

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor: 10. Elektrotechnika, elektronika a telekomunikace

RGB Laserový projektor

Šimon Hrouda

Brno 2024

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

RGB LASEROVÝ PROJEKTOR

RGB LASER PROJECTOR

AUTOR	Šimon Hrouda
ŠKOLA	Gymnázium Brno-Řečkovice, p. o., Terezy Novákové 2, 621 00 Brno
KRAJ	Jihomoravský
ŠKOLITEL	Tomáš Rohlínek a Mgr. Kateřina Vídenková
OBOR	10. Elektrotechnika, elektronika a telekomunikace

Brno 2024

Prohlášení

Prohlašuji, že svou práci na téma *RGB Laserový projektor* jsem vypracoval/a samostatně pod vedením Tomáše Rohlínka a Mgr. Kateřiny vídenkové a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Dále prohlašuji, že tištěná i elektronická verze práce SOČ jsou shodné a nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a změně některých zákonů (autorský zákon) v platném změní.

V Brně dne: _____

Šimon Hrouda

Poděkování

Děkuji svým konzultantům Tomáši Rohlínkovi a Mgr. Kateřině Vídenkové za obětavou pomoc, podnětné připomínky a nekonečnou trpělivost, kterou mi během práce poskytovali.

Tato práce byla provedena za finanční podpory Jihomoravského kraje.

jihomoravský kraj



Anotace

Klíčová slova

Annotation

Keywords

Obsah

Úvod	8
1 Cíle	9
2 Laser scanning	10
2.1 Akusticko-optické skenery	10
2.2 Hranolové skenery	11
2.3 Galvanometrové skenery	12
2.3.1 Galvanometr	12
3 Laserová projekce	15
3.1 Využití laserové projekce v průhledových displejích (HUD)	15
3.2 Princip laserové projekce	16
4 Použité technologie, techniky, pojmy	18
4.1 GitHub	18
4.2 Node.js	18
4.3 SPI	19
4.4 I ² C	19
4.5 Pull-up a pull-down rezistory	20
4.6 PWM (Pulse Width Modulation)	20
4.7 Přerušení (Interrupt)	21
5 Hardware	22

5.1	Řídící jednotka – Raspberry Pi	24
5.2	Set galvanometrů se zrcátky	24
5.2.1	Výběr skeneru	24
5.2.2	Zapojení galvanometrového setu	25
5.2.3	Bipolární diferenciální analogový signál	25
5.2.4	Zahřívání čipů řídící desky galvanometrů	26
5.3	RGB laserový modul	27
5.3.1	Dichroická zrcadla	28
5.3.2	Zapojení laserového modulu	28
5.3.3	Tvorba barev s laserovým modulem	28
5.4	Displej z tekutých krystalů (LCD)	29
5.5	Rotační enkodér	29
5.5.1	Čtení pozice z rotačního enkodéru	31
5.6	HAT deska plošných spojů	31
5.6.1	Zdroj -15 V	32
5.6.2	Generátor analogového signálu	34
5.6.3	Voltmetr baterií	35
5.7	Napájení	36
5.7.1	Akumulátory	38
5.7.2	BMS modul	38
5.7.3	Relé modul	39
5.7.4	Nabíjecí obvod	39
5.7.5	Vypínač	41
5.7.6	Napěťové větve	41
5.8	Pouzdro	41
5.8.1	Priority designu	42
5.8.2	Konstrukce	43
6	Software	47
6.1	Komunikace mezi programy	48
6.2	Lasershaw	49
6.2.1	Inspirace open-source	49

6.2.2	Funkce programu	49
6.2.3	Využité knihovny	50
6.2.4	Struktura programu	54
6.3	wifi_manager	57
6.3.1	Funkce programu	57
6.3.2	Využité knihovny	58
6.3.3	Struktura programu	59
6.4	UI	60
6.4.1	Využité knihovny	60
6.4.2	Struktura programu	65
6.5	web_ui	69
6.5.1	Využité knihovny	70
6.6	discord bot	77
6.7	Instalační skript	77
7	Diskuze	79
	Literatura	88
	Seznam obrázků	90
	Seznam ukázek kódu	90

Note:
preferuj
itemize
nad
čarkou
odděle-
nýma
sezna-
mama

Úvod

Laser scanning, technologie rychle se pohybujícího laserového paprsku, je využívána v mnoha oblastech od laserového promítání, efektů na diskotékách [1] a průhledových displejů (Průhledový displej – anglicky Head-Up Display (HUD)) v letadlech, autech či brýlích pro rozšířenou realitu [2], přes čtení čárových kódů [3] a 3d tisk [4], po skenování 3D modelů [5] i Zemského povrchu [6].

Bohužel ale neexistují žádné uživatelsky přívětivé open-source platformy, kde by se s touto technologií mohli seznámit zájemci o její rozvíjení.

Kapitola 1

Cíle

Cílem práce je vytvořit pro technologii laser scanningu jednoduché uživatelské prostředí a následně toto prostředí spolu s návodom, jak svůj projektor zprovoznit, vystavit na server github.com, kde ho případní zájemci najdou. Toto uživatelské prostředí by mělo sloužit jako začáteční bod, který zaujme mladé zájemce a umožní jim si technologii vyzkoušet. V případě, že zájemce techologie zaujme, mělo by pro ně být jednoduché program pozměnit nebo si jinak přizpůsobit chování projektoru.

Kapitola 2

Laser scanning

Jako Laser scanning se označuje technologie využívající rychle se pohybující laserový paprsek. Mezi tradiční techniky pohybu paprsku spadají akusticko-optické skenery, hranolové skenery a galvanometrové skenery [7].

2.1 Akusticko-optické skenery

Princip Akusticko-optických (AO) skenerů spočívá ve změně indexu lomu materiálu optiky, když ním prochází akustická vlna. Tato změna indexu lomu způsobí změnu směru paprsku, který optikou prochází. [8]

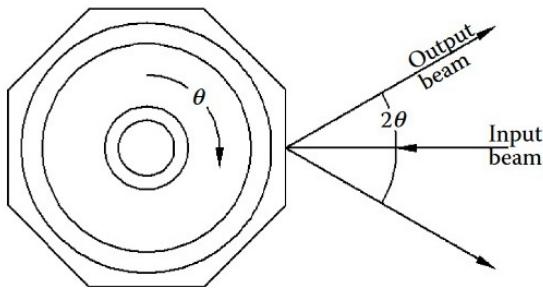
AO skenery, se nejlépe hodí do systémů s rozlišením¹ přibližně 1 000 bodů. Další charakteristikou AO skenerů je možnost přenést paprsek na libovolný bod v časovém úseku řádově 10 μs . [8]

Existuje mnoho systémů využívajících AO skenery, možná nejzajímavější jsou laserové tiskárny, které naplno využívají schopnosti AO skenerů. [8]

¹Rozlišení laserových skenerů je počet všech různých pozic, do kterých je skener schopen nasměrovat paprsek.

2.2 Hranolové skenery

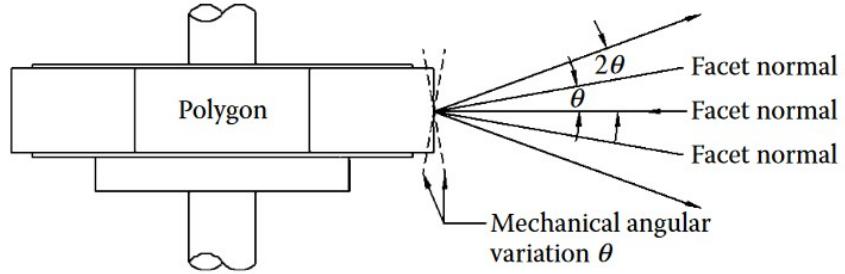
Hranolové skenery se vyznačují rotujícím hranolem se zrcadlivými stranami (dále „zrcátky“). Při rotaci hranolu se mění úhel dopadu laserového paprsku na zrcátko, a díky tomu se mění směr odraženého paprsku, viz obrázek 2.1 [8].



Obrázek 2.1: Mechanika polygonových skenerů [8].

S jedním hranolem by hranolové skenery byly schopny směřovat paprsek pouze v jedné rovině – při projekci by bylo možné vykreslit maximálně čáru. Tuto limitaci lze kompenzovat přidáním malého rozdílu ve směrování každé strany hranolu, viz. obrázek 2.2. S touto úpravou každá strana hranolu vykreslí jednu svoji přímku lehce posunutou vůči přímkám ostatních stran. Hranol s n -úhelníkovou podstavou je schopen vykreslit n přímek. Další možností je kombinovat původní pravidelný hranol s galvanometrem, kdy galvanometr nastaví jednu souřadnici paprsku a hranol na této souřadnici vykreslí přímku.

Tento typ skeneru se využívá hlavně pro senzory skenující na přímce (např. skenery čárových kódů [3]), nebo při rastrovém procházení plochy (například 3D skenování, nebo promítání rastrových obrázků, viz obrázek 2.3).



Obrázek 2.2: Úhlová rozdílnost zrcátek polygonového skeneru a paprsky od nich odražené [8].



Obrázek 2.3: příklad projekce laserového projektoru s polygonovým skenerem [9]

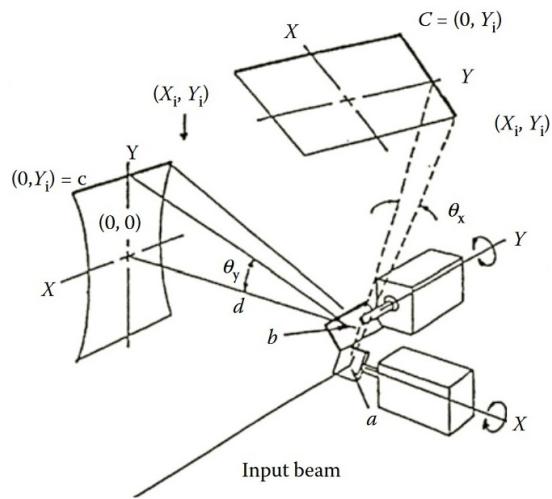
2.3 Galvanometrové skenery

V galvanometrových skenerech je paprsek odrážen dvěma zrcátky připevněnými na páru galvanometrů. Ty bývají situovány tak, aby každý galvanometr ovládal jednu osu pohybu paprsku, viz obrázek 2.4. [8]

Narozdíl od hranolových skenерů je s galvanometrovým skenerem možné zastavit libovolnou osu pohybu – vykreslovat čáry orientované jakýmkoliv směrem.

2.3.1 Galvanometr

Slovem galvanometr se označuje přístroj úrčený k detekci nebo měření velice malého elektrického proudu [10]. Galvanometry při měření využívají interakce magnetického pole trvalého magnetu a cívky protékané proudem. Tato interakce vy-



Obrázek 2.4: konstrukce galvanometrových skenerů [8].

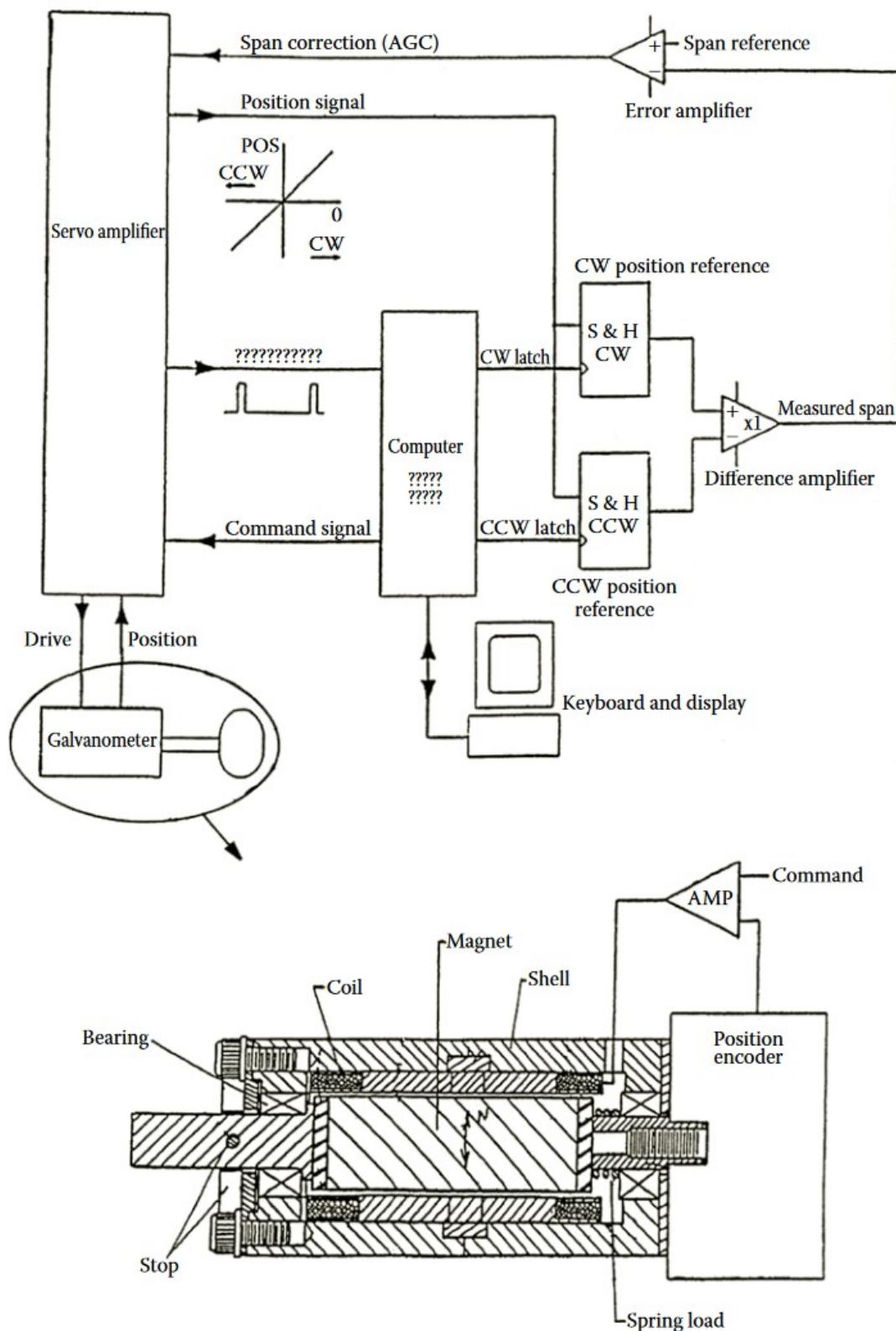
chýlí ručičku ukazující na stupnici, nebo zrcátko odrážející paprsek, který dopadá na stupnici. [11]

Můžeme tedy výchylku galvanometru přesně ovládat proudem, který ním protéká.

Galvanometry se dají rozdělit na galvanometry bez zpětné vazby (open-loop) a se zpětnou vazbou. K těm bývají připojeny ovládací obvody, které z galvanometrů získávají informace o jejich pohybu a podle nich regulují signál posílaný do galvanometrů. [11]

Dále se dělí dle pohyblivé součástky. V galvanometru je buď trvalý magnet pevně ukotven a cívka pohyblivá (moving coil), nebo naopak (moving magnet).

Dnes se v kontextu laserových skenerů prakticky vždy používají galvanometry s pohyblivým magnetem a se zpětnou vazbou. Ta je zajištěna čtením z variabilního kondenzátoru umístěného v galvanometru.



Kapitola 3

Laserová projekce

Laserová projekce spadá mezi laser scanning technologie. Často je využívána v zábavním průmyslu, hlavně k vytváření laser shows a vektorových projekcí. U laser shows diváci sledují vzory, které vytváří samotný paprsek ve vzduchu. U projekcí diváci sledují obrazce vykreslené paprskem dopadajícím na promítací plátno. [1]

Tyto efekty nejsou populární pouze v klubech, nýbrž i na koncertech nebo živých představeních. Je-li laser dostatečně silný, je možné promítat na obrovské plochy, jako například hráze, vodní plochy, nebo dokonce hory. [1]

3.1 Využití laserové projekce v průhledových displejích (HUD)

Laserová projekce se také využívá v průhledových displejích. Příklady HUD jsou vidět na obrázcích 3.1 a 3.2.

V tomto odvětví zatím převládají jiné technologie. Technologie průhledových displejů se dají rozdělit následovně:

- Technologie vyzařujících displejů, např. cathode ray tube (CRT), organic light-emitting diode (OLED) nebo vacuum fluorescent display (VFD).

Note:
myslim,
ze po-
měr
stupňo-
vací -
s carkou



Obrázek 3.1: HUD v letadle Boeing 737-800 [12]



Obrázek 3.2: HUD ve stíhacím letounu F16 [12]

- Technologie podsvícených displejů, např. liquid crystal display (LCD).
- Technologie laserových displejů, např. liquid crystal on silicon (LCoS) nebo laser scanning displeje založené na pohybu mikrozrcadel. [12]

V prvních průhledových displejích bylo využito CRT. Ale kvůli své neskladnosti, vysoké spotřebě elektriny a škodlivé radiaci, byla nahrazena technologií LCD. Dnes se v průhledových displejích letadel využívají LCD. V automobilech se využívají LCD a VFD. Bohužel, VFD jsou limitovány množstvím informací, které mohou zobrazit. LCD průhledové displeje jsou limitovány svým maximálním jasem. S novou technologií OLED sice je možné vytvořit tenký a průhledný displej dosahující vyššího jasu, než LCD HUD. I tento displej bohužel oproti vnějšímu světu má stále relativně nízký jas, také má vysokou cenu a krátkou životnost. Oproti vyzařujícím a podsvíceným displejům jsou laserové displeje nadřazené. [12]

3.2 Princip laserové projekce

Když se laserový paprsek pohybuje dostatečně rychle, lidské oko ho vnímá jako spojitou linku světla – tomuto jevu se říká *persistence of vision* nebo *persistence of impression* [13]. Čím rychleji se paprsek pohybuje, tím méně intenzivní připadá oku zmíněná linka. Bod je možné vykreslit, jestliže paprsek zůstane na jednom místě bez pohybu po úrčitou dobu. [1]

Vykreslení čáry je základní a nejjednoduší operací, jakou laserový projektor může vykonat. Například k vykreslení Úsečky z bodu A do bodu B projektor nasměruje laserový paprsek na bod A, zapne laser a pohybuje paprskem k bodu B. [1]

K vykreslení složitějších obrázků jsou potřeba tzv. blank lines, kdy projektor otáčí zrcátka stejně, jako kdyby vykresloval přímku, ale laser nesvítí. Blank lines spojují každé dvě vykreslené linky, které na sebe přímo nenavazují. [1]

Nestihne-li projektor vykreslovat obraz dostatečně rychle, výsledná projekce nebude stabilní. Lidské oko vždy uvidí pouze části obrazu v časové návaznosti. Tomuto jevu se říká „flickering“. [1]

Kapitola 4

Použité technologie, techniky, pojmy

4.1 GitHub

Github je online platforma, která umožňuje vývojářům ukládat a sdílet jejich kód. Na githubu je kód ukládán v repozitářích. V repozitářích je kromě samotného kódu uložena i historie změn.

Jakýkoliv jiný uživatel si může z githubu stáhnout repozitář, pozměnit ho a následně autorovi navrhnut začlenění jeho změn do původního repozitáře.

Celá výsledná softwarová výbava projektoru byla nahrána na server github.com do repozitáře [phuid/laser_projector](https://github.com/phuid/laser_projector). Do tohoto repozitáře budou nahrány i jakékoliv pozdější změny softwarové výbavy.

4.2 Node.js

Node.js je open-source runtime (česky běhové prostředí) pro javascript, které umožňuje vývojářům spouštět javascriptový kód mimo prohlížeč [14]. Umožňuje tedy programovat kód pro server ve stejném jazyce, který běží v prohlížeči. Díky tomu je oblíbený mimo jiné pro tvoření serverů hostujících webové stránky.

4.3 SPI

SPI (anglicky Serial Peripheral Interface) je komunikační rozhraní používané k přenosu dat mezi ovládajícím zařízením (často mikrokontrolerem) a jedním, nebo více periferními integrovými obvody (periferiemi). Používá separátní vodič (linky) pro hodinový signál (SCK), kterým mikrokontroler ovládá rychlosť přenosu dat, a datové linky. Datové linky jsou fixně využívané k posílání dat buď periferií – linka POCI (Peripheral Out, Controller In), nebo mikrokontrolerem – linka PICO (Peripheral In, Controller Out). Dále používá separátní vodič(e) pro výběr periferie, se kterou mikrokontroler komunikuje. Tomuto vodiči se říká chip select (zkráceně CS). Pro každou periferii je potřeba připojit separátní CS vodič. [15]

4.4 I²C

I²C (anglicky Inter-Integrated Circuit) je komunikační protokol umožňující komunikaci mezi jednou nebo více periferiemi a jedním, nebo více mikrokontrolery. Podobně jako SPI je úřčen ke komunikaci na krátké vzdálenosti v jednom zařízení. Narozdíl od SPI využívá pouze dva vodiče (zvané SDA a SCL) k výměně informací včetně výběru připojené periferie, se kterou mikrokontroler chce komunikovat. Zároveň umožňuje fungování systémů s několika kontrolery. [16]

Dva vodiče při použití tohoto protokolu stačí i na výběr zařízení, se kterým chce mikrokontroler komunikovat. Po začátku komunikace totiž sběrnicí¹ mikrokontroler pošle tzv. adresu zařízení se kterým chce komunikovat [16]. Adresa zařízení je sestavena ze sedmi bitů a může tedy dosahovat 127 různých hodnot (kromě systémů s 10bitovou adresou; Ty jsou ale vzácné). Většinou je pro danou periferii nastavena od výroby, u některých je ale možné ji softwarově změnit.

4.5 Pull-up a pull-down rezistory

Note:
possibly
zbytecne
TODO:
oddelat

Pokud při čtení z GPIO pinů mikrokontrolerů či jiných zařízení k danému pinu není nic připojeno, je těžké předpovědět, jaký stav na tomto pinu přečteme. Když tento fenomén nastane, říká se, že je pin floating (plovoucí) [17]. Může nastat například, chceme-li číst stav tlačítka, přes které by byl pin připojen na zem. Když tlačítko není zmáčknuté, pin na zem připojen není a tudíž je floating.

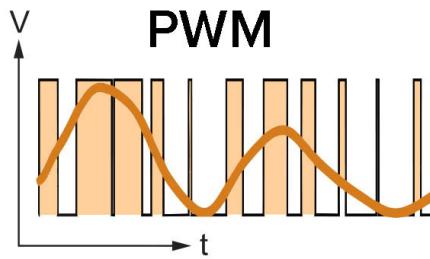
V takových případech se k pinu připojuje tzv. pull-up rezistor, který zaručí, že z pinu v případě, že tlačítko stisknuté není, přečteme hodnotu HIGH. Rozdíl mezi pull-up a pull-down rezistory je pouze v tom, že pull-up rezistory připojují pin k napětí, na kterém zařízení pracuje (často 3,3 nebo 5 V) a pull-down rezistory rezistory připojují pin k zemi. [17]

4.6 PWM (Pulse Width Modulation)

PWM (z anglického Pulse Width Modulation) se využívá jako alternativa analogového řízení v případech, kdy je potřeba řídit analogovou proměnnou binárním signálem, tedy signálem nabývajícím hodnot „zapnuto/vypnuto“. V PWM signálu je konstantní perioda a proměnný je čas, kdy má binární signál hodnotu „zapnuto“, tomuto času se říká „pulse width“ a vyplývá z něj jméno techniky. Konečná hodnota („střída“, anglicky „duty cycle“) signálu se dá získat jako poměr času „pulse width“ a periody signálu. Střída hodnoty 100 % tedy znamená, že signál má neustále hodnotu „zapnuto“, střída hodnoty 0 % naopak znamená neustálé „vypnuto“. [18][19]

Výsledný signál při využití této techniky je ilustrován na obrázku 4.1. V reálném využití bývá frekvence PWM signálu v rádech kHz.

¹Sběrnice I²C je systém zařízení propojený protokolem I²C na dvou stejných vodičích.



Obrázek 4.1: PWM signál s měnící se střídou; Střída naznačena oranžovou křivkou.
Převzato a upraveno z [20]

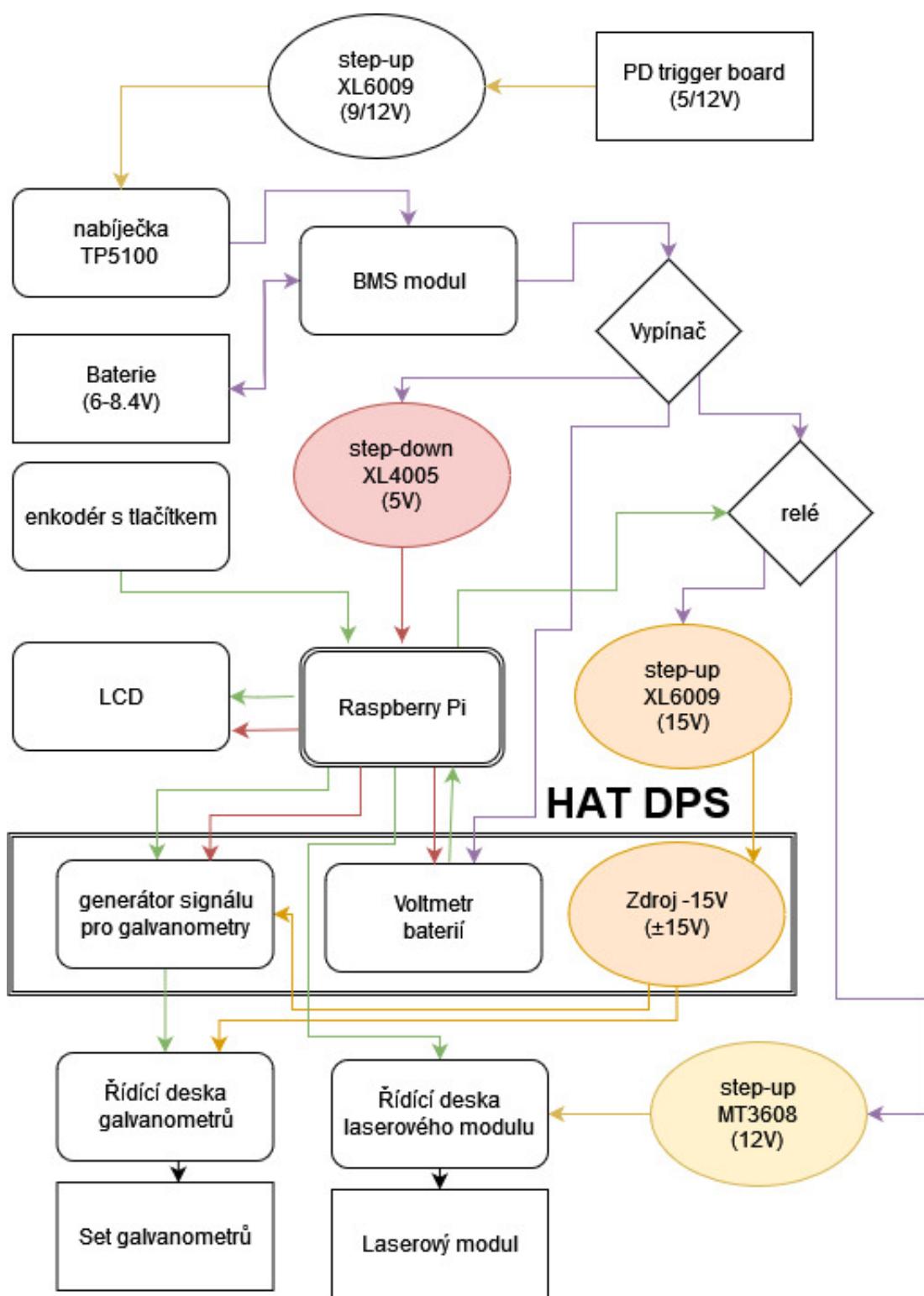
4.7 Přerušení (Interrupt)

Přerušení nastává, nastane-li za běhu programu náhle potřeba obsloužit hardwarovou událost. Procesor při něm přeruší vykonávání sledu instrukcí, obslouží přerušení a poté pokračuje v předchozí činnosti [21].

Kapitola 5

Hardware

Tato kapitola se zabývá fyzickou konstrukcí a zapojením vyrobeného laserového projektoru. Ten se skládá z řídící jednotky, galvanometrů s ovládací elektronikou, laseru, chlazení a napájení. Všechny tyto součástky jsou uloženy v pouzdře vytisknutém na 3D tiskárně. Celkové schéma zapojení je vidět na obrázku 5.1.



Obrázek 5.1: Celkové schéma zapojení hardwarových komponentů; Signálové linky jsou znázorněny zeleně, napětí z baterie žlutavě, napětí 5 V červeně a další napětí odstíny oranžové.

5.1 Řídící jednotka – Raspberry Pi

Jako hlavní řídící jednotka byl použit jednodeskový počítač Raspberry Pi, který je vidět na obrázku 5.2. To hned z několika důvodů:

- Jednoduché připojení k Wi-Fi sítím — Raspberry Pi je možné připojit k blízké Wi-Fi síti nebo si může vytvořit vlastní, na které funguje jako wifi přístupový bod (access point).
- Operační systém Raspberry Pi OS — Běží na něm operační systém založen na Linuxu, je pro něj jednoduché vytvářet programy a také programy můžou jednoduše interagovat s potenciálně připojenou klávesnicí, myší nebo monitorem.
- Vysoký výkon — Oproti mikrokontrolerům nabízí podstatně vyšší výkon, ten se může hodit, potřebujeme-li spouštět několik programů zároveň nebo chceme-li procesovat video z kamery.

Raspberry Pi na sobě má 40 pinový GPIO header, skrze který je možné interagovat s připojenými čipy a moduly.

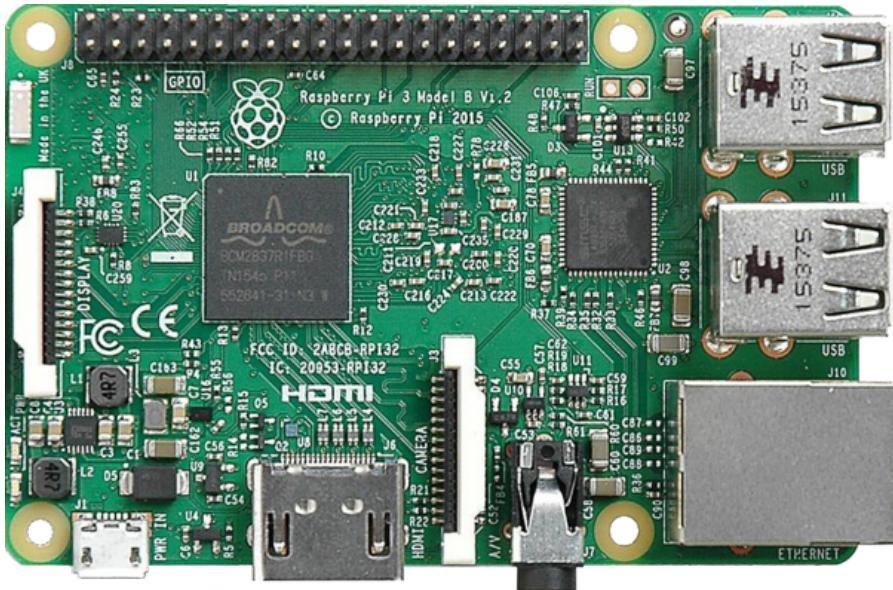
5.2 Set galvanometrů se zrcátky

5.2.1 Výběr skeneru

Pro tuto práci byl vybrán galvanometrový skener, protože je nejdostupnější a protože potenciálním uživatelům nejlépe představí technologii.

Oproti hranolovým skenerům jim toží dívá více možností, jak s paprskem pohybovat. Můžou se rozhodnout, že jej využijí jako hranolový skener, pokud nahrají soubor procházející promítací plochu po rádcích.

Oproti dalším typům skenerů je názornější, ostatní typy skenerů jsou totiž příliš malé a není na nich vidět princip funkce nebo je jejich fungování nadmíru abstraktní a těžko pochopitelné.



Obrázek 5.2: Pohled zvrchu na Raspberry Pi [22]

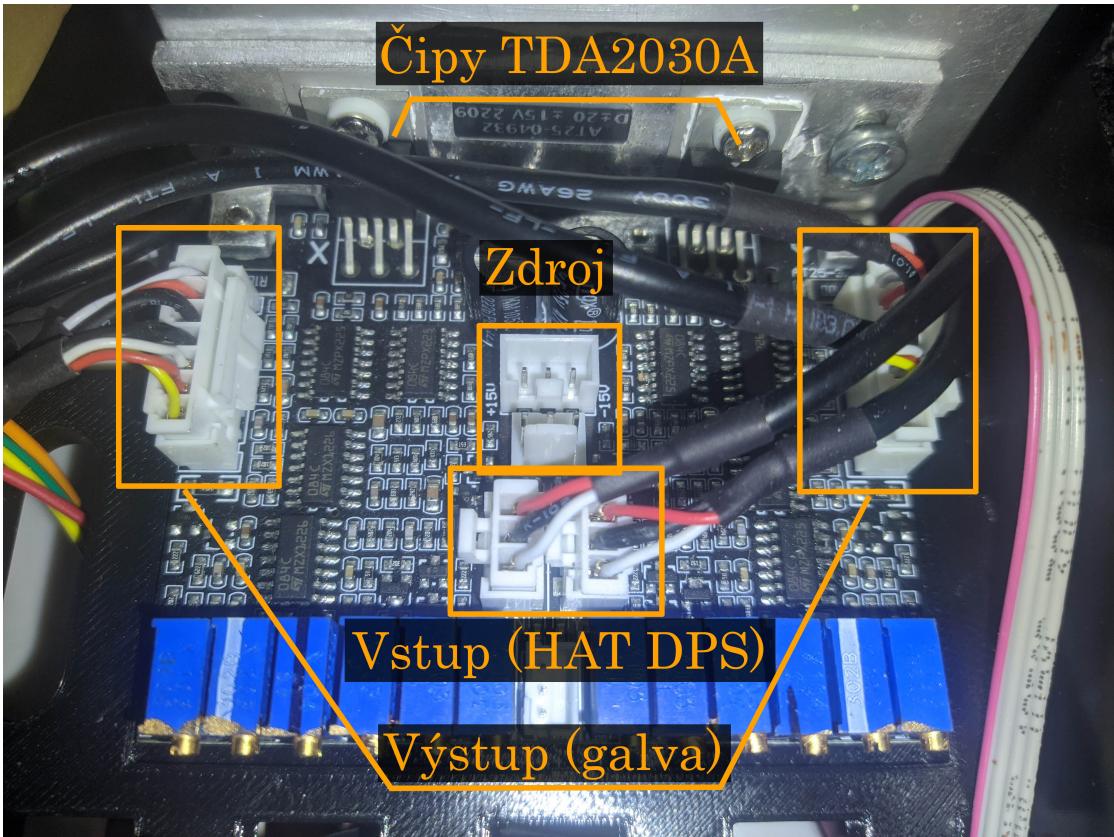
5.2.2 Zapojení galvanometrového setu

Samotné galvanometry jsou zapojeny do řídící desky, která s nimi byla zakoupena, ta je vidět na obrázku 5.3.

Řídící deska požaduje symetrický zdroj napětí 15 V, tzn. +15 V a -15 V a samozřejmě připojení k zemi. Také přijímá dva bipolární diferenciální analogové signály s rozsahem diferenciálního napětí -10 V až +10 V. Každý signál udává vychýlení jednoho ze dvou galvanometrů, což obvykle znamená výslednou pozici laserového paprsku v osách X a Y.

5.2.3 Bipolární diferenciální analogový signál

Diferenciální signál je signál přenášený dvěma vodiči, každý z nich přenáší stejný signál, jen s opačnou polaritou. Vodič označený (+) je považován za nosič základního signálu, zatímco vodič označený (-) je považován za nosič invertovaného signálu. Výsledné diferenciální napětí je napětí na základním nosiči vůči napětí na obráceném nosiči, tzn. $V_{dif} = V_{(+)} - V_{(-)}$. [23]



Obrázek 5.3: Řídící deska galvanometrů s vyznačenými konektory a hřejícími čipy

Bipolární signál znamená, že na napětí každém z vodičů (+) a (−) může dosahovat kladných i záporných hodnot [23].

Tudíž cheme-li disáhnout diferenciálního napětí $+10\text{ V}$, musí mít základní signál napětí $+5\text{ V}$ a obrácený signál -5 V . Záporné diferenciální napětí bude ve chvíli, kdy je napětí základního signálu záporné a napětí obráceného signálu kladné.

5.2.4 Zahřívání čipů řídící desky galvanometrů

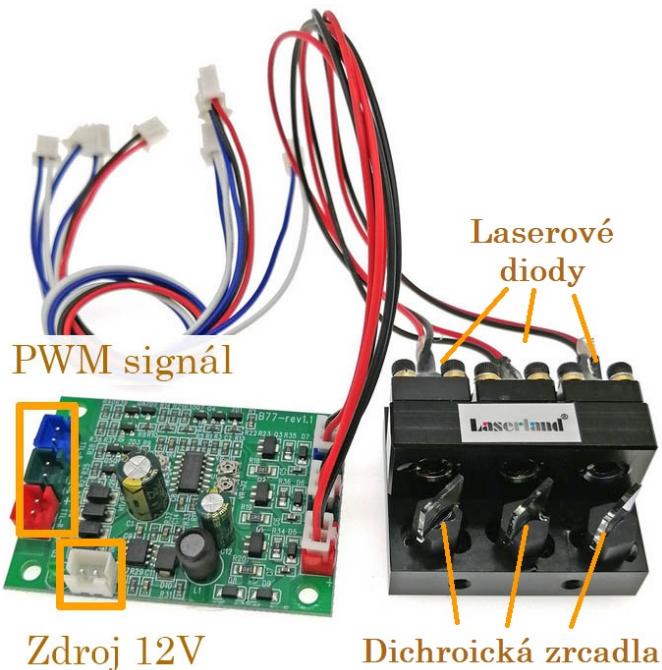
Dva z čipů na řídící desce se při chodu systému výrazně zahřívají. Na tyto čipy je naštěstí už od výroby desky připevněna malá hliníková destička. Ta má sloužit jako chladič, ale i s ní se čipy v otevřeném prostoru zahřívají na teploty blízké $60\text{ }^{\circ}\text{C}$. Dva zmíněné čipy jsou čipy TDA2030A od firmy STMicroelectronics. Ty by mely

dle datasheetu vydržet až 150 °C, ale dá se předpokládat, že v uzavřeném pouzdru budou čipy dosahovat vyšších teplot, než v otevřeném prostoru. I kdyby nedosáhly pro sebe kritických 150 °C, rozhodně není žádoucí, aby uvnitř projektoru desky dosahovaly vysokých teplot.

I proto byl do projektoru zabudován chladič. Více o způsobu jeho připevnění a distribuci chlazení mezi ostatní komponenty se dočtete v kapitole 5.8.1.

5.3 RGB laserový modul

Jako zdroj laserového paprsku byl využit RGB laserový modul, skládající se ze řídící desky, tří barevných laserových diod o vlnových délkách 660 nm (červená), 450 nm (modrá) a 520 nm (zelená) a tří dichroických zrcátek, která slouží ke spojení paprsků z diod do jednoho. Celý modul je vidět na obrázku 5.4.



Obrázek 5.4: Laserový modul s řídící deskou a vyznačenými konektory

5.3.1 Dichroická zrcadla

Dichroická zrcadla jsou zrcadla s výrazně rodílnými odrazovými nebo průchodo-vými vlastnostmi pro dvě různé vlnové délky odraženého/procházejícího světla [24].

Většina dichroických zrcadel jsou dielektrická zrcadla, která se skládají z mnoha tenkých vrstev různě opticky propustných materiálů. Existují ale také krystalická zrcadla, jež odrážlivá vrstva se skládá z monokrystalického materiálu, typicky polovodiče [24].

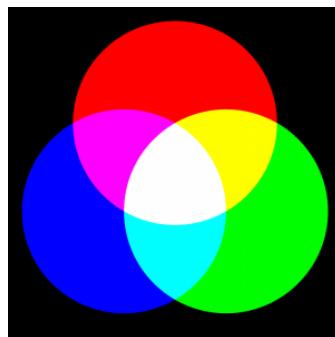
Mezi jejich využití spadá například oddělování infračerveného záření při aplikacích, kdy je nežádoucí zahřívání ozařovaného objektu [24].

5.3.2 Zapojení laserového modulu

K modulu je od výroby připojená řídící deska. Ta požaduje napětí 12 V a přijímá binární signál pro každou diodu. Je-li na signál připojena zem, korespondující dioda nesvítí. Je-li na něj připojeno 2,7 až 5 V, korespondující dioda svítí.

5.3.3 Tvorba barev s laserovým modulem

S laserem je tedy možné vytvořit 7 barev – červenou, zelenou, modrou, žlutou, tyrkysovou, purpurovou a bílou, viz obrázek 5.5.



Obrázek 5.5: Sedm základních barev laserového modulu

Naštěstí i zde je možné uplatnit jev persistance of vision a sice pomocí techniky PWM popsané v kapitole 4.6. Nastavíme-li tedy střídu signálu pro červenou diodu na 100 % a pro zelenou diodu na 50 % výsledný paprsek bude pro lidské oko mít barvu s rgb hodnotou (255, 127, 0) neboli oranžovou.

I tato technologie má ovšem své limitace, řídící deska laserového modulu zvládá přijímat signál o maximální frekvenci 35 kHz (Raspberry Pi je schopno vysílat s frekvencí až 40 kHz), což vzhledem k rychlosti pohybu laseru v některých případech nemusí stačit. Může se stát, že při vykreslování křivky se paprsek stihne posunout dříve, než uplyne perioda pwm signálu. Pokud se tak stane, budou různé sousední části křivky mít různou barvu.

Tento efekt je nejviditelnější při projekci tmavých barev, ale dá se zmírnit zvětšením času, po který paprsek setrvá na jednom bodě po jeho vykreslení, v tu chvíli se ale může stát, že nastane „flickering“ popsaný v sekci 3.2.

Note:
TODO
obrázek
nedosta-
tečně
rychleho
pwm

5.4 Displej z tekutých krystalů (LCD)

Pro zobrazování informací uživateli přímo na zařízení byl využit alfanumerický¹ LCD s řadičem HD44780 a s rozlišením 20 x 4 znaky. K displeji je také připojen I²C převodník, který slouží jako prostředník mezi řadičem displeje a Raspberry Pi. Komunikační protokol LCD totiž využívá podstatně více vodičů, než I²C sběrnice, kterou Raspberry Pi komunikuje s převodníkem.

5.5 Rotační enkodér

Rotační enkodér je typ pozičního senzoru používaný k měření rotace otáčivé hřídele [26]. Existuje mnoho druhů enkodérů, rozdělují se dle signálu, který vydávají a dle technologie, kterou měří rotaci hřídele [26]. V této práci je použit mechanický inkrementální enkodér s tlačítkem.

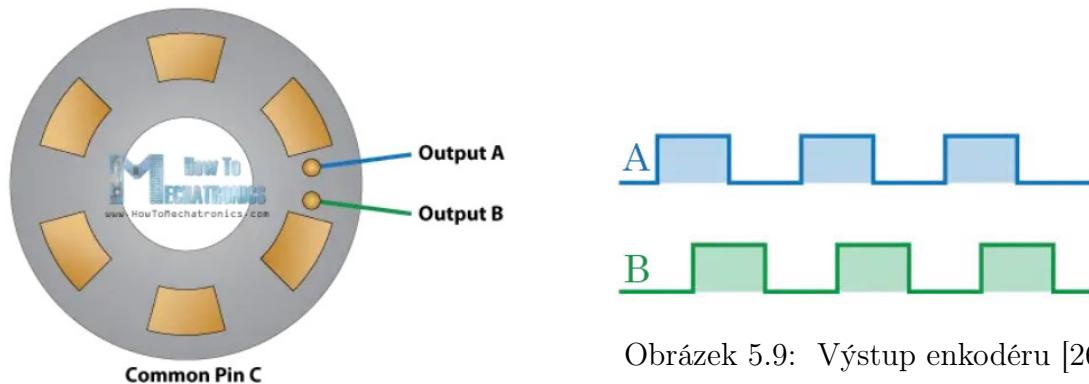
¹Alfanumerický – Řídící jednotka displeji místo pixelů posílá celé znaky, které sám vykresluje.



Obrázek 5.6: Displej z tekutých kryštalů (LCD); Převzato a upraveno z [25]

Obrázek 5.7: I²C převodník napojený na LCD [25]

Na obrázku 5.8 je vidět, jak enkopodér funguje uvnitř. Dva kontakty A a B při rotaci získávají a ztrácí spojení s kontaktem C. Připojíme-li ke kontaktu C zem a ke kontaktům A a B pull-up rezistory (klidně softwarově), dá se toto získávání a ztrácení kontaktu zaznamenat do grafu na obrázku 5.9 jako dva signály obdélníkového průběhu vzájemně fázově posunuté o 90 stupňů.



Obrázek 5.8: Vnitřní schéma enkopodéru [26]

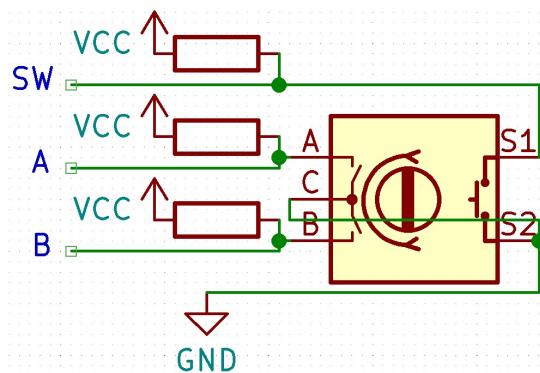
Obrázek 5.9: Výstup enkopodéru [26]

Použitý rotační enkopodér má další dva kontakty připojené k tlačítku pod rotující hřídelí. Ke čtení stisknutí tlačítka je potřeba připojit jeden kontakt k zemi, tedy

ke kontaktu C a druhý kontakt na pull-up rezistor (klidně softwarově). Celé zapojení enkodéru je naznačeno na obrázku 5.10. Kontakty tlačítka jsou v něm označeny S1 a S2.

5.5.1 Čtení pozice z rotačního enkodéru

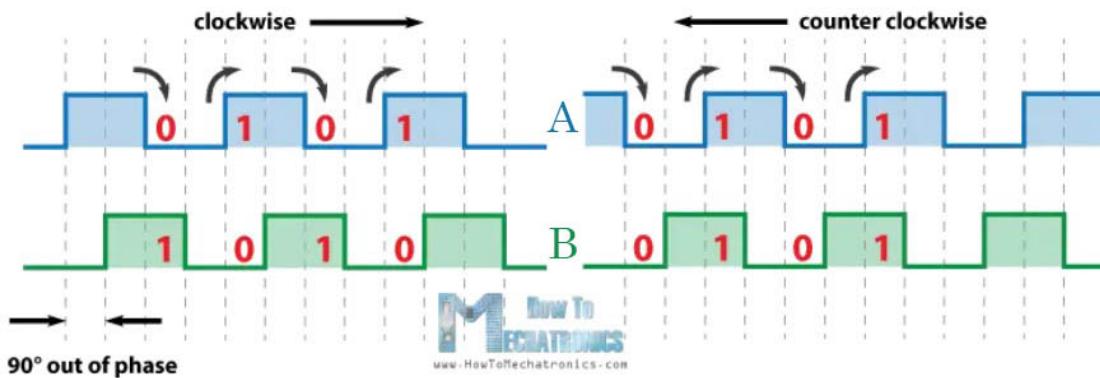
U rotačního enkodéru se při každé otáčce změní připojení pinů několikrát. Chceme-li pozorovat pouze počet těchto změn, stačí spočítat změny na jednom kontaktu. Pokud je ale zapotřebí pozorovat i směr otáčení, je nutné pozorovat stav obou kontaktů. Pokud se enkodér otáčí po směru hodinových ručiček, kontakt A bude fázově posunut o 90 stupňů napřed oproti kontaktu B. Pokud se enkodér otáčí proti směru hodinových ručiček, bude naopak kontakt B o 90 stupňů napřed oproti kontaktu A. Časový průběh stavu kontaktů při otáčení je naznačen na obrázku 5.11.



Obrázek 5.10: Schéma zapojení rotačního enkodéru.

5.6 HAT deska plošných spojů

Pro ovládání výše popsaného hardwaru je zapotřebí několik specifických obvodů. Kvůli jejich specifičnosti tyto obvody nejsou volně dostupné k zakoupení na předem vytvořených destičkách. Proto bylo zapotřebí je z jednotlivých součástek vyrobit na míru.



Obrázek 5.11: Časový průběh stavu kontaktů rotačního enkodéru při otáčení hřídelí na obě strany

Obvody byly navrženy v programu KiCad, celé schéma desky je k nalezení v příloze 1 na konci tohoto dokumentu. Další výrobní dokumenty z programu KiCad jsou k nalezení v souborové příloze ve složce „pcb“. Následně pro ně v tomtéž programu byla nadesignována deska plošných spojů. Mezi obvody patří:

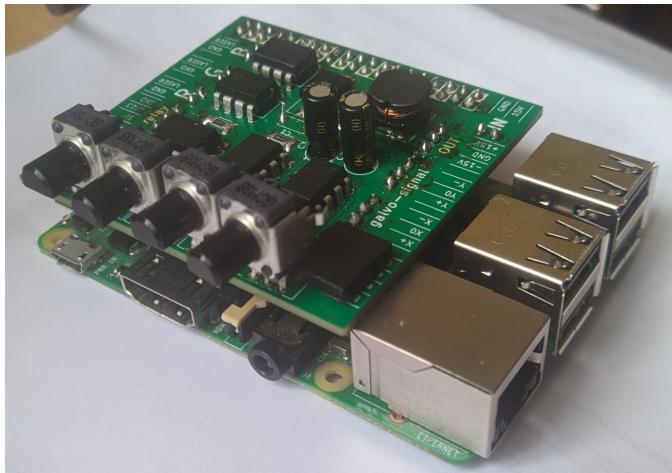
- Zdroj -15 V pro galvanometry.
- Generátor signálu pro řídící desku galvanometrů.
- Voltmetr baterií.

Kromě nich byly na desku přidány konektory k jednotlivým barevným vstupům laseru, LCD displeji a k rotačnímu enkodéru, které jsou přímo napojeny na 40 pinový GPIO konektor Raspberry Pi. Deska byla designována jako tzv. HAT, to znamená, že sama na tomto konektoru drží a nezabírá o moc více místa, než samotné Raspberry Pi. Deska připevněná na RPi je vidět na obrázku 5.12

5.6.1 Zdroj -15 V

Napětí -15 V je získáváno obvodem napěťového invertoru založeném na obvodu ze zdroje [27]. Jeho zapojení je na obrázku 5.13, potřebuje zdroj napětí $15V$.

V aktuálním stavu tento zdroj nefunguje, to bude brzo napraveno.



Obrázek 5.12: HAT DPS připevněná na Raspberry Pi

Centrem obvodu je integrovaný obvod AOZ1282 od výrobce Alpha & Omega Semiconductor označený U6. Tento integrovaný obvod obsahuje spínací transistor (ve zjednodušeném schématu prvek SW), PWM regulační obvod pracující na frekvenci 450 kHz s napěťovou referencí 0,8 V, který reguluje čas připojení induktoru L1. K němu je připojen bootstrapový kondenzátor C1, ten zajišťuje plovoucí buzení pro integrovaný spínač. Dále je k němu připojený výkonový induktor L1, jehož hodnota byla zvolena dle rovnic uvedených ve zdroji [28], respektive podle dále uvedené rovnice 5.1. [27]

$$L = \frac{-U_{OUT} \times U_{IN}}{0,4 \times 2 \times I_{OUT} \times f_s \times (U_{IN} - U_{OUT})} = \frac{-(-15V) \times 15V}{0,4 \times 2 \times 1A \times 450kHz \times (15V - (-15V))} \approx 21\mu H$$

Rovnice 5.1: Výpočet ideální indukčnosti cívky pro invertující obvod

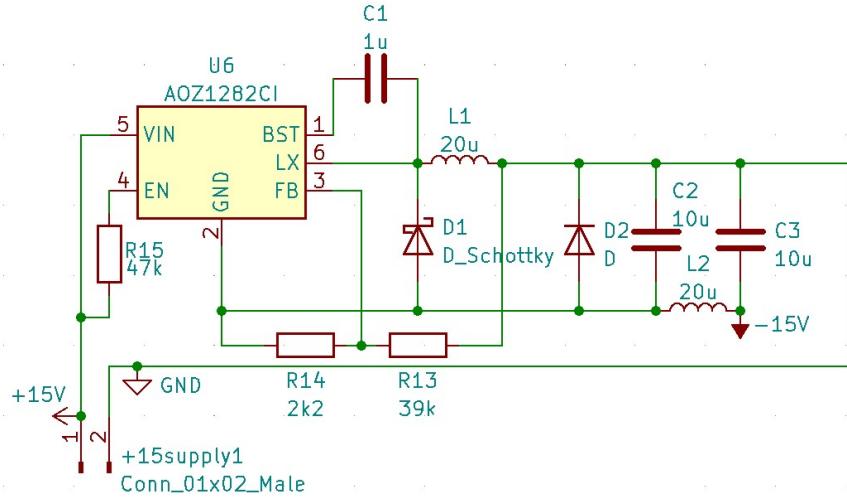
L — Indukčnost spínaného induktoru

U_{IN} — Vstupní napětí do invertujícího obvodu

U_{OUT} — Výstupní napětí z invertujícího obvodu

I_{OUT} — Výstupní proud z invertujícího obvodu

f_s — Frekvence spínacího regulátoru



Obrázek 5.13: Zapojení invertujícího obvodu

Na FB pin integrovaného obvodu je připojen napěťový dělič tvořený odpory R13 a R14, který integrovanému obvodu dodává zpětnou vazbu o výstupním napětí. Hodnoty R13 a R14 jsou voleny tak, aby při 15 V, tedy požadovaném výstupním napětí, bylo na výstupu děliče napětí 0,8 V, tedy referenční napětí integrovaného spínaného regulátoru. Schottkyho dioda SS56, označená D1, slouží k zadržení změny polarity induktoru. Usměrňovací dioda D2 je v propustném stavu při pravotním spuštění měniče, kdy je přes ní napájen U1 po dobu náběhu výstupního záporného napětí. Posledním prvkem je výstupní vyhlazovací filtr tvořený kondenzátory C3 a C4 společně s induktorem L2. Jeho úkol je minimalizovat výstupní napěťové zvlnění zdroje. [27]

5.6.2 Generátor analogového signálu

Jak je popsáno v sekci 5.2, řídící deska galvanometrů přijímá dva bipolární diferenční analogové signály v rozpětí -5 V až $+5 \text{ V}$.

Obvod, který se stará o vytváření tohoto signálu, je založený na obvodu ze zdroje [29]. Vytváření tohoto signálu je rozděleno do dvou částí. Nejdříve D/A převodník připojený k RPi vytvoří signál v rozpětí 0 až 5 V a následně je tento signál pomocí

invertujících operačních zesilovačů převeden na požadované rozpětí a invertován. Jednotlivé části tohoto obvodu jsou blíže popsány v následujících kapitolách.

D/A převodník

K generování signálu v rozpětí 0–5 V byl využit dvoukanálový D/A převodník² MCP4822. Tento čip podporuje komunikaci přes rozhraní SPI, pracuje s napájecím napětím 5 V a s 12bitovým rozlišením (je schopen vygenerovat 4 096 různých napětí) na dvou kanálech [30]. RPi komunikuje s čipem pomocí rozhraní SPI popsaném v kapitole 4.3.

Operační zesilovače

K modifikaci signálu z DAC na bipolární diferenciální analogový signál slouží pro každý kanál jeden čip TL082 [31], který obsahuje dva operační zesilovače. Ty jsou zapojeny dle schématu na obrázku 5.14.

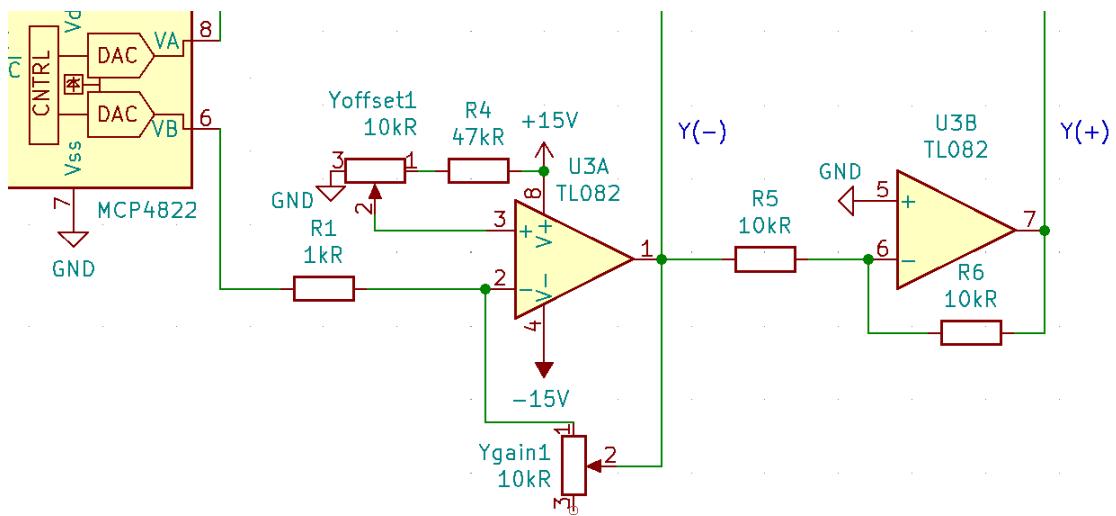
Signál první operační zesilovač zesílí/zeslabí a vertikálně posune dle nastavení potenciometrů Ygain (zesílení) a Yoffset (posun) a zároveň invertuje. Tento invertovaný signál následně druhý operační zesilovač opět invertuje. Tím získavá základní signál pro řídící desku galvanometrů.

5.6.3 Voltmetr baterií

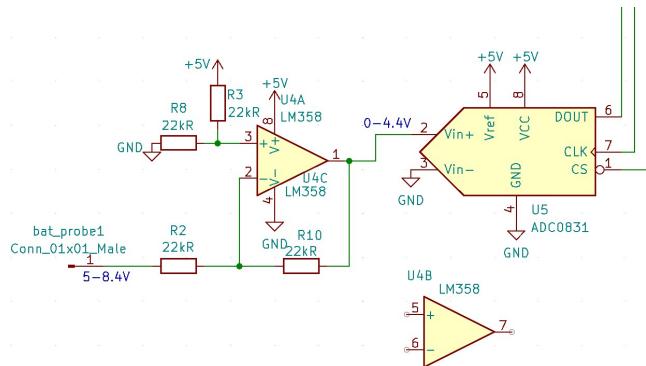
Obvod voltmetru baterií je vidět na obrázku 5.15.

Jako voltmetr baterií slouží A/D převodník ADC0831 od firmy Texas Instruments Incorporated [32]. Ten je označen U5 a je zapojen společně s operačním zesilovačem LM358 od stejného výrobce, který je označen U4. Operační zesilovač je zapojen jako rozdílový zesilovač podle zdroje [33] tak, aby od napětí baterií, které se může pohybovat v rozsahu 6 V až 8,4 V, odečítal 5 V. Díky tomu se napětí, která měří A/D převodník pohybují mezi hodnotami 1 V a 3,4 V. S jeho osmibitovým

²D/A převodník je obvod, který na základě instrukcí přijatých digitálně generuje analogové napětí.



Obrázek 5.14: Zapojení čipu TL082 pro jeden kanál řídící desky galvanometrů



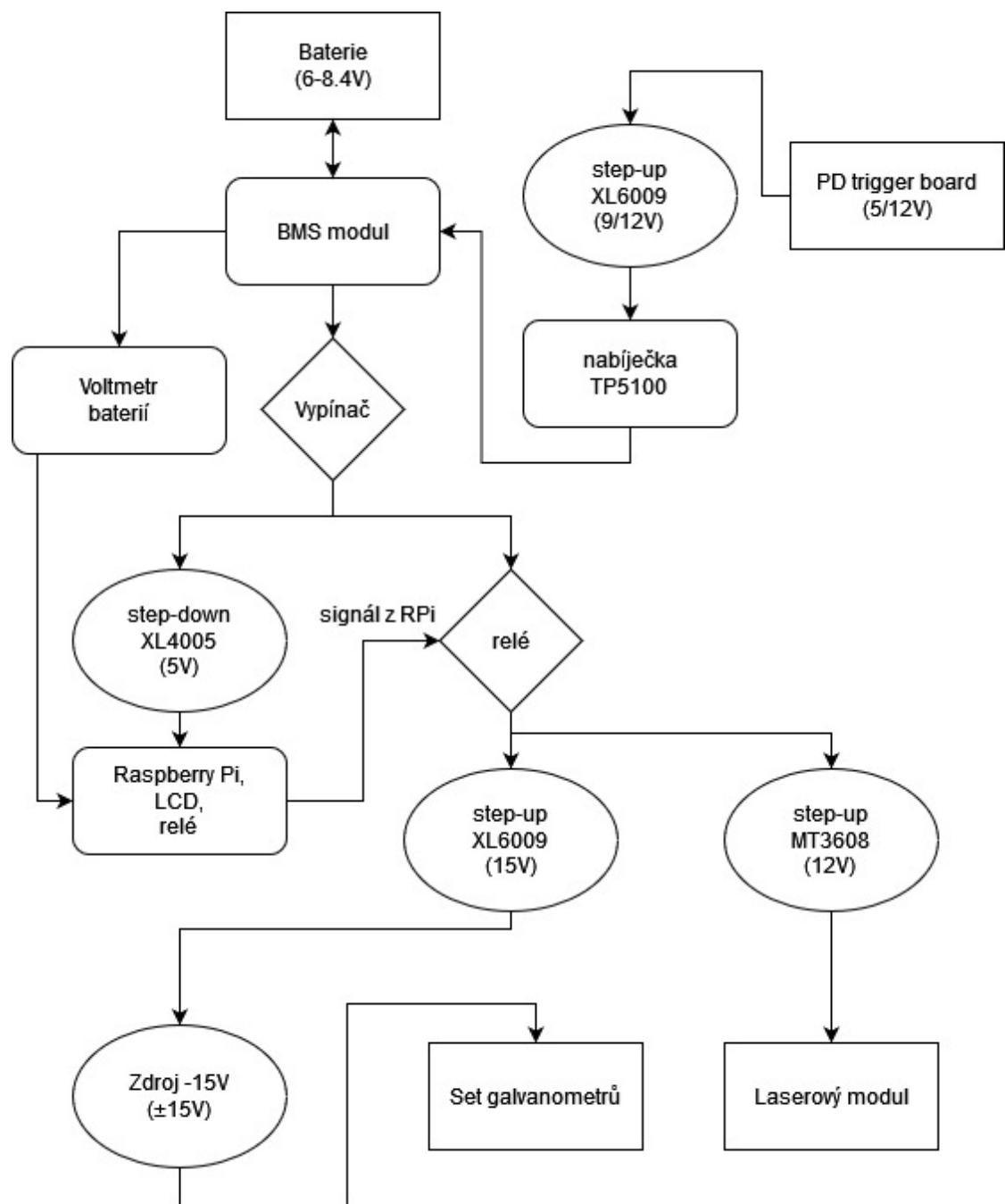
Obrázek 5.15: Schéma obvodu voltmetu baterie

rozlišením a referenčním napětím 5 V v tomto rozpětí může naměřit 122 různých napětí.

A/D převodník tyto data posílá do Raspberry Pi pomocí rozhraní SPI a to s nimi dále pracuje.

5.7 Napájení

Celkové schéma napájení je vidět na obrázku 5.16.



Obrázek 5.16: Celkové schéma napájení komponentů projektoru

5.7.1 Akumulátory

K napájení projektoru byly využity 4 Lithium-iontové akumulátory Samsung INR 18650 s kapacitou 3 450 mAh a jmenovitým napětím 3,7 V. Ty byly zapojeny nejdříve po dvojicích paralelně a následně byly tyto dvojice zapojeny sériově. Konečný článek tedy dosahuje jmenovitého napětí 7,4 V.

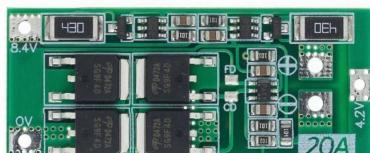
5.7.2 BMS modul

Na baterie byl napojen ochranný BMS (Battery management system) modul, který ji chrání před následujícími stavami:

- odběr vysokého proudu (zkrat),
- přebití,
- vybití,
- nevybalancované články.

Modul články balancuje a v případě, že nastane jiný z nežádoucích stavů, ji odpojí.

Modul je na obrázku 5.17



Obrázek 5.17: BMS Modul se třemi kontakty pro sérii baterií (0V, 4.2V a 8.4V) a výstupními kontakty ((+) a (-)) [34]



Obrázek 5.18: Relé modul [35]

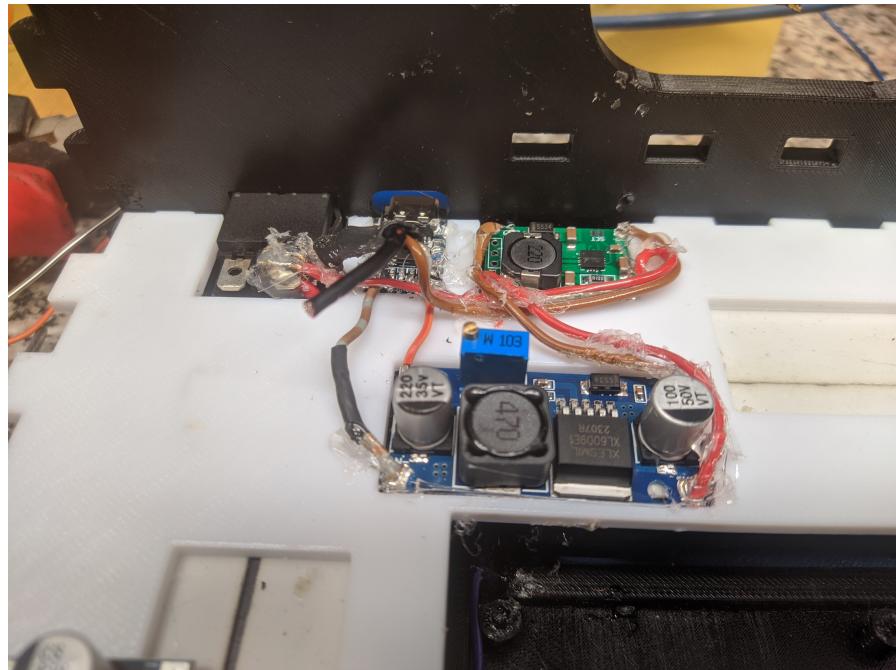
5.7.3 Relé modul

V projektoru byl využit relé modul pro připojování komponentů s vysokým odběrem pouze ve chvílích, kdy jsou využívány. Jedná se o set galvanometrů, laserový modul a větrák, ty jsou připojeny jen ve chvílích, kdy projektor promítá.

Relé modul je ovládán jedním kontaktem spojeným s Raspberry Pi a je umístěn mezi bateriemi a měniči napětí, proto stačí pouze jeden na více napěťových větví. Je vidět na obrázku 5.18

5.7.4 Nabíjecí obvod

Zapojený nabíjecí obvod je vidět na obrázku 5.19.



Obrázek 5.19: Zapojený nabíjecí obvod

Power Delivery (PD) trigger deska

K nabíjení baterie je využíván USB-C port podporující moderní protokoly rychlého nabíjení (hlavně Power Delivery a Quick Charge). Ten se nachází na desce

s integrovaným obvodem, který přes port komunikuje s adaptérem v zásuvce, pokud adaptér podporuje rychlé nabíjení, čip od něj vyžadá napětí 12 V, které deska převádí na výstupní kontakty viditelné na obrázku 5.20. Protože deska „vyvolá“ dané napětí, označuje se PD trigger board (deska).



Obrázek 5.20:
Power Delivery
trigger board [36]



Obrázek 5.21: step-up měnič
s čipem XL6009 [37]



Obrázek 5.22: Modul
nabíječky dvou sériově
zapojených Li-ion bate-
rií [38]

Step up měnič s čipem XL6009

Pokud ovšem připojený adaptér nepodporuje žádný rychlonabíjecí protokol, na výstupech desky bude napětí pouze 5 V, na kterém standartně běží USB připojení. Proto je k PD trigger board připojen step-up měnič nastavený na 9 V. Pokud z PD desky bude vycházet napětí 5 V, step-up jej zvýší na 9 V. Pokud z PD desky bude vycházet 12 V, napětí step-up projde beze změny.

TP5100

K ovládání průběhu nabíjení byl využit modul pro nabíjení Li-ionových baterií s čipem TP5100. Ten zajišťuje konstantní proud a napětí, které posílá na kontakty BMS obvodu. Další z jeho funkcí je automatické ukončení nabíjení ve chvíli, kdy baterie dosáhnou napětí 8,4 V. Je to jedinečný modul, který umožňuje nabíjení dvoučlánkových lithium-iontových akumulátorů.

5.7.5 Vypínač

K bateriím je neustále připojený jen BMS modul a nabíjecí obvod, všechny ostatní obvody jsou přemostěny vypínačem. Je tedy možné baterie nabíjet, i když jsou všechny ostatní obvody odpojené. Vypínač je vidět na obrázku 5.19.

5.7.6 Napěťové větve

Různé komponenty projektoru pracují s různými napětími. Je tedy potřeba napětí baterií převést na několik napěťových větví. Jedná se o větve:

- 5V — Napětí Raspberry Pi, LCD a relé modulu; Zajištěno step-down měničem s čipem XL4005 (viz obrázek 5.23.)
- 12V — Napětí Laserového modulu; Zajištěno step-up měničem s čipem MT3608 (Viz obrázek 5.24.)
- Symetrické napětí ± 15 V — Napětí řídící desky galvanometrů; Kladná větev je zajištěna step-up měničem s čipem XL6009 (viz obrázek 5.21.), záporná větev je zajištěna obvodem zdroje -15 V na HAT DPS (viz kapitola 5.6.1).



Obrázek 5.23: step-down měnič s čipem XL4005 [39]



Obrázek 5.24: step-up měnič s čipem MT3608 [40]

5.8 Pouzdro

V rámci popularizace technologie se může hodit projektor předvádět na různých místech. Proto bude potřeba, aby byl projektor přenosný a při přemisťování se nerozobil.

Pouzdro je tedy navrženo tak, aby bylo odolné proti nárazům a zachovalo všechny součásti v bezpečí. Každá součástka má své místo, kde je držena ze všech stran. Pro výrobu pouzdra by bylo vhodné využít technologii 3D tisku. Ta umožňuje tvorbu komplexních geometrických tvarů přímo pro potřeby konkrétního modelu a zároveň umožňuje snadnou iteraci a úpravu designu pouzdra při nalezení chyb.

Pouzdro bylo navrženo v programu Autodesk Fusion.

5.8.1 Priority designu

Chlazení

Největší část projektoru je hliníkový chladič s větrákem. Už od začátku práce byl tento chladič vybrán, aby byl připevněn k řídící desce galvanometrů. Problematika jejího zahřívání je popsána v kapitole 5.2.4. Jak bylo popsáno v této kapitole, na řídící desce galvanometrů je připevněna hliníková destička, která chladí čipy. Chladič byl připevněn právě na ni.

Aktivní chladič³ se ale hodí i pro ostatní součástky. Vzduch, který nasaje, je totiž distribuován celou vnitřní konstrukcí projektoru a chladí tak všechny vnitřní součástky. Tomuto proudění byla věnována zvláštní pozornost při designu konstrukce pouzdra.

Přístup k portům Raspberry Pi

Aby bylo možné je používat, je potřeba zajistit jednoduchý přístup k portům Raspberry Pi. Dále je potřeba od nich odlišit nabíjecí port.

Modularita, jednoduchá konstrukce

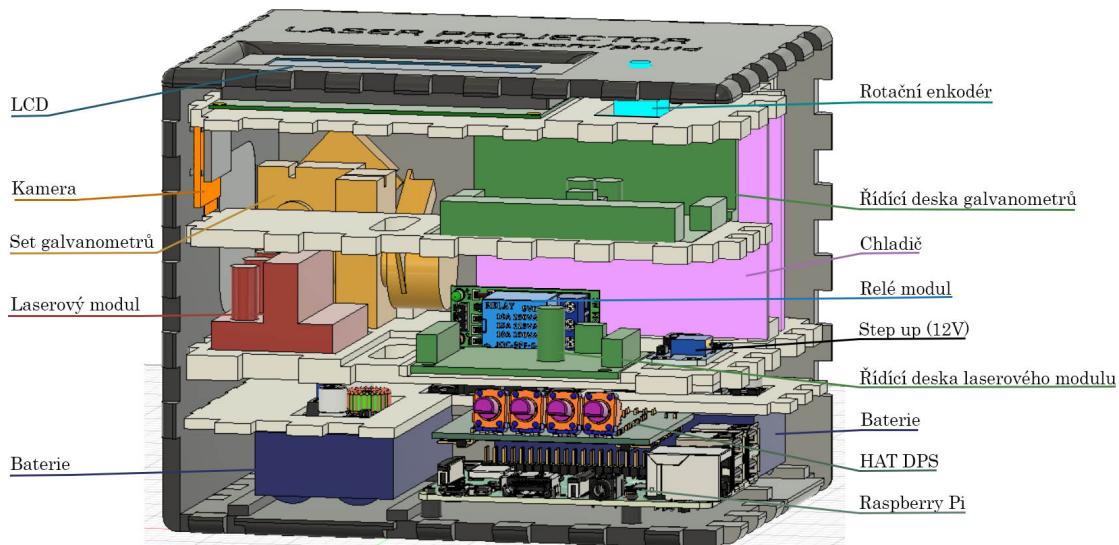
Pouzdro bylo designováno, také aby bylo modulární. Aby bylo možné při prototypování vyměnit pouze jednu součástku, která nesedí, místo tisknutí celého pouzdra

³Chladič s větrákom, který aktivně vytváří proud vzduchu.

od začátku. S modularitou bylo zároveň dosaženo jednoduché konstrukce, v jakékoliv části stavby je možné dočasně odstranit díly, aby bylo možné upravit připevnění vnitřní elektroniky.

5.8.2 Konstrukce

Pouzdro se skládá ze čtyř vertikálních stěn s otvory, do kterých zapadají horizontální desky. Horizontální desky v sobě mají vždy z vrchní strany vyhloubené prohlubně, do kterých pasují elektronické součástky, které na nich leží. Díky tomu se elektronické součástky při pohybu projektoru volně nepohybují ve vnitřních prostorách.



Obrázek 5.25: Pohled do projektoru s odstraněnou přední stěnou v programu Autodesk Fusion

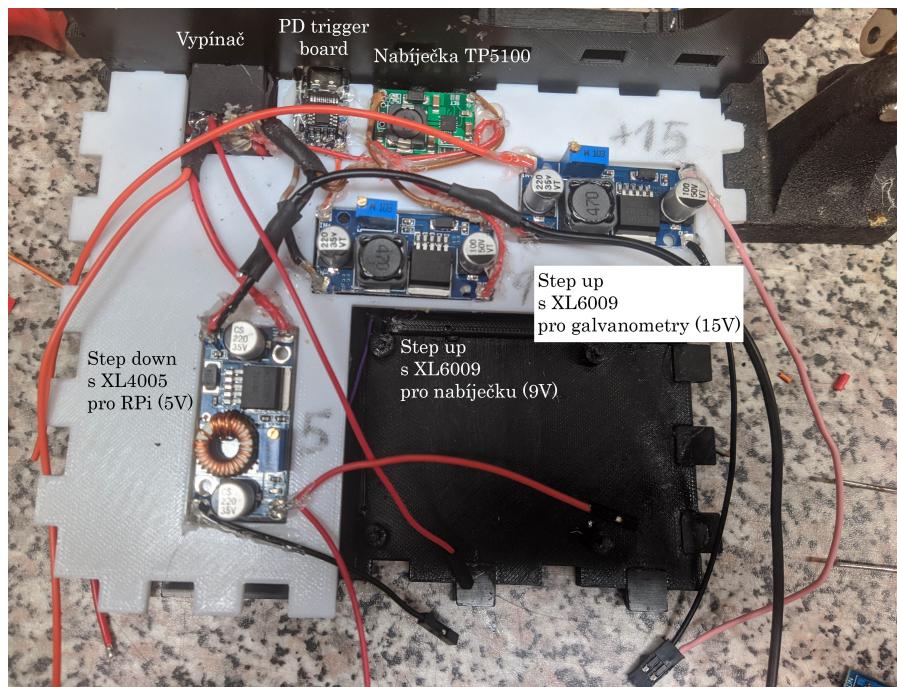
Tyto prohlubně jsou hlavní důvod, proč byl k výrobě dílů využit 3D tisk místo například laserového řezání.

Elektronika je v prohlubnách často držena i lepidlem z tavné pistole. To bylo přidáno s původním cílem upevnit k elektronice kabely z ní vedoucí i za izolaci. Kdyby

kabely držely jen za vodičové drátky k elektronice připájené, drátky by se v průběhu času kvůli vibracím při přenášení polámaly.

Jak je vidět na obrázku 5.25, pouzdro je rozděleno do pěti patr. Jednotlivá patra nezabírají celou horizontální plochu projektoru. Často spojují pouze tři ze čtyř stěn, nebo jsou v nich otvory, kterými může proudit vzduch. V prvním patře se nachází baterie (zvýrazněny modře), destička s BMS obvodem a v neposlední řadě Raspberry Pi. Na něm je připevněná HAT deska plošných spojů, která zasahuje do druhého patra. Druhé patro je vidět na obrázku 5.26 a nachází se v něm obvod nabíjení baterie (PD trigger deska, step up měnič a nabíječka Li-ion článků). Dále se v něm nachází step down měnič napájející Raspberry Pi a step up měnič napájející galvanometry.

Note:
tecka @
obr.5.26



Obrázek 5.26: Kompletně nainstalované druhé patro (Na obrázku je modul TP5100 zapojen s opačnou polaritou, ve výrobku byla chyba opravena, ale tato fotka je stále nejlepší ilustrace.)

Ve třetím patře je upeněn chladič, foukající na nabíjecí obvod a step up měniče pod ním, set galvanometrů (zvýrazněn žlutě), laserový modul (zvýrazněn červeně) a jeho řídící deska (zvýrazněna zeleně), step up měnič pro laser a relé modul. Galvanometry zasahují až do čtvrtého patra, kde je upevněna jejich řídící deska. Ta je mimo jiné připevněna na chladič párem šroubků viditelných na obrázku 5.3 ze strany 26, mezi desku a chladič byla aplikována teplovodivá pasta. V prostorách čtvrtého patra je také na stěnu upevněna kamera. V pátém patře se nachází LCD a rotační enkodér.

V horizontálních deskách oddělujících patra od sebe jsou otvory pro vzduch, jak je zmíněno v sekci 5.8.1, také na přední stěně pouzdra jsou otvory pro vyfukování vzduchu. Na přední stěně je také otvor pro přístup k potenciometrům na HAT DPS a otvor pro přístup k portům Raspberry Pi, ten je i na pravé boční stěně. Na zadní stěně je otvor pro nasávání vzduchu větrákem, vypínač, a otvory pro USB-C port PD trigger desky a statusové diody nabíječky TP5100. Na levé boční stěně jsou pak otvory pro kameru a laserový výstup galvanometrů. Ve dně jsou otvory pro výměnu baterií a ve stropní stěně jsou otvory pro LCD a rotační enkodér.

Note:
fotka
zepredu
a zprava



Obrázek 5.27: Pohled na projektor ze strany hrany sousedící se zadní a pravou boční stěnou

Kapitola 6

Software

Tato kapitola se zabývá softwarovou výbavou laserového projektoru. Popisuje klíčové programy, jejich funkce a způsob, jakým mezi sebou komunikují. Mezi tyto programy patří program lasershow, který obsluhuje vykreslování, programy pro interakci s uživatelem (dále „frontendové programy“) jako UI, web_ui a discord_bot, a také program wifi_manager, který spravuje wifi připojení Raspberry Pi. Programy wifi_manager a lasershow se dají označit za backendové programy, protože neinteragují s uživatelem.

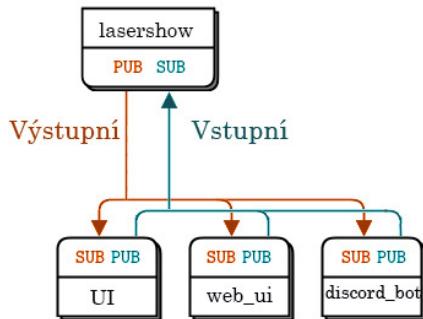
V neposlední řadě popisuje také instalací skript, který výrazně zjednoduší instalaci všech zmíněných programů a jejich závislostí.

Program lasershow umí číst projekční soubory ve formátu .ild. Uživatel může takový soubor vytvořit například v softwaru Laserworld Showeditor V7 [41] nebo může do frontendových programů nahrát soubor v populárním vektorovém formátu .svg a frontendové programy využijí skript svg2ild.py dostupný z [42].

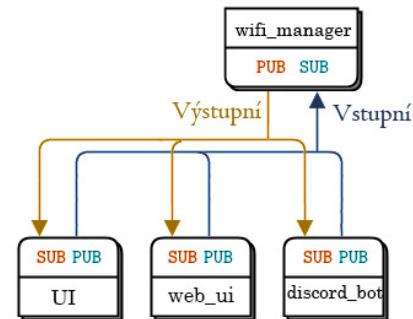
6.1 Komunikace mezi programy

Výše zmíněné frontendové a backendové programy jsou propojeny síťovými sockety, které zprostředkovává knihovna ZeroMQ [43]. Ta nabízí frontu¹ zpráv, bez potřeby samostatně běžícího brokeru.

Tato knihovna je využita k vytvoření dvou socketů pro každý backendový program, jedním backendový program přijímá příkazy od uživatele prostřednictvím frontendových programů (vstupní socket) a do druhého posílá informace o svém stavu frontendovým programům (výstupní socket), aby je zprostředkovaly uživateli. Toto spojení je schématicky naznačeno na obrázcích 6.1 a 6.2.



Obrázek 6.1: Komunikace mezi programy vstupním socketem na portu 5555



Obrázek 6.2: Komunikace mezi programy výstupním socketem na portu 5556

Jednotlivé příkazy, které můžou programy lasershow přijímat, jsou popsány v příloze v souboru README.md. Tento soubor se totiž na serveru github.com, kam byl projekt vystaven, zobrazuje přímo na domovské stránce projektu a kdokoliv, koho by projekt zajímal si je tam rychle najde.

¹Ve frontě jsou zprávy seřazeny od té nejdříve odeslané.

6.2 Lasershow

Program lasershow se stará o vykreslování obrazu podle příkazů od frontendových programů, patří tedy mezi backendové programy.

Je psán v komplilovaném jazyce C++, jeho běh je tedy rychlejší, než kdyby byl využit interpretovaný jazyk. Interpretované jazyky bývají pomalejší, protože interpreter musí kód překládat za běhu a tím oddaluje jeho exekuci [44]. To je důležité, protože při vykreslování každá ztracená milisekunda může znamenat nežádoucí efekty. I při vykreslování jednoduchých obrazců zaseknutí paprsku na jednom místě způsobí přesvětlený bod.

6.2.1 Inspirace open-source

Program byl inspirován projektem rpi-lasershow [45]. Zmíněný projekt posloužil jako předloha struktury promítání, ovšem všechny jeho části byly přepsány, aby umožňovaly pokročilé funkce výsledného programu lasershow.

Velkou výhodou oproti zmíněnému programu je možnost s programem interagovat. Původní program lasershow totiž po spuštění jednoduše přečetl soubor, který mu byl předán jako argument, a začal ho promítat. Promítání bylo možné přerušit pouze zastavením programu.

6.2.2 Funkce programu

Díky nově implementovaným interakcím je možné ve frontendových programech ovládat mnoho aspektů projekce popsaných v následujících kapitolách. Program lasershow je schopen přijímat příkazy od frontendových programů a reagovat na ně i při vykreslování probíhající projekce. Program také všechny informace o probíhající projekci a nadcházejících projekcích posíla frontendovým programům, aby je zobrazily uživateli.

Spouštění projekcí

Je možné vybrat soubor k projekci, vytvářet frontu projekcí, které se spustí za sebou, nastavit opakování jedné projekce, nebo celé fronty a také se posouvat v časové ose projekce (tzn. například skočit na 200. snímek právě probíhající projekce).

Nastavení projekce

Nastavením projekcí jsou myšleny:

- Ovládání jasu jednotlivých laserových diod.
- Deformace obrazu pro kompenzaci úhlu vůči plátnu.
- Změna velikosti a posun projekce.
- Rychlosti vykreslování – ta je úrčena časem, kdy projektor čeká mezi vykreslením jednotlivých bodů.
- Rychlosť časového průběhu projekce – ta je úrčena časem, po kterém projektor přeskočí na další snímek projekce.
- Vykreslování vždy úplných snímků – program může buď snímek přerušit ve chvíli, kdy je překročen čas úrčený pro snímek, nebo může po uplynutí tohoto času vždy dokreslit body snímků, které nestihl vykreslit.
- Časově přesného snímkování – při zapnutí časově přesného snímkování program vždy vybere k projekci snímek, který by se měl v danou chvíli promítat podle času, který uplynul od začátku projekce místo jednoduchého pokračování ve frontě snímků.

Note:
přidám
obrázky
ukazu-
jící
jednot-
livá
nasta-
vení

6.2.3 Využité knihovny

cppzmq

Knihovna `cppzmq` [46] je knihovna zprostředkující sockety ZeroMQ v jazyce C++. Využití této knihovny je vidět v ukázce kódu 6.1.

Ukázka kódu 6.1: Využití knihovny cppzmq v programu lasershow

```
1 // Clone server Model One
2
3 #include "zmq.hpp"
4 #include "zmq_addon.hpp"
5 #include <iostream>
6 #include <string>
7
8 int main(void)
9 {
10     zmq::context_t ctx(1); // Třída context_t je používána k vytvoření všech
11     → soketů u v programu.
12
13     zmq::socket_t publisher(ctx, zmq::socket_type::pub); // Vytvoření třídy
14     → socket_t reprezentující výstupní socket. Druhý argument určuje, zda do
15     → socketu můžeme zapisovat (publish) nebo z něj číst (subscribe).
16     publisher.bind("tcp://*:5556"); // Funkcí bind je socketu přiřazena adresa.
17     → V tomto případě socket pracuje s protokolem tcp na portu 5556.
18
19     zmq::socket_t command_receiver(ctx, zmq::socket_type::sub); // Vytvoření
20     → třídy reprezentující vstupní socket.
21     command_receiver.bind("tcp://*:5557"); // Vstupnímu socketu je přiřazena
22     → adresa.
23     command_receiver.set(zmq::sockopt::subscribe, ""); // Jestliže je socket
24     → typu sub, je třeba nastavit, na která téma bude reagovat. V tomto
25     → případě je nastaveno, že bude reagovat na všechna téma.
26
27     while (true)
28     {
29         zmq::message_t received; // Vytvoření třídy message_t, která
30         → reprezentuje zprávu. Při príjmání zpráv je třeba mít předem
31         → vytvořenou třídu message_t, do které se zpráva uloží.
32         if (command_receiver.recv(received, zmq::recv_flags::none)) // Příjem
33             → zprávy. Příjem je blokující, dokud není zpráva přijata. Při použití
34             → argumentu zmq::recv_flags::dontwait by byl příjem neblokující.
35             → Funkce vrací true, pokud byla zpráva úspěšně přijata, jinak vrací
36             → false.
```

```

23     {
24         std::cout << "přijato: \" " << received.to_string() << "\\" " <<
25             std::endl;
26
27         std::string string_to_be_sent = "potvrzuji, přijal jsem zprávu \" " +
28             received.to_string() + "\\\";
29
30         zmq::message_t to_be_sent(string_to_be_sent.c_str(),
31             string_to_be_sent.length() + 1); // Vytvoření zprávy, která bude
32             odeslána.
33         publisher.send(to_be_sent, zmq::send_flags::none); // Odeslání zprávy.
34     }
35
36     else {
37         std::cout << "Nebyla úspěšně přijata žádná zpráva."
38     }
39
40 }
41
42     return 0;
43 }

```

ADCDACPi

Knihovna ADCDACPi [47] zjednodušuje komunikaci s D/A převodníkem mcp4822, využitím v generátoru signálu pro galvanometry, viz kapitola 5.6.2, přes rozhraní SPI. Její použití je vidět na ukázce kódu 6.2

Ukázka kódu 6.2: Využití knihovny ADCDACPi v programu lasershow

```

1 #include "ABE_ADCDACPi.h"
2
3 int main()
4 {
5
6     ABElectronics_CPP_Libraries::ADCDACPi adcdac; // Definice třídy ADCDACPi, s
7             jejímiž metodami budeme pracovat.
8
9     if (adcdac.open_dac() != 1) // Inicializace SPI komunikace s D/A převodníkem
10        (DAC). Pokud se otevření nezdaří, program skončí.

```

```

9     {
10         return (1);
11     }
12
13     for (int i = 0; i < 4096; i++)
14     {
15         adcdac.set_dac_raw(i, 1); //nastavení hodnoty napětí na první kanál DAC,
16         → hodnota napětí bude rovna (i/4096 * Vref), kde Vref je referenční
17         → napětí DAC.
18         adcdac.set_dac_raw(i, 2); //nastavení hodnoty napětí na druhý kanál DAC
19     }
20
21     adcdac.close_dac(); // Uzavření SPI komunikace s DAC.
22     return (0);
23 }
```

pigpio

Knihovna pigpio [48] umožňuje ovládání GPIO pinů Raspberry Pi. Je využita k posílání PWM signálu do laserového modulu. Příklad využití PWM funkcí knihovny je vidět v ukázce kódu 6.3.

Ukázka kódu 6.3: Využití PWM funkcí knihovny pigpio v programu lasershow

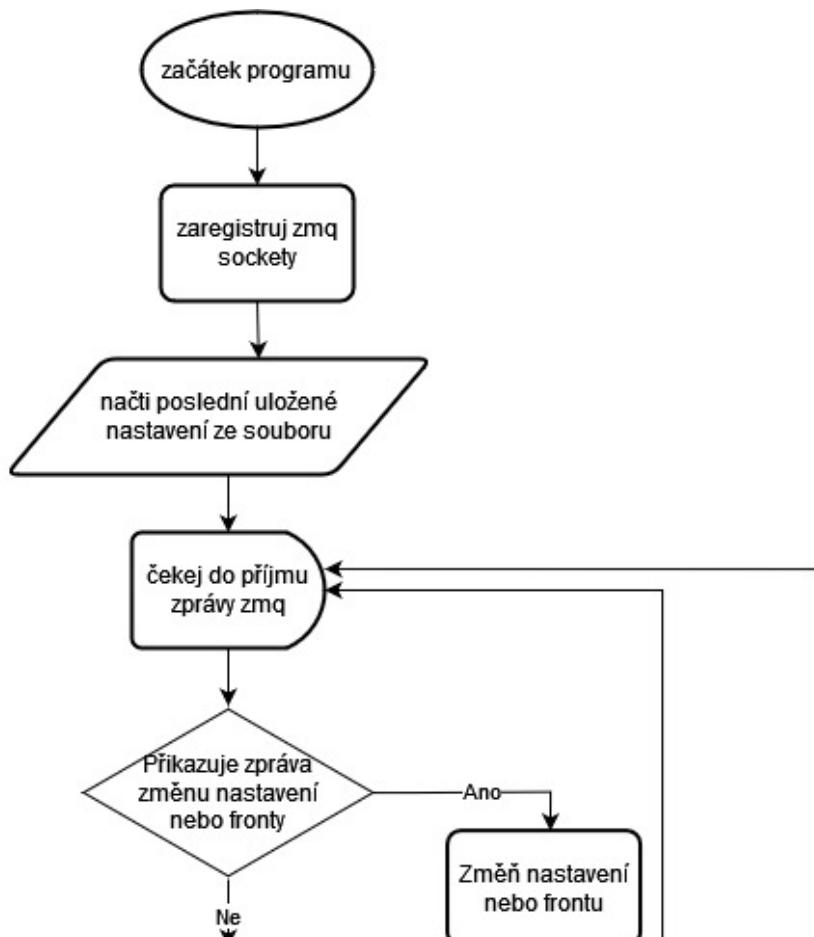
```

1 #include <pigpio.h>
2 #include <iostream>
3
4 constexpr int PIN = 17;
5
6 int main()
7 {
8     if (gpioInitialise() < 0) // Inicializace knihovny pigpio. Pokud se
9         → inicializace nezdaří, program skončí. Funkce musí být zavolána před
10        → použitím dalších funkcí knihovny pigpio.
11    {
12        std::cout << "Nepodařilo se inicializovat knihovnu" << std::endl;
13        return 1;
14 }
```

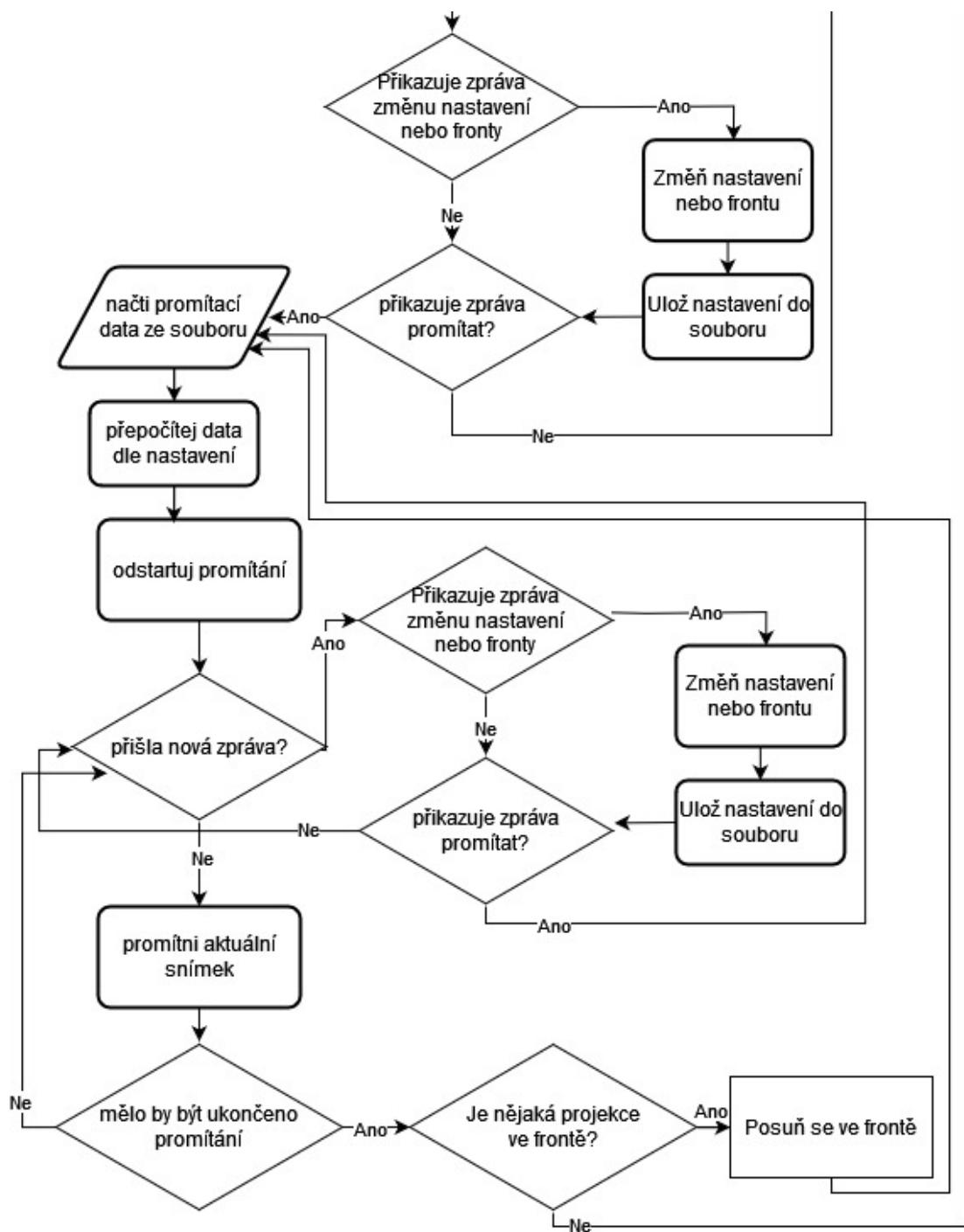
```
12     }
13
14     gpioSetMode(PIN, PI_OUTPUT); // Nastavení pinu 17 pro výstup.
15     gpioSetPullUpDown(PIN, PI_PUD_DOWN); // Připojení pull-down rezistoru pro pin
16     ↵ 17.
17     gpioSetPWMfrequency(PIN, 40000); // Nastavení frekvence PWM na 40 kHz
18     ↵ (maximální hodnotu).
19     gpioSetPWMrange(PIN, 255); // Nastavení rozsahu PWM na 255.
20     gpioPWM(PIN, 127); // Nastavení střídy PWM na 50 %.
21
22     gpioDelay(1000000); // 1 sekunda čekání.
23
24     gpioWrite(PIN, 0); // Vypnutí laseru.
25     gpioTerminate(); //deinitializace knihovny pigpio
26 }
```

6.2.4 Struktura programu

Struktura programu je vidět ve zjedonušeném obrázku 6.3 a je popsána na dalších stránkách.



Obrázek 6.3: Diagram programu lasershow; Pokračuje na další stránce.



Tento program pomocí knihovny ZeroMQ zaregistrouje svůj vstupní a výstupní socket a přihlásí se na tom vstupním k odběru zpráv.

Následně načte konfigurační soubor lasershow.cfg, který obsahuje uložené nastavení z posledního běhu programu a čeká na příchozí zprávy.

Jakmile přijde zpráva, přečte ji. Pokud je požadována změna nastavení, okamžitě ji provede, aktuální nastavení si uloží do souboru lasershow.cfg a přepočítá informace o promítaném obrazu, je-li to nutné (Například, změní-li se nastavení trapezoid-horizontal, přepočítá souřadnice vykreslovaných bodů). Jestliže je požadováno vykreslení obrazu ze souboru, načte soubor a začne obraz vykreslovat. Přitom průběžně posílá informace o stavu vykreslování do výstupního socketu. I při vykreslování obrazu tento program zpracovává zprávy a pokyny ze vstupního socketu.

6.3 wifi_manager

Program wifi_manager se přímo nepodílí ani na projekci, ani na interakci s uživatelem. Mění totiž WiFi připojení, pokud uživatel prostřednictvím nějakého frontendového programu tuto změnu vyžádá.

Program wifi_manager je napsaný v jazyce JavaScript s využitím runtime Node.js.

6.3.1 Funkce programu

Program umožňuje uživateli měnit nastavení WiFi připojení Raspberry Pi. Nastavením WiFi připojení jsou myšleny tři stavy. Prvním stavem je „WiFi“, kdy Raspberry Pi vyhledává v okolí známé sítě, ke kterým by se mohlo připojit. Druhým stavem „Hotspot“, v tomto stavu Raspberry Pi vysílá vlastní WiFi síť ve které se stalo přístupovým bodem. A třetím stavem je „WiFi připojení vypnuto“, kdy je WiFi anténa Raspberry Pi neaktivní.

Program má také svůj konfigurační soubor, kam si, podobně jako lasershow, ukládá aktuální nastavení, aby si ho „pamatoval“ i po restartu systému.

Služby obsluhující WiFi připojení

Spuštění přístupového bodu na Raspberry Pi totiž není tak jednoduchá, je pří ní potřeba změnit nastavení dvou služeb umožňujících provoz hotspotu a restartovat je. Tyto služby jsou hostapd a dnsmasq. Hostapd je služba umožňující využití Raspberry Pi jako přístupového bodu. Spravuje například ověření totožnosti přijovaných zařízení [49]. Dnsmasq je služba, která umožňuje přiřazovat připojeným zařízením IP adresy [50].

Naopak při přechodu do normálního „WiFi“ módu je potřeba tyto služby deaktivovat a spustit službu dhcpcd. Ta se chová jako klient pro komunikaci s přístupovým bodem protokolem DHCP [51]. Umožňuje tedy přístupovému bodu dynamicky přiřadit Raspberry Pi lokální IP adresu.

WiFi manager tedy existuje, aby sjednotil toto nastavování do jednoho programu. Bylo by totiž nepraktické, kdyby musely stejnou akci provádět ostatní programy, které jsou psané v různých jazycích.

Ke spuštění těchto služeb je využíván manažer systému a služeb systemd. Ten je součástí mnoha systémů založených na kernelu Linux. Při každém zapnutí systému je spuštěn jako první program a stará se o spuštění všech ostatních programů. [52]

6.3.2 Využité knihovny

`child_process`

V programu je využita funkce `execSync(command[, options])` z knihovny `child_process` pro spuštění systémových příkazů. To je potřeba pro zastavení zmíněných systémových služeb a k umožnění změn v jejich nastavení.

Tyto služby jsou zastavovány a spuštěny pomocí softwaru systemd a jeho nástroje v příkazovém řádku zvaném `systemctl`. V mém kódu tudíž používám funkci `execSync` k zastavení služeb hostapd a dnsmasq následovně `execSync("sudo systemctl disable hostapd dnsmasq")`.

Zeromq.js

Dále je v programu využita knihovna zeromq.js [53] pro komunikaci s ostatními programy. Jak je použita je vidět v ukázce kódu 6.4.

Ukázka kódu 6.4: Využití knihovny zeromq.js v programu wifi_manager

```
1 var zmq = require("zeromq"),
2     publisher = zmq.socket("pub"), // Vytvoření objektu publisher
3     → representujícího výstupní socket. Argument "pub" značí, že program bude
3     → zprávy do socketu posílat
4     command_receiver = zmq.socket("sub"); // Vytvoření objektu command_receiver
5     → representujícího vstupní socket. Argument "sub" značí, že program bude
5     → zprávy ze socketu přijímat
6
7     publisher.bindSync("tcp://*:5558"); // Přiřazení adresy a portu socketu
7     → publisher
8
9     command_receiver.bindSync("tcp://*:5559"); // Přiřazení adresy a portu socketu
9     → command_receiver
10    command_receiver.subscribe(""); // Přihlášení k odběru všech zpráv
11
12    command_receiver.on("message", (msg) => { // Registrace callback funkce, která
12      → se spustí při přijetí zprávy
13      console.log("Přijatá zpráva: ", msg.toString()); // Vypsání přijaté zprávy do
13      → konzole
14      publisher.send("prijal jsem zprávu: " + msg.toString()); // Odeslání zprávy
14      → do socketu publisher
15  });

```

6.3.3 Struktura programu

Registruje pro sebe dva sockety stejně jako lasershow, jedním přijímá příkazy týkající se nastavení WiFi a druhým odesílá zpětnou vazbu o aktuálním WiFi připojení. Zároveň registruje callback funkci, která se spustí při přijetí zprávy vstupním socketem.

Když příjme příkaz pro změnu připojení, zkонтroluje, jestli připojení tomuto příkazu už nevyhovuje. Pokud mu nevyhovuje, nejdříve si zapíše do svého konfiguračního souboru požadovaný stav wifi připojení. Následně zastaví služby hostapd, dnsmasq a dhcpcd, změní jim nastavení dle konfiguračních souborů předem uložených ve své složce a opět některé z nich spustí, podle toho jakého stavu WiFi připojení se snaží dosáhnout. Pro „Hotspot“ stav je potřeba, aby byly spuštěny služby hostapd, dnsmasq a dhcpcd, pro normální „WiFi“ stav je potřeba spuštěná pouze služba dhcpcd.

Může také příjmat příkazy, které pouze požadují hlášení o nastavení WiFi připojení a nechtějí toto nastavení nějak měnit. Při přijetí takového příkazu wifi_manager přečte svůj konfigurační soubor a pošle do svého výstupního socketu zprávu odpovídající aktuálnímu nastavení.

6.4 UI

Program UI je psaný v jazyce C++ stejně jako program lasershow. Tento program ovládá OLED displej, který je připojený na Raspberry Pi pomocí rozhraní I²C, a přijímá vstup od uživatele čtením rotačního enkodéru s tlačítkem. Také přijímá informace od programů lasershow a wifi_manager a posílá jim vstup od uživatele.

Na LCD displeji má uživatel díky tomuto programu přístup k menu, kde jsou zobrazeny aktuální informace o promítání a wifi připojení a kde uživatel může měnit nastavení dvou backendových programů.

Note:
TODO
fotosss

6.4.1 Využité knihovny

wPi_soft_lcd

Hlavní využitou knihovnou je wPi_soft_lcd [54]. Tato knihovna umožňuje jedno-duchou komunikaci s LCD prostřednictvím I²C převodníku. Její použití je vidět v ukázce kódu 6.5.

Ukázka kódu 6.5: Využití knihovny wPi_soft_lcd v programu UI ke komunikaci s LCD

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include "soft_lcd.h"
4
5 int main()
6 {
7     if (geteuid() != 0) // Ověření, zda je program spuštěn s administrátorskými
    → právy, kdyby nebyl, mohl by mít systém problém s nastavením PWM
    → podsvícení
8     {
9         printf("This program must be run as root.\n");
10        return 1;
11    }
12
13    lcd_t *lcd = lcd_create(23, 24, 0x3f, 2); // Inicializace komunikace s I2C
    → převodníkem na LCD (argumenty (číslo pinu s připojením SCL, číslo pinu s
    → připojením SDA, I2C adresa zařízení, počet řádků LCD))
14    if (lcd == NULL)
15    { // Funkce lcd_create vrací NULL, pokud se inicializace nezdařila
16        printf("LCD: Cannot initialize\n");
17        return 1;
18    }
19
20    char battery[] = { // definice vlastního znaku
21        0b01110,
22        0b10001,
23        0b10001,
24        0b10001,
25        0b10001,
26        0b10001,
27        0b10001,
28        0b11111};
29
30    lcd_create_char(lcd, 1, battery); // Odeslání vlastního znaku do LCD, druhý
    → argument určuje, jaký ascii kód znak nahrazuje
```

```

31
32     lcd_clear(lcd); // Vymazání obsahu LCD
33
34     lcd_print(lcd, "Stav baterie \01"); // Vypsání prvního řádku
35
36     for (int i = 100; i >= 0; i--)
37     {
38         lcd_pos(lcd, 1, 0);           // Nastavení pozice kurzoru na začátek druhého
39         → řádku
40         lcd_printf(lcd, "%d%%", i); // Vypsání hodnoty i v procentech
41         lcd_backlight_dim(lcd, static_cast<float>(i) / 100.f); // Nastavení jasu
42         → podsvícení LCD
43         usleep(1000); // Čekání po dobu jedné ms
44     }
45
46     return 0;
47 }
```

wiringPi

Předchozí popsaná knihovna pro posílání signálu I²C sběrnicí používá knihovnu wiringPi, která umožňuje ovládání GPIO pinů. Abych nepřidával do jednoho programu dvě knihovny interagující s hardwarem, stejnou knihovnu využívám i pro čtení dat z enkodéru. Její využití k detekci změn na pinech enkodéru je vidět na ukázce kódu 6.6.

Ukázka kódu 6.6: Využití knihovny wiringPi v programu UI k registraci přerušení (interruptů)

```

1 #include <wiringPi.h>
2 #include <iostream>
3
4 constexpr int PIN = 22;
5
6 void my_callback() // Funkce, která se zavolá při změně levelu na pinu PIN
7 {
8     std::cout << "Změna levelu na pinu " << PIN << " detekována" << std::endl;
```

```

9    }
10
11  int main()
12  {
13      if (wiringPiSetup() < 0) // Inicializace wiringPi knihovny, při úspěchu vrací
14          →  0
15      {
16          std::cerr << "Nepodařilo se inicializovat wiringPi knihovnu" << std::endl;
17          return 1;
18      }
19      pinMode(PIN, INPUT); // Nastavení pinu PIN pro vstup, to je nutné udělat před
20          →  dalším používáním pinu
21      pullUpDnControl(PIN, PUD_UP); // Připojení pull-up rezistoru na pin PIN, aby
22          →  byl level pinu HIGH, když není stisknuto tlačítko
23      wiringPiISR(PIN, INT_EDGE_BOTH, *my_callback); // Nastavení callback funkce
24          →  pro pin PIN. Ta je zavolána při změně levelu na pinu PIN
25
26      while (true)
27      {
28          delay(1000);
29      }
30      return 0;
31  }

```

Je nepraktické využívat ve dvou programech jinou knihovnu na interakci s hardwarem, proto v budoucnu budou knihovna wPi_soft_lcd a celý tento program přepsány tak, aby využívaly modernější knihovnu pigpio popsanou v kapitole 6.2.3 stejně jako program lasershow.

cppzmq

Samozřejmě program také využívá knihovnu cppzmq [46] a popsanou v kapitole 6.2.3. Program ji používá podobně jako program lasershow, ne však stejně.

Největším rozdílem je, že místo funkce `void zmq::socket_t::bind` používá funkci `void zmq::socket_t::connect(const char *addr_)`. Funkci bind je totiž po-

třeba zavolat přesně jednou pro každý socket. Jestliže ji už pro sockety zavolal program lasershow, popřípadě wifi_manager a socket tedy už je registrovaný k danému portu, je třeba zavolat funkci `connect`.

Dále samozřejmě stejně jako všechny frontendové programy do vstupního socketu zprávy posílá a z výstupního socketu zprávy čte. Celková ukázka využití knihovny frontendovým programem v jazyce C++ je v ukázce kódu 6.7.

Ukázka kódu 6.7: Využití knihovny cppzmq v programu UI

```
1 // Clone client Model One
2
3 #include "zmq.hpp"
4 #include "zmq_addon.hpp"
5 #include <iostream>
6 #include <chrono>
7
8 int main(void)
9 {
10     zmq::context_t ctx(1); // Třída context_t je potřebná k vytvoření všech
11     → soketů v programu.
12
13     zmq::socket_t subscriber(ctx, zmq::socket_type::sub); // Vytvoření třídy
14     → soket_t reprezentující výstupní socket. Druhý argument určuje, zda do
15     → socketu můžeme zapisovat (publish) nebo z něj číst (subscribe).
16     subscriber.connect("tcp://localhost:5556"); // Funkcí connect se socket
17     → připojí k socketu, který jiný program zaregistroval / zaregistruje na
18     → danou adresu.
19     subscriber.set(zmq::sockopt::subscribe, ""); // Nastavení socketu pro
20     → příjemání všech zpráv.
21
22     zmq::socket_t command_sender(ctx, zmq::socket_type::pub);
23     command_sender.connect("tcp://localhost:5557");
24
25     while (true)
26     {
27         zmq::message_t received; // Vytvoření třídy message_t, do které můžeme
28         → zapsat přijatou zprávu.
```

```

22     while (subscriber.recv(received, zmq::recv_flags::dontwait)) // Metodou
→       recv se pokusíme přijmout zprávu ze socketu. Pokud je v socketu
→       zpráva, uloží se do proměnné received a cyklus pokračuje. Díky
→       argumentu zmq::recv_flags::dontwait je metoda neblokující. Tímto
→       způsobem přečteme všechny zprávy, které jsou u socketu.
23     {
24         std::cout << "received: \""
→           received.to_string() << "\""
25     }
26
27     std::string user_input;
28     getline(std::cin, user_input); // Přečtení vstupu od uživatele.
29
30     zmq::message_t msg(user_input.c_str(), user_input.length()); //
→       Vytvoření zprávy, kterou chceme odeslat.
31     command_sender.send(msg, zmq::send_flags::none); // Odeslání zprávy.
32   }
33   return 0;
34 }
```

6.4.2 Struktura programu

Program začne inicializací komunikace s LCD. Poté pokračuje registrací interruptů na pinech, ke kterým je připojen enkodér. A následně se připojí k socketům aplikací lasershow a wifi_manager.

V další části programu je definováno samotné menu. To má podobu struktury menu_option. V ukázce kódu 6.8 je vidět, jak je tato struktura dufinována.

Ukázka kódu 6.8: Definice struktury menu_option v programu UI

```

1   template <typename T>
2   struct menu_val
3   {
4       T num;
5       T min;
6       T max;
```

```

7     T change_by = 1;
8 };
9
10 enum menu_option_style
11 {
12     NESTED_MENU = 0,
13     VALUE,
14     SELECTION, // all children are style text, nest_selected is written to value after button is pressed
15     TEXT,
16     ROOT_MENU,
17     UNDEFINED
18 };
19
20 struct menu_option
21 {
22     std::string name; // Text zobrazený uživateli
23
24     std::string command_name; // interní název položky pro komunikaci s backendovými programy
25
26     menu_option_style style = UNDEFINED; // Typ položky; od něj se odvíjí přístup k položce
27
28     std::vector<menu_option> nested_menu_options = {};// Podpoložky této položky
29     uint8_t nest_selected = 0; // Vybraná podpoložka
30     uint8_t nest_scroll = 0; // Interní hodnota; Určuje které podpoložky mají zrovna být vykresleny
31     bool nest_option_active = 0; // Určuje, jestli je zrovna vybraná podpoložka aktivní
32     bool redraw = 0; // Vlajka indikující, zda má být obsah položky znova vykreslen, například po změně hodnoty
33
34     menu_val<float> value; // hodnota položky, například u položek typu VALUE
35
36     bool has_function = 0;
37     void (*function)(zmq::socket_t &, menu_option &);
38 };

```

Díky ní je možné definovat menu poměrně jednoduchým způsobem. 7krácený příklad je v ukázce kódu 6.9

Ukázka kódu 6.9: Příklad definice menu v programu UI

```
1     menu_option root = {
2         .name = "ROOT",
3         .style = ROOT_MENU,
4         .nested_menu_options = {
5             {
6                 .name = "progress%",
7                 .command_name = "progress",
8                 .style = VALUE,
9                 .value = {0, 0, 100, 0.5},
10            },
11            {
12                .name = "current_frame",
13                .command_name = "current_frame",
14                .style = VALUE,
15                .value = {1, 1, 1, 1},
16            },
17            {
18                .name = "-no out received-",
19                .style = TEXT
20            },
21            {
22                .name = "STOP",
23                .command_name = "STOP",
24                .style = TEXT,
25                .has_function = 1,
26                .function = send_option_command
27            },
28            {
29                .name = "PAUSE",
30                .command_name = "PAUSE",
31                .style = TEXT,
32                .has_function = 1,
33                .function = send_option_command
34            },
35            {
36                .name = "PROJECT",
37                .style = NESTED_MENU,
```

```

38     .has_function = 1,
39     .function = fill_with_files,
40 },
41 {
42     .name = "options",
43     .style = NESTED_MENU,
44     .nested_menu_options = {
45         {
46             .name = "screen_brightness",
47             .command_name = "screen_brightness",
48             .style = VALUE,
49             .value = {50, 0, 100},
50         },
51         {
52             .name = "repeat",
53             .command_name = "repeat",
54             .style = VALUE,
55             .value = {0, 0, 1, 1},
56         },
57         {
58             .name = "point_delay",
59             .command_name = "point_delay",
60             .style = VALUE,
61             .value = {0, 0, 10000, 10},
62         },
63         {
64             .name = "target_frame_time",
65             .command_name = "target_frame_time",
66             .style = VALUE,
67             .value = {0, 0, 10000, 1},
68         },
69         {
70             .name = "trapezoid_horizontal",
71             .command_name = "trapezoid_horizontal",
72             .style = VALUE,
73             .value = {0, -1.f, 1.f, 0.05},
74         },
75         {

```

```

76         .name = "trapezoid_vertical",
77         .command_name = "trapezoid_vertical",
78         .style = VALUE,
79         .value = {0, -1.f, 1.f, 0.05},
80     },
81 },
82 .has_function = 1,
83 .function = read_options,
84 }});

```

Následuje nekonečný cyklus, ve kterém program příjmá zprávy od programů laser-show a wifi_manager a volá funkci `bool menu_interact(lcd_t *lcd, zmq::socket_t &command_sender, menu_option &parent_menu_option, bool redraw = 0)`, která rekurzivně prochází menu následujícím způsobem. Vždy, pokud má atribut `parent_menu_option.nest_selected` hodnotu true a atribut `style` objektu `parent_menu_option.nested_menu_options[nest_selected]` je roven NESTED_MENU, zavolá sebe samou, ale jako argument `parent_menu_option` použije právě objekt `parent_menu_option.nested_menu_options[nest_selected]`.

Jakmile takto najde zrovna aktivní položku menu, přečte proměnnou, kde je uložená relativní pozice enkodéru oproti poslednímu čtení a dle této proměnné se změní proměnnou `parent_menu_option.nest_selected`.

Note:
TODO:
před
ode-
vzdanim
osetrit
line
breaks
tady

Pokud narazí na aktivní položku menu, jejíž atribut `style` je roven VALUE, chování funkce se lehce změní. V takovém případě se změní hodnota, kterou tato položka menu reprezentuje. Ta je uložena v atributu `value` právě vybrané položky.

6.5 web_ui

Narozdíl od předchozích dvou zmiňovaných programů je program `web_ui` psaný v jazyce javascript, ten nepatří mezi nejrychlejší, ale díky runtime Node.js a knihovnám http a formidable v něm bylo časově nenáročné vytvořit http web server.

Tento server běží na portu 3000 a je dostupný z lokální sítě (tzn. přímo z Raspberry Pi na adresu `http://localhost:3000` nebo z jakéhokoliv zařízení na stejně lokální síti na IP adresu RPi). Program je využíván pro jednoduchou interakci s uživatelem.

Ten může pomocí webového prohlížeče ovládat laserový projektor pár kliknutími nebo zadávat vlastní příkazy klávesnicí.

Ve webovém prostředí jsou zatím konzole pro ssh přístup k Raspberry Pi a pro komunikaci s programy wifi_manager a lasershow. Také je v něm formulář, kde si uživatel může jednoduše vybrat soubor k projekci. To ve výběrovém menu, kde se mu zobrazí soubory už uložené ve složce pro ně určené. Dále se ve webovém prostředí nachází formulář, kterým uživatel do této složky může soubor nahrát. Soubory může nahrávat ve formátech .svg a .ild. Aktuální stav webového prostředí je vidět na obrázku ??.

Note:
TODO:
obrazek

V blízké budoucnosti do prostředí přibudou prvky, které mohou ovládat frontu, nastavení programů lasershow a wifi_manager. Později by měla přibýt funkce kreslení v prohlížeči, kdy uživatel bude moct v prohlížeči nakreslit požadovaný tvar a projektor ho už v průběhu kreslení bude promítat. Tato funkce by měla v budoucnu být propojena s kamerou připojenou na Raspberry Pi. V tu chvíli uživatel bude mít možnost v prohlížeči kreslit do obrazu reálného světa.

6.5.1 Využité knihovny

Http

Knihovna http je využita k jednoduchému vytvoření interaktivního http serveru, ke kterému se uživatel může připojit ze svého prohlížeče. Příklad takového serveru je vidět v ukázce kódu 6.10

Ukázka kódu 6.10: Příklad http serveru v jazyce javascript s knihovnou http

```
1 const fs = require('fs');
2 const path = require('path');
3 const http = require('http');
4
5 const server = http.createServer((req, res) => {
6   // Get the file path from the request URL
7   const filePath = path.join(__dirname, req.url);
8 }
```

```

9   // Check if the file exists
10  fs.access(filePath, fs.constants.F_OK, (err) => {
11    if (err) {
12      // File not found
13      res.statusCode = 404;
14      res.end('File not found');
15    } else {
16      // Read the file and send it as the response
17      fs.readFile(filePath, (err, data) => {
18        if (err) {
19          // Error reading the file
20          res.statusCode = 500;
21          res.end('Internal server error');
22        } else {
23          // Set the appropriate content type and send the file data
24          const ext = path.extname(filePath);
25          let contentType = 'text/plain';
26          if (ext === '.html') {
27            contentType = 'text/html';
28          } else if (ext === '.css') {
29            contentType = 'text/css';
30          } else if (ext === '.js') {
31            contentType = 'text/javascript';
32          }
33
34          res.setHeader('Content-Type', contentType);
35          res.end(data);
36        }
37      });
38    }
39  });
40});
41
42 const port = 3000;
43 server.listen(port, () => {
44   console.log(`Server running on port ${port}`);
45 });

```

Zeromq.js, ssh2, socket.io a xterm.js

Http server tedy pošle prohlížeči uživatele kód, který v tomto prohlížeči běží. Poté je knihovna socket.io je využita ke komunikaci mezi kódem běžícím v prohlížeči uživatele (klientem) a kódem běžícím na Raspberry Pi (serverem). Společně s knihovnou zeromq.js umožňují tomuto programu fungovat jako prostředník mezi klientem a backendovými programy. Díky knihovně ssh2 také tímto spojením je možné přímo z prohlížeče psát do příkazového řádku.

Knihovna xterm.js běžící v kódu klienta je využita k dosažení interaktivity a vzhledu konzolí, které se uživateli ve webovém rozhraní zobrazují.

V ukázkách kódu 6.11 a 6.12 je vidět, jak server předává klientovi data od backendových programů a jak je klient uživateli vykresluje.

Ukázka kódu 6.11: Kód programu web_ui běžící na serveru. Ten příjmá informace pomocí knihovny zeromq.js a předává je klientovi pomocí knihovny socket.io.

```
1 //server
2 /* části, kde knihovna http odesílá, nebo příjmá soubory jsou přeskočeny */
3
4 var zmq = require("zeromq"),
5     lasershow_sender = zmq.socket("pub"),
6     wifiman_sender = zmq.socket("pub"),
7     lasershow_receiver = zmq.socket("sub"),
8     wifiman_receiver = zmq.socket("sub"); // definice zeromq socketů; Ta už byla
→ několikrát podrobně popsána u programů wifi_manager a UI
9
10 var io = require("socket.io")(server); // inicializace knihovny socket.io
11 var SSHClient = require("ssh2").Client; // inicializace knihovny ssh2
12
13 //připojení zeromq socketů
14 lasershow_sender.connect("tcp://localhost:5557");
15 lasershow_receiver.connect("tcp://localhost:5556");
16 lasershow_receiver.subscribe("");
17
18 wifiman_sender.connect("tcp://localhost:5559");
```

```

19 wifiman_receiver.connect("tcp://localhost:5558");
20 wifiman_receiver.subscribe("");
21
22 /* definice zpracování HTTP requestů knihovnou http */
23
24 io.on("connection", function (socket) { // Definice funkce volané při připojení
25   ↪ klienta
26   var conn = new SSHClient(); // Spuštění ssh terminálu, do něho server
27   ↪ vypisuje systémové příkazy
28   conn
29     .on("ready", function () { // Definice funkce spuštěné při spuštění ssh
30       ↪ terminálu
31       conn.shell(function (err, stream) { // Připojení k rozhraní terminálu,
32         ↪ při připojení se zavolá námi definovaná funkce, která dostane
33         ↪ argumenty err a stream. Argument err je použit k detekci errorů,
34         ↪ argument stream je využit pro samotnou komunikaci s terminálem
35         if (err) // Ověření, že se terminál otevřel bez problémů. Pokud je
36         ↪ objekt err evaluován jako true, něco se pokazilo.
37         return socket.emit(
38           "sshdata",
39           "\r\n*** SSH SHELL ERROR: " + err.message + " ***\r\n"
40         );
41         socket.on("sshdata", function (data) { // Jestliže socket.io příjme
42           ↪ zprávu týkající se tématu "sshdata", vypíše ji do terminálu
43           stream.write(data);
44         });
45         stream
46           .on("data", function (d) { // Definice funkce, která je zavolána,
47             ↪ jestliže terminál chce vyslat nějaké data. V tom případě je
48             ↪ server pošle klientovi
49             socket.emit("sshdata", d.toString("binary"));
50           })
51           .on("close", function () { // Definice funkce zavolané v případě, že
52             ↪ je stream terminálu uzavřen
53             socket.emit("sshdata", "**ssh stream closed**");
54           });
55         });
56       });
57     });

```

```

46     .on("close", function () { // Definice funkce zavolané v případě, že je
47         → terminál uzavřen
48         socket.emit("sshdata", "\r\n*** SSH CONNECTION CLOSED ***\r\n");
49     })
50     .on("error", function (err) { // Definice funkce zavolané v případě, že v
51         → terminálu nastane chyba
52         socket.emit(
53             "sshdata",
54             "\r\n*** SSH CONNECTION ERROR: " + err.message + " ***\r\n"
55         );
56     });
57
58     conn.connect({ // Samotné připojení k ssh terminálu, využívá rsa klíče
59         → místo hesel
60         host: config.sshHost,
61         username: config.sshUsername,
62         privateKey: fs.readFileSync(config.sshKeyPath),
63     });
64
65     socket.on("LASERSHOWdata", function (data) { // Když od klienta přijde
66         → zpráva úrčená pro program lasershow, server ji přepošle do příslušného
67         → zeromq socketu
68         lasershow_sender.send(data.toString());
69     });
70     socket.on("WIFIMANdata", function (data) { // Když od klienta přijde zpráva
71         → úrčená pro program wifi_manager, server ji přepošle do příslušného
72         → zeromq socketu
73         wifiman_sender.send(data.toString());
74     });
75
76     lasershow_receiver.on("message", (msg) => { // Když zeromq socketem příjde
77         → zpráva od programu lasershow, server ji přepošle klientovi s označním, od
78         → kterého programu zpráva přichází
79         io.sockets.emit("LASERSHOWmsg", msg.toString() + "\n\r");
80     });
81     wifiman_receiver.on("message", (msg) => { // Když zeromq socketem příjde zpráva
82         → od programu wifi_manager, server ji přepošle klientovi s označním, od
83         → kterého programu zpráva přichází

```

```

73     io.sockets.emit("WIFIMANmsg", msg.toString() + "\n\r");
74 });
75
76 /* spuštění http serveru */

```

Ukázka kódu 6.12: Kód programu web_ui běžící v prohlížeči. Ten příjmá informace od serveru knihovny socket.io a zobrazuje je uživateli kromě jiného pomocí knihovny xterm.js.

```

1 // klient
2 // knihovny jsou přiádny v html kódu
3
4 class terminal {
5     constructor(container_id) { // díky tomuto konstruktoru je jednoduché k více
6         ← terminálům knihovny xterm.js přiřadit jejich html elementy a nastavit
7         ← jejich vlastnosti
8         this.container = document.getElementById(container_id);
9         this.term = new Terminal({
10             cursorBlink: true,
11             allowTransparency: true,
12             drawBoldTextInBrightColors: true,
13             fontSize:
14                 1 * parseFloat(getComputedStyle(document.documentElement).fontSize),
15             theme: {
16                 background: "rgba(0, 0, 0, 0)",
17             },
18         });
19         this.fit = new FitAddon.FitAddon();
20         this.term.loadAddon(this.fit);
21         this.term.open(this.container);
22         this.fit.fit();
23     }
24     var terminals = { // výše popsaný konstruktor je využit k vytvoření 3 xterm
25         ← terminálů
26         ssh: new terminal("ssh-terminal-container"),
27         lasershow: new terminal("lasershow-terminal-container"),

```

```

27     wifiman: new terminal("wifiman-terminal-container"),
28   };
29
30 // Browser -> Backend
31 terminals.ssh.term.onData(function (ev) { // Pokud do ssh terminálu uživatel
→ něco zadá, klient to pošle serveru pro zpracování v ssh
32   socket.emit("sshdata", ev.toString());
33 });
34
35 // Browser -> Backend
36 var lasershow_line = ""
37 terminals.lasershow.term.onData(function (ev) { // Pokud uživatel něco zadá do
→ terminálů naležícím komunikaci s backendovými programy, tento text je
→ ukládán do řetězce lasershow_line a je odeslán až, když uživatel stiskne
→ enter
38   string = ev.toString();
39   lasershow_line += string;
40   terminals.lasershow.term.write(ev);
41   if (string.match(/\n|\r/gi) != null) { // hledání klávesy enter pomocí regexu
→ (regulárního výrazu)
42     lasershow_line = lasershow_line.replace(/\n|\r/gi, "")
43     socket.emit("LASERSHOWdata", lasershow_line);
44     terminals.lasershow.term.write("\r> ");
45     lasershow_line = "";
46   }
47 });
48 //to stejné platí i pro wifi_manager
49
50
51 // Backend -> Browser
52 socket.on("sshdata", function (data) { // Při přijetí dat pocházejících z ssh
→ jsou vypsány do náležitého terminálu
53   terminals.ssh.term.write(data);
54 });
55 // Backend -> Browser
56 socket.on("LASERSHOWmsg", function (data) { // Při přijetí dat pocházejících od
→ programu lasershow jsou vypsány do náležitého terminálu
57   console.log(data);

```

```
58     terminals.lasershow.term.write(data.replace(/\n/g, "\n\r"));
59 });
60 // Backend -> Browser
61 socket.on("WIFIMANmsg", function (data) { // Při přijetí dat pocházejících od
62   ← programu wifi_manager jsou vypsány do náležitého terminálu
63   terminals.wifiman.term.write(data.replace(/\n/g, "\n\r"));
64 });
```

6.6 discord bot

Posledním programem, který by měl být využíván k interakci s uživatelem je discord_bot. Ten bude naprogramován a uveden do provozu v blízké budoucnosti.

Měl by sloužit k interakci s uživatelem v chattovací aplikaci discord. Bude také psaný v jazyce javascript v runtime Node.js, stejně jako předchozí programy se přihlásí k socketům knihovnou zmq, ale na rozdíl od nich tento program bude moct interagovat s uživatelem přes internet, ať už je kdekoli na světě.

Pomocí knihovny discord.js se přihlásí k předem vytvořenému bot účtu, který může na předem vytvořeném discord serveru čekat na zprávy od uživatele, ty posílat do vstupního socketu a posílat uživateli zpětnou vazbu, kterou příjme z výstupního socketu.

6.7 Instalační skript

Nedílnou součástí softwarové výbavy projektoru je instalační skript. Ten je psaný v příkazovém jazyce bash, který odpovídá sekvenci příkazů v příkazovém řádku. Instalační skript umožňuje instalaci celého projektu pouze třemi příkazy, viz ukázka kódu 6.13.

Ukázka kódu 6.13: Příkazy potřebné k instalaci projektu

```
1 git clone https://github.com/phuid/laser_projector.git
2 cd laser_projector
3 bash install.sh
```

Instalační skript stáhne a nainstaluje nainstaluje veškeré závislosti a knihovny ostatních programů. Stáhne také samotný interpreter Node.js. Následně zkompluluje programy psané v C++ a nainstaluje a nastaví služby potřebné k vysílání hotspotu. Poté pomocí manažeru procesů pm2 dostupného z [55] nastaví automatické zapínání ostatních programů a po potvrzení uživatelem systém restartuje, aby provedené změny nabyla efekt.

Kapitola 7

Diskuze

Existuje nemalé množství open-source implementací laserových projektorů.

Většina z nich ale není uživatelsky přívětivá, to neznamená, že jsou tyto projekty špatné, ale hodí se uživatelům, kteří jsou pokročilejší programátoři. Jednou z nich byl i tento projekt inspirován. Ta se jmenuje rpi-lasershow [45]. Obsahuje základní vykreslovací program, který umí číst soubory formátu .ild. Její hlavní problém spočívá v neexistujícím uživatelském prostředí.

Nad ostatní implementace vystupuje projekt openlase [56]. Ten je koncipován jako knihovna, kterou zájemci mohou začlenit do vlastních projektů. Hardware k němu navžený ale vyžaduje připojení k počítači i zásuvce, což by mohlo být nepraktické, kdyby se hodilo projekci předvést potenciálním zájemcům o technologii naživo.

Grafické uživatelské prostředí obsahuje projektor vypracovaný v diplomové práci Bc. Pavla Svobody [57]. Ten je ale, stejně jako openlase, konstruován s připojením k počítači a zásuvce.

Za zmínku také stojí projekt zpracovaný v youtube videu kanálu „Ben Makes Everything“ [9], který obsahuje i aplikaci na mobilní telefon. Ta s projektem komunikuje přes bluetooth. Tento projektor ale uživatelům umožňuje vypsat jen text, žádné vlastní obrázky ani videa. Zajímavé na tomto projektu je, že je založen

na hranolovém skeneru, který tvůrce vytvořil ze starého hard-disku. Je tedy poměrně dostupný pro kohokoliv s přístupem ke 3d tiskárně, která byla při výrobě projektoru použita. Představení hranolových skenerů je užitečné vzhledem k jejich častému využití v aplikacích jiných, než je vykreslování obrazu. Grafiky vykreslené galvanometrovými skenery můžou být zajímavější, než ty vykreslené skenery hranolovými.

Když opustíme sféru open-source, samozřejmě existuje spousta komerčních řešení. Například od firem Pangolin nebo Laserworld. Ty jsou často výkonnější, než výše popsané projekty a jsou využívány pro vytváření profesionálních laserových efektů.

Velice zajímavým zařízením je Laser Cube [58] od firmy Wicked Lasers. Je to komerční laserový projektor pro širokou veřejnost s velice jednoduchým uživatelským prostředím. Open-source řešení včetně této práce by se k němu daly přirovnat jako alternativy pro technicky zdatné jedince, kteří jsou schopni s technologií pracovat sami.

Závěr

V rámci této práce byl navrhnut a sestrojen funkční RGB laserový projektor, kromě jednoho vadného obvodu, kvůli kterému musí projektor zatím být připojen k zásuvce.

K tomuto projektoru byla naprogramována softwarová výbava umožňující promítat barevné obrazce a skrze uživatelské prostředí ovládat nastavení promítání přímo na zařízení, z webového prohlížeče, nebo přes chatovací aplikaci discord odkudkoliv na světě. Tímto uživatelským prostředím je také možné nastavovat WiFi připojení projektoru.

Celý projekt byl vystaven na webovou platformu pro sdílení open-source projektů github.com, kde ho může najít kdokoliv.

Literatura

1. PURKHET, Abderyim; HALABI, Osama; FUJIMOTO, Tadahiro; CHIBA, Norishige. Accurate and efficient drawing method for laser projection. *The Journal of the Society for Art and Science*. 2008, roč. 7, č. 4, s. 155–169.
2. UREY, H; HOLMSTROM, S; BARAN, U; AKSIT, K; HEDILI MK and Eides, O. MEMS scanners and emerging 3D and interactive augmented reality display applications. In: *2013 Transducers & Eurosensors XXVII: The 17th International Conference on Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS & EUROSENSORS XXVII)* [online]. IEEE, 2013, s. 2485–2488 [cit. 2024-02-14]. Dostupné z: https://scholar.archive.org/work/54b6eo4syvb2zbhrgbvgiwick6e/access/wayback/https://s3.amazonaws.com/pstorage-ryerson-5010877717/28122132/Chao_Fan.pdf.
3. EASTMAN, Jay. Brief history of barcode scanning. In: *OSA Century of Optics* [online]. Optical Society, 2015, s. 128–133 [cit. 2024-01-28]. Dostupné z: <https://opg.optica.org/books/download.cfm?product=11&v=128>.
4. QUAN, Haoyuan; ZHANG, Ting; SHEN LUO, Hang Xu and; NIE, Jun; ZHU, Xiaoqun. Photo-curing 3D printing technique and its challenges. *Bioactive Materials* [online]. 2020, roč. 5, č. 1, s. 110–115 [cit. 2024-01-28]. ISSN 2452-199X. Dostupné z DOI: <https://doi.org/10.1016/j.bioactmat.2019.12.003>.

5. EDL, MMTJ; MIZERÁK, Marek; TROJAN, Jozef. 3D laser scanners: history and applications. *Acta Simulatio* [online]. 2018, roč. 4, č. 4, s. 1–5 [cit. 2024-01-28]. Dostupné z: <https://doi.org/10.3997/2214-4609.201403279>.
6. PFEIFER, Norbert; BRIESE, Christian. Laser scanning—principles and applications. In: *Geosiberia 2007-international exhibition and scientific congress*. European Association of Geoscientists & Engineers, 2007, cp–59.
7. HOLMSTRÖM Sven TS and Baran, Utku; UREY, Hakan. MEMS laser scanners: a review. *Journal of Microelectromechanical Systems*. 2014, roč. 23, č. 2, s. 259–275.
8. MARSHALL, Gerald F; STUTZ, Glenn E. *Handbook of optical and laser scanning* [online]. Taylor & Francis, 2012 [cit. 2024-01-28]. Dostupné z: <https://library.oapen.org/bitstream/handle/20.500.12657/41669/9781439808801.pdf?sequence=1>.
9. BEN MAKES EVERYTHING. *DIY Laser Projector - Built from an old hard drive* [online]. [cit. 2024-01-28]. Dostupné z: https://www.youtube.com/watch?v=u9TpJ-_hBR8. Zdroj obrázku 2.3.
10. INTERNATIONAL ELECTROTECHNICAL COMMISSION. *Electropedia: The World's Online Electrotechnical Vocabulary* [online]. [cit. 2024-01-31]. Dostupné z: <https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=313-01-02>.
11. WIKIPEDIA CONTRIBUTORS. *Galvanometer — Wikipedia, The Free Encyclopedia* [online]. 2023. [cit. 2024-01-31]. Dostupné z: <https://en.wikipedia.org/w/index.php?title=Galvanometer&oldid=1171003585>.
12. CHAO, Fan; HE, Siyuan; CHONG, James; MRAD, Ridha Ben; FENG, Lei. Development of a micromirror based laser vector scanning automotive HUD. In: *2011 IEEE International Conference on Mechatronics and Automation* [online]. IEEE, 2011, s. 75–79 [cit. 2024-02-13]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/5985634>.

13. TYNDALL, J. *Notes of a Course of Nine Lectures on Light: Delivered at the Royal Institution of Great Britain, April 8-June 3, 1869.* Longmans, Green, 1870. Dostupné také z: <https://books.google.cz/books?id=x1UIAAAAIAAJ>.
14. PŘISPĚVATELÉ WIKIPEDIE. *Node.js — Wikipedia, The Free Encyclopedia* [online]. 2024. [cit. 2024-02-13]. Dostupné z: <https://en.wikipedia.org/w/index.php?title=Node.js&oldid=1197985648>.
15. ELECTRONICS, SparkFun. *Serial Peripheral Interface (SPI) - SparkFun Learn* [online]. [cit. 2024-02-11]. Dostupné z: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>.
16. ELECTRONICS, SparkFun. *I2C - SparkFun Learn* [online]. [cit. 2024-02-11]. Dostupné z: <https://learn.sparkfun.com/tutorials/i2c/all>.
17. ELECTRONICS, SparkFun. *Pull-up Resistors - SparkFun Learn* [online]. [cit. 2024-02-11]. Dostupné z: <https://learn.sparkfun.com/tutorials/pull-up-resistors>.
18. TOULSON, Rob; WILMSHURST, Tim. Chapter 4 - Analog Output. In: TOULSON, Rob; WILMSHURST, Tim (ed.). *Fast and Effective Embedded Systems Design*. Oxford: Newnes, 2012, s. 57–75. ISBN 978-0-08-097768-3. Dostupné z DOI: <https://doi.org/10.1016/B978-0-08-097768-3.00004-0>.
19. PŘISPĚVATELÉ WIKIPEDIE. *Pulzně šířková modulace — Wikipedie: Otevřená encyklopédie* [online]. 2023. [cit. 2024-02-08]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Pulzn%C4%9B_%C5%A1%C3%AD%C5%99kov%C3%A1_modulace&oldid=23044621.
20. THOMSON INDUSTRIES, INC. *What is Pulse Width Modulation (PWM)?* [online]. [cit. 2024-02-08]. Dostupné z: <https://www.thomsonlinear.com/en/support/tips/what-is-pwm>.
21. PŘISPĚVATELÉ WIKIPEDIE. *Přerušení — Wikipedie: Otevřená encyklopédie* [online]. 2023. [cit. 2024-02-14]. Dostupné z: <https://cs.wikipedia.org/w/index.php?title=P%C5%99eru%C5%A1en%C3%AD%C3%AD%C5%AD&oldid=22983219>.

22. RUSSELL BARNES, Raspberry Pi Ltd. *Get started with your new Raspberry Pi* [online]. [cit. 2023-02-09]. Dostupné z: <https://magpi.raspberrypi.com/articles/get-started-new-raspberry-pi>.
23. INTERNATIONAL LASER DISPLAY ASSOCIATION. The ILDA Standard Projector [online]. 1999, s. 11, 28 [cit. 2024-01-24]. Dostupné z: https://www.ilda.com/resources/StandardsDocs/ILDA_ISP99_rev002.pdf.
24. PASCHOTTA, R. *Dichroic Mirrors* [RP Photonics Encyclopedia]. RP Photonics AG, 2005 [cit. 2024-02-05]. Dostupné z DOI: [10.61835/1un](https://doi.org/10.61835/1un). Available online at https://www.rp-photonics.com/dichroic_mirrors.html.
25. LASKAKIT. *20x4 LCD displej 2004 modrý + I2C převodník* [online]. [cit. 2023-02-08]. Dostupné z: <https://www.laskakit.cz/20x4-lcd-displej-2004-modry-i2c-prevodnik/>.
26. DEJAN NEDELKOVSKI, HowToMechatronics.com. *How Rotary Encoder Works and How To Use It with Arduino* [online]. [cit. 2024-02-08]. Dostupné z: <https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/>.
27. ŠLEHOFER, Jan. Ampalyzer – Analyzátor audio zesilovačů [online]. 2020 [cit. 2024-02-08]. Dostupné z: <https://prihlaska.soc.cz/archiv42/getWork/hash/72744387-58f2-11ea-9fea-005056bd6e49>.
28. TEXAS INSTRUMENTS INCORPORATED. Basic Calculation of a Boost Converter's Power Stage [online]. 2022 [cit. 2024-02-09]. Dostupné z: <https://www.ti.com/lit/an/slva372d/slva372d.pdf?ts=1707394558861>.
29. INSTRUCTABLES, DELTAFLO. *Arduino Laser Show With Real Galvos* [online]. [cit. 2024-01-22]. Dostupné z: <https://www.instructables.com/Arduino-Laser-Show-With-Real-Galvos>.
30. MICROCHIP TECHNOLOGY INC. MCP4821/MCP4822 12-Bit DACs with Internal VREF and SPI™ Interface Datasheet [online]. 2005 [cit. 2024-01-24]. Dostupné z: <https://ww1.microchip.com/downloads/en/devicedoc/21953a.pdf>.

31. TEXAS INSTRUMENTS INCORPORATED. TL082 Wide Bandwidth Dual JFET Input Operational Amplifier Datasheet [online]. 2013 [cit. 2024-01-24]. Dostupné z: https://www.ti.com/lit/ds/symlink/tl082-n.pdf?ts=1706076567444&ref_url=https%253A%252F%252Fwww.google.com%252F.
32. TEXAS INSTRUMENTS INCORPORATED. ADC0831-N/ADC0832-N/ADC0834-N/ADC0838-N 8-Bit Serial I/O A/D Converters with Multiplexer Options [online]. 2013 [cit. 2024-02-09]. Dostupné z: <https://www.ti.com/lit/ds/symlink/adc0831-n.pdf?ts=1707440787302>.
33. MASARYKOVA UNIVERZITA. *Operační zesilovače* [online]. [cit. 2024-02-09]. Dostupné z: https://is.muni.cz/el/sci/jaro2017/F5090/um/E17_P8.pdf.
34. LASKAKIT. *USB-C PD/QC přepínač napájecího napětí* [online]. [cit. 2023-02-09]. Dostupné z: <https://www.laskakit.cz/usb-c-pd-qc-prepinac-napajecihonapeti/>.
35. LASKAKIT. *1-kanál 5V relé modul, Low level, 250VAC 10A* [online]. [cit. 2023-02-09]. Dostupné z: <https://www.laskakit.cz/1-kanal-5v-rele-modul--low-level--250vac-10a/>.
36. LASKAKIT. *Ochrana Li-Ion baterie 2S 20A* [online]. [cit. 2023-02-09]. Dostupné z: <https://www.laskakit.cz/ochrana-li-ion-baterie-2s-20a/>.
37. LASKAKIT. *Step-up boost měnič s XL6009, modrá* [online]. [cit. 2023-02-09]. Dostupné z: <https://www.laskakit.cz/step-up-boost-menic-s-xl6009--modra/>.
38. LASKAKIT. *Nabíječka Li-ion článku TP5100, 2A 1-2S* [online]. [cit. 2023-02-09]. Dostupné z: <https://www.laskakit.cz/nabijecka-li-ion-clanku-tp5100--2a-1-2s/>.
39. LASKAKIT. *Step-down měnič s XL4005* [online]. [cit. 2023-02-09]. Dostupné z: <https://www.laskakit.cz/step-down-menic-s-xl4005/>.
40. LASKAKIT. *Step-up boost měnič s MT3608* [online]. [cit. 2023-02-09]. Dostupné z: <https://www.laskakit.cz/step-up-boost-menic-s-mt3608/>.

41. AG, Laserworld (Switzerland). *Laserworld Showeditor V7 - Complete Version* [online]. [cit. 2024-02-12]. Dostupné z: <https://www.showeditor.com/en/downloads/showeditor-software/download/2-laserworld-showeditor-software-download/291-laserworld-showeditor-complete-version.html>.
42. GITHUB.COM/MARCAN. *openlase/tools/svg2ild.py* [online]. [cit. 2024-02-12]. Dostupné z: <https://github.com/marcan/openlase/blob/master/tools/svg2ild.py>.
43. ZEROMQ. *ZeroMQ An open-source universal messaging library* [online]. [B.r.]. [cit. 2024-02-09]. Dostupné z: <https://zeromq.org/>.
44. KWAME, Ampomah Ernest; MARTEY, Ezekiel Mensah; CHRIS, Abilimi Gilbert. Qualitative assessment of compiled, interpreted and hybrid programming languages. *Communications*. 2017, roč. 7, č. 7, s. 8–13.
45. TESKAC, Tomislav. *LaserShow on Raspberry Pi Zero* [online]. [cit. 2024-02-12]. Dostupné z: <https://github.com/tteskac/rpi-lasershow>.
46. ZEROMQ. *cppzmq* [online]. [cit. 2024-02-09]. Dostupné z: <https://github.com/zeromq/cppzmq>.
47. AB ELECTRONICS UK. *ADCDACPi* [online]. [cit. 2024-02-09]. Dostupné z: https://github.com/abelectronicsuk/ABElectronics_CPP_Libraries/tree/master/ADCDACPi.
48. GITHUB.COM/JOAN2937. *pigpio library* [online]. [cit. 2024-02-09]. Dostupné z: <https://abyz.me.uk/rpi/pigpio/sitemap.html>.
49. MALINEN, Jouni. *hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/WPA3/EAP/RADIUS Authenticator* [online]. [cit. 2024-02-13]. Dostupné z: <https://w1.fi/hostapd/>.
50. KELLEY, Simon. *Dnsmasq* [online]. [cit. 2024-02-13]. Dostupné z: <https://dnsmasq.org/doc.html>.
51. MARPLES, Roy. *dhcpcd* [online]. [cit. 2024-02-13]. Dostupné z: <https://roy.marples.name/projects/dhcpcd>.

52. SYSTEMD. *System and Service Manager* [online]. [cit. 2024-02-13]. Dostupné z: <https://systemd.io/>.
53. ZEROMQ. *ZeroMQ.js* [online]. [cit. 2024-02-12]. Dostupné z: <https://github.com/zeromq/zeromq.js>.
54. GITHUB.COM/ELECTRONICAYCIENCIA. *wPi_soft_lcd* [online]. [cit. 2024-02-11]. Dostupné z: https://github.com/electronicayciencia/wPi_soft_lcd.
55. PM2. *PM2 - Home* [online]. [cit. 2024-02-12]. Dostupné z: <https://pm2.keymetrics.io/>.
56. MARTIN, Hector. *OpenLase Open Realtime Laser Framework* [online]. [cit. 2024-02-12]. Dostupné z: <https://marcan.st/projects/openlase/>.
57. SVOBODA, Bc. Pavel. *LASEROVÝ PROJEKTOR* [online]. [cit. 2024-02-12]. Dostupné z: <https://dspace.vut.cz/server/api/core/bitstreams/a55c1118-9166-4449-806a-c2a73bfef66/content>.
58. LASERS, Wicked. *LaserCube / Portable Laser Show / WiFi Show Laser Projector* [online]. [cit. 2024-02-12]. Dostupné z: <https://www.lasersos.com/>.

Seznam obrázků

2.1	Mechanika polygonových skenerů [8].	11
2.2	Úhlová rozdílnost zrcátek polygonového skeneru a paprsky od nich odražené [8].	12
2.3	příklad projekce laserového projektoru s polygonovým skenerem [9]	12
2.4	konstrukce galvanometrových skenerů [8].	13
2.5	Zapojení a vnitřní konstrukce galvanometrů [8].	14
3.1	HUD v letadle Boeing 737-800 [12]	16
3.2	HUD ve stíhacím letounu F16 [12]	16
4.1	PWM signál s měnící se střídou; Střída naznačena oranžovou křivkou. Převzato a upraveno z [20]	21
5.1	Celkové schéma zapojení hardwarových komponentů; Signálové linky jsou znázorněny zeleně, napětí z baterie fialově, napětí 5 V červeně a další napětí odstíny oranžové.	23
5.2	Pohled zvrchu na Raspberry Pi [22]	25
5.3	Řídící deska galvanometrů s vyznačenými konektory a hřejícími čipy	26
5.4	Laserový modul s řídící deskou a vyznačenými konektory	27
5.5	Sedm základních barev laserového modulu	28
5.6	Displej z tekutých krystalů (LCD); Převzato a upraveno z [25]	30
5.7	I ² C převodník napojený na LCD [25]	30
5.8	Vnitřní schéma enkodéru [26]	30
5.9	Výstup enkodéru [26]	30

5.10	Schéma zapojení rotačního enkodéru	31
5.11	Časový průběh stavu kontaktů rotačního enkodéru při otáčení hřídelí na obě strany	32
5.12	HAT DPS připevněná na Raspberry Pi	33
5.13	Zapojení invertujícího obvodu	34
5.14	Zapojení čipu TL082 pro jeden kanál řídící desky galvanometrů	36
5.15	Schéma obvodu voltmetu baterie	36
5.16	Celkové schéma napájení komponentů projektoru	37
5.17	BMS Modul se třemi kontakty pro sérii baterií (0V, 4.2V a 8.4V) a výstupními kontakty ((+) a (-)) [34]	38
5.18	Relé modul [35]	38
5.19	Zapojený nabíjecí obvod	39
5.20	Power Delivery trigger board [36]	40
5.21	step-up měnič s čipem XL6009 [37]	40
5.22	Modul nabíječky dvou sériově zapojených Li-ion baterií [38]	40
5.23	step-down měnič s čipem XL4005 [39]	41
5.24	step-up měnič s čipem MT3608 [40]	41
5.25	Pohled do projektoru s odstraněnou přední stěnou v programu Autodesk Fusion	43
5.26	Kompletně nainstalované druhé patro (Na obrázku je modul TP5100 zapojen s opačnou polaritou, ve výrobku byla chyba opravena, ale tato fotka je stále nejlepší ilustrace.)	44
5.27	Pohled na projektor ze strany hrany sousedící se zadní a pravou boční stěnou	46
6.1	Komunikace mezi programy vstupním socketem na portu 5557	48
6.2	Komunikace mezi programy výstupním socketem na portu 5556	48
6.3	Diagram programu lasershow; Pokračuje na další stránce.	55

Seznam ukázek kódu

6.1	Využití knihovny cppzmq v programu lasershow	50
6.2	Využití knihovny ADCDACPi v programu lasershow	52
6.3	Využití PWM funkcí knihovny pigpio v programu lasershow	53
6.4	Využití knihovny zeromq.js v programu wifi_manager	59
6.5	Využití knihovny wPi_soft_lcd v programu UI ke komunikaci s LCD	60
6.6	Využití knihovny wiringPi v programu UI k registraci přerušení (interruptů)	62
6.7	Využití knihovny cppzmq v programu UI	64
6.8	Definice struktury menu_option v programu UI	65
6.9	Příklad definice menu v programu UI	66
6.10	Příklad http serveru v jazyce javascript s knihovnou http	70
6.11	Kód programu web_ui běžící na serveru. Ten příjmá informace pomocí knihovny zeromq.js a předává je klientovi pomocí knihovny socket.io.	72
6.12	Kód programu web_ui běžící v prohlížeči. Ten příjmá informace od serveru knihovny socket.io a zobrazuje je uživateli kromě jiného pomocí knihovny xterm.js.	75
6.13	Příkazy potřebné k instalaci projektu	77

