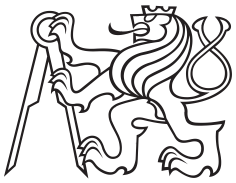
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra radioelektroniky

Zařízení pro automatické testování napájecích zdrojů

František Beránek

Školitel: doc. Ing. Stanislav Vítek, Ph.D.
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Beránek** Jméno: **František** Osobní číslo: **492027**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Zařízení pro automatické testování napájecích zdrojů

Název bakalářské práce anglicky:

Automatic Power Supply Testing Equipment

Pokyny pro vypracování:

Navrhněte zařízení pro automatické testování napájecích zdrojů. Zařízení bude umožňovat otestování nově vyrobený napájecí zdroj, vygenerovat a automaticky uložit report o testu. Zařízení by mělo disponovat minimálně 12 analogovými vstupy pro měření napětí, připínat zátěže na 6 jednotlivých výstupních větvích napájecího zdroje, odpojovat a připojovat 230V napájení zdroje. Bude průběžně ukládat hodnoty napětí a proudů jednotlivými větvemi v intervalech 10 min po dobu minimálně 3 h. Ukládání bude probíhat v ovládací aplikaci na PC, připojeném pomocí USB.

Seznam doporučené literatury:

[1] BROWN, Geoffrey. Discovering the STM32 microcontroller. Cortex, 2012, 3: 34.
[2] KREJČIŘÍK, Alexandr. Napájecí zdroje I, BEN–technická literatura, 2. vydání, Praha 1997, 351 stran.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Stanislav Vítek, Ph.D. katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **22.01.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

doc. Ing. Stanislav Vítek, Ph.D.	doc. Ing. Stanislav Vítek, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

Rád bych poděkoval firmě TSE spol. s.r.o. za možnost práce na zajímavých projektech a kolegům Ing. Radku Lustovi a Bc. Radimu Sejkovi za cenné rady. Dále bych rád poděkoval vedoucímu práce za příjemnou spolupráci při tvorbě bakalářské práce i za veškerou výuku v průběhu studia.

Prohlášení

„Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

v Praze 10.5.2022

Abstrakt

Práce se zabývá návrhem a realizací systému pro automatické testování zdrojů. Systém se skládá z aplikace pro PC tvořené pomocí frameworku Qt a přípravku s mikrokontrolerem STM32F072RB. Tyto funkční bloky jsou propojeny sběrnicí USB. Systém má za úkol po dobu definovanou v zadání zatěžovat testovaný zdroj a měřit jeho výstupní napětí. Hodnoty by měly být zaznamenávány aplikací a zapsány do protokolu o testu.

Klíčová slova: napájecí zdroj; testování; automatizace; Qt; mikrokontroler; STM32

Školitel: doc. Ing. Stanislav Vítek, Ph.D.

Abstract

This thesis describes design and making of a system for automatic power supply testing. System consists of a PC app created in a Qt framework and a device with STM32F072RB microcontroller. These two blocks are connected with an USB bus. This system should load tested power supply for a duration determined in the assignment and measure all output voltages. Measured values are handed over to an app and writen into a test report.

Keywords: power supply; testing; automation; Qt; microcontroller; STM32

Title translation: Automatic power supply testing device

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Jako každý nový výrobek musí i zdroje projít testovací fází, při které se ověří jejich funkčnost. Konkrétně pro zdroj jde o provoz při maximální možné zátěži. Již v dřívější době byl za tímto účelem ve firmě TSE spol. s.r.o. vytvořen přípravek skládající se z rezistorů připojených na jednotlivé větve zdroje a mikrokontroleru s připojeným displejem. Firmware přípravku umožňoval zobrazování průběhu testu a upozornění obsluhy, která v daných časových intervalech měřila napětí na zatíženém zdroji a naměřené hodnoty zapisovala do tištěného protokolu. Tento přístup je sice plně funkční, ale vyžaduje častý zásah obsluhy a nevyužívá plně potenciálu mikrokontroleru, který obsahuje i analogově/digitální převodník a umožňuje tak automatické měření.

Převážně vzhledem k časové náročnosti procesu zahoření pro obsluhu bylo rozhodnuto o nutnosti automatizace tohoto procesu. Jelikož byl původní přípravek navržen s ohledem na budoucí možnost automatizace, přistoupil jsem prvně k návrhu aplikace, která by se zařízením komunikovala pomocí sériového portu a k úpravě firmwaru i hardwaru tohoto přípravku. Toto řešení neumožňovalo další rozvoj a množství drobných improvizovaných úprav se negativně projevilo na spolehlivosti.

Tato práce popisuje návrh a realizaci aplikace a nového přípravku, který stejně jako předchozí provede proces zahoření zdroje zcela automaticky. Navíc oproti předchozí verzi hardwareově umožní připojení několika zdrojů a jejich postupné otestování. Tím dojde k urychlení, jelikož připojení nového zdroje k otestování může být provedeno zatímco je jiný zdroj právě testován.

V kapitole 2 je navrhnut způsob, jakým by měl výsledný systém fungovat. Tento návrh vychází ze zadání i některých rozšiřujících požadavků. Kapitola 3 rozebírá konkrétní návrh hardwaru. Software pro mikrokontroler v přípravku je rozebrán v kapitole 4. Následující část 5 popisuje vzniklou aplikaci pro PC. V kapitole 6 jsou rozebrány dosažené výsledky, které jsou porovnány se zadáním.

Kapitola 2
Návrh principu fungování

Tato kapitola se zabývá rozborem zadání a jeho rozdělením na jednotlivé funkční celky. Tyto celky jsou dále použity k návrhu blokového schématu. Tím je popsána funkce jednotlivých bloků a jejich propojení mezi sebou.

2.1 Rozbor zadání

Nejdůležitějším bodem pro funkčnost přípravku je možnost měřit dostatečné množství napětí. Zdroj, pro který je přípravek konstruován, poskytuje na svém výstupu celkem šest stejnosměrných napětí (plus napětí záložního akumulátoru) a jedno střídavé. Zařízení má být schopno i měření proudu při zátěži na jednotlivých větvích. Toto měření by mělo sloužit převážně k detekci chybného připojení zátěže. Vzhledem k účelu není pro měření proudu požadována velká přesnost. Ve výsledném řešení popsaném v této práci není měření proudů implementováno, ale to je určeno pouze softwarem, který by šlo snadno upravit.

Důležitou funkcionalitou je i měření záložní baterie. To probíhá při odpojeném napájení ze sítě. Z toho důvodu je kruciální, aby výsledné zařízení mělo možnost připojování síťového napětí k testovanému zdroji. Při odpojeném síťovém kabelu zdroj přejde do nouzového režimu a jeho řídící logika je napájena právě ze záložní baterie a dále napájí celou pětivoltovou větev zdroje.

Zdroj má vyvedené i některé logické vstupy. Konkrétně jde o řízení topení, které je připojováno k síťovému napětí, a řízení pětivoltové větve při provozu ze záložní baterie.

Zařízení by mělo být plně řízeno z počítače připojeného pomocí USB. I navzdory tomu jsem se rozhodl do návrhu zapracovat jak tlačítka, tak displej, který bude obsluhu informovat o současném stavu testu. K upozornění obsluhy při chybě, či změně fáze testování by měl sloužit akustický prvek. Tlačítka by měly sloužit hlavně k okamžitému odpojení zátěží a uvedení zdroje do výchozího stavu v případě jakéhokoliv problému a současného přerušení komunikace s počítačem. Jejich přesná funkce může být dále pozměněna pomocí softwaru mikrokontroleru.

Za účelem ukládání dat je třeba vytvořit aplikaci, která bude s přípravkem komunikovat, bude zobrazovat aktuální stav testu a také naměřené hod-

noty, aby mohla obsluha v případě nevyhovujícího zdroje test předčasně ukončit a postoupit k dalšímu. Dále musí aplikace ukládat naměřená data a vygenerovat z nich výslednou zprávu.

Komunikaci mezi přípravkem a aplikací je třeba zabezpečit tak, aby bylo možné rozeznat případné přerušení spojení, například při nechtěném odpojení USB kabelu. V takovém případě je nutné upozornit obsluhu, avšak jelikož má být zařízení schopné provést celý testovací proces zcela samo, je potřeba takovou skutečnost i zapsat do logu, který bude možné uložit zároveň s výslednou zprávou a pokusit se o dokončení testu.

Pro umožnění postupného testování více zdrojů v řadě je potřeba navrhnout systém připojování zdrojů, díky kterému se jednotlivé zdroje navzájem nebudou ovlivňovat. Dále je potřeba zajistit správné nastavení výchozího stavu pro zdroje čekající na test (zdroj je připojen na napájení ze sítě, odpojen od zátěže a logické vstupy jsou správně definovány). Maximální možný počet testovaných zdrojů by měl jít snadno změnit a tuto změnu musí zařízení rozpoznat a zareagovat na ní.

2.2 Připojování zdrojů

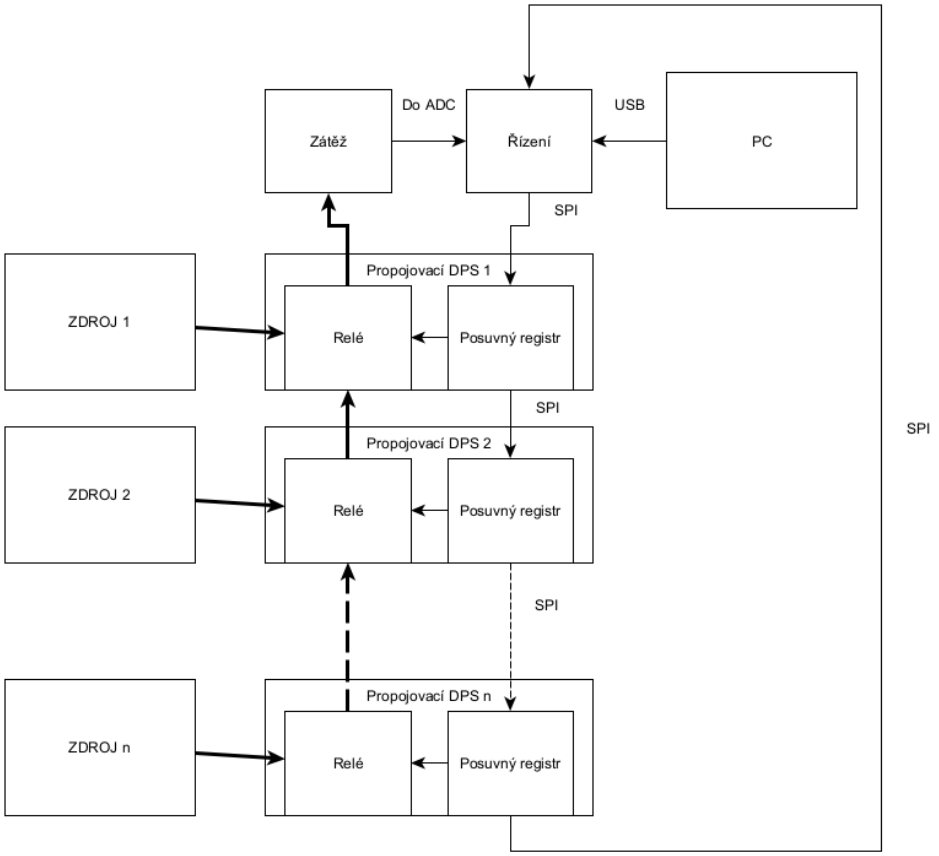
V první fázi bylo potřeba navrhnout způsob, kterým by bylo možné postupně otestovat několik zdrojů. Jako nejjednoduší a zároveň funkční způsob se ukázalo připojování všech větví zdroje i jeho logických vstupů pomocí relé. Při sepnutí relé se zdroj připojí k zátěži a logické vstupy jsou připojeny k výstupům mikrokontroleru.

Jelikož počet připojených relé desek a tím i počet připojených zdrojů by měl jít snadno změnit, rozhodl jsem se pro řízení relé použít posuvné registry se sériovým vstupem a paralelním výstupem. Ty lze snadno řadit za sebe způsobem zvaným daisy chain[8]. Každá relé deska bude obsahovat jeden registr, jehož výstupy budou řídit jednotlivá relé a indikační led. Aby bylo možné zjistit počet relé desek a tím i maximální možný počet připojených zdrojů, je potřeba poslední desku propojit zpět k mikrokontroleru. Při tomto zapojení je možné vyslat kontrolní znak a posílat ho zkrz daisy chain zapojení, dokud nedojde k jeho přijetí zpět. Počet potřebných posunutí znaku odpovídá počtu připojených desek.

Výsledné zapojení celého přípravku popisuje obrázek 2.1.

2.3 Zátěže

Tento blok by měl realizovat hlavní část celého přípravku, tj. samotnou zátěž. Vzhledem k určení zdroje není potřeba testovat jeho různé dynamické parametry. Test má pouze prokázat, že při delším provozu zdroje při maximálním dovoleném proudu nedojde k jeho poškození. Měřen je tedy pouze pokles napětí po připojení zátěže a jeho průběh v čase, konkrétně po dobu tří hodin. Z tohoto důvodu bude jako zátěž sloužit série resistorů s pevnou hodnotou.

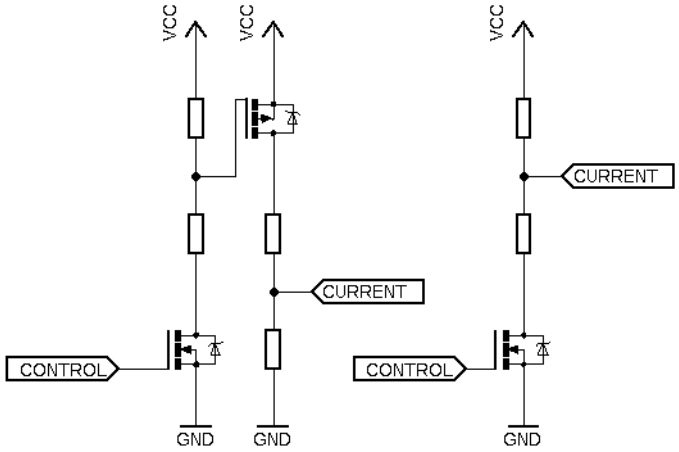


Obrázek 2.1: Blokové schéma zapojení

2.3.1 Zátěže pro stejnosměrné větve

Každá větev zdroje bude připojena na dvě sady odporových děličů. Jedna bude sloužit jako klasický napěťový dělič pro umožnění měření napětí pomocí mikrokontroleru. Druhá bude mít takovou hodnotu elektrického odporu, aby zatěžovala danou větev maximálním dovoleným proudem dle parametrů zdroje. Při tomto proudu by se na posledním resistoru děliče, tj. resistoru připojenému nejblíže záporné svorce větve, mělo objevit měřitelné napětí. Z této hodnoty úbytku napětí na resistoru lze pomocí Ohmova zákona určit proud resistorem.

Výkonový proudový dělič musí být možné připojovat a odpojovat. Zprvu jsem zátěže chtěl spínat pomocí transistorů typu MOSFET s kanálem P, ale jelikož má každá větev jiné napětí, byla by realizace takového zapojení relativně složitá. Toto zapojení zobrazuje obrázek 2.2. Na něm je tranzistor s kanálem P řízen pomocí druhého tranzistoru s kanálem N, který slouží jako invetor. Tento invertor slouží k uzpůsobení pro řízení pozitivní logikou (logická jedna na vstupu znamená připojení zátěže) a zároveň umožňuje spínání zátěže na libovolném napětí. Dělič napětí na vstupu tranzistoru P slouží k nastavení napětí Gate-Source a tím k nastavení pracovního bodu. Výhodou takového



Obrázek 2.2: Spínání zátěže

zapojení by naopak bylo, že by transistory byly připojené na kladnou svorku napájecího napětí a děliče by byly až pod nimi. Naopak při použití transistorů s kanálem N bude měření úbytku na posledním resistoru zatíženo aditivní chybou vlivem úbytku napětí na samotném transistoru. Toto zapojení je také zobrazeno na obrázku 2.2 pro porovnání. Vzhledem k povaze měření, nízké přesnosti výkonových rezistorů a zanedbatelnému odporu kanálu transistorů jsem se rozhodl tento jev zanedbat.

2.3.2 Zátěž pro střídavou větev

Střídavá větev je v testovaném zdroji tvořena samostatným sekundárním vynutím na vstupním transformátoru. Porucha této větve je z tohoto důvodu velmi nepravděpodobná. Proto byla v původním přípravku pouze připojena na zátěž, opět tvořenou resistory s pevnou hodnotou, a zatěžována po celou dobu testu. Ve svém návrhu jsem se rozhodl toto změnit a i tuto větev alespoň částečně během testu monitorovat. Vzhledem k vnitřnímu zapojení zdroje musí tato větev zůstat plně galvanicky oddělena od zbytku zdroje i od samotného přípravku. Za tímto účelem jsem se rozhodl měřit pouze proud a to pomocí sensoru s Hallovou sondou [9].

Vzhledem k relativně vysokému výkonu na této větvi (přibližně 250 W) nebude její zátěž umístěna na desce, ale bude tvořit samostatný blok připojený k zbytku zátěže konektorem. Připojování zátěže provádí relé.

2.4 Řízení

Řídící část zařizení bude tvořena zapojením s mikrokontrolerem. Ten musí řídit předchozí bloky. Dále musí ovládat displej a akustický prvek, číst signály z tlačítek a především komunikovat s řídící aplikací v PC, které bude i předávat naměřená data.

Kapitola 3
Návrh přípravku

Následující kapitola popisuje návhrh hardwarové části přípravku. Při návrhu vznikly celkem dvě verze přípravku. Druhá verze pouze opravuje některé drobné chyby, např. špatný pinout součástky atd., a došlo také k úpravě rozvržení pro snazší zástavbu finálního zařízení do pouzdra. Principiálně nedošlo při navrhování druhé verze k žádné úpravě, a proto jsem se rozhodl popisovat rovnou druhou verzi řešení.

Jednotlivé funkční bloky popsané v předchozí kapitole, tj. propojení, zátěž a řízení, jsou tvořeny samostatnými deskami plošných spojů.

3.1 Blok propojení

Při návrhu jsem se rozhodl začít právě blokem pro propojení zdroje a zátěže. na tento blok jsou kladeny jasné nároky odvíjející se od vlastností samotného zdroje.

Jak bylo popsáno v kapitole 2, připojení zdroje k žátěži a k řízení by mělo proběhnout pomocí sepnutí sady relé. Pro tento účel byly vybrány relé s cívkou pro napětí 24 VDC a uspořádáním kontaktů DPDT (Double Pole Double Throw, tj. 2x přepínací). Tyto relé byly pozůstatkem z předchozí výroby a plně splňují požadavky, tj. především maximální proud kontaktů. Výběr této komponenty zásadně ovlivňuje další návrh zařízení, protože klade nové nároky na napájení celého přípravku.

Dalším důležitým prvkem tohoto bloku je řízení napájení zdroje. k tomu slouží další relé, které je ovšem zapojeno jako rozepínací. Při tomto zapojení je zdroj ve výchozím stavu připojen k síti, což mu umožňuje dobíjet vestavěný akumulátor. Zároveň je zdroj odpojen od zátěže.

Řízení bloku obstarává mikrokontroler pomocí posuvného registru na desce. Konkrétně jde o integrovaný obvod SN74HC595 [10]. Data jsou do něj přenášeny pomocí sběrnice SPI a pomocí impulsu na řídícím vstupu RCLK se přenášejí na osmibitový paralelní výstup. Pro řízení relé jsou potřeba pouze dva výstupy. Jeden výstup ovládá připojení síťového napětí a druhý řídí připojení zdroje k zátěži. Další výstupy jsou použity pro indikaci stavu testu obsluze pomocí připojených LED. Tři LED jsou použity pro zobrazení průběhu testu a čtvrtá indikuje případnou chybu. Zbylé výstupy zůstávají nevyužité.

K cívce každého relé je dále připojena antiparalelní dioda zamezující proražení okolní logiky při přerušení proudu cívky. Spínání proudu cívkami je realizováno pomocí tranzistorů MOSFET s kanálem typu N. Na jejich gate je přiveden přímo signál z výstupu posuvného registru. Cívky všech relé připojujících zdroj k zátěži jsou řízeny společným tranzistorem a jsou propojeny paralelně. Při neměřeném odporu cívky 900 Ω a napětí 24 V je proud jednou cívkou dle Ohmova zákona

$$I_{\text{relay}} = \frac{U}{R} = \frac{24}{900} = 26,67 \text{ mA}. \tag{3.1}$$

Při celkovém počtu osmi relé je proud spínaný tranzistorem $I = n \cdot I_{relay} = 213,33$ mA. Tranzistory tohoto typu v běžném a malém pouzdře SOT23-3 jsou schopny spínat proudy v řádu jednotek Ampér, a tak je toto zapojení vyhovující.

V dalším kroku bylo potřeba vyřešit propojení jednotlivých desek tak, aby bylo možné v budoucnu snadno navýšit jejich počet a aby také bylo možné desky od sebe znovu oddělit, například při opotřebení relé, či jiné závadě. Po prozkoumání různých konektorů typu deska-deska se mi jako nejsnazší a funkční jevilo použití standardních pinových hřebínků s roztečí pinů 2,54 mm v úpravě pro SMD osazení. Ty jsou vždy použity v páru na spodní i vrchní straně desky v opačných provedeních (samec/samice). Po osazení pak stačí desky na sebe jednodušše nasunout a pro posílení mechanické pevnosti sestavu spojit pomocí sloupků. Zvýšenou pozornost bylo potřeba při návrhu věnovat propojení vstupních a výstupních dat posuvného registru tak, aby první deska přijímala data od mikrokontroleru a ostatní desky už přijímaly výstup předchozího registru. Naopak data na výstupním pinu poslední desky se musí vracet do mikrokontroleru.

Dále deska obsahuje už jen další konektory, které se dají rozdělit do tří skupin: připojení zdroje, připojení zbytku přípravku a přívod síťového napětí. Pro připojení zdroje slouží konektor IEC 320-2-2/F pro přivedení síťového napětí do zdroje, konektor IEC 60320C8 pro připojení výstupu zdroje pro topení (jde o síťové napětí, které je možné logickými vstupy zdroje odpojovat) a nakonec konektor Molex 39-30-1200, který přivádí veškeré vstupy a výstupy zdroje. Se zbytkem přípravku se propojovací deska připojuje prostřednictvím dvou konektorů. Řídící logika a napájení je přivedeno na konektor Molex 22-05-7108-10. Logické vstupy zdroje a jeho výstupní napětí vedou na konektor Molex 39-30-1200.

Rozmístění jednotlivých prvků na desce jsem se snažil volit tak, aby jednotlivé skupiny konektorů dle rozdělení v předchozím odstavci byly vždy na jednom místě. Z toho důvodu jsou konektory pro připojení zdroje orientované při jedné hraně desky. Tyto konektory budou po zabudování muset být dostupné obsluze. Na protilehlé straně pak jsou konektory pro připojení zbytku zařízení. Ty budou obsluze nedostupné, aby nedošlo k odpojení kabelových svazků nebo dokonce záměně vstupního a výstupního konektoru.

Rozměry desky nebyly určeny předem. Dopředu se sice počítá se zabudováním například do počítačové bedny, ale u přípravku se neklade žádný důraz na opakovatelnost výroby, a proto je možné zvolenou bednu upravit

podle hotového zařízení. Do rohů desky jsem dále umístil otvory o průměru 3,2 mm, které by měly sloužit k umístění již zmíněných sloupků.

Schéma zapojení a návrh DPS viz příloha.

3.2 Zátěž

Dalším blokem při návrhu byla zátěž. Narozdíl od předchozí desky propojení bude zátěž jediná pro celé zařízení. Testování zdrojů tak musí probíhat vždy po jednom a není možné testovat více zdrojů v jeden čas. Vzhledem k relativně vysokému příkonu celého zdroje by souběžné testování více zdrojů zbytečně zatěžovalo elektroinstalaci a celé zařízení by se zásadně komplikovalo.

3.2.1 Stanovení rozměrů

Vnější rozměry této desky jsem při návrhu volil stejné jako u desky propojení. Dále jsem použil i stejně umístěné otvory v rozích. Tak bude možné navrch spojených propojovacích desek připevnit i desku zátěže a získat tak kompaktní sestavu. Zátěž bude pochopitelně produkovat i určité množství tepla, se kterým je třeba počítat. Z toho důvodu jsem na desku umístil další čtveřici otvorů pro montáž počítačového větráku o průměru 120 mm. Ty jsou standardně nápájeny napětím 12 V, které je proto potřeba na desce také vytvořit.

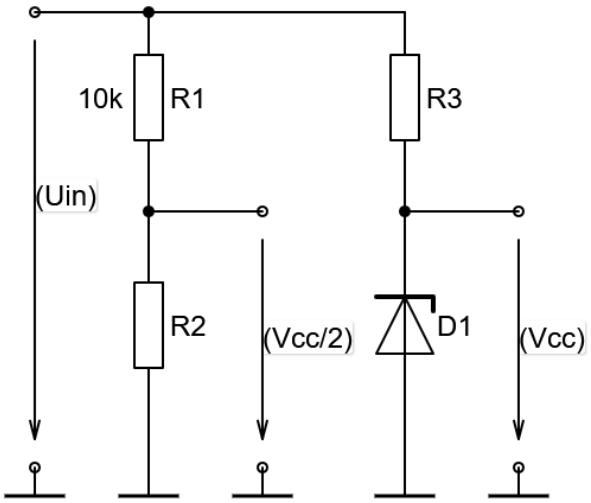
3.2.2 Výpočet děličů pro měření napětí

Protože výstup děliče bude připojen na analogový vstup mikrokontroleru, jehož vstupní impedance je veliká, je možné výpočet zjednodušit a uvažovat dělič jako nezatížený. Proud tímto děličem může být minimální. Pro zjednodušení výsledného listu součástek pro osazení jsem se rozhodl použít pro všechny děliče odpor R_1 (dle označení na obrázku 3.1) o hodnotě 10 kΩ. Tato hodnota i pro nejvyšší výstupní napětí zdroje 24 V zajišťuje proud děličem menší než $\frac{U}{R} = \frac{24}{10000} = 2,4$ mA.

Oproti původnímu zadání jsem se rozhodl do návrhu zakomponovat systém pro jednoduchou detekci správného připojení zdroje. K tomuto účelu jsem použilo hradlo NAND s osmi vstupy. Toto hradlo používá napájecí napětí stejné jako mikrokontroler vybraný v kapitole 3.3.1, tj. 3,3 V. Všechny vstupy hradla je možné pomocí SMD propojek připojit na toto napájecí napětí. To je důležité, jelikož jeden analogový vstup je na desce navíc a je zamýšlen pro možné budoucí rozšíření. Dále je díky tomu možné zařízení testovat bez připojeného zdroje.

Z předchozího odstavce je jisté, že je potřeba použít dva děliče napětí. První bude mít na výstupu napětí rovné logické 1, tzn. $\frac{3,3}{2}$ V$< U_2 <=$ 3,3 V. Druhý dělič sloužící k měření analogové hodnoty bude mít výstupní napětí kolem poloviny referenčního napětí analogových vstupů. Jako referenční napětí se pro zjednodušení v tomto návrhu používá napájecí napětí.

Vzhledem k možné nemalé změně vstupního napětí na výstupu zdroje při zatížení jsem se pro získání logické hodnoty na vstupu hradla rozhodl



Obrázek 3.1: Dělič napětí

U_{in}[V]	5	9,4	12	15	24
$R_2[\Omega]$ vypočítaný	6667	2703	2000	1538	909
$R_2[\Omega]$ volený	7500	3000	2200	1500	910

Tabulka 3.1: Hodnoty odporů děličů

použít zapojení se Zenerovo diodou namísto děliče napětí, viz obrázek 3.1. Odpor R_3 je pro takové zapojení určen jako

$$R = \frac{U_{in} - U_D}{I_D}, \tag{3.2}$$

kde U_{in} je vstupní napětí, U_D je Zenerovo napětí diody a I_D je proud diodou potřebný k správnému nastavení pracovního bodu diody. Po provedení tohoto výpočtu jsem ovšem došel k zjištění, že pro některé větve bych nemohl použít resistory v pouzdře 0805 z důvodu překročení maximálního doporučeného výkonu. Proto jsem pro všechny odpory R_3 použil také hodnotu 10 kΩ. Jelikož je proud tekoucí do vstupu logického hradla minimální, mělo by toto zapojení být plně funkční a případně lze rezistor snadno zaměnit.

Při výpočtu děliče pro měření napětí jsem prvně určil hodnotu odporu R_2 tak, aby výstupní napětí děliče bylo přibližně polovinou napájecí napětí logiky (3,3 V), které je použito jako referenční, dle vzorce 3.3.

$$U_2 = U_1 \cdot \frac{R_2}{R_1 + R_2} \tag{3.3}$$

Hodnoty odporu pro děliče popisuje tabulka 3.1. Jak je vidět, v některých případech je volená hodnota jiná než vypočítaná. to vyplývá z toho, že hodnoty napětí jsou nominální a u reálného zdroje se liší. Při volbě jsem proto vycházel i z předchozích zkušeností.

větev	$R[\Omega]$	$R_2[\Omega]$	$R_1[\Omega]$
5 V/1,5 A	3,33	1,1	2,23
5 V/50 mA	100	33	67
12 V/1,2 A	10	1,38	8,62
15 V/0,15 A	100	11	89
24 V/0,5 A	48	3,3	44,7
24 V/0,2 A	120	8,25	111,75

Tabulka 3.2: Hodnoty odporů děličů

3.2.3 Výpočet děličů zátěže

Pro tyto děliče je na rozdíl od předchozích kritický hlavně celkový odpor, který lze určit z napětí a maximálního povoleného proudu dané větve dle Ohmova zákona. Dále je potřeba určit odpor R_2 tak, aby na něm při plném zatížení vznikal úbytek přibližně $3,3/2$ V opět pomocí Ohmova zákona. Odpor R_1 pak určíme jako $R - R_2$, kde R je celkový odpor děliče. Vypočtené hodnoty zobrazuje tabulka 3.2.

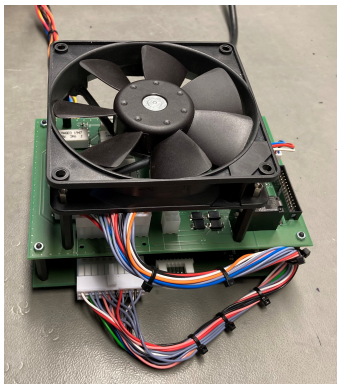
Pro zátěž je kritická i hodnota maximálního výkonu pro daný resistor. Rozhodl jsem se použít odpory s maximálním dovoleným výkonem 5 W. Proto bylo nutné některé resistory z návrhu nahradit sério-paralelní kombinací tak, aby žádný resistor nebyl přetěžován. Pro některé méně výkoné větve zdroje jsem použil resistory v pouzdře M2512 s maximálním připuštěným výkonem 0,5 W pro zmenšení výsledné zátěže.

3.2.4 Ochrana analogových vstupů

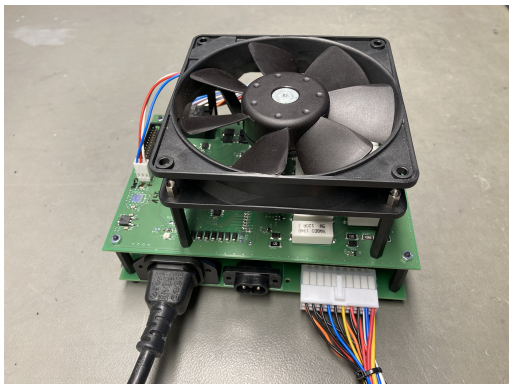
Jelikož se napětí na výstupu zdroje v průběhu testu mění a může dojít i k poškození zdroje, je nutné alespoň částečně zabezpečit analogové vstupy mikrokontroleru. Tato ochrana je v přípravku realizována právě na desce zátěže a to antiparalelně připojenou Zenerovo diodou. Pokud by došlo k nárůstu napětí na výstupu zdroje, měla by se tato dioda otevřít a napětí by nemělo překročit úroveň napájecího napětí. Pokud by k takové situaci došlo, bylo by dobré to pomocí softwaru vyhodnotit, jelikož však v takovém případě stejně zdroj nevyhoví daným mezím, není to nutně potřeba.

3.2.5 Napájení

Na této desce je pro funkčnost všech jejích částí potřeba hned několik různých napětí. Pro cívku relé je z desky řízení přivedeno napětí 24 V. Logické hradlo je napájeno napětím 3,3 V, které je rovněž přívedeno z desky řízení. Dále je potřeba napájet větrák. Pro ten je počítáno s napětím 12 V, kterých je na desce dosaženo lineárním stabilizátorem 7812. Napětí 12 V je sice vytvořeno i na desce řízení, ale vzhledem k nemalému proudu, který větrák odebírá, jsem se rozhodl použít samostatný zdroj. Posledním potřebným napětím je 5 V. Ty slouží k napájení proudového senzoru a jsou na desce vytvořeny dalším lineárním stabilizátorem, tentokrát 7805. Kolem obou stabilizátorů



(a) : Propojení relé desky a zátěže



(b) : Připojení zdroje

Obrázek 3.2: Sestava relé desky a zátěže

jsou rozmístěny prokovy sloužící k lepšímu odvodu tepla pomocí rozlité mědi (souvyslé plochy mědi) připojené na společnou zápornou svorku napájení. Stabilizátory jsou dále vybavené blokovacími kondenzátory na vstupy a výstupu podle datasheetů výrobce. [11]

Sestava relé desky a zátěže je na fotografiích 3.2. Zátěž pro střídavou větev je zobrazena na obrázku 3.3. Schéma zapojení a návrh DPS viz příloha.

3.3 Řízení

Posledním hardwarovým blokem je blok řízení. Ten má za úkol kromě ovládání obou předchozích částí i komunikaci s řídící aplikací v PC.

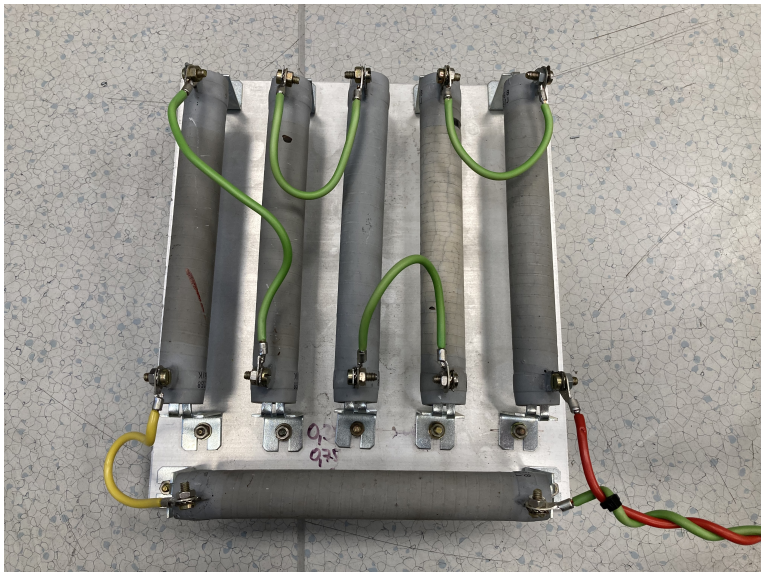
3.3.1 Výběr mikrokontroleru

Při výběru mikrokontroleru jsem v první řadě zvážil všechny požadavky. Nejdůležitějším bodem je dostatečný počet analogových vstupů umožňujících měření napětí a nepřímo i proudů. Zdroj má celkem osm měřených větví. Pro šest z nich bude měřeno napětí i proud, pro baterii pouze napětí a pro střídavé napětí pouze proud. Z toho vyplývá, že mikrokontroler musí disponovat minimálně čtrnácti analogovými vstupy.

Pro řízení posuvných registrů a tím i připojování zdrojů k žátěži slouží sběrnice SPI. Tu je možno emulovat softwarově na libovolných GPIO (general purpose input/output) pinech, ale většina mikrokontrolerů bývá vybavena hardwerem, který práci s tímto rozhraním usnadňuje.

Komunikace s PC má probíhat pomocí rozhraní USB. Některé mikrokontrolery jsou vybaveny hardwarem pro práci s touto sběrnicí, ale pokud nejsou dá se snadno využít i převodník UART-USB. Pro jednoduchost výsledné DPS jsem preferoval integrované rozhraní.

Dalším řízeným prvkem je displej. Vybraný displej komunikuje pomocí již zmíněného rozhraní SPI, takže nemění nároky na mikrokontroler.



Obrázek 3.3: Zátěž pro střídavou větev

Posledním parametrem je dostatečné množství GPIO pinů. Odhadem jde o deset pinů schopných pracovat jako digitální výstup a dva piny jako vstup, ideálně s možností vyvolání přerušení při detekci hrany vstupního signálu.

Vzhledem k vyjímečné situaci na trhu v poslední době, tj. nedostatku polovodičových součástek, se objevil další a nejvíce rozhodující parametr a to dostupnost. I u běžně dostupných mikrokontrolerů se dodací doba v čase vzniku této práce pohybuje v řádech několika měsíců. Proto jsem se rozhodl použít mikrokontroler STM32F072RB, který jsem již delší dobu vlastnil v podobě vývojového kitu Nucleo. Tento mikrokontroler splňuje všechny dané požadavky, včetně přímé podpory USB. Z pohledu výpočetního výkonu je pak pro danou aplikaci dokonce předimenzovaný.

3.3.2 Napájení

Celý přípravek by měl být napájen pomocí adaptéru připojeného prostřednictvím standardního barrel jacku. Adaptér by měl poskytovat napětí 24 V, tj. nejvyšší napětí použité v přípravku. Zbylá nižší napětí jsou vytvářena pomocí lineárních stabilizátorů. Pro případ potřeby připojení dalšího větráku, je i na desce řízení vytvářeno napětí 12 V pomocí stejného stabilizátoru jako na desce zátěže. Pro mikrokontroler je dále vytvářeno napětí 3,3 V pomocí stabilizátoru LD1117ADT33TR. Ten je napájen nikoliv ze vstupního napětí 24 V, ale z napětí 12 V, aby nebyla překročena hodnota maximálního vstupního napětí udávána výrobcem. Všechna napájecí napětí jsou vyvedena na konektory, aby je v případě potřeby bylo možné využít při budoucí úpravě přípravku.

3.3.3 Uživatelské rozhraní

V bloku řízení jsou obsaženy prvky pro interakci s uživatelem. Konkrétně jde o displej, tlačítka a piezoelektrický akustický prvek.

Tlačítka jsou připojena na digitální vstup mikrokontroleru tak, aby v klidovém stavu byl vstup na hodnotě logické nuly. Po stisku je vstup připojen k napájecímu napětí, tj. logické jedna. Pro zajištění klidového stavu je potřeba v softwaru mikrokontroleru nakonfigurovat použití interních pull-down resistorů. Pro potlačení zákmitů na vstupu jsou použity kondenzátory.

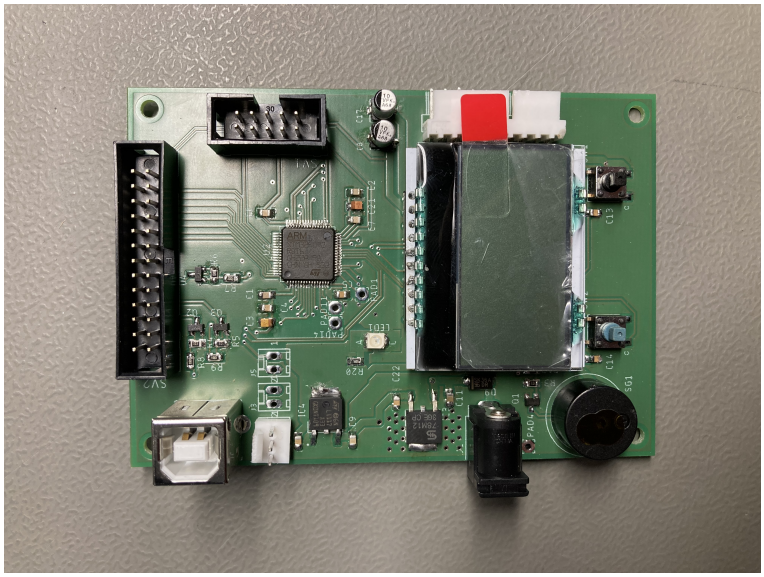
Jako akustický signalizační prvek jsem použil aktivní piezo bzučák. Jeho výhodou je snadné používání, kdy stačí tento prvek pouze připojit k napájecímu napětí. Nevýhodou může být fakt, že frekvenci tónu nelze měnit. Pro daný účel není tato skutečnost překážkou. Prvek je řízen mikrokontrolerem pomocí tranzistoru.

Aby mohl obslužný personál sledovat průběh testu přímo na přípravku, je na desce řízení displej. Vzhledem k účelu použití jsem se rozhodl pro LCD displej s alfanumerickým řadičem. Pro zjednodušení jsem vybíral displej s napájecím napětím 3,3 V. Dále vzhledem k velkému množství již využitých pinů jsem volil displej schopný komunikace pomocí sériového rozhraní. Často jsou používaná rozhraní I2C a SPI. Přednost jsem dával SPI, jelikož je v návrhu již použito pro komunikaci s posuvnými registry a při správném zapojení není problém ho využít i pro komunikaci s displejem. Zmíněné požadavky splňoval displej DOGS164W-A firmy Electronic Assembly. Tento displej má navíc možnost výběru panelu podsvícení. Zvolil jsem variantu s RGW (Red, Green, White) podsvícením, které je také možné použít jako signalizační prvek.

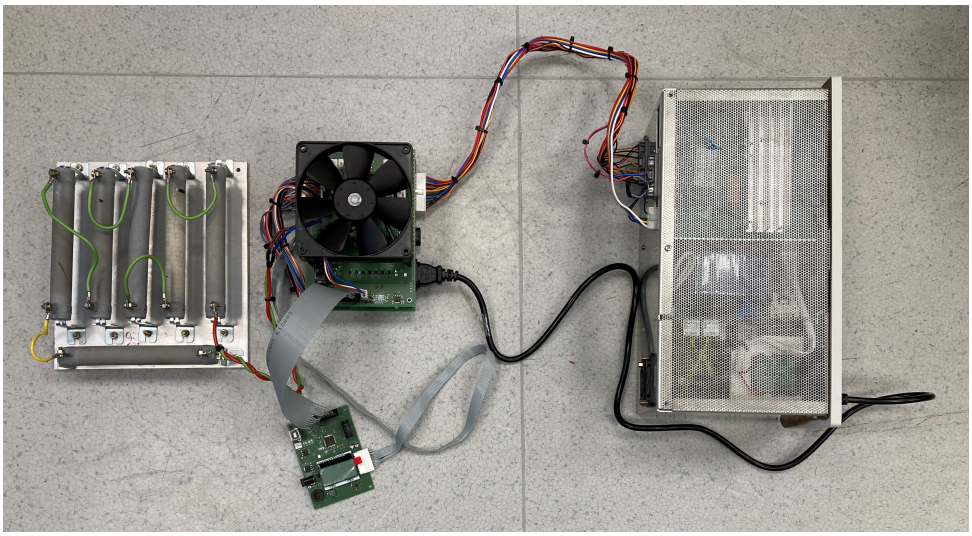
3.3.4 Rozmístění

Jelikož nebude tato deska ve výsledném zařízení nijak mechanicky spojena se zbylými deskami a bude s nimi propojena jen pomocí kabelů, je možné rozměry desky a rozmístění prvků navrhovat nezávisle na zbytku přípravku. Základním požadavkem je přístup uživatele ke všem podstatným prvkům. na to je třeba myslet hlavně u konektorů, kde některé musí být po zabudování přístupné (USB, barrel jack pro napájecí napětí), zatímco zbylé nesmí překážet.

Blok řízení ukazuje obrázek 3.4 a kompletní přípravek před zabudováním je zobrazen na obrázku 3.5. Schéma zapojení a návrh DPS viz příloha.



Obrázek 3.4: Řídící blok zařízení



Obrázek 3.5: Kompletní přípravek s připojeným zdrojem

Kapitola 4

Software mikrokontroleru

Následující kapitola rozebírá vzniklý zdrojový kód pro mikrokontroler v zařízení. Veškerý software mikrokontroleru je tvořen v jazyku C. S vyjímkou HAL (hardware abstraction layer) knihoven a některých základních knihoven jazyka C je kód vlastní. Kompletní zdrojový kód je součástí přílohy.

4.1 Prostředí STM32CUBE

Pro programování jsem použil prostředí STM32CUBE od firmy ST Microelectronics. K jeho použití jsem se rozhodl hlavně kvůli usnadnění celého procesu tvorby softwaru a to především při inicializaci použitých periferií MCU. Díky grafickému nástroji, ve kterém je možné nastavit parametry jednotlivých periferií (například přiřazení pinů, rychlost přenosu pro sběrnice, zdroj hodinového signálu atd.), nemusí programátor dokonale znát vnitřní uspořádání MCU. V tomto nástroji je možné jednotlivé piny také pojmenovat, což přispívá k čitelnosti kódu.

Po dokončení nastavení dojde k vygenerování kódu. Ten obsahuje především soubory main.c a main.h. Soubor main.h obstrává zahrnutí zdrojových souborů pro HAL knihovny pomocí direktivy preprocesoru include a obsahuje makra pro přiřazení pinů k názvům definovaných v grafickém nástroji. Soubor main.c obsahuje základní funkce pro inicializace periferií dle nastavení v grafickém nástroji a funkci main, která tyto inicializace provede. V obou souborech jsou pomocí komentářů vyznačena místa, ve kterých může programátor tvořit svůj vlastní kód. Veškerý kód, který by byl mimo tyto vyznačená místa by byl při příštím použití grafického nástroje přepsán.

Jako programátor jsem po celou dobu používal vývojový kit Nucleo. Programátor je k mikrokontroleru připojen pomocí rozhraní SWD (serial wire debug). Toto rozhraní umožňuje, jak již název napovídá, debugování softwaru. Komunikace s programátorem je v použitém prostředí velmi jednoduchá a je ovládána pomocí přehledného grafického rozhraní v rámci IDE (Integrated development environment).

4.2 Komunikace s PC

Komunikace s PC probíhá po sběrnici USB. Mikrokontroler obsahuje periferii pro obsluhu této sběrnice, a proto je možné ho přímo k rozhraní připojit. I přes to je však implementace softwaru pro řízení hardwarové jednotky USB netriviální. Naštěstí je při konfiguraci mikrokontroleru v grafickém nástroji zmíněném v předchozí kapitole možné do projektu zahrnout i oficiální API pro práci s USB ve zvoleném režimu. V tomto případě jde o režim device v configuraci Communication Device Class, tj. zařízení pro komunikaci. API poté převezme kontrolu nad obsluhou celé periferie a uživatel už jen přebírá z předem definovaného přijímacího bufferu data, nebo je pomocí funkce CDC_Transmit_FS vkládá do vysílacího bufferu.

Pro buffery používá API klasická pole. To je jednoduchý a funkční systém, ovšem při příjmu více dat za sebou dochází k přepisování staršího obsahu novým. Tento problém jsem řešil implementací kruhové fronty.

Kruhová fronta je elegantní způsob pro relativně jednoduchou realizaci paměťového prvku typu FIFO (first in-first out). Data je tak možné postupně přikládat na konec fronty a z jejího začátku je odebírat a zpracovávat. Použití fronty by bylo možné celé API přizpůsobit, ale k tomu by bylo vyžadováno hlubší proniknutí do jeho zdrojových kódů. Pro zjednodušení jsem se i s přihlédnutím k výkonovým a paměťovým dispozicím mikrokontroleru rozhodl zanechat původní řešení s zapsáním do pole a pouze ho doplnit o zapsání dat i do vlastní implementace kruhové fronty. I s tímto vysoce neoptimálním řešením problému jsem nezaregistroval žádný problém při příjmu dat a ani zdaleka jsem se nepřiblížil k využití celé paměti mikrokontroleru.

Komunikace mezi přípravkem a PC aplikací probíhá pomocí definovaných paketů. Strukturu paketu popisuje obrázek 4.1. Nad jednotlivými celky paketu je uvedeno jejich označení a v závorkách délka v bytech.

Start (2 B)		Type (1 B)	Data (n B)		CA (1 B)	Stop (3 B)		
>	>					<	<	\n

Obrázek 4.1: Struktura paketu

Startovací a koncová sekvence slouží k synchronizaci. Díky nim je možné v komunikaci od sebe jednotlivé pakety oddělit a rozpoznat i případný nedokončený paket.

Komunikace se během testování jevila spolehlivě, ale i přes to jsem se rozhodl v paketu zahrnout alespoň jednoduchý systém pro detekci chyb. Použit je byte získaný ořízntutím kontrolního součtu všech bytů dat a bytu typu paketu (označeno jako CA - Control Addiction). Samozřejmě existují i mnohem lepší metody jako například CRC, ale ty jsou náročnější na implementaci i při použití hardwarových jednotek mikrokontroleru a knihoven pro aplikace. Zabezpečení není tak kritické také z důvodu, že kód CRC je implementován přímo v rámci standardu USB [12].

Byte „type“ paketu slouží pro označení různých druhů paketů. Do PC jsou odesílany pakety typu: data, data měření baterie, fáze testu, číslo datového paketu, potvrzení, refresh paket a ukončení ze strany uživatele. Refresh paket je pravidelně odesílán jednou za sekundu, aby bylo v PC možné rozeznat případné přerušení komunikace. Paket s číslem označující pořadí dat je odesílán, aby bylo možné rozeznat případnou ztrátu dat. Paket s označením fáze testů je odesílán, aby aplikace dokázala zobrazovat současný stav testu.

Z PC do mikrokontroleru jsou posílány pakety typu spustit test, zastavit test, pozastavit test, kalibrace a navázání komunikace. Paket kalibrace vyžádá od mikrokontroleru jeden datový paket, který slouží k výpočtu převodních konstant pro aplikaci. Paket pro navázání komunikace je poslán hned po připojení aplikace k mikrokontroleru. Mikrokontroler na něj musí odpovědět paketem potvrzení, ve kterém posílá jako data počet připojených relé desek.

Celá komunikace je řízena pomocí funkce comHandler volané v nekonečné smyčce funkce main. Funkci comHandler popisuje vývojový diagram 4.2.

4.3 Řízení událostí

Jelikož je pro lepší čitelnost a přehlednost celý kod rozdělen do několika knihoven, bylo nutné vymyslet systém, kterým by bylo možné mezi knihovnami sdílet důležité informace. Například spuštění testu je podmíněno příchodem paketu s požadavkem na start testu. Knihovna starající se o zpracování příchozích dat proto musí být schopna předat informaci o požadavcích aplikace knihovně, která se stará o řízení průběhu testu. Toho jsem docílil použitím globální proměnné flags. Tato proměnná je deklarována v souboru main.c a sdílena v dalších souborech pomocí klíčového slova extern.

Proměnná flags je datového typu Flags definovaného pomocí klíčového slova typedef. Jde o bitové pole, které je pro větší přehlednost rozděleno na menší celky. Například pro každou instrukci přijatou z PC aplikace je použit jeden bit. Všechny bity indikující přijatou instrukci jsou sloučeny do bitového pole, které je součástí struktury Flags.

Tento přístup je plně funkční a z mého pohledu i přehledný, ovšem nebyl by vhodný, pokud by byl použit v projektu s více vývojáři. V mém případě bylo jednoduché ohlídat správné nastavování a nulování příznaků, ale pokud by se na projektu podílelo více lidí, bylo by lepší striktněji nastavit přístupy k jednotlivým příznakům.

Například struktura Flags obsahuje i bitové pole time. kde jsou uchovány příznaky o uplynutí určitých časových intervalů. Veškerou obsluhu tohoto bitového pole má za úkol funkce clkHandler a zbylé funkce by je měli používat pouze pro čtení. Proto by bylo lepší řešení pomocí různých mechanismů jazyka C přístup k těmto bitům omezit tak, aby s vyjímkou funkce clkHandler byly striktně pro čtení.

Další nedokonalost je ve skutečnosti, že pro některé knihovny jsou nasdíleny i příznaky, které nejsou pro danou knihovnu relevantní. Tyto nedokonalosti ovšem v žádném případě nesnižují funkčnost programu, pouze nabádají k zvýšené opatrnosti při psaní dalšího kódu.

4.4 Časování

Zařízení provádí nemalé množství činností, které vyžadují správné načasování. Dále je v některých částech programu použita funkce delay pro pozastavení běhu programu na určitý čas nebo se čeká na dokončení přenosu dat. Proto bylo potřeba vyřešit problém časování.

Základem řešení je využití čítače, který odměřuje interval dlouhý deset milisekund. Po uplynutí této doby je vyvoláno přerušení, v jehož obsluze je nastaven příznak v bitovém poli flags. Tento bit je pak sledován podmínkou v nekonečné smyčce funkce main. Díky tomu je zbylý program smyčky spuštěn pouze jednou za tento úsek a například čekání na dokončení přenosu na sběrnici SPI nenarušuje časování zbylých procesů, dokud netrvá dobu blízkou nebo dokonce delší než deset milisekund.

Jako první je v nekonečné smyčce fukce main volána funkce clkHandler. Ta se stará o správu všech bitů proměnné flags souvisejících s časováním a o proměnné určující čas. Na základě nového nastavení proměnných a příznaků mohou další funkce vykonávat úkoly (například posílání refresh zprávy probíhá při každém nastavení flagu o uplynutí jedné vteřiny).

Nevýhodou tohoto řešení je fakt, že nelze rozenat stav, kdy by program ve smyčce funkce main trval déle než je stanovený interval. V takovém případě by totiž došlo k vícenásobnému nastavení příslušného flagu v obsluze přerušení čítače a pro zbylý běh programu by se situace nelišila od běžného chodu. Tento problém by se dal řešit napřílad sledováním flagu i v obsluze přerušení. Pokud by došlo k přerušení a flag by byl nastaven dalo by se říci, že se zbytek programu zpožďuje. V takovém případě by bylo možné při déle trvajícím zpoždění například resetovat mikrokontroler, avšak během celého vývoje a testování jsem se s tímto jevem nesetkal.

4.5 Řízení průběhu testu

Průběh testu je řízen na základě stavového automatu. Ve výchozím stavu jsou všechny zdroje odpojeny od zátěže a připojeny k napájení. Při požadavku na start se stavový automat posune. Následně jsou změřena napětí naprázdno. Ve chvíli kdy je měření napětí dokončeno, stav testu se opět posune kupředu a zdroj je připojen k zátěži a je změřeno napětí zatíženého zdroje.

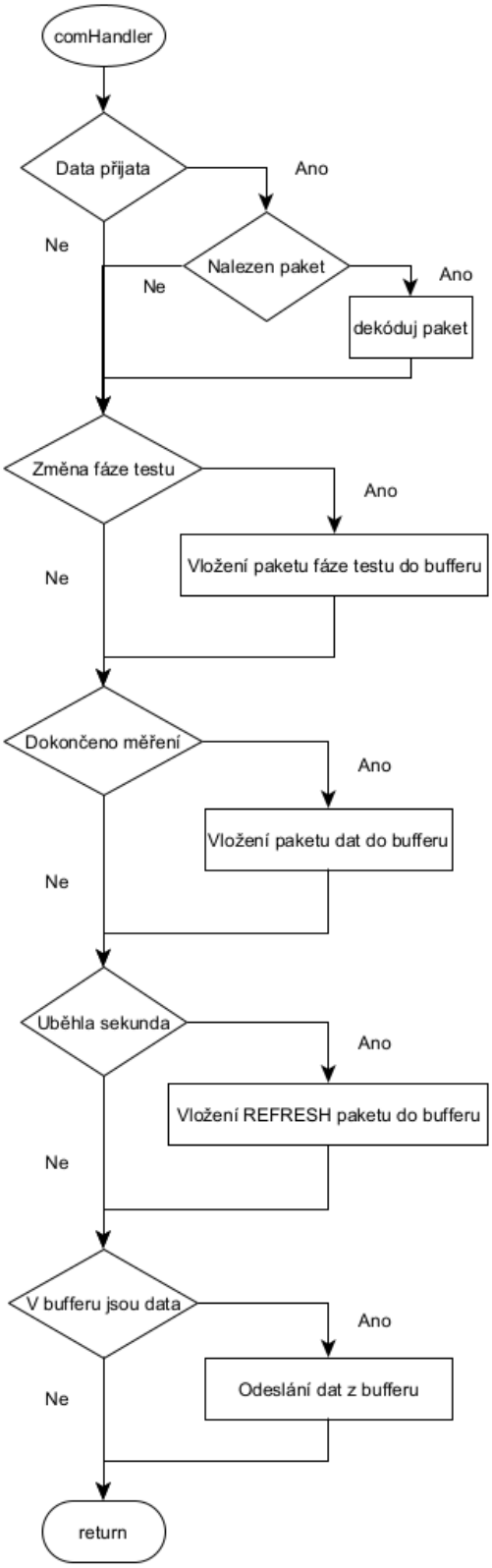
V průběhu hlavního testu je na základě časovacích proměnných a příznaků spouštěno měření každých deset minut. Po dokončení hlavního testu je stavový automat posunut, zdroj je odpojen od sítě a přechází se na měření baterie.

V průběhu celého testu jsou sledovány přiznaky přijetí instrukce k ukončení testu a o stisku ukončovačího tlačítka. V takovém případě je zdroj uveden do výchozího stavu a test je ukončen.

4.6 Funkce main

Funkce main je vzhledem k napsaným knihovnám a obslužným funkcím velmi jednoduchá a přehledná. Inicializace periferií mikrokontroleru je zařízeno pomocí funkcí vygenerovaných automaticky a není se o ně proto potřeba již nijak starat. Dále jsou vytvořeny buffery a displej a posuvné registry jsou nastaveny do výchozího stavu.

V nekonečné smyčce funkce main jsou jednou za deset milisekund volány všechny obslužné funkce, například pro spávu hodin, komunikace, uživatelského rozhraní a podobně.



Obrázek 4.2: Vývojový diagram funkce comHandler

Kapitola 5
Aplikace pro PC

Tato kapitola popisuje návrh a tvorbu aplikace pro PC. Ta má za úkol komunikovat s přípravkem, řídit průběh testu, zobrazovat výsledky a stav a vygenerovat protokol s kompletními výsledky testu. Průběh testu je potřeba v aplikaci sledovat takovým způsobem, aby byly odhaleny případné chyby komunikace. Chyby přenosu dat musí být zaznamenány do logu a uloženy spolu s protokolem.

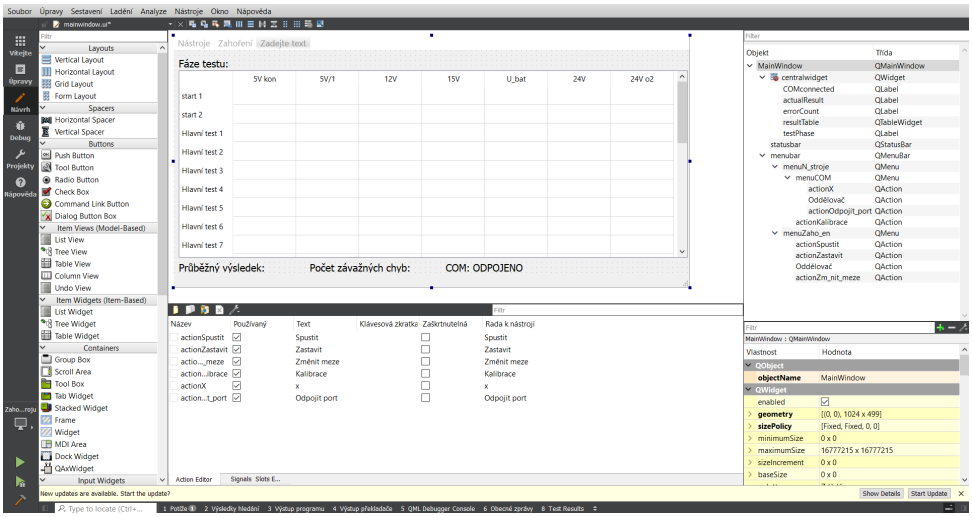
Aplikace je tvořena pomocí frameworku Qt. Jde o rozsáhlý a univerzální nástroj pro tvorbu aplikací. Jednou z hlavních výhod tohoto frameworku je možnost použití napříč platformami [1]. V této práci je použit jazyk C++, avšak podporované jsou i jiné jazyky jako například Python a JavaScript. Kompletní zdrojový kód a aplikace pro operační systém Windows (64-bit) je součástí přílohy.

5.1 Návrh uživatelského rozhraní

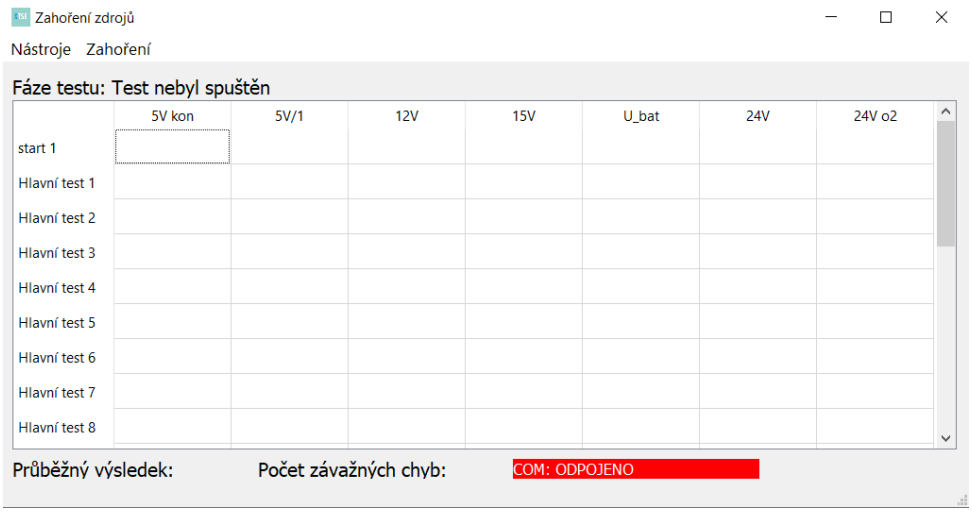
Pro návrh grafického rozhraní aplikace (označováno jako GUI, tj. Graphical User Interface) jsem využil nástroj Qt Creator [5], viz orázek 5.1. Díky tomuto nástroji je možné rozložení prvků navrhnout v grafickém prostředí a vidět tak podobu aplikace ještě před kompilací kódu. Po vložení jednotlivých prvků a nastavení jejich vlastností dojde k vygenerování hlavičkového souboru obsahující funkci pro vytvoření nadefinovaného rozhraní.

Tento soubor jsem sice při dalším vývoji aplikace značně pozměnil, ale jeho použití významě usnadnilo prvotní fázi návrhu. Převážně pro méně zkušeného vývojáře je tvorba pomocí grafického rozhraní výhodná, jelikož umožňuje snadno a přehledně objevovat dostupné předdefinované prvky a jejich možnosti. Ke změnám ve vygenerovaném hlavičkovém souboru jsem přistoupil až při změnách kódu majících za cíl zvýšení čitelnosti. Pokud je totiž rozložení celého okna navrhnuto najednou, je i vygenerován jediný soubor, který se stává dlouhým a nepřehledným.

Dominujícím prvkem okna je tabulka výsledků testu. Do ní jsou průběžně zaznamenávány naměřené hodnoty. Nad tabulkou je uváděna aktuální fáze testu. Zbylé důležité informace (průběžný výsledek, počet závažných chyb a stav sériového portu) jsou uváděny pod tabulkou. Při spodní hraně okna je umístěn status bar. Při vrchní hraně je umístěn menu bar obsahující menu



Obrázek 5.1: Nástroj pro tvorbu GUI



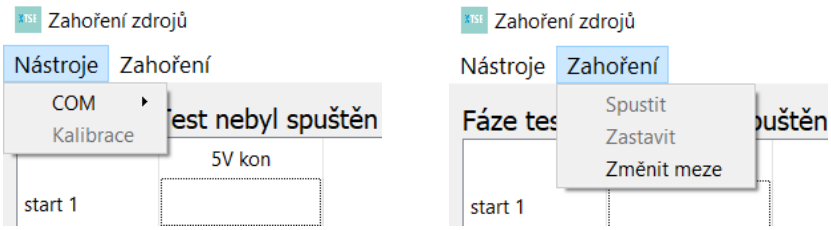
Obrázek 5.2: Výsledné okno aplikace

„Nástrojů“ a „Zahoření“. Okno spuštěné aplikace je na obrázku 5.2 a obě rozbalená menu na obrázcích 5.3a a 5.3b.

5.2 Signály a sloty

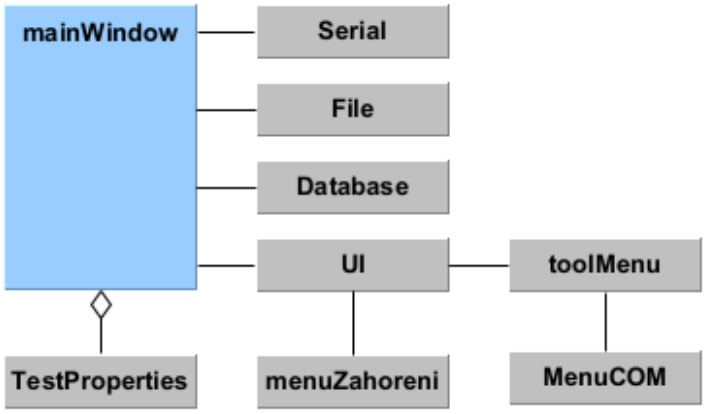
Pro propojení jednotlivých softwarových bloků aplikace v Qt slouží systém signálů a slotů. Dle [4] by signál měl být vyvolán při události, kterou reprezentuje. Slot je pak funkcí, která je spuštěna při zaznamenání příslušného signálu. Signály a sloty se propojují pomocí funkce „connect“. Jeden signál může být propojen s libovolným množstvím slotů a stejně tak může být jeden slot reakcí na větší množství signálů. Pokud je k vyvolanému signálu připojeno více slotů, jsou sloty volány podle pořadí, v jakém byly se signálem propojeny.

Třídy definované přímo frameworkem Qt osahují důležité signály i sloty,



(a) : Menu „Nástroje“ **(b) :** Menu „Zahoření“

Obrázek 5.3: Menu v menu bar



Obrázek 5.4: Diagram tříd aplikace

které pro svůj účel potřebují. Uživatel si při návrhu a tvorbě vlastních tříd může definovat i vlastní signály a sloty. Signály musí vždy být definovány jako věřejné funkce. Sloty naopak mohou být i privátní funkce. Díky propojení se signálem je tak možné volat i privátní a tím pádem přímo nedostupnou funkci. Propojení ovšem musí proběhnout v rámci třídy, které tento privátní slot náleží.

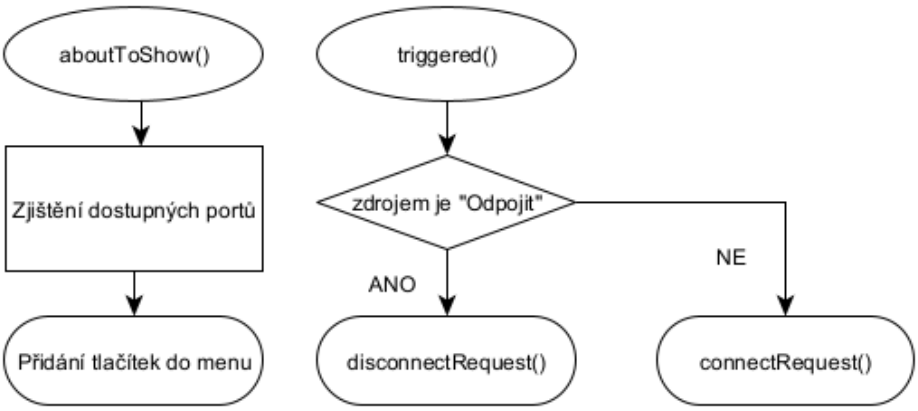
5.3 Struktura tříd aplikace

Aplikace je pro lepší čitelnost kódu rozdělena do několika tříd. Třídy a jejich vzájemné vztahy jsou popsány diagramem na obrázku 5.4.

Hlavní třídou je MainWindow dědící po QMainWindow. Jak název napovídá, tvoří tato třída hlavní okno aplikace.

Do hlavního okna jsou již ve funkci konstruktoru přidány veškeré grafické prvky. Ty jsou definovány ve třídě UI. Tato třída je původně dána kódem vygenerovaným po návrhu GUI v programu Qt Creator. Pro ještě větší přehlednost a snazší propojení některých signálů a slotů jsem z vygenerovaného souboru vyjmul všechny prvky menu lišty a nahradil je vlastními třídami toolMenu, menuZahoreni a MenuCOM. všechny tyto třídy dědí po QMenuBar.

Pro ukládání informací o testovaném zdroji slouží třída testProperties.



Obrázek 5.5: Menu COM

Aby byla aplikace schopna postupně testovat více zdrojů, udržuje třída mainWindow frontu instancí třídy testProperties.

Pro práci se sériovým portem slouží třída Serial. Tato třída dědí po QSerialPort a rozšiřuje ji tak, aby bylo možné pracovat s pakety.

O tvoření a editaci veškerých souborů aplikace se stará třída File dědící po QFile.

Poslední třída Database obstarává ukládání dat do SQLite3 databáze.

5.4 Propojení s přípravkem

Pro práci se sériovým portem v Qt slouží třída QSerialPort [3]. Pro ještě lepší přizpůsobení konkrétnímu použití jsem vytvořil vlastní třídu Serial dědící po QSerialPort. Díku tomu jsem získal možnost používat všechny veřejné funkce definované v QSerialPort, ale zároveň je rozšířit o vlastní.

Před samotným připojením k přípravku je potřeba získat jméno portu, na kterém přípravek komunikuje. Toho je docíleno tak, že v menu „Nástroje“ je vloženo další menu „COM“. Pokud uživatel na toto menu najede kurzorem, je vyvolán signál aboutToShow. Pomocí obslužného slotu jsou nalezeny dostupné porty a ty jsou jako tlačítka přidány do menu. Po stisknutí tlačítka je volán slot, který přípojí vybraný port. Celou funkci menu popisuje vývojový diagram 5.5. Funkce connectRequest a disconnectRequest jsou vyvolané signály, které jsou propojeny se sloty třídy Serial.

Samotné připojení, odpojení a další práci s portem zařizuje třída Serial. Pokud jsou přijata data z přípravku, je volán slot pro čtení. V něm dochází ke čtení dat a dekódování paketu. Načtený paket je předáván zbytku aplikace opět prostřednictvím signálu.

Třída Serial se stará také o kontrolu propojení. To je zařízeno pomocí paketu REFRESH. Ten je z přípravku posílán každou vteřinu. V aplikaci je pomocí instance třídy QTimer tento čas měřen a v případě, že časovač vyprší před příchodem paketu, je stav portu vyhodnocen jako neaktivní.

Stav portu je v aplikaci zobrazován pomocí barevně zvýrazněného textového štítku. Text i barevné schéma štítku je dáno stavem, který nabývá hodnot odpojeno, připojeno a neaktivní. Stav je aktualizován na základě signálu statusChanged.

5.5 Nastavení akcí menu

Pro snazší orientaci uživatele v menu baru jsou některé akce deaktivovány a není je proto možné vyvolat. Například pro spuštění testu platí, že přípravek musí být připojen a musí komunikovat. Veškeré aktualizace v nastavení dostupnosti akcí jsou prováděny automaticky na základě vyvolaných signálů.

5.6 Práce se soubory

Pro práci se soubory v Qt slouží třída QFile. Pro aplikaci jsem vytvořil třídu File dědící po QFile.

5.6.1 Souborový systém

V aplikaci je potřeba pracovat hned s několika soubory. V první řadě jde o samotný protokol, který je potřeba vytvořit a vyplnit. Dále v případě, že dojde k chybě v průběhu testu, je potřeba zapsat čas a popis chyby do logu.

Kromě předchozích souborů je potřeba udržovat i některé informace důležité pro chod a uživatelskou přívětivost aplikace. Jejich seznam a popis účelu je dán následujícím výčtem.

mustr.txt Šablona protokolu

meze.txt Krajní hodnoty pro jednotlivá napětí

workers.txt Pracovníci dříve provádějící test

defaultPath.txt Adresa složky, do které byl uložen poslední protokol

kalibrace.txt Převodní konstanty pro přepočet hodnoty AD převodníku na napětí

Všechny informace v souborech z předchozího odstavce by bylo lepší uchovávat v jediném strukturovaném souboru, například typu JSON. Díky tomu by mohl být kód třídy File mnohem přehlednější. Tento v současnosti použitý přístup byl ovšem pro začátek jednodušší. Nový způsob řešení právě se souborem typu JSON je součástí nové verze aplikace, která ovšem není v době vzniku této práce dostatečně otestována a doladěna.

5.6.2 Generování protokolu

V první verzi aplikace docházelo k generování jednoduchého reportu v podobě textového souboru. Grafická podoba protokolu je velmi strohá, ale jsou v něm přehledně obsaženy veškeré důležité informace.

Textový soubor je avšak jednoduše editovatelný a snadno by mohlo dojít k jeho úpravě. Z toho důvodu je v současné aplikaci generován protokol typu pdf. Práci s tímto typem souboru zajišťují třídy QPrinter a QTextDocument. I při jejich použití může být vygenerování graficky složitějšího dokumentu netriviálním úkolem, a proto jsem neměnil zápis dat do textového souboru v průběhu testu. K vygenerování výsledné zpravy v souboru typu pdf dojde až při konci testu přepisem celého textového dokumentu. Úprava této části aplikace je jedním z budoucích cílů.

5.7 Databáze

Po zkušenostech z předchozí výroby jsem se rozhodl nad rámec zadání přidat do aplikace podporu pro práci s databází. Díky tomu by mělo být možné sledovat vývoj napětí na jednotlivých větvích zdrojů v delším časovém horizontu. Například při změně některé ze součástek je pak možné vyhodnotit její dopad na funkci zdroje.

Použil jsem databázi SQLite3 a knihovnu sqlite3.h [7]. Strukturu databáze popisuje obrázek 5.6. Veškeré naměřené hodnoty jsou ukládány do tabulky „Hodnoty“ a pomocí cizího klíče „id_zdroje“ jsou přiřazeny ke konkrétnímu zdroji v tabulce „Zdroje“. Databáze je ukládána ve stejné složce jako soubory aplikace. Pokud by do databáze z nějakého důvodu nebylo možné zapsat data, měl by být tento fakt v budoucnu zapsán do logu. Zatím však tato chyba není nijak ošetřena, jelikož jde o nadstavbu původního zadání.

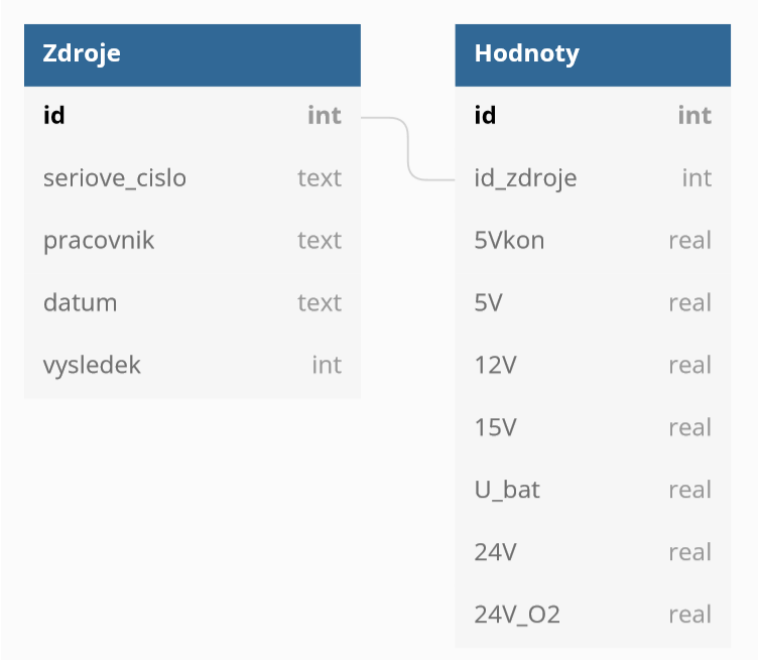
5.8 Podpora pro práci s více zdroji

V průběhu vývoje byla aplikace uzpůsobena, aby umožňovala využít schopnost přípravku testovat postupně několik zdrojů. Toho je docíleno pomocí třídy testProperties. Tato třída obsahuje všechny informace o testovaném zdroji. Pro všechny zdroje, které mají být otestovány je vytvořena a inicializována nová instance této třídy a je uložena do fronty.

Po dokončení probíhajícího testu je zkontrolován počet zdrojů k testu ve frontě. Jsou-li ve frontě další zdroje, je vyzvednuta následující instance třídy testProperties a pro ní je zahájen další test.

5.9 Spuštění testu

Před samotným zahájením testu je potřeba od uživatele získat data do protokolu. Toho je docíleno sérií dialogových oken zobrazených na obrázcích 5.7.



Obrázek 5.6: Struktura databáze

V první fázi je získáno číslo zdroje k testu (obrázek 5.7a). Toho je docíleno pomocí seznamu volných indexů. Pokud má tedy přípravek například pět relé desek, je seznam tvořen indexy od jedné do pěti. Pokud uživatel vybere například index dva, je pro daný zdroj spuštěn test. Při dalším spuštění dialogu budou nabídnuty jen indexy 1, 3, 4 a 5. Index dva je znovu nabídnut až ve chvíli, kdy neprobíhá ani není ve frontě test na této relé desce.

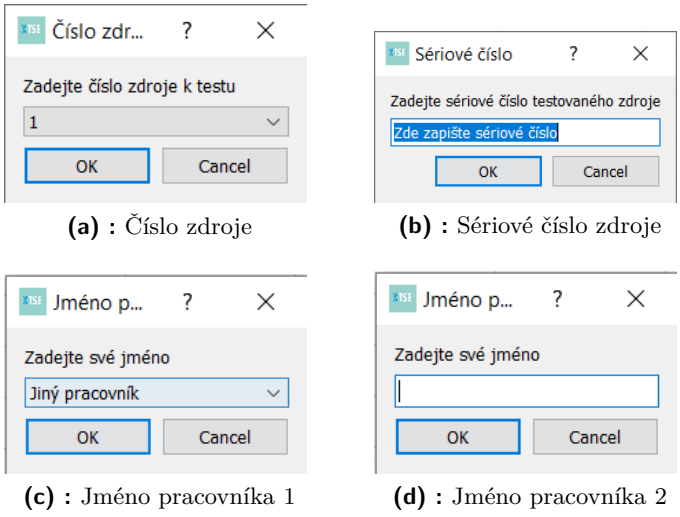
Druhým krokem je získání sériového čísla (obrázek 5.7b). To je uživatelem vloženo jako textový řetězec.

V dalším dialogu pracovník obsluhy vkládá své jméno. Pokud pracovník již dříve test prováděl, je jeho jméno zapsáno v souboru workers.txt. V tom případě stačí jméno vybrat ze seznamu (obrázek 5.7c). V opačném případě je zvolena možnost „Jiný pracovník“ a v dalším dialogovém okně je jméno vloženo v podobě textového řetězce (obrázek 5.7d).

Po získání všech důležitých informací je potřeba získat umístění pro uložení protokolu a jeho jméno. Jméno je standartně ve formátu sériového čísla s připojeným datem testu, kde datum je formátován DD_MM_RRRR. Výchozí umístění je voleno podle souboru defaultPath.txt, pokud v souboru již je uložena nějaká dříve použitá cesta. V opačném případě je volena cesta k umístění samotné aplikace. Dialog pro uložení je na obrázku 5.8.

Pokud by jakékoliv dialogové okno nebylo od uživatele potvrzeno tlačítkem Ok, je start testu přerušen.

Po vyplnění a potvrzení všech dialogů je do přípravku odeslán startovací paket. Spolu s odesláním je spuštěn časovač. Pokud první naměřená data z přípravku nepřijdou do intervalu odpočítávaného časovačem, je hlášena chyba a uživatel má možnost pokus opakovat nebo ukončit test. Při ukončení



(a) : Číslo zdroje **(b) :** Sériové číslo zdroje

(c) : Jméno pracovníka 1 **(d) :** Jméno pracovníka 2

Obrázek 5.7: Dialogy při spuštění testu

testu dojde i k smazání protokolu, ve kterém je prozatím pouze překopírovaná šablona.

5.10 Ukončení testu

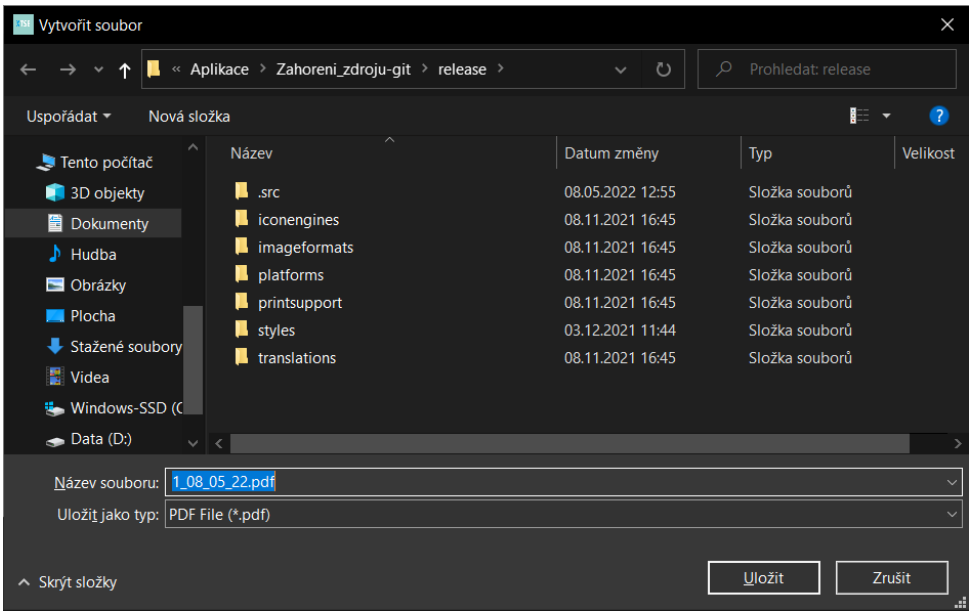
K ukončení testu může dojít třemi způsoby: ukončení pomocí tlačítka zařízení, ukončení z aplikace, test byl úspěšně dokončen. Pokud je test ukončen uživatelem z přípravku, jde o signál, že přípravek není v pořádku a proto je fronta zdrojů k otestování vymazána.

Nezávisle na způsobu ukončení testu je zbytek procedury stejný. Protokol je vygenerován je zkontrolován počet chyb v průběhu testu. Pokud nějaké chyby nastaly, má uživatel možnost nahlédnout do logu, aby se mohl rozhodnout, zda je test validní. Výsledek testu se zapisuje i do databáze. Podle počtu instancí třídy testProperties ve frontě je rozhodnuto o spuštění dalšího testu.

5.11 Kalibrace

Kalibrování přípravku je velmi jednoduchá procedůra. Aplikace si od zařízení vyžádá jeden datový paket. Po jeho přijetí je spuštěna série dialogů, ve kterých uživatel zadává naměřená napětí na jednotlivých větvích. Z naměřených hodnot a hodnot AD převodníků zaslaných z přípravků jsou poté určeny převodní konstanty, které jsou uloženy v souboru kalibrace.txt.

Tento přístup je funkční, ale předpokládá, že se měřená napětí v průběhu kalibrace nemění a že převodník má nulový offset.



Obrázek 5.8: Dialog pro uložení protokolu

Kapitola 6

Zhodnocení dosažených výsledků

Tato kapitola se zabývá rozborem výsledků dosažených v práci. Řešení se skládá za dvou základních bloků, tj. aplikace pro PC a zařízení s mikrokontrolerem. Oba bloky jsou propojeny pomocí sběrnice USB. Aplikace je schopna se zařízením komunikovat a přijatá data zpracovávat a nakonec ukládat do protokolu. Při běhu testu jsou ošetřeny všechny možné chyby. Zařízení zadaným způsobem testuje připojený zdroj a měří v průběhu testu jeho výstupní napětí.

Jediný bod zadání, kterého nebylo v rámci práce dosaženo, je měření proudů zátěží. Přípravek je navržen tak, aby tohoto měření byl schopen, ale softwarově tato funkcionalita není zprovozněna. Měření proudů zatím nebylo implementováno pouze kvůli svojí redundaci v rámci celého řešení zařízení. Jelikož je totiž zátěž tvořena odpory s pevnou hodnotou, lze proud určit pomocí naměřeného napětí a celkového odporu zátěže na dané větvi.

Aby bylo měřené napětí co nejpřesnější, je důležité hodnotu z AD převodníku správně přepočítat na hodnotu napětí. Toho může být docíleno pomocí hodnoty napěťové reference a dělícího poměru napěťového děliče. V takovém případě bude měření zatíženo nepřesností odporů děliče. Ve svém řešení jsem se rozhodl převod realizovat pomocí převodních konstant získaných z jednoho vzorku hodnoty AD převodníku a napětí naměřeného obsluhou pomocí voltmetru. Při zkušebním provozu byla maximální odchylka hodnoty měřené voltmetrem a přípravkem v řádu jednotek setin, což považuji za dobrý výsledek.

Komunikace mezi přípravkem a aplikací je zabezpečena pomocí paketů. Pro odhalení chyby přenosu je paket vybaven bytem kontrolního součtu. Původním plánem bylo implementovat systém CRC, ale během testování výsledného zařízení jsem nezaznamenal chybu v komunikaci, a proto jsem od tohoto cíle odstoupil a věnoval se prospěšnějším rozšířením. Bezchybovost komunikace je dána i použitím sběrnice USB, navíc s relativně malou vzdáleností zařízení.

Oproti zadání jsem v řešení implementoval některé rozšiřující funkce. Nejvýznamnější z nich je možnost přípravku otestovat postupně několik zdrojů. Cílem celého přípravku bylo ušetřit čas pracovníkům výroby. Díky tomuto rozšíření může obsluha k přípravku připojit několik zdrojů, zadat informace do aplikace a k přípravku se vrátit až po otestování celé skupiny. Jelikož má ovšem zařízení jedinou zátěž, nemůže dojít k paralelnímu testování více

zdrojů.

Dalším rozšířením je ukládání dat do databáze. V minulosti se stávalo, že například nová šarže určité součástky ovlivnila výsledky testování celé série napájecích zdrojů. Takový jev by v budoucnu díky databázi mělo být možné odhalit. Aplikace data do databáze pouze ukládá. Z toho důvodu bych se v budoucnu rád pokusil o vytvoření aplikace pro procházení a filtrování databáze za účelem odhalování výkyvů výstupních napětí. Pro použitý typ databáze ovšem existují i volně dostupné programy s tímto zaměřením, které jsou pro základní úkony postačující.

Grafické rozhraní aplikace je navíc doplněno o některé prvky na zařízení. Pro okamžitou kontrolu stavu testu obsahuje každá relé deska skupinu LED. Ty na první pohled určují, v jaké fázi testu se daný zdroj právě nachází a obsluha proto nemusí chodit až k obslužnému PC. Na displeji je zobrazován čas do konce testu a index testovaného zdroje. Průběžný výsledek testu a naměřené hodnoty jsou zobrazovány v aplikaci. Díky tomu je možné test nevyhovujícího zdroje předčasně ukončit a urychlit tím proces testování.

Kapitola 7

Závěr

Cílem této práce bylo navržení a realizace hardwarového a softwarového řešení pro testování napájecího zdroje. Systém se skládá z aplikace v PC a přípravku s mikrokontrolerem. Obě části spolu komunikují tak, aby aplikace řídila práci přípravku, který provádí samotný test. Přípravek naopak předává naměřená data do aplikace, kde jsou zpracována a ve formě protokolu ukládána.

Zátěž pro testování zdroje je realizována pomocí rezistorů s pevnou hodnotou. Tato realizace neumožňuje měření dynamických charakteristik zdroje, což ovšem nebylo součástí zadání. Návrh zátěže je proveden především s ohledem na jednoduchost ovládání a správné nadimenzování výkonové zatížitelnosti. Předem bylo počítáno i s odvodem vzniklého tepla.

Oproti zadání umožňuje výsledné řešení i postupné otestování několika zdrojů, aby mohla obsluha vykonávat jiné úkony delší dobu. Toho je docíleno pomocí propojovací desky. Způsob připojování je navržen s ohledem na maximální oddělení jednotlivých zdrojů od sebe i od zátěže mimo dobu trvání testu. Návrh desky probíhal s ohledem na zapracování do hotového zařízení.

Řídicí část přípravku byla navržena se snahou o minimální rozměry desky plošných spojů a pro celé zařízení platí snaha o minimální seznam součástek. Díky použitému mikrokontroleru zvládá přípravek všechny důležité úkony, tj. komunikace s PC, měření analogových napětí a řízení všech periferií. Nad rámec zadání obsahuje přípravek displej pro zobrazování průběhu testu i případná budoucí rozšíření.

Komunikace mezi PC a přípravkem probíhá pomocí paketů, které zajišťují zabezpečení přenosu. Pakety díky jasně definované struktuře usnadňují dekódovaní dat při příjmu. Pravidelně posílaná zpráva umožňuje aplikaci odhalit případné přerušení komunikačního kanálu. Jelikož test může probíhat i bez přítomnosti obsluhy, je tato funkce kruciální pro správné fungování celého systému.

Aplikace přehledným způsobem zobrazuje průběh testu a přijímaná data od přípravku. Na konci testu aplikace generuje protokol s naměřenými daty a informacemi o testovaném zdroji. Nad rámec zadání jsou data dále ukládána i od SQLite3 databáze pro možnost pozdějšího zpracování a vyhodnocení kvality výroby v čase.

Pomocí dialogových oken a logového souboru jsou ošetřeny možné chybové stavy. Použití logu nebo dialogu závisí na konkrétní situaci. Dialogy jsou

voleny pouze v případě, že je při dané chybě obsluha stále u PC a může zasáhnout.

Práce popsala návrh a realizaci konkrétního funkčního řešení systému pro testování zdrojů. Tento systém je navržen pro práci s konkrétním zdrojem, ale použité principy je snadno možné přenést.

Literatura

[1] *Qt* [online]. Qt Company [cit. 2022-05-04]. Dostupné z: `https://www.qt.io/`

[2] *Qt Open Source Licensing* [online]. Qt Company [cit. 2021-12-09]. Dostupné z: `https://www.qt.io/download-open-source`

[3] *QSerialPort Class* [online]. Qt Company [cit. 2021-12-07]. Dostupné z: `https://doc.qt.io/qt-5/qserialport.html`

[4] *Qt Signals & Slots* [online]. Qt Company [cit. 2021-12-06]. Dostupné z: `https://doc.qt.io/qt-5/signalsandslots.html`

[5] *Qt Creator* [online]. Qt Company [cit. 2021-12-08]. Dostupné z: `https://www.qt.io/product/development-tools`

[6] *Qt Forum* [online]. Qt Company [cit. 2021-12-08]. Dostupné z: `https://forum.qt.io/`

[7] *SQLite3* [online]. [cit. 2021-04-05]. Dostupné z: `https://www.sqlite.org`

[8] *SPI Daisy Chain* [online]. Maxim Integrated[cit. 2022-03-09]. Dostupné z: `https://www.maximintegrated.com/en/design/technical-documents/app-notes/3/3947.html`

[9] RAMSDEN, Edward. *Hall-Effect Sensors: Theory and Application* [online]. 2nd ed. Boston: Newnes, 2006 [cit. 2022-03-10]. ISBN 9780080523743. Dostupné z: databáze ProQuest

[10] *SN74HC595: datasheet* [online]. Dallas: Texas Instruments, 2022 [cit. 2022-03-10]. Dostupné z: `https://www.ti.com`

[11] *µA78Mxx Positive-Voltage Regulators* [online]. Dallas: Texas Instruments, 2015 [cit. 2022-04-07]. Dostupné z: `https://www.ti.com/lit/ds/symlink/ua78m.pdf?ts=1649318279810&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FUA78M`

[12] *ALEXON, Jan. USB Complete: The Developer's Guide. 5th edition. Madison: Lakeview Research, 2015. ISBN 9781931448284.*

Příloha A

Seznam příloh

- Zdrojový kód a kompilovaná aplikace (OS Windows 64-bit)
- Zdrojový kód softwaru mikrokontroleru (projekt STM32Cube IDE)
- Schéma a návrh jednotlivých DPS (formát GERBER a Eagle)