FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Zařízení pro ovládání aplikace Adobe Lightroom s pomocí hardwarového ovladače
Student:	Vítězslav Macháček
Vedoucí:	Ing. Pavel Kubalík, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

1) Prozkoumejte existující řešení umožňující ovládání aplikace Adobe Lightroom z hardwarového ovladače.
2) Pomocí metod softwarového inženýrství navrhněte vlastní řešení vyhovující níže uvedeným požadavkům.
3) Navrhněte vlastní zařízení fungující jako samostatný ovladač splňující tyto požadavky:
 - komunikace s aplikací bude realizována přes USB a Bluetooth,
 - ovladač bude umožňovat čtení vstupů z enkodérů, tlačítek a Hallových senzorů,
 - ovladač bude napájen z baterie.
4) Navržený ovladač zrealizujte a naprogramujte.
5) Pro PC vytvořte vlastní aplikaci umožňující zpracování dat z ovladače a jejich přenos do aplikace Adobe Lightroom Classic.
6) Aplikace bude mít uživatelské rozhraní, které bude umožňovat její nastavení a toto nastavení bude ukládat do lokální databáze.
7) Výsledné řešení řádně otestujte.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 27. března 2023 v Praze.

Bakalářská práce

ZAŘÍZENÍ PRO OVLÁDÁNÍ APLIKACE ADOBE LIGHTROOM S POMOCÍ HARDWAROVÉHO OVLADAČE

Vítězslav Macháček

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Pavel Kubalík, Ph.D.
16. května 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především panu Ing. Pavlovi Kubalíkovi, Ph.D. za vedení při psaní teto práce. Dále bych chtěl poděkovat mé rodině za podporu, trpělivost a korekturu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené

V Praze dne 16. května 2024

Abstrakt

Tato práce se zabývá ovládáním aplikace Adobe Lightroom Classic pomocí hardwarového ovladače. Krom návrhu a tvorby ovladače a firmwaru pro něj se práce zabývá také návrhem a implementací doprovodné aplikace. Tato aplikace propojuje ovladač s Adobe Lightroom Classic a umožňuje nastavení chování ovladače. Aplikace je imlementována v jazyce Rust a Typescript pomocí frameworku Tauri. Ovladač je postaven na MCU z rodiny nRF52 od Nordic Semiconductors a ke komunikaci s aplikací používá USB nebo Bluetooth LE.

Klíčová slova hardwarový ovladač, Adobe Lightroom, Rust, Tauri, USB, Bluetooth, Typescript

Abstract

This thesis deals with controlling Adobe Lightroom Classic by hardware controller. It deals with desing and implementation of firmware and hardware for the controller and also deals with desing and implementation of accompaniyng app. This app connects together the controller and Adobe Lightroom Classic and is used to set the behavior of the controller. The app is implemented in Rust and Typescript with help of Tauri framework. The controller is based on MCU from the nRF52 family created by Nordic Semiconductors. It comunnicates with application by USB or Bluetooth LE.

Keywords hardware controller, Adobe Lightroom, Rust,Tauri,USB, Bluetooth, Typescript

Seznam zkratek

SDK	Software develpoment kit
CAF	Common Architecture Framework
RTOS	Realit Time Operation System
BAS	Battery Service
HID	Human Interface Device
GATT	Generic attribute profile
ATT	Attribute protcol
GAP	Generic Access protocol
BLE	Bluetooth Low Energy
MCU	Microcontroller unit
USB	Universal Serial Bus

Kapitola 1
Úvod

Adobe Lightroom je nástroj pro úpravu digitálních fotografií, který je hojně používán jak profesionály, tak amatéry. S rostoucí jednoduchostí pořizování fotografií roste také potřeba jejich rychlé a jednoduché úpravy. Hardwarové ovladače mohou fotografům pomoci se zvýšením efektivity a jednoduchosti úprav.

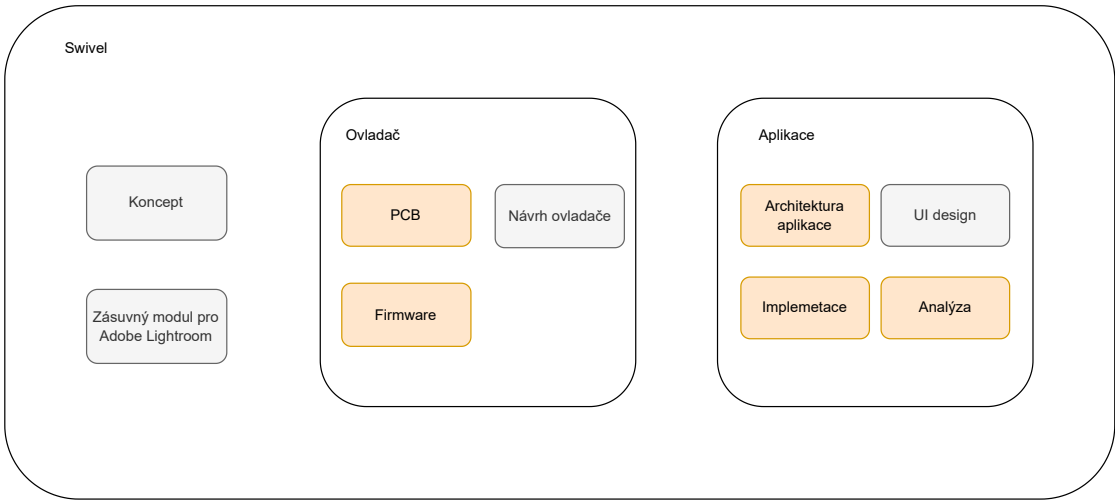
Ovladače nabízejí různé fyzické prvky, jako jsou tlačítka, kolečka nebo posuvníky. Tyto prvky umožňují rychle a jednoduše nastavit parametry fotky, například expozici, kontrast či další parametry. Využití těchto prvků může být pohodlnější a intuitivnější než úprava fotografie pomocí myši či touchpadu.

Tato práce je součásti projektu Swivel, jehož cílem je vytvořit jednoduchý hardwarový ovladač pro Lightroom. Práce se zabývá analýzou požadavků na doprovodnou aplikaci, hardware a firmware ovladače. V první části (kapitola 3) popisuje již existující řešení a to jak komerční, tak otevřená. Druhá část (kapitola 4) se zabývá analýzou a návrhem jednotlivých částí a výběrem vhodných technologií. Popis realizace se nachází ve třetí části (kapitola 5) a poslední část (kapitola 6) se zabývá testováním celého řešení. A to jak z pohledu uživatele, tak i automatickým testováním či testováním vlastností ovladače.

Kapitola 2

Upřesnění zadání

Jak již bylo zmíněno, tato práce je součástí projektu Swivel. Jeho cílem je vytvořit hardwarový ovladač pro Adobe Lightroom Classic. Cílem této práce je vytvořit aplikaci pro komunikaci s ovladačem a Adobe Lightroom Classic, hardwarovou část ovladače a firmware pro vytvořený hardware. V diagramu 2.1 jsou barevně označeny části, kterými se tato práce zabývá, a šedě části, které jsou již mimo rozsah této práce.



Obrázek 2.1 Znázornění součástí projektu Swivel (oranžovými částmi se zabývá tato práce)

Kapitola 3
Existující řešení

Tato kapitola popisuje již dostupné ovladače pro Adobe Lightroom. Existuje sice více řešení než zde bude uvedeno, ale i ta vynechaná se principiálně příliš neliší od těch popisovaných. Všechna uvedená řešení jsou až na jedno komerčního rázu.

3.1 Loupedeck

Loupedeck [1] je první společnost, která přišla na trh s vlastím hardwarovým ovladačem pro Adobe Lightroom. A to s ovladačem Loupedeck original v roce 2016[2]. Aktuálně mají v nabídce čtyři verze ovladače a to Loupedeck+, Loupedeck CT, Loupedeck Live a Loupedeck Live S a kromě podpory Adobe Lightroom podporují mnoho dalších aplikací pro kreativní tvorbu či živé vysílání.

Tyto ovladače fungují na principu mapování vstupu (zmáčknutí tlačítka, otočení encoderu) na změnu parametru fotky v Adobe Lightroom. V doprovodné aplikaci samozřejmě lze nastavit, co který prvek dělá.

3.1.1 Aplikace

V aplikaci lze nastavit, jaké akce nebo změny se mají vykonat po interakci s ovladačem. Jak lze na obrázku 3.1 vidět, tak nastavení aplikace je hierarchické. Nastavení pro aplikaci může mít několik pracovních prostředí. Každé pracovní prostředí obsahuje skupiny ovládacích prvků (encodery, tlačítka, dotyková tlačítka), ke kterým lze přiřadit stránky, které obsahují už samotné akce. Mezi těmito stránkami se dá přecházet nezávisle na ostatních ovládacích prvcích.

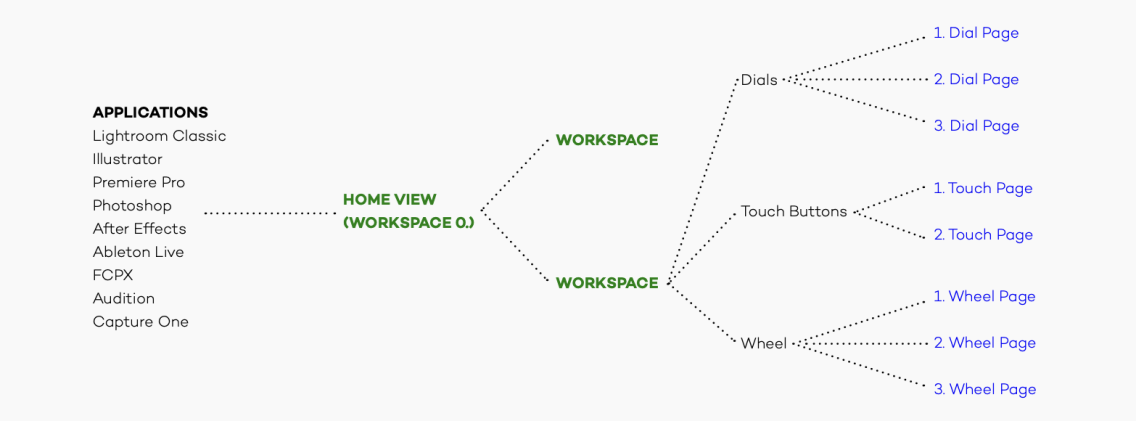
Vedle této hierarchie ještě existuje nastavení některých kulatých a hranatých tlačítek, které existuje pouze jedno pro aplikaci. Na tyto tlačítka lze nastavit právě přepínání mezi jednotlivými pracovními prostředími.

Ovladače Loupedeck mívají ještě několik tlačítek se speciální funkcionalitou, která však nelze změnit. Například tlačítko Fn, které umožňuje použít sekundární funkcionalitu prvků.

3.1.2 Ovladače

Ovladače firmy Loupedeck se dají rozdělit do dvou skupin. A to na ovladače Loupedeck+ a Original v jedné skupině a Loupedeck CT, Live a Live S ve skupině druhé.

První skupina je primárně zaměřená na ovládání Adobe Lightroom a obsahuje opravdu velké množství ovládacích prvků, přičemž druhá skupina obsahuje menší ovladače, které se vyznačují kombinací dotykového displeje a dalších ovládacích prvků[1].



Obrázek 3.1 Hierarchie nastavení v aplikaci Loupedeck [3]

3.2 Tourbox

Nejmladší z uvedených společností, které se zabývají výrobou a vývojem ovladačů pro Adobe Lightroom, je Tourbox[4]. Oproti ostatním řešením je Tourbox menší a neklade si za cíl stát se primárním způsobem ovládání Lightroomu, ale spíše jen pomocníkem. Je tvarován tak, aby byl použitelný v jedné ruce. Uživatel tak může jednou rukou pracovat s grafickým tabletem či myší a zároveň druhou rukou používat Tourbox pouze na přepínání nebo nastavování nástrojů, které se jinak ovládají myší či tabletem.

Tourbox má v nabídce dvě verze jejich ovladače a to Tourbox Neo a Tourbox Elite. Tourbox Elite je novější verze, která oproti předchozí verzi podporuje bezdrátovou komunikaci přes Bluetooth a má haptickou zpětnou vazbu.

3.2.1 Aplikace

Doprovodná aplikace [5] k ovladači je na rozdíl od stejné aplikace od Loupedeck velmi jednoduchá. Neobsahuje žádné složité hierarchické nastavení, nabízí pouze presety. Preset je soubor s nastaveními, co mají jednotlivé ovládací prvky nebo jejich kombinace, provádět za akci.

Akce lze rozdělit do tří skupin:

Klávesové zkratky jsou kombinace kláves stisknutých v jeden moment. Tourbox má v sobě uložený seznam zkratek, které lze přiřadit, ale je možné si vytvořit i vlastní klávesové zkratky.

Akce v podporovaných aplikacích - tyto akce lze ještě dále rozdělit na dvě skupiny - pro encodery a pro tlačítka.

Makra jsou posloupnosti akcí, které se vykonají po jeho aktivaci. Tyto akce mohou být klávesové zkratky, otevírání souborů nebo webových stránek, zadání textu, časové zpoždění či akce s myší.

Jednotlivé presety se buď dají přepínat manuálně nebo podle aktuálně aktivního programu. Pro Adobe Lightroom lze vytvořit dva presety a to podle toho v jestli se uživatel nachází v módu úprav fotografií nebo jejich výběru.

3.3 Monogram CC

Další ovladač pochází od společnosti Monogram CC [6]. Jejich ovladač se liší od ostatních řešení tím, že je plně modulární. Jednotlivé moduly se spojují pomocí panelů a pinů. Monogram nabízí

čtyři typy modulů. A to moduly s tlačítky, encodery a lineárními potenciometry. Čtvrtý modul se jmenuje *Orbiter* a krom otočného prstence nabízí ve středu tlakově senzitivní disk.

3.3.1 Aplikace

Aplikace pro Monogram CC je svým přístupem podobná aplikaci pro Tourbox. Pracuje s presety, které mohou být navázány na program. Každému ovládacímu prvku lze nastavit, co bude ovládat, ale nejsou nepodporovány kombinace několika prvků. Oproti Tourboxu je možno nastavit rozsahy parametrů.

3.4 MIDI2LR

MIDI2LR [7] je jediné open source řešení pro ovládání Adobe Lightroom pomocí hardwarového ovladače. Dále se od ostatních řešení odlišuje tím, že je to pouze aplikace, která umožňuje použít ovladače komunikující přes protokol MIDI pro ovládání Adobe Lightroom. Mezi tyto ovladače patří například *Behringer X-TOUCH MINI*.

Samotná aplikace umožňuje nastavit pro každý prvek funkci, kterou má vykonávat. Toto nastavení se dá uložit do profilů. Mezi jednotlivými profily je možno přecházet dvěma způsoby. Za prvé lze přiřadit profil k aktivnímu módu nebo aktuálně používanému nástroji v Adobe Lightroom. Druhým způsobem je nastavení přepínání profilů ovládacím prvkem na ovladači.

Kapitola 4
Analýza a návrh

Tato kapitola se věnuje analýze technologií potřebných pro ovládání Adobe Lightroom pomocí hardwarového ovladače. Také popisuje návrh konceptu ovládaní Adobe Lightroom, architekturu aplikace, firmwaru ovladače a analýzu návrhu elektroniky ovladače.

4.1 Adobe Lightroom

Pod Adobe Ligthroom [8] se skrývají dvě verze aplikace. A to Adobe Lightroom Classic a CC (Creative Cloud). CC verze je nová verze Adobe Lightroom, která se od verze Classic liší větší integrací s cloudovými službami od Adobe a zjednodušeným ovládacím rozhraním. Hlavní rozdíl pro nás je ten, že pouze verze Classic umožňuje rozšířit aplikaci pomocí zásuvného modulu napsaného v programovacím jazyce Lua. Tento modul umožňuje komunikovat s dalšími aplikacemi a ovládat Adobe Lightroom[9].

4.1.1 Adobe Lightroom Classic

Jak již bylo napsáno, naše aplikace bude podporovat pouze Adobe Lightroom Classic. Proto se podíváme blíže na to, jak funguje.

Adobe Lightroom Classic je rozdělen do sedmi modulů. Každý z těchto modulů má svou specifickou funkci. Pro nás jsou hlavní moduly Library a Develop, protože tyto moduly slouží k vybírání a úpravě fotografií. Ostatní moduly jsou moduly sloužící k další práci s fotografiemi. Například pro tvorbu fotoknih, přípravu pro tisk fotografií či jejich geolokaci [10].

4.2 Koncept ovládání

Oproti jiným ovladačům, které se primárně zaměřují na zpřístupnění jednotlivých efektů/parametrů pomocí jednotlivých ovládacích prvků a z toho důvodu jich mají větší počet, má ovladač minimum ovládacích prvků. Na vizualizaci lze vidět 4.1, že má dva otočné prstence, jeden se zarážkami a druhý s hladkým chodem. A dále dvě tlačítka a jeden dvoupolohový přepínač.

Koncept ovládaní rozděluje úpravu fotek na dvě části. V první části si uživatel vybere fotografie k úpravě. Tato část se nazývá *Select mode*. V druhé části, nazvané *Edit mode*, vybrané fotografie upravuje. Tyto části korespondují s módy *Library* a *Develop* v Adobe Lightroom. Mezi těmito částmi se uživatel přepíná pomocí přepínače na ovladači.

V druhé části, si uživatel jednoduše vybere a seřadí parametry, které chce upravovat (vytvoří si svůj pracovní postup - *flow*). A pak při úpravě postupně prochází vybrané parametry a na snímku je upraví, na rozdíl od jejich hledání na ovladači či v rozhraní Adobe Lightroom. Přičemž změna

Obrázek 4.1 Vizualizace ovladače

jednotlivých parametrů bude probíhat pomocí horního prstence, který nemá zarážky, a pohyb mezi parametry pomocí spodního prstence se zarážkami.

V první části již uživatel nemá takovou volnost ve výběru funkcí. Na prstencích jsou přednastavené funkce na pohyb mezi fotkami a jejich hodnocení (Přiřazování hvězdiček).

Jak bylo výše zmíněno, ovladač má i tlačítka. Pro tato tlačítka si uživatel může nastavit akce (resetování úpravy, zobrazení fotografie před a po úpravě) a tato nastavení jsou vázaná na aktuální část procesu úpravy fotografií.

4.3 Funkční a nefunkční požadavky

Tato podkapitola se zabývá funkčními a nefunkčními požadavky pro ovladač a aplikaci. Tyto požadavky vycházejí z konceptu ovládání a ze zadání práce. Funkčními požadavky se myslí požadavky na ovladač či aplikaci a nefunkčními se myslí jejich omezení.

4.3.1 Ovladač

Tato sekce řeší funkční a nefunkční požadavky na desku ovladače.

4.3.1.1 Funkční požadavky

- **F1 – Komunikace s počítačem**
  Ovladač posílá počítači, jaké interakce s ním probíhají, a odpovídá na dotazy, v jakém stavu se nachází.
- **F2 – Ovládací prvky**
  Ovladač má ovládací prvky umožňující:
  - Posouvání ve *flow*/posouvání mezi fotkami

  - Upravování aktuálně vybraného parametru/úprava počtu hvězdiček
  - Přepínání módu ovladače
  - Tlačítka na provádění akcí

- **F3 – Stav baterie**
  Ovladač umí zjistit stav baterie a reportuje ho počítači.

- **F4 – Komunikace s uživatelem**
  Ovladač umí uživateli sdělit údaje o svém stavu. Například stav připojení přes Bluetooth či hodnotu parametru v Adobe Lightroom.

4.3.1.2 Nefunkční požadavky

- **N1 – Komunikace s počítačem**
  Ovladač komunikuje s počítačem pomocí USB nebo Bluetooth.

- **N2 – Tvar ovladače a pozice ovládacích prvků**
  Protože je ovladač součástí projektu Swivel, tak musí mít tvar, aby se vešel do ovladače, a některé komponenty, například ovládací prvky, dioda a USB konektor, musí být umístěny na požadovaných pozicích.

- **N3 – Způsoby komunikace ovladače s uživatelem**
  Ovladač by měl informovat uživatele o stavu připojení pomocí LED diody a pomocí vibrací při pokusu o překročení maximální nebo minimální hodnoty parametr.

4.3.2 Aplikace

Tato sekce řeší funkční a nefunkční požadavky na doprovodnou aplikaci.

4.3.2.1 Funkční požadavky

- **F1 – Tvorba a úprava *flow***
  V aplikaci lze vytvořit a upravovat *flow*. To znamená, že lze přidávat a odebírat jednotlivé kroky, měnit jejich pořadí a upravovat jejich parametry. *Flow* je možné také pojmenovat a případně smazat.

- **F2 – Ukládání *flow***
  Aplikace ukládá jednotlivá *flow* lokálně a umožňuje jejich načítání, úpravu a mazání.

- **F3 – Vyhodnocování *flow***
  Aplikace vyhodnotí *flow* podle přijatých interakcí od ovladače.

- **F4 – Aktivní *flow***
  V aplikaci si lze vybrat, které *flow* je aktuálně aktivní.

- **F5 – Nastavení tlačítek**
  Aplikace umožňuje nastavení akcí, které se budou provádět po stisknutí tlačítek. Nastavení je separátní pro vybírání a úpravu fotek.

- **F6 – Komunikace s ovladačem**
  Aplikace je schopná komunikovat s ovladačem. Při připojení zjistí, v jaké stavu je ovladač, a podle toho se nastaví.

- **F7 – Ovládání Adobe Lightroom Classic**
  Aplikace komunikuje s Adobe Lightroom. Pomocí této komunikace lze Adobe Lightroom ovládat a případně z něho číst informace o aktuálně vybrané fotce případně ohledně jeho aktuálního stavu. Dále je možné upravovat parametry fotky.

- **F8 – Ukazatel aktuálního stavu**
  Aplikace má speciální stavové okno, které ukazuje aktuální, předchozí a následující krok. Pokud to u daného kroku dává smysl, tak udává i jeho hodnotu. Toto okno si může uživatel umístit, kam chce, a je vždy viditelné před ostatními okny.

  Aplikace také ukazuje, jestli je ovladač připojen.

- **F9 – Detekce typu stisknutí tlačítek**
  Aplikace detekuje jaký typ stisknutí uživatel provedl. Rozlišuje mezi jednoduchým a dvojitým stisknutí a dlouhým podržením.

4.3.2.2 Nefunkční požadavky

- **N1 – Operační systémy**
  Aplikace podporuje operační systémy Windows a MacOS.

- **N2 – Komunikace s Adobe Lightroom Classic**
  Aplikace komunikuje s Adobe Lightroom pomocí zásuvného modulu, který je součástí projektu Swivel.

- **N3 – Ovládání Adobe Lightroom Classic**
  Aplikace ovládá Adobe Lightroom pouze, pokud je Adobe Lightroom aktivní aplikací.

4.4 Případy použití

- **UC1 – Vytvoření nového *flow***
  Aktéři: Uživatel, Aplikace
  Scénář:
  1. Uživatel klikne na tlačítko vytvoření nového *flow*.
  2. Aplikace vytvoří nový *flow* a přepne uživatele na jeho zobrazení.

- **UC2 – Přidání nového kroku do *flow***
  Aktéři: Aplikace, Uživatel
  Scénář:
  1. Uživatel stiskne tlačítko na přidání kroku.
  2. Aplikace přidá krok na konec *flow*.

- **UC3 – Odebrání kroku z *flow***
  Aktéři: Aplikace, Uživatel
  Scénář:
  1. Uživatel označí krok ve *flow*.
  2. Aplikace uživateli ukáže nabídku na změnu *flow*.
  3. Uživatel klikne na ikonu odstranění kroku.
  4. Aplikace odstraní krok z *flow*.

- **UC4 – Změna kroku ve *flow***
  Aktéři: Aplikace, Uživatel
  Scénář:
  1. Uživatel vybere krok, který chce upravit.
  2. Aplikace nabídne uživateli nabídku s možnými úpravami.
  3. Uživatel upraví krok.

4. Aplikace uloží upravený krok.

- **UC5 – Vybrání aktivního *flow***
  Aktéři: Aplikace, Uživatel
  Scénář:
  1. Uživatel vybere, které *flow* chce aktivovat ze seznamu *flow*.
  2. Aplikace aktivuje vybrané *flow*.

- **UC6 – Zobrazení *flow***
  Aktéři: Aplikace, Uživatel
  Scénář:
  1. Uživatel vybere ze seznamu, které *flow* chce zobrazit.
  2. Aplikace vybrané *flow* zobrazí.

- **UC7 – Nastavení funkce tlačítka**
  Aktéři: Aplikace, Uživatel
  Scénář:
  1. Uživatel vybere možnost úpravy funkce tlačítka.
  2. Uživatel vybere, které tlačítko chce upravit.
  3. Uživatel vybere funkci, kterou chce tlačítku přiřadit.
  4. Aplikace uloží vybranou funkci.

- **UC8 – Posunutí se o krok ve *flow***
  Aktéři: Aplikace, Uživatel
  Scénář:
  1. Uživatel otočí spodním prstencem.
  2. Ovladač pošle informaci o změně hodnoty prstence aplikaci.
  3. Aplikace posune *flow* o jeden krok.

- **UC9 – Změna módu**
  Aktéři: Uživatel, Ovladač, Aplikace, Lightroom
  Scénář:
  1. Uživatel přepne přepínač módu na ovladači.
  2. Ovladač detekuje změnu a pošle informaci o změně aplikaci.
  3. Aplikace změní svůj stav a přepne Lightroom do vyžadovaného módu.

- **UC10 – Změna hodnoty parametru fotky**
  Aktéři: Uživatel, Aplikace, Ovladač, Lightroom
  Scénář:
  1. Uživatel vybere krok s parametrem, pomocí ovladače, který chce upravit.
  2. Uživatel otočí horním otočným prstencem ovladače.
  3. Ovladač pošle aplikaci zprávu o otočení.
  4. Aplikace přijme zprávu, zpracuje ji a pošle Adobe Lightroom příkaz, aby změnil vybraný parametr.

- **UC11 – Informování uživatele o nízkém stavu baterie**
  Aktéři: Uživatel, Aplikace, Ovladač
  Scénář:

1. Když ovladač detekuje, že baterie je nabitá na méně než 20 %, tak pošle zprávu aplikaci.
2. Ovladač začne signalizovat nízký stav baterie.
3. Aplikace pošle uživateli notifikaci o tom, že je baterie skoro vybitá.

- **UC12 – Zpětná vazba když uživatel dosáhne koncové hodnoty parametru a pokusí se pokračovat v úpravě**
  Aktéři: Uživatel, Aplikace, Ovladač, Lightroom
  Scénář:
  1. Uživatel se pokusí nastavit hodnotu parametru mimo rozsah.
  2. Aplikace vyhodnotí, že se uživatel pokusil nastavit hodnotu mimo mezní hodnoty a pošle o tom informaci ovladači.
  3. Ovladač dá uživateli zpětnou vazbu (vibrací), že překročil mezní hodnotu.

4.5 Desktopové aplikace

Desktopové aplikace lze rozdělit na dva druhy. A to na nativní a mezi-platformní.

Nativní aplikace jsou takové aplikace, které jsou vytvořené pro použití na specifické platformě. Platformou může být operační systém nebo specifický hardware. Výhodou tohoto přístupu je lepší přístup k specifickým funkcím dané platformy, nativní uživatelské prostředí a lepší možnost optimalizace výkonu. Velkou nevýhodou nativních aplikací je potřeba vytvořit a udržovat více aplikací v různých technologiích, pokud je potřeba danou aplikaci dodávat pro více platforem[11].

Tento problém řeší mezi-platformní aplikace. Tyto aplikace využívají technologie, které umožňují vytvářet aplikace, které mohou fungovat na více platformách ze stejného kódu. Tyto technologie umožňují přístup k nativním funkcím, které jsou společné mezi jednotlivými platformami, ale již nemusí umožňovat přístup k specifickým funkcím jedné platformy.

Electron je open-source mezi-platformní framework umožňující vývoj aplikací v JavaScriptu, HTML a CSS. A to díky tomu, že každá aplikace obsahuje Chromium a Node.js Tyto použité technologie umožňují využití celého ekosystému kolem webového vývoje, což zrychluje a zjednoduše vývoj.[12].

Tauri je open-source framework pro vývoj mezi-platformních aplikací pomocí kombinace Rustu a HTML vykresleného ve *Webview*. Aplikace vytvořené pomocí Tauri jsou malé, protože používají *Webview* z OS, na kterém běží[13]. To je rozdílné oproti Electronu, jehož součástí je celé Chromium. Díky podpoře Rustu, lze jednoduše interagovat s OS[14].

Qt je framework pro vývoj jak desktopových aplikací, tak i aplikací pro různá embedded zařízení. Krom vývoje samotné aplikace také nabízí nástroje na její design, testování a optimalizaci. Qt je C++ framework, ale má podporu i pro jiné programovací jazyky, například pro Python[15]. Na rozdíl od Tauri a Electronu je open-source pouze část frameworku a to pod LGPL nebo GPL licencí. Tauri a Elektron používají MIT licenci. Pro přístup k celému frameworku je nutné platit měsíční poplatek[16].

4.6 USB sběrnice

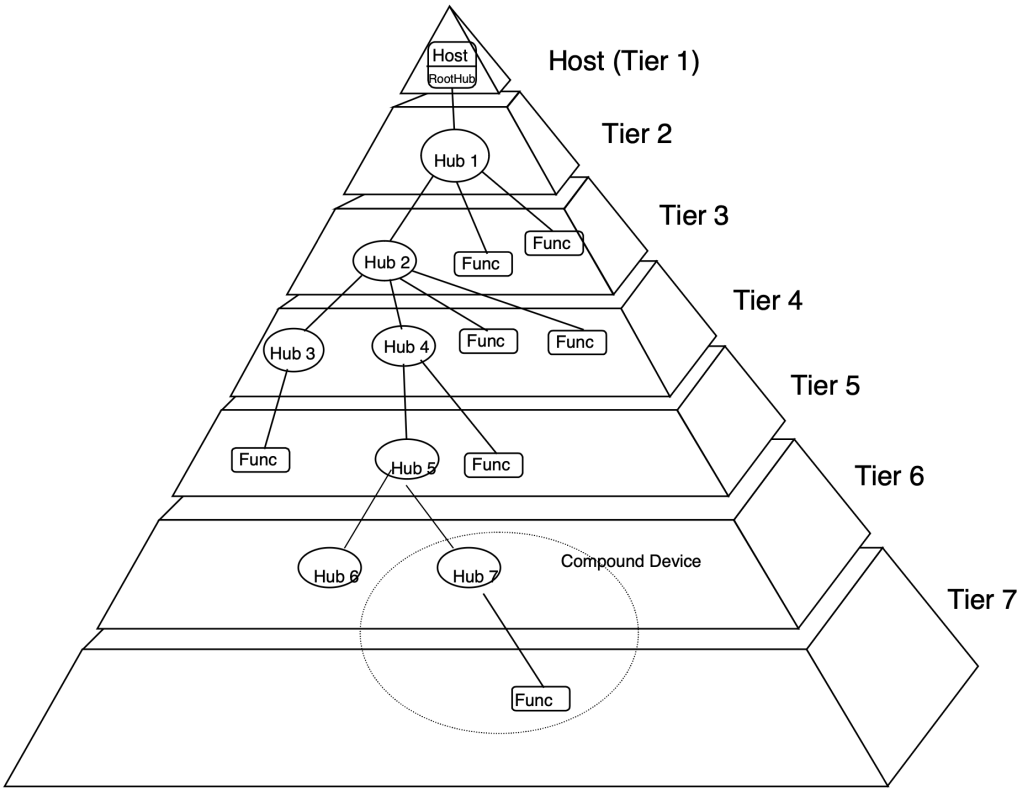
USB vzniklo kvůli potřebě na komunikační rozhraní s větší přenosovou rychlostí než sériový a paralelní rozhraní. První verze specifikace USB byla vydána v roce 1996 a k uživatelům se poprvé dostalo s vydáním Windows 95 OEM Service Release 2. Z důvodu limitované, nekvalitní a nestabilní podpory této verze USB bylo k dispozici jen málo zařízení podporující tuto verzi USB. Se zvyšováním výpočetního výkonu rostla i potřeba rychlejší komunikace a z toho důvodu

vznikly další revize USB, které podporují větší přenosové rychlosti[17]. Aktuálně nejnovější verze je USB4, která podporuje rychlost přenosu dat až 80 Gbps[18].

Pro potřeby ovladače stačí verze USB 2.0, proto se nebudeme zabývat funkcionalitami vyšších verzí USB. Tato verze USB podporuje tři rychlosti a to Low, High a Full speed, které mají přenosovou rychlost 1,5 Mbps, 12 Mbps a 480 Mbps[17].

4.6.1 Topologie USB sběrnice

USB sběrnice se sestává ze dvou hlavních součástí a to z jednoho *USB host* a více *USB device*. Ve středu sběrnice, která má topologii víceúrovňové hvězdy 4.2, se nachází *USB host* a uzly tvoří *USB device*. Tyto zařízení mohou sloužit buď jako koncové zařízení, které poskytuje nějakou službu (myš, klávesnice) nebo jako rozbočovač. Rozbočovače vytvářejí další úroveň, kterých může být jen sedm a v poslední úrovni mohou být pouze koncová zařízení. Existují také složená zařízení, která v sobě mají rozbočovač, na který jsou připojená koncová zařízení.



Obrázek 4.2 Topologie USB sběrnice

4.6.2 Komunikace

USB nabízí komunikační služby mezi aplikacemi na straně Hosta a USB funkcemi jednotlivých zařízení. Každá funkce může používat jiný způsob komunikace, tento způsob komunikace je definován v takzvaném *endpointu*. Zařízení se může sestávat z více nezávislých *endpointů*, kde každý *endpoint* má své číslo a směr komunikace. Společně s adresou zařízení je možné jednotlivé *endpointy* identifikovat. Každý *endpoint* obsahuje tyto informace:

- směr komunikace, Buď od zařízení k hostu (směr *IN*) nebo opačně (směr *OUT*)

- maximální velikost paketu
- číslo endpointu
- způsob řešení chyb při komunikaci
- typ přenosu
- maximální latenci nebo frekvenci přenosu
- přenosovou rychlost

Nulový endpoint je speciální endpoint určený ke komunikaci s ovladačem USB zařízení na hostitelském počítači. Tento endpoint umožňuje čtení konfigurace zařízení a zároveň umožňuje nastavení chování zařízení. Tento endpoint musí být k dispozici jakmile je zařízená připojené, má napájení a přijalo reset signál.

USB pipe představuje spojení mezi jedním endpointem na straně zařízení a softwarem na straně hostitelského zařízení. Existují dva druhy *pipe*.

Message pipe přenáší data, která přenáší mají význam pro USB. Společně s nulovým endopointem vytváří takzvanou *Default Control Pipe*. Tato *pipe* je použitá pro nastavení zařízení.

Stream pipe přenáši data, která nemají význam pro USB. Je to typ *pipe*, který využívají všechny endpointy krom nulového.

S každou *pipe* je krom endpointu spojena přenosová kapacita, její stav a typ přenosu, který využívá. Každý typ přenosu má své vlastnosti, které jsou popsané v následujícím seznamu:

Controll transfer podporuje komunikaci určenou pro nastavení a získávání stavu jednotlivých funkcí. Tento způsob komunikace používá nulový *endpoit* pro zjištění konfigurace zařízení a jeho případné nastavení.

Isochronous transfer je určený pro zařízení, která potřebují nízkou latenci a konstantní přenos dat. Většinou se takto přenáší časové závislá data. Tento typ přenosu negarantuje doručení všech dat. Pouze přijímač dokáže poznat, kdy došlo k chybě a kolik dat bylo ztraceno.

Interrupt transfer je navržený pro zařízení, která nepotřebují posílat velké množství dat pravidelně. Ale potřebují mít garantovanou maximální dobu, do kdy budou data odeslána, a pokud se odeslání nezdaří, tak budou znovu odeslána později.

Bulk tranfer slouží pro přenos velkého množství dat, u kterého nutně nezávisí na času doručení.

Komunikace na USB sběrnici je řízená hostitelským zařízením. Protože lze za běhu přidávat nebo odebírat nová zařízení, tak host provádí enumeraci. Enumerace je způsob jak identifikovat zařízení, přiřadit jim unikátní adresy a zjistit jejich konfiguraci. A také slouží k detekci odpojených zařízení.

4.6.3 USB zařízení

USB zařízení, anglicky *USB device*, mohou být rozbočovače nebo koncová zařízení. Pro identifikaci zařízení a zjištění co za funkce poskytuje slouží USB deskriptor. Tento deskriptor obsahuje základní informace o zařízení.

- verzi USB specifikace
- třídu a podtřídu zařízení
- jméno zařízení

- jméno výrobce
- id výrobce a id produktu
- sériové číslo
- počet konfigurací

Zařízení musí mít aspoň jednu konfiguraci. Konfigurace jsou popsané svým vlastním deskriptorem. Konfigurace se skládá z několika rozhraní. Jednotlivý rozhraní se mohou skládat z několika endpointů. Endpointy nemůžou být sdílené mezi rozhraními, leda by se jednalo o alternativní nastavení rozhraní. V jeden moment může být aktivní pouze jedna konfigurace, která je vybrána hostitelským zařízením.

4.6.4 USB třídy

USB třídy slouží k sloučení zařízení nebo rozhraní s podobnými atributy a funkcemi do jedné skupiny. Toto sloučení zjednodušuje vývoj ovladačů na straně hostitelských zařízení, protože specifikuje jakým způsobem komunikují zařízení dané třídy a jaká data si posílají.

Třídy můžou mít své vlastní deskriptory, které blíže popisují způsob komunikace. Mezi takové třídy se řadí *Human Interface Device class*. Do této třídy patří zařízení, se kterými interagují lidé, jako například myši, klávesnice, joysticky či další zařízení například čtečky čárových kódu nebo teploměry[**usbdeviceworkinggroupUniversalSerialBus**].

HID přidává své dva vlastní deskriptory. První deskriptor se nazývá *Report descriptor* a popisuje jaká data pošle rozhraní, ke kterému je přiřazen. Tyto informace využívá ovladač USB pro rozhodování, co s přijatými daty udělat. Například deskriptor od myši popisuje, jaká část přijatých dat znamená pohyb myši a která znamenají stisknutí tlačítek. Druhý deskriptor je *Physical descriptor* a popisuje, kterou částí těla se se zařízením interaguje.

Report deskriptor se skládá z položek, které popisují, jaká data a s jakým účelem se posílají.

HID zařízení komunikují s hostitelským zařízením buď pomocí *Default control pipe* nebo pomocí *interrupt transfer*. Komunikace přes *Default control pipe* slouží ke konfiguraci zařízení a přenosu deskriptorů a *Interrupt transfer* slouží pro posílání dat hostitelskému zařízení s nízkou latencí nebo přijímání dat od hostitelského zařízení[19].

4.7 Bluetooth Low Energy

Bluetooth Low Energy, zkráceně BLE, je technologie pro bezdrátovou komunikaci mezi zařízeními. Tento komunikační standart je dnes dostupný na většině prodávaných zařízení pro koncové uživatele. Oproti předchozí verzi *Bluetooth Classic* je *Bluetooth Low Energy* energeticky méně náročný a také přiřazuje více práce a odpovědnosti zařízením, která mají více energie než zařízením s malými bateriemi či jinak omezenými zdroji energie. Dále podporuje více způsobů komunikace než **Bluetooth Classic**, které podporuje pouze *point to point* komunikaci, *BLE* podporuje jak *point to point*, tak i *mesh network* nebo *broadcasting*[20]. Tento protokol funguje na frekvenci 2,4 GHz, kterou je možné využívat bez licence. Tuto frekvenci využívá také WiFi a další bezdrátové komunikační protokoly.

4.7.1 Attribute protocol

Tento protokol rozlišuje zařízení na server a klienta. Stejné zařízení se může ve stejný moment vystupovat jako server a zařízení, a to i při komunikaci s pouze jedním dalším zařízením

Server vytváří jeden nebo více atributů, které jsou uložené v tabulce atributů. Klient může interagovat s atributy pomocí těchto operací:

Read čtení hodnoty atributu

Write Command zápis hodnoty atributy bez potvrzení

Write Request zápis hodnoty atributy s potvrzením

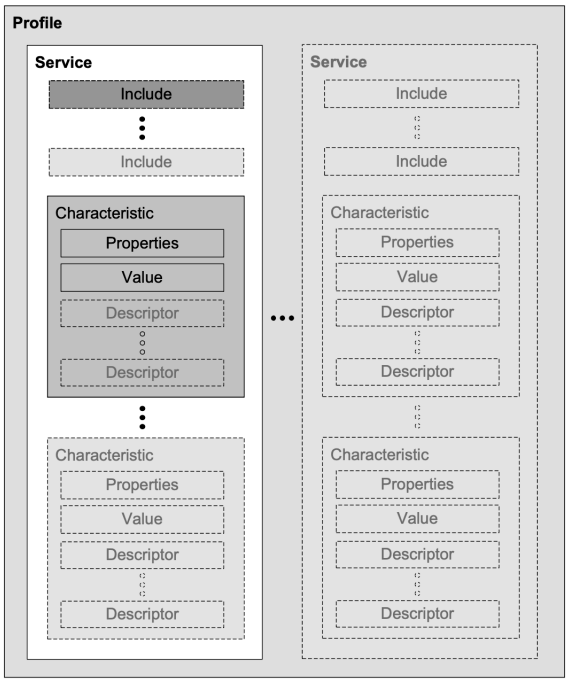
Notification a Indication Server zašle klientovi informace o atributu bez nutnosti, aby klient inicioval komunikaci. Pokud se jedná o notifikaci, tak klient nemusí potvrdit přijetí, v případě indikace musí potvrdit přijetí do 30 sekund od odeslání[21].

Jednotlivé atributy se skládají z následujících položek:

- UUID které označuje typ atributu.
- Identifikátor (*handle*), které identifikuje daný atribut v tabulce atributů.
- Pravomoci, co lze s daným atributem dělat. Tyto pravomoci jsou definovány vyšší vrstvou protokolu. Definují například, jestli ho lze číst či upravovat a zda je potřeba zabezpečená komunikace, autorizace či autentizace.
- Hodnota atributu, atribut může mít jak fixní velikost, tak i dynamickou velikost[21].

4.7.2 Generic attribute profile

GATT je komunikační protokol postavený na protokolu ATT. Tento protokol vytváří z atributů z ATT datovou strukturu služeb, charakteristik a deskriptorů. Tuto strukturu můžete vidět na diagramu 4.3. Tento protokol také dělí zařízení na server a klienty, přičemž stále platí, že zařízení může mít obě role ve stejnou dobu.



Obrázek 4.3 Struktura GATT[21]

Jednotlivé služby jsou identifikovány buď 16-bitovým UUID, které je přidělené *Blutetooth SIG* nebo 128-bitové UUID. Služba spojuje charakteristiky a dává jim kontext, jak je využívat.

Charakteristika jsou už samotná data s identifikátorem a definicí, jak může klient s daty interagovat a jaká k tomu potřebuje oprávnění. Dále se k charakteristice může vázat deskriptor.

Ten může přidávat kontext k dané charakteristice. Například deskriptor *Characteristic User Description Desriptor* přidává k charakteristice její textový popis a deskriptor *Client Characteristic Configuration Descriptor*, zkráceně *CCCD* má vlastní instanci pro každého klienta a umožňuje klientovi nastavit, jestli chce být notifikovaný nebo indikovaný o dané charakteristice. Toto nastavení se provádí pomocí bitového pole, ze kterého se aktuálně využívají pouze dva bity[21].

4.7.3 Generic acces protocol

GAP je protokol, který se stará o vytvoření spojení mezi zařízeními, jejich objevování a komunikaci bez vytvoření spojení. Toto je rozdíl oproti GATT, který řeší komunikaci po vytvoření spojení. GAP této funkcionality dosahuje pomocí vysílání a zpracovávání *advertising* paketů. Tyto pakety mohou obsahovat informace o jménu a typu zařízení, podporovaných službách, a jestli je zařízení objevitelné a jestli podporuje vytvoření spojení. GAP udává zařízením čtyři role. Zařízení mohou tyto role v průběhu času měnit a mohou se reprezentovat více rolí najednou. Tyto role jsou:

Broadcaster vysílá informace pomocí *advertising* paktů. Tyto pakety obsahují informace o zařízení a jeho funkcích a službách.

Observer přijímá *advertising* události.

Peripheral je zařízení, které má spojení se zařízením v roli *Central*.

Central je zařízení, které iniciuje vytvoření spojení s dalším zařízením[21].

4.7.4 Zprostředkování HID přes Bluetooth Low Energy

Bluetooth SIG definuje profile *HID over BLE*, který umožňuje přenášet HID přes BLE. Je nadstavba nad *Generic Attribute Protokol*. Tento profile se skládá z několika GATT služeb[22].

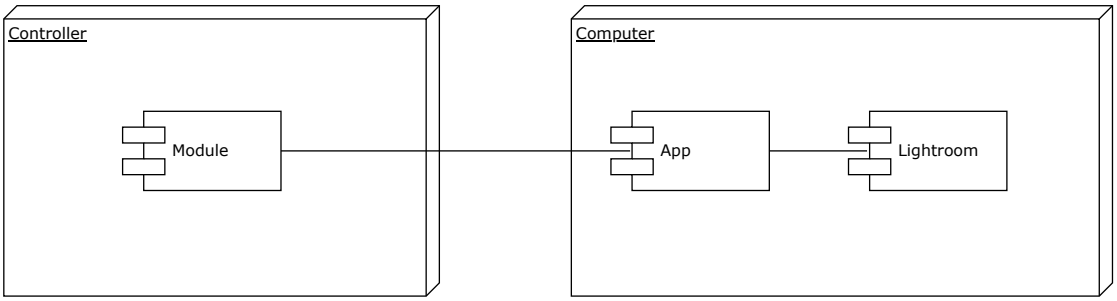
HID service Tato služba poskytuje HID funkcionalitu. Pomocí GATT charakteristik umožňuje hostitelskému zařízení přečíst USB deskriptory a přečíst či zapsat jednotlivé reporty. Vstupní reporty mají CCC deskriptor a tudíž podporují notifikace nebo indikace[23].

Battery service Tato služba informuje hostitelské zařízení o aktuálním stavu baterie a je také založena na GATT charakteristikách[24].

Device information service Tato služba poskytuje informace o zařízení. V této službě jsou všechny charakteristiky definovány jako *optional*[25], ale *HID over GATT profile* udává, že musí být přítomna charakteristika *PnP ID*. Tato charakteristika obsahuje USB *Vendor* a *Product* ID[25].

4.8 Architektura projektu

Projekt se skládá ze tří hlavních částí, a to z *aplikace*, *zásuvného modulu pro Adobe Lightroom* a *ovladače*. Každá z těchto částí má jiné požadavky na svou funkcionalitu a technologie, které budou rozebrány ve vlastních kapitolách. Znázornění projektu a interakce mezi jednotlivými částmi lze vidět v diagramu 4.4.



Obrázek 4.4 Diagram projektu

4.9 Aplikace

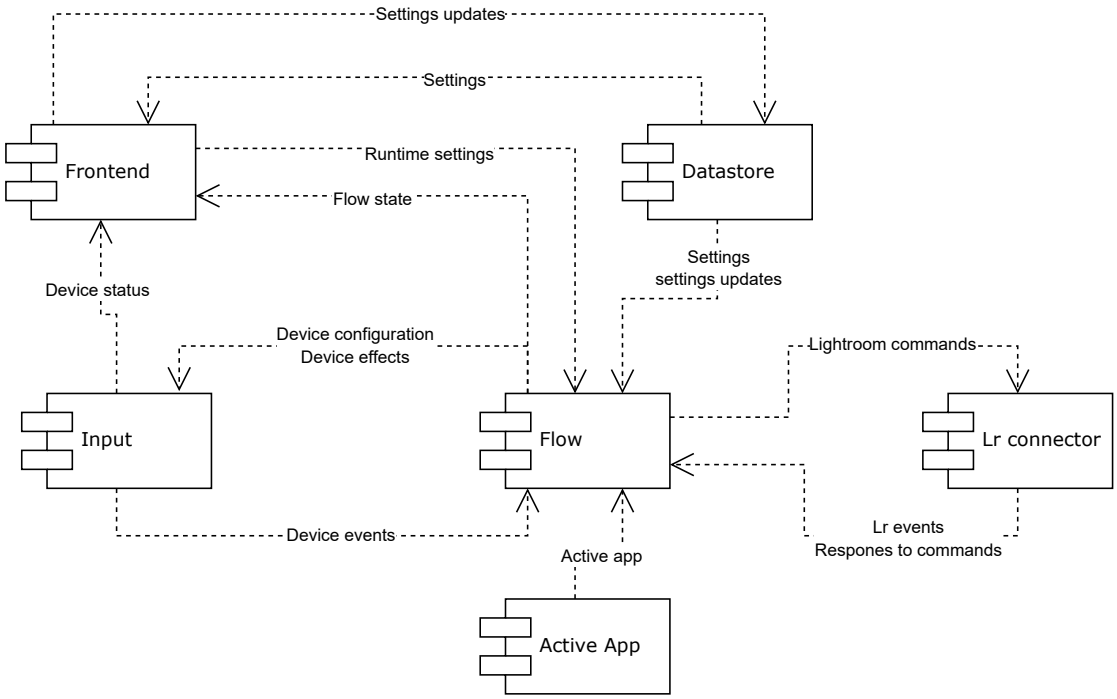
Aplikace je hlavní součást celého projektu. Zde se potkává interakce uživatele s ovladačem, jeho *flow* a další nastavení a z této kombinace vzejdou akce, které se mají provést v Adobe Lightroom. Aplikaci lze rozdělit na dvě hlavní části. A to na *backend* a *uživatelské rozhraní*.

Pro vývoj aplikace bude zvolen framework Tauri. A to kvůli tomu, že umožňuje využít jakýkoliv webový framework pro vývoj uživatelského prostředí a na rozdíl od Electron.js je výsledná aplikace malá, protože neobsahuje celé Chromium.

4.9.1 Backend

Backendová část aplikace je místo, kde se potkávají všechny části celého projektu. Má na starosti ukládání a aktualizaci *flow* F2, komunikaci s ovladačem F6, Adobe Lightroom F7 a uživatelem pomocí uživatelského prostředí F8 a také má na starosti vyhodnocení samotného *flow* F3.

Kvůli požadavku na udržovatelnost a jednoduchou rozšiřitelnost aplikace, bude aplikace rozdělena na několik částí. Každá část má svůj jeden hlavní úkol. Budou vytvořeny tři hlavní komponenty, a to komponenta zodpovědnou za komunikaci s ovladačem, další odpovědnou za komunikaci s Adobe Lightroom a třetí, která bude vyhodnocovat *flow* a s pomocí komunikačních komponent komunikovat s Adobe Lightroom a ovladačem. Dále budou potřeba komponenty na ukládání nastavení, sledování aktivní aplikace a komunikaci s uživatelským rozhraním. Jednotlivé komponenty a to, jak mezi budou komunikovat je znázorněno v diagramu 4.5.



Obrázek 4.5 Diagram backendu aplikace

4.9.1.1 Vyhodnocení *flow*

Samotná myšlenka *flow* je relativně jednoduchá. Uživatel si vybere krok, který chce upravit a pak změní jeho hodnotu. Ale do vyhodnocování vstupují další proměnné. Například akce přiřazené tlačítkům, aktuální vybraný mód nebo jestli je Adobe Lightroom aktivní. V diagramu 5.3 je znázorněno, jak by vyhodnocení mělo vypadat.

Pokud se nacházíme v módu úprav fotek, tak prvně dostane vstup *flow*, pokud ho nezpracuje, tak ho dostanou tlačítka, a pokud ani oni ho nezpracují, tak se nic nestane. A pokud jsme v módu výběru fotek, tak vstupy prvně dostane vstup *flow* pro výběr fotek a následně případně tlačítka. A toto se stane jen, pokud je aktivní aplikací Adobe Lightroom.

4.9.1.2 Databáze a ukládání konfigurace

Pro splnění požadavku F2 aplikace musí aplikace lokálně ukládat *flow* a nastavení. Nabízí se dva způsoby, jak tento požadavek vyřešit. Prvním z nich je použít lokální databázi, jako například *SQLite*. Druhý přístup, který lze využít, je ukládat nastavení aplikace do jednoho v libovolném námi vybraném formátu. Lze použít například *JSON*, *TOML* či jiný formát nebo vytvořit specifický pro toto použití.

Výhodou uložení dat do textového formátu je jeho jednoduchost. Hlavně díky již existujícím knihovnám umožňujícím jednoduše převádět data z různých formátů do struktur jazyka a naopak. Dále oproti uložení v databázi není potřeba řešit převod mezi reprezentací v aplikaci a v databázi a migraci. Migrace ale mohou být i výhodou pro databázi, protože při změně dat migrace umožňují jednoduše převést data ze starého schématu na nové.

4.9.1.3 Komunikace s uživatelským rozhraním

Tauri poskytuje dva způsoby, jak komunikovat s uživatelským prostředím. První způsob Tauri nazývá *Command* a slouží k volání backendu z uživatelského prostředí. Umožňuje volat funkce

v backendu, předávat jim argumenty a přijímat vrácená data.

Druhý způsob je nazývaný *Event*. A umožňuje komunikaci, jak od backendu k uživatelskému prostředí, tak opačně. Další rozdíl oproti *Command* je ten, že je to jednosměrná komunikace. Příjemce nemůže odpovědět na *Event*. A *Event* může být buď globální nebo pro specifické okno. Pro přijetí události musí uživatelské prostředí či backend poslouchat.

Pro funkcionalitu uživatelského prostředí musí backend poskytovat CRUD operace nad *flow* a funkcemi tlačítek. Dále je potřeba čtení aktuálního stavu olvadače a *flow*. Tyto operace budou poskytovaný pomocí *Command*. A backend bude vytvářet události o změnách stavu *flow* a ovladače.

4.9.2 Uživatelské rozhraní

Uživatelské rozhraní můžeme rozdělit na dvě části. První část je o tvorbě a úpravě *flow* a dalších nastavení a druhá část se týká informování o stavu ovladače a aktuální pozici ve *flow*.

Aktuální stav *flow* se bude nacházet v samostatném okně, které bude vždy nad ostatními okny. Toto stavové okno bude ukazovat aktuální, následující a předchozí krok ve *flow* a informace o aktuálním kroku. Pro některé to bude hodnota upravovaného parametru, pro jiné to může být název aktuálně vybraného presetu.

Úprava *flow*, nastavení a výběr aktivního *flow* se bude nacházet společně s ukazatelem stavu ovladače (jestli je připojen a úroveň nabití baterie) v hlavním okně aplikace.

Pro vývoj uživatelského rozhraní aplikace lze použít libovolné technologie pro vývoj webových aplikací, protože bylo vybralo *Tauri* pro vývoj aplikace. Toto umožňuje použít JavaScriptovou knihovna *React* pro vývoj uživatelského prostředí. Tato knihovna umožní rozdělit uživatelské prostředí na jednotlivé komponenty. Dále zjednoduší tvorbu prostředí díky velkému množství knihoven, kde využijeme například knihovnu *dnd kit*[26] pro změnu pořadí ve *flow*.

4.10 Ovladač

Hlavním úkolem ovladače je zachytit interakce uživatele F2 a poslat je počítači F1. Dále je nutné dodržet rozmístění jednotlivých prvků, které je dané designem ovladače N2. Protože ovladač nebude provádět náročné výpočty a zároveň bude napájen z baterií, tak je ideální použít MCU.

4.10.1 Výběr MCU

Pro splnění požadavků práce je potřeba MCU, které je schopno komunikovat s počítačem pomocí USB a Bluetooth podle N1 a zároveň má dostatek pinů na komunikaci s periferiemi podle F2. Krom těchto dvou požadavků je výhodné, aby bylo MCU dostupné ve formě modulu, který má certifikaci od FCC a CE RED. A to kvůli zjednodušení návrhu PCB a zjednodušení případné certifikace celého produktu. A pak další požadavky jsou, co nejmenší spotřeba energie, cena modulu a jednoduchost pájení.

Z dostupných možností byly vybrány 3 nejvhodnější MCU a to ESP32-S3 od společnosti Espressif, NRF52833 od Nordic semiconductor a DA14695 od Renesas. Všechny tato MCU splňují požadavky na jejich rozhraní a jsou dostupná v certifikovaných modulech.

Pro nRF52833 existuje modul MS88SF21, který má všechny kontakty dostupné, ale nemá vyvedený VBUS na externí kontakt. Takže také není vhodný pro použití v ovladači, protože potřebujeme VBUS k detekci připojení USB. Pro nRF52833 existují také další moduly jako například MK07A od společnosti MOKOsmart či E73-2G4M08S1E od EBYTE, které mají kontakty jak ze stran, tak i zespodu. Bohužel jiné moduly s kontakty pouze na stranách pro NRF52833 s potřebnými certifikacemi neexistují. To samé platí pro DA14695. S dostupností kontaktů je na tom nejlépe ESP32-S3 s modulem ESP32-S3-WROOM-1, který má všechny potřebné kontakty dostupné. Má jeden velký kontakt zespodu, ten ale není nutné napájet k plné funkcionalitě.

Modul	MCU	spotřeba při odesílání	spotřeba při přijímání	zdroj
ESP32-S3-WROOM-1-N4	ESP32-S3	285mA@3.3V	97mA@3.3V	[27]
MK07A	NRF52833	7mA@3V	5,2mA@3V	[28]
MS88SF21	NRF52833	9,6mA@3V	5,2mA@3V	[29]
DA14695MOD	DA14695	3mA@3V	1,8mA@3V	[30]

Tabulka 4.1 Porovnání modulů podle spotřeby

Modul	MCU	Cena	Zdroj
ESP32-S3-WROOM-1-N4	ESP32-S3	3 USD	[31]
MK07A	NRF52833	6 USD	[32]
MS88SF21	NRF52833	5 USD	[33]
DA14695MOD	DA14695	14 USD	[34]

Tabulka 4.2 Porovnání modulů podle ceny

ESP32-S3 má sice nejlépe dostupné kontakty, ale má největší spotřebu energie při komunikaci 4.1. A požadavek na nízkou spotřebu má větší váhu než jednoduchost pájení. Z toho důvodu ESP32-S3 vyřadíme z výběru.

Takže zbývají pouze dva moduly a to DA14695MOD a MK07A. A z těchto dvou modulů bude použit MK07A i přesto, že má přibližně dvojnásobnou spotřebu při komunikaci. A to protože je levnější než DA14695MOD.

4.10.1.1 Výběr ostatních komponent

Pro sledování stavu baterie F3 bude použit čip MAX17260 od Analog Devices. Tento čip umí sledovat úroveň nabití baterie a odhadovat, za jak dlouho se baterie vybije či nabije a pro komunikaci s MCU používá sběrnici I2C. Na nabíjení baterie použijeme čip MCP73831.

Pro komunikaci stavu ovladače s uživatelem F4 a N3 bude použita programovatelná LED typu WS2812B v balení 5050. Této diodě lze jednoduše nastavit její barvu v RGB formátu. Tato funkcionalita umožní jednoduše vytvářet různé animace pro informování uživatele o stavu ovladače. Mezi tyto stavy patří například nízký stav baterie, zda-li je baterie nabitá, párování a další. Pro haptickou zpětnou vazbu bude použit vibrační motor a čip DRV2605L od Texas Instruments. Tento čip je určený k ovládání vibračních motorů a má v sobě zabudovanou knihovnu efektů a také s ním lze komunikovat přes I2C sběrnici.

Otáčení prstenců bude sledováno pomocí rotačních enkodéru. Spodní enkodér, který uživatel používá pro pohyb mezi fotkami nebo kroky *flow*, bude mít zarážky. Horní enkodér, který uživateli slouží pro nastavení hodnot parametrů, bude mít hladký chod bez zarážek.

V přepínači je umístěn magnet a magnetické pole tohoto magnetu je snímáno pomocí hallových senzorů a to přesně DRV5032 od společnosti Texas Instruments v balení SOT-23. V tomto balení existuje velké množství dalších hallových senzorů, takže pokud by tento senzor byl z nějakého důvodu nevyhovující, tak ho lze jednoduše vyměnit za jiný.

4.10.2 Firmware ovladače

Firmware ovladače bude muset sledovat jak uživatel interaguje s jednotlivými ovládacími prvky, sledovat stav baterie, komunikovat s aplikací běžící na počítači a zároveň reagovat komunikovat uživateli změny ve svém stavu.

4.10.2.1 nRF Connect SDK a Zephyr RTOS

nRF Connect SDK je kit na vývoj firmwaru pro řadu MCU nRF52 a další MCU od Nordic semiconductor. Je založený na Zephyr RTOS. Zephyr je projekt spadající pod Linux Foundation, jehož cílem je vytvořit operační systém pro systémy s omezenými zdroji[35].

Zephyr poskytuje vývojářům mnoho služeb, zde uvedu jen ty základní:

- Vlákna
- Obsluhovaní přerušení
- Správu paměti
- Více vláknovou synchronizaci
- Mezi vláknovou komunikaci
- Správu spotřeby energie
- Podporu Bluetooth Low Energy 5.0

NRF Connect SDK ještě rozšiřuje funkčnost Zephyr RTOS o další knihovny a potřebné nástroje. Poskytuje například knihovnu Application Event Manager. Tato knihovna je určená pro vývoj *event driven* aplikací. Dělí aplikaci na moduly, které spolu komunikují pomocí zpráv. Jednotlivé moduly se mohou přihlásit k odběru zprávy a Application Event Manager za běhu programu zajistí, že zprávu dostanou, pokud se k ní přihlásili[36].

Na této knihovně je postavena další knihovna jménem Common Application Framework. Tato knihovna poskytuje hotové moduly, které podporují správu Bluetooth spojení, detekci stisknutí tlačítek či sledování senzorů[37].

Krom těchto dvou knihoven obsahuje knihovny pro různé Bluetooth LE služby. Například *GATT Human Interface Device Service*, *Bluetooth Advertising*, *GATT Battery service* a mnoho dalších.

4.10.2.2 Návrh architektury FW

Pro vývoj firmware použijeme CAF, protože obsahuje moduly, které řeší správu připojování přes Bluetooth a další funkcionalitu, kterou nebude muset vytvářet sami.

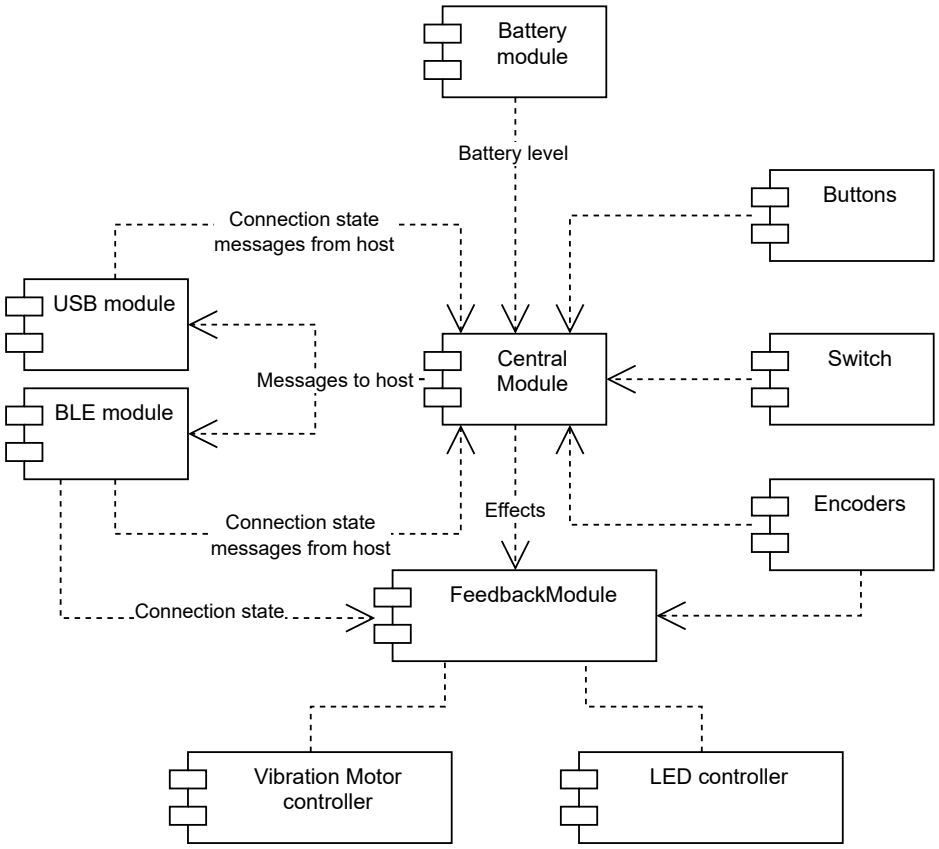
K modulům, které poskytuje CAF je potřeba přidat další moduly a události. A to modul který spravuje jednotlivé vstupy, modul, který kontroluje stav baterie, modul zpětné vazby a nakonec moduly, které se starají o komunikaci mezi ovladačem a počítačem. Jejich schématické znázornění a vzájemnou interakci popisuje diagram 4.6.

4.10.3 Komunikace mezi firmwarem a aplikací

Komunikaci lze rozdělit na dvě fáze. První fáze obsahuje navázání spojení, zjištění stavu zařízení. Druhá fáze je primárně o posílání zpráv o změnách stavu ovladače či změnách konfigurace ovladače.

V první fázi je potřeba zjistit stav ovladače. První zpráva bude požadavek na zaslání informací o verzi firmwaru a hardwaru ovladače. Toto je důležité, pokud by se v budoucnosti změnila komunikaci či přidala funkcionalitu hardwaru, tak to bude možné poznat na straně aplikace a přizpůsobit se tomu. Dále následuje požadavek na zaslání aktuálního stavu ovladače. A po této první fázi může již pokračovat normální komunikace. Toto je schématicky znázorněno v diagramu 4.7.

Během normální komunikace ovladač posílá aplikaci změny stavu, aplikace ovladači posílá změny nastavení a to, že má ovladač přehrát efekt na vibračním motoru. Toto nastavení obsahuje



Obrázek 4.6 Diagram firmwaru

velikost kroků vnitřního enkodéru a efekt, který má přehrát na vibračním motoru, pokud se změní hodnota enkodéru.

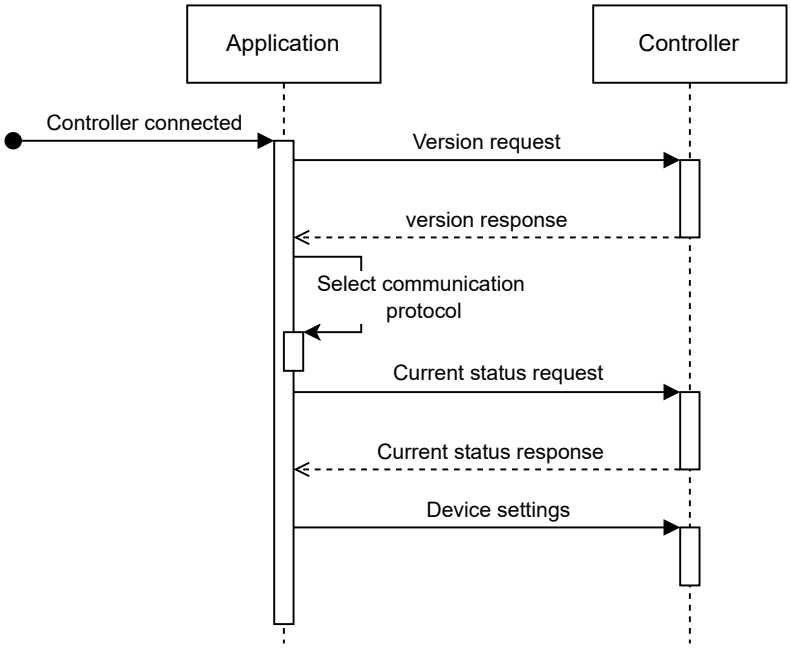
Komunikace mezi FW a aplikací bude probíhat přes USB či Bluetooth. Ovladač svými požadavky na komunikaci spadá do USB třídy Human Interface Device. Tuto komunikaci podporuje i Bluetooth LE pomocí GATT profilu[22]. HID třída používá pro svou komunikaci *interrupt transport*, který garantuje, že zprávy vždy dorazí a že dorazí do určité doby[19].

4.10.4 Kódování zpráv

Ke komunikaci bude použita třída HID, která má maximální velikost jednoho paketu 64 bajtů. Pro co nejjednodušší způsob komunikace je nutné, aby se každá zpráva vešla do jednoho paketu. Pokud by se nevešla, tak by bylo nutné přidat jak na stranu aplikace, tak i na stranu firmwaru logiku pro rozdělení jedné zprávy do více paketů.

Obsah paketů v HID je definovaným pomocí report deskriptoru. Ten definuje, jaká data paket obsahuje. Například pro myš může obsahovat změnu pozice, změnu otočení kolečka a stav tlačítek. V ovladači by bylo možné použít stejný přístup, ale to by znemožnilo měnit obsah paketů za běhu ovladače. Pro ovladač bude ideální nadefinovat podobu paketu jako 64 bajtový buffer a využít jiný způsob kódování.

Pro kódování zpráv bude využit binární formát *Protocol buffer*. Tento formát byl vytvořen společností Google Inc. je jazykově neutrální a na rozdíl od formátů typu JSON a XML přináší lepší kompresi dat[38]. Dalším velkým rozdílem mezi *Protocol Buffers* a JSON či XML je ten, že struktura zprávy musí být nejdříve nadefinována. Zprávy se definují v *.proto* souborech. Každá zpráva se může skládat z dalších zpráv nebo základních datových typů. Poté, co si uživa-



Obrázek 4.7 Komunikace ovladače a aplikace

tel nadefinuje strukturu zpráv je nutné *.proto* soubory zkompilovat pomocí *proto*kompilátoru. Tento kompilátor vygeneruje kód v požadovaném jazyce pro serializaci a deserializaci uživatelem nadefinovaných zpráv[39].

V Zephyr RTOS je již zabudovaná podpora pro použití *Protocol Buffers* pomocí knihovny Nanopb. Tato knihovna je vhodná pro použití v mikrokontrolérech, protože její velikost je malá a také umožňuje serializaci bez využití dynamické alokace paměti[40].

Kapitola 5
Implementace

Tato kapitola se zabývá implementací firmwaru, aplikace a desky podle návrhu z předchozí kapitoly. Popisuje jednotlivé použité technologie a přístupy k vývoji.

5.1 Firmware

Firmware je implementovaný v kombinaci programovacích jazyků C a C++. Základem firmwaru je knihovna *Common Aplication Framework* z *nRF Connect SDK*. Z této knihovny využijeme moduly:

- Bluetooth LE Advertising module
- Bluetooth LE State module
- Bluetooth LE Bond module
- Buttons module
- Click detector module
- Settings loader module
- Sensor manager module

5.1.1 Konfigurace

Zephyr RTOS používá pro konfiguraci hardwaru *devicetree*. *Devicetree* je hierarchická struktura, která popisuje hardware. Tato struktura se skládá z *node*, které v sobě drží informace o daném hardwaru. V Zephyru se používá pro popis dostupného hardwaru na desce a jeho úvodní konfiguraci. Pro konfiguraci softwaru používá Zephyr *Kconfig*. Pomocí *Kconfig* se například zapíná CAF a jeho před připravené moduly.

5.1.1.1 Popis hardwaru pomocí Devicetree

Devicetree soubor, který popisuje naší desku se dá rozdělit do dvou částí. V první části se zapínají a nastavují periferie MCU. Například je potřeba nastavit, na jaké piny budou přiřazeny sběrnici I2C a na jaké frekvenci bude komunikovat. Přesně toto lze vidět v ukázce 1, kde se nejdříve do *node pinctrl* přidají dvě skupiny pinů, jedna pro normální funkci a jedna pro funkci v low power módu, a pak se v *I2C* bloku přiřadí.

V I2C *node* lze dále vidět pole *compatible*. Toto pole určuje, co tato *node* popisuje a který *devicetree binding* se má použít. V tomto případě to znamená, že popisuje *TWI master with EasyDMA*, což je ovladač pro I2C periferii na MCU od Nordic Semiconductors. Další pole je *status*, které popisuje, jestli se daný hardware používá nebo ne. A nakonec se tam nachází *node drv2605l*, která reprezentuje zařízení na I2C *DRV2605L* s adresou *0x5A* sběrnici. Protože pro toto zařízení není vytvořený ovladač, tak používá generický ovladač pro I2C zařízení.

```
&pinctrl {
    i2c0_default: i2c0_default {
        group1 {
            psels = <NRF_PSEL(TWIM_SDA, 0, 26)>, <NRF_PSEL(TWIM_SCL, 0, 27)>;
        };
    };

    i2c0_sleep: i2c0_sleep {
        group1 {
            psels = <NRF_PSEL(TWIM_SDA, 0, 26)>, <NRF_PSEL(TWIM_SCL, 0, 27)>;
            low-power-enable;
        };
    };
};

&i2c0 {
    compatible = "nordic,nrf-twim";
    status = "okay";
    clock-frequency = <400000>;

    pinctrl-0 = <&i2c0_default>;
    pinctrl-1 = <&i2c0_sleep>;
    pinctrl-names = "default", "sleep";

    drv2605l: drv2605l@5a {
        compatible = "i2c-device";
        reg = <0x5a>;
        label = "DRV2605L";
    };

};
```

Výpis kódu 1 Konfigurace I2C sběrnice v Devicetree

A v druhé části se vytváří *root node*, která reprezentuje novou desku. Tato *node* je označena */*. V této *node* se nacházejí další *node*, které reprezentují další komponenty na desce. Tyto komponenty jsou encodery, tlačítka, přepínač a pin na ovládání zařízení s větší spotřebou.

5.1.1.2 Konfigurace pomocí systému Kconfig

Kconfig slouží ke konfiguraci aplikace během kompilace. Toto nastaveni se oproti *Devicetree* soustředí na nastavení funkcionalit na straně firmwaru. Například zapnutí různých GATT služeb nebo ovladače USB a dalších periférií. Toto nastavení se nachází v souboru *prj.conf* v kořenovém adresáři firmwaru.

V kódu je nastavení reprezentováno makry v souboru *autoconf.h*. Toto umožňuje vynechání nepotřebného kódu z finální aplikace a ušetřit prostor v úložišti mikrokontroleru.

5.1.2 Moduly

Přestože je Zephyr a nRF Connect SDK napsáno v programovacím jazyce C, tak podporuje i C++ a to i moderní revize jako je C++20. Sice nepodporuje všechny vlastnosti C++. Chybí například destrukce globálních objektů a podpora částí standardní knihovny, které závisejí na operačním systému[41]. Ale i přes tyto nedostatky nám umožní podpora C++ zjednodušit si vývoj. A to díky přesunutí logiky kolem zpracovávání událostí do nadřazené třídy v modulech bude stačit pouze implementovat funkce, které budou události zpracovávat. Tato rodičovská třída *ModuleBase* se skládá z jednotlivých tříd *Handler*.

Třída *Handler* je generická třída. Jako parametry dostane strukturu události a ukazatel na funkci, která převádí generickou událost na událost na zadanou událost pokud je to správná událost. Jinak vrací nulový ukazatel. Tato třída dále obsahuje jednu virtuální funkci *handle*, která je určená ke zpracovávání události. A pak obsahuje funkci *receive*, která vrací *true* nebo *false* podle toho, jestli přijala správnou událost.

Generická třída *ModuleBase* spojuje několik tříd *Handler* dohromady. Třídy *Handler* jsou dodány jako *parameter pack* a při přijetí zprávy je na nich zavolána metoda *receive* pomocí *fold expression*.

Použití *ModuleBase* lze vidět na modulu *FeedbackModule* v ukázce kódu 2. Pro vytvoření jednotlivých tříd *Handler* je využito makro, které z názvu události vytvoří základní třídu *Handler* pro danou událost.

```
class FeedbackModule
: public ModuleBase<HANDLE_EVENT(module_state_event),
                    HANDLE_EVENT(encoder_event),
                            HANDLE_EVENT(feedback_config_event),
                            HANDLE_EVENT(ble_peer_operation_event),
                    HANDLE_EVENT(ble_peer_event)>
{
      public:
        struct Config {
                uint8_t end_effect = 14;
                uint8_t step_effetct = 14;
        };

        FeedbackModule();

        static bool HandleEvents(const app_event_header *aeh);

      private:
        virtual void handle(const module_state_event &event) override;
        virtual void handle(const encoder_event &event) override;
        virtual void handle(const feedback_config_event &event) override;
        virtual void handle(const ble_peer_operation_event &event) override;
        virtual void handle(const ble_peer_event &event) override;
 };
```

Výpis kódu 2 Feedback module

5.1.2.1 Komunikační moduly

Komunikace mezi počítačem a ovladačem zajišťuje několik modulů. A to jeden centrální, který má řeší, jaký protokol použít pro komunikaci, serializaci a deserializaci zpráv a odpovídá na zprávy z aplikace. Pro serializaci a deserializaci zpráv z formátu *Protobuffer* používá knihovnu *nanopb*.

Další dva spravují komunikační protokoly, jeden spravuje *USB HID* a druhý *HID over GATT*. Oba používají stejný *HID descriptor*, který popisuje, co náš ovladač je a jak komunikuje. V návrhu jsme se rozhodli, že komunikace bude probíhat pomocí 64-bajtových paketů. V ukázce kódu 3 lze vidět *HID descriptor*, kde v hlavičce popisujeme, co náš ovladač je, protože nezapadá do žádné předpřipravené kategorie, tak říkáme, že se jedná o generický ovladač a pak následuje kolekce *endpointů*. Definujeme dva *endpointy*, jeden ve směru *OUT* a druhý *IN* se stejným obsahem a to 64 bajtů s hodnotou od 0 do 255.

```
        HID_USAGE_PAGE(HID_USAGE_GEN_DESKTOP),
        HID_USAGE(HID_USAGE_GEN_DESKTOP_UNDEFINED),
        HID_COLLECTION(HID_COLLECTION_APPLICATION),
        // output
        HID_LOGICAL_MIN8(0),
        HID_LOGICAL_MAX8(0xFF),
        HID_REPORT_SIZE(8),
        HID_REPORT_COUNT(64),
        HID_USAGE(HID_USAGE_GEN_DESKTOP_UNDEFINED),
        HID_INPUT(0x02),
        // intput
        HID_LOGICAL_MIN8(0),
        HID_LOGICAL_MAX8(0xFF),
        HID_REPORT_SIZE(8),
        HID_REPORT_COUNT(64),
        HID_USAGE(HID_USAGE_GEN_DESKTOP_UNDEFINED),
        HID_OUTPUT(0x02),
        HID_END_COLLECTION,
```

Výpis kódu 3 USB HID decriptor

Pro komunikaci přes *Bluetooth LE* je nutné se nejdříve připojit k počítači. Toto zajišťují moduly z *CAF*. Přesněji moduly *ble state*, *ble adv* a *ble bond*. Modul *ble bond* byl upraven, protože neumožňoval nastavit vymazání *bond*, pokud uživatel podržel více tlačítek najednou a nevytvářel událost o smazání *bond*. Kvůli chybějící události nemohl *FeedbackModule* informovat uživatele o smazání *bond*.

Pro připojení k počítači je nutné mít správně nastavené informace v advertising paketech. Měly by krom informací o jméně obsahovat také informace o poskytovaných službách. *NRF Connect SDK* poskytuje způsob, jak nastavit informace v advertising paketech. Tyto informace jsou do paketů přidány pomocí *Bluetoothe LE advertising providers*. Některé jsou již součástí *nRF Connect SDK*. Například přidání jména, nastavení flagů a informace, jestli je zařízení v párovacím módu. Ale chybí provider pro nastavení, jaké služby v GATT zařízení podporuje. Tento provider je implementován v ukázce kódu 4. Je to velice jednoduchý kód. Pokud je ovladač v pairing módu, tak se přidá do advertising dat 16-bitové UUID pro HIDS, BAS a DIS a nakonec je tato funkce zaregistrována jako dodavatel advertising dat.

```
#include <zephyr/bluetooth/uuid.h>
#include <bluetooth/adv_prov.h>


static int get_data(struct bt_data *ad,
                const struct bt_le_adv_prov_adv_state *state,
                    struct bt_le_adv_prov_feedback *fb)
{
        ARG_UNUSED(fb);

        if (!state->pairing_mode) {
                return -ENOENT;
        }

        static const uint8_t data[] = {
                BT_UUID_16_ENCODE(BT_UUID_HIDS_VAL),
                BT_UUID_16_ENCODE(BT_UUID_BAS_VAL),
                BT_UUID_16_ENCODE(BT_UUID_DIS_VAL),
        };

        ad->type = BT_DATA_UUID16_ALL;
        ad->data_len = sizeof(data);
        ad->data = data;

        return 0;
}

BT_LE_ADV_PROV_AD_PROVIDER_REGISTER(uuid16_all, get_data);
```

Výpis kódu 4 Bluetooth LE advertising data provider

5.1.2.2 Moduly vstupů

Pro sledování stavu tlačítek je použit modul *Buttons module* a pro detekci, jestli došlo ke krátkému, dlouhému nebo dvojitému stisknutí se využívá modul *Click detector module*. Typ stisknutí je využíván jen lokálně na ovladači pro mazání informací o připojených zařízení.

Další vstupy jsou dva enkodery, které jsou ovládány otočnými prstenci. Tyto enkodery kontroluje jeden modul, a pokud zjistí, že se jejich hodnota změnila, tak vytvoří událost.

Poslední vstup je přepínač, který je kontrolován vlastním modulem. Tento modul čte stav hallova senzoru, který snímá pozici magnetu v přepínači.

5.1.2.3 Modul zpětné vazby

Tento modul má na starosti komunikaci ovladače s uživatelem pomocí vibračního motoru a diody. To znamená informovat ho o stavu připojení, stavu baterie a reagování na změnu hodnoty vnitřního encoderu pomocí vibračního motoru. Případně pouštění efektů na vibračním motoru na povel od aplikace.

Také ovládá napájení napájení diody a vibračního motoru, takže může při delší neaktivitě ovladače toto napájení vypnout pro úsporu energie v baterii.

5.1.2.4 Sledování stavu baterie

Stav baterie sleduje čip MAX17260 od společnosti Analog Devices. Tento čip pomocí *Maxim ModelGauge™ m5* algoritmu monitoruje stav baterie. Tento algoritmus kombinuje počítač coulombů a měření napětí pro co nejpřesnější měření baterie. Krom aktuálního nabití baterie poskytuje také měření napětí a proudu a také poskytuje odhady doby do vybití či nabití baterie.

Zephyr RTOS podporuje čip MAX17262. Tento čip komunikuje stejně jako MAX17260 přes I2C sběrnici. Rozdílné jsou jen v pouzdrech, ve kterých jsou dodávané a v tom, že MAX17260 potřebuje vnější měřící rezistor.

Takže pro měření sledování stavu baterie je použit ovladač již přítomný v *Zephyr RTOS* v kombinaci s modulem *Sensor Manager module* z CAF. Tento modul je nastaven, tak aby každým 5 minut přečetl stav baterie a vytvořil událost. Tuto událost pak zpracují komunikační moduly a předají tuto informaci aplikaci a BAS.

5.2 Zprávy mezi ovladačem a aplikací

Jak bylo nastíněno v návrhu, pro komunikaci mezi ovladačem a aplikaci je použit protokol *Protobuffer*. Byla vytvořena definice zpráv, protože zprávy tohoto protokolu nejsou sebe popisující.

Protobuf podporují *tagged union* pomocí klíčového slova *oneof*[42]. Tato vlastnost umožňuje mít dvě hlavní zprávy *FromSwivel* a *ToSwivel*, které v sobě obsahují *oneof*. Toto nerozbije požadavek na zpětnou kompatibilitu, protože *Protobuf* jsou schopné zprávu dekódovat, pokud obsahuje položky navíc. Je pouze nutné udržovat stejné číslování zpráv v jejich definici.

Ve zprávě *ToSwivel* se může nacházet požadavek na informace o ovladači, stavu ovladače, nastavení ovladače nebo požadavek na to přehrát efekt na vibračním motoru. Na požadavky reaguje ovladač svou zprávou *ToSwivel*. Varianta s informacem i o ovladači obsahuje verzi firmwaru a revizi hardwaru a varianta se stavem obsahuje informace o úrovni nabití baterie a o pozici přepínače. *FromSwivel* dále obsahuje varianty informující o změně stavu vstupů nebo o poklesu úrovně nabití baterie.

Protobuf nemá zabudovaný způsob, jak poznat, zda aplikace již přečetla celou zprávu. Proto knihovna *nanopb* nebo knihovna *protobuf*, kterou používáme v aplikaci podporují *length delimited encoding*. Tento způsob zpracování zpráv nejprve napíše velikost zprávy zakódovanou metodou *variant* a poté zapíše požadovanou zprávu. Teoreticky bychom mohli posílat menší zprávy přes *USB HID*, ale *Windows* s *MacOs* očekávají 64-bajtovou zprávu a menší nepřijmou. Proto je využit způsob *length delimited encoding* pro posílání zpráv.

5.3 Aplikace

Podle výběru v návrhu je aplikace implementována pomocí frameworku *Tauri*. Tento framework využívá programovací jazyk Rust pro vývoj backendu a pro vývoj uživatelského prostředí umožňuje použít webové technologie.

Pro implementaci funkcionality je využita knihovna *Actix*. Actix je *Actor framework*. Každá komponenta, která byla specifikována v návrh je v aplikaci reprezentovaná jako *Actor*. Toto nám umožňuje oddělit odpovědnost jednotlivých komponent, protože *Actor* může komunikovat s ostatními *actor* pouze pomocí zpráv. Na rozdíl od CAF, který je využit ve firmware a podporuje pouze poslání zprávy modulům, které jsou přihlášeny k jejimu přijeti, tak *Actix* posílá zprávu jednomu specifickému *actor* a ten na ní může odpovědět. Posílání jedné zprávy více *actor* je možné implementovat pomocí *Recipient*, což je struktura, která umí odeslat určitý typ zprávy jednomu příjemci.

5.3.1 Komunikace s Adobe Lightroom

O navázání a udržování komunikace s Adobe Lightroom se stará samostatný *actor*. Komunikace s Adobe Lightroom probíhá pomocí soketů, ale oproti normálním soketům, které jsou schopny komunikovat obousměrně, sokety v Adobe Lightroom jsou schopny komunikovat pouze jednosměrně. Takže je potřeba vytvořit dvě spojení. A je nutné obě dvě spojení iniciovat, protože Adobe Lightroom pouze poslouchá na dvou portech.

Actix podporuje přidání streamu dat jako zdroj zpráv pro *actor*. Takže lze zprávu z Adobe Lightroom zpracovat stejně jako zprávu od jiného *actor*. Pro oddělení vyhodnocování zpráv a jejich parsování použijeme *FramedRead* stream. Tento stream dekóduje přijaté bajty do objektů a tyto objekty jsou pak zpracovány.

Pro odesílání zpráv do Adobe Lightroom je využit stream *FramedWrite*. Ten funguje na opačném principu, z objektů dělá stream bytů, který je následně odeslán do Adobe Lightroom.

Jak již bylo zmíněno v sekci 4.1, tak zásuvný modul je napsaný v jazyce Lua s rozšířením od Adobe pro ovládání Adobe Lightroom Classic. Toto rozšíření přidává funkci *pcall*, která vyhodnocuje do ní předaný řetězec jako kód. Díky této funkci můžeme poslat zásuvnému modulu volání funkcí a on je vyhodnotí. Například změna hodnoty kontrastu o 1 vypadá takto 5. A samozřejmě zásuvný modul podporuje i další operace jako je změna hodnocení, přepínání modulů nebo čtení presetů a sledování změn stavu fotografie.

```
changeByValue("Contrast",1)
```

Výpis kódu 5 Změna hodnoty parametru

5.3.2 Komunikace s ovladačem

Komunikaci s ovladačem zajišťuje další *actor*. Tento *actor*, krom samotné komunikace, také řeší typ stisknutí tlačítka či konfiguraci ovladače.

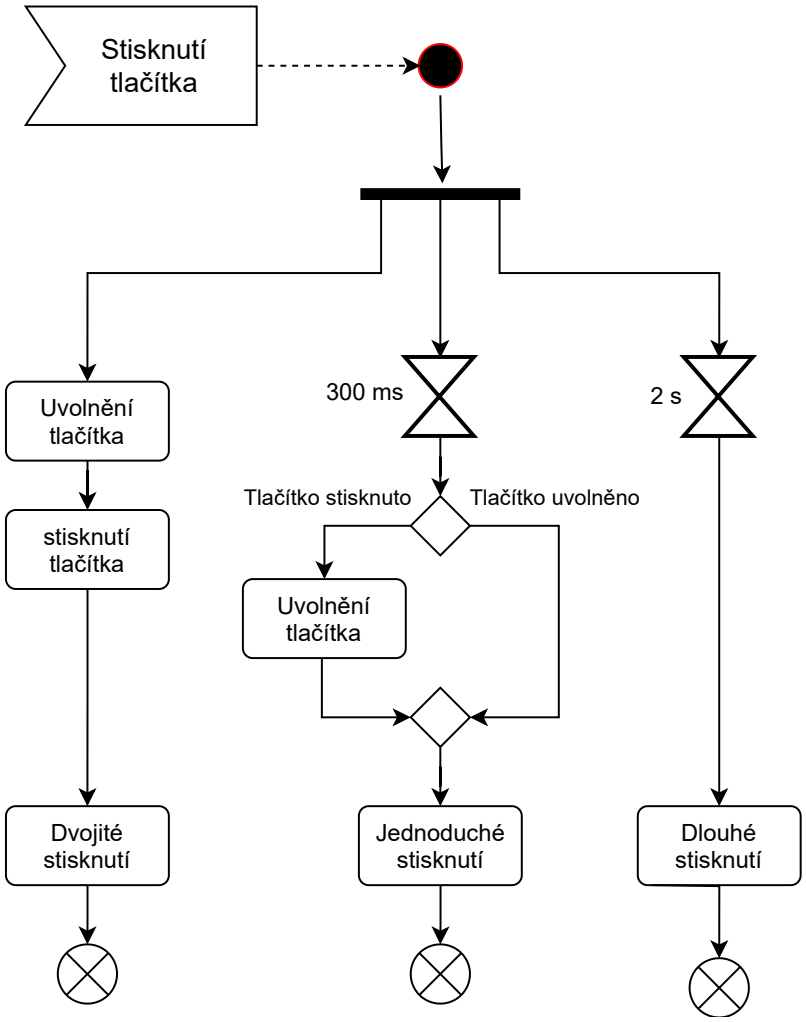
Jak již bylo několikrát zmíněno, ovladač komunikuje pomocí USB HID protokolu. Pro komunikaci pomocí tohoto protokolu je potřeba interagovat s API jednotlivých operačních systémů. Ale existuje jednoduší způsob. Použít knihovnu, která je meziplatformní a umožňuje komunikovat s HID zařízeními přes společné rozhraní.

Pro Rust existuje knihovna *hidapi-rs*. Tato knihovna je abstrakce nad stejnojmennou knihovnou *hidapi*[43]. Což je knihovna napsaná v programovacím jazyku C a umožňuje komunikovat s HID zařízeními[44].

Pro serializaci a deserializaci zpráv z ovladače je použita knihovnu *rust-protobuf*.

5.3.3 Detekce typu stisknutí

Požadavek F9 určuje, že aplikace má detekovat jednoduché a dvojité stisknutí a dlouhé podržení. Mohla by být využita již existující detekce na ovladači, ale protože je jednodušší aktualizovat aplikaci než firmware, tak je detekce implementována na straně aplikace.



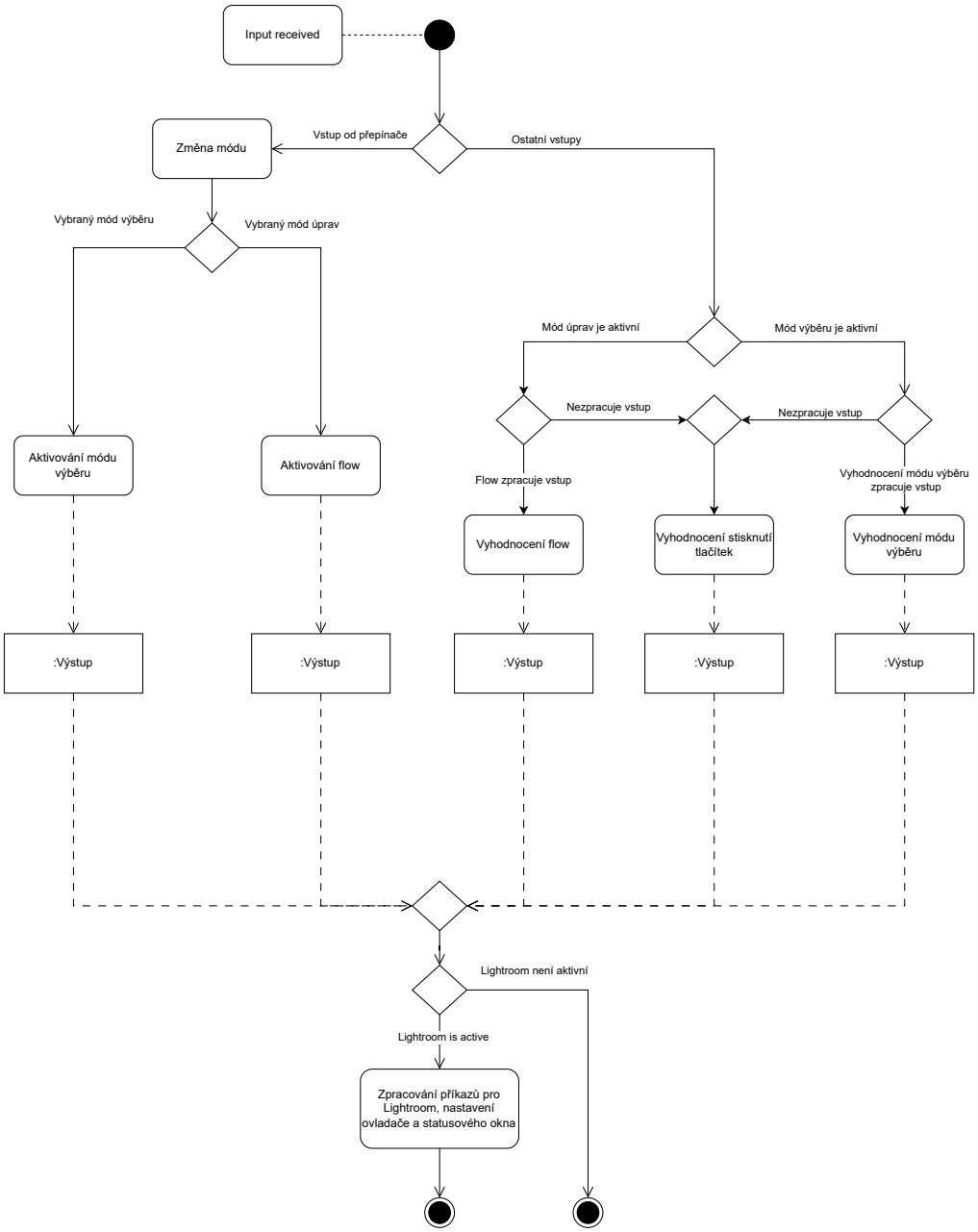
Obrázek 5.1 Vyhodnocení typu stisknutí

Tato detekce se nachází v modulu, který se stará o komunikaci s ovladačem. V diagramu 5.1 je znázorněno, jak detekce stisknutí funguje. Důležitá byla volba intervalu od prvního stisknutí, kdy se další stisknutí bude počítat jako dvojité stisknutí. Pokud by byla moc dlouhá, tak by se často stávalo, že uživatel omylem pošle dvojité stisknutí místo dvou po sobě jdoucích stisknutí a pokud by byla moc krátká, tak by uživatel nebyl schopen provést dvojité stisknutí. Po testování různých intervalů bylo zjištěno, že 300 milisekund je ideální interval.

5.3.4 Vyhodnocení interakcí s ovladačem

Interpretaci uživatelových interakcí s ovladačem má na starosti modul *FlowManager*. Tento modul obsahuje všechna nastavení a stav potřebný pro vyhodnocení interakcí. V kapitole 4.9.1.1 bylo navrženo, jak by se měly interakce s ovladačem vyhodnocovat.

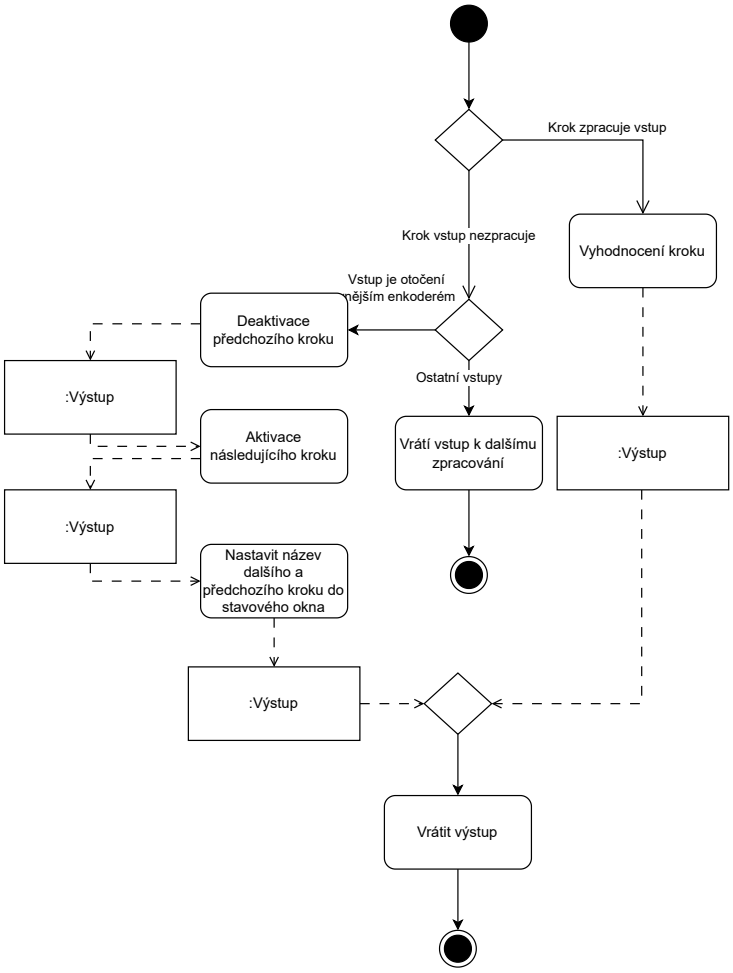
V této kapitole je shrnuto, jak se opravdu vyhodnocuje a interaguje se stavovým oknem a nastavením ovladače. Toto shrnutí je rozděleno na dvě části pro zjednodušení vysvětlení. V první části je zachyceno celé vyhodnocení a v druhé části je popsáno vyhodnocení *flow*.



Obrázek 5.2 Vyhodnocení vstupu

Vyhodnocování funguje na principu, že pokud pro danou funkci není tento vstup validní, tak ho vrátí. Toto reprezentuje enum *ExecutionResult*, jehož dvě hodnoty jsou buď *ProcessInput* nebo *InputProcessed* a reprezentují vstup ke zpracování nebo výsledek zpracování vstupu. Výsledek zpracování může krom příkazů pro Adobe Lightroom obsahovat i nastavení ovladače, stav stavového okna nebo efekt pro ovladač.

První část je zachycena v diagramu 5.2. Prvně se rozhoduje, jaký mód je aktivní. Pokud je to mód výběru, tak vstup prvně dostane funkce na změnu hodnocení, pak funkce na navigaci mezi fotkami a nakonec tlačítka. Pokud je aktivní mód pro úpravu fotek, tak vstup dostane prvně *flow* a pak případně tlačítka. Pro oba módy platí, že pokud ani po tomto nedojde ke zpracování vstupu, tak ho dostane vyhodnocení přepínače módů.



Obrázek 5.3 Vyhodnocení flow

Druhá část je zachycena v diagramu 5.3. Zde se řeší už vyhodnocení samotného *flow*. Ve *flow* prvně dostane vstup samotný krok. Pokud ho nezpracuje a jedná se o pohyb spodním enkodérem, tak dojde ke změně aktivního kroku. Při změně kroku je potřeba nejdříve deaktivovat předchozí krok a následně aktivovat nový krok. Aktivace a deaktivace slouží k nastavení ovladače a aktualizaci stavového okna.

Flow je centrum funkcionality aplikace. V aplikaci je *flow* reprezentováno jako pole kroků. Kroky musejí splňovat *StepTrait*. *StepTrait* je *Trait*, což je soubor funkcí a je to způsob, jak v Rustu implementovat společné chování. Dají se přirovnat k rozhraním v jiných programovacích jazycích.

Kroky krom funkce *execute*, která můžu konzumovat vstup a vrátit výstup, implementují také funkce *activate* a *deactivate*. Tyto funkce slouží k aktivaci a deaktivaci kroků a krom nastavení interního stavu kroků slouží k získání nastavení ovladače a informacích o hodnotě parametru, který upravují. Celou definice *StepTraitBase* lze vidět v ukázce kódu 6.

```
pub trait StepTraitBase {
    fn execute(&mut self, val: InputType, photo: &LrPhoto) -> ExecutionResult;
    fn create_config(&self, photo: &LrPhoto) -> DeviceConfig;
    fn activate(&mut self, photo: &LrPhoto) -> OutputType;
    fn deactivate(&mut self, photo: &LrPhoto) -> OutputType;
}
```

Výpis kódu 6 Step trait

5.3.5 Ukládání dat

V sekci 4.9.1.2 byly řešeny výhody a nevýhody ukládání dat v databázi oproti ukládání v souboru. Hlavní výhoda databáze pro toto použití je podpora migrací. Protože nám umožňují jednoduše při změně tvaru dat převést data do nového tvaru. Tuto výhodu, ale maže knihovna *Serde*. Tato knihovna umí převést data z a do různých formátů a zároveň umožňuje definovat výchozí hodnotu pro nové položky ve struktuře. Takže pokud přidáme novou položku do některé z ukládaných struktur a v programu jí definujeme výchozí hodnotu, tak tuto položku nastaví na výchozí hodnotu a nevadí, že v uložených datech chybí. A při odstranění položky ignoruje položky v uložených datech navíc.

Ukládaná data jsou také velmi jednoduchá. V databázi by se jednalo o dvě tabulky bez vazeb mezi sebou. Takže by použití lokální relační databáze by přidávalo zbytečnou složitost.

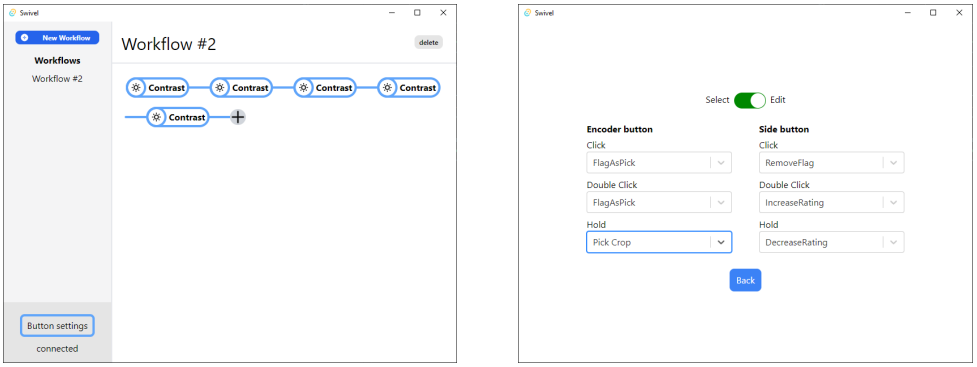
Proto je využit pro ukládání textový formát JSON. Další výhodou je, že tuto knihovnu již využívá *Tauri* pro kódování zpráv mezi backendem a uživatelským rozhraním.

Ukládání dat spravuje další *actor*, tento *actor* při přijetí zprávy o změně nastavení odešle všem přihlášeným *actor* informaci o změně.

5.4 Uživatelské rozhraní

Uživatelské prostředí je vytvořeno pomocí JavaScriptové knihovny React. Je použita, protože umožňuje rozdělit uživatelské rozhraní na jednotlivé komponenty a také má velké množství kompatibilních knihoven. Uživatelské rozhraní je vlastně webová aplikace.

Uživatelské rozhraní lze rozdělit na dvě části. Jedna slouží pro vytváření a úpravu *flow* a druhá slouží pro nastavení funkcí tlačítek. Obě části lze vidět v 5.4.



(a) Úprava flow **(b)** Nastavení tlačítek

Obrázek 5.4 Uživatelské rozhraní

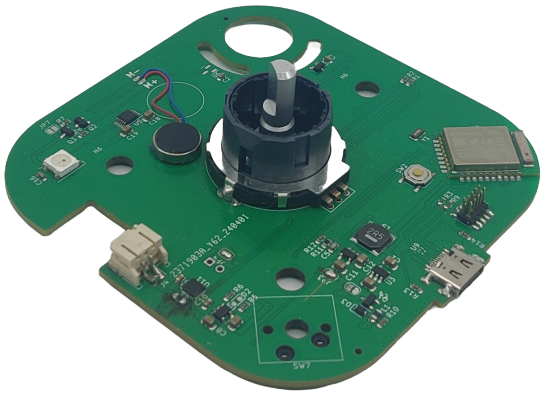
Krom hlavního okna aplikace poskytuje ještě stavové okno. Toto okno je vždy zobrazeno nad ostatními okny a ukazuje aktuální stav *flow* a hodnotu upravovaného parametru.

5.4.1 Komunikace s backendem

Pro zobrazení flow a dalších nastavení je nutné získat data z backendu aplikace. *Tauri* poskytuje funkci *invoke* pomocí, které umožňuje volat vybrané funkce v backendu. Tato funkce je asynchronní a její volání tedy není blokující. Pro oddělení kódu, který má na starosti získávání a aktualizaci nastavení a *flow* od kódu pro jejich zobrazení je použita knihovna *Nanostores* a její integrace do Reactu.

5.5 Ovladač

Tištěný spoj ovladače byl navržen pomocí programu KiCAD. Což je open-source nástroj pro tvorbu schémat a tištěných spojů. Byly vytvořeny dvě verze desek. Druhá verze řeší pár problémů nalezených při práci s první verzí. Druhou verzi lze vidět na obrázku 5.5.



Obrázek 5.5 Druhá verze desky ovladače

5.5.1 První verze

První verze ovladače je založená na nRF52833 v modulu MK07A od firmy MOKOSmart. Schéma zapojení lze vidět v příloze A . Na schématu lze vidět dva způsoby změny napětí z 5V na 3.3V. Jeden využívá LDO neboli *Low Dropout Regulator*, tento regulátor snižuje napětí tak, že přebytečnou energii přemění na teplo. Druhý způsob využívá spínaný regulátor, který je efektivnější než *LDO* v převodu napětí. V první verzi je použit spínaný regulátor AP61100 od firmy Diodes Incorporated a LDO TLV5733. V průběhu testování bylo zjištěno, že AP61100 nezvládá převést minimální napětí baterie 3.7 V na potřebných 3.3 V.

Další problém nastal při manuálním osazováním desky. Kvůli kontaktům na spodní straně modulu MK07A se podařilo úspěšně osadit pouze jednu desku.

Dále bylo zjištěno, že i přestože programovatelná dioda WS2812B využívá ke komunikaci 5 voltovou logiku, tak s ní lze komunikovat bez použití převodníků z 3,3 voltové logiky.

5.5.2 Druhá verze

Schéma druhé verze lze nalézt v příloze B. Oproti první verzi využívá nRF52840 v modulu E73--2G4M08S1C od firmy EBYTE, protože je společnost JLCPCB má v nabídce součástek, které lze využít pro strojové osazení. NRF52840 je čip vyšší řady než v první verzi a využívá stejné programovací prostředí jako původní čip.

Dále byl vyměněn spínací regulátor AP61100 za AP3428 a dioda WS2812B byla připojena přímo na nRF52840.

A pro osazování byla využita služba automatického osazení desek od společnosti JLCPCB.

Kapitola 6
Testování

6.1 Automatické testování

Aplikace je testována jednotkovými, které ověřují základní funkcionalitu aplikace. Tyto testy lze využít ke kontrole, jestli změny nerozbili nějakou funkcionalitu aplikace. Tyto testy lze spouštět automaticky při nahrání nové verze do verzovacího systému Git a zajistit, že v repozitáři bude vždy funkční verze aplikace.

Testy lze spustit pomocí příkazu *cargo test*.

6.2 Uživatelské testování

Výsledek práce, ovladač a aplikace, byly podrobeny uživatelskému testování. Testování probíhalo podle scénáře v příloze C. Tímto scénářem byli uživatelé proveden moderátorem, který se jich na začátku zeptal na jejich zkušenosti s focením a používání Adobe Lightroom. V průběhu měl uživatel k dispozici uživatelský manuál k aplikaci a ovladači (příloha C). Po projití scénáře se moderátor zeptal uživatele na otázky, které jsou uvedeny na konci scénáře.

Uživatele lze rozdělit do dvou skupin podle toho, jestli fotografování je jejich primární zdroj obživy nebo jestli je to jen doplněk. Do první skupiny spadají dva uživatelé a do druhé skupiny 4 uživatelé. Většina uživatelů hodnotí své zkušenosti s Adobe Lightroom tak, že jsou schopni Lightroom používat k úpravám a tříděním fotografií, ale nejsou experty v jeho využití.

Všichni uživatelé hodnotí celkový dojem z aplikace a ovladače pozitivně a jsou si schopni představit, že by ho používali běžně pro úpravu fotografií. Zde jsou přiložena hodnocení od vybraných uživatelů.

- Úpravu fotek to dělá příjemnější. Můžu se více soustředit na fotku místo posuvníků v Adobe Lightroom. Haptická odezva od ovladače je super, ale někdy hodnoty hodně skáčou.
- Manuál je místy lehce zmatený, ale po rozkoukání v aplikaci jsem se orientovala, co a jak dělat. A samotná úprava mi přišla příjemná a výborné bylo, že jsem nemusela hledat jednotlivé parametry a mohla se více se soustředit na fotku.

Ze sledování uživatelů, jejich reakcí a odpovědí byla identifikována místa, kde by se naše aplikace dala vylepšit pro jednodušší a intuitivnější ovládání. Hlavně se jednalo o změnu barev a velikostí některých prvků, aby byly jednoduše použitelnější, také automatické vybrání textových polí při otevření nabídek nebo odstranění zelené barvy z přepínače ve výběru funkcí pro tlačítka.

Zpětná vazba na ovladač se především týkala jeho zpětné vazby. Například to, že vibrace, když se uživatel pokusí upravit parametr mimo jeho rozsah, jsou moc dlouhé a nečekané. Nebo

že při dlouhém podržení tlačítka by uživatelé čekali, že akce se stane v moment, kdy uběhne doba, po kterou ho mají držet, a ne až v moment, kdy ho pustí. Tyto problémy byly opraveny. Dále si někteří uživatelé stěžovali na pomalou odezvu, tento problém se projevoval pouze někdy a nepovedlo se vypozorovat, co ho způsobuje. Připojení a odpojení zařízení tento problém vyřešilo.

Cenu, kterou by byli uživatelé ochotni zaplatit za ovladač, byla v rozmezí od 2000 Kč do 6000 Kč.

Řešení nalezených problémů a optimalizace uživatelského prostředí bude probíhat i dále po skončení této práce.

6.3 Měření spotřeby elektrické energie

V této sekci je popsáno měření spotřeby elektrické energie ovladačem. A to druhé verze ovladače s finální verzí firmwaru při použití spínaného zdroje.

Spotřebovaná elektrická energie byla dopočítána z měření poklesu napětí přes rezistor s odporem 50 ohmů a napětí baterie. Toto měření bylo provedeno pomocí osciloskopu a hodnoty napětí byly odečítány každé 4 milisekundy po dobu přibližně 23 vteřin. V průběhu měření ovladač posílal každých 500 milisekund zprávu počítači o pohybu enkodérů. Výsledky měření lze vidět v tabulce 6.1, kde jsou hodnoty zprůměrované po 1 vteřině. Z měření vyplývá že ovladač má přibližnou spotřebu 11,6 mW při komunikaci bez použití vibračního motoru.

Vteřiny	**napětí[V]**	**Pokles napětí přes rezistor[mV]**	**proud[mA]**	**výkon[mW]**
1	3,97	148,03	2,96	11,76
2	3,97	146,40	2,93	11,62
3	3,97	149,02	2,98	11,84
4	3,98	145,34	2,91	11,56
5	3,98	143,65	2,87	11,42
6	3,97	145,63	2,91	11,56
7	3,97	143,46	2,87	11,40
8	3,97	146,27	2,93	11,61
9	3,98	144,51	2,89	11,49
10	3,98	142,94	2,86	11,37
11	3,97	149,18	2,98	11,84
12	3,97	141,63	2,83	11,26
13	3,97	144,99	2,90	11,52
14	3,98	145,86	2,92	11,60
15	3,98	143,39	2,87	11,41
16	3,98	148,51	2,97	11,82
17	3,97	146,02	2,92	11,60
18	3,98	147,42	2,95	11,72
19	3,97	147,30	2,95	11,70
20	3,97	144,70	2,89	11,50
21	3,97	149,18	2,98	11,84
22	3,97	145,47	2,91	11,55
23	3,97	147,81	2,96	11,73
Pruměrná hodnota	**3,97**	**145,95**	**2,92**	**11,60**

Tabulka 6.1 Výsledek měření spotřeby elektrické energie

Závěr

Cílem této práce bylo navrhnout a vytvořit řešení pro ovládání aplikace Adobe Lightroom pomocí hardwarového ovladače. Toto řešení se skládá ze dvou hlavních částí. První část je aplikace, která komunikuje s ovladačem a s Adobe Lightroom. Druhá část je samotný hardwarový ovladač, který komunikuje s aplikací přes Bluetooth nebo USB.

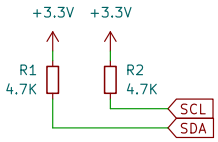
Cíle práce byly splněny v plném rozsahu zadání. Řešení umožňuje ovládat Adobe Lightroom pomocí hardwarového ovladače. Uživatel si může jednoduše nastavit chování ovladače.

Další rozvoj tohoto řešení se může ubírat dvěma směry. První směr je rozšíření aplikace přidáním podpory pro Adobe Photoshop či další grafické aplikace. Druhý směr je vylepšení hardwaru ovladače. Například výměnnou vnitřního otočného prstence za BLDC motor pro lepší haptickou zpětnou vazbu.

Příloha A

Schéma první verze desky

I2C pull up resistors



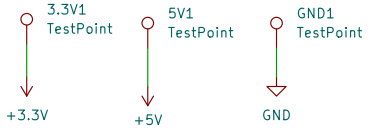
Inputs

File: Hw.kicad_sch

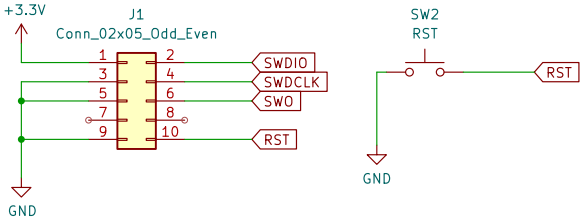
Power

File: Power.kicad_sch

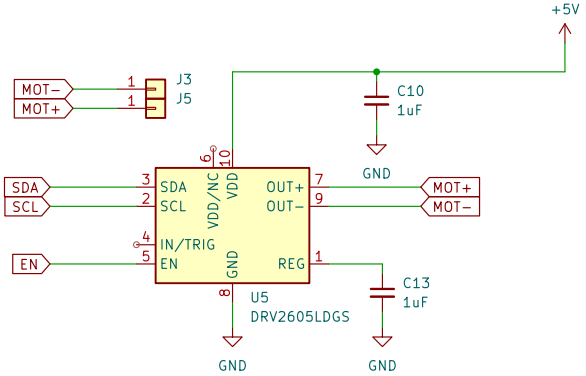
Test points



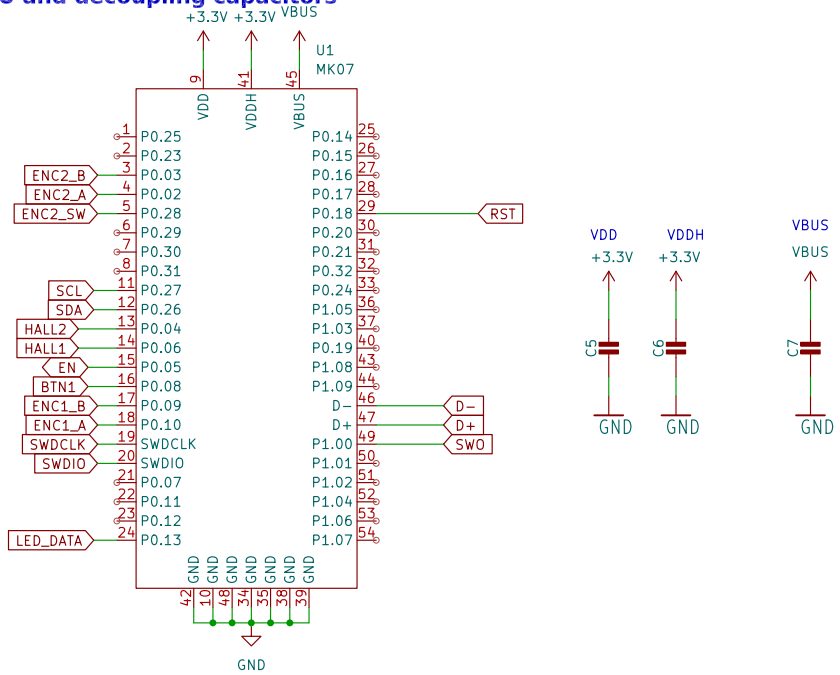
Debug connector and reset button



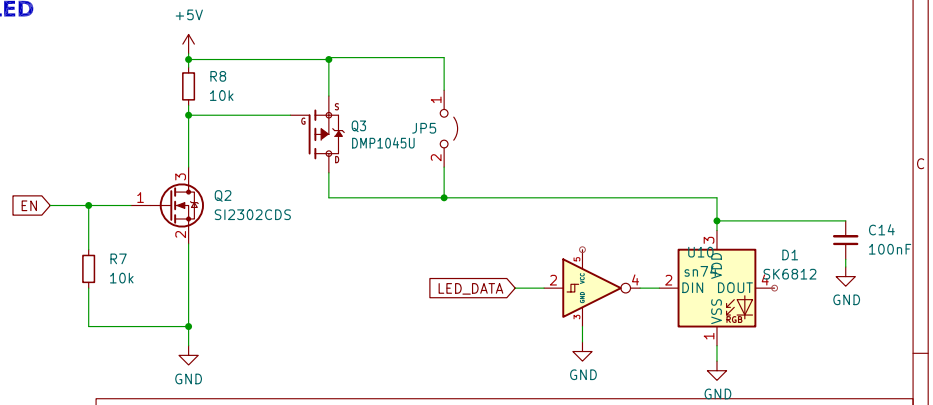
Vibration Motor Driver



MCU and decoupling capacitors



LED

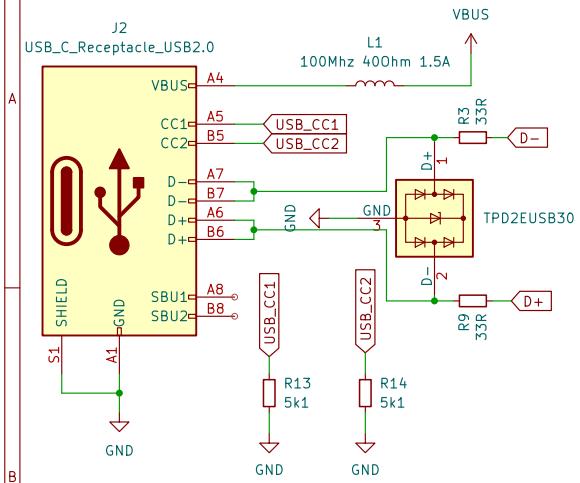


Sheet: /
File: EncoderPcb.kicad_sch

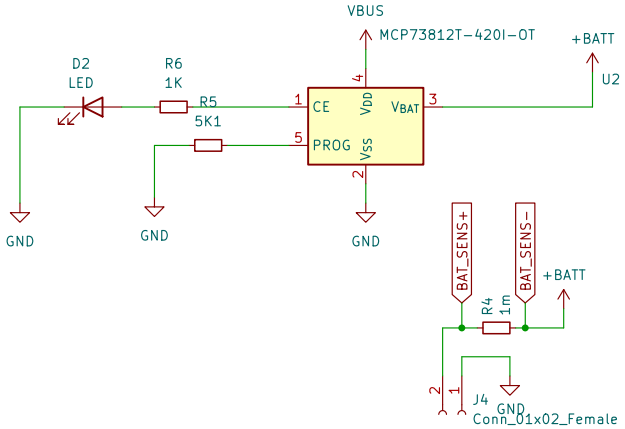
Title:

Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.1		Id: 1/3

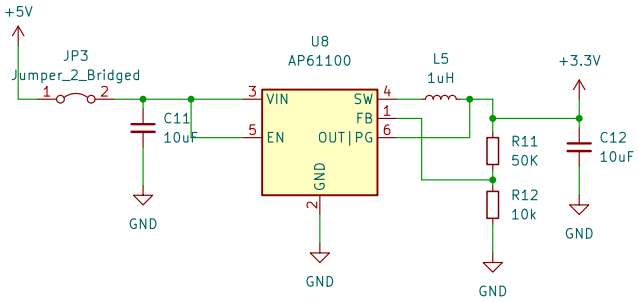
USB connector



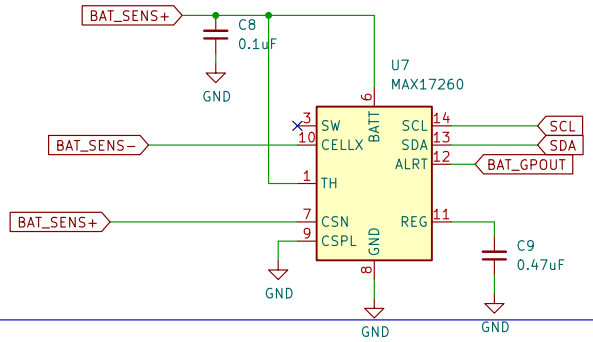
Battery charger



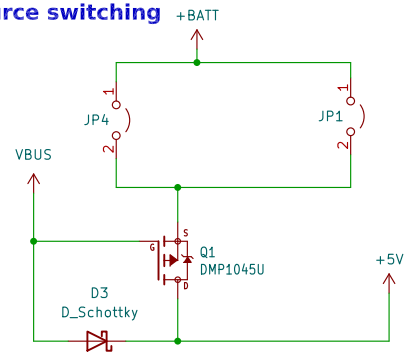
5V to 3.3V BUCK



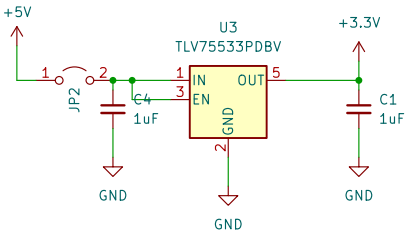
Fuel Gauge



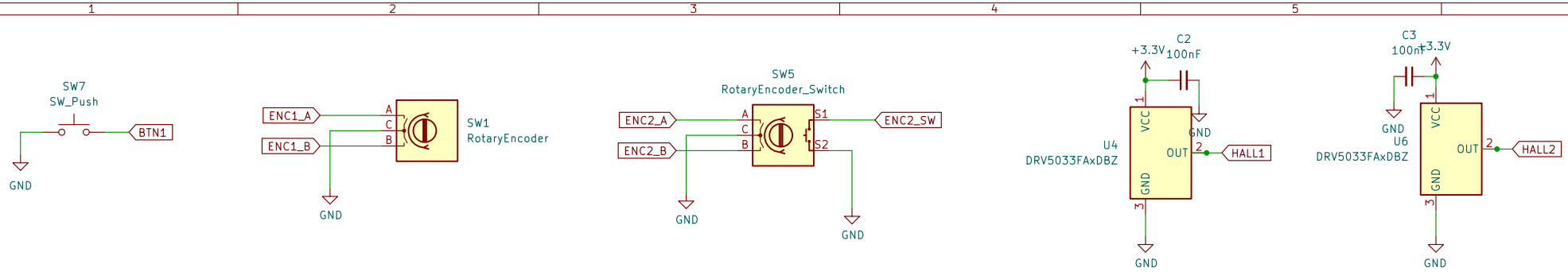
Power source switching



5V to 3.3V LDO



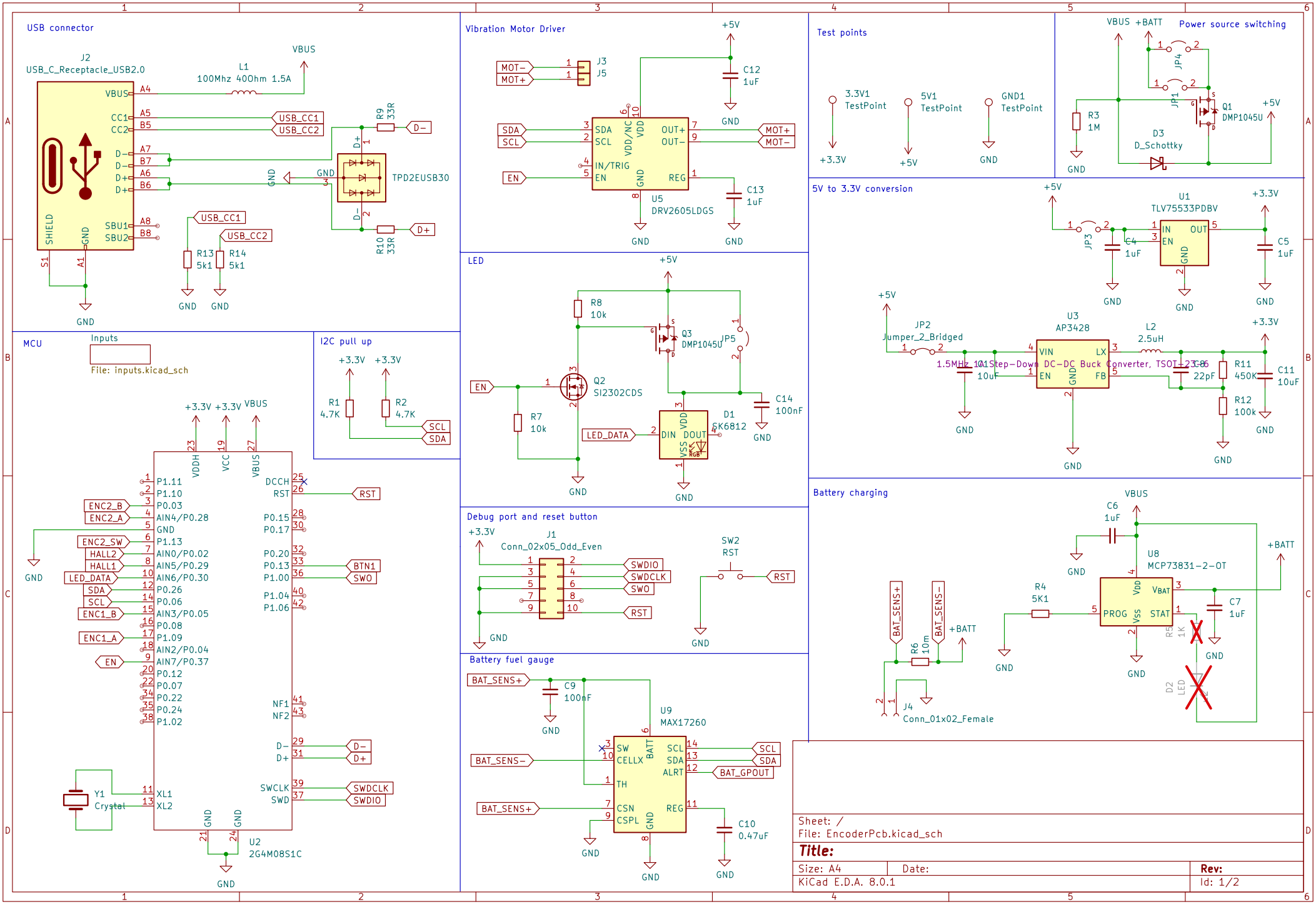
Sheet: /Power/		
File: Power.kicad_sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.1		Id: 3/3



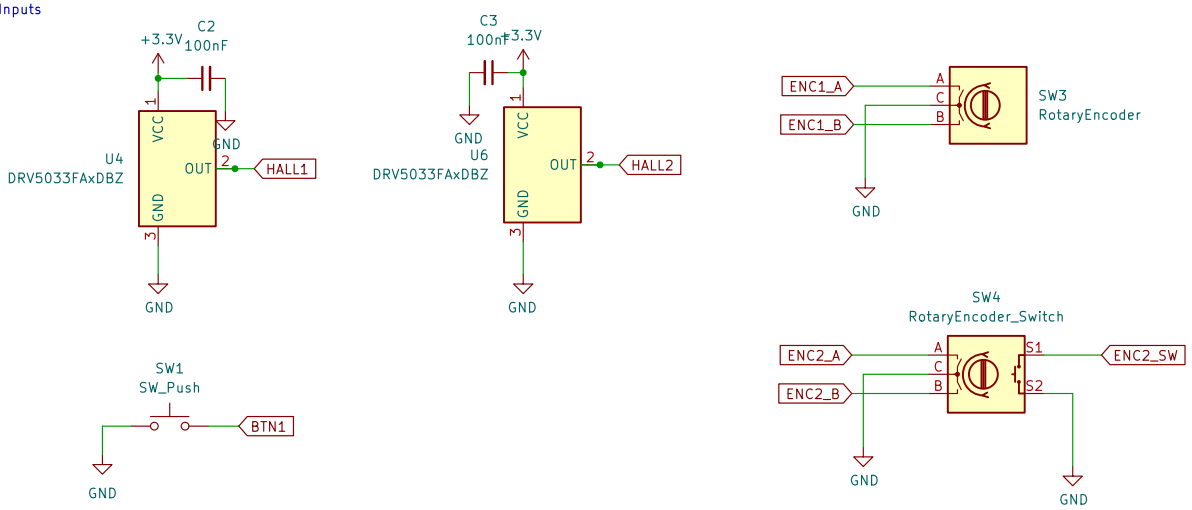
Sheet: /Inputs/		
File: Hw.kicad_sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.1		Id: 5/3

Příloha B

Schéma druhé verze desky


USB connector
J2
USB_C_Receptacle_USB2.0
L1
100Mhz 400hm 1.5A
VBUS
CC1
CC2
D−
D+
SHIELD
GND
SBU1
SBU2
S1
A1
USB_CC1
USB_CC2
R9 33R
R10 33R
R13 5k1
R14 5k1
TPD2EUSB30
Vibration Motor Driver
MOT−
MOT+
J3
J5
C12 1uF
SDA
SCL
IN/TRIG
EN
VDD/NC
OUT+
OUT−
REG
C13 1uF
U5
DRV2605LDGS
Test points
3.3V1 TestPoint
5V1 TestPoint
GND1 TestPoint
Power source switching
+BATT
JP4
JP1
R3 1M
D3
D_Schottky
Q1
DMP1045U
5V to 3.3V conversion
U1
TLV75533PDBV
JP3
C4 1uF
C5 1uF
JP2
Jumper_2_Bridged
U3
AP3428
L2
2.5uH
1.5MHz 1A Step−Down DC−DC Buck Converter, TSOT−23−6
C10 22pF
R11 450K
R12 100k
C11 10uF
LED
R8 10k
Q3
DMP1045U
JP5
Q2
SI2302CDS
R7 10k
LED_DATA
D1
SK6812
C14 100nF
MCU
Inputs
File: inputs.kicad_sch
I2C pull up
R1 4.7K
R2 4.7K
SCL
SDA
VDDH
VCC
P1.11
P1.10
P0.03
AIN4/P0.28
P1.13
AIN0/P0.02
AIN5/P0.29
AIN6/P0.30
P0.26
P0.06
AIN3/P0.05
P0.08
P1.09
AIN2/P0.04
AIN7/P0.37
P0.12
P0.07
P0.22
P0.24
P1.02
DCCH
RST
P0.15
P0.17
P0.20
P0.13
P1.00
P1.04
P1.06
NF1
NF2
D−
D+
SWCLK
SWD
XL1
XL2
Y1
Crystal
U2
2G4M08S1C
ENC2_B
ENC2_A
ENC2_SW
HALL2
HALL1
LED_DATA
SDA
SCL
ENC1_B
ENC1_A
EN
BTN1
SWO
SWDIO
SWDCLK
Debug port and reset button
J1
Conn_02x05_Odd_Even
SW2
RST
Battery fuel gauge
BAT_SENS+
BAT_SENS−
C9 100nF
U9
MAX17260
SW
CELLX
TH
CSN
CSPL
BATT
ALRT
BAT_GPOUT
C10 0.47uF
Battery charging
C6 1uF
U8
MCP73831−2−OT
R4 5K1
R5 1K
R6 10m
C7 1uF
J4
Conn_01x02_Female
D2
LED
Sheet: /
File: EncoderPcb.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.1
Id: 1/2

Inputs



Sheet: /Inputs/		
File: inputs.kicad_sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.1		Id: 2/2

Příloha C

Uživatelská příručka k aplikaci a scénář testování aplikace

Uživatelská příručka pro aplikaci Swivel

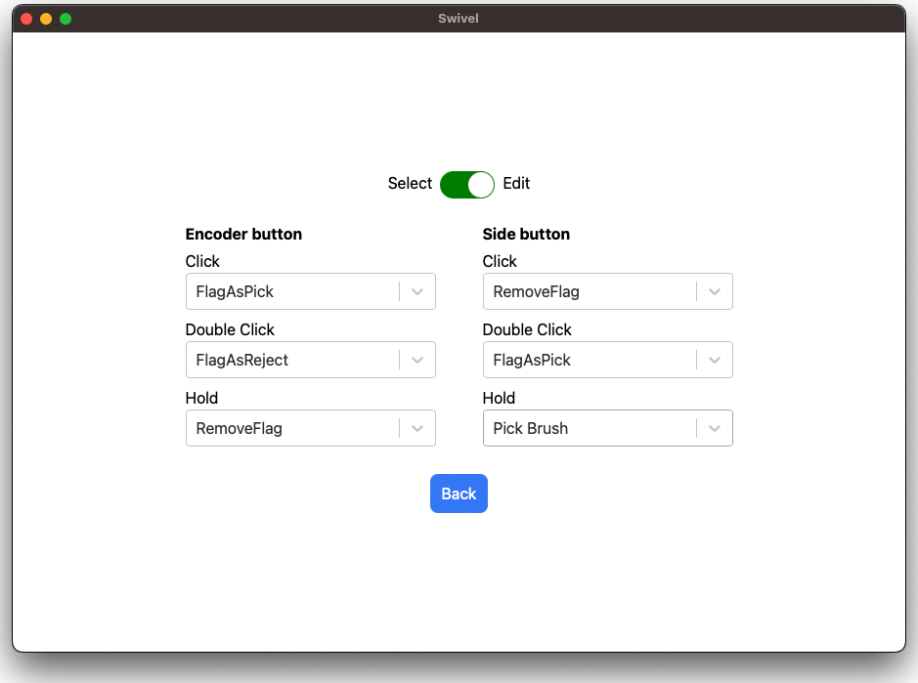
Tato aplikace má dva módy, jeden pro úpravu fotek a druhý pro výber fotek. Mód je aktivován podle aktivního modulu v Adobe Lightroom. Pro přepnutí mezi módy lze využít přepínač na ovladači.

V modulu `Library` je aktivován mód pro výběr fotek. V tomto módu slouží spodní otočný prstenec pro přesun mezi fotkami a horní prstenec pro změnu počtu hvězdiček. A funkce tlačítek lze nastavi v Nastavení tlačítek

V modulu `Develop` je aktivován mód pro úpravu fotek. V tomto módu je aktivní Flow. A funkce tlačítek lze nastavit v Nastavení tlačítek.

Nastavení tlačítek

Tlačítka lze nastavit pro každý mód zvlášť. Mezi módy se přepíná pomocí přepínače nad nastavením tlačítek.



Každé tlačítko může vykonat funkci podle typu stisknutí. Aktuálně jsou podporovány tři typy stisknutí: krátké, dlouhé a dvojité.

Flow

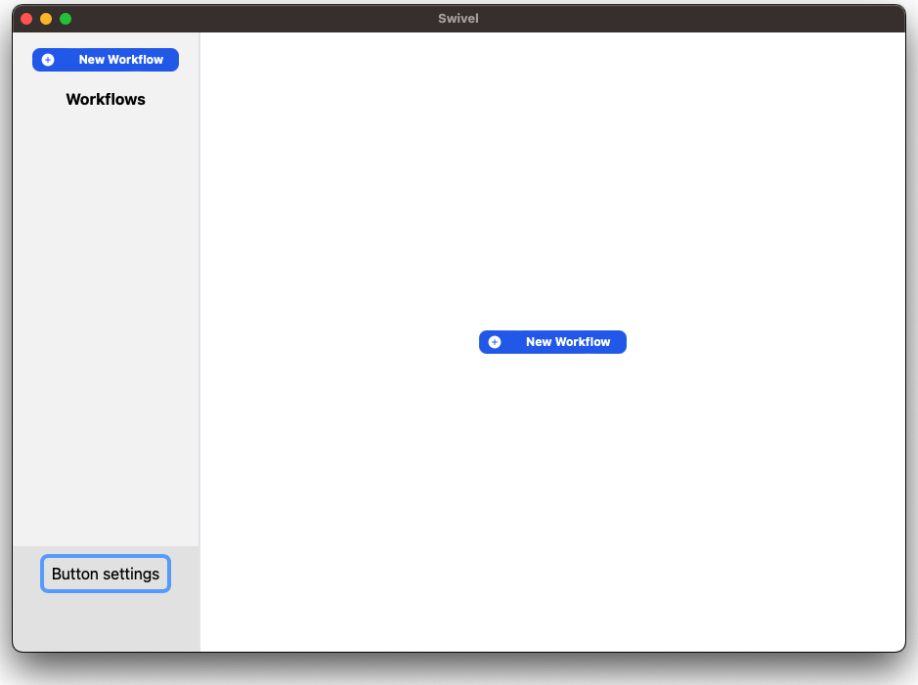
Flow je základ módu úprav. Definuje jaké parametry můžete upravovat. Flow se skládá z kroků. Každý

krok má nějakou funkcionalitu. Může to být úprava jednoho parametru fotky, nebo aplikace presetu.

Pro přesun mezi jednotlivými kroky použijte vnější otočný prstenec a pro jejich aplikaci vnitřní otočný prstenec.

Vytváření nového flow

Pro vytvoření nového flow klikněte na tlačítko `New Workflow` v levem horním rohu. Nebo uprostřed pokud nemáte žádný flow.



Vytváření a úprava kroků

Po vytvoření nového flow mát před sebou prázdný flow. Klikněte na tlačítko se znakem `+`. Toto přidá krok, tento krok můžete změnit v nabídce, která se vám otevře po kliknutím levým tlačítkem myši na krok. Můžete si vybrat mezi kroky:

Parameter

Umožňuje upravit jeden parametr fotky. Například jas, kontrast, saturace, atd.

Preset

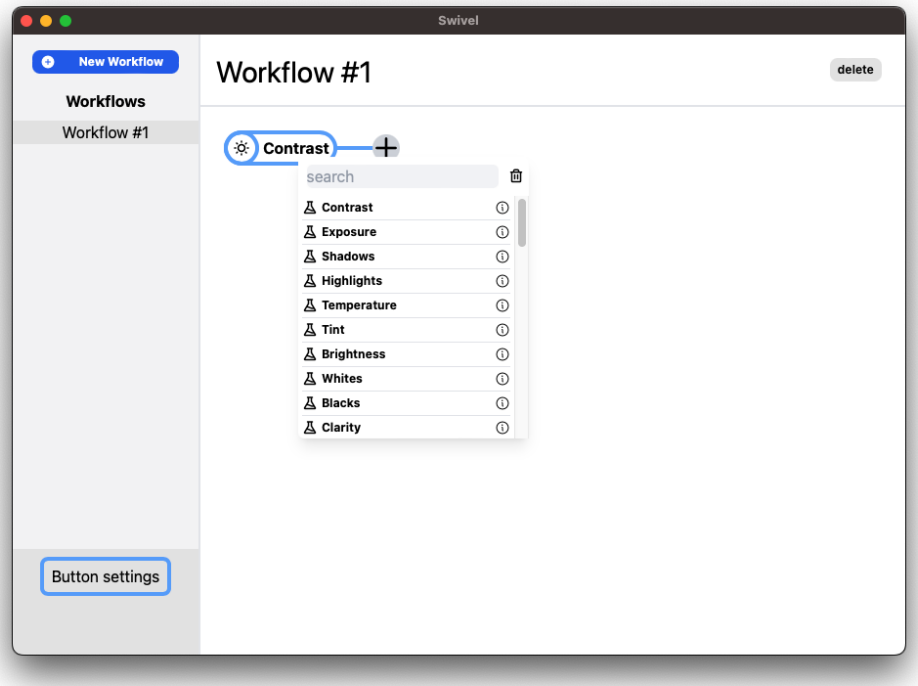
Aplikuje vybraný preset na fotku. Pro výběr presetů ze kterých chete vybírat v průběhu úpravy klikněte na krok pravým tlačítkem a vyberte kroky. Pro toto je potřeba mít zaplý Lightroom s nainstalovaným pluginem.

Ratings

Umožnuje měnit hodnocení fotky.

Navigation

Umožňuje se přesouvat mezi fotkami.



Stavové okno

Aplikace má stavové okno, které ukazuje při úpravě fotek některé informace. Například jaký krok je aktuálně vybrán a jeho hodnotu.

Pro otevření tohoto okna najdětě ikonu aplikace v liště apliakcí a otevřete kontextové menu. Zde vyberte `Show status window`.

Scénář testování aplikace Swivel App a Adobe Lightroom

úvod

- Já jsem [moderátor] a jsem student FITu a vytvářím ovladač a aplikaci swivel pro Lightroom. A potřebuji otestovat funkčnost aplikace a ovladače. * Chtěl bych s Vámi dneska projít aplikaci a ovladač a zjistit Vaší zpětnou vazbu na použtelnost aplikace a ovladače. Velmi mi to pomůže s vychytáním chyb a zlepšením uživatelského zážitku.
- Testování by mělo trvat maximálně 30 minut.
- Nahlas prosím komentujte, kde si myslíte že jsem, co vidíte na jednotlivých stránkách a jak rozumíte tomu co vidíte. * Svoje kroky (interakce) zdůvodňujte.
- Neexistuje žádná správna ani špatná odpověd. Netestuji Vás, ale aplikaci a ovladač.
- Mohu Vás zastavit a zeptat se na něco konkrétního či Vás nasměrovat, pokud odbočíte ze scénáře.

Uvodní otázky a situace

Otázky

- Jak často fotíte?
- Živíte se focením?
- Jak schopný jste s Ligtroomem?
- Jak vypadá Váš normální setup při úpravě fotek (počítač, monitory, myš, klávestnice, tablet)?

Úvod do situace

Vrátil jste se z výletu, kde jste fotil a nyní chcete upravit fotky. Zatím jste si přetáhl fotky do počítače a připojil ovladač k počítači pomocí Bluetooth a zapl aplikaci.

A teď se chystáte upravit fotky v Lightroomu, ale nejdří si musíte nastavit co chcete dělat pomocí ovladače.

Scénář

1. Vytvořte si flow a nastavte funkce tlačítek, tak aby jste byli schopnto provést tyto akce.
    - Při výběru fotek označit fotku vlaječkou nebo jí odmítnout a nastavit hodnocení.
    - Při úpravě fotek:
        - Aplikovat na fotku preset

        - Resetovat úpravu fotky
        - Nastavit tyto parametry na fotce pokud je potřebujete zopakovat řekněte si:
            - Exposure
            - Contrast
            - Highlights
            - Shadows
            - Whites
            - Blacks
2. Otevřete si Lightroom a otevřete si stavové okno aplikace.
3. Ohodnoťe prvních deset fotek, případně je označte vlaječky.
4. Na vámi vybranou fotku aplikujte preset a doupravte fotku pomocí vyrbaných kroků.
5. Vytvořte si nové flow podle vašich preferenci
6. Upravte další fotku.
7. Odstraňte ovladač ze seznamu spárovaných zařízení.
8. Odstraňte párování i z ovladače.

Nápomocné otázky

Při zmatení

- Kde byste tuto funkcionalitu hledal.
- Co byste očekával, že se má stát.

Negativné reakce na funkcionalitu

- Je to v pořádku. Pověz nám co konkrétně na Vás působí negativně.

Závěrečné otázky

- Jak na Vás působila aplikace a jaký z toho máte pocit?
- Jak jste se v aplikaci orientoval?
- Jak na Vás působila zpětná vazba od ovladače?
- Jak Vám vyhovuje tento přístup k používání.
- Myslíte že by Vám toto mohlo ušetřit čas úpravy fotografií?
- Kolik byste byl za takovýto ovladač a aplikaci ochoten zaplatit?
- Mohl byste mi shrnout 3 věci co pro Vást byli nejzajvavější a co naopak nejméně.
- Chcete mi sdělit ještě něco na co jsme se Vás nezeptali?

Bibliografie

1. LOUPEDECK LTD. *Products* [online]. [cit. 2024-05-15]. Dostupné z: `https://loupedeck.com/products/`.
2. LOUPEDECK LTD. *Loupedeck* [online]. [cit. 2024-05-15]. Dostupné z: `https://www.indiegogo.com/projects/1915965`.
3. LOUPEDECK LTD. *General User Guide for Loupedeck CT* [online]. 2017. [cit. 2024-05-15]. Dostupné z: `https://www.bhphotovideo.com/lit_files/594191.pdf`.
4. TOURBOX TECH INC. *TourBox - The Ultimate Controller for Creators* [online]. [cit. 2023-05-15]. Dostupné z: `https://www.tourboxtech.com/`.
5. TOURBOX TECH INC. *TourBox User Manual, Easy-to-use Controller for Creators* [online]. [cit. 2023-05-15]. Dostupné z: `https://www.tourboxtech.com/en/manual.html`.
6. MONOGRAM. *Modular Productivity Tool for Creative Pros* [online]. [cit. 2024-05-15]. Dostupné z: `https://monogramcc.com/`.
7. RSJAFFE. *MIDI2LR by Rsjaffe* [soft.]. 2024. [cit. 2024-05-15]. Dostupné z: `https://rsjaffe.github.io/MIDI2LR/`.
8. ADOBE INC. *Lightroom vs. Lightroom Classic | Adobe* [online]. [cit. 2023-04-13]. Dostupné z: `https://www.adobe.com/products/photoshop-lightroom-classic/lightroom-cc-vs-lightroom-classic.html`.
9. ADOBE INC. *Creative Cloud - Lightroom Classic* [online]. [cit. 2024-04-14]. Dostupné z: `https://developer.adobe.com/lightroom-classic/`.
10. ADOBE INC. *Lightroom Classic User Guide* [online]. [cit. 2023-04-17]. Dostupné z: `https://helpx.adobe.com/content/help/en/lightroom-classic/user-guide.html`.
11. OXD. *Cross-Platform vs. Native Application Development: How to Choose?* [online]. oxd.com. [cit. 2024-04-14]. Dostupné z: `https://oxd.com/insights/cross-platform-vs-native-application-development-how-to-choose/`.
12. OPENJS FOUNDATION AND ELECTRON CONTRIBUTORS. *Electron* [soft.]. 2024. [cit. 2024-03-02]. Dostupné z: `https://electronjs.org/`.
13. TAURI CONTRIBUTORS. *Tauri Architecture | Tauri Apps* [online]. 2024-01-08. [cit. 2024-03-02]. Dostupné z: `https://tauri.app`.
14. OPENJS FOUNDATION AND ELECTRON CONTRIBUTORS. *Introduction | Electron* [online]. [cit. 2024-03-02]. Dostupné z: `https://electronjs.org/docs/latest/`.
15. QT GROUP. *Qt | Tools for Each Stage of Software Development Lifecycle* [online]. [cit. 2024-03-18]. Dostupné z: `https://www.qt.io`.

16. QT GROUP. *Qt - Obligations of the GPL and LGPL* [online]. [cit. 2024-03-18]. Dostupné z: https://www.qt.io/licensing/open-source-lgpl-obligations.

17. AXELSON, Jan. *USB Complete: The Developer's Guide* [online]. Madison, UNITED STATES: Lakeview Research, 2015 [cit. 2024-03-09]. ISBN 978-1-931448-29-1. Dostupné z: http://ebookcentral.proquest.com/lib/techlib-ebooks/detail.action?docID=1980104.

18. USB IMPLEMENTERS' FORUM. *USB4® | USB-IF* [online]. [cit. 2024-03-09]. Dostupné z: https://www.usb.org/usb4.

19. USB IMPLEMENTERS' FORUM. *Device Class Definition for Human Interface Devices (HID)*. [B.r.]. Dostupné také z: https://www.usb.org/sites/default/files/hid1_11.pdf.

20. BLUETOOTH SIG, INC. *Bluetooth Technology Overview* [online]. Bluetooth® Technology Website. [cit. 2024-05-09]. Dostupné z: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/.

21. BLUETOOTH SIG, INC. *Bluetooth Core Specification*. 2023. Dostupné také z: https://www.bluetooth.com/specifications/specs/core-specification-5-4/.

22. BLUETOOTH SIG, INC. *HID over GATT Profile Specification 1.0* [online]. 2011. [cit. 2024-05-09]. Dostupné z: https://www.bluetooth.com/specifications/specs/hid-over-gatt-profile-1-0/.

23. BLUETOOTH SIG, INC. *HID Service Specification 1.0*. 2011. Dostupné také z: https://www.bluetooth.com/specifications/specs/human-interface-device-service-1-0/.

24. BLUETOOTH SIG, INC. *Battery Service* [online]. Bluetooth® Technology Website. [cit. 2024-05-09]. Dostupné z: https://www.bluetooth.com/specifications/bas-1-1/.

25. BLUETOOTH SIG, INC. *Device Information Service* [online]. Bluetooth® Technology Website. [cit. 2024-05-10]. Dostupné z: https://www.bluetooth.com/specifications/dis-1-2/.

26. DEMERS, Claudéric. *Dnd Kit – a Modern Drag and Drop Toolkit for React* [soft.]. 2024. [cit. 2024-05-15]. Dostupné z: https://dndkit.com/.

27. ESPRESSIF SYSTEMS. *ESP32-S3-WROOM-1*. 2023. Dostupné také z: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.

28. MOKOSMART. *MK07 nRF52833 Module Mesh Module - MOKOSmart* [online]. MOKOSmart #1 Smart Device Solution in China. [cit. 2024-05-13]. Dostupné z: https://www.mokosmart.com/bluetooth-mesh-nrf52833-module-mk07/.

29. MINEW. *Powerful nRF52833 Module MS88SF2 Specification*. 2020. Dostupné také z: https://www.minew.com/uploads/MS88SF2_V1.0-nRF52833-Datasheet.pdf.

30. RENESAS. *DA1469x*. 2022. Dostupné také z: https://www.renesas.com/us/en/document/dst/da1469x-datasheet?r=1606281.

31. MOUSER ELECTRONICS INC. *ESP32-S3-WROOM-1-N8 Espressif Systems | Mouser* [online]. Mouser Electronics. [cit. 2024-05-13]. Dostupné z: https://www.mouser.com/ProductDetail/Espressif-Systems/ESP32-S3-WROOM-1-N8?qs=Wj%2FVkw3K%252BMC1mq0KGPvoSQ%3D%3D.

32. MOKOSMART. *MOKOsmart* [online]. MOKOSmart #1 Smart Device Solution in China. [cit. 2024-05-15]. Dostupné z: https://www.mokosmart.com/.

33. MINEW. *Mesh Compatible Module | nRF52840-MS88SF21* [online]. Minewstore. [cit. 2024-05-13]. Dostupné z: https://www.minewstore.com/product/nrf52840-ms88sf2/.

34. PREMIER FARNELL LIMITED. *DA14695MOD-00F3200C - Bluetooth-Modul, Bluetooth LE 5.2, -96dBm* [online]. [cit. 2024-05-13]. Dostupné z: https://de.farnell.com/renesas/da14695mod-00f3200c/bluetooth-modul-ble-5-2-96dbm/dp/4229231.

35. NORDIC SEMICONDUCTORS. *nRF Connect SDK - Nordic Semiconductor* [online]. [cit. 2024-03-02]. Dostupné z: https://www.nordicsemi.com/Products/Development-software/nRF-Connect-SDK.

36. NORDIC SEMICONDUCTORS. *Application Event Manager — nRF Connect SDK 2.5.99 Documentation* [soft.]. 2024. [cit. 2024-03-07]. Dostupné z: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/libraries/others/app_event_manager.html#app-event-manager.

37. NORDIC SEMICONDUCTORS. *Common Application Framework — nRF Connect SDK 2.5.99 Documentation* [soft.]. 2024. [cit. 2024-03-07]. Dostupné z: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/libraries/caf/index.html.

38. LUIS, Álvaro; CASARES, Pablo; CUADRADO-GALLEGO, Juan J.; PATRICIO, Miguel A. PSON: A Serialization Format for IoT Sensor Networks. *Sensors* [online]. 2021, vol. 21, no. 13, s. 4559 [cit. 2024-03-24]. ISSN 1424-8220. Dostupné z DOI: 10.3390/s21134559.

39. GOOGLE INC. *Protobuf Overview* [online]. Protocol Buffers Documentation. [cit. 2023-03-02]. Dostupné z: https://protobuf.dev/overview/.

40. AIMONEN, Petteri. *Nanopb* [soft.]. nanopb, 2024. [cit. 2024-03-24]. Dostupné z: https://github.com/nanopb/nanopb.

41. ZEPHYR. *C++ Language Support — Zephyr Project Documentation* [online]. [cit. 2024-03-29]. Dostupné z: https://docs.zephyrproject.org/latest/develop/languages/cpp/index.html.

42. GOOGLE INC. *Language Guide (Proto 3)* [online]. [cit. 2024-04-14]. Dostupné z: https://protobuf.dev/programming-guides/proto3/.

43. RUCKERBAUER, Roland. *Hidapi - Rust* [soft.]. 2024. [cit. 2024-04-13]. Dostupné z: https://docs.rs/hidapi/2.6.1/hidapi/index.html.

44. HIDAPI. *Hidapi* [soft.]. libusb, 2024. [cit. 2024-04-13]. Dostupné z: https://github.com/libusb/hidapi.