VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

USB HID MONITOR

USB HID MONITOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ONDŘEJ BUDAI**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. RICHARD RŮŽIČKA, Ph.D., MBA**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19630/2016/xbudai00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Budai Ondřej**
Obor: Informační technologie
Téma: **USB HID Monitor**
USB HID Monitor
Kategorie: Vestavěné systémy

Pokyny:

1. Nastudujte standard USB HID, terminologii, deskriptory, zaměřte se na to, jak komunikovat se zařízeními HID z linuxu (libusb, libhid, ...). Seznamte se s vytvářením grafických uživatelských rozhraní v Linuxu (GTK, QT, ...).
2. Navrhněte nástroj pro interakci se zařízením typu USB HID v reálném čase, který by umožňoval manipulovat s výstupy (např. rozsvítit LED), zobrazovat stav vstupů a podporoval konfiguraci ve vhodné formě (např. jako GUI). Nástroj by měl umožňovat alespoň základní sadu operací dle standardu HID a dovolovat snadné rozšiřování o další.
3. Navržený nástroj implementujte.
4. Demonstrujte funkčnost implementovaného nástroje s vhodným zařízením.

Literatura:

- Dle pokynů vedoucího a konzultanta.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Růžička Richard, doc. Ing., Ph.D., MBA**, UPSY FIT VUT
Konzultant: Škarvada Jaroslav, Ing., Ph.D., UPSY FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Cílem této práce je navrhnout a implementovat aplikaci pro detailní analýzu struktury a komunikace USB HID zařízení. Vytvořená aplikace je schopna vizualizovat různé informace získané ze zařízení. Podporuje obousměrnou komunikaci mezi počítačem a HID zařízením a je dostupná pro operační systémy Windows a GNU/Linux pod svobodnou licencí. Aplikace s podobnou množinou vlastností do současné chvíle neexistovala. Její vznik je tedy velmi pozitivní pro všechny vývojaře USB HID zařízení.

Abstract

The goal of this thesis is to design and implement application for in-depth analysis of USB HID devices' structure and communication. The resulting application is capable of visualization various information retrieved from device. It can handle bidirectional communication between computer and HID device and it's available for operating systems Windows and GNU/Linux under libre software license. Application with this set of features currently does not exist. Thus, its creation is really beneficial for all USB HID developers.

Klíčová slova

USB, HID monitor, HID analyzátor, C++, Qt, libusb, svobodný software

Keywords

USB, HID monitor, HID analyzer, C++, Qt, libusb, free software

Citace

BUDAI, Ondřej. *USB HID Monitor*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Richard Růžička, Ph.D., MBA

USB HID Monitor

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a to pod vedením pana doc. Ing. Richarda Růžičky, PhD., MBA. Další informace mi poskytl konzultant Ing. Jaroslav Škarvada, PhD., z firmy Red Hat Czech. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ondřej Budai
14. května 2017

Poděkování

Na tomto místě bych rád poděkoval panu doc. Ing. Richardovi Růžičkovi, PhD., MBA. za vedení práce. Dále bych rád poděkoval Ing. Jaroslavovi Škarvadovi, PhD., za konzultace ohledně vývoje projektu Hidviz.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce se zabývá návrhem a implementací grafického nástroje pro interaktivní analýzu periferního USB zařízení třídy HID. Hlavní motivací vzniku takového nástroje je, že aktuálně neexistuje žádná podobná aplikace. To jest taková aplikace, která by přímo cílila na tuto skupinu zařízení, byla dostupná na více platformách a byla k dispozici pod svobodnou licencí, což například usnadňuje její nasazení ve výuce.

Pro tento nástroj se nabízí 2 hlavní případy užití: V první řadě může pomoci vývojářům HID zařízení při vývoji a testování. Druhým případem užití je podpora reverzního inženýrství stávajících zařízení. Příkladem může být tvorba ovladačů pro systém GNU/Linux pro zařízení, která na tomto operačním systému nejsou výrobcem podporována.

Cílem je tedy navrhnout aplikaci, která by umožnila vizualizovat vnitřní strukturu HID zařízení a jeho aktuální stav. V případech, kdy to zařízení umožňuje, by měla také poskytovat uživateli možnost tento stav zařízení měnit. V případě HID klávesnice by tedy měla zobrazit seznam všech vstupů – kláves – a zvýraznit stisknuté. Dále by měla zobrazit všechny výstupy – indikační LED diody (Num Lock, Scroll Lock a Caps Lock) – a umožnit změnu jejich stavu.

Výsledná aplikace je pojmenovaná Hidviz a je dostupná pod svobodnou licencí na webové službě GitHub[1]. Nachází se tam také návod na sestavení aplikace pro operační systémy Windows a GNU/Linux. Uživatelům je zde dostupná i možnost nahlásit chybu v aplikaci, případně požádat o implementaci nové funkcionality.

Práce je členěna na několik základních částí: Prvně jsou řazeny teoretické kapitoly, které čtenáři vysvětlí základní principy sběrnice USB (kapitola 2) a zařízení třídy HID (kapitola 3) a představí aplikace s podobným zaměřením (kapitola 4). Po teoretické části následuje kapitola 5 popisující návrh aplikace a kapitola 6 popisující její implementaci. Konečně kapitola 7 se věnuje několika technikám, které byly použity pro to, aby se Hidviz stal používaným nástrojem v cílové komunitě.

Na konci práce jsou řazeny 2 přílohy: První popisuje obsah přiloženého CD a druhá demonstruje použití vytvořené aplikace na několika zařízeních.

V textu práce je použito mnoho termínů spojených se sběrnicí USB a třídou HID. Tyto termíny jsou uváděny v českém jazyce, nicméně za prvním výskytem je vždy kurzívou v závorce uveden i původní termín v angličtině, neboť česká terminologie není v mnohých případech ustálená nebo se anglická terminologie používá častěji.

[1] Projekt Hidviz na Githubu – https://github.com/ondrejbudai/hidviz/

Kapitola 2

USB – Univerzální sériová sběrnice

USB je zkratka pro *Universal Serial Bus*, což do češtiny překládáme jako univerzální sériová sběrnice. V této práci je uveden pouze stručný přehled vlastností souvisejících s tvorbou aplikace převzatý z [1], [2], [3] a [4].

2.1 Topologie

Fyzická topologie USB se označuje jako vícestupňová hvězda. Veškerá komunikace je typu master-slave. Původně byla sběrnice navržena jako half-duplex, nicméně ve verzi 3.0 přibyla možnost full-duplex komunikace v režimu nazvaném Super-speed. Prvkem master v topologii sběrnice USB je tzv. hostitel (*host*). Roli hostitele obvykle zastává osobní počítač. Ten disponuje jedním nebo více řadiči (*controller*). Řadič je připojen ke kořenovému rozbočovači (*root hub*), jehož porty jsou k dispozici pro připojování periferních zařízení. Místo periferního zařízení je možné připojit další rozbočovač, který vytváří další úroveň topologie. Tímto způsobem je možné rozbočovače řetězit až do sedmé úrovně. Celkem je možné k jednomu řadiči připojit až 127 periferních zařízení a rozbočovačů.

Pro práci se sběrnicí je důležitý fakt, že logická topologie USB má podobu jednoduché hvězdy. To znamená, že pro uživatelské aplikace komunikující s USB zařízeními se tato zařízení jeví jako kdyby byla připojena přímo k hostiteli. Rozbočovače jsou v této situaci pro aplikace takříkajíc „neviditelné“. Operační systém má ovšem stále možnost komunikovat i s rozbočovači, pokud je to potřeba.

Periferní zařízení (dále jen zařízení) se dělí do několika tříd, například hromadná úložiště (*mass storage*), tiskárny, rozbočovače nebo pro tuto práci stěžejní HID zařízení. HID je zkratka pro *Human interface device* a v českém překladu znamená *Zařízení pro komunikaci člověka a stroje*. Termín USB HID zařízení nebo zkráceně HID zařízení proto bude používán pro periferní zařízení dle standardu USB, které patří do třídy HID. Třída HID je dle standardu označena číslem 3 a je definovaná sadou dokumentů, z nichž hlavní je [5]. Příkladem HID zařízení může být například klávesnice, myš nebo herní ovladač. Více informací o těchto zařízeních nalezne čtenář v kapitole 3.

2.2 Komunikace

Každé zařízení může obsahovat až 32 koncových bodů (16 vstupních a 16 výstupních), se kterými může hostitel komunikovat. Spojení mezi konkrétním koncovým bodem konkrétního USB zařízení a hostitelem se nazývá kanál (*pipe*). Existují 2 základní typy kanálů: proudový

(*stream*) a zpráv (*message*). Proudové jsou jednosměrné a přenášejí data, která nejsou definovaná specifikací USB. Kanály zpráv jsou obousměrné a jejich formát je specifikací pevně definovaný[1]. Každé zařízení musí podporovat alespoň jeden kanál zpráv označovaný jako řídící kanál 0, kterým probíhá základní konfigurace zařízení.

Nejmenší komunikační jednotkou důležitou pro tuto práci je přenos[2] (*transfer*). Jednotlivé přenosy jsou přenášeny skrze kanály, přičemž jeden přenos je vždy přenášen pouze přes jeden kanál. Veškerou komunikaci vzhledem k topologii master-slave vždy iniciuje USB hostitel. Vzhledem k tomu, že kanál spojuje vždy koncový bod zařízení a hostitele, je zřejmé, že veškerá komunikace pomocí USB probíhá vždy mezi periferním zařízením a hostitelem. Přímá komunikace dvou periferních zařízení tedy možná není.

Existují 4 druhy přenosů: řídící (*control*), přenosy přerušení (*interrupt*), izochronní (*isochronous*) a hromadné (*bulk*). Pro HID zařízení jsou důležité první 2 typy přenosů.

2.2.1 Řídící přenosy

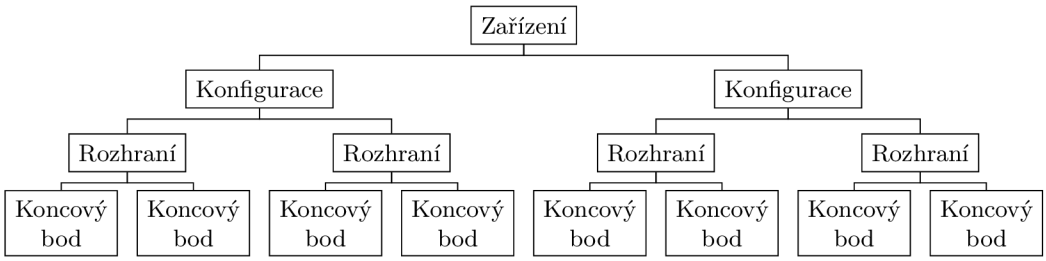
Řídící přenos je základním typem přenosu, pomocí kterého probíhá zjišťování informací o zařízení a jeho konfigurace. Každé zařízení musí podporovat základní sadu USB příkazů, které jsou posílány pomocí řídícího přenosu skrze řídící kanál 0. U HID zařízení lze pomocí řídícího přenosu měnit jeho stav. Pokud však zařízení obsahuje koncový bod schopný přijímat přenosy přerušení, je preferováno jeho použití.

2.2.2 Přenosy přerušení

Přenosy přerušení jsou specifické tím, že mají ve sběrnici garantovanou odezvu. Jsou tedy výhodné na posílání malých zpráv, u kterých je vhodné mít minimální zpoždění. Tento typ přenosu se u HID zařízení používá na získávání informací o jeho stavu a na změnu tohoto stavu.

2.3 Struktura USB zařízení

USB zařízení lze dělit do menších struktur. Zařízení může mít několik konfigurací (*configurations*), přičemž v jednom okamžiku se může nacházet pouze v jedné z nich a v každé může mít zcela jiné vlastnosti. Každá konfigurace má několik rozhraní (*interfaces*). Toto umožňuje, že zařízení může najednou spadat do více tříd a plnit tak současně více funkcí. Konečně může mít každé rozhraní přiděleno několik koncových bodů, kterými může komunikovat. Tato struktura je znázorněna na obrázku 2.1.



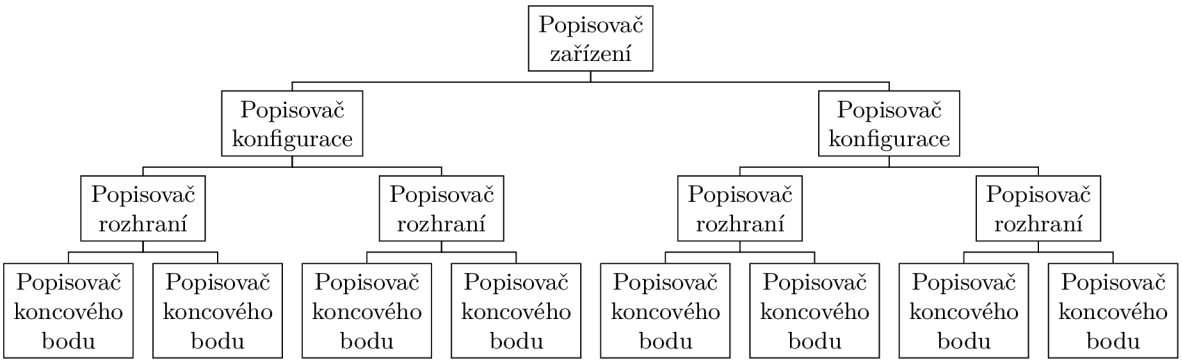
Obrázek 2.1: Struktura USB zařízení

[1] K dispozici je ale i rezervovaný typ zprávy, který si může výrobce zařízení sám definovat.

[2] Existují i nižší komunikační jednotky jako transakce nebo pakety, které ovšem nejsou pro tuto práci podstatné.

2.4 Popisovače

Každé zařízení obsahuje popisovače (*descriptors*), které slouží hostiteli k tomu, aby mohl zařízení rozpoznat a použít ke komunikaci s ním vhodný ovladač. Popisovač lze chápat jako datovou strukturu, která obsahuje informace o dané části USB zařízení. Každá část USB zařízení (viz obrázek 2.1) má svůj vlastní popisovač. Struktura popisovačů je tedy velmi podobná struktuře USB zařízení a je zobrazena na obrázku 2.2.



Obrázek 2.2: Struktura USB popisovačů

Kromě standardních popisovačů, které popisují základní strukturu USB zařízení, existují i speciální popisovače specifické pro jednotlivé třídy. Velké množství informací o HID zařízení je uloženo v HID popisovači, který bude detailně popsán v sekci 3.2.

Kapitola 3

HID – Zařízení pro komunikaci člověka a stroje

HID je zkratka z anglického výrazu *Human interface device*, což lze do češtiny přeložit jako *zařízení pro komunikaci člověka a stroje*. V této práci bude tento termín používán pro periferní zařízení sběrnice USB třídy 3, tedy USB HID zařízení nebo zkráceně pouze HID zařízení. Existují nicméně i další technologie, které převzaly myšlenky HID. Patří mezi ně například standard pro bezdrátovou komunikaci Bluetooth a jeho HID profil nebo I^2C. Popis HID v tomto textu je převzatý z [1] a [5].

Třída HID se skládá primárně ze zařízení, kterými člověk ovládá počítač. Typickými příklady jsou klávesnice, myši, herní zařízení, ovládací panely nebo ovladače pro letecké či jiné simulátory. HID zařízení mohou zároveň i informovat uživatele o stavu počítače. Například klávesnice obsahuje 3 informační LED diody – Num Lock, Scroll Lock a Caps Lock. Dalším příkladem může být vibrační odezva u herních volantů. Do HID třídy se dále řadí i zařízení, která nekomunikují přímo s člověkem, ale poskytují podobná data. Příkladem mohou být voltmetry, čtečky čárových kódů nebo meteostanice. Naopak zařízení, které pracují s velkými datovými toky, do HID třídy nespadají – počítačový monitor tedy rozhodně mezi HID zařízení nepatří.

Další důležitou myšlenkou třídy HID je, že se snaží neomezovat výrobce v tom, jaké konkrétní funkce bude zařízení mít. Třída proto dovoluje funkce libovolně kombinovat, a tudíž není problém vyrobit různorodá zařízení s rozličnými funkcemi. Pokud tedy potřebuje výrobce vyrobit počítačovou myš kombinovanou s teploměrem, třída HID mu poskytuje dostatečnou flexibilitu. Tento fakt nicméně značně zvyšuje složitost hostitelských ovladačů, které musí detailně porozumět struktuře HID zařízení před tím, než s ním začnou komunikovat.

Při komunikaci s HID zařízeními je nutné postupovat ve dvou základních krocích. V prvním kroku je potřeba ze zařízení načíst a zpracovat HID popisovač, který obsahuje informace o struktuře zařízení. Z této struktury lze následně odvodit strukturu zpráv, které posílá zařízení hostiteli a které obsahují informace o aktuálním stavu zařízení a jeho funkcí. Až ve druhém kroku je možné zahájit periodické čtení HID zpráv, získat z nich data o jednotlivých prvcích HID zařízení a tato data dle určení zařízení dále použít. Například v případě herního ovladače je potřeba, aby ovladač z příchozí zprávy extrahoval informace o poloze ovládacích pák a předal informace leteckému simulátoru, který přenastaví polohu ocasních ploch.

3.1 Prvky HID

HID zařízení se mohou skládat z mnoha prvků. Například klávesnice se obvykle skládá z přibližně 100 kláves a 3 LED diod. Tyto prvky se v rámci HID třídy dělí na 3 kategorie - vstup (*input*), výstup (*output*) a vlastnost (*feature*).

Vstupní prvek slouží ke komunikaci směrem od zařízení k hostiteli. Ve smyslu HID zařízení ho tedy můžeme chápat jako prostředek komunikace od člověka směrem k počítači. Příkladem může být tlačítko myši nebo plynový pedál.

Výstupní prvek slouží ke komunikaci opačné – směrem od hostiteli k zařízení. Pomocí výstupního prvku tedy může počítač informovat uživatele o jeho interním stavu. Příkladem může být indikační LED dioda Caps Lock.

Prvek vlastnost je určen k nastavování samotného HID zařízení a není tedy přímým prostředkem komunikace počítače s člověkem. Nicméně změna nastavení může ovlivnit způsob, jakým ke komunikaci člověka s počítačem dochází. Příkladem takové vlastnosti je změna rozlišení pohybového senzoru na myši. U vlastnosti je nutné zajistit oboustrannou komunikaci – hostitel musí mít možnost zjistit, jak je vlastnost daného zařízení aktuálně nastavená, a zároveň možnost toto nastavení změnit.

Každý prvek má množinu charakteristik, které jej popisují. Například každý prvek musí mít specifikováno, kolik bitů je potřeba na zakódování jeho stavu a jaký je rozsah těchto stavů.

Speciálním druhem prvku je kolekce, která umožňuje shlukovat dohromady prvky obsahující nějakou společnou vlastnost. Příkladem může být klávesnice s integrovaným dotykovým panelem. Prvky klávesnice by pak mohly být uloženy v jedné kolekci, zatímco prvky dotykového panelu v kolekci druhé[1].

3.2 HID popisovač

Informace o struktuře HID zařízení jsou uloženy v takzvaném HID popisovači. Tento popisovač lze vhodnou USB zprávou ze zařízení načíst skrze řídící přenos přes kanál 0.

Základní jednotkou dat v HID popisovači je položka (*item*). HID popisovač je tedy soubor více po sobě jdoucích položek. Položky se dělí do 2 hlavních skupin:

Hlavní položky Každá hlavní položka HID popisovače reprezentuje jeden nebo více prvků HID zařízení. Typická myš má v HID popisovači jednu položku pro obě osy senzoru změny polohy a jednu položku pro všechna tlačítka.

Globální a lokální položky Tyto položky informují o charakteristice prvků. Pokud program zpracovávající HID popisovač narazí na položku z této skupiny, uloží si záznam o druhu charakteristiky a její hodnotě do paměti. Jakmile později načte program hlavní položku, přiřadí prvku, který tuto položku reprezentuje, všechny doposud načtené charakteristiky. Pokud program narazí na položku reprezentující charakteristiku, jejíž hodnotu již má načtenou, hodnota z této nové položky přepíše hodnotu starou.

Ukázka zpracování jednoduchého HID popisovače je ukázána na obrázku 3.1. Zpracovávající program si nejprve uloží do paměti hodnotu charakteristiky a a b. Když poté načte

[1] Tento problém je však možné řešit i jinými cestami – klávesnice i panel mohou být každý definovaný v jiném rozhraní USB zařízení, případně mohou být dokonce úplně rozděleny na 2 samostatná USB zařízení, která jsou připojena do jednoho integrovaného rozbočovače (tzv. složené USB zařízení).

položku obsahující informaci o existenci prvku X, přiřadí tomuto prvku všechny dříve načtené charakteristiky. Následně program načte novou hodnoty charakteristiky a, která přepíše původní uloženou hodnotu. Prvek Y tedy bude mít stejnou hodnotu charakteristiky b jako prvek X, ale jinou hodnotu charakteristiky a.

Obsah HID popisovače

Charakteristika	**Charakteristika**	**Prvek**	**Charakteristika**	**Prvek**
a = 10	b = 1	X	a = 5	Y

Vzniklá reprezentace HID zařízení

Prvek X	**Prvek Y**
a = 10	a = 5
b = 1	b = 1

Obrázek 3.1: Ukázka zpracování jednoduchého HID popisovače obsahujícího 2 prvky

3.2.1 Použití

Nejdůležitější charakteristikou prvků HID zařízení je použití (*usage*), které přiřazuje prvkům sémantické role. Použití dává tedy prvkům význam. Bez použití by stav prvků představoval pouze abstraktní data bez konkrétního významu. Právě podle použití může hostitel určit význam prvku HID zařízení a při změně jeho stavu zareagovat příslušnou akcí. Pro příklad mějme prvek s použitím Tlačítko 1 (*Button 1*), který je obsažen v kolekci s použitím Myš (*Mouse*) a kolekci s použitím Ukazatel (*Pointer*). Pokud tento prvek změní v rychlém sledu svůj stav z hodnoty 0 na hodnotu 1 a zpět, operační systém na základě jeho použití vyhodnotí tyto změny jako jednoduché kliknutí levým tlačítkem myši a předá tuto informaci aplikaci, nad kterou se aktuálně nachází kurzor.

Jednotlivá použití jsou rozdělena do očíslovaných skupin. V rámci popisovače lze uvést použití absolutně (tedy včetně skupiny) nebo uvést číslo skupiny pomocí charakteristiky strana použití (*usage page*) a poté ho uvádět relativně – bez čísla skupiny.

Veškerá použití jsou definovaná v dokumentu [6] a dalších doplňujících dokumentech.

3.2.2 Vybrané charakteristiky

Popisovač HID zařízení může obsahovat mnoho druhů charakteristik. V následujících odstavcích uvedu pouze několik vybraných, které jsou bezpodmínečně nutné pro základní funkcionalitu těchto zařízení.

Logické minimum *(Logical minimum)*
Logické maximum *(Logical maximum)*

Logické minimum a maximum informují hostitele, v jakém rozsahu hodnot je kódován stav daného prvku.

Počet zpráv
(Report count)

Charakteristika počet zpráv informuje o tom, kolik daný prvek obsahuje podprvků. Toto je obzvlášť výhodné, pokud zařízení obsahuje mnoho prvků s podobnými vlastnostmi. Tyto prvky lze pak spojit do jednoho, uvést postupně více použití[2] a nastavit odpovídající počet zpráv.

Velikost zprávy
(Report size)

Velikost zprávy určuje, kolik bitů bude daný prvek využívat na zakódování svého stavu v HID zprávě. V případě, že prvek obsahuje více zpráv (dle charakteristiky počet zpráv), pak jsou stavy jednotlivých podprvků v HID zprávě uvedeny spojitě za sebou. Celkem tedy stav prvku obsadí počet bitů, který odpovídá součinu počtu zpráv a jejich velikosti.

3.2.3 Ukázka HID popisovače

Na obrázku 3.2 je ukázán příklad popisovače pro počítačovou myš. Všechny vstupy se nacházejí ve dvou vzájemně vnořených kolekcích, které určují typ zařízení. První vstup je shluk 8 podprvků – tlačítek. Tomu odpovídá i počet zpráv, který je roven 3. Následuje vstup kombinující osy X a Y, kde každá z os je kódována na 16 bitů v rozsahu od -32767 do 32767. Kolečko myši je umístěno ve vlastním prvku, jehož stav se kóduje pouze na 8 bitech v rozsahu od -127 do 127.

```
Usage Page (Generic Desktop)
Usage (Mouse)
Collection (Application)
    Usage (Pointer)
    Collection (Physical)
        Usage Page (Button)
        Usage Minimum (Button 1)
        Usage Maximum (Button 8)
        Logical minimum (0)
        Logical maximum (1)
        Report count (8)
        Report size (1)
        Input
        Usage Page (Generic Desktop)
        Logical minimum (-32767)
        Logical maximum (32767)
        Report size (16)
        Report count (2)
        Usage (X)
        Usage (Y)
        Input
        Logical minimum (-127)
        Logical maximum (127)
        Report size (8)
        Report count (1)
        Usage (Wheel)
        Input
    End Collection
End Collection
```

Obrázek 3.2: Ukázka HID popisovače počítačové myši, zvýrazněné řádky představují hlavní položky

[2]Pro tyto účely existují také charakteristiky minimální použití (*usage minimum*) a maximální použití (*usage maximum*).

3.3 HID zprávy

Pomocí HID zpráv informuje zařízení hostitele o změně stavu některého z jeho prvků. Zpráva lze ovšem zaslat i v opačném směru – hostitel může změnit stav prvku zařízení. Příchozí zprávy (tedy od zařízení k hostiteli) vždy proudí skrze přenosy přerušení. U odchozích zpráv je situace složitější – pokud zařízení podporuje příjem přenosů přerušení, měl by hostitel využít primárně ty, nicméně je možné je odesílat i pomocí řídícího přenosu.

Ve zprávě jsou uvedeny stavy jednotlivých prvků postupně za sebou v takovém pořadí, v jakém jsou uvedeny v HID popisovači. Vícebajtové hodnoty jsou uloženy ve formátu little-endian.

Ukázka HID zprávy odeslané myší se nachází na obrázku 3.3. Formát zprávy odpovídá popisovači uvedeném dříve v textu na obrázku 3.2. Tato myš se aktuálně nachází ve stavu, kdy má stisknuté tlačítko 1 a 3, což odpovídá levému a prostřednímu tlačítku[3]. Pohybuje se rychlostí 40[4] ve směru osy X a rychlostí 20 proti směru osy Y – tyto údaje odpovídají pohybu nahoru doprava. Kolečko se pohybuje rychlostí 2 směrem dolů.

Pozice bajtu	**3**	**2**	**1**	**0**
Použití	Kolečko	Y	X	Tlačítka
Data	2	-20	40	0 0 0 0 0 1 0 1

Obrázek 3.3: Ukázka HID zprávy počítačové myši

[3]Prostřední tlačítko je na dnešních myších realizováno jako stisknutelné kolečko myši.

[4]Tato rychlost je bez jednotek, o přepočet na pohyb o daný počet pixelů se stará operační systém.

Kapitola 4

Existující řešení

V současné době existuje několik aplikací, kterými lze provádět analýzu HID zařízení. Ze svobodných řešení lze použít například nástroj lsusb z projektu usbutils nebo aplikaci usbhid-dump. Z komerčních řešení se nabízí aplikace USBlyzer.

4.1 Nástroj lsusb

Tato konzolová aplikace obsažená ve většině linuxových distribucí umožňuje uživateli vypsat kompletní strukturu USB zařízení. Kromě toho umožňuje i výpis podrobných informací pojících se s vybranými třídami USB zařízení včetně třídy HID. U této třídy dovede nástroj lsusb vypsat strukturované informace o HID popisovači ve formátu čitelném pro člověka. Nevýhodou této aplikace je, že před tím, než lze získat informace o HID popisovači, musí uživatel ručně odpojit od zařízení ovladač[1].

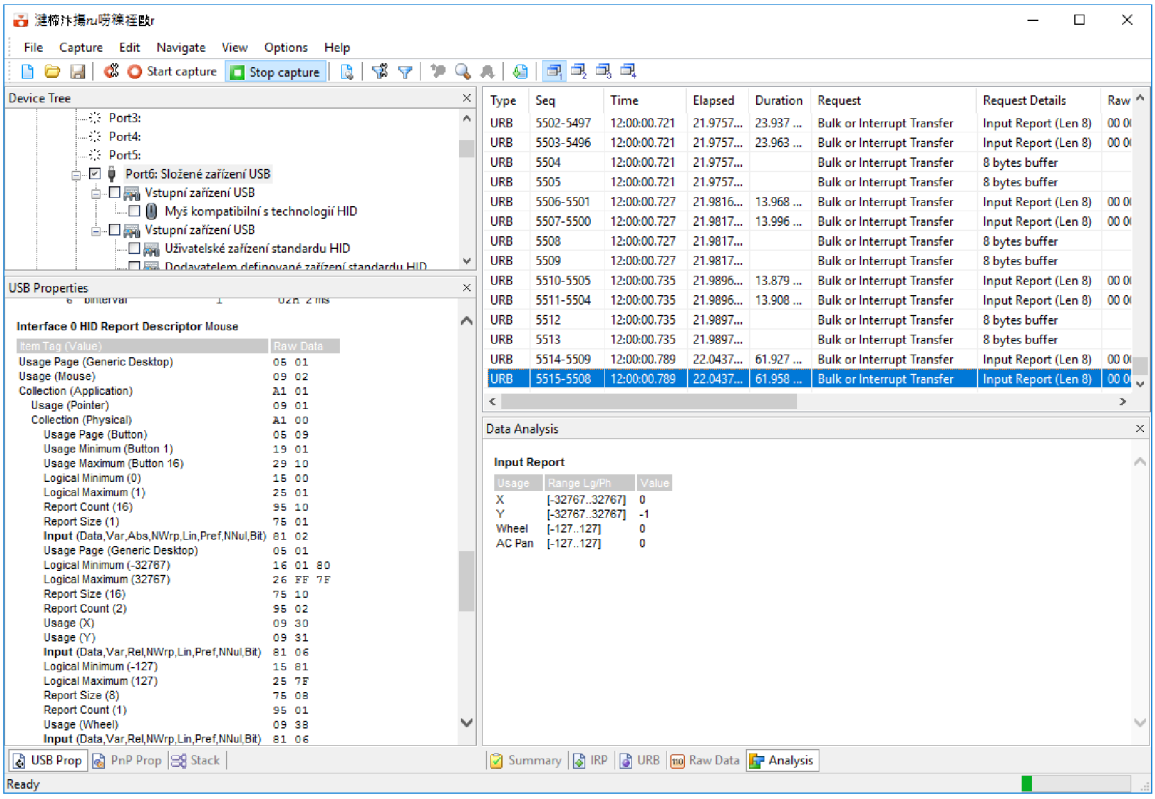
4.2 Nástroj usbhid-dump

Konzolový nástroj usbhid-dump se specializuje přímo na analýzu USB HID zařízení. Nabízí 2 základní funkce: Stejně jako nástroj lsusb umí ze zařízení získat HID popisovač. Bohužel jej však již neumí nijak zformátovat a zobrazí ho pouze jako sekvenci čísel v šestnáctkové soustavě. Druhou funkcí je čtení příchozích HID zpráv, které jsou bohužel opět kódované pouze nestrukturovaně v šestnáctkové soustavě. Zprávy nicméně nelze zasílat opačným směrem, tedy z počítače do HID zařízení. Oproti nástroji lsusb ovšem usbhid-dump umí samostatně odpojit ovladače a po skončení práce je zase připojit.

4.3 USBlyzer

Grafický nástroj USBlyzer umožňuje velmi detailní analýzu struktury USB zařízení a jejich komunikace. Nástroj podporuje analýzu nejen obecných USB struktur, ale i struktur jednotlivých tříd, mezi kterými nechybí ani HID. Prostředí tohoto nástroje je ilustrováno na obrázku 4.1. Jak je z obrázku patrné, nástroj umí přehledně zobrazit HID popisovač i HID zprávu.

[1] Pokud v operačním systému Linux nějaká aplikace potřebuje komunikovat po sběrnici USB, musí nejprve od tohoto zařízení odpojit jeho ovladač.



Obrázek 4.1: Ukázka prostředí aplikace USBlyzer

Problémem tohoto nástroje je jeho komerční licence. Volně dostupná je pouze trial verze, kterou může uživatel používat pouhých 33 dní. Po této době si musí zakoupit licenci, jejíž cena byla v době psaní tohoto textu stanovena na 200 amerických dolarů. Další nevýhodou je chybějící podpora jiných operačních systému než Microsoft Windows.

4.4 Zhodnocení

Pro analýzu USB HID zařízení za použití svobodných nástrojů se nabízí kombinace prvních dvou – tedy lsusb a usbhid-dump. První jmenovaný umožňuje získat dobře čitelný USB popisovač, zatímco díky druhému může uživatel získat HID zprávy. Problémem tohoto postupu je absence možnosti zasílat zprávy směrem do HID zařízení a žádný strukturovaný pohled na data v HID zprávě. Druhou variantou je použití komerční aplikace USBlyzer, která ovšem není šířená pod svobodnou licencí a je dostupná pouze pro operační systém Windows.

Cílem této bakalářské práce je tedy vytvořit nástroj, který odstraní nedostatky výše zmíněných nástrojů a usnadní tak analýzu USB HID zařízení maximální možnou měrou. Aplikace vytvořená v rámci této práce by měla mít tedy možnost zobrazit uživateli čitelnou reprezentaci HID popisovače a zpráv včetně vizualizace struktury zařízení. Aplikace by také měla být multiplatformní a šířená pod svobodnou licencí.

Kapitola 5

Návrh

V rámci návrhu jsem se nejprve zaměřil na uživatelské rozhraní aplikace a následně jsem navrhl základní architekturu aplikace.

5.1 Grafické rozhraní

Před návrhem grafického rozhraní jsem vybral platformu, na kterou budu monitor vyvíjet, abych mohl v návrhu počítat se specifiky vybrané platformy.

Analýzu USB zařízení obvykle není nutné provádět mobilně, a proto jsem se rozhodl, že ideální platformou bude klasický osobní počítač. Ten je výhodný i tím, že obvykle obsahuje mnoho USB portů, ke kterým lze připojit analyzovaná zařízení. Dnešní chytré mobilní telefony a tablety sice již také dovolují připojovat periferní USB zařízení, nicméně jejich použití pro tuto aplikaci je stále ještě limitované – například omezenými možnostmi komunikace přes sběrnici USB pro uživatelské aplikace. Tento problém naštěstí na osobních počítačích není. Uživatelé také jistě ocení velké rozměry obrazovky, na který bude možné přehledně zobrazit širokou škálu informací o analyzovaném zařízení.

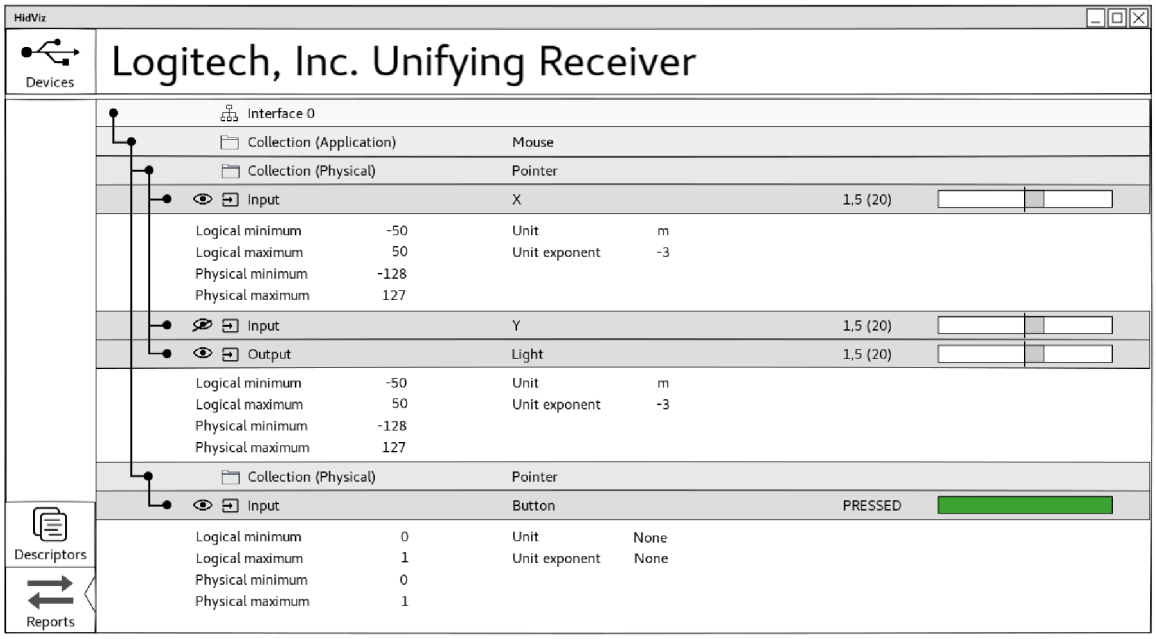
5.1.1 Obecný návrh

Po výběru platformy jsem vytvořil návrh na obrázku 5.1. Horní části dominuje hlavní ovládací panel. V levé části panelu se nachází velké tlačítko otevírající dialogové okno na výběr analyzovaného zařízení. Pravou část pak zabírá jméno aktuálně vybraného zařízení. V levé části okna se nachází postranní panel. Jeho spodní část přepíná mezi různými pohledy na analyzované zařízení, zatímco jeho vrchní část je určená pro detailnější nastavení aktuálně zvoleného pohledu. Hlavní část okna zabírá samotná analýza. V rámci návrhu jsem zpracoval 2 pohledy – hlavní pohled, který přehledně zobrazuje strukturu a stav HID zařízení, a pohled na nezpracovaný HID popisovač.

5.1.2 Hlavní pohled

Na obrázku 5.1 je zobrazen hlavní pohled analýzy. Ten se skládá ze stromového pohledu na strukturu zařízení. Uzly stromu odpovídají jednotlivým prvkům HID zařízení. Tento pohled je výhodný ze dvou důvodů: Prvně je z něj jasně viditelná struktura zařízení. Je z něj na první pohled možné zjistit, z jakých kolekcí se zařízení skládá a jaké vstupní nebo výstupní prvky obsahuje. Za druhé je z něj možné rychle určit stav všech prvků. Rozhodl jsem se, že aplikace by měla podporovat chytrou vizualizaci prvků – prvek se 2 stavy (například tlačítko

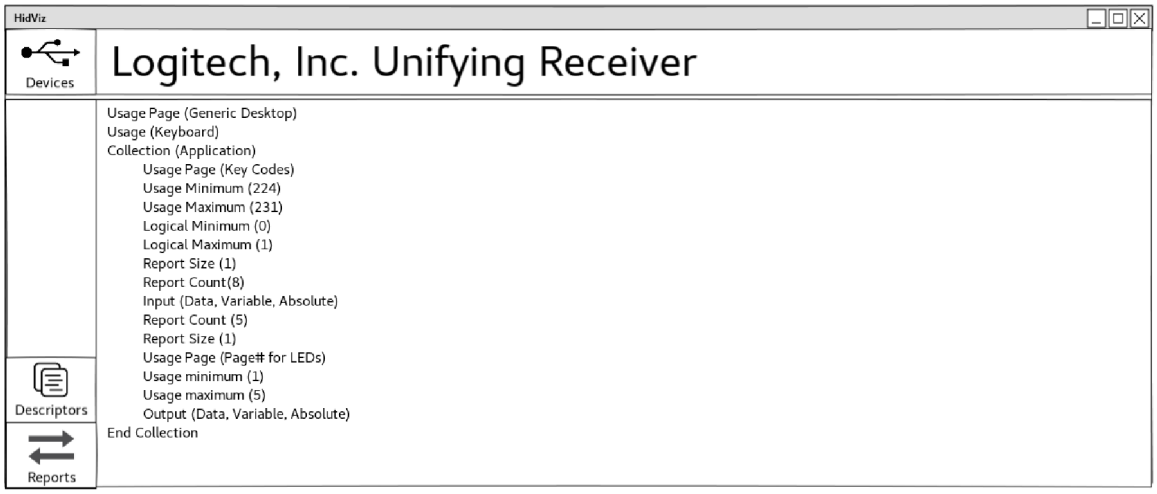
nebo LED dioda) by měl být vizualizován jinak než prvek s mnoha stavy (například poloha na grafickém tabletu).



Obrázek 5.1: Grafický návrh hlavního pohledu

5.1.3 Pohled na popisovač

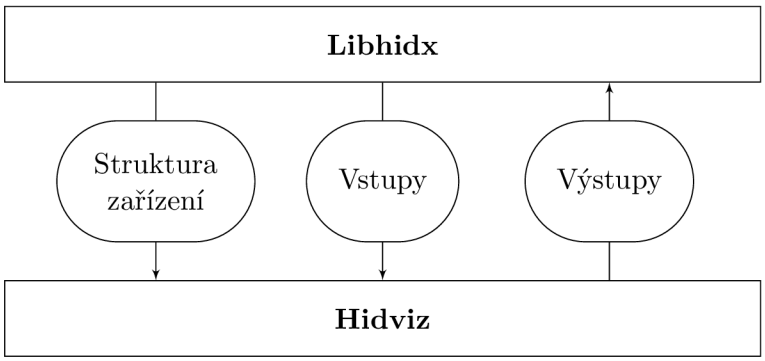
Návrh pohledu na popisovač, který uživateli poskytuje informace o nezpracovaném HID popisovači, je zobrazen na obrázku 5.2. Jednotlivé položky jsou pouze přeloženy z číselné do textové reprezentace, jinak není popisovač nijak upravován.



Obrázek 5.2: Grafický návrh pohledu na popisovač

5.2 Architektura aplikace

Rozhodl jsem se pro striktní rozdělení části, která je zodpovědná za vykreslování grafického rozhraní, a části komunikující s HID zařízeními. Věřím, že druhá část by mohla být znovupoužitelná i v jiných projektech zabývajících se komunikací s HID zařízeními. Tuto knihovnu jsem nazval *Libhidx*. Architektura komunikace obou částí je ilustrovaná na obrázku 5.3.



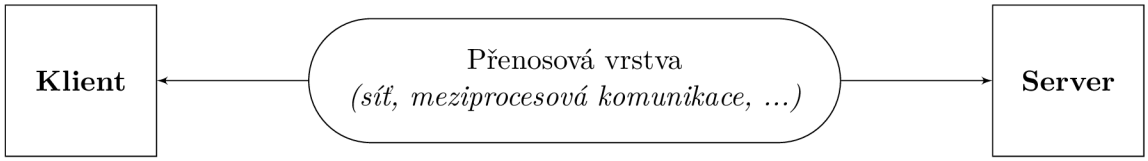
Obrázek 5.3: Architektura nástroje Hidviz a knihovny Libhidx

5.2.1 Architektura knihovny Libhidx

Během analýzy problému jsem zjistil, že pokud potřebuje aplikace pracovat na operačním systému Linux se sběrnicí USB, musí být spuštěna s právy superuživatele. To ovšem znamená problém, neboť v moderních distribucích[1] je spouštění grafických aplikací pod superuživatelem problematické. Mnou navržené řešení je rozdělit knihovnu na 2 moduly, které spolu budou komunikovat. Architektura je názorně ilustrována na obrázku 5.4.

První modul knihovny Libhidx, také nazývaný jako klient, bude slinkovaný s aplikací Hidviz a bude tudíž fungovat i s omezenými právy. Cílem tohoto modulu je poskytnout vývojářům kvalitní aplikační rozhraní bez toho, aniž by museli znát podrobnosti o poměrně složité architektuře knihovny.

Druhý modul, také nazývaný server, bude vykonávat samotnou práci s USB sběrnicí na základě pokynů z prvního modulu. Server bude spuštěn ve vlastním procesu, který bude mít práva superuživatele.



Obrázek 5.4: Architektura knihovny Libhidx

[1] Konkrétně jde o distribuce, které používají zobrazovací server Wayland. Více informací je dostupných na https://bugzilla.redhat.com/show_bug.cgi?id=1274451.

Kapitola 6

Implementace

6.1 Obecné aspekty implementace

Jak už vyplývá z návrhu, aplikace je rozdělena na 2 základní části: První částí je knihovna Libhidx a druhou částí je samotný USB HID monitor, tedy aplikace Hidviz.

Jako primární cílovou platformu jsem zvolil operační systém GNU/Linux, neboť s vývojem na něm mám nejvíce zkušeností.

6.1.1 Výběr programovacího jazyka

Po výběru platformy bylo potřeba zvolit vhodný programovací jazyk. Nejprve jsem zvolil jazyk Python, který umožňuje velmi vysokou míru abstrakce a jedná se o jazyk skriptovací, tudíž není potřeba trávit zbytečný čas sestavováním programu. Nicméně vzhledem k tomu, že jsem jako hlavní podpůrné knihovny zvolil Qt psané v C++ a libusb v C (viz další kapitoly), rozhodl jsem se, že bude lepší využít jazyk C++, neboť pro tento jazyk je v obou knihovnách lepší podpora a dokumentace. V projektu je tedy využito moderní C++ v normě z roku 2014[1].

6.1.2 Sestavování aplikace

K sestavování programu ze zdrojových kódů je použit program CMake. Jedná se o nástroj multiplatformní, který umožňuje generovat nejen konfigurační soubory pro klasický linuxový program make, ale také pro jiná vývojová prostředí jako jsou například Xcode pro MacOS nebo Microsoft Visual Studio pro operační systém Windows[7]. Oproti klasickému programu make má také výhodu v tom, že je schopen po změně hlavičkového souboru zkompilovat pouze ty zdrojové soubory, kterých se změna týká, což výrazně šetří čas rekompilace projektu[2]. CMake si také velmi dobře rozumí s projektem Qt navzdory tomu, že Qt obsahuje vlastní nástroj qmake s podobnými funkcemi. Pomocí programu CMake je také realizován instalační proces na systému Linux. V budoucnu by také bylo možné využít jeho komponentu CTest pro jednotkové testy[7].

[1] V době řešení této práce to tedy byla aktuální verze jazyka.

[2] Při použití programu make je možné vytvořit takovou konfiguraci, která bude optimalizovat proces sestavení podobným způsobem. Oproti použití programu CMake však tento postup není zcela triviální.

6.2 Implementace grafického uživatelského prostředí

Prvním krokem při tvorbě uživatelského prostředí byl výběr knihovny, kterou pro tento účel použiji. Mými požadavky byla podpora více platforem (alespoň Windows a Linux), velká rozšířenost, moderní vzhled a kvalitní programátorské rozhraní. Při prvotním výběru jsem za vhodné kandidáty zvolil knihovny GTK+[3] a Qt[4], které obě splňují všechna moje kritéria[8][9].

Po důkladnějším porovnání jsem zvolil knihovnu[5] Qt, konkrétně ve verzi 5. Hlavním argumentem pro Qt pro mě bylo hlavně subjektivně lepší aplikační rozhraní. To je dle mého názoru dosaženo tím, že projekt Qt využívá objektového přístupu v jazyce C++, zatímco projekt GTK+ používá jazyk C, který neumožňuje tak vysokou míru abstrakce.

6.2.1 Návrh uživatelského rozhraní v Qt

Většina uživatelských rozhraní v Hidvizu je vytvořena pomocí aplikace Qt Designer. Tato aplikace umožňuje programátorovi vytvářet uživatelská rozhraní velmi pohodlným způsobem ve stylu WYSIWYG[6].[8] Programátor tedy může vizuálně naaranžovat potřebné prvky v okně tak, jak potřebuje, a následně rozhraní uložit do souboru ve formátu XML. CMake pak během sestavování tento soubor přeloží speciálním překladačem do podoby klasického hlavičkového souboru jazyka C++, který zajistí vytvoření všech potřebných struktur v knihovně Qt dle návrhu. Knihovna samozřejmě dovoluje i ruční vytvoření rozhraní čistě pomocí jazyka C++, ale využití Qt Designeru tuto činnost velice zjednodušuje.

6.2.2 Hlavní pohled

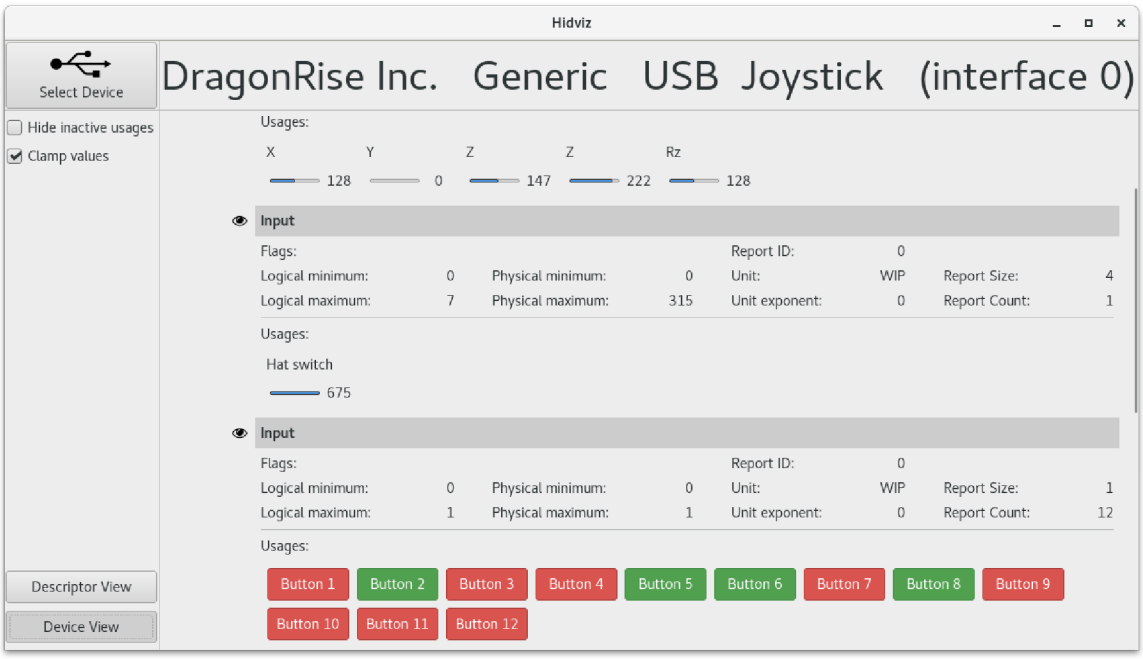
Na obrázku 6.1 je ukázána hotová implementace hlavního panelu aplikace. Pro porovnání se může čtenář podívat i na výše uvedený obrázek 5.1 v kapitole o návrhu. Z porovnání je vidět, že panel doznal jistých změn oproti návrhu tak, aby ještě lépe reflektoval strukturu HID zařízení.

Obrázek ukazuje konkrétní zařízení, joystick, který je právě pomocí aplikace Hidviz analyzován. Základní rozvržení aplikace zůstalo v souladu s návrhem zachováno. Aplikace tedy obsahuje horní panel s tlačítkem pro výběr zařízení a jménem aktuálně vybraného zařízení. Na levé straně je panel s aktuálně dvěma nastaveními (viz přílohu B).

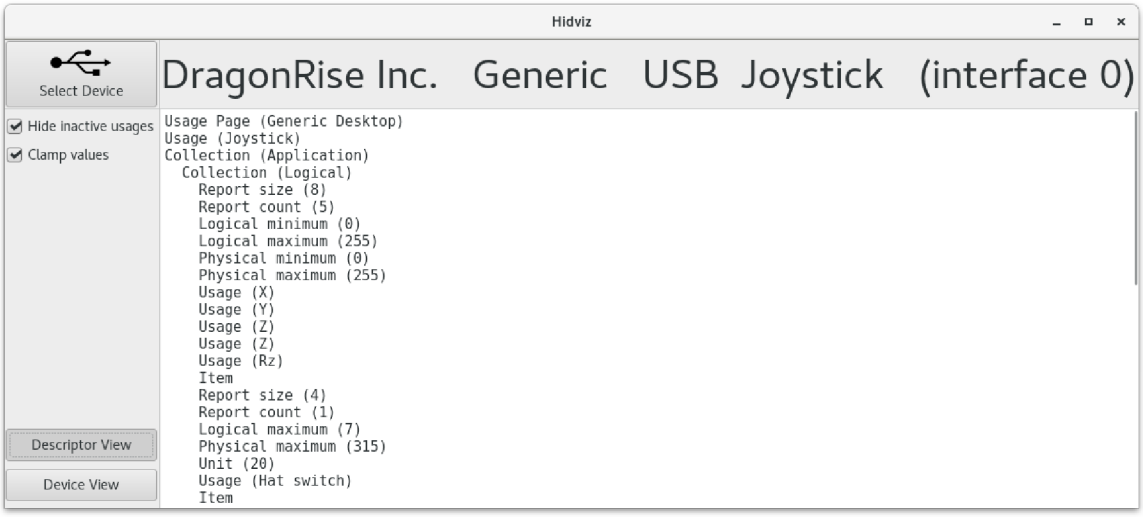
6.2.3 Pohled na popisovač

Podoba pohledu na popisovač v aplikaci Hidviz je zobrazena na obrázku 6.2. Tento pohled se s návrhem prakticky shoduje.

[3] Projekt GTK+ – https://www.gtk.org/
[4] Projekt Qt – https://www.qt.io/
[5] Vzhledem k jeho rozsahu bývá projekt Qt označován též jako aplikační rámec (*framework*).
[6] *What you see is what you get.* V českém překladu tedy: „Co vidíš, to dostaneš.“



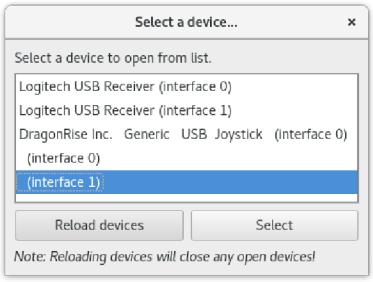
Obrázek 6.1: Výsledná podoba hlavního pohledu v aplikaci Hidviz



Obrázek 6.2: Výsledná podoba pohledu na popisovač v aplikaci Hidviz

6.2.4 Výběr zařízení

Po kliknutí na tlačítko Vybrat zařízení (*Select device*) se zobrazí dialog k tomu určený. Tento dialog je zobrazen na obrázku 6.3. Obsahuje pouze jednoduché rozhraní, ve kterém může uživatel vybrat požadované zařízení k analýze nebo obnovit seznam.



Obrázek 6.3: Dialogové okno sloužící k vybrání analyzovaného zařízení

6.3 Implementace knihovny

Podle návrhu jsem knihovnu implementoval jako 2 části, které spolu komunikují. První část je klientská a zahrnuje v sobě veřejně přístupné aplikační rozhraní, které je využíváno aplikací Hidviz. Během inicializace této části dochází k vytvoření pomocného procesu serveru, který je spuštěn s právy superuživatele, a je zodpovědný za samotnou komunikaci s HID zařízeními připojenými na USB sběrnici.

Zde je velmi důležité poznamenat, že 2 procesy se používají pouze v operačním systému GNU/Linux. Knihovna sestavená a spuštěná v prostředí OS Windows nepoužívá serverový proces. Všechno zde tedy probíhá v rámci jednoho procesu, přičemž multiprocesová komunikace je pouze simulována.

6.3.1 Serverová část

O komunikaci s HID zařízeními se stará serverová část. Původně jsem pro účely komunikace zamýšlel využití knihovny HIDAPI[7]. Vzhledem k jejímu intuitivnímu aplikačnímu rozhraní je použití této knihovny na komunikaci s HID zařízeními velmi jednoduché. Další výhodou je fakt, že je multiplatformní. Můj plán ovšem ztroskotal na tom, že knihovna HIDAPI neumí ze zařízení přečíst HID popisovač. Z tohoto důvodu tedy nešlo tuto knihovnu použít.

Poté, co se ukázalo řešení s použitím knihovny HIDAPI jako nefunkční, jsem se rozhodl pro použití knihovny libusb[8]. Tato knihovna umožňuje práci nejen s HID zařízeními, ale obecně se všemi USB zařízeními a to na mnoha platformách[10]. Jedinou nevýhodou je fakt, že díky obecnějšímu zaměření neobsahuje knihovna funkce zaměřené přímo na komunikaci s HID zařízeními, a tudíž jsem tyto funkce musel v mém projektu implementovat sám.

Na tomto místě stojí za zmínku projekt libusbnet[9], který umožňuje volat všechny funkce libusb vzdáleně přes TCP spojení. Bohužel projekt je již několik let neudržovaný a navíc podporuje pouze libusb řady 0.1.x, která už je dnes považována za zastaralou. Libhidx používá libusb řady 1.0.x.

6.3.2 Komunikace mezi serverem a klientem

Implementaci komunikace lze rozdělit na 2 samostatné body: Prvním bodem je způsob kódování zpráv, které si klient se serverem vyměňují. Druhou částí je samotný mechanismus výměny zpráv – tedy cesta, prostřednictvím které si procesy mezi sebou zprávy zasílají.

[7]Web projektu HIDAPI – http://www.signal11.us/oss/hidapi/
[8]Web projektu libusb – http://libusb.info/
[9]Web projektu libusbnet – https://github.com/vavrusa/libusbnet

Formát zpráv mezi serverem a klientem

Ke kódování zpráv mezi serverem a klientem se používá metoda *Protocol Buffers* (zkráceně *Protobuf*) vyvinutá společností Google. Pokud chceme Protobuf použít, je nejprve nutné napsat zdrojový Protobuf soubor, který specifikuje podobu zpráv. Syntaxe tohoto souboru je velmi podobná deklaraci struktur tak, jak ji známe z jazyků C nebo C++. Tento soubor se následně speciálním kompilátorem přeloží na dvojici souborů (hlavičkový a zdrojový) v jazyce C++, který obsahuje třídy odpovídající zprávám v Protobuf souboru[11].

Ukázku překladu z formátu Protobuf do jazyka C++ je možno vidět na obrázku 6.4. V levém sloupci je uveden kód v jazyce Protobuf, v pravém pak v (zjednodušeném) jazyce C++. Pomocí nástroje Protobuf jsou takto abstrahována veškerá volání knihovny libusb potřebné v rámci aplikace Hidviz. Volání funkce je vždy kódováno jako zpráva se dvěma podzprávami - požadavkem (*Request*), který posílá klient serveru, a odpovědí (*Response*), která míří směrem opačným. Každá položka zprávy se do jazyka C++ převede vždy jako dvojice metod – jedna na nastavení hodnoty (*setter*) a druhá na její získání (*getter*).

Pomocí deklarace `package` lze všechny zprávy v Protobufu příhodně vnořit do jmenného prostoru v jazyce C++.

```
package libhidx.buffer;

message DetachKernelDriver {
    message Request {
        required uint64 handle = 1;
        required int32 interfaceNumber = 2;
    }

    message Response {
        required int32 retvalue = 1;
    }
}
```

```
namespace libhidx::buffer {
class DetachKernelDriver {
    class Request {
        uint64_t handle();
        void set_handle(uint64_t&);
        int32_t interfacenumber();
        void set_interfacenumber(int32_t&);
    }
    class Response {
        int32_t retvalue;
        void set_retvalue(int32_t&);
    }
}
}
```

Obrázek 6.4: Ukázka překladu jazyka Protobuf (vlevo) do C++ (vpravo)

Protobuf umožňuje jako pole zprávy uvést i jinou zprávu. Tímto způsobem se velmi dobře kódují vnořené struktury, které se v libusb používají často.

Nakonec je třeba ještě upozornit, že při použití Protobufu je nutné ke kódu přilinkovat i knihovnu se stejným jménem. Tento proces byl však triviální, neboť CMake nabízí vestavěnou podporu jak pro kompilaci Protobuf souborů do C++, tak pro linkování knihovny Protobuf.

Před samotným odesláním zprávy se před zprávu zakódovanou pomocí Protobufu připojí ještě hlavička. Ta má pevnou velikost 11 bajtů a je složena z velikosti zprávy (8 bajtů) a číselného identifikátoru zprávy (3 bajty). Z důvodu snazšího ladění jsou oba tyto údaje kódovány pomocí ASCII. Velikost zprávy je počítána ze součtu velikostí samotné zprávy a číselného identifikátoru. Schématicky je formát zprávy naznačen na obrázku 6.5.

8 bajtů	3 bajty	n bajtů
Velikost zprávy	**Identifikátor zprávy**	**Zpráva kódovaná pomocí Protobufu**

Obrázek 6.5: Formát zprávy zasílané mezi klientem a serverem

Předávání zpráv mezi serverem a klientem

K předávání zpráv se používá unixová schránka (*unix domain socket*) nastavená na režim `SOCK_STREAM`. Toto nastavení můžeme považovat za obdobu TCP protokolu, ale pouze pro přenosy v rámci jednoho počítače[12]. K práci se schránkami využívám knihovnu Asio[10].

Použité řešení hodnotím velmi kladně díky jeho flexibilitě. Prvně je možné velmi snadno upravit kód Libhidx tak, aby bylo možné komunikovat i pomocí protokolu TCP přes běžné počítačové sítě. To je možné díky tomu, že knihovna Asio nabízí stejné rozhraní pro Unixové a TCP schránky[13]. V této práci jsem se ovšem rozhodl spojení pomocí TCP neimplementovat, protože bych zároveň s tím musel vyřešit i bezpečnostní otázky. Komunikace přes veřejné sítě by si totiž vyžádala implementaci šifrovaného přenosu a autentizace uživatele, což by výrazně zvýšilo složitost projektu.

Druhou důležitou vlastností pro flexibilitu mnou navrženého protokolu je fakt, že umožňuje bezproblémovou komunikaci napříč různými platformami. Hlavička je kódovaná čistě v ASCII, které je přítomno ve stejné podobě na většině platforem. Datová část zprávy je pak kódovaná a dekódovaná pomocí Protobufu, který také garantuje stejnou podobu zprávy na všech platformách. Díky tomu je možné bez problému zajistit komunikaci mezi různými platformami – například 32bitové s 64bitovými nebo little-endian s big-endian.

6.3.3 Klientská část

Klientská část je tvořená aplikačním rozhraním v jazyce C++. Tato část je během sestavování přilinkována ke zdrojovým kódům aplikace Hidviz. Plní 2 hlavní funkce: Poskytuje aplikační rozhraní s poměrně vysokou míru abstrakce pro komunikaci s USB zařízeními a zpracovává přijatá data. Zpracování deskriptoru je implementováno pomocí zdrojového kódu, který byl převzat z linuxového jádra a upraven z jazyka C do jazyka C++.

[10] Nikoli z ní vycházející knihovnu Boost.Asio. Knihovna Asio je v projektu nastavená tak, aby nebyla závislá na žádné části projektu Boost.
Web projektu Asio – https://think-async.com/

Kapitola 7

Hidviz jako svobodný software

Myšlenka této práce původně vznikla na popud zadání[1] zveřejněného zaměstnancem společnosti Red Hat Ing. Jaroslavem Škarvadou, PhD. Jedním z požadavků tohoto zadání bylo, aby vytvořený nástroj byl zveřejněn pod svobodnou licencí.

7.1 Verzování

Pro verzování Hidvizu je použit verzovací systém Git. Veřejný repozitář projektu[2], odkud si může kdokoliv projekt stáhnout a následně sestavit a používat, je umístěn na webové službě GitHub. Zde také mohou uživatelé hlásit chyby, případně navrhovat nové funkce aplikace.

7.2 Převzaté části kódu

V kódu aplikace Hidviz i knihovny Libhidx je převzato několik částí z jiných projektů šířených pod svobodnou licencí. Soubory obsahující takový kód jsou vypsané v tabulce 7.1.

Soubor	Zdroj	Licence
`libhidx/src/Parser.cc`	Linux[3]	GPL 2+
`hidviz/src/FluidLayout.*`	Qt dokumentace[4]	BSD
`libhidx/src/Usages.cc`	GitHub[5]	Apache License 2

Tabulka 7.1: Tabulka převzatých částí kódu

7.3 Licencování

Hidviz je šířen pod svobodnou licencí GNU General Public License verze 3. Tato licence byla zvolena kvůli její všeobecné popularitě a také kvůli tomu, že je kompatibilní se všemi převzatými částmi kódu.

[1] URL webu obsahující originální zadání – https://diplomky.redhat.com/thesis/show/434/usb-hid-monitor

[2] URL veřejného repozitáře – https://github.com/ondrejbudai/hidviz

[3] https://www.kernel.org/

[4] http://doc.qt.io/qt-5/qtwidgets-layouts-flowlayout-example.html

[5] https://github.com/smokris/usb-hid-usage

7.4 Průběžná integrace

Repozitář projektu je spojen se 2 službami, které poskytují služby průběžné integrace (*continuos integration*). Tyto služby po každém novém sestavení (*commit*) projektu stáhnou veškeré zdrojové kódy a zkusí je sestavit. V případě, že proces sestavení selže, informují mne o tom e-mailovou zprávou.

Použité služby se jmenují Travis CI[6] a AppVeyor[7]. Obě nabízejí své služby pro projekty se svobodnou licencí zdarma. První jmenovaná testuje sestavení na operačním systému GNU/Linux. Konkrétně se testuje sestavování pomocí dvou nejpoužívanějších kompilátorů, tedy GCC[8] a Clang[9], a to v jejich několika různých verzích. Dohromady se tak testuje 7 variant. Druhá služba slouží k testování sestavení na platformě Windows. Na Windows se kompiluje pomocí nástroje MSYS2[10], který používá kompilátor MinGW-w64[11] založený na GCC.

Průběžná integrace je velmi výhodným nástrojem, který pomáhá odhalovat chyby v projektu. Pravidelné testování na více platformách mi poskytlo velký komfort ve chvílích, kdy jsem na systému GNU/Linux vytvořil změnu v kódu, která učinila kód na platformě Windows nesestavitelným. V budoucnu je možné do průběžné integrace zařadit i spouštění automatických testů aplikace, které ještě více podpoří kvalitu projektu. Bylo by také možné nastavit, aby se sestavené binární soubory odesílaly na nějaký veřejný server a byly tím okamžitě dostupné zájemcům o testování aktuálních verzí Hidvizu.

7.5 Soubor README

Pro větší komfort zájemců o projekt je ke zdrojovým souborům přibalen soubor `README.md`, který stručně informuje o funkcích programu, způsobu, jak ho sestavit a nainstalovat, a jeho licenci. V souboru je také umístěn snímek obrazovky Hidvizu a aktuální stav obou služeb průběžné integrace. Služba GitHub tento soubor automaticky zobrazuje na domovské stránce projektu, takže každý návštěvník dostane velmi rychle veškeré informace, které o projektu potřebuje.

7.6 Balíčky v linuxových distribucích

Po dokončení vývoje byly pro Hidviz vytvořeny 2 balíčky pro snazší instalaci v prostředí linuxových distribucí. První balíček, určený pro distribuci Fedora, vytvořil konzultant této práce Jaroslav Škarvada. V době odevzdání práce se tento balíček nacházel ve stádiu recenzování vývojáři této distribuce. Druhý balíček pro tento projekt vytvořil s mojí pomocí Clayton Craft pro distribuci Arch Linux. Tento balíček je již přístupný všem uživatelům Arch Linuxu a distribucím z něj odvozených v komunitně spravovaném repozitáři AUR[12].

[6] Web služby Travis CI – https://travis-ci.org/
[7] Web služby AppVeyor – https://www.appveyor.com/
[8] Web kompilátoru GCC – https://gcc.gnu.org/
[9] Web kompilátoru Clang – https://clang.llvm.org/
[10] Web projektu MSYS2 – http://www.msys2.org/
[11] Web projektu MinGW-w64 – https://mingw-w64.org/
[12] Do repozitáře AUR může přidat svůj balíček kdokoliv. Populární balíčky jsou z něj později přesouvány do hlavního repozitáře Arch Linuxu.

Kapitola 8

Závěr

V mé bakalářské práci se mi podařilo navrhnout funkční USB HID monitor, který umožňuje pohodlně pracovat s USB periferními zařízeními třídy HID. Vytvořená aplikace Hidviz umožňuje zobrazit obsah HID popisovače, zpracovat ho a následně využít k vizualizaci stavu zařízení. Nástroj dovoluje i komunikaci opačným směrem – umožňuje tedy i změnu stavu zařízení. Hidviz je multiplatformní nástroj, který podporuje operační systémy Linux a Windows, a je dostupný pod svobodnou licencí pro širokou veřejnost.

Abych uživatelům v co největší míře usnadnil použití aplikace, připravil jsem balíčky pro 2 linuxové distribuce a operační systém Windows. Pro ty, kteří si chtějí aplikaci sami sestavit, jsem připravil soubor README s kompletními informacemi a nastavil jsem průběžnou integraci, která zajišťuje pravidelné testování sestavitelnosti projektu.

Do budoucna lze aplikaci rozvíjet mnoha směry. V mé práci nejsou pokryty některé rozšiřující HID specifikace. Hidviz by se tedy dal například rozšířit o podporu silové odezvy (*force feedback*), se kterou umí pracovat mnoho herních ovladačů. Dále by se aplikace mohla rozšířit o funkci záznamu komunikace HID zařízení. Uživatel by tak mohl jednou nasnímat sekvenci komunikace a následně ji podrobně rozebírat, aniž by už přitom musel mít zařízení fyzicky dostupné. I pro knihovnu libhidx se nabízí velké možnosti rozšíření. Během implementace jsem zjistil, že neexistuje žádná udržovaná knihovna, která by podporovala vzdálenou komunikaci s USB zařízeními připojenými k jinému hostiteli. Libhidx aktuálně podporuje pouze podmnožinu funkcí pro práci s USB sběrnicí potřebnou pro práci s HID zařízeními. Jejím rozšířením o chybějící funkce by se tak knihovna Libhidx mohla stát univerzálním řešením pro tento problém.

Věřím, že má bakalářská práce splnila všechny body zadaní, a aplikace vzniklá v jejím rámci má potenciál stát se běžně užívaným nástrojem v odvětví vývoje a reverzního inženýrství HID zařízení.

Literatura

1. AXELSON, Jan. *USB complete: everything you need to develop custom USB peripherals*. 2. vyd. Madison, WI: Lakeview Research, c2001. ISBN 978-0965081955.
2. HYDE, John. *USB design by example*. 2. vyd. Boston: Intel Press, 2001. ISBN 09-702-8465-9.
3. GOOK, Michael. *Hardwarová rozhraní: průvodce programátora*. 1. vyd. Brno: Computer Press, 2006. ISBN 80-251-1019-2.
4. *Universal Serial Bus Specification 2.0*. USB Implementer's Forum, 2000.
5. *Device Class Definition for Human Interface Devices (HID) 1.11*. USB Implementer's Forum, 2001.
6. *HID Usage Tables 1.12*. USB Implementer's Forum, 2004.
7. *CMake Reference documentation* [online]. New York: Kitware Inc., 2017 [cit. 2017-04-12]. Dostupné z: https://cmake.org/documentation/.
8. BLANCHETTE, Jasmin; SUMMERFIELD, Mark. *C++ GUI programming with Qt 4*. 2nd ed. Upper Saddle River: Prentice-Hall, 2008. ISBN 0132354160.
9. *GTK+ 3 Reference Manual* [online]. The GNOME Foundation, 2017 [cit. 2017-04-12]. Dostupné z: https://developer.gnome.org/gtk3/stable/.
10. *libusb-1.0 API Reference* [online] [cit. 2017-04-14]. Dostupné z: http://libusb.sourceforge.net/api-1.0/.
11. *Protocol Buffers Developer Guide* [online]. Google [cit. 2017-04-13]. Dostupné z: https://developers.google.com/protocol-buffers/docs/overview.
12. STEVENS, Richard W. *Unix network programming*. 3rd ed. Boston: Addison-Wesley, 2004. ISBN 0131411551.
13. KOHLHOFF, Christopher M. *asio C++ library documentation* [online]. 2015 [cit. 2017-04-13]. Dostupné z: http://think-async.com/Asio/asio-1.10.6/doc/.

Přílohy

Příloha A

Obsah přiloženého CD

K technické zprávě bakalářské práce je přiloženo CD obsahující zdrojové kódy aplikace a technické zprávy. Dále obsahuje sestavenou aplikaci Hidviz pro Windows a programovou dokumentaci. Soubory na CD jsou uvedeny v tabulce A.1.

Soubor	**Obsah**
hidviz-src.tar.gz	Zdrojové kódy aplikace Hidviz
hidviz-win.zip	Spustitelný soubor pro Windows
hidviz-docs.tar.gz	programová dokumentace
text.tar.gz	Zdrojové kódy technické zprávy
xbudai00.pdf	Technická zpráva ve formátu PDF
README	Návod ke zprovoznění aplikace Hidviz na Windows

Tabulka A.1: Struktura přiloženého CD

V případě nečitelnosti CD je možné jeho obsah stáhnout na následujícím odkazu:
https://stud.fit.vutbr.cz/~xbudai00/bakalarskaprace.iso

Příloha B

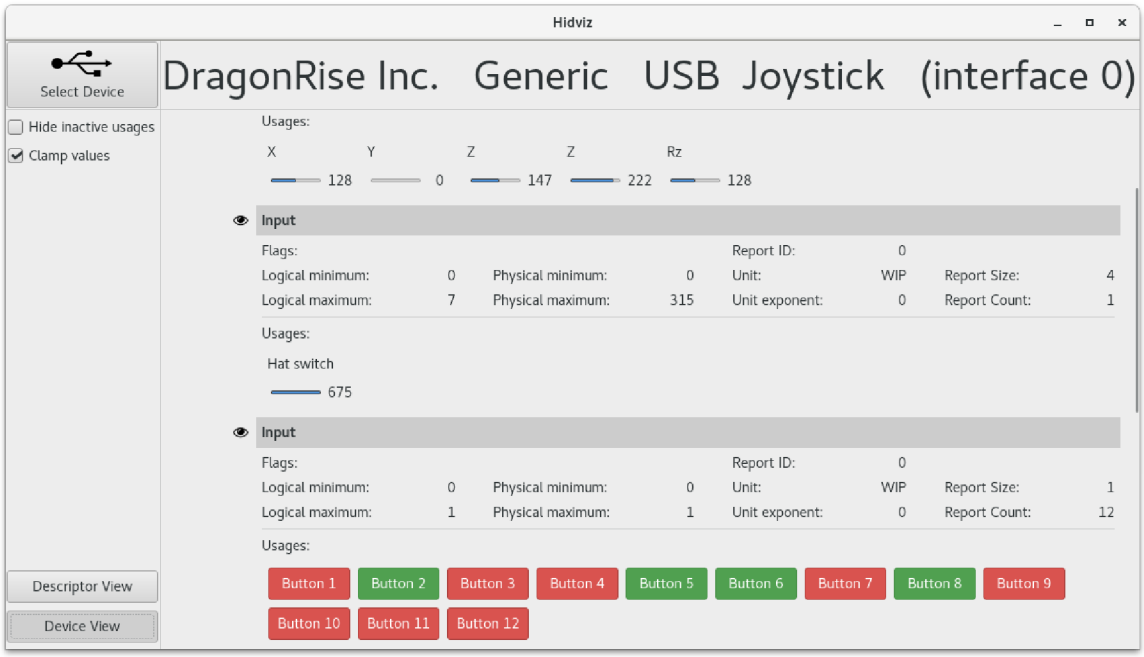
Demonstrace aplikace

V této části práce je demonstrováno použití aplikace na analýzu několika zařízení – herního ovladače, počítačové myši a klávesnice. Demonstrovaná zařízení ukazuje obrázek B.1. Myš a klávesnice jsou standardní, není tedy potřeba je více rozebírat. Herní ovladač se skládá z dvou hlavních analogových páček, čtyřsměrného kříže (*hat switch*), který se nachází na zařízení vlevo nahoře, a tlačítek, které se nachází v pravé části, nad páčkami a ze zadní strany ovladače.

Obrázek B.1: Demonstrovaná zařízení

B.1 Herní ovladač

Na obrázku B.2 je použit Hidviz k analýze herního ovladače. Prvním zobrazeným prvkem jsou sdružené osy analogových[1] páček. Zajímavé je, že ačkoliv páčky mají dohromady pouze 4 osy, zařízení má definovaných os 5, přičemž dvě z nich vždy obsahují stejná data. Jaká byla motivace výrobce pro toto řešení se mi bohužel nepodařilo zjistit. Stav všech prvků, které se mohou nacházet ve více než 2 stavech, vizualizuje Hidviz pomocí posuvníku, vedle kterého se zobrazuje číselná hodnota. Data o stavu ovládacího kříže jsou kódována na 4 bitech. 8 stavů je použito na kódování jednotlivých směrů (severní, severovýchodní, východní, atd.), 1 stav značí, že kříž není stisknut a ostatní jsou nevyužité. Posledním prvkem jsou tlačítka. Těch je 12 a mohou být stisknuta všechna nezávisle, a proto je jejich stav kódován na 12 bitech. Dvoustavové prvky vizualizuje Hidviz jako obdélníky měnící svou barvu podle stavu.

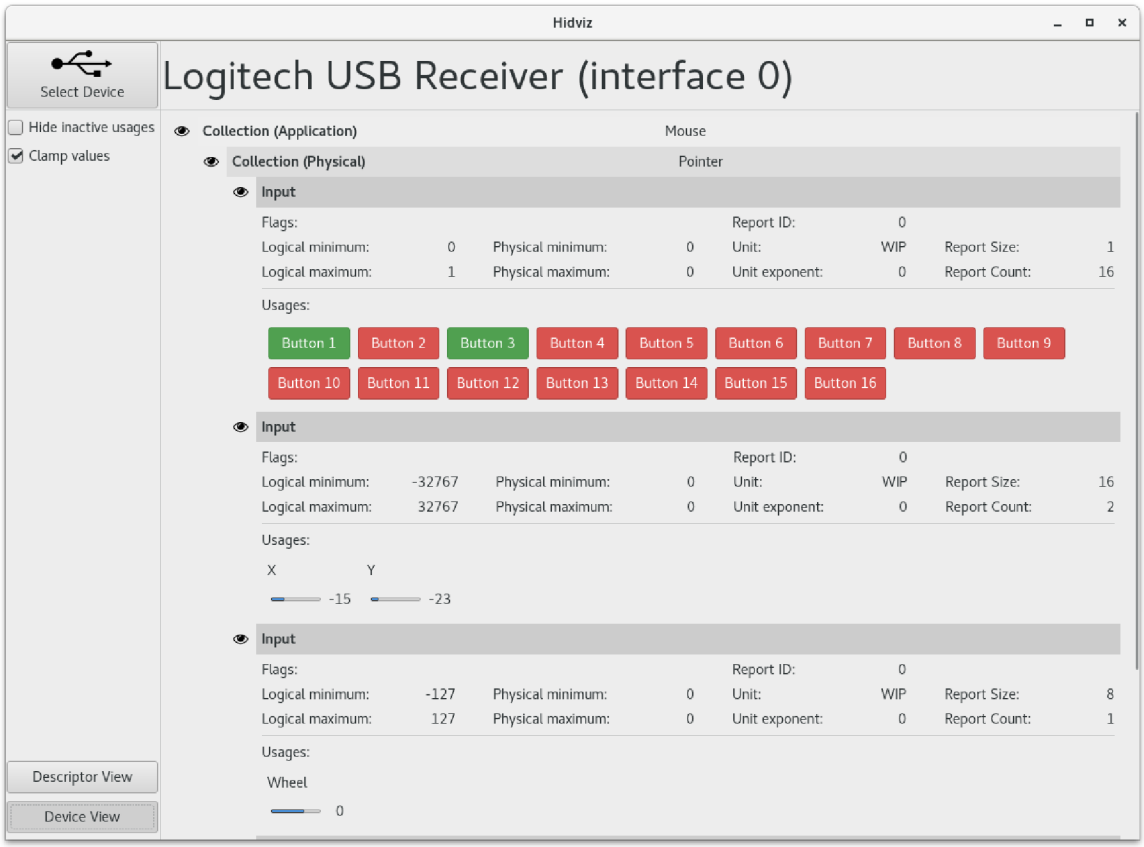


Obrázek B.2: Hidviz použitý pro analýzu joysticku

[1] Slovem analogové je myšleno, že prvek není pouze dvoustavový, ale může se nacházet i ve více stavech.

B.2 Počítačová myš

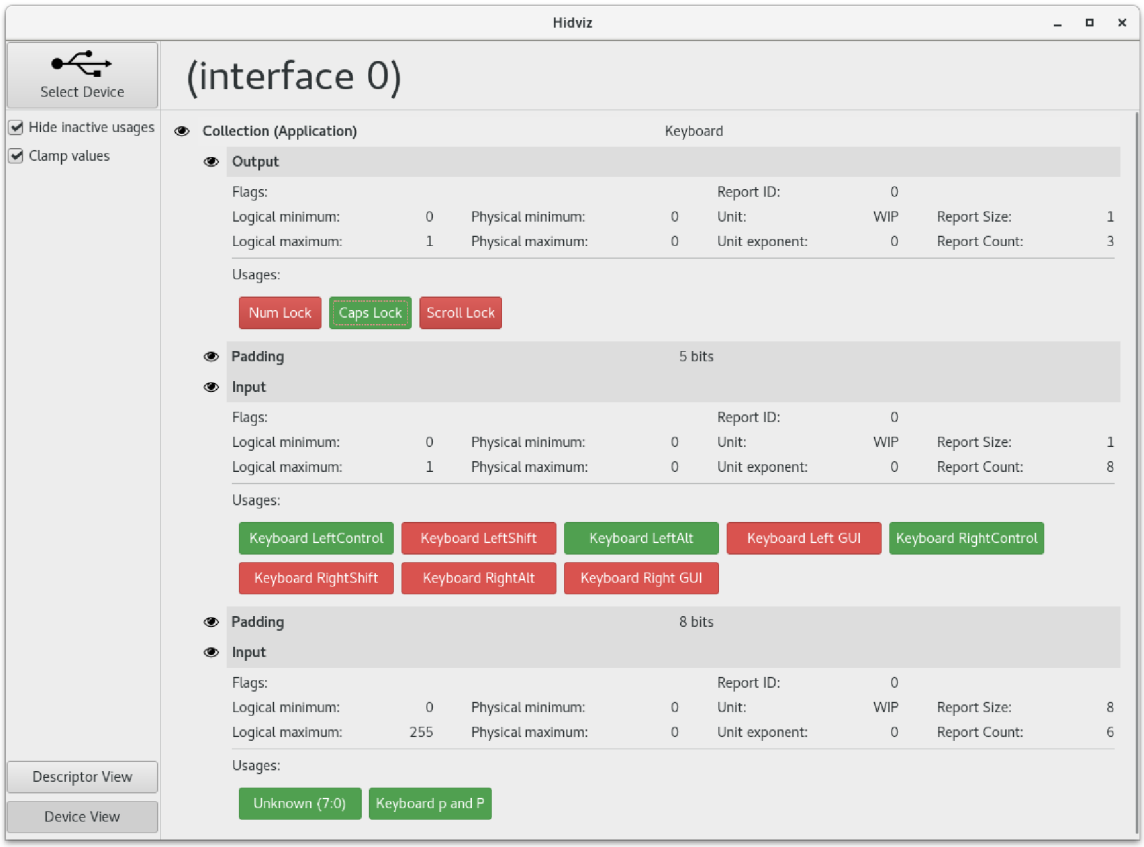
Obrázek B.3 ukazuje Hidviz během analýzy počítačové myši. Prvním prvkem myši jsou sdružená tlačítka. Ačkoliv analyzovaná myš obsahuje pouze 3 tlačítka, Hidviz zobrazuje tlačítek 16. To je dáno nejspíše tím, že myš je bezdrátová a přijímač je připraven i na komunikaci se složitějšími myšmi. Druhým prvkem jsou sdružené osy pohybu myši. Zajímavostí je, že obě osy jsou kódované na 16 bitech v rozsahu od -32767 do 32767. Ovšem během testování jsem zjistil, že hodnota při velmi prudkých pohybech zřídkakdy překročí hodnotu 1000. Z tohoto důvodu jsem implementoval nastavení na ořezaní hodnot (*clamp values*), které způsobí, že vizualizace pomocí posuvníku nepoužívá maxima deklarovaná zařízením, ale maxima zjištěná empiricky. Pokud tedy během komunikace dosahoval stav osy X maximálně hodnoty 800, je tato hodnota použita jako maximum pro vizualizaci. Posledním prvkem, který myš obsahuje je kolečko. Zde je opět velmi užitečné ořezávání hodnot, neboť rozsahu hodnot -127 až 127 při reálném provozu prakticky není možné dosáhnout.



Obrázek B.3: Hidviz použitý pro analýzu počítačové myši

B.3 Klávesnice

Konečně obrázek B.4 ilustruje analýzu klávesnice pomocí Hidvizu. Prvním prvkem jsou výstupy LED diod Num Lock, Caps Lock a Scroll Lock. Ty jsou vizualizovány jako tlačítka, kterými lze LED diody vypínat a zapínat. Druhý prvek obsahuje všechny pomocné klávesy – tedy Control, Alt, Shift a GUI[2]. Všechny klávesy jsou duplikované (každá má levou a pravou variantu) a lze je všechny stisknout naráz, neboť jsou kódovány na 8 bitech. Posledním prvkem jsou všechny ostatní klávesy. Zatímco u pomocných kláves je pro každou klávesu v HID zprávě rezervován jeden bit, u všech ostatních kláves se posílají pouze kódy stisknutých kláves s tím, že lze poslat maximálně takový počet stisknutých kláves, jaký je uveden v poli Report count. V případě, že uživatel stiskne více kláves, klávesnice odesílá místo kódů stisknutých kláves pouze kód chybový. V případě klávesnice je velmi užitečné nastavení Skrýt neaktivní použití (*Hide inactive usages*), které způsobí to, že se zobrazují pouze stisknuté klávesy namísto všech kláves, kterých může zařízení definovat stovky.



Obrázek B.4: Hidviz použitý pro analýzu klávesnice

[2]Klávesa GUI obsahuje na většině klávesnicí logo operačního systému Windows.