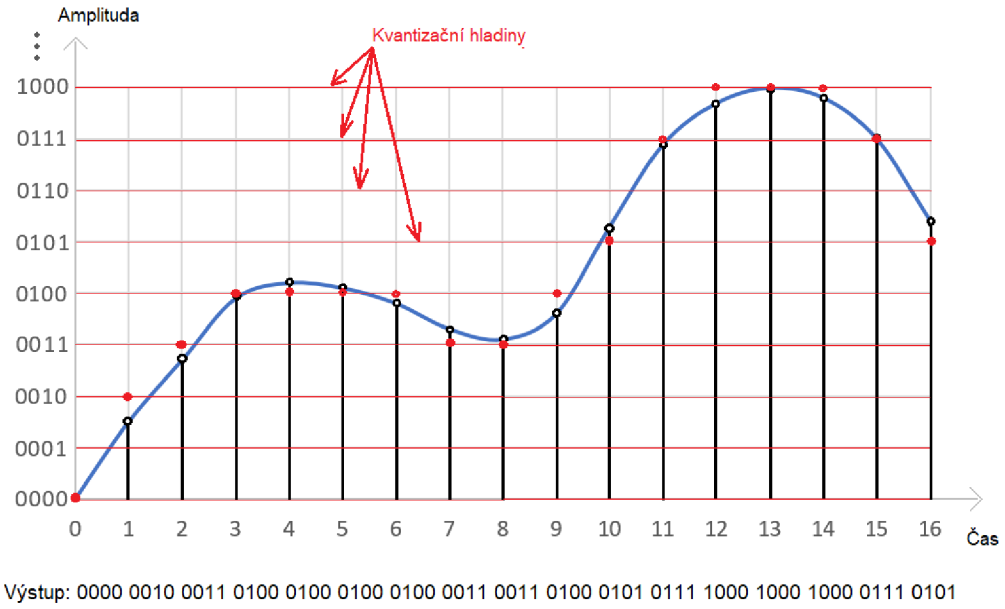
V první fázi převodu je provedeno vzorkování analogového spojitého signálu, kdy jsou odebrány vzorky aktuální úrovně signálu (typicky napětí) jak je naznačeno na obr. 1.2. Takto se ze spojitého signálu stává signál diskrétní.



Obr. 1.2: Vzorkování

Následuje krok kvantování. Jelikož jsme schopni pomocí počítače pracovat pouze s hodnotou určité přesnosti, je nutno rozdělit vzorky do konečného počtu napěťových úrovní (kvantizačních úrovní). Každý vzorek je tedy následně přiřazen do napěťové úrovně s hodnotou, která je nejblíže naměřenému napětí vzorku. Při tomto kroku dochází ke ztrátě části informace a vzniká tzv. kvantizační chyba. Tato chyba může nabývat hodnot od - ½ do + ½ velikosti kvantizační úrovně. U každého A/D převodníku je uvedena jeho přesnost pomocí počtu bitů. Například A/D převodník pracující s rozsahem signálu 5 V a s přesností 12 bitů je schopen vytvořit 4096 (2^{12}) kvantizačních hladin. Při vydělení napěťového rozsahu počtem kvantizačních hladin získáme vzdálenost jednotlivých napěťových úrovní. V uvedeném případě by od sebe byly tyto úrovně vzdáleny přibližně 1,22 mV. Poslední fází převodu je kódování.

V tomto kroku je každé napěťové úrovni přiřazena číselná posloupnost jedniček a nul a získáváme tak požadovaný digitální signál, který lze dále zpracovávat. Krok kvantování a kódování je ilustrován na obr. 1.3.



Obr. 1.3: Kvantování a Kódování

2 ESP32 - základní vlastnosti a parametry

ESP32 je centrální prvek celého zařízení, jehož funkce je řízení všech periferií, čtení dat, jejich zpracování a interpretace pro uživatele. Jedná se o mikrokontrolér podporující Wifi i Bluetooth komunikační technologie. Uveden na trh byl v roce 2016 firmou Espressif Systems, která sídlí v Šanghaji. Jedná se o velice moderní mikrokontrolér, který se svými rozměry $1{,}80 \times 2{,}55$ cm v kombinaci s širokou škálou funkcí vytváří skvělý čip pro použití v IoT (Internet věcí z anglického Internet of Things), automatizaci domácnosti, implementace dálkového ovládání různých projektů a mnoho dalších.

Shrnutí některých klíčových informací, vlastností a parametrů, kterými čip ESP32 disponuje dle oficiální dokumentace firmy Espressif Systems [1]:

- 32-bitový LX6 mikroprocesor s nastavitelnou frekvencí vnitřních hodin od 80 do 240 MHz a vnitřní pamětí 448 KB ROM, 520 KB SRAM (+ 8 KB RTC FAST a 8 KB RTC SLOW, rovněž typu SRAM, které lze využít i při režimu spánku)
- Podpora připojení externí paměti pomocí SPI sběrnice s možností adresace až 16 MB paměti FLASH a 8 MB SRAM.
- Wi-fi rozhraní podporující standardy IEEE 802.11b/g/n pracující na frekvenci 2,4 GHz s podporou zabezpečení dat pomocí WEP, WPA + WPA2 a jiných
- Podpora Bluetooth rozhraní + BLE (Bluetooth low energy) pro aplikace vyžadující nízkou spotřebu energie
- Potřebné napájecí napětí 3,0 - 3,6 V s proudovým odběrem pohybující se při v běžném provozu okolo 80 mA. Ovšem špičkový odběr může dosahovat velikostí přesahující i hodnotu 200 mA a to například při vysílání datových jednotek pomocí Wi-fi rozhraní. Naopak pro aplikace vyžadující velice nízký proudový odběr je čip schopen přejít do nejrůznějších režimů spánku, kdy dochází k odpojení nevyužívaných periferií čipu a proudový odběr se může snížit až na hodnoty desítek mikroampér.
- 38 pinů, z nichž je 32 GPIO (z anglického general purpose input-output) disponující nejrůznějšími parametry jako je například podpora sběrnic I^2C, UART, SPI, vybavení vnitřními AD převodníky, DA převodníky, funkce generování PWM signálu a jiné.
- Rozsah pracovní teploty -40 až +85 °C

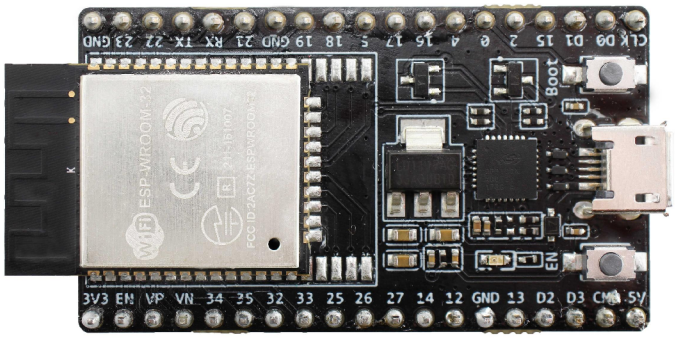
2.1 Moduly čipu ESP32

Firma Espressif Systems dodává čip ESP32 v nejrůznějších provedeních. Níže jsou zmíněny některé dnes dostupné moduly.

- ESP32-WROOM-32 je jeden z nejmenších ($1{,}80 \times 2{,}55\,\text{cm}$) dodávaných modulů firmou Espressif Systems a její využití je především pro koncové produkty. Tento modul je použit i v této práci.

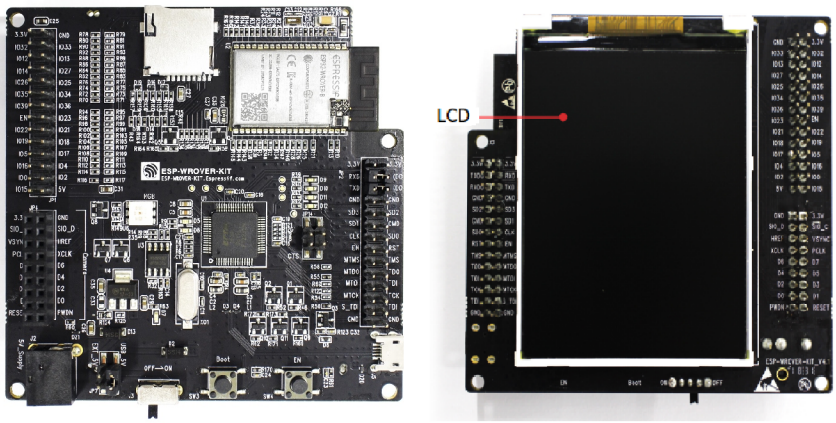
Obr. 2.1: ESP32-WROOM-32 [2]

- ESP32 DevKitC V4 je velice často používaný vývojový modul. Jeho součástí je již zmíněný ESP32-WROOM-32 modul, ke kterému jsou přidány podpůrné periferie pro zjednodušení práce s čipem. Jedná se především o USB/UART převodník CP2102, díky kterému lze čip pohodlně programovat a zároveň je pomocí USB modul napájen. Dále modul disponuje například tlačítkem pro restart, indikační LED diodou a také je pro ulehčení práce uživatele vyvedena většina GPIO pinů na okraj desky.



Obr. 2.2: ESP32 DevKitC V4 [2]

- ESP-WROVER-KIT V4.1 je opět vývojový modul, který na první pohled zaujme svými rozměry. Tento modul je vhodný pro velmi komplexní a náročné aplikace. Obsahuje například port pro MicroSD kartu, externí PSRAM paměť o velikosti 8 MB, vstupy pro kameru, LCD displej, rozhraní JTAG pro debugování čipu, dvě možnosti napájení pomocí USB nebo 5 mm jack konektoru, má vyvedeny všechny GPIO piny pro jednoduchý přístup a spoustu dalších užitečných periferií.



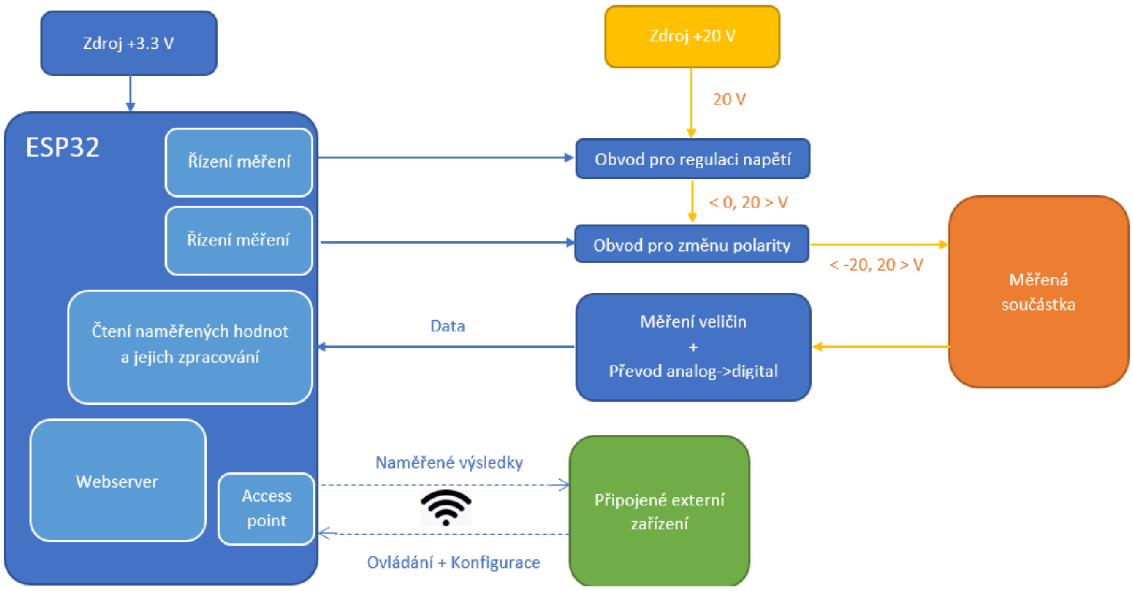
Obr. 2.3: ESP-WROVER-KIT V4.1 [2] [3]

2.2 Ovládání a programování

Čip ESP32 je výrobcem dodáván se základním firmwarem, který umožňuje ovládání čipu pomocí tzv. AT příkazů (z anglického Attention). Pro vytvoření vlastního řídícího programu a ovládání čipu včetně jeho vyvedených GPIO pinů je možné do čipu nahrát jiný firmware. Těchto firmwarů je dnes dostupné velké množství. V práci bude využit Arduino firmware, umožňující psát řídící program v programovacím jazyce Arduino, což je v podstatě vybraná množina funkcí jazyků C a C++. Je tedy využíván klasický C/C++ kompilátor pro překlad do strojového jazyka. K tvorbě řídícího programu je využito open-source vývojové prostředí Platform.io, které se v dnešní podobě využívá jako plugin vývojových prostředí jako jsou například Visual Studio, Atom.io či Eclipse. Platform.io umožňuje automatické nahrání Arduino firmwaru a taktéž kompilaci a nahrání řídícího programu na čip ESP32. [4] [5]

3 Funkce zařízení - blokové schéma

Tato kapitola je zaměřena pouze na popis principu funkce celého zařízení. Pro zjednodušení a lehčí pochopení funkce zařízení prozatím nejsou řešeny konkrétní součástky a jejich zapojení. Touto problematikou se zabývá následující kapitola 4. Pro popis funkce navrhovaného zařízení bylo vytvořeno blokové schéma, které je zachyceno na obrázku 3.1.



Obr. 3.1: Blokové schéma zařízení

Centrální prvek celého zařízení je čip ESP32, který je napájen zdrojem napětí 3,3 V. Tento čip se stará o ovládání připojených periferií, sběr dat a jejich zpracování. Dále umožňuje uživateli připojení k zařízení pomocí bezdrátové sítě WiFi. Je důležité podotknout, že součástí zadání je provozovat čip jako Access point (přístupový bod), čili je vytvářena zcela nová bezdrátová síť, bez připojení k internetu, čímž je umožněno provozování zařízení prakticky kdekoliv, nezávisle na okolní síti.

Na vytvořené síti bude zařízení hostovat jednoduchý web server a uživateli připojenému k síti bude k tomuto serveru umožněn přístup pomocí webového prohlížeče. Na tomto serveru je uživateli umožněno konfigurovat parametry měření a následně výsledky měření graficky zobrazit a data exportovat.

Další částí zařízení jsou obvody umožňující proměření dané součástky. Výsledné zařízení má být schopno proměřit voltampérovou charakteristiku měřené součástky od velikosti napětí -20 až +20 V. Tudíž je zapotřebí zdroj, který je schopen dodat minimálně hodnotu 20 V. Výsledné napětí je ovšem vyšší z důvodů, které jsou vysvětleny v dalších kapitolách. Toto napětí je při samotném měření regulováno

pomocí obvodů ovládaných čipem ESP32. Zařízení je rovněž schopeno změnit polaritu napětí na měřené součástce. Díky těmto vytvořeným periferiím bude zařízení splňovat požadavky měření specifikované v zadání práce.

Dále je nutné, aby zařízení mohlo na měřené součástce získat (naměřit) informaci o aktuálním napětí a procházejícím proudu. Tyto veličiny jsou zpracovány obslužnými obvody (A/D převodník) a čipem ESP32 jsou zpracovány na data, z kterými zařízení dále pracuje. V průběhu měření je také kontrolováno, zda nedochází k překročení nastavených úrovní napětí a proudu a v případě, že jsou tyto hodnoty překročeny bude měření ukončeno předčasně, aby nedošlo k poškození měřené součástky.

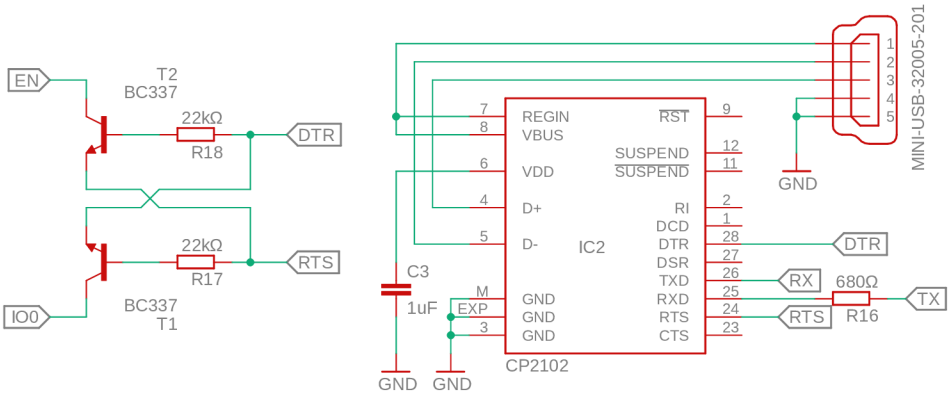
Tímto byla jednoduše vysvětlena funkce celého zařízení a další kapitola se již věnuje návrhu reálného schématu zapojení a výběru konkrétních součástek.

4 Návrh schématu zapojení

V této kapitole je postupně vysvětleno zapojení celého zařízení a v podkapitolách jsou pro každou nově uvedenou část zapojení vybrány konkrétní součástky s vysvětlením jejich výběru a u některých i porovnání s jinými alternativami.

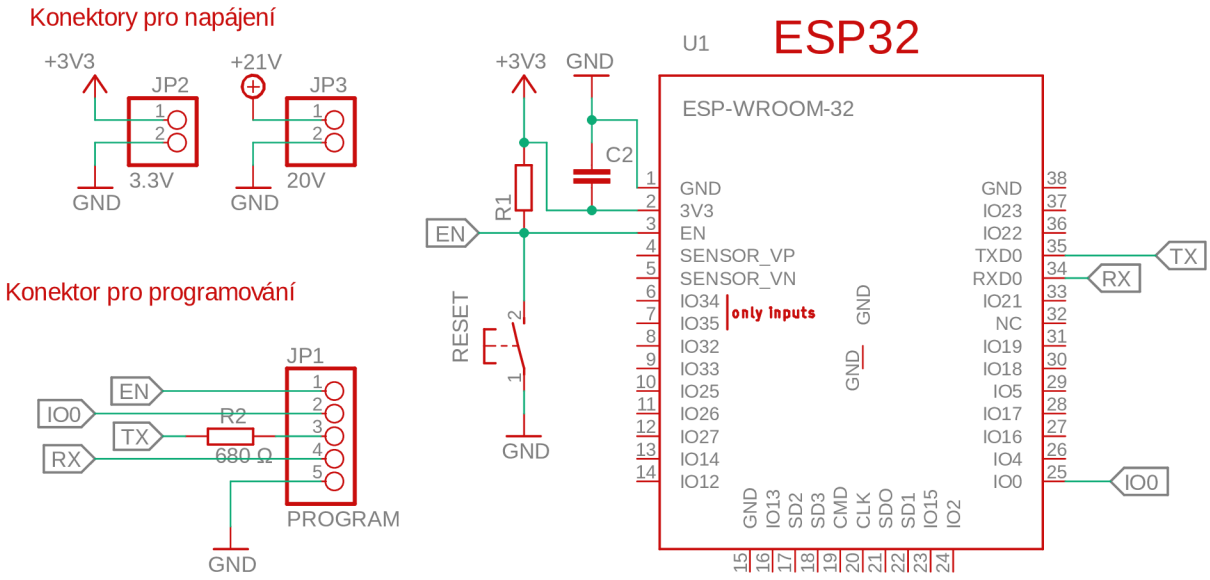
4.1 ESP32, napájení a konektor pro programování

Napájení pro ESP32 je v technické dokumentaci [1] definováno v rozmezí od 3,0 do 3,6 V s typickou hodnotou 3,3 V. Pro naši aplikaci bylo zvoleno napájení pomocí step-down konvertoru, na jehož výstupu je právě stejnosměrné napětí 3,3 V. Typicky je v co nejmenší vzdálenosti napájecího pinu čipu připojen také kondenzátor (mezi pinem 3V3 a GND), který slouží k redukci případného poklesu napájecího napětí při prudkém zvýšení odebíraného proudu. Pro regulovaný zdroj napětí byl zvolen step-up konvertor s nastavitelným výstupem minimálně 20 V (Konečné nastavení konvertoru bude mít na výstupu napětí vyšší z důvodů, které budou vysvětleny v kapitole 4.2 Regulace napětí). Oba konvertory budou napájeny pomocí 12 voltového síťového adaptéru. Schéma zapojení, z kterého bude později vygenerován návrh desky plošných spojů reprezentuje oba konvertory pouze pomocí dvoupinových konektorů (prvky JP2 a JP3), jelikož se nebudou nacházet přímo na desce. Pro umožnění programování a komunikaci pomocí rozhraní UART využívá ESP32 piny s názvem RX a TX. Pro umožnění komunikace a automatického programování byl vytvořen adaptér obsahující USB/UART převodník CP2102 a dva bipolární NPN tranzistory. Zapojení adaptéru je převzato z technické dokumentace ESP32 DevKit modulu [6], který tento převodník používá. Pro automatické programování jsou využity kromě pinu RX a TX také piny EN a IO0 (+ společná zem GND). Schéma zapojení adaptéru je zachyceno na obr. 4.1.



Obr. 4.1: Adaptér pro automatické programování

Adaptér není součástí konečného zařízení, proto bude v návrhu zahrnut pouze konektor (JP1), který má vyvedeny již zmíněné piny RX, TX, EN, IO0 a GND (a rezistor R2 pro redukci vysokofrekvenčního šumu na vysílacím pinu TX s hodnotou 680 Ω). Zmíněný pin EN slouží k automatickému restartu čipu ESP32 ve chvíli, kdy se na pinu nachází logická nula (je uzemněn). Proto musí být při běžném provozu na tento pin přivedena logická jednička, což je v zařízení realizováno pomocí pull-up rezistoru R1 s hodnotou 10 kΩ. Pro jednodušší práci bylo také přidáno tlačítko pro manuální reset (opět převzato ze zapojení ESP32 DevKit modulu [6]). Výstupní schéma zapojení kapitoly 4.1 je zobrazeno na obr. 4.2.



Obr. 4.2: Základní zapojení čipu ESP32

4.1.1 Zvolené komponenty pro realizaci zapojení kapitoly 4.1

Pro realizaci zařízení byl vybrán model čipu ESP-WROOM-32, který byl již představen v kapitole 2.1 Moduly čipu ESP32.

Jako 3,3 V zdroj byl zvolen běžně dostupný modul step-down konvertoru XL4015. Napěťový vstup tohoto konvertoru je stanoven v technické dokumentaci [7] v rozmezí 8 až 36 V (lze tedy použít 12 V síťový adaptér) a výstup v rozmezí 1,25 až 32 V (splňuje požadavky pro napájeni ESP32).

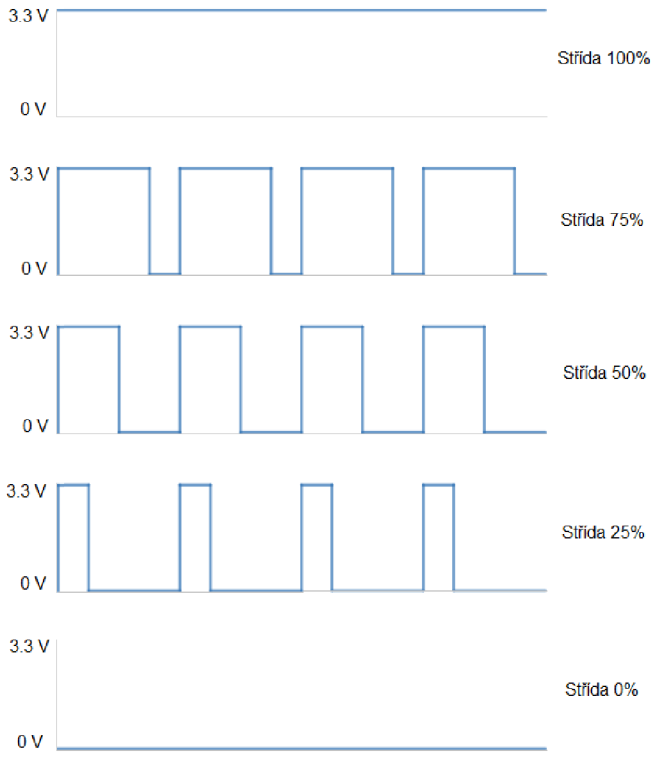
Jako 20 V zdroj byl zvolen také běžně dostupný modul step-up konvertoru XL6009. Napěťový vstup tohoto konvertoru je stanoven v technické dokumentaci [8] v rozmezí 3 až 30 V (lze použít 12 V) a výstup v rozmezí 5 až 35 V (opět splňuje požadavky pro potřeby měření). Maximální proud je 2 A.

4.2 Regulace napětí

Pro řešení této části obvodu je nutno nejdříve objasnit princip pulzně šířkové modulace, demodulace a funkci dolní propusti.

4.2.1 Pulzně šířková modulace

Pulzně šířková modulace (dále označována jako PWM z anglického Pulse Width Modulation) je způsob přenosu analogového signálu pomocí dvouúrovňového digitálního signálu. Signál je přenášen jako posloupnost různě širokých impulzů přenášených v pravidelných časových intervalech. Šířka těchto pulzů je dána tzv. střídou, která reprezentuje (v procentech) čas zastoupení logické jedničky (šířku impulzu) v jedné periodě signálu. Velikost střídy je funkcí vstupního analogového signálu. Příklad hodnot střídy a odpovídajícímu signálu je uveden na obr. 4.3.

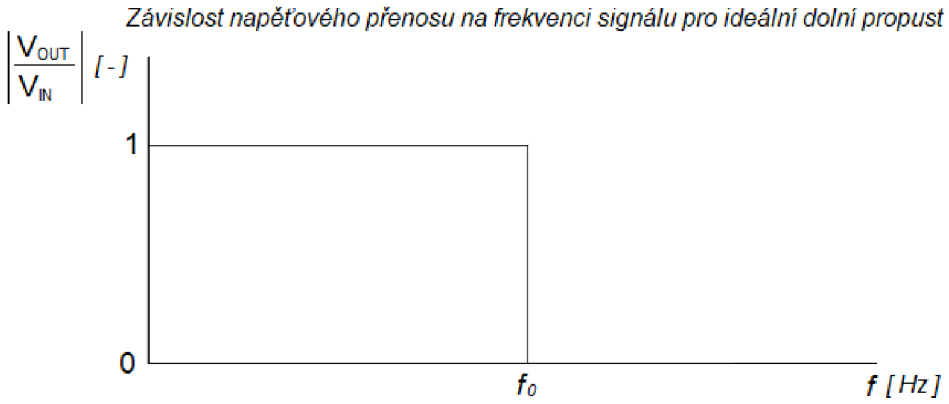


Obr. 4.3: Pulzně šířková modulace

Zpětná demodulace z digitálního na analogový signál je velmi jednoduchá. Lze ji realizovat pomocí tzv. dolní propusti, která v ideálním případě propustí pouze stejnosměrnou část signálu, která je rovna průměrné hodnotě napětí přenášeného signálu a zároveň odpovídá původnímu analogovému signálu, který byl přenášen. Princip dolní propusti je podrobněji popsán v následující kapitole.

4.2.2 Dolní propust

Dolní propust (dále DP) je nízkofrekvenční filtr. Jak již bylo zmíněno, v ideálním světě by DP propustila pouze stejnosměrnou složku signálu, respektive všechny nízkofrekvenční složky signálu, jejichž frekvence nepřekračuje tzv. mezní frekvenci f_0 a všechny složky signálu s frekvencí vyšší než f_0 by byly utlumeny na nulovou hodnotu. Frekvenční přenosová charakteristika takového filtru je znázorněna na obr. 4.4. Napěťový přenos filtru je definován jako poměr výstupního napětí V_{OUT} a vstupního V_{IN}.



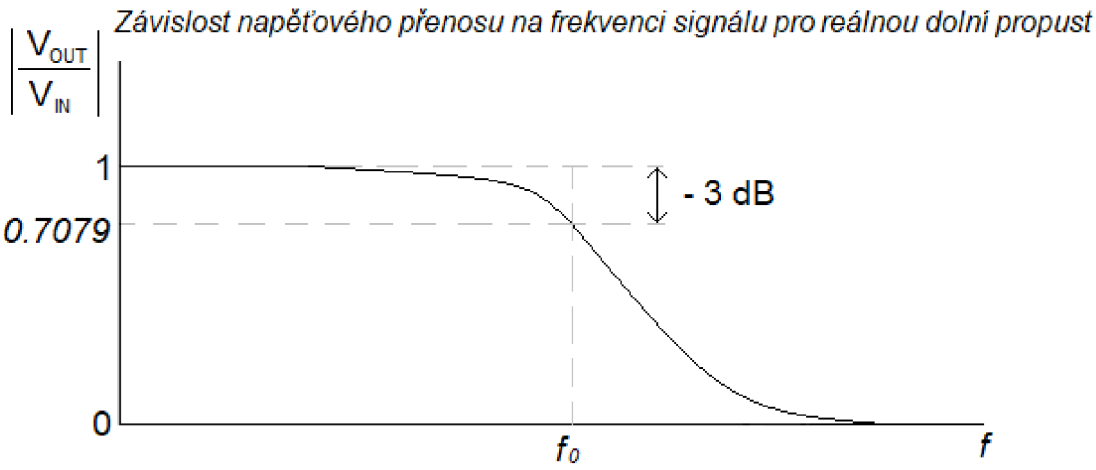
Obr. 4.4: Přenosová charakteristika ideální DP

Ovšem realita se od teoretického modelu filtru liší, jelikož charakteristiky reálných filtrů nemohou mít nekonečně velkou strmost přenosové funkce na mezní frekvenci f_0, tak jak to bylo znázorněno na obr. 4.4. Reálná DP má strmost s konečnou velikostí a její graf je spojitá funkce. Dále je nutno nově definovat mezní frekvenci f_0. U reálné DP je definována jako frekvence, kdy dojde k přenosovému poklesu o 3 dB (což odpovídá přibližně poklesu o 29,21 %). Příklad frekvenční přenosové charakteristiky reálné DP je znázorněn na obr. 4.5.

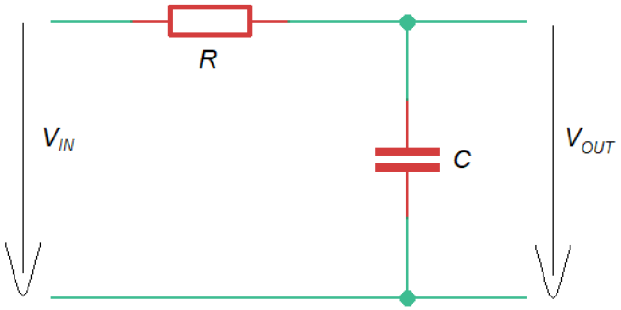
Existuje několik druhů realizací DP, pro účel této práce bude zmíněna pouze jedna varianta, která bude i použita. Jedná se o pasivní DP 1. řádu. Tato DP se skládá z rezistoru (odpor R) a kondenzátoru (kapacita C). Tento filtr je často označován jako RC článek. Jeho zapojení je znázorněno na obrázku 4.6.

Pasivní filtry neobsahují zesilovače, což znamená že maximum jejich přenosové charakteristiky nemůže přesáhnout hodnotu 1. Reálná DP tedy v zásadě vždy signál zeslabuje (např. z důvodu ztrát na rezistoru). Mezní frekvence udávající pokles signálu o 3 dB je u RC článku definována vztahem

$$f_0 = \frac{1}{2 \cdot \pi \cdot R \cdot C} \tag{4.1}$$



Obr. 4.5: Přenosová charakteristika reálné DP



Obr. 4.6: Zapojení RC článku

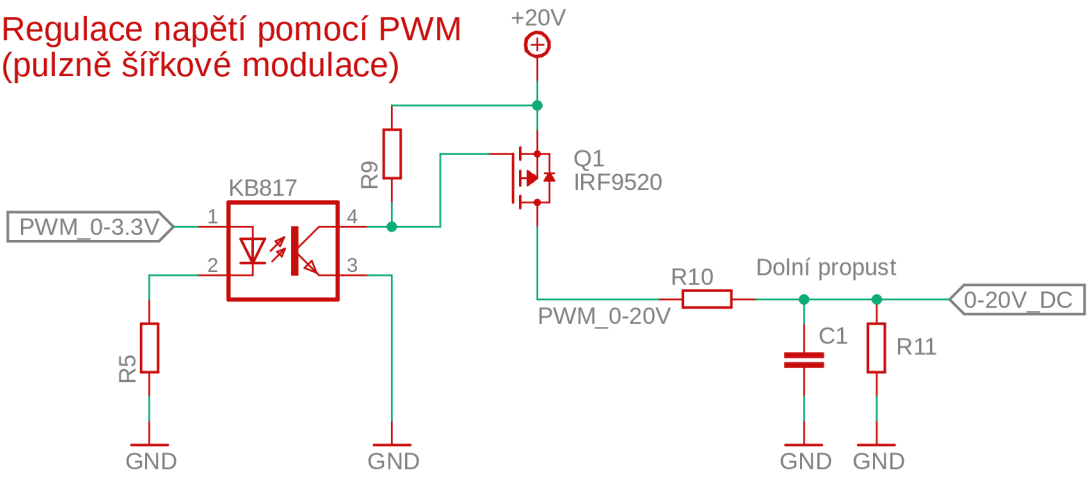
4.2.3 Obvod pro regulaci napětí

Pro dodržení požadavků měření je zapotřebí navrhnout zapojení takovým způsobem, aby bylo zařízení schopno dodat napětí v rozsahu od -20 až +20 V na svorkách, kde bude připojena měřená součástka. Byl zvolen zdroj kladného stejnosměrného napětí +20 V. Obvod pro regulaci tohoto napětí je rozdělen na dvě části:

- Obvod pro regulaci napětí od 0 do +20 V
- Změna polarity napětí na měřené součástce pro získání konečného rozsahu -20 až +20 V

Regulace napětí od 0 do +20 V

Při manipulaci s napětí dosahující hodnot až 20 V pomocí čipu ESP32 vzniká problém, jelikož tento čip funguje na 3,3 V logice a jakékoliv přímé spojení s napětím 20 V by čip nenávratně zničilo. Bylo zvoleno ovládání pomocí pulzně šířkové modulace (PWM). Cílem zapojení je pomocí změny střídy měnit napětí na měřené součástce. Signál PWM bude vytvářet čip ESP32 na pinu GPIO33. Čip je schopen generovat PWM signál o amplitudě 3,3 V a střídou s přesností 16 bitů (65536 úrovní od 0 - 100 % střídy), což ve výsledku umožňuje velice jemnou regulaci napětí. Tento řídící signál bude přenesen na 20 V větev obvodu pomocí optronu, čímž dosáhneme galvanického oddělení (a tím ochrany čipu). Tento optron by mohl přímo sloužit ke spínání připojených 20 V. Ovšem optrony mají relativně velký odpor v sepnutém stavu (stav, kdy jimi prochází proud) a při proudu dosahujícím hodnot až 0,2 A by mohlo docházet k jeho přehřívání. Proto bylo zvoleno zapojení s přidáním výkonového MOSFET tranzistoru, jehož odpor je v sepnutém stavu minimální (v řádu jednotek ohmů). Zapojení je zachyceno na obr. 4.7.



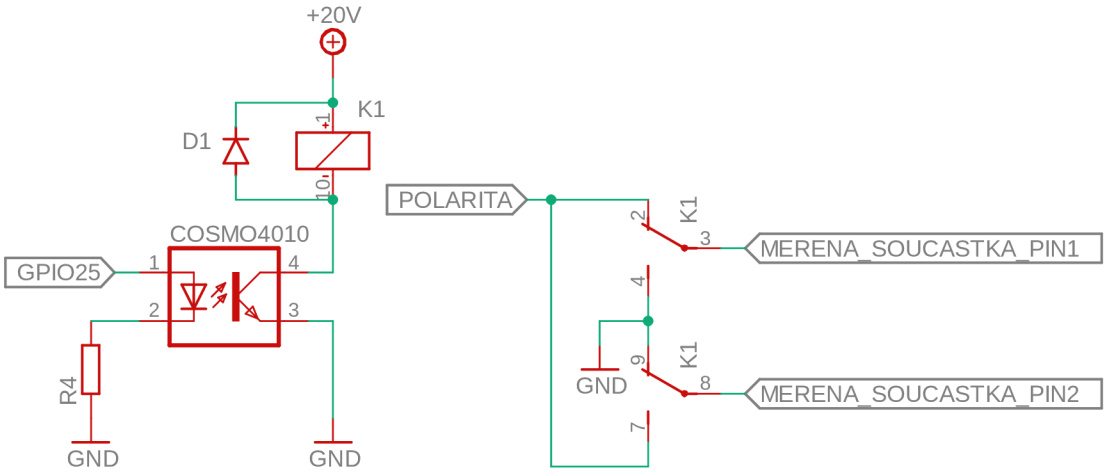
Obr. 4.7: Převod 3,3 V PWM na 20 V PWM

Funkce obvodu je následující. Při přivedení logické jedničky na anodu optronu dojde k sepnutí uzlu připojeného ke kolektoru na zem. Při logické nule na anodě je naopak uzel připojený ke kolektoru pouze propojen k napájecímu napětí 20 V pomocí pull-up rezistoru. Získáváme tedy inverzní signál generovaného PWM signálu. Uzel kolektoru je zároveň připojen k GATE terminálu P-channel (P kanál) MOSFET tranzistoru. Tento typ tranzistoru se nachází v sepnutém stavu pokud se na jeho GATE terminálu nachází logická nula (je uzemněn). Rozepnutý stav naopak pokud je připojen k logické jedničce (připojen k napájecímu napětí). Drain pin MOSFET tranzistoru je zapojen přímo na napájecí napětí 20 V. Tímto zapojením

bylo dosaženo, že výstupní pin MOSFET tranzistoru (SOURCE) kopíruje průběh PWM signálu generovaného čipem ESP32 s amplitudou dosahující 20 V. Nyní již stačí signál vyfiltrovat pomocí DP (RC článek) a je tak získán regulovatelný zdroj stejnosměrného napětí. Dále byl zapojen rezistor paralelně ke kondenzátoru, jehož úloha je umožnit vybíjení kondenzátoru i za podmínek, že měřena součástka nabývá vysokých hodnot odporu, kdy by kondenzátor byl rychleji nabíjen než vybíjen a docházelo by k nabití na maximální hodnotu PWM signálu (20 V). Hodnoty jednotlivých odporů, kondenzátoru a parametry konkrétních komponent jsou určeny a zdůvodněny v kapitole 4.2.4 Zvolené komponenty pro realizaci zapojení kapitoly 4.2.

Změna polarity napětí

Pro rozšíření napěťového rozsahu vytvořeného zdroje o záporné hodnoty napětí bylo zvoleno jednoduché řešení pomocí elektromagnetického relé. Spínání cívky elektromagnetického relé pomocí ESP32 je opět řešeno pomocí optronu (pomocí pinu GPIO25). Zapojení elektromagnetického relé bylo realizováno takovým způsobem, aby při jeho sepnutí došlo ke změně polarity na měřené součástky (zdroj napětí přepojen na zem, zem přepojena na zdroj napětí). Při rozepínání cívky zaniká magnetické pole cívky a vzniká tak impulzní záporné napětí dosahující špiček až pětinásobku spínaného napětí, což by mohlo poškodit některé části zařízení. Byla proto zapojena dioda paralelně k cívce, která toto záporné napětí bezpečně svede k zemi. Zapojení elektromagnetického relé je zachyceno na obr. 4.8.



Obr. 4.8: Změna polarity napětí

4.2.4 Zvolené komponenty pro realizaci zapojení kapitoly 4.2.

Je potřeba aby bylo zařízení schopno dodat minimálně napětí 20 V a proud 0,2 A na svorky měřené součástky. Podle ohmova zákonu to znamená, že zařízení musí být navrhnuto takovým způsobem, aby bylo schopno dodat tyto hodnoty i v případě zapojení 100 Ω rezistoru, kdy dojde k dosáhnutí těchto hodnot napětí a proudu ve stejné chvíli (při menším odporu dojde dříve k překročení proudu, při větším odporu dojde dříve k překročení napětí). Navrhované hodnoty komponent budou voleny v závislosti na této hodnotě (označeno R_{mer} jako odpor měřené součástky). Je zjevné, že při zapojení nabíjecího rezistoru (R_{10} dále označen R_{nab}) u DP a paralelní kombinaci odporu měřené součástky (R_{mer}) a vybíjecího rezistoru (R_{11} dále R_{vyb}) vzniká napěťový dělič. Nepatrnou roli hraje i odpor MOSFET tranzistoru (R_{MOSFET} v sepnutém stavu. Pro napětí přivedené na svorky měřené součástky (U_{mer}) proto platí následující vztah:

$$U_{mer} = U_{zdroj} \cdot \frac{R_{mer}||R_{vyb}}{R_{mer}||R_{vyb} + R_{nab} + R_{MOSFET}} \tag{4.2}$$

Z tohoto vztahu plyne, že pro dosáhnutí napětí U_{mer} velikosti 20 V je zapotřebí aby napájecí napětí U_{zdroj} bylo vyšší než 20 V. Tuto hodnotu nelze určit dokud nejsou pevně dány hodnoty odporů R_{vyb} a R_{nab}. Tyto rezistory jsou součástí již několikrát zmiňované dolní propusti (RC článku). Pro určení hodnot jednotlivých komponent je zapotřebí vědět jakou má mít filtr mezní frekvenci. Tato frekvence je dána generovaným signálem PWM.

Na první pohled by se zdálo, že ze vztahu pro výpočet mezní frekvence (4.1) stačí pouze zvolit jednoduše dostupné hodnoty odporu a kapacity a náležitě přizpůsobit frekvenci PWM signálu. Bohužel, při zvyšování frekvence se začínají uplatňovat parazitní vlastnosti reálných součástek. U optronu začíná při vyšších frekvencích hrát roli parametr doby přechodu z logické jedničky na logickou nulu a naopak. Běžně je v technické dokumentaci tato doba uváděna jako doba odezvy (anglicky response time) a je rozdělena na dva různé časy a to doba náběhu t_r (anglicky rise time) a doba pádu t_f (anglicky fall time). Doba t_r reprezentuje dobu přechodu z logické nuly na logickou jedničku a doba t_f dobu přechodu z logické jedničky na logickou nulu. Druhá součástka, jejíž parametry jsou frekvenčně závislé je MOSFET tranzistor. Při vyšších frekvencích se začíná uplatňovat parazitní kapacita GATE terminálu, respektive doba jejího vybíjení a nabíjení. V praxi to znamená jak rychle jsme schopni přesunout náboj z nebo do GATE terminálu. Průchodem náboje za čas je přímo definován elektrický proud. Z toho vyplývá že čím vyšší proud protéká do či od GATE terminálu, tím kratší je doba jeho nabíjení či vybíjení. Opět z ohmova zákona vyplývá, že tento proud je přímo úměrný napětí a nepřímo na odporu. Zvyšování napětí v této

části obvodu nepřipadá v úvahu, takže jediná možnost jak se s danou situací vypořádat je snížit odpor. Při vybíjení hraje roli odpor optronu při sepnutém stavu a při nabíjení velikost pull-up rezistoru. Tyto hodnoty jsme schopni ovlivnit vhodným výběrem součástek. Tímto byly zmíněny všechny podstatné závislosti, s kterými musí být počítáno při volbě součástek a může následovat samotná volba těchto součástek.

Jako kompromis byla zvolena frekvence PWM signálu 250 Hz. Optron spínající PWM signál byl zvolen KB817. V technické dokumentaci [9] je uvedena typická doba náběhu $t_r = 4\,\mu s$ a dobu pádu $t_f = 3\,\mu s$. Při frekvenci 250 Hz je doba jedné periody dána vztahem $T = \frac{1}{f}$ a je rovna hodnotě $4000\,\mu s$, což znamená že použitím tohoto optronu vzniká chyba okolo 0,1 %. Spínací charakteristika optronu je pro rychlé vybíjení GATE terminálu dostatečná.

Optron pro spínání elektromagnetického relé byl zvolen KP4010, zde byl pouze jediný požadavek a to schopnost spínat alespoň napětí o velikosti U_{zdroj}. V technické dokumentaci [10] je uvedeno, že je KP4010 schopen spínat až 300 V, což je vice než dostatečné zahrnující i vysoké špičky při rozepínání elektromagnetického relé. Pro porovnání, kdyby byl použit KP4010 i pro spínání PWM signálu, tak s maximální dobou odezvy dosahující až hodnoty $250\,\mu s$, by při přenosu PWM signálu mohlo způsobit chybu až 6,25 %. Zmíněná míra nepřesnosti je pro navrhované zařízení nepřípustná a proto byl použit již zmíněny KB817.

Jako P-kanálový MOSFET tranzistor byl zvolen IRF9520. Tranzistor je schopen spínat až 100 V, opět více než dostatečné pro účely práce. Odpor tranzistoru v sepnutém stavu (udávaný v technické dokumentaci [11] jako Drain-Source On-state Resistance) je roven $R_{MOSFET} = 0,6\,\Omega$. Doba odezvy se pohybuje v desítkách nanosekund, což je v porovnání s odezvou optronu zanedbatelná hodnota.

Pull-up rezistor nabíjející GATE terminál IRF9520 byl zvolen o velikosti $680\,\Omega$, jakožto kompromis mezi rychlostí nabíjení a přehříváním. Bude použit výkonový rezistor, který je schopen vydržet větší výkonové ztráty.

Nabíjející rezistor v DP byl zvolen o velikosti $R_{nab} = 10\,\Omega$ pro minimalizaci ztrát napětí, které na tomto rezistoru vznikají. Minimální kapacita kondenzátoru, který je součástí DP, je následně určena úpravou vztahu (4.1) pro mezní kmitočet $f_0 = 250\,\text{Hz}$.

$$C = \frac{1}{2 \cdot \pi \cdot f_0 \cdot R_{nab}} = \frac{1}{2 \cdot \pi \cdot 250 \cdot 10} = 63,6\,\mu F \tag{4.3}$$

Dolní propust s těmito hodnotami R a C působící na signál s kmitočtem 250 Hz vytváří útlum o velikosti pouze o 3 dB (nebo zesílení o velikosti -3 dB). Použitím DP s těmito parametry by stále vznikalo relativně vysoké zvlnění signálu. Proto byla zvolena vyšší kapacita 470 μF, čímž je mezní kmitočet f_0 snížen na hodnotu 34 Hz. Důsledek této úpravy je znatelně větší útlum pro signál s kmitočtem 250 Hz.

Nyní již z rovnice 4.2 stačí vyjádřit napájecí napětí U_{zdroj} a dosadit zvolené hodnoty komponent. Vybíjející rezistor R_{vyb} v DP byl zvolen o velikosti 100 Ω, jakožto kompromis mezi minimalizací potřebného napájecího napětí, přehříváním odporu, rychlosti vybíjení a výsledným zvlněním signálu.

$$U_{zdroj} = U_{mer} \cdot \frac{R_{mer}||R_{vyb} + R_{nab} + R_{MOSFET}}{R_{mer}||R_{vyb}} =$$

$$= 20 \cdot \frac{100||100 + 10 + 0,6}{100||100} = 20 \cdot \frac{\frac{100 \cdot 100}{100+100} + 10 + 0,6}{\frac{100 \cdot 100}{100+100}} = 24,24\,V$$

Tímto bylo získáno konečné napětí U_{zdroj}, které musí být schopen pro potřeby této práce step-up convertor dodávat. Je nutné dodat, že paralelním zapojením vybíjejícího rezistoru a měřené součástky, která může nabývat různých odporů (u diody například i dynamicky proměnný odpor s měnícím se napětím), vzniká nelinearita, která je dána nelineárním charakterem vybíjecí a nabíjecí křivky kondenzátoru. V konečném důsledku to znamená, že s měnící se střídou roste výstupní napětí nelineárně. Nicméně střídu a tedy i výsledné napětí lze díky dostatečně jemné (16 bitové - 65536 úrovní) přesnosti nastavovat bez problémů napříč této nelinearitě a výsledné zařízení tímto jevem není významně omezeno.

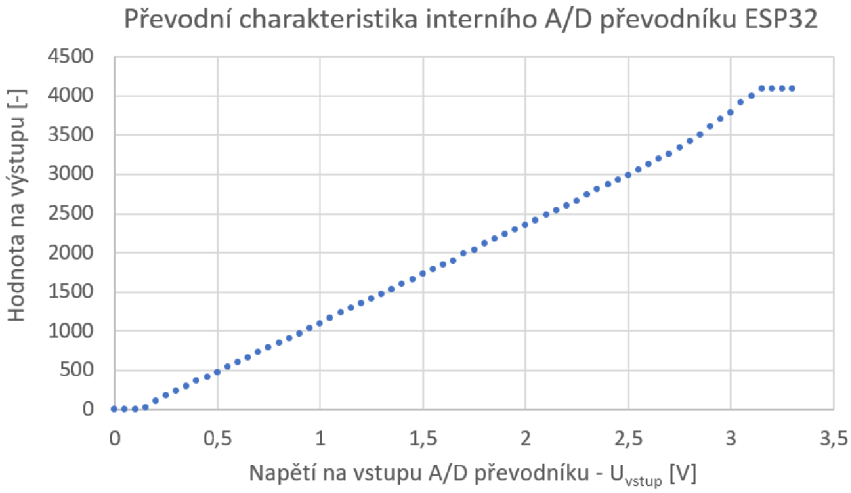
Poslední komponent, který byl představen v této kapitole je elektromagnetické relé. Bylo zvoleno RELEMP-24, které funguje jako dva řízené přepínače. Jediný požadavek na elektromagnetické relé byla schopnost spínat alespoň napájecí napětí U_{zdroj}.

4.3 Měření napětí a proudu

Jak již bylo vysvětleno v teoretickém úvodu práce, měření elektrických veličin se provádí pomocí tzv. A/D převodníku. Konkrétní způsob měření těchto veličin a výběr komponent bude popsán v následujících kapitolách.

4.3.1 Výběr A/D převodníku

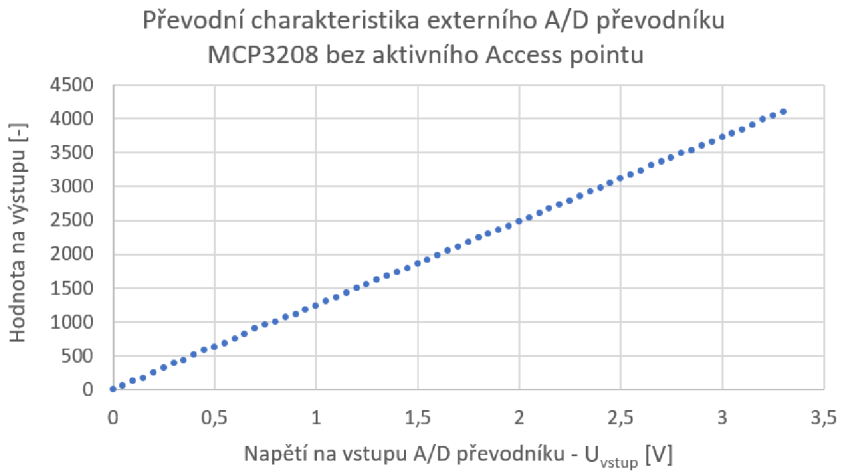
V zařízení bude využit klasický A/D převodník měřící napětí. Při výběru konkrétního typu převodníku byl jako první zvažován zabudovaný převodník v čipu ESP32. Bohužel tento převodník není dostatečně kvalitní, jelikož není v celém měřícím rozsahu lineární. Na rozsahu 3,3 V bylo naměřeno, že při napětích menších než přibližně 0,2 V převodník prakticky nedetekuje žádné napětí. V intervalu od 0,2 do 3,1 V dosahuje téměř lineárního charakteru. V poslední části charakteristiky od 3,1 V do napájecí napětí 3,3 V měří konstantně 3,3 V. Naměřená převodní charakteristika je znázorněna v grafu na obr. 4.9.



Obr. 4.9: Převodní charakteristika vnitřního A/D převodníku ESP32

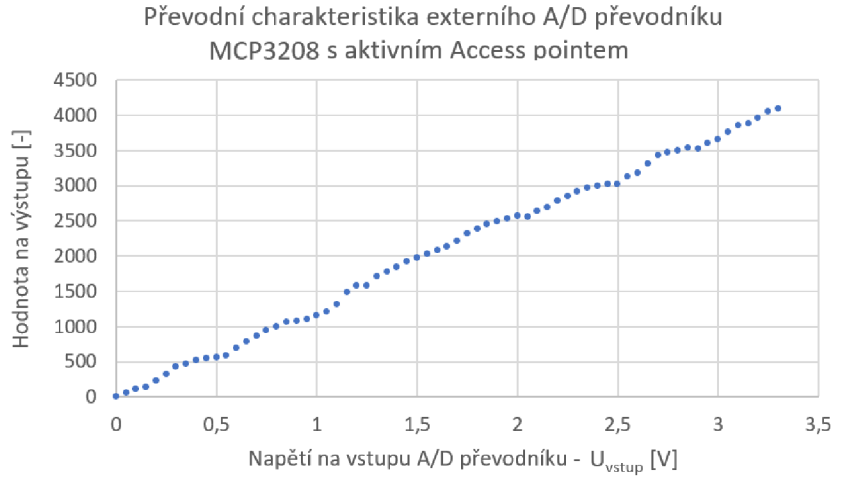
Dalším parametrem převodníku je jeho přesnost, která je 12 bitová (4096 úrovní). Tato přesnost není pro uvažovanou aplikaci omezující, ovšem v kombinaci s již zmíněnou nepřesnou převodní charakteristikou není tento převodník dostatečný pro účely zařízení. Z těchto důvodů musí být pro zařízení použit externí A/D převodník, který je lineární v celém svém měřícím rozsahu. Externí převodníky typicky komunikují se zařízením pomocí různých sběrnic. Běžnou volbou pro tyto aplikace jsou A/D převodníky používající sběrnici SPI. Byl tedy vyzkoušen 12 bitový A/D převodník MCP3208, který pro komunikaci používá právě SPI sběrnici. Převodník je podle technické dokumentace [12] schopen měřit v rozsahu od - 0,6 V do referenčního napětí, které může být nastaveno až na hodnotu 5 V. Dalším důležitým parametrem je

linearita převodní charakteristiky, která byla naměřena a je graficky zobrazena na obr. 4.10.



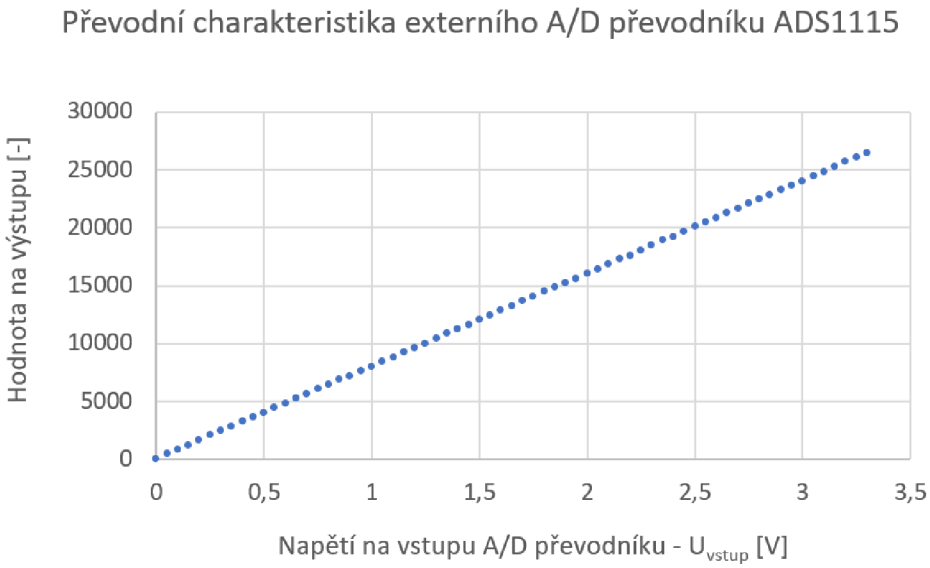
Obr. 4.10: Převodní charakteristika MCP3208

Téměř všechny parametry tohoto převodníku jsou pro navrhované zařízení dostatečné. Bohužel bylo zjištěno, že komunikace pomocí sběrnice SPI s čipem ESP32 nepodává konzistentní výsledky měření při současném chodu Wifi Access pointu. Dochází k deformaci převodní charakteristiky, respektive ke znatelnému kolísání naměřené hodnoty na aktuálně měřeném napětí. Přesná příčina nebyla zjištěna. Nejpravděpodobnější možností je nekompatibilita knihoven pro práci se sběrnicí SPI a ovládání Wi-fi rozhraní ESP32. Naměřená převodní charakteristika (přibližná, nebylo možné při zmíněném kolísání ustálit výsledky na pevných hodnotách) je graficky zobrazena na obr. 4.11.



Obr. 4.11: Převodní charakteristika MCP3208 s aktivním Access pointem

Z tohoto důvodu byla zvolena varianta s použitím sběrnice I^2C, která již tyto problémy nevykazovala. Pro zařízení byl vybrán ADS1115 A/D převodník používající sběrnici I^2C, disponující 4 vstupními kanály, s přesností 16 bitů (65536 úrovní) a s programovatelným vnitřním zesilovačem, který umožňuje měření napěťového rozsahu až +/- 6,144 V (bude ovšem využit rozsah +/- 4,096 V pro větší přesnost). Převodní charakteristika je lineární (i při současném chodu Access pointu) a je graficky zobrazena na obr. 4.12.

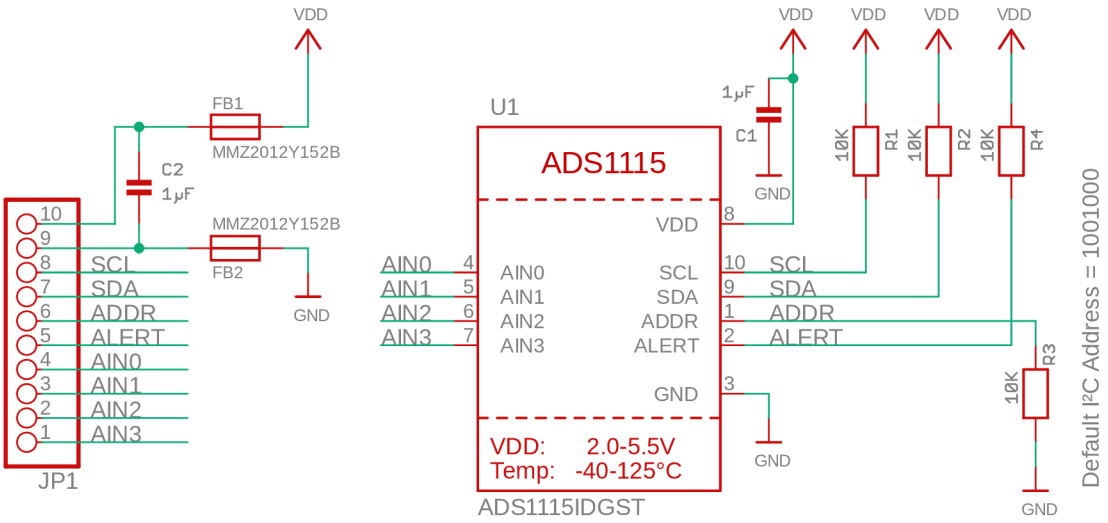


Obr. 4.12: Převodní charakteristika převodníku ADS1115

Zařízení je v dalších kapitolách navrhováno takovým způsobem, že převodník bude měřit pouze kladné hodnoty napětí, což má za důsledek, že je využita pouze polovina měřícího rozsahu, který převodník nabízí. V konečném důsledku to znamená snížení přesnosti na 15 bitů (32768 úrovní), což je stále více než dostatečné pro účely zařízení. Díky těmto parametrům převodník splňuje veškeré požadavky a je ze všech tří porovnávaných variant převodníků nejvhodnější pro navrhované zařízení [13].

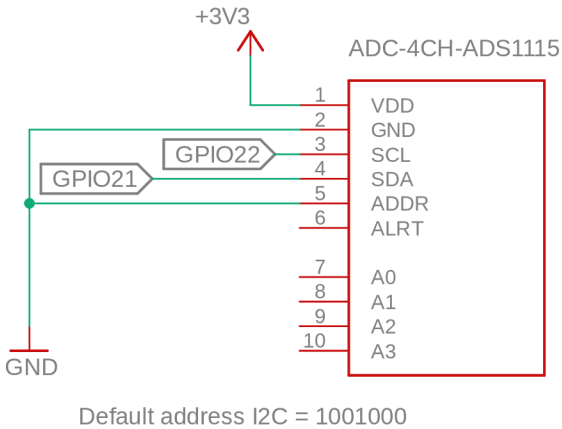
Jelikož má převodník vyvedených 10 pinů a má velice malé rozměry 3×3 mm, byla zvolena varianta připojení modulu s již vyvedenými piny pro zjednodušení pájení výsledného zařízení. Modul bude k desce plošných spojů připojen pomocí distančních šroubů. Vnitřní zapojení modulu ADS1115 je znázorněno na obr. 4.13.

Modul obsahuje tři pull-up rezistory a jeden pull-down rezistor pro zajištění jasně definované logické úrovně pinů, dva kondenzátory pro redukci případného poklesu napájecího napětí při prudkém zvýšení odebíraného proudu a dva feritové korálky (anglicky ferrite beads) pro útlum vysokofrekvenčního šumu napájení.



Obr. 4.13: Schéma zapojeni modulu ADS1115 [16]

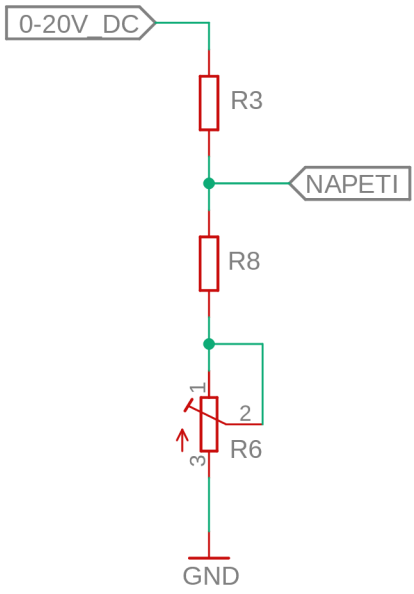
Samotné zapojení modulu k zařízení je následující. Modul komunikuje s čipem ESP32 pomocí pinů SDA (data) připojeného k pinu GPIO21 a SCL (clock) k GPIO22. Napájecí napětí převodníku ADS1115 je totožné s ESP32 a to 3,3 V, které je připojeno k pinu VDD. Zemnící pin GND a ADD jsou připojeny k zemi. Pin ADD slouží k definici I^{2}C adresy, kterou v tomto zařízení není potřebné definovat, jelikož je ADS1115 jediné zařízení, které tuto sběrnici využívá. Uzemněním tohoto pinu je definována implicitní adresa 1001000. Piny označené A0 až A3 jsou piny pro vstup měřeného napětí. Jelikož má zařízení měřit v rozsahu až 20 V, je nutné před přivedením napětí na převodník signál pomocnými obvody přizpůsobit. Návrh těchto obvodů je popsán v následujících dvou kapitolách. Zapojení A/D převodníku, popsané v této kapitole je zobrazeno na obr. 4.14.



Obr. 4.14: Připojení modulu ADS1115 k zařízení

4.3.2 Měření napětí

Měření napětí je velmi jednoduché. Pro převod napětí na bezpečnou hodnotu byl použit napěťový dělič 5:1. Je tak dosáhnuto i při přivedení maximálního napětí 20 V bezpečné napětí 3,33 V na vstupu A/D převodníku, což je bezpečné napětí pro nastavený rozsah 4,096 V. Pro jemnější nastavení byl přidán do obvodu odporový trimr. Díky této úpravě bude umožněno nastavení napěťového poměru takovým způsobem, aby byl využit co největší napěťový rozsahu A/D převodníku. (s mírnou rezervou z důvodu tolerancí součástek, zvlnění signálu apod). Hodnoty odporů pro realizaci napěťového děliče jsou voleny v řádu desítek kΩ, aby procházel co nejmenší proud a nemusel být opět přizpůsobován zdroj napětí U_{zdroj}. Zapojení obvodu pro měření napětí je znázorněno na obr. 4.15.

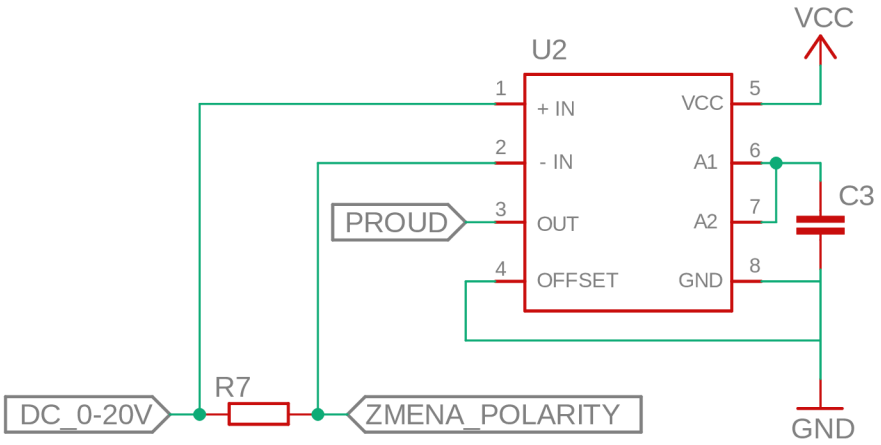


Obr. 4.15: Napěťový dělič pro měření napětí

4.3.3 Měření proudu

Měření proudu je mírně komplikovanější. Jelikož vybraný A/D převodník je schopen pracovat pouze s hodnotou napěťové úrovně a ne velikostí proudu, je zapotřebí proud vstupující do měřené součástky na napětí převést. Běžně používaný způsob je měřit napětí na součástce a pomocí znalosti jejího odporu poté určit proud pomocí ohmova zákona. Ovšem v tomto případě není odpor součástky známý (uživatel může připojit součástku o různých parametrech) a je tedy zapotřebí zvolit jiný způsob měření. Je využít tzv. shunt rezistor (R7) s velice malou hodnotou odporu (1 Ω), který je umístěn sériově před měřenou součástku, což znamená že jim prochází stejný

proud. Je zvolen velice nízký odpor pro minimalizaci zkreslení, které tento rezistor svým umístěním zavádí do měřených výsledků. Na shunt rezistoru je naměřen relativně malý úbytek napětí (v řádu milivoltů), který je pro přímé měření pomocí A/D převodníku příliš nízký. Jako řešení tohoto problému byl zvolen operační zesilovač typu current sense (detekce proudu), který je schopen generovat na svém výstupu napětí, které je úměrné proudu procházející shunt rezistorem. Toto již zesílené napětí je dále přímo přivedeno na vstup A/D převodníku a pomocí řídícího programu ESP32 přepočteno zpět na proud díky znalosti velikosti odporu shunt rezistoru a známé hodnoty zesilovací konstanty zesilovače. Zesilení bylo zvoleno tak aby při průchodu maximálního proudu 200 mA došlo k co nejefektivnějšímu využití napěťového rozsahu A/D převodníku, obdobně jako při měření napětí. Při volbě konkrétního zesilovače bylo důležité aby snesl vstupní napětí vyšší než 20 V a byl dostupný s požadovaným zesílení. Pro zařízení byl v původním návrhu zvolen MAX4080, který je schopný pracovat s vstupním napětím o velikosti až 76 V a disponuje zesílením o hodnotě 20 (Gain) [14]. Ovšem po otestování tohoto zesilovače bylo zjištěno, že není vůbec schopen zesilovat napětí menší než 4 V (v technické dokumentaci je označeno parametrem Input common-mode range 4,5 V - 76 V). Proto bylo zapotřebí zvolit zesilovač jiný a to konkrétně LMP8601, který je již schopen zesilovat napětí v rozsahu od –4 V do 27 V a rovněž je jeho zesilovací konstanta rovna 20 (Gain). Zapojení zesilovače je převzato z technické dokumentace zesilovače LMP8601 [15] (zapojení pro highside current sensing - měření proudu před zátěží) a je zachyceno na obr. 4.16. Jak měření napětí tak proudu je umístěno před obvodem měnícím polaritu, znaménko naměřené veličiny bude tedy řešeno softwarově. Jelikož je shunt rezistor zapojen až za místem, kde je měřeno napětí, je zapotřebí také softwarově provést korekci naměřeného napětí (odečíst napěťový úbytek na shunt rezistoru, vyřešeno kalibrací popsanou v kapitole 7.5.1).



Obr. 4.16: Obvod pro měření proudu

Zapojením shunt rezistoru s hodnotou 1 Ω před měřenou součástku opět vzniká napěťový dělič. Při zapojení 99 Ω rezistoru jako měřené součástky prochází proud 0,2 A při napětí 20 V. Docílíme tak teoreticky maximálního úbytku napětí na shunt rezistoru o velikosti 0,2 V (při procházejícím proudu 0,2 A) a po zesílení operačním zesilovačem konstantou 20 získáváme napětí přibližně 4 V, což je bezpečné pro nastavený rozsah převodníku 4,096 V (reálně je napětí nižší z důvodu parazitních vlastností reálných součástek).

4.4 Kompletní schéma zapojení

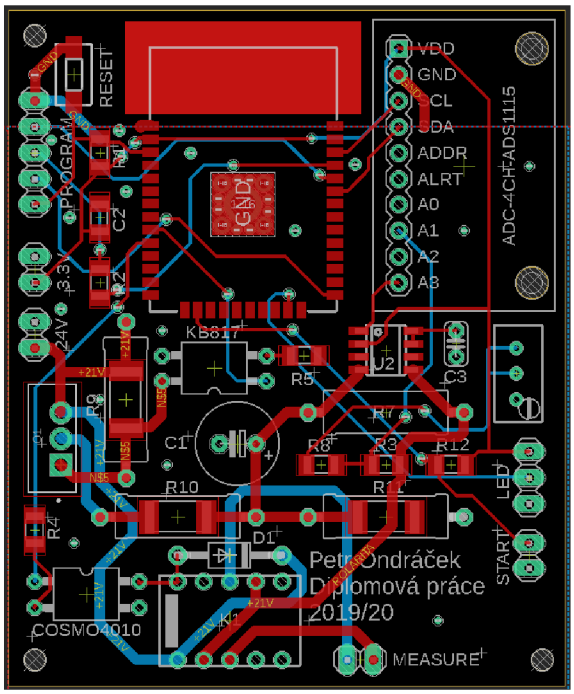
Poslední úpravy, které byly provedeny bylo přídání konektorů pro LED indikaci (ovládáné pinem GPIO15), pro tlačítko start (ovládané pinem GPIO13) a uzemnění pinu GPIO02 pro vyhnutí nedefinovaného stavu a tím i zvýšení stability zařízení. Kompletní schéma zapojení (s drobnými úpravami popsanými v kapitole 5 Návrh desky plošných spojů) je v příloze na obr. A.

5 Návrh desky plošných spojů

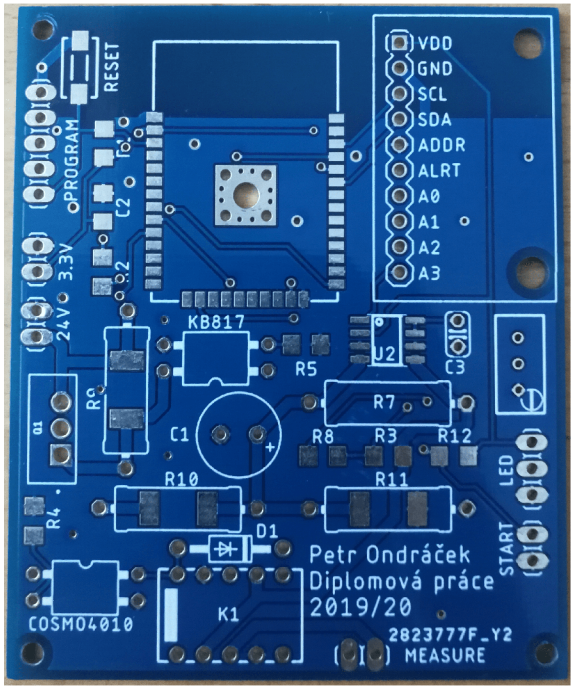
Z hotového schématu zapojení byla v editoru Eagle vytvořena deska plošných spojů (dále označována jako DPS). Při rozmísťování součástek bylo nutné umístit anténu čipu ESP32 takovým způsobem, aby nedocházelo k většímu rušení. Toho bylo docíleno umístěním antény na kraj desky a vynechání zemnící plochy v jejím okolí. Ostatní součástky byly umístěny co nejblíže u sebe pro minimalizaci plochy desky. Následně bylo provedeno vzájemné propojení součástek v souladu s navrhnutým schématem zapojení. Pro částí obvodu, kde bude ve výsledném zařízení procházet větší proud a napětí (až 200 mA a 20 V), byly zvoleny širší spoje (1,016 mm). Pro ostatní spoje byla zvolena základní šířka (0,3048 mm). Po propojení všech částí zařízení, byl zbývající prostor (kromě prostoru okolo antény ESP32) uzemněn. Tato úprava je často využívána v praxi pro kvalitnější uzemnění a efektivnější odvod tepla ze zařízení. Rezistory R9, R10 a R11 byly na desku umístěny dvakrát pomocí technologií SMT i THT. Důvod pro toto zapojení je z důvodu, že výkonové SMD rezistory z řady 2516, jsou dostupné pouze do výkonu 1 W. Ve výsledném zařízení ovšem může vzniknout výkonová ztráta až 4 W ($P = I \cdot U = 0,2\,A \cdot 20\,V$), proto byla zvolena i možnost zapojení rezistorů technologie THT pro případ, že by se zařízení při měření více přehřívalo. Po sestrojení, naprogramování a otestování zařízení bylo ovšem zjištěno, že měření netrvá více než několik desítek sekund a nedochází k většímu přehřátí zařízení. Tudíž ve je ve finálním zařízení dostatečné využít SMT variantu. Další specifická úprava v návrhu DPS bylo přidání otvorů pod zemnící plošinu GND čipu ESP32 pro zjednodušení procesu pájení čipu k desce. Poslední úpravou bylo nadefinování děr na okraji desky, pomocí kterých bude zařízení uchyceno v ochranné krabičce pomocí šroubů. Upravené schéma zapojení pro výrobu DPS je v příloze dokumentu na obr. A. Po těchto úpravách vznikl konečný návrh desky plošných spojů zachycený na obr. 5.1.

5.1 Výroba desky plošných spojů

Pomocí programu Eagle byl vyexportovaný soubor ve formátu gerber (.gbr), obsahující vrstvy potřebné pro výrobu DPS a následně byl tento soubor odeslán na výrobu do firmy JLCPCB [17]. Výsledný produkt je zachycený na obr. 5.2.



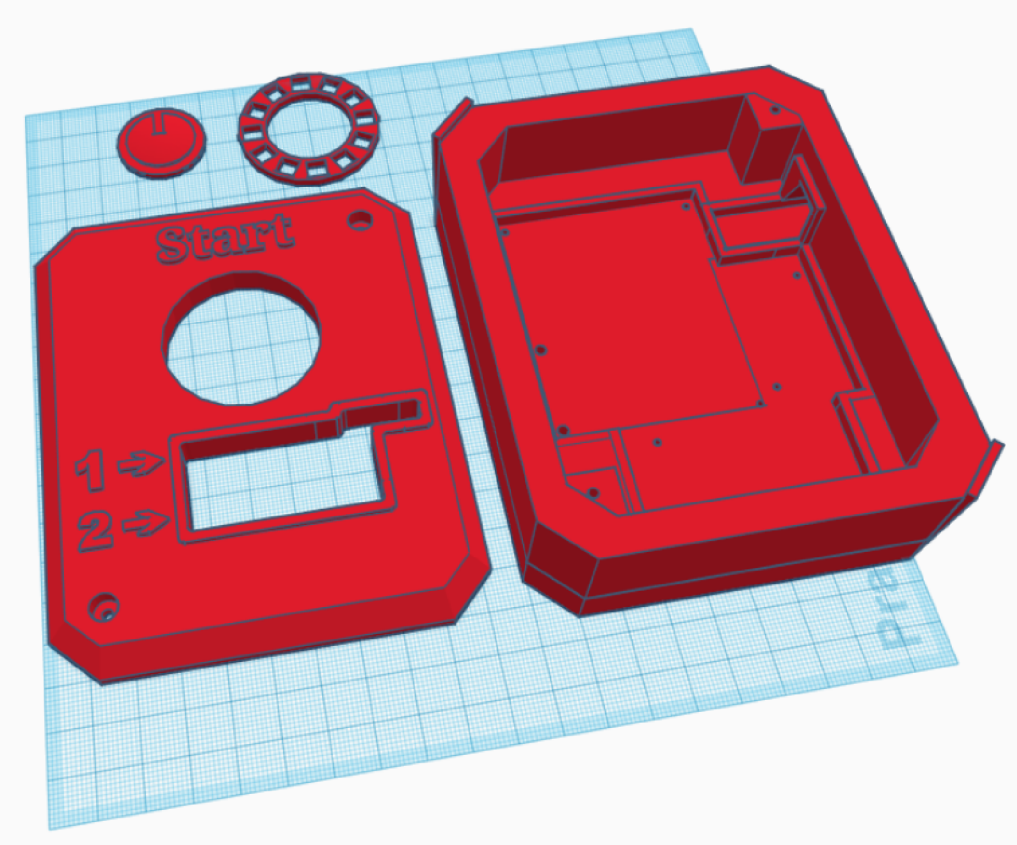
Obr. 5.1: Návrh desky plošných spojů



Obr. 5.2: Výsledná deska plošných spojů

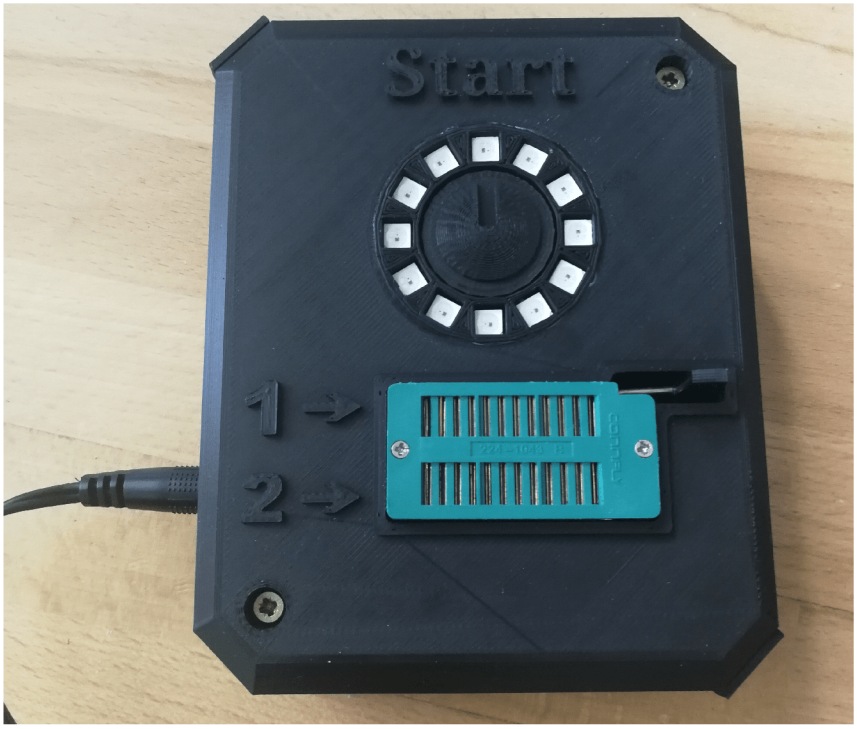
6 Sestavení zařízení a 3D-tisk krytu

Návrh 3D modelu krytu zařízení, ve kterém je zařízení umístěno byl vytvořen v bezplatném online editoru TinkerCAD a je zachycen na obrázku 6.1.

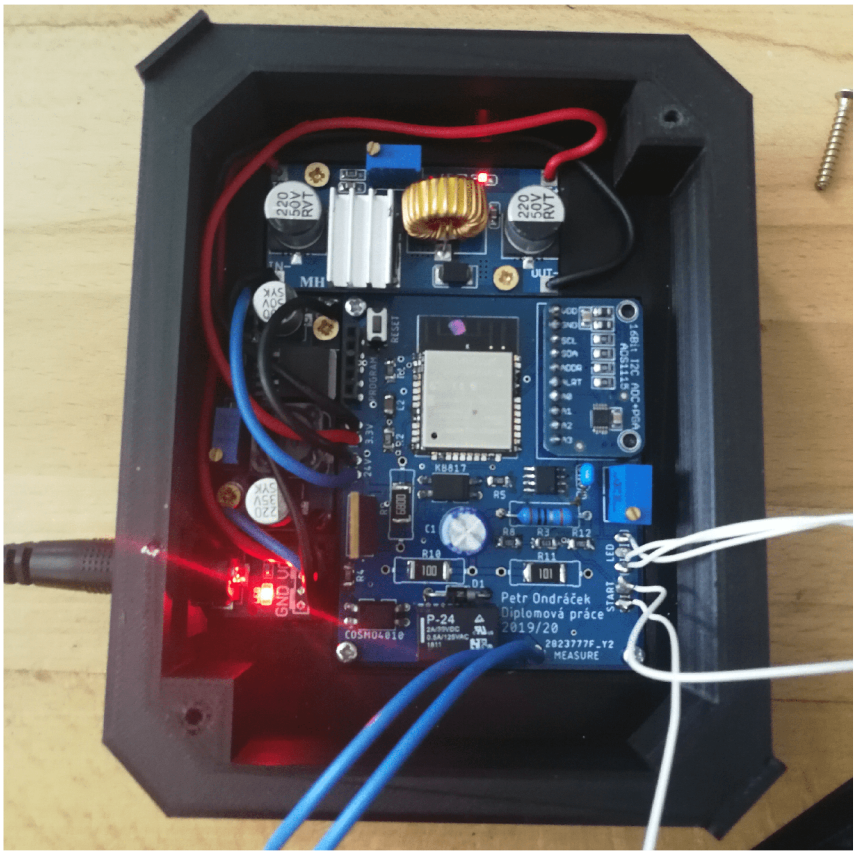


Obr. 6.1: 3D model krytu zařízení

Z tohoto editoru byl model vyexportován ve formátu .stl, který byl následně v open-source programu Cura rozdělen do jednotlivých vrstev (slicing) a převeden do formátu .gcode, který již 3D tiskárna může přímo použít pro tisk fyzického objektu. Pro tisk byla použita domácí 3D tiskárna Anycubic I3 Mega s použitím černého filamentu z materiálu PLA (polylactid acid), což je lehce rozložitelný bioplast. Po vytisknutí krytu, osazení desky plošných spojů, propojení ostatních částí zařízení (convertory, svorkovnice, LED pásek a start tlačítko) a spojení všech těchto částí dohromady bylo zařízení kompletní a je zachyceno na obrázcích 6.2 a 6.3. Tímto je hardwarová část práce kompletní a lze přestoupit ke tvorbě řídícího programu.



Obr. 6.2: Finální zařízení - zavřený kryt



Obr. 6.3: Finální zařízení - otevřený kryt

7 Programování zařízení

Tato kapitola je zaměřena na detailní popis řídícího programu zařízení. Pro vytvoření řídícího programu byl zvolen Arduino framework, který umožňuje vytvořit program pomocí vybraných funkcí a syntaxe programovacího jazyka C/C++. Část kódu, který je zasílán uživateli a slouží k zobrazení webového rozhraní je vytvořen pomocí značkovacího jazyka HTML, kaskádových stylů CSS a objektově orientovaného skriptovacího jazyku JavaScript. Vývojové prostředí Platform.io při programování automaticky překladá celý program do strojového jazyka a tento přeložený program následně nahraje do fyzické FLASH paměti čipu ESP32 přes sériovou linku. Toto spojení je vyžadováno pouze při programování zařízení a je realizováno pomocí sběrnice USB ze strany počítače a rozhraní RS232 (UART) ze strany čipu ESP32. Propojení a kompatibilitu mezi zařízeními zajišťuje USB/UART převodník, který byl již popsán v kapitole 4.1.

7.1 Knihovny

Pro vytvoření řídícího programu byly využity následující knihovny:

C/C++ knihovny:

- **Arduino.h** - knihovna obsahující informace pro správný chod programu psaném v Arduino frameworku na kompatibilních zařízeních, včetně ESP32 (vývojové prostředí Arduino IDE má tuto knihovnu zabudovanou, v Platform.io je knihovnu nutno přidat ručně)
- **WiFi.h** - knihovna umožňující práci s Wifi rozhraním čipu ESP32
- **Wire.h** - knihovna umožňující komunikaci zařízení pomocí sběrnice I^2C
- **Adafruit_ADS1015.h** - knihovna zjednodušující práci s převodníky ADS1015 a ADS1115
- (**driver/adc.h** - knihovna pro práci se zabudovaným AD převodníkem v čipu ESP32, ve finální verzi programu nebyla využita, pouze při testování, které bylo zmíněno v kapitole 4.3.1)
- **Adafruit_NeoPixel.h** - knihovna zjednodušující práci s ovládáním LED pásku

JavaScript knihovna:

- **Chart.js** - knihovna umožňující vykreslit různé typy grafů ve webovém rozhraní využitím canvas HTML elementu

7.2 Základní struktura programu

Základní struktura programu psaném v Arduino frameworku se skládá ze dvou funkcí a to konkrétně z funkcí *setup()* a *loop()*. Standardní průběh programu je následující. Při zapnutí zařízení dojde k nejprve k inicializaci všech nadefinovaných globálních proměnných a naimportování všech nadefinovaných knihoven. Dále program ihned přejde k výkonu instrukcí nacházejících se ve funkci *setup()*. Tato funkce je běžně využívána na konfiguraci a inicializaci nejrůznějších periferií a funkcí zařízení, které se mají vykonat pouze jednou na začátku programu (ovšem k podobným nastavením může docházet i kdekoliv jinde v průběhu programu). Jakmile dojde program na konec funkce *setup()* přejde ihned k výkonu funkce *loop()*, kterou začne vykonávat v nekonečné smyčce (po vykonání poslední instrukce přejde program zpět k výkonu první instrukce ve funkci *loop()*).

Výpis 7.1: Základní struktura programu

```
#include <Arduino.h>
#include <WiFi.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>
#include <driver/adc.h>
  //import použitých knihoven

int prikladGlobalniPromenne = 0;

void setup(){
  // vykonána jako první a pouze jednou
}

void loop(){
  // vykonávána ihned po funkci setup() v nekonečné smyčce
}
```

7.3 Konfigurace Wifi a webového rozhraní

Jelikož bude průběh celého měření záviset na parametrech, které určí uživatel, je zapotřebí vytvořit nějaké rozhraní, ve kterém bude moci uživatel tyto parametry nastavovat. Pro účely práce bylo zvoleno ovládání pomocí webového rozhraní, které si uživatel bude moci zobrazit například ve svém chytrém telefonu nebo počítači pomocí webového prohlížeče. Propojení se zařízením bude realizováno pomocí sítě Wifi. Zařízení bude tedy fungovat jako přístupový bod (access point) vytvářející

velice jednoduchou lokální IP síť pomocí bezdrátové technologie Wifi. Zařízení bude tedy fungovat nezávisle na okolní síti a nebude mít přístup k internetu. Zároveň bude zařízení hostovat jednoduchý webserver, který bude sloužit k zobrazení a interakci zmíněného webového rozhraní s uživatelem, který je k této nově vytvořené síti připojen.

7.3.1 Základní konfigurace

Základní konfigurace je s použitím knihovny Wifi.h velice jednoduchá. Stačí pouze nadefinovat globální proměnné reprezentující síťové parametry vytvářené sítě (objekty). Konkrétně to je *ssid* (jméno sítě), *password* (heslo pro připojení k síti), *gateway* (IP adresa brány), *local_IP* (IP adresa access pointu a zároveň i IP adresa na které bude uživateli zpřístupněno webové rozhraní), *subnet* (maska sítě) a *server* (vytvoří přímo webový server jako objekt, nutno definovat i port na kterém bude poslouchat příchozí požadavky). Dále stačí pouze v již zmíněné funkci *setup()* access point inicializovat pomocí příkazu *Wifi.softAP()* a nakonfigurovat příkazem *Wifi.softAPConfig()* s použitím nadefinovaných síťových parametrů. Nakonec je ještě nutné inicializovat server příkazem *server.begin()*.

Výpis 7.2: Základní nastavení Wifi a webového rozhraní

```
const char* ssid     = "Diplomka";
const char* password = "123456789";
WiFiServer server(80);
IPAddress local_IP(1,1,1,1);
IPAddress gateway(1,1,1,1);
IPAddress subnet(255,255,255,0);

setup(){
WiFi.softAP(ssid, password);
WiFi.softAPConfig(local_IP, gateway, subnet);
server.begin();
}
```

7.3.2 Obsluha klienta

Nyní je již jednoduchá síť i webserver vytvořen a je pouze zapotřebí ošetřit obsluhu klienta při připojení k tomuto serveru. Což v tomto případě znamená posílání HTML dokumentu pro zobrazení webové stránky klientovi a poslouchat jeho požadavky pomocí metody GET. Na základě těchto požadavku je zařízení ovládáno a zpětně modifikuje HTML dokument, který je uživateli odesílán (např. nastavené

hodnoty napětí, proudu či přímo naměřená data v grafu). Celá struktura kódu starající se o obsluhu klienta je částečně zachycena ve výpisu 7.3 s krátkými poznámkami. Pro využití zařízení nebyla vyžadována možnost připojení více zařízení zároveň a tato možnost není tedy ani kódově implementována. Kód pro obsluhu je převzat z oficiálních stránek Arduino pro obsluhu Wifi webserveru a je minimálně upravován pro zachování spolehlivé funkce komunikace mezi serverem a klientem. [18]

Výpis 7.3: Obsluha klienta

```
loop(){
WiFiClient client = server.available();  // čekání na klienta
  if (client)                     //->Pokud se klient připojí,
  { boolean requestDone = true; //vytvoří proměnnou definující
                                  //ukončení http requestu
    while (client.connected())     //->Cyklus while probíhá
    {                              //dokud je klient připojen
      if (client.available())      //->Pokud od klienta
      {  char c = client.read();  //přichází byty, načte je
         header += c;             //do proměnné header
         if (c == '\n')
         {
         if (requestDone)
           {              //->Pokud je příchozí byte newline a
                          //requestDone je true, znamená to
                          //příchod dvou newline znaků zasebou
                          //čímž je ukončen klientův HTTP
                          //request a je odeslána odpověď
      //->Je zde také vyhodnocen obsah proměnné header a v
      //závislosti na ní je klientovi odeslán HTML kód pomocí
      //funkce client.println(), který slouží k zobrazení
      //uživatelského rozhraní.
      //Podrobněji bude popsáno v další kapitole.

           } else requestDone = true;   //->Pokud je příchozí
                    //byte newline, změní requestDone na true
         } else if (c != '\r')   //->Pokud není příchozí byte
           requestDone=false;    //znak return, změní
      }                          //requestDone na false
    }
    header = "";              //->Vyprázdnění proměnné header
    client.stop();            //->Ukončení spojení s klientem
  }
```

7.4 Uživatelské rozhraní

Veškerý kód, který bude popsán v této kapitole bude vykonán pouze při úspěšném připojení a vyřízení celého HTTP požadavku klienta. Umístění kódu bylo popsáno v předchozí kapitole ve výpisu 7.3. Pro jednodušší orientaci je také celý kód řídícího programu k dispozici jako součást přílohy diplomové práce (příloha C).

7.4.1 Struktura HTML dokumentu

Nyní je zapotřebí vytvořit odpověď na klientův HTTP request. Jak bylo již dříve zmíněno, tato odpověď bude uživateli odesílat HTML dokument (obsahující HTML kód, CSS styly a javascriptový kód), pomocí kterého je na uživatelově zařízení ve webovém prohlížeči zobrazeno uživatelské rozhraní a prostřednictvím tohoto rozhraní je uživatel schopen se zařízením interagovat. HTML je značkovací jazyk pro tvoru webových stránek a struktura HTML dokumentu se skládá ze čtyř základních značek. Jsou to značky *!DOCTYPE*, *html*, *head* a *body*.

Značka *!DOCTYPE* se nachází vždy na uplném začátku html dokumentu a slouží k definici typu dokumentu (často uváděno jako DTD - Document type declaration). Pomocí značky *!DOCTYPE* lze také například přepínat mezi různými módy stylů CSS, například pro různé vykreslování stránek při použití různých internetových prohlížečů. Pro účely práce je použit pouze klasický zápis *<!DOCTYPE html>*, který přepne dokument do standardního módu HTML5 (nejnovější verze jazyku HTML).

Značka *html* slouží k ohraničení celého html souboru. Jedná se o párovou značku. První značka *<html>* se nachází ihned za značkou *!DOCTYPE* a druhá značka *</html>* se nachází na konci celého HTML dokumentu.

Značka *head* je hlavička dokumentu, která obsahuje různá metadata popisující obsah dokumentu. Opět se jedná o párovou značku zapisovanou pomocí *<head>* a *</head>*, které hlavičku dokumentu ohraničují. Jelikož se blok kódu nacházející se v hlavičce načítá dříve než blok kódu nacházející se v těle dokumentu (*body*), jedná se o vhodné a často využívané místo pro přednačtení různých elementů jako jsou skripty a definice stylů. Obsah hlavičky se na webové stránce nevykresluje.

Poslední značka *body* popisuje tělo dokumentu. Nachází se v html souboru pod hlavičkou a opět se jedná o párovou značku zapisovanou pomocí značek *<body>* a *</body>*, které tělo dokumentu ohraničují. V těle dokumentu se nachází veškeré elementy, které se zobrazí na webové stránce.[19] [20]

Bez správného použití těchto značek není zaručeno že bude zpráva obsahující HTML dokument správně interpretována na straně klienta a proto musí být vždy v

dokumentu zahrnuty. Základní struktura HTML dokumentu a tedy i základ zprávy, která je uživateli vytvořeným zařízením odesílána je zachycena ve výpisu 7.4

Výpis 7.4: Struktura HTML dokumentu

```
<!DOCTYPE html>
<html>
  <head>
  Obsah hlavičky HTML dokumentu
  </head>
  <body>
  Obsah těla HTML dokumentu
  </body>
</html>
```

Pro odeslání této zprávy klientovi pomocí navrženého zařízení byla v řídícím programu využita funkce *client.println()* z knihovny Ethernet.h (součástí Wifi.h), které jsou řádky HTML dokumentu předávány jako argument v podobě textového řetězce (string). Ukázka odeslání předešlého HTML dokumentu klientovi je zachyceno ve výpisu 7.5.

Výpis 7.5: Odeslání HTML dokumentu klientovi

```
client.println("<!DOCTYPE html>")
client.println("<html>")
  client.println("<head>")
  client.println("Obsah hlavičky HTML dokumentu")
  client.println("</head>")
  client.println("<body>")
  client.println("Obsah těla HTML dokumentu")
  client.println("</body>")
client.println("</html>")
```

Díky přechozím krokům je nýní již všechno připraveno ke správné funkci uživatelského rozhraní navrženého zařízení a můžeme se tedy pustit do vlastní tvorby podoby a funkce rozhraní.

7.4.2 Nastavení atributů měření

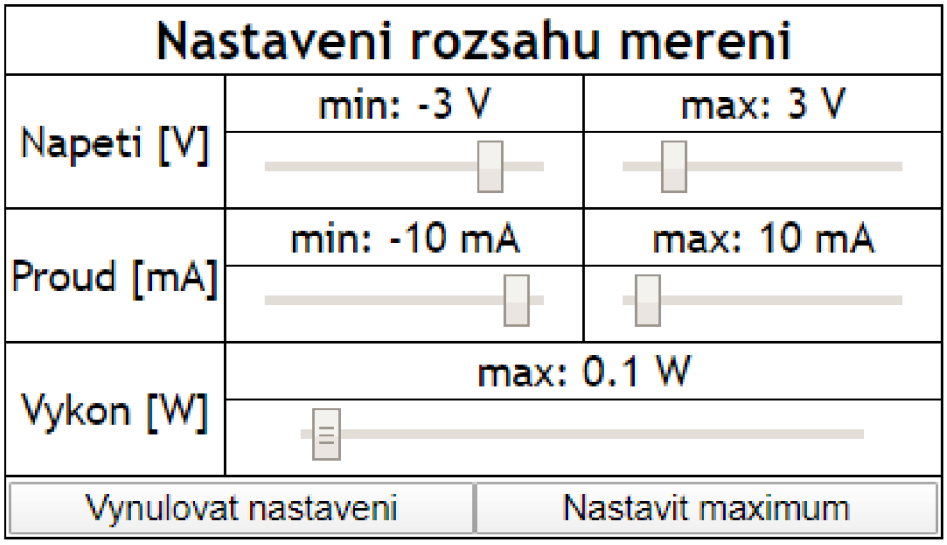
Jako první věc, kterou musí rozhraní obsahovat je možnost nastavení maximální a minimální hodnoty napětí a proudu a maximální hodnoty výkonu. Toto nastavení je realizováno pomocí posuvníků (sliderů) a to kvůli jednoduchému způsobu omezení

hodnot, které může uživatel nastavit. Například u nastavení maximálního napětí je požadováno aby šlo nastavit pouze hodnoty od 0 do 20 voltů (maximální kladný napěťový rozsah, kterého je zařízení schopno dosáhnout). Při nastavování hodnot například pomocí textového pole by mohlo dojít k přesáhnutí těchto hodnot či zadání jiných symbolů než je očekáváno a bylo by zapotřebí všechny tyto situace ošetřit. Byl tedy zvolen posuvník pro svoji jednoduchost v těchto ohledech. Po vytvoření posuvníku je zapotřebí nastavenou hodnotu uložit do proměnné. Pro získání informací od uživatele je použita HTTP metoda GET, jejíž obsah je po odeslání požadavku klientem uložen v proměnné header (popsáno v kapitole 7.3.2). Je tedy zapotřebí po nastavení hodnoty sliderem odeslat HTTP požadavek a to pomocí přesměrování na URL ve tvaru IP addresa webserveru + rozlišující řetězec + hodnota slideru + ukončující řetězec "&". V URL webové stránky by pak příklad takového požadavku vypadal následovně http:/1.1.1.1/?maxNapeti=10&. Toto je realizováno pomocí javascriptové funkce, která získá aktuálně nastavenou hodnotu slideru, poskládá z ní požadovaný tvar URL a uživatele na toto URL přesměruje čímž je i odeslán HTTP požadavek zpět k zařízení. Dále již stačí z textového řetězce v proměnné header zjístit pomocí rozlišujícího retězce o jakou proměnnou se jedná (v tomto připadě je rozlišující řetězec "maxNapeti="), nastavovanou hodnotu z řetězce header vyseparovat (v tomto případě "10") a uložit ji do proměnné pro další použití v programu. Příklad kódové realizace posuvníku je zachycen ve výpisu 7.6

Výpis 7.6: Realize nastavení atributu přes webové rozhraní

```
//globální proměnné na začátku programu
float maxNapeti=3;                //defaultní nastavení 3 V
String valueString1 = String(3); //pomocná proměnná

// ... kód po úspěšném vyřízení HTTP požadavku ...

//Získání nastavené hodnoty z odeslaného HTTP požadavku
//uloženého v proměnné header (před znovunačtením stránky)
if(header.indexOf("GET /?maxNapeti=")>=0) {
   pos1 = header.indexOf('='); //pozice začátku
   pos2 = header.indexOf('&'); //pozice konce
   valueString1 = header.substring(pos1+1, pos2);
   //vyseparování nastavené hodnoty
   maxNapeti = valueString1.toFloat();
   //převod na desetinné číslo a uložení pro další použití
   }

//... začátek odesílaného HTML dokumentu ...
//     (vše pomocí funkce client.println)
//následující kód je uvnitř <body> html dokumentu a vytváří
//posuvník, který na základě své nastavené hodnoty odešle
//HTTP požadavek ve tvaru http://1.1.1.1/?maxNapeti=hodnota&
<input type="range" min="0" max="20" step="0.1"
 value = "+valueString1+" class="slider1" id="slider1"
 onchange="mojeFunkce1(this.value)"> //vytvoření posuvníku

<script>
//získání hodnoty nastavené na posuvníku
var slider1 = document.getElementById("slider1");
slider1.oninput = function()
  {slider1.value = this.value;
  }
//funkce měnící URL po změně nastavené hodnoty na posuvníku
function mojeFunkce1(hodnota)
  {window.location.replace("http://1.1.1.1/?maxNapeti="
  +hodnota+"&");
  }
</script>
```

Tímto způsobem byly vytvořeny i další posuvníky a bylo tak vytvořeno jednoduché menu pro nastavení všech požadovaných atributů měření. Pro přehlednost byly přidány popisky a celé menu bylo zformátováno do tabulky. Výsledné menu pro nastavení, které je zobrazeno v prohlížeči je zachyceno na obrázku 7.1



Obr. 7.1: Podoba menu pro nastavení atributů měření

7.4.3 Inicializace měření a zobrazení výsledků

Samotná inicializace měření je realizována pomocí jednoduchého tlačítka a HTML tagu *href* pro přesměrování na jiné URL (není předávána proměnná hodnota a není tedy ani zapotřebí použít javascriptovou funkci jako v předchozím případě). Stejným způsobem byly realizovány i tlačítka pro vynulování a nastavení maximálních hodnot v menu nastavení. Zpracování řetězce header pak probíhá obdobně jak při nastavování atributů. Pro vynulování a nastavení maximálních hodnot sou pak pouze změněny hodnoty globálních proměnných. Implementace tlačítek je naznačena ve vypisu 7.7. Samotná implementace a popis funkce měření je obsáhlejší a je popsána v samostatné kapitole 7.5.

Výpis 7.7: Implementace tlačítek

```
//zpracování HTTP požadavku (pred načtením stránky)
if(header.indexOf("GET /FullMereni")>=0){ zahajitMereni();}
if(header.indexOf("GET /VseMax")>=0){//nastavení glob. prom.}
if(header.indexOf("GET /Nula")>=0){//nastavení glob. prom.}

//... součást <body> html dokumentu ...
//definice tlačítek
<a href="/Nula"><button class="button">Vynulovat nastaveni
                        </button></a>
<a href="/VseMax"><button class="button">Nastavit maximum
                        </button></a>
<a href="/FullMereni"><button class="button">Start mereni
                        </button></a>
```

Zobrazování výsledku měření je realizováno pomocí grafu (voltampérová charakteristika). Pro vykreslední grafu byl využit html element *<canvas>*, který slouží pouze jako obal či nádoba (container). Pro grafický výstup uvnitř tohoto elementu musí být použit JavaScript. Existuje spousta JavaScriptových knihoven pro vytváření grafů a jiných grafických výstupů. Pro účely práce byla vybrána knihovna Chart.js. Implementace JavaScriptové knihovny je v této aplikaci mírně problematické. Na běžných webových stránkách se JavaScriptové knihovny načítají dynamicky až na klientovi takovým způsobem, že server poskytne odkaz na stránku, kde si uživatel knihovnu automaticky stáhne a rovnou ji i použije pro vykreslení dané stránky. Implementace tohoto způsobu je zachyceno ve výpisu 7.8

Výpis 7.8: Implementace JavaScriptové knihovny odkazem [21]

```
//použití knihovny poskytnutím odkazu klientovi
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/
2.5.0/Chart.min.js">
//prace s knihovnou
</script>
```

Ovšem v síti vytvořené navrženým zařízením nemá uživatel přístup k internetu a tento způsob by nebyl realizovatelný. Dalším způsobem jak zpřístupnit knihovnu uživateli je mít jí uloženou jako soubor přímo na serveru a když ji bude uživatel potřebovat, jednoduše mu jí server odešle. Omezením tohoto přístupu je fakt, že server, který by danou knihovnu ve formě souboru odesílal musí sám disponovat souborovým systémem. Tím ovšem mikrokontroler ESP32 implicitně nedisponuje.

Existuje způsob jak souborový systém na ESP32 implementovat a to například pomocí knihovny SPIFFS, která umožní přístup do zabudované FLASH paměti čipu ve formě klasického souborového systému. Tento přístup je již na navrženém zařízení realizovatelný, ovšem přináší značné množství nevýhod. Jedná se například o složitější implementaci, problemy spojené s obsluhou klienta a provozem souborového systému ve stejný okamžik, větší paměťové nároky a všeobecně vznik složitější struktury řídícího programu. Z těchto důvodů byla vytvořena alternativa, která je inspirována těmito dvěma přístupy.

Javascriptová knihovna, která je získána pomocí prvního způsobu je pouze velký textový dokument, který je stáhnut z odkazu specifikovaného v atributu *src* (při otevření tohoto odkazu je přímo otevřen textový dokument s javascriptovým kódem). Následně je celý javascriptový kód vepsán do dokumentu, který klient využívá pro vykreslení webové stránky. Jelikož je knihovna pouze textový dokument, lze ji v zařízení přímo uložit jako textový řetězec a ten použít na začátku skriptu pro vykreslení grafu. Tento způsob je v porovnání s použitím SPIFFS knihovny značně jednodušší na implementaci a nepotřebuje pro svůj chod souborový systém. Při porovnání s prvním způsobem nevyžaduje připojení k internetu a je tedy na zařízení realizovatelný. Při ukládání bylo zapotřebí upravit některé speciální znaky pro správnou funkci knihovny (například změnit znaky " na \" nebo % na \%). Dále je nutno brát v potaz, že funkce *client.print()* může odeslat pouze řetězec o konečné délce (omezená velikost bufferu) a z tohoto důvodu byl textový řetězec rozdělen na čtyři menší podřetězce. Implementace JavaScriptové knihovny Chart.js, která je v zařízení použita je zachycena ve výpisu 7.9

Výpis 7.9: Alternativní implementace JavaScriptové knihovny Chart.js

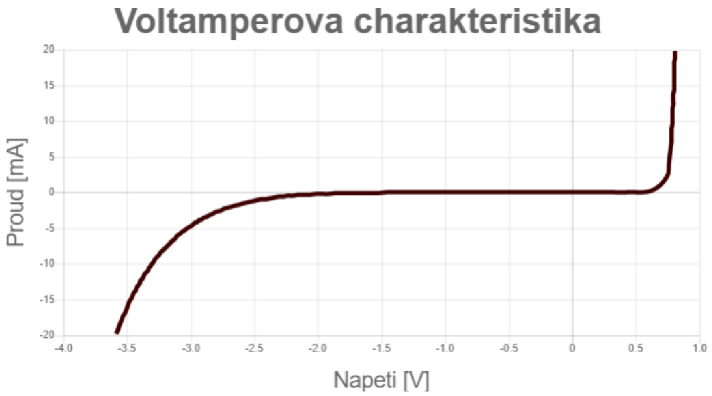
```
const char* library ="část obsahu knihovny Chart.js";
const char* library2 ="další část obsahu knihovny";
...
client.println("<script>");
client.println(library);
client.println(library2);
client.println(library3);
client.println(library4);
//prace s knihovnou
</script>
```

Po načtení knihovny lze již přejít k samotné tvorbě grafu. Vytvoření grafu pomocí knihovny Chart.js je velice jednoduché, stačí pouze vytvořit nový objekt (ve výpisu pod názvem *scatterChart*) ze třídy Chart a nalinkovat ho na HTML element canvas podle jeho ID ("MujGraf"). Dále již stačí nastavit atributy jako je například typ

grafu (*scatter* - tečkovaný graf), popisky os, velikost fontu a především samotná data. Naměřená data jsou uložena ve dvou polích *poleNapeti* a *poleProudu* (způsob naplnění těchto polí je popsán v kapitole 7.5). Způsob vytvoření grafu, nalinkování na element *canvas* a předání naměřených dat je zachycen ve výpisu 7.10 a podoba takto vytvořeného grafu (s již naměřenými hodnotami) na webovém rozhraní je zachycena na obrázku 7.2

Výpis 7.10: Vytvoření grafu pomocí knihovny Chart.js

```
//definice html elementu canvas
<canvas id="MujGraf" width=\"800\" height=\"450\"></canvas>
//použití knihovny Chart.js
var scatterChart=new Chart(document.getElementById("MujGraf"),
          {type: 'scatter', data: {
           datasets: [{
                fill:false,
                data: [{" //nameřená data...
//předání všech naměřených hodnot ve formátu
//{x: hodnota, y: hodnota,}
            static char outstr[15];
            for(int i=0;i<pocetMereni-1;i++){
              client.println("}, {");
                  client.print("x: ");
                  client.print(poleNapeti[i]);
                  client.print(",");
                  client.print("y: ");
                  client.print(poleProudu[i]);
                  client.print(",");}
```



Obr. 7.2: Ukázka grafu vytvořeného pomocí knihovny Chart.js

7.4.4 Export naměřených výsledků

Další funkce, kterou zařízení umožňuje je exportování naměřených hodnot. Bylo rozhodnoto, že pro účely práce stačí možnost exportu ve dvou formátech a to ve formatu .txt a formátu .csv (comma separated value - hodnoty oddělené čárkou). Této funkce bylo dosáhnuto opět pomocí jednoduchých tlačítek (html element *button*) a javascriptové funkce. Kódová implementace exportu naměřených výsledku ve formátu .csv je zachycena ve výpisu 7.11. Tato tato funkce pracuje takovým způsobem, že po zmáčknutí tlačítka uživatelem je zavolána javascriptová funkce, v které jsou nejdříve nadefinovány dvě proměnné. První proměnná obsahuje text, který je poskládán z naměřených hodnot (opět získány z *poleNapeti* a *poleProudu*), oddělených čárkou a znakem $\backslash n$ (endline - konec řádku). Druhá proměnná definuje nový html element, který se nebude na webové stránce zobrazovat. Tento obecný element je následně změněn na odkaz odkazující na dynamicky vytvořený soubor obsahující pouze text definovaný v proměnné text. Dále je přidán atribut umožňující stažení tohoto souboru po kliknutí na vytvořený odkaz pod jménem "Namerene-Hodnoty.csv"(parametr obsahuje i příponu souboru .csv). Následně je tento element dynamicky přidán do zobrazovaného html dokumentu, je nasimulováno kliknutí na tento element a opět je tento element dynamicky odstraněn ze zobrazovaného html dokumentu. Tímto je docíleno, že po uživatelově kliknutí na tlačítko je stáhnut soubor obsahující zformátované hodnoty, které byly zařízením naměřeny a uloženy do RAM paměti. Obdobným způsobem bylo vytvořeno i tlačítko pro stáhnutí .txt souboru a .csv s hodnotami oddělenými středníkem (z důvodu kompatibility s novější verzí programu MS Excel). [22]

Výpis 7.11: Export naměřených dat ve formátu .csv

```
<button onclick=\"downloadtxt()\"> //tlačítko

function downloadcsv1(){ // funkce volaná tlačítkem
        //formátování obsahu exportovaného souboru
        client.print("var text = \"");
        client.print("Napeti [V], Proud [mA],\\n");
         for(int i=0;i<pocetMereni-1;i++){
           client.print(poleNapeti[i]);
           client.print(",");
           client.print(poleProudu[i]);
           client.print(",\\n");}
        client.println("\";");

//vytvoření obecného html elementu
  var element = document.createElement('a');
//element není na stránce zobrazen
  element.style.display = 'none';
//změna atributu elementu na odkaz na soubor obsahující
//text obsažený v proměnné text + definice formátu
  element.setAttribute('href','data:text/plain;charset=utf-8,'
  + encodeURIComponent(text));
//přidání atributu pro možnost stažení souboru
  element.setAttribute('download',\"NamereneHodnoty.csv\");
//dynamické přidání elementu do html dokumentu
  document.body.appendChild(element);
  element.click();");   //simulace kliknutí na odkaz
//dynamické oddělání elementu z html dokumentu
  document.body.removeChild(element);
}
```

7.5 Implementace měření napětí a proudu

Poslední chybějící část programu je samotná implementace proměření uživatelem připojené součástky. O tuto činnost se stará funkce *zahajitMereni()*, která je volána programem po té co uživatel nastaví parametry měření a stiskne tlačítko pro zahájení měření (již popsáno v kapitole 7.4.3). Kódová implementace celé funkce je příliš obsáhla pro ukázku pomocí výpisu a byl tedy alespoň vytvořen blokový diagram zachycující průběh celé funkce, který se nachází v příloze dokumentu B. Funkce se stará o to aby byla součástka na základě uživatelem zadaných parametrů proměřena

v daných napěťových a proudových intervalech a naměřené hodnoty byly uloženy do paměti pro použití při vykreslování grafu na webovém rozhraní a stahování (také již bylo popsáno v předchozích kapitolách). Vstupní proměnné jsou tedy *maxNapeti*, *maxProud*, *maxVykon*, *minNapeti* a *minProud* (nastavené uživatelem) a výstupní proměnné jsou *poleNapeti*, *poleProudu* a *pocetMereni*. Význam těchto proměnných lze z jejich pojmenování lehce pochopit. Před vlastním měřením musí být AD převodník zapnut a nastaven pro měření napěťového rozsahu 4,096 V (viz kapitola 4.3). V řídícím programu, který je součástí přílohy práce, je toto provedeno při zapnutí zařízení ve funkci *setup()* pomocí příkazů *ads.begin()* a *ads.setGain(GAIN_ONE)*. Dále je zapotřebí vyčištit pamět od předchozího měření (vynulování *poleNapeti* a *poleProudu*). Následně je přepnuto magnetické relé do pozice pro měření kladné části voltampérové charakteristiky, je zvolena nejmenší velikost kroku pro zvyšování hodnoty střídy PWM signálu (*stepsize = 1*) a je zahájena první iterace měření. Každá iterace měření probíhá takovým způsobem, že je nejdříve na pinu GPIO33 aktivován PWM signál se střídou rovnou součtu předchozí nastavené hodnoty PWM a aktualně nastavené hodnoty *stepsize* (v první iteraci rovno jedné). Hodnota střídy D_n (dutycycle) v n-té iteraci je pak dána výrazem 7.1.

$$D_{n[\%]} = \frac{PWM_n}{65535} \cdot 100 = \frac{PWM_{n-1} + stepsize}{65535} \cdot 100 \tag{7.1}$$

Dále je inkrementován počet měření o jedničku a zařízení začne číst hodnoty z AD převodníku. Načtená hodnota N_U z AD převodníku je úměrná hodnotě měřeného napětí U a musí být na toto napětí přepočítána pomocí vzorce 7.2.

$$U = N_U \cdot k_1 \tag{7.2}$$

Analogicky platí to stejné pro naměřenou hodnotu N_I při měření proudu. Plati vzorec 7.3

$$I = N_I \cdot k_2 \tag{7.3}$$

Konstanty k_1 a k_2 pro přepočet hodnot naměřených AD převodníkem na hodnoty napětí a proudu byly určeny při kalibraci zařízení, která je popsána v následující kapitole 7.5.1. Po převodu jsou hodnoty uloženy do proměnných *poleNapeti* a *poleProudu* a v každé iteraci je kontrolováno zda nebyly překročeny nastavené hodnoty *maxNapeti*, *maxProudu* a *maxVykon*. Pokud ano, iterace je ukončena a následuje měření záporné části voltampérové charakteristiky. Pokud ne, je zvýšena velikost střídy PWM signálu o hodnotu *stepsize* a pokračuje další iterace měření. Velikost

hodnoty *stepsize* je během měření přizpůsobována pro zmenšení celkového počtu měřených hodnot a tedy i doby měření. Zároveň je také dbáno na zachování správného tvaru výsledné charakteristiky (aby například v ohybu voltampérové charakteristiky diody nebylo naměřeno jen několik hodnot a nedošlo tak k větší deformaci naměřené křivky). Toto přizpůsobení je realizováno na základě sledování velikosti změny proudu mezi dvěmi po sobě naměřenými hodnotami. Pokud je změna příliš velká (nastaveno při změně větší než 1 mA), dojde k detekci této rychlé změny v průběhu voltampérové charakteristiky, je zmenšena hodnota *stepsize* a tato část charakteristiky je pak s dostatečnou přesností proměřena. Implementace tohoto přizpůsobení není nezbytné pro celkovou funkci zařízení. Stačilo by celé měření realizovat s konstantním krokem *stepsize* (pro nejpřesnější měření by byl krok *stepsize* roven jedné). Ovšem při takovém měření by zařízení na plném napěťovém rozsahu (-20 až +20 voltů) muselo uskutečnit 131072 počet měření ($2 \cdot 2^{16}$, což odpovídá plnému rozsahu 16 bitového PWM signálu generovaného čipem ESP32 pro obě polarity měření). Takové měření by trvalo řádově několik minut a i vykreslení takto velkého množství hodnot na webové rozhraní je problematické, jelikož odesílání hodnot a zobrazení grafu trvá také velmi dlouho, opět v řádu jednotek minut a může dojít i k automatickému ukončení spojení (timeout). Tento problém by se dal vyřešit zvětšením velikosti hodnoty kroku (*stepsize*) například na 100. Ovšem v tomto případě by kvůli velkému neměnnému kroku nemusela být charakteristika proměřena ve všech místech s dostatečným detailem. Z těchto důvodů byla vytvořena již zmíněná jednoduchá detekce nárůstu změny proudu a přizpůsobení velikosti kroku *stepsize*. Při testování podávala tato metoda dostatečně přesné výsledky při krátké době měření (jednotky až desítky sekund, závislé na nastaveném napěťovém a proudovém rozsahu) a nebyl tedy navrhován složitější algoritmus. Implementace přizpůsobení a zrychlení měření slouží pouze pro zpříjemnění práce se zařízením. Časová náročnost měření nebyla přímo brána jako faktor v zadání diplomové práce.

Po proměření kladné části voltampérové charakteristky dojde k vynulování střídy PWM signálu a přepnutí magnetického relé, čímž dojde k přepojení pinů měřené součástky. Dále je uskutečněno stejné měření pro naměření záporné části voltampérové charakteristiky. Ovšem při těchto iteracích jsou využity namísto parametrů *maxNapeti* a *maxProud* parametry *minNapeti* a *minProud*. Hodnota *maxVykon* je využita i v záporné části charakteristiky, neboť platí vztah $P = U \cdot I = (-U) \cdot (-I)$. Po proměření celé voltampérové charakteristky je znovu načteno webové rozhraní s již vykreslenou charakteristikou a je také uživateli umožněno naměřená data stáhnout. Kódová implementace celého řídícího programu je součástí přílohy diplomové práce.

7.5.1 Kalibrace zařízení

Jak již bylo zmíněno, převodní konstanty k_1 a k_2 slouží k převodu hodnot naměřených AD převodníkm na realné hodnoty napětí a proudu, které prochází připojenou součástkou. Proto musí být pro zajištění přesnosti tyto konstanty správně nastaveny (zkalibrovány). Pro tyto účely byl vypůjčen velmi přesný referenční zdroj (kalibrátor) Precision Calibrator Type 5395B od firmy Kistler, který je zachycen na obr. 7.3.

Obr. 7.3: Precision Calibrator Type 5395B od firmy Kistler

V technické dokumentaci kalibrátoru je pro výstupní referenční napětí uváděna nejistota měření menší než 0,01 % [23]. Pomocí kalibrátoru bylo tedy vygenerováno přesné napětí 1 V, které bylo přivedeno na svorky, kde je při běžném měření připevněna měřená součástka a následně byla na AD převodníku naměřena hodnota N_U = 1411. Úpravou výrazu 7.2 lze dopočítat převodní konstantu napětí k_1.

$$k_1 = \frac{U}{N_U} = \frac{1}{1411}$$

Při kalibraci konstanty k_1 byl odpojen step-up konvertor, aby nedošlo k zkreslení naměřených výsledků. Pro určení převodní konstanty proudu k_2 byl opět step-up konvertor připojen, byl také připojen rezistor (s hodnotou $R = 100\ \Omega$) jako měřená součástka a bylo vygenerováno napětí 1 V (nyní bez použití kalibrátoru, ale pomocí step-up konvertoru a zjištěné konstanty k_1). Pomocí AD převodníku byla na druhém kanálu naměřena hodnota $N_I = 1369$ odpovídající hodnotě proudu procházejícího zapojeným rezistorem. Pomocí ohmova zákona byl dopočítán proud procházející

rezistorem a opět po úpravě výrazu 7.3 byla dopočítána převodní konstanta proudu k_2 (upravena pro zobrazení v jednotkách mA).

$$I = \frac{U}{R} = \frac{1}{100} = 0,01A = 10mA$$

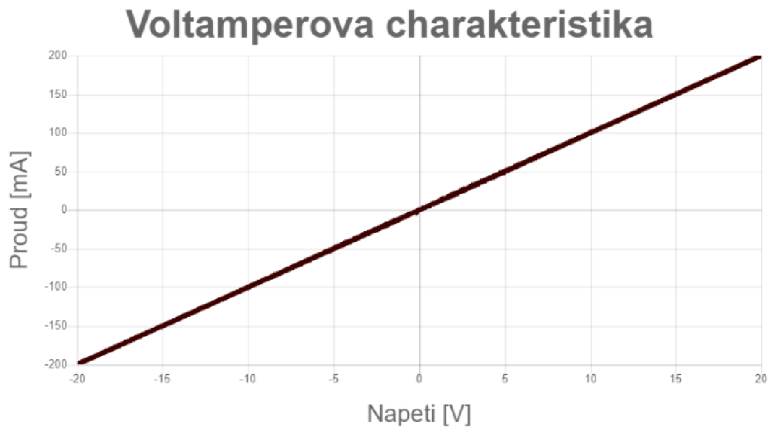
$$k_2 = \frac{I_{[mA]}}{N_I} = \frac{10}{1369}$$

Hodnoty konstant k_1 a k_2 jsou implementovány v řídícím programu zařízení pod názvy *prevodKonstNapeti* a *prevodKonstProud*.

8 Testování zařízení

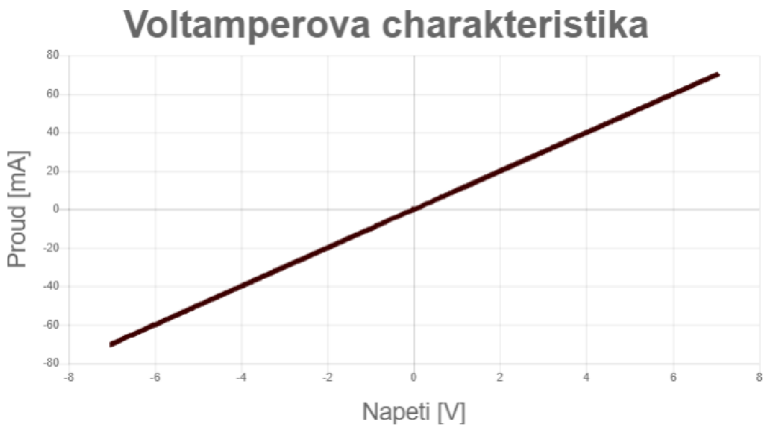
V rámci diplomové práce bylo požadováno otestovat vytvořené zařízení proměřením několika elektrických komponent (různé typy diod). Naměřené voltampérové charakteristiky těchto komponent jsou zobrazeny na obrázcích níže.

- Rezistor 100 Ω - voltampérová charakteristika je pro rezistor přímka a byla naměřena na plném napěťovém (-20 V až +20 V) i proudovém (-200 mA až 200 mA) rozsahu. Tím bylo otestováno splnění požadavků na maximální rozsahy měření.



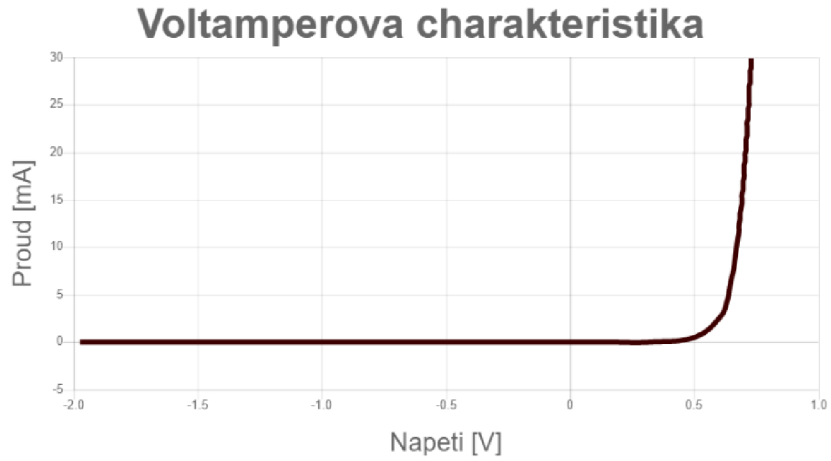
Obr. 8.1: Rezistor 100 Ω

- Rezistor 100 Ω - rezistor byl znovu proměřen i pro demonstraci nastavení výkonového omezení, zvolena hodnota 0,5 W (měření automaticky ukončeno na přibližně 7,07 V a 70,7 mA)



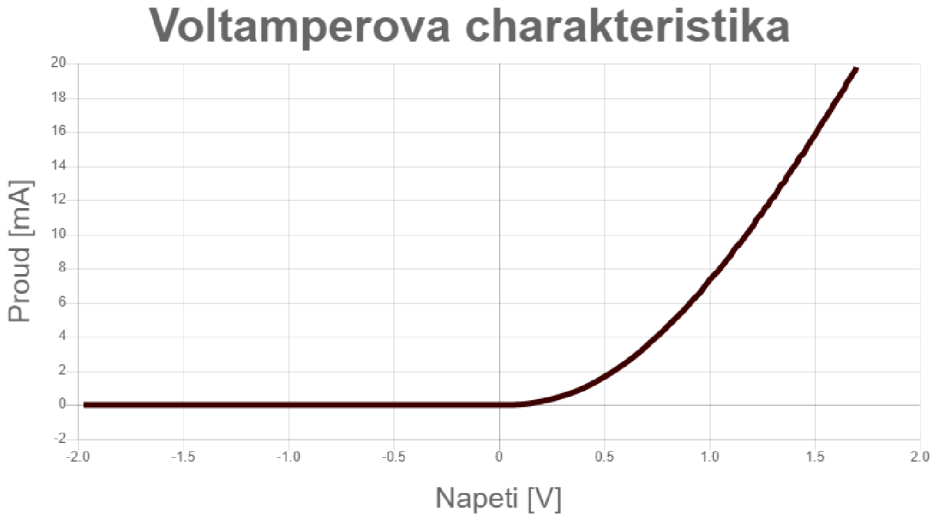
Obr. 8.2: Demonstrace výkonového omezení

- Křemíková dioda - prahové napětí ~0,7 V (nastaveno proudové omezení -30 a +30 mA, napěťové omezení -2 a +2 V, výkonové omezení 0,1 W)



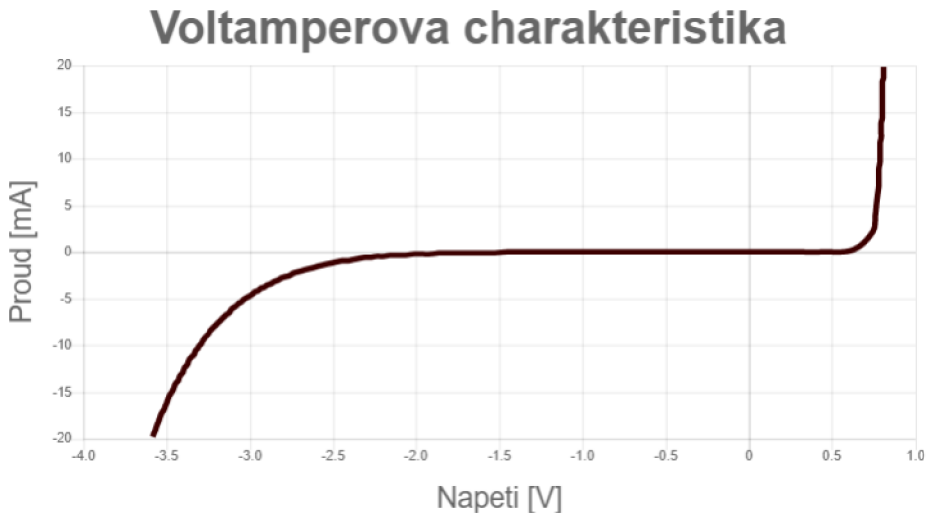
Obr. 8.3: Křemíková dioda

- Germaniová dioda - prahové napětí ~0,3 V (nastaveno proudové omezení -20 a +20 mA, napěťové omezení -2 a +2 V, výkonové omezení 0,1 W)



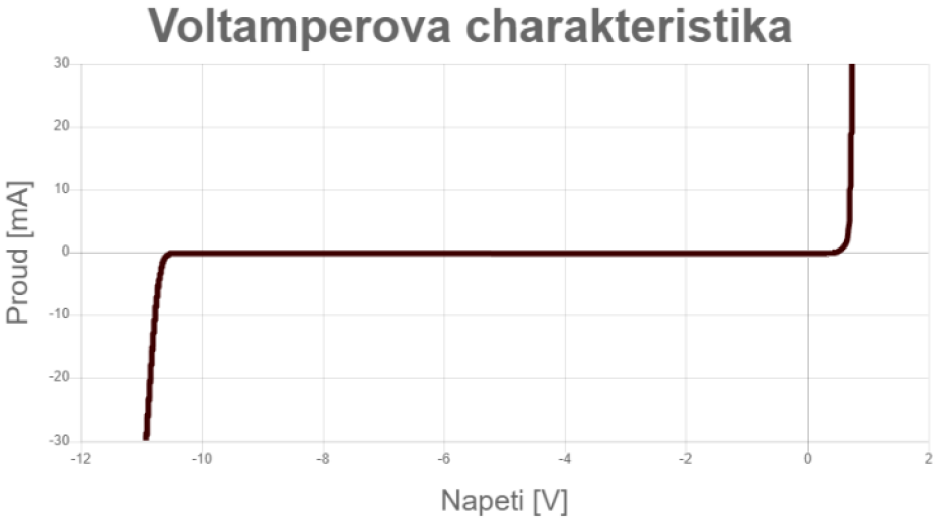
Obr. 8.4: Germaniová dioda

- Zenerova dioda 3,3 V - zenerovo napětí ~3,3 V v závěrném směru, prahové napětí v propůstném směru je klasických ~0,7 V - křemík (nastaveno proudové omezení -20 a +20 mA, napěťové omezení -5 a +5 V, výkonové omezení 0,2 W)



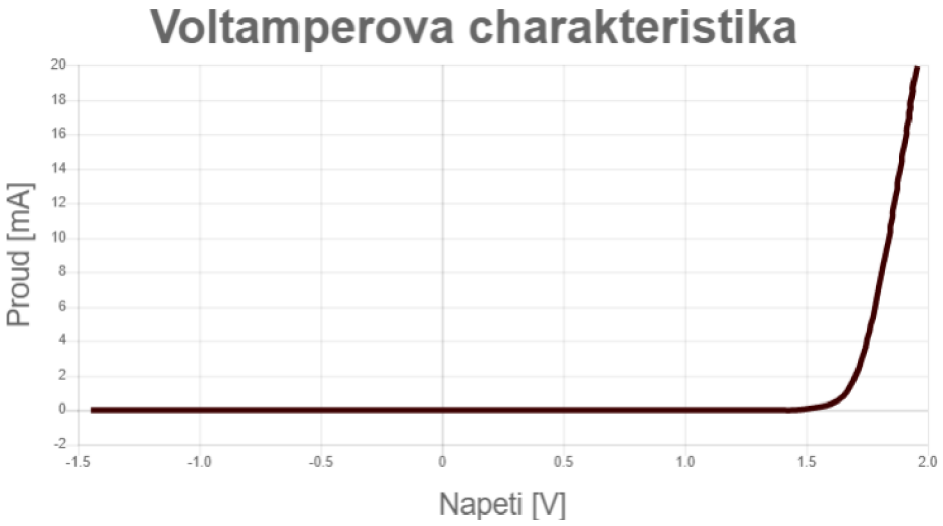
Obr. 8.5: Zenerova dioda 3,3 V

- Zenerova dioda 11 V - zenerovo napětí ~11 V v závěrném směru, prahové napětí v propůstném směru opět ~0,7 V - křemík (nastaveno proudové omezení -30 a +30 mA, napěťové omezení -15 a +15 V, výkonové omezení 0,5 W)

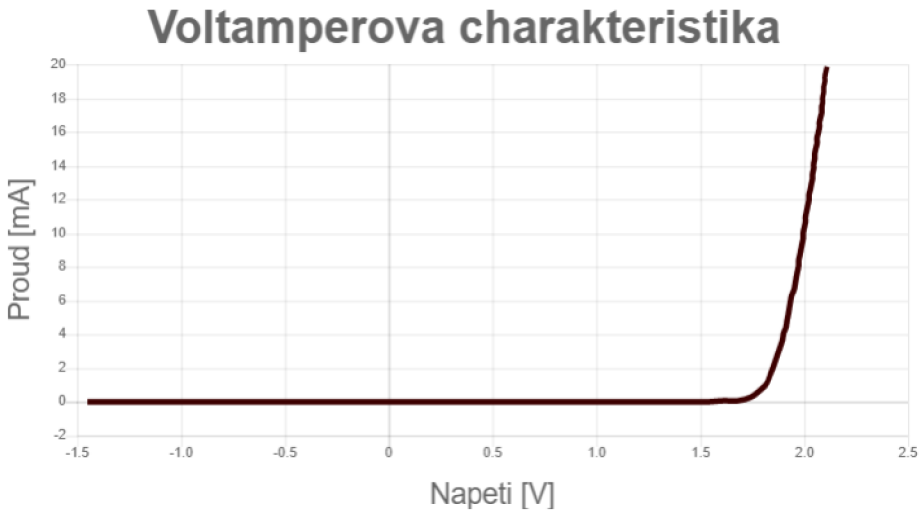


Obr. 8.6: Zenerova dioda 11V

- LED diody - Červená a zelená barva pro porovnání (nastaveno proudové omezení -20 a +20 mA, napěťové omezení -1,5 a +2,5 V, výkonové omezení 0,1 W)

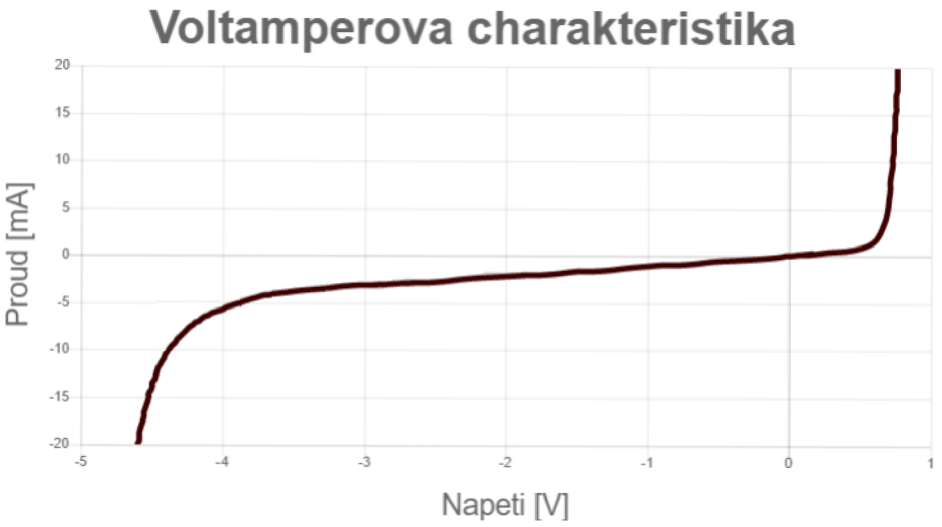


Obr. 8.7: Červená LED



Obr. 8.8: Zelená LED

- C-E přechod bipolárního NPN tranzistoru 2N3904 - lavinový průraz v závěrném směru (nastaveno proudové omezení -20 a +20 mA, napěťové omezení -10 a +10 V, výkonové omezení 0,2 W)



Obr. 8.9: Lavinový průraz bipolárního NPN tranzistoru 2N3904

9 Závěr

V rámci diplomové práce byl vysvětlen základní princip měření elektrického napětí a proudu pomocí digitálního zařízení, kde byl vysvětlen i princip analogově digitálního převodníku. V práci byl představen čip ESP32, byly popsány jeho vlastnosti a parametry a také byly uvedeny některé moduly, které se dají dnes zakoupit. Dále se práce zabývá návrhem vlastního zařízení. Nejdříve je funkce celého zařízení jednoduchým způsobem vysvětlena na blokovém schématu. Následují kapitoly, které jsou zaměřené na postupný návrh schématu zapojení, kde jsou již vybrány konkrétní součástky s odůvodněním jejich výběru a případně jsou i porovnány s jinymi alternativami například v případě analogově digitálního převodníku. Dále je v práci popsán proces vytváření návrhu desky plošných spojů a její realizace. Rovněž je navrhnut ochranný kryt zařízení, který byl vytisknut na 3D tiskárně. Po sestavění zařízení byl vytvořen návrh řídícího programu, který byl do zařízení nahrán. Řídící program umožňuje zařízení vytvořit jednoduchou Wifi síť, do které se uživatel připojuje, hostování webserveru vytvářející uživatelské rozhraní, na kterém je uživatel schopen konfigurovat parametry měření, zařízení ovládat, zobrazit naměřené výsledky a je zde také naimplementováno samotné ovládání periférií čipu pro proměření součástky připojené uživatelem. Při vytváření řídícího programu je také provedena jednoduchá kalibrace převodních konstant pro zajištění přesných naměřených výsledků. Na závěr práce je zařízení otestováno na několika komponentech čímž je zkontrolováno splnění zadání diplomové práce.

Literatura

[1] Portál Espressif, *ESP-WROOM-32 datasheet* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf>.

[2] Portál Espressif, *ESP32 Modules and Boards* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://docs.espressif.com/projects/esp-idf/en/latest/hw-reference/modules-and-boards.html>.

[3] Portál Espressif, *ESP-WROVER-KIT V4.1 Getting Started Guide* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://docs.espressif.com/projects/esp-idf/en/latest/hw-reference/get-started-wrover-kit.html>.

[4] Portál Arduino, *FAQ: Arduino Software* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://www.arduino.cc/en/main/FAQ>.

[5] Portál esp32.net, *ESP32 Features and specifications* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <http://esp32.net/>.

[6] Portál Espressif, *ESP32 Devkit v4 schematic* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://dl.espressif.com/dl/schematics/esp32_devkitc_v4-sch.pdf>.

[7] Portál alldatasheet.com, *Step-down convertor XL4015 datasheet* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/1134361/XLSEMI/XL4015.html>.

[8] Portál alldatasheet.com, *Step-up convertor XL6009 datasheet* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/1132228/XLSEMI/XL6009.html>.

[9] Portál alldatasheet.com, *KB817 datasheet* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/113354/KINGBRIGHT/KB817.html>.

[10] Portál cosmo-ic.com, *KP4010 datasheet* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <http://www.cosmo-ic.com/object/products/KP4010.pdf>.

[11] Portál alldatasheet.com, *IRF9520 datasheet* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/923970/VISHAY/IRF9520.html>.

[12] Portál alldatasheet.com, *MCP3208 datasheet* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/74937/MICROCHIP/MCP3208.html>.

[13] Portál alldatasheet.com, *ADS1115 datasheet* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/292735/TI/ADS1115.html>.

[14] Portál farnell.com, *MAX4080 datasheet* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <http://www.farnell.com/datasheets/2000918.pdf?_ga=2.195166378.708340872.1575319088-214445790.1572817293>.

[15] Portál alldatasheet.com, *LMP8601 datasheet* [online]. [cit. 9. 5. 2020]. Dostupné z URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/841710/TI1/LMP8601.html>.

[16] Portál adafruit.com, *ADS1115 schematic* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://learn.adafruit.com/adafruit-4-channel-adc-breakouts/downloads>.

[17] Portál jlcpcb.com, *PCB manufacturer* [online]. [cit. 5. 12. 2019]. Dostupné z URL: <https://jlcpcb.com/>.

[18] Portál Arduino, *WiFi Web Server* [online]. [cit. 1. 4. 2020]. Dostupné z URL: <https://www.arduino.cc/en/Tutorial/WiFiWebServer>.

[19] Portál htmlquick.com, *An html document structure* [online]. [cit. 9. 5. 2020]. Dostupné z URL: <https://www.htmlquick.com/tutorials/an-html-document-structure/2.html>.

[20] Portál jakpsatweb.cz, *Struktura dokumentu HTML* [online]. [cit. 9. 5. 2020]. Dostupné z URL: <https://www.jakpsatweb.cz/html/struktura.html>.

[21] Portál cdnjs.cloudflare.com, *Chart.js library* [online]. [cit. 9. 5. 2020]. Dostupné z URL: <https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.5.0/Chart.min.js>.

[22] Portál ourcodeworld.com, *How to create a file and generate a download with javascript in the browser without a server* [online]. [cit. 1. 4 2020]. Dostupné z URL: <https://ourcodeworld.com/articles/read/189/how-to-create-a-file-and-generate-a-download-with-javascript-in-the-browser-without-a-server>.

[23] Portál pei-france.com, *Kistler Calibrator 5395B datasheet* [online]. [cit. 13. 5 2020]. Dostupné z URL: <https://www.pei-france.com/uploads/tx_etim/27016_PEI3425_Kistler_DS_calibrateur_5395B.pdf>.

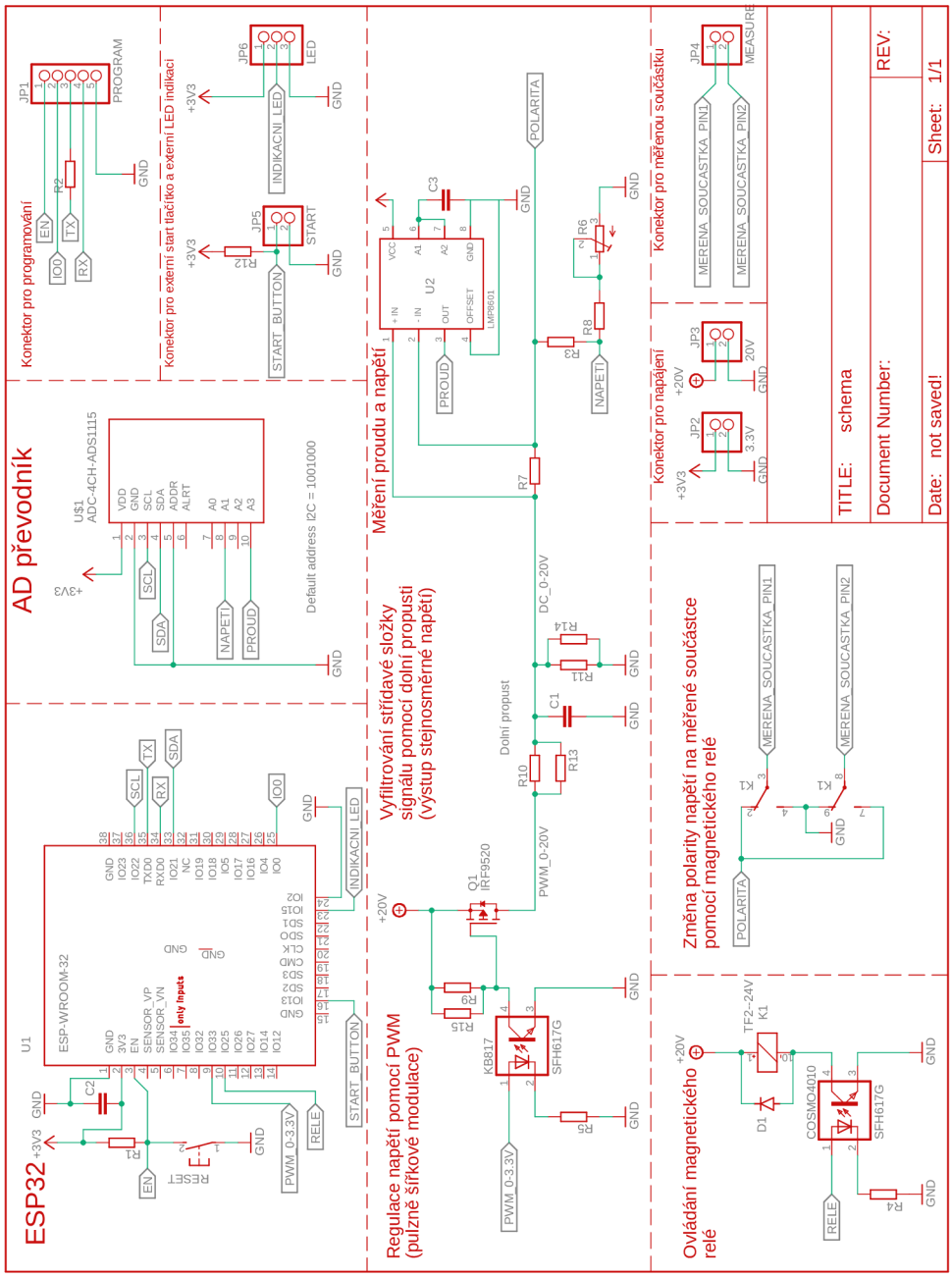
Seznam symbolů, veličin a zkratek

AP	Access Point, přístupový bod
A/D	Analog-digital (převodník)
BLE	Bluetooth Low Energy, Bluetooth s nízkou spotřebou energie
CSS	Cascading Style Sheets, Kaskádové styly
C-E	Přechod tranzistoru kolektor-emitor
DP	Dolní propust
DPS	Deska plošných spojů
DTD	Document Type Definition, definice typu dokumentu
f	Frekvence, jednotka Hertz
GND	Ground, zemnící pin
GPIO	General-purpose Input/Output, Univerzální vstupní/výstupní pin
HTML	Hypertext Markup Language, značkovací jazyk pro tvorbu internetových stránek
HTTP	Hypertext Transfer Protocol, protokol pro přenost HTML dokumentů
I	Elektrický proud, jednotka Ampér
IEEE 802.11	Standard pro lokální bezdrátové sítě od institutu pro elektrotechnické a elektronické inženýrství
IoT	Internet of Things, Internet věcí
IP	Internet Protocol, internetový protokol
I^2C	Inter-Integrated Circuit
JTAG	Joint Test Action Group, standard a architektura pro testování zařízení
LCD	Liquid crystal display, displej z tekutých krystalů
LED	Light Emitting Diode, světelná dioda
MOSFET	Metal Oxide Semiconductor Field Effect Transistor, polem řízený tranzistor se strukturou kov(M)-oxid(O)-polovodič(S)
MS	Microsoft
P	Elektrický výkon, jednotka Watt
PLA	Polylactid acid
PSRAM	Pseudo-Static Random Access Memory, Pseudostatická paměť s náhodným přístupem
PWM	Pulse Width Modulation, pulzně šířková modulace
R	Elektrický odpor, jednotka Ohm
RAM	Random Access Memory, Paměť s náhodným přístupem
ROM	Read Only Memory, Paměť pouze pro čtení
RTC	Real Time Clock, Hodiny reálného času

SMD	Surface Mount Device, zařízení pro povrchovou montáž
SMT	Surface Mount Technology, technologie pro povrchovou montáž
SPI	Serial Peripheral Interface, sériové periferní rozhraní
SPIFFS	Souborový systém
SRAM	Static Random Access Memory, Statická paměť s náhodným přístupem
THT	Through Hole technology, montáž součástek s drátovými vývody
U	Elektrické napětí, jednotka Volt
UART	Universal asynchronous receiver-transmitter, univerzální asynchronní přijímač-vysílač
URL	Uniform Resource Locator, jednotná adresa zdroje
USB	Universal Serial Bus, univerzální sériová sběrnice
WEP	Wired Equivalent Privacy, soukromí ekvivalentní drátovým sítím
WPA, WPA2	Wi-Fi Protected Access, chráněný přístup k Wi-Fi
3D	Three-dimensional, trojrozměrný

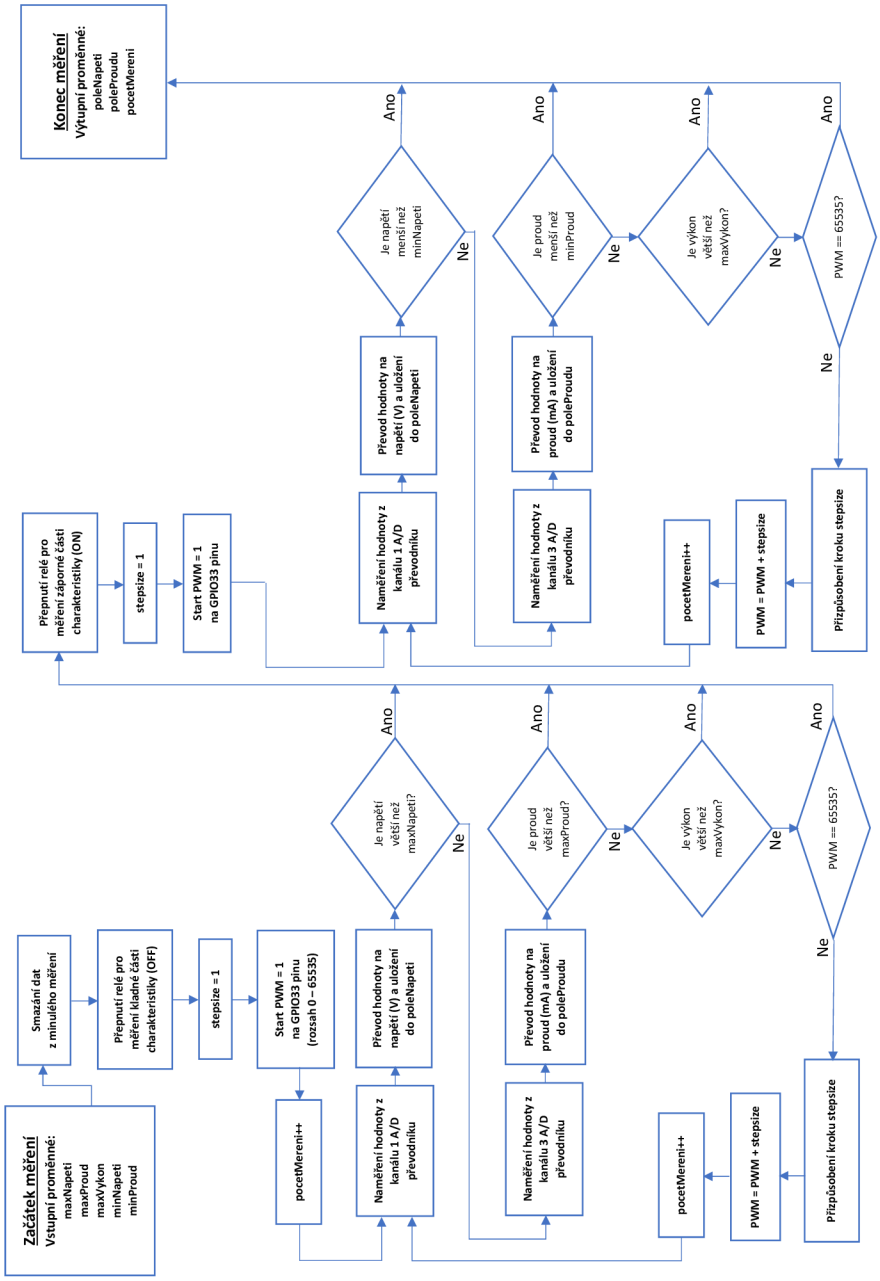
Seznam příloh

A Schéma zapojení



Obr. A.1: Schéma zapojení

B Blokový diagram funkce měření



Obr. B.1: Blokový diagram funkce měření napětí a proudu

C Obsah přiloženého CD

Na přiloženém CD se nachází zdrojový kód dvou formátech a to v jednoduchém .txt souboru a také jako .zip složka, kterou lze importovat přímo do vývojového prostředí Atom.io s Platform.io pluginem. Funkčnost softwaru byla otestována na verzi atom.io 1.46.0 x64 a platform.io verzi 4.3.3. Dále jsou přiloženy soubory pro editor Eagle obsahující schéma zapojení a model desky plošných spojů. Byla použita verze Eagle 9.5.0 free. A poslední soubor obsahuje 3D model ochranného krytu ve formátu .stl. Tento model lze zobrazit v již zmiňované internetové aplikaci TinkerCAD.soubor lze importovat do internetové aplikace TinkerCAD a model zobrazit.

```
/ ..... kořenový adresář přiloženého CD
├── 1.Diplomová_práce
│   └── Diplomová Práce - Petr Ondráček.pdf
├── 2.Zdrojový_kód
│   ├── Atom.io+Platform.io - Řídící program.zip
│   └── Zdrojový kód.txt
├── 3.Eagle
│   ├── Eagle - Deska plošných spojů.brd
│   └── Eagle - Schéma zapojení.sch
└── 4.TinkerCAD
    └── TinkerCAD - Model ochranného krytu.stl
```