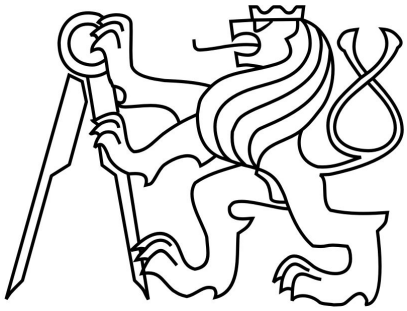
ČESKÉ VYSOKÉ ÚČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA TEORIE OBVODŮ



DIPLOMOVÁ PRÁCE

Aplikační podpora pro sběr dat při měření hemodynamických parametrů

Vypracoval: Bc. Filip Albert

Vedoucí práce: Ing. Jan Havlík, Ph.D.

Leden 2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra teorie obvodů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Filip A l b e r t

Studijní program: Biomedicínské inženýrství a informatika

Obor: Biomedicínské inženýrství

Název tématu: Aplikační podpora pro sběr dat při měření hemodynamických parametrů

Pokyny pro vypracování:

1. Seznamte se s problematikou měření hemodynamických parametrů a kyslíkové saturace.
2. Seznamte se s problematikou komunikace po sběrnicí USB, zejména s rozhraním HID.
3. Navrhněte a implementujte aplikaci komunikující s vybraným měřicím zařízením (zařízení pro měření hemodynamických parametrů nebo pulsní oximetr).
4. Navrhněte a implementujte nutné změny firmware zvoleného měřicího zařízení.

Seznam odborné literatury:

[1] Webster, John G. Encyclopedia of Medical Devices and Instrumentation. New Jersey: John Wiley & Sons Ltd., 2006. 9780471732877.
[2] Rozman, Jiří. Elektronické přístroje v lékařství. Praha: Academia, 2006. 80-200-1308-3

Vedoucí diplomové práce: Ing. Jan Havlík, Ph.D.

Platnost zadání: do konce zimního semestru 2017/2018

L.S.

prof. Ing. Pavel Sovka, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 26. 9. 2016

Poděkování

Děkuji svému vedoucímu práce Ing. Janu Havlíkovi, Ph.D. za zajímavé téma a za velmi vstřícný přístup, kdykoli jsem o cokoli požádal.

Prohlášení autora práce

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

V Praze dne 9. ledna 2017 …….

podpis autora práce

Abstrakt

Práce seznamuje čtenáře s problematikou měření hemodynamických parametrů a kyslíkové saturace z hlediska fyziologického i technického. Dále popisuje komunikaci po sběrnici USB, zejména za využití třídy HID, a seznamuje čtenáře s programovacím jazykem C#, který byl využit v praktické části práce. Na teoretickou část navazuje podrobný rozbor použité knihovny pro komunikaci s HID zařízením, který rovněž slouží k snadnějšímu pochopení této knihovny pro její případné budoucí uživatele. Použití knihovny je demonstrováno na vypůjčeném zařízení pulsního oxymetru vytvořením GUI aplikace, pomocí které je možné data ze zařízení číst, vypočítat hodnotu kyslíkové saturace a tepové frekvence, zobrazit fotopletysmografickou křivku a data exportovat do formátu CSV pro případné další využití.

Klíčová slova: hemodynamika, kyslíková saturace, USB HID, C#, pulsní oxymetrie

Abstract

The thesis introduces theory of measuring hemodynamic parameters and oxygen saturation, both from the physiological and technical point of view. Also describes communication using USB, especially its class HID, and introduces basics of C# programming language, which was used in practical part of the thesis. Then the thesis describes in details structure of used HID library, which aims to help reader, who wants to use the library, to understand its structure. Usage of the library is further demonstrated on pulse oximeter device through written GUI application, which provides data reading, blood oxygen saturation and heart rate measurement, photoplethysmogram plotting and export to CSV data format for further use.

Keywords: hemodynamics, oxygen saturation, USB HID, C#, pulse oximetry

Obsah

1. Úvod

Cílem této diplomové práce je seznámit se s problematikou měření hemodynamických parametrů, komunikací po rozhraní USB HID a kombinací těchto znalostí napsat aplikaci, která komunikuje se skutečným zařízením pulsního oxymetru. K tomuto účelu byl využit pulsní oxymetr Ing. Jana Dvořáka popsaného v jeho bakalářské práci [1].

Práce je členěna do dvou hlavních kapitol Teorie a Program. V první kapitole Teorie jsou popsána témata, jejichž znalost je nutná k pochopení dané problematiky. Nejprve jsou popsány nejdůležitější hemodynamické parametry, kterými jsou krevní tlak a srdeční výdej, a dále kyslíková saturace, která představuje důležitou veličinu zejména během anestezie. Tyto parametry jsou popsány jak z hlediska fyziologického, tak i z hlediska jejich měření od historie až po jednotlivé měřící techniky.

Dále se teoretická část zaměřuje na sběrnici USB, zejména pak rozhraní HID, které použité zařízení používá ke komunikaci s počítačem. Nakonec se práce zaměřuje na stručné a ucelené zasvěcení do jazyka C#, jehož syntaxe je v příslušné textové části rovněž použita.

Druhá kapitola se zabývá praktickou komunikací po sběrnici USB mezi počítačem a vybraným pulsním oxymetrem. V úvodu je detailně popsána struktura a použití knihovny C# USB HID, která slouží ke komunikaci s HID zařízeními. Tato knihovna byla získána jako open-source bez jakékoli dokumentace a její popis si v této práci klade za cíl zjednodušit její pochopení pro případné budoucí uživatele.

Práce se dále zaměřuje na samotnou aplikaci, která byla napsána za účelem čtení dat ze zařízení a jejich dalšímu zpracování. Je zde popsána samotná komunikace s pulsním oxymetrem, výpočet parametru kyslíkové saturace, zobrazení fotopletysmografické křivky v reálném čase a následné získání srdeční frekvence. Kapitolu uzavírá popis prostředí GUI aplikace.

2. Teorie

V teoretickém úvodu jsou popsány metody měření hemodynamických parametrů a kyslíkové saturace a vysvětleny všechny hlavní termíny, do kterých popis programu přímo zasahuje, nebo které jsou nutné k pochopení souvislostí. Těmito termíny jsou zejména pulsní oxymetrie, komunikace po sběrnici USB a její třídě HID a struktura programovacího jazyka C#.

2.1. Měření hemodynamických parametrů

Oběhová soustava je důležitá orgánová soustava, která slouží k transportu živin, plynů, odpadních látek, hormonů a dalších z tkání nebo do tkání. Transportním mediem je krev. Krev je z fyzikálního hlediska vazká tekutina a hemodynamika je nauka o proudění krve v oběhové soustavě.

Hemodynamika je popsána celou řadou parametrů, které se dělí na tlakové, objemové a dynamické. Z těchto parametrů je možné jmenovat například krevní tlak, srdeční výdej, rezistivitu a další. Často se k těmto mírám uvádí i tzv. index, což je hodnota vztažená k ploše pacientova těla. Díky indexu je možné porovnávat hodnoty hemodynamických parametrů různě tělesně stavěných pacientů. [50]

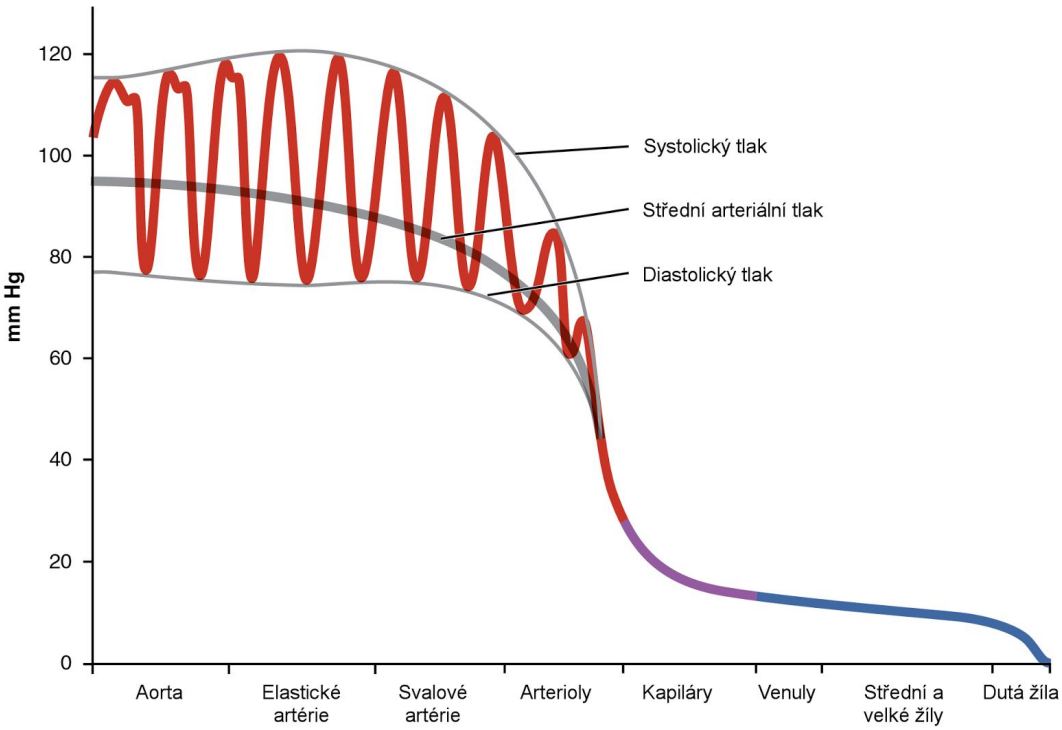
Anatomie oběhové soustavy člověka dělá z hemodynamiky poměrně složitou disciplínu. Každá část oběhové soustavy je typicky popsána různými hodnotami hemodynamických parametrů. Systém má určité parametry v aortě a artériích, jiné parametry v žilním systému, v plicnici a rovněž další parametry různé v obou komorách a síních. Typicky měříme pouze dané parametry v dané části systému, nicméně při specializovaných vyšetřeních je nutné znát hemodynamické parametry konkrétní vyšetřované oblasti. [50]

2.2. Krevní tlak a jeho měření

Krevní tlak je tlak, kterým cirkulující krev působí na stěnu cévního systému. Krevním tlakem je ve většině případů chápán tlak arteriální. Je dán dvěmi hodnotami - tlakem systolickým a tlakem diastolickým. Systolický tlak představuje maximální hodnotu tlaku během jedné srdeční periody. Diastolický tlak naopak představuje minimální tlak mezi dvěmi srdečními periodami. Krevní tlak je měřen v milimetrech rtuti (mmHg) měřícího válce, vůči vlivu, kterým působí tlak atmosferický. Za normální

hodnotu se nejčastěji považuje 120/80 mmHg[1], kde první hodnota představuje tlak systolický (SBP) a druhá hodnota představuje tlak diastolický (DBP). Původně se tlak měřil rtuťovým manometrem v kombinaci s manžetou a fonendoskopem. Dnes je možné krevní tlak měřit různými metodami neinvazivně i invazivně, za pomoci různých typů přístojů.

Jak již bylo zmíněno v úvodu do hemodynamiky, měřené hemodynamické parametry je vždy nutné zmiňovat v souvislosti s místem, ve kterém jsou měřeny. Těmito místy se myslí pravá a levá síň, pravá a levá komora, plicní artérie a aorta. O tlaku v aortě mluvíme jako o systémovém krevním tlaku a pokud nebude zmíněno jinak, bude krevní tlak v této práci uvažován jako tlak systémový. Na obrázku 1 je uveden graf typického vývoje krevního tlaku od aorty až po dutou žílu.



Obrázek 1: Typický průběh krevního tlaku od aorty až po dutou žílu. [41]

Je nutné si uvědomit, že krev teče po tlakovém gradientu z místa s vyšším tlakem do místa s nižším tlakem. Hodnota krevního tlaku tedy od aorty směrem k duté žíle klesá. Je ovlivněna několika faktory, srdečním výdejem, poddajností cév (neboli cévním odporem), objemem krve a její viskozitou a délkou a průměrem cévy. Poddajnost cév kompenzuje krevní tlak zvětšením průměru cévy. Je-li naopak céva

[1] Je nutné si ale uvědomit, že veškeré hemodynamické parametry jsou úzce svázané s měřeným subjektem a normalita by spíše měla být vyjádřena intervalem než hodnotou.

málo poddajná, například u starých lidí, tlak není v daných situacích dostatečně kompenzován. [41]

2.2.1. Střední arteriální tlak

K hodnotě systolického a diastolického tlaku se dále uvadí střední arteriální tlak (MAP - *mean arterial pressure*). Střední arteriální tlak představuje průměrný tlak za jednu srdeční periodu a je dán srdečním výdejem (CO - *cardiac output*), cévním odporem (SVR - *systemic vascular resistance*) a centrálním žilním tlakem (CVP - *central venous pressure*). Jedná se o velmi důležitý parametr, protože jeho hodnota přímo ovlivňuje hodnotu orgánové perfúze. Hodnota středního arteriálního tlaku se vypočítá

$$MAP = (CO \cdot SVR) + CVP.$$

V praxi se ale hodnota centrálního žilního tlaku z důvodu jeho malé hodnoty zanedbává.[2] Podstatně jednodušší metodou výpočtu středního arteriálního tlaku je výpočet ze znalosti tlaku systolického a tlaku diastolického, který je dán

$$MAP = \frac{SBP + 2 \cdot DBP}{3}.$$

Průměrování přes jednu hodnotu systolického tlaku a dvě hodnoty diastolického tlaku je proto, že diastola, neboli srdeční relaxace, trvá přibližně dvakrát déle než systola, neboli srdeční kontrakce. Normální hodnota středního arteriálního tlaku je přibližně 90 mmHg. [41, 51]

2.2.2. Měření krevního tlaku

Krevní tlak byl poprvé popsán a měřen anglickým fyziologem Stephenem Halesem v roce 1733. Experiment, který Hales provedl, spočíval ve svázání ležící klisny a zasunutí dlouhé duté trubice do její artérie. Hales poté v trubici pozoroval výšku sloupce krve, která se měnila během každého srdečního cyklu. První použití rťutového manometru pro měření krevního tlaku však provedl až v roce 1828 francouzký fyziolog Poiseuille tak, že spojil rtuťový manometr s kanilou naplněnou uhličitanem draselným, činidlem proti srážlivosti krve, a kanilu vsunul přímo do

[2] Průměrná hodnota centrálního žilního tlaku se udává 4,5 mmHg, což je přibližně dvacetkrát méně než průměrná hodnota středního arteriálního tlaku.

artérie pokusného zvířete. Poiseuille mimo jiné zjistil, že pulzatilní tlak je možné měřit i v menších artériích. Poiseuilleho vynález pomohl Carlu Ludwigovi v roce 1847 sestrojit první kymograf, zařízení na záznam krevního tlaku a později dalších fyziologických parametrů. V roce 1855 potom přislo první neinvazivní měření provedené německým fyziologem Karlem von Vierordtem, které spočívalo v měření protitlaku, kterým působí krevní tlak na škrtidlo. Měření nebylo vzhledem k přesnosti úspěšně, nicméně jeho přesnost byla později vylepšena jinými vědci. V 70. letech 19. století byla poprvé použita gumová manžeta naplněná vodou spojená s rtuťovým manometrem a v roce 1889 byla voda nahrazena vzduchem. Konečně v roce 1896 byla italským doktorem Scipionem Riva-Roccim vynalezena technika, která se používá dodnes. Technika spočívá v utažení manžety okolo celého obvodu horní končetiny a zvyšování tlaku v manžetě tak dlouho, dokud neustane změna ve výšce rťuti měřícího manometru. Touto metodou je možné měřit systolický tlak. Měření diastolického tlaku pomocí Riva-Rocciovým přístrojem představil v roce 1905 ruský chirurg Nikolaj Sergejevič Korotkov, který za pomoci stetoskopu detekoval srdeční ozvy, způsobené krví navracející se zpět do artérie. Tímto byla představena auskultační metoda, která je stále dnes hojně používanou metodou pro měření krevního tlaku. [42]

Krevní tlak je úzce svázán se srdečním pulzem. Frekvence pulzu nám dává informaci o srdečním tepu a síla pulzu nám dává informaci o krevním tlaku. Pulz se typicky měří palpačně v místech, kde artérie vede v blízkosti tělesného povrchu. Těmito místy jsou karotida a radiální artérie v zápěstí. Nicméně touto metodou není možné krevní tlak změřit a je proto nutné použít citlivějších a přesnějších přístrojů. [41]

Klasickým přístrojem v měření krevního tlaku je rtuťový tonometr. Je považován za nejpřesnější přístroj k neinvazivnímu měření. Důvodem je použití rtuti, která v čase nemění své fyzikální vlastnosti. Nicméně vzhledem k její toxicitě se od jejího používání stále více upouští a v některých zemích již bylo její použití zakázáno úplně. V rámci Evropské unie platí od roku 2009 zákaz uvádění nových rtuťových měřicích přístrojů na trh. Regulace se již používaných přístrojů netýká.[3]

Jednu z náhrad rtuťového tonometru představuje aneroidní tonometr uzavřený pružnou membránou. Deformace membrány se převádí na otáčivý pohyb zobrazovacího prvku. Nevýhodou těchto zařízení je náchylnost k různým fyzikálním jevům jako je teplota, mechanické otřesy, stárnutí materiálu a další. [52]

[3] Dle směrnice Evropského parlamentu a Rady 2007/51/ES z 25. září 2007.

Velkým přínosem v měření krevního tlaku bylo vyvinutí elektronického sfygmomanometru, který funguje na principu oscilometrické metody, popsané dále v textu. Rovněž využívá tlakování manžety okolo paže ve výšce srdce, nicméně meří oscilometrické pulzace, ze kterých stanovuje střední arteriální tlak a systolický a diastolický tlak dopočítává. I přes vyšší nepřesnost oproti klasickému rtuťovému tonometru, se s výhodou používá hlavně díky jednoduchosti použití a lepší odolnosti vůči šumu. Nicméně vzhledem k nutnosti přesného výpočtu systolického a diastolického tlaku, je přístoj velmi nepřesný při nestandardním chování srdce, jako je například srdeční aritmie, arterioskleróza a další. [52]

Ke stanovení krevního tlaku se rovněž využívají invazivní metody. Díky invazivitě je možné kromě arteriálního tlaku měřit i centrální žilní tlak, či tlak v plicnici. Pro stanovení arteriálního tlaku se využívá arteriálního katétru s tlakovým senzorem, který se zavádí buď do brachiální artérie nebo femorální artérie. Metoda se využívá zejména na jednotkách intenzivní péče a představuje pro pacienta určitá rizika jako je infekce, trombóza, tepenné krvácení a další. [52]

2.2.2.1. Auskultační metoda

Auskultační metoda představuje klasickou neinvazivní metodu měření krevního tlaku a byla zmíněna v historickém přehledu v úvodu do měření krevního tlaku. Auskultace znamená vyšetření poslechem a přesně o tom tato metoda je. Využívá sfygmomanometru, přístroje k měření tlaku krve, a stetoskopu. Technika spočívá v nasazení manžety na pacientovu paži do výšky úrovně srdce a napuštění manžety vzduchem tak, aby tlak, kterým manžeta na pacientovu paži působí, dočasně uzavřel tok v pacientově artérii. Poté lékař přiloží stetoskop do předloketní jamky a upouští vzduch v manžetě, dokud nezačne slyšet první z korotkovových ozev. Korotkovovy ozvy vznikají při obnovení toku krve artérií, kdy je ale artérie tlakem v manžetě deformovaná, což vytváří turbulentní proudění za vzniku zmíněných ozev. V této fázi měření lékař odečte hodnotu měřeného tlaku a dostává systolický tlak. Poté pokračuje v upouštění tlaku v manžetě, dokud neuslyší poslední z korotkovových ozev. Tím dostává, po odečtení hodnoty tlaku manometru, tlak diastolický. V tomto případě již manžeta artérii nedeformuje a v artérii je obnoveno původní laminární proudění.

2.2.2.2. Oscilometrická metoda

Oscilometrická metoda je další z metod neinvazivního měření krevního tlaku. Velmi se podobá metodě auskultační, avšak nevyžaduje použití fonendoskopu. Využívá se zde elektronického přístroje pro vyhodnocení pulzací a manžety. Manžeta je nasazena na pacientovu paži v úrovni srdce a natlakována tak, aby plně zaškrtila artérii, kterou obepíná. Tlak v manžetě je následně uvolňován a detekční zařízení zaznamenává vzniklé pulzace. Hodnota tlaku pro maximum amplitudy představuje střední arteriální tlak, ze kterého se poté vypočítají empiricky zjištěné hodnoty tlaku systolického a diastolického. Systolický tlak odpovídá tlaku v manžetě při amlitudě pulzací v 55% maxima měřeného před dosažením maxima a diastolický tlak odpovídá tlaku v manžetě při 85% maxima měřeného po dosažení maxima.

Tato metoda je náchylná na volbu parametrů zmíněných výše. Vzhledem k velké variabilitě různých fyziologických parametrů v populaci, zejména způsobených srdečními onemocněními, může být oscilometrická metoda nepřesná. Nicméně její jednoduchá obsluha z ní dělá vhodnou metodu na orientační měření krevního tlaku. [52]

2.3. Srdeční výdej a jeho měření

Srdeční výdej, neboli minutový objem srdeční, je množství krve, kterou srdeční komora přečerpá za jednu minutu. Je možné ho zapsat vztahem jako

$$CO = SV \cdot HR,$$

kde SV představuje systolický objem a HR představuje tepovou frekvenci. Jednotkou se nejčastěji udává litr za minutu. Srdeční výdej je tedy ovlivněn srdečním tepem anebo systolickým objemem. Je-li jedna z těchto veličin zvýšena, zvýší se i srdeční výdej. Hodnota srdečního výdeje je regulována tělem zejména přes autonomní nervovou soustavu, pozitivně ovlivněna sympatikem, a dále hormonálně. U zdravého dospělého člověka v klidu se za normální považují hodnoty 4 až 6,5 l/min. Pro srdeční index potom normální hodnota vychází přibližně 2,5 l/min na m^2. Nicméně hodnota srdečního výdeje přímo souvisí s aktuální potřebou těla dopravit do tkání kyslík, a tudíž není možné normální hodnotu přesně stanovit. [43]

2.3.1. Měření srdečního výdeje

Metod měření srdečního výdeje je celá řada. Používají se zde metody invazivní, jako jsou diluční metody, i metody neinvazivní, jako je například echokardiografie. Dříve hojně používané invazivní metody jsou díky vývoji přesnějších měřících technologií stále častěji nahrazovány metodami neinvazivními. Metody pro výpočet srdečního výdeje se velmi liší cenou, přesností, pacientovým pohodlím a bezpečností.

2.3.1.1. Fickova metoda

První výpočet stanovení srdečního výdeje byl představen již v roce 1887. Byl stanoven na základě znalosti okysličení arteriální krve (CaO_2), okysličení smíšené žilní krve (CvO_2) a spotřebě kyslíku (VO_2) jako

$$CO = \frac{VO_2}{CaO_2 - CvO_2}.$$

Ač u měření srdečního výdeje neexistuje konkrétní metoda, která by byla všeobecně považována za zlatý standard, jako je tomu například u měření krevního tlaku, je Fickova metoda mnohými považována za velmi přesnou a často se bere jako metoda referenční. Nicméně její použití je složité a nepraktické. [44, 45, 46]

2.3.1.2. Dopplerovské metody

Ke stanovení srdečního výdeje je možné využít rovněž sonografii. Metod založených na ultrazvuku je více, nicméně jejich metodika se vzájemně příliš neliší. Princip je následující. Matematicky je možné vyjádřit Dopplerův jev jako

$$\Delta f = \frac{2 \cdot f_t \cdot v}{c} \cdot \cos \Theta,$$

kde Δf je měřená změna frekvence, f_t je frekvence vyslaná vysílačem, v je relativní rychlost mezi vysílačem a měřeným prvkem, tedy krevními komponentami, c je rychlost zvuku v prostředí, tedy ve tkáni, a cosinus vyjadřuje úhel odrazu. Jelikož jsou všechny hodnoty ve vztahu kromě rychlosti známé, je možné rychlost ze vztahu vyjádřit

$$v = \frac{\Delta f \cdot c}{2 \cdot f_t \cdot \cos \Theta}.$$

Provedením integrace maxim rychlostí přes jeden srdeční cyklus dostaneme délku, která představuje výšku sloupce krve cévou vyvolaným tímto srdečním cyklem. Změřením poloměru cévy je potom možné vypočítat objem krve, tj. systolický objem. Ze znalosti systolického objemu a tepové frekvence je poté možné dopočítat srdeční výdej. [45]

2.3.1.3. Diluční metody

Diluční metody jsou založeny na principu injekce indikátoru do krevního řečiště a stanovení koncentrace tohoto indikátoru ve zvoleném místě krevního řečiště. Metoda je založena na faktu, že se zvyšujícím se srdečním výdajem roste rychlost krve v řečišti a tedy i indikátoru měřeným místem. Indikátorem se v tomto případě rozumí barvivo, radioizotop, látka dané teploty, a další. Různých druhů indikátorů je celá řada, nicméně je nezbytné, aby byly všechny tělem dobře snášeny a odbourány. Výpočet srdečního výdeje pro všechny metody se dá zobecnit následujícím vztahem

$$CO = \frac{n_{ind}}{\int_0^\infty c_{ind} \mathrm{d}t},$$

kde CO je srdeční výdej, n_{ind} je celkové látkové množství indikátoru a c_{ind} je koncentrace indikátoru v místě měření. [43]

Použití vstřikovaného indikátoru pro měření srdečního výdeje bylo poprvé popsáno již v roce 1921. Nicméně až v roce 1932 bylo popsáno využití této metody v praxi. Jako indikátor byla použita indocyaninová zeleň. Nicméně nutnost odebírání krevních vzorků a ruční analýza dělali tuto metodu velmi nepraktickou a časově náročnou. [47]

Nejčastěji používanou diluční metodou je transkardiální termodiluční metoda, která byla představena v 70. letech 20. století. Jako indikátor používá zchlazený 5% fyziologický roztok, který vytváří teplotní rozdíl vzhledem k teplotě krevního řečiště. Teplotní rozdíl by měl dosahovat alespoň 10°C, nicméně často se používá roztok s teplotou nižší než 8°C. Objem indikátoru se volí v závislosti na váze pacienta, často od 10 do 15 mL. Měření probíhá tak, že se do pravé srdeční síně zavede proximální výstup pulmonárního katétru, přes který se vstříkne indikátor. V plicnici, do které se zavede část katétru s termistorem, se potom měří změna teploty krve. Převodním vztahem se z naměřené křivky vypočítá srdeční výdej, který je nepřímo úměrný

ploše pod danou křivkou. Nevýhodou této metody je její invazivita a případné komplikace s tím spojené, mezi které patří infekce, arytmie, tvorba trombů či poranění srdeční a plicní tkáně. [44, 45]

V 90. letech byla představena transpulmonární termodiluční metoda, která spolu s pulmonárním katétrem využívá přidaný arteriální katétr s termistorem. V případě této metody je nutné měřit teplotu vstřikovaného fyziologického roztoku. Srdeční výdej je dán vtahem

$$CO = \frac{(T_b - T_i) \cdot V_i \cdot K}{\int_0^\infty T_b \mathrm{d}t},$$

kde V_i je objem vstříknutého indikátoru, T_b je teplota krve, T_i je teplota indikátoru a K je konstanta. [45]

Mezi další diluční metody patří transpulmonární lithiová diluční metoda popsaná v roce 1993. Její výhodou je nižní invazivita oproti výše zmiňovaným dilučním metodám. Namísto pulmonárního katétru využívá centrální žilní katétr, pomocí kterého vstřikuje roztok chloridu lithného. Měření probíhá za použití arteriálního katétru s lithiovým detektorem, do kterého je peristaltickou pumpou rychlostí 4,5 ml/min vháněna krev. Ta generuje na senzoru napětí. Pomocí Nernstovi rovnice je z měřeného napětí možné následně vypočítat koncentraci lithia. Srdeční výdej se vypočítá

$$CO = \frac{60 \cdot n_{ind}}{\int_0^\infty n_{Li} \mathrm{d}t \cdot (1 - PCV)},$$

kde n_{ind} je molární množství vstřikovaného lithia za sekundu, konstanta 60 slouží k převodu na minuty, n_{Li} je měřená koncentrace lithia a PCV je procentuální zastoupení červených krvinek, ze kterého je dopočítáno procentuální zastoupení krevní plazmy, ve které je lithium rozptýleno. Výhodou použití lithia je jeho přirozená absence v krevním řečišti, což má za následek nižší šum při měření, a je možné ho použít ve velmi malé koncentraci. Dále je to zejména netoxicita a vysoká clearance. [46, 47]

U dilučních metod je nutné brát v úvahu jisté technické nároky na průběh měření. Nutná je správná pozice katétru, přesné množství vstřikovaného indikátoru, fáze dechového cyklu, pozice pacientova těla, a další. Měření se několikrát opakuje a výsledek je dán průměrem z naměřených hodnot. Chyba měření se u dilučních metod pohybuje okolo 10%. [44]

2.3.1.4. Ostatní metody stanovení srdečního výdeje

Kromě výše zmíněných metod existují další metody založené na odlišných principech. Například již v roce 1959 byla popsána metoda využívající elektrickou kardiografii. Její princip spočívá v průchodu střídavého proudu srdcem o velmi nízké amplitudě a měření impedance, která se v závislosti na objemu krve v srdci mění. Nevýhodou této metody je ale její nízká přesnost. Dále je možné zmínit využití magnetické rezonance, kde se využívá faktu, že rezonanční vlastnosti protonů v jádře se mění v závislosti na jejich rychlosti. Ač je metoda velmi přesná, pro své náklady se v praxi nepoužívá. [43, 45]

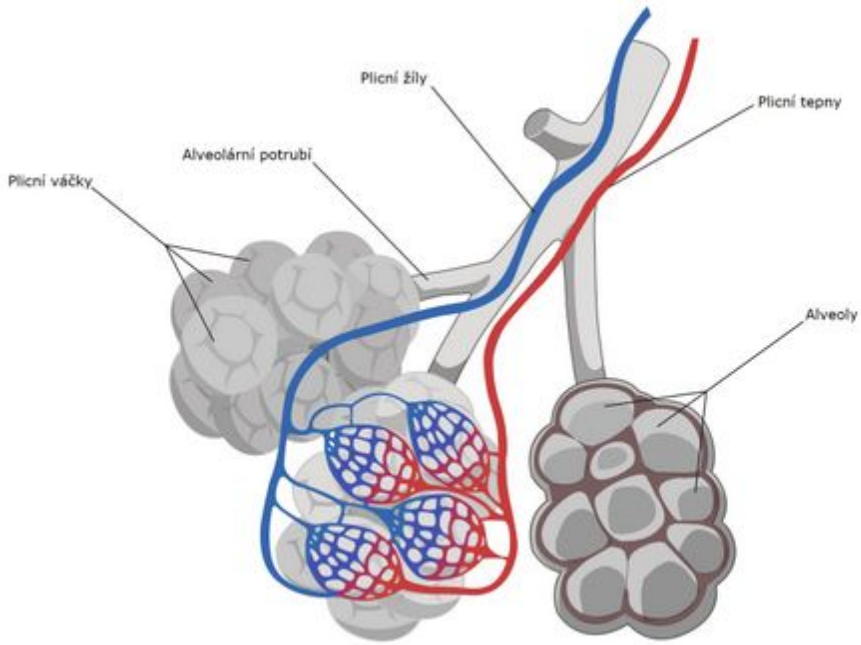
2.4. Fyziologie dýchání a kyslíková saturace

Stručná fyziologie nutná k pochopení pulsní oxymetrie je následující. Vdechovaný kyslík se v plicích váže na hemoglobin (Hb), pomocí kterého je transportován krevním řečištěm do tkání. V tkáních se účastní procesu zvaného buněčné dýchání, které buňce dodává energii. Pulsní oxymetrie měří kyslíkovou saturaci, tj. poměr okysličeného hemoglobinu vůči celkovému hemoglobinu. Pro správnou funkci buněčného dýchání je vyžadována vysoká hodnota kyslíkové saturace, typicky 95 - 100%.

Hemoglobin je metaloprotein obsažený v červených krvinkách všech obratlovců. Vyjma vody tvoří hemoglobin 96% hmotnosti červené krvinky a na gram své hmotnosti je schopen navázat 1,34 mL kyslíku, čímž zvyšuje množství transportovaného kyslíku vůči kyslíku rozpuštěného v krvi přibližně sedmdesátkrát. Jedna molekula hemoglobinu je schopna navázat až čtyři molekuly kyslíku, nicméně pokud není hemoglobin kyslíkem plně obsazen, je jeho náchylnost k navázání dalších molekul kyslíku velmi vysoká. Totéž platí i v případě uvolňování kyslíku z hemoglobinu. Pokud se kyslík na hemoglobin naváže, mluvíme o oxyhemoglobinu (O_2Hb). Kromě kyslíku může hemoglobin vázat i oxid uhličitý (CO_2), potom mluvíme o karboxyhemoglobinu (COHb), může být ve formě methemoglobinu (MetHb), který vzniká, je-li krev vystavena působení dusičnanů a dusitanů, či v méně běžných formách jako jsou sulfhemoglobin (SfHb) a karboxysulfhemoglobin (COSfHb). [6, 27, 35]

Hlavní funkcí hemoglobinu je transport kyslíku z kapilár plicních sklípků do zbytku těla. V plicních sklípcích (alveolech), znázorněných na obrázku 2, dochází k jevu zvanému difúze. Oxid uhličitý navázaný na hemoglobin je uvolněn a kyslík je

naopak na hemoglobin navázán. Při tomto procesu oxygenace hemoglobinu vzniká tzv. oxyhemoglobin (O_2Hb), který je krevním řečištěm transportován do tkání. Kyslík je v dané tkáni uvolněn, aby mohl proběhnout první děj buněčného dýchání - glykolýza. Buněčné dýchání představuje jeden z klíčových způsobů, kterým získává buňka energii. Uvolněná energie se v buňce používá k syntéze adenosintrifosfátu (ATP). ATP je důležitým zdrojem energie pro různé procesy uvnitř buňky. Mezi tyto procesy patří například biosyntéza, vnitrobuněčný a membránový transport, výroba proteinů či syntéza RNA. [27, 28]



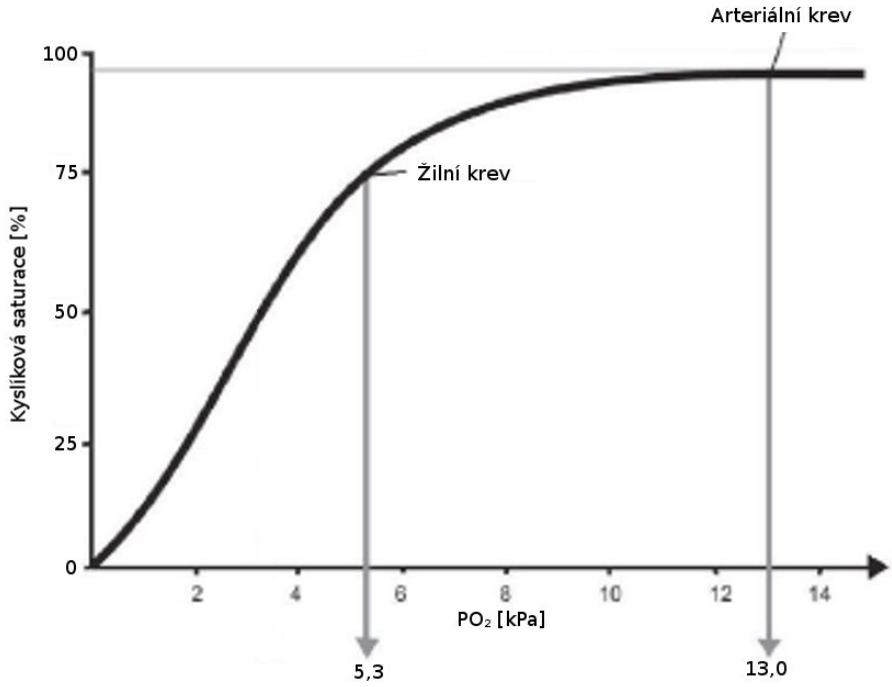
Obrázek 2: Difúze v plicních sklípcích [38]

Jak již bylo zmíněno výše, je schopen hemoglobin na sebe navázat i jiné plyny, například další dýchací plyn - oxid uhličitý. To je rovněž velmi užitečné při buněčném dýchání, protože jedním z odpadních produktů energetické přeměny v buňce je právě oxid uhličitý. Hemoglobin, ze kterého se po uspěšném transportu kyslíku stává deoxyhemoglobin, je takto schopen transportovat 20 - 25% celkového oxidu uhličitého, vytvořeného buněčným dýcháním. Zbylý oxid uhličitý je použit pro udržení acidobazické rovnováhy a malá část je rovněž jako kyslík rozpuštěna v krvi. Deoxyhemoglobin je následně oběhovou soustavou transportován zpět do plic, kde dochází k dalšímu procesu difúze, a celý děj se opakuje znovu. [27]

Na vázání, resp. uvolňování, kyslíku na, resp. z, hemoglobinu se podílí několik faktorů. Mezi tyto faktory patří pH, koncentrace CO_2, parciální tlak kyslíku a další. Ve zkratce se dá říci, že pokud se dostane oxyhemoglobin do místa s nízkým pH a velkou koncentrací oxidu uhličitého, jeho schopnost vázat kyslík je velmi malá a

naopak jeho schopnost kyslík uvolňovat je velká. Naopak v plicním sklípku, kde je hodnota pH vyšší a koncentrace oxidu uhličitého nižší, má hemoglobin velkou schopnost na sebe kyslík vázat. [27]

Parciální tlak kyslíku je úměrný míře saturace kyslíku v krvi a je dán tzv. hemoglobin-kyslíkovou disociační křivkou (viz obrázek 3). Na obrázku jsou znázorněny dva významné body, a to hodnota kyslíkové saturace pro arteriální krev a hodnota kyslíkové saturace pro žilní krev. Kyslíková saturace pro arteriální krev se za normálních podmínek, tj tělesné teploty 37°C a pH 7,4, pohybuje okolo 97% a kyslíková saturace pro žilní krev okolo 75%. V případě zvýšení teploty či snížení pH vnitřního prostředí lidského těla se tato křivka posouvá na ose parciálního tlaku kyslíku směrem doleva. V opačném případě se posouvá doprava. Její tvar představuje sigmoida a nikoli přímka, což je dáno výše zmíněnou náchylností k navázání či uvolnění všech pozic na hemoglobinu pro kyslíkovové molekuly. [35]



Obrázek 3: Hemoglobin-kyslíková disociační křivka [26]

2.5. Pulsní oxymetrie

Pulsní oxymetrie je neinvazivní metoda měření kyslíkové saturace (SO_2) bez nutnosti kalibrace. Kyslíková saturace je definovaná jako míra množství kyslíku krvi, založená na detekci hemoglobinu a deoxyhemoglobinu. V klasickém případě používá měření z prstu či ušního laloku, kde jde o měření tzv. periferální kyslíkové saturace (SpO_2). Nicméně ta se pro účely měření a za normálních podmínek

zásadně neliší od přesnější arteriální kyslíkové saturace (SaO_2) a je snažší ji měřit. Neinvazivita měření a relativně nízké nároky na přesnost jsou hlavními důvody použití SpO_2 jako měřené veličiny. Kromě pulsní oxymetrie jsou známé další dvě metody měření kyslíkové saturace - invazivní měření in vivo pomocí katetru s fibrooptickým vláknem a invazivní měření in vitro analýzou krevních vzorků. [28, 39]

2.5.1. Historie

Metoda měření kyslíkové saturace se původně prováděla měřením množství kyslíku z krevních vzorků. Nevýhoda této metody byla v první řadě její invazivita a v druhé řadě neposkytovala měření v reálném čase, což je z fyziologického hlediska vzhledem k potřebám velmi zásadní nedostatek. [37]

Za otce oxymetrie je považován vídeňský profesor Karl Matthes, který zároveň v roce 1935 zkonstruoval první souvisle měřící oxymetr. Zjistil také, že červené světlo není oxyhemoglobinem pohlcováno, kdežto redukovaný hemoglobin toto světlo pohlcuje. V roce 1949 Robert Brinkman a William Ziilstra začali měřit kyslíkovou saturaci ze zařízení umístěného na čele pomocí odraženého světla. Tím zjistili, že Beer-Lambertův zákon funguje na odraženém světle stejně jako při měření absorbce přímo. V roce 1960 byl Michaelem Polanyim vyvinut první fibrooptický oxymetr. V této době se rovněž začalo uvažovat o využití oxymetrie při anestezii. [39]

Metoda pulsní oxymetrie byla náhodně objevena v roce 1972 japonským biomedicínským inženýrem Takuo Aoyagim, když pracoval na vylepšení měření barvivové diluce pomocí ušního denzitometru. Vylepšení nebylo úspěšné, protože naměřené hodnoty, jak se následně ukázalo, byly ovlivněny měnící se hodnotou kyslíkové saturace. Nicméně Aoyagi rozpoznal přínos tohoto objevu pro měření kyslíkové saturace. První komerčně používané zařízení bylo vyrobeno v roce 1975 společností Nikon Kohden a v roce 1977 bylo společností Minoruta Camera Company doplněno o prstovou verzi snímače a fiberoptickou sondu. [5, 13]

Do klinické praxe, konkrétně v situacích, kdy není pacient schopen sám regulovat příjem kyslíku, byl pulsní oxymetr uveden až v roce 1983 Williamem Newem a Markem Yeldermanem. Tento počin představoval zlom ve využití metody a pulsní oxymetrie se záhy velmi rychle rozšířila a stala se nezbytnou součástí nemocničního vybavení. Hlavním přínosem bylo její využití při anestezii, což výrazně přispělo k ochraně pacienta. V roce 1988 byla standardizována britským sdružením AAGB&I[4] a v roce 1990 americkým sdružením ASA[5]. [39]

[4] The Association of Anaesthetists of Great Britain & Ireland

2.5.2. Výpočet kyslíkové saturace

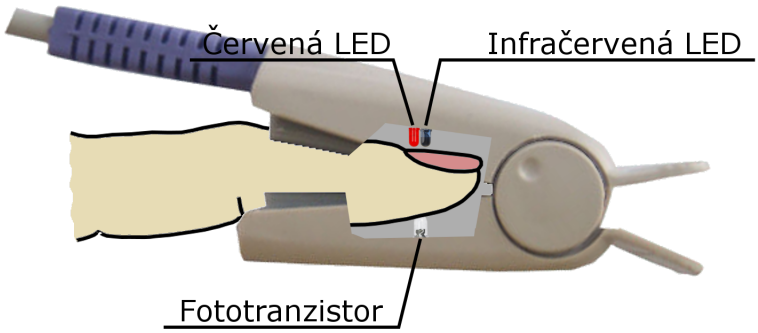
Hodnota kyslíkové saturace (SO_2) je udávána jako poměr oxyhemoglobinu (O_2Hb) vůči sumě všech složek hemoglobinu

$$SO_2 = \frac{O_2Hb}{O_2Hb + Hb + COHb + MetHb + SfHb + COSfHb}.$$

Nicméně měření karboxyhemoglobinu (COHb), methemoglobinu (MetHb), sulfhemoglobinu (SfHb) a karboxysulfhemoglobinu (COSfHb) se z praktických důvodů zanedbává, čímž se výsledný vztah pro výpočet kyslíkové saturace redukuje na

$$SO_2 = \frac{O_2Hb}{O_2Hb + Hb}.$$

Je nutné mít na paměti, že vliv ostatních forem hemoglobinu není v učitých situacích zanedbatelný a jeho vypuštění z výpočtu vnáší do výsledku pro SO_2 nepřesnost. Vlivy a důvody tohoto faktu jsou diskutovány v příslušné kapitole v textu níže.



Obrázek 4: Znázornění použití senzoru pulsního oxymetru.

2.5.3. Princip měření

Při klasickém měření kyslíkové saturace se používá metody, kdy se tkáň prosvěcuje světlem o dvou vlnových délkách. Důvodem je fakt, že kyslík navázaný na hemoglobin mění jeho barvu ze světle červené na tmavě červenou a tím mění i jeho vlastnost absorpce světla. Průběh spektra absorpce záření v hemoglobinu a v oxyhemoglobinu je znázorněn na obrázku 5. Hemoglobin hůře absorbuje

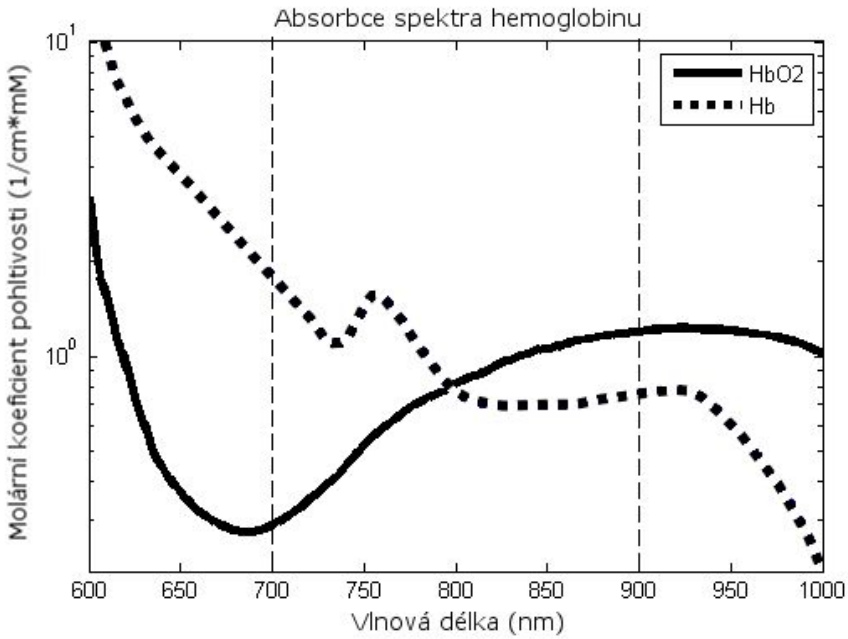
[5] American Society of Anesthesiologists

infračervené světlo o vlnové délce 940 nm než oxyhemoglobin, ale naopak absorbuje lépe červené světlo o vlnové délce 660 nm. Přesněji řečeno se měří intenzita světla, která projde skrze měřenou tkáň. Oba světelné signály se spouští několikrát za vteřinu a ve střídavých intervalech, aby se neovlivňovaly. Naměřené signály se normují a následně se vypočítá jejich poměr. Tím se docílí toho, že pulsace v arteriální krvi, které způsobují lokální zvětšení objemu krve měřeným místem, neovlivní výsledné měření kyslíkové saturace. Nakonec je nutné přiřadit výsledný poměr k tabulkové hodnotě, která je dána Beer-Lambertovým zákonem.

Beer-Lambertův zákon popisuje útlum intenzity monochromatického světla při průchodu homogenním roztokem látky s absorpční vlastností. Vztah byl poprvé empiricky odvozen již v roce 1729 Bouguerem a následně nezávisle odvozen Lambertem v roce 1760 a Beerem v roce 1852. Je vyjádřen exponenciálním vztahem

$$\phi = \phi_0 \cdot 10^{-\varepsilon_\lambda c l}, \quad (\phi_0 > \phi),$$

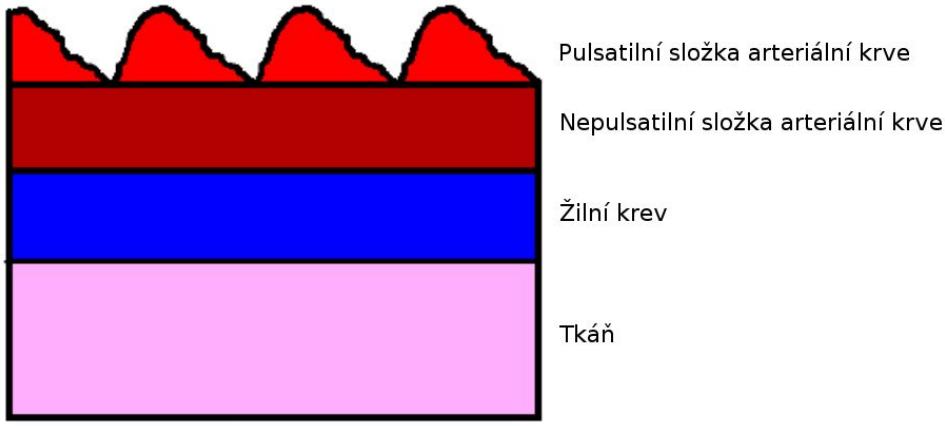
kde Φ_0 je vstupní intenzita, ε je absorpční koeficient závislý na vlnové délce světla, c je koncentrace roztoku látky a l je vzdálenost, kterou světlo materiálem urazí.



Obrázek 5: Absorbce světla hemoglobinu a oxyhemoglobinu při různých vlnových délkách.

Fotodetektor pulsního oxymetru přichází do kontaktu s nepohlceným světlem z dané LED. Měřený signál je invertován použitím invertujícího operačního zesilovače.

Tím se získá signál, který reprezentuje světlo, které bylo prstem pohlceno, a dělí se na dvě složky - stejnosměrnou a střídavou. Stejnosměrná složka reprezentuje světlo pohlcené v tkáni, žilní krvi a nepulsatilní složce arteriální krve. Naproti tomu střídavá složka reprezentuje světlo pohlcené pulsatilní složkou arteriální krve. Pulsatilní složka arteriální krve při maximu amplitudy pohlcuje přibližně 0,5% až 1% z celkového pohlceného světla. Jednotlivé složky jsou graficky znázorněny na obrázku 6. [28]



Obrázek 6: Složky podílející se na absorpci světla prstem při použití pulsního oxymetru. [28]

Hodnota kyslíkové saturace je úměrná poměru střídavých složek signálu (AC) pro periody 660 nm a 940 nm normovaných přes jejich stejnosměrné složky (DC). Tím je získán následující vztah

$$R = \frac{AC_{660}/DC_{660}}{AC_{940}/DC_{940}}$$

který se dále průměruje přes několik hodnot, čímž se docílí snížení nepřesnosti měření. Hodnota kyslíkové saturace (SpO_2) je poté dána empiricky získaným vztahem, tzv. kalibrační křivkou, často v zařízeních reprezentovanou již ve formě převodní tabulky, kterou má zařízení uloženou ve své paměti. Kalibrační křivku je možné aproximovat například lineární lomenou funkcí ve tvaru

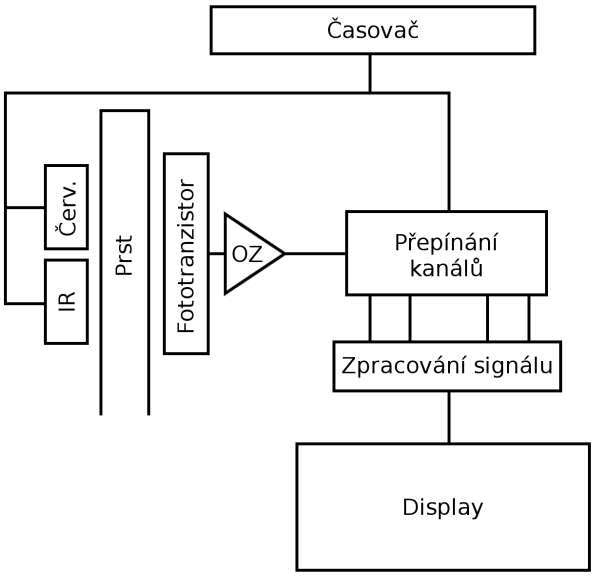
$$SpO_2 = \frac{k_1 - k_2 R}{k_3 - k_4 R}$$

kde hodnoty k_1, k_2, k_3 a k_4 jsou dány při kalibraci zařízení. Nicméně vzhledem k tomu, že běžné měření kyslíkové saturace nevyžaduje vysokou přesnost, používá se lineární vztah

$$SpO_2 = k_1 - k_2 R$$

Tento lineární vztah má malou chybu pro hodnoty SpO_2 větší než 50%. Vzhledem k tomu, že hodnota SpO_2 se běžně pohybuje v rozmezí 97% ± 2% a hypoxická hypoxie u některých lidí nastává již při poklesu SpO_2 pod 90%, je aproximace tímto lineárním vztahem dostačující. Pro běžnou praxi je možné uvažovat kalibrační křivku [1, 53]

$$SpO_2 = 110 - 25R.$$



Obrázek 7: Základní blokové schéma pulsního oxymetru. [36]

2.5.4. Konstrukce pulsního oxymetru

Každý pulsní oxymetr se v základu skládá ze dvou LED (červené a infračervené), fototranzistoru, zesilovače, logické jednotky na přepínání kanálů, bloku zpracování signálu a zobrazovací jednotky. Doplňkovými prvky dále jsou bezdrátový přenos, zvuková signalizace a další. Základní blokové schéma pulsního oxymetru je znázorněno na obrázku 7.

Základním prvkem každého pulsního oxymetru je sonda, která se skládá z červené LED emitující světlo o vlnové délce 660 nm, infračervené LED emitující

světlo o vlnové délce 940 nm a světlodetekujícího prvku, nejčastěji fototranzistoru, generujícího měřený signál. Pulsní oxymetr se dále skládá z časovače a prvku přepínajícího měřené kanály. Tyto obvody zajišťují přepínání LED v krátkých intervalech, například 500 μs, které se průměrují, aby se eliminovaly chyby nesprávných hodnot. Bývají často realizovány pomocí mikrokontroléru. [36]

Zesilovač (OZ) slouží jednak k zesílení poměrně slabého signálu fotodetektoru, a jednak jako převodník proudu na napětí. Důvodem je fakt, že fotodioda reaguje na změnu měřeného světla změnou proudu na výstupu, kdežto preferovanou veličinou pro přenos elektrického signálu je napětí. [35]

Filtrace (zpracování signálu) je nezbytnou součástí zpracování naměřeného signálu pulsního oxymetru. Základní frekvence pulsu nese informaci o tepové frekvenci, nicméně některé vyšší harmonické rovněž nesou přidané užitečné informace. Doporučená šířka pásma zpracovaného signálu se udává 0,01 Hz až 15 Hz. Hlavním rušivým elementem při měření pulsním oxymetrem je pohyb pacienta, jehož frekvenční pásmo se překrývá s pásmem užitečného signálu. Rušivé vlivy dále představuje elektronika zařízení či okolní světlo. Kromě filtrace elektronickým filtrem se používají i další techniky na snížení dopadu rušivých vlivů. Například vliv okolního světla se eliminuje tak, že měření probíhá třífázově. Nejprve je změřena hodnota dána červenou LED, poté hodnota dána infračernevou LED a nakonec je změřena hodnota na fototranzistoru při vypnutí obou LED. Od naměřených hodnot je hodnota daná okolním světlem odečtena. Mimo tuto techniku je nutné samotný senzor od okolního světla co nejlépe odstínit. Používají se zejména tři způsoby a to správné umístění LED a fototranzistoru tak, aby světlo procházelo tkání obsahující arteriální krev, omezení vlivu okolního světla a rozptýleného světla z LED a rovněž omezení spektrální charakteristiky světla dopadajícího na fototranzistor použitím filtrů, které mnohdy bývají již součástí pouzdra součástky. [30, 34, 35]

Zobrazovací jednotka (display) slouží jako zpětná vazba uživateli pulsního oxymetru. Často zobrazovanými informacemi jsou kyslíková saturace, tepová frekvence, pletysmografická křivka či změna amplitudy vyjádřená sloupcovým grafem. [36]

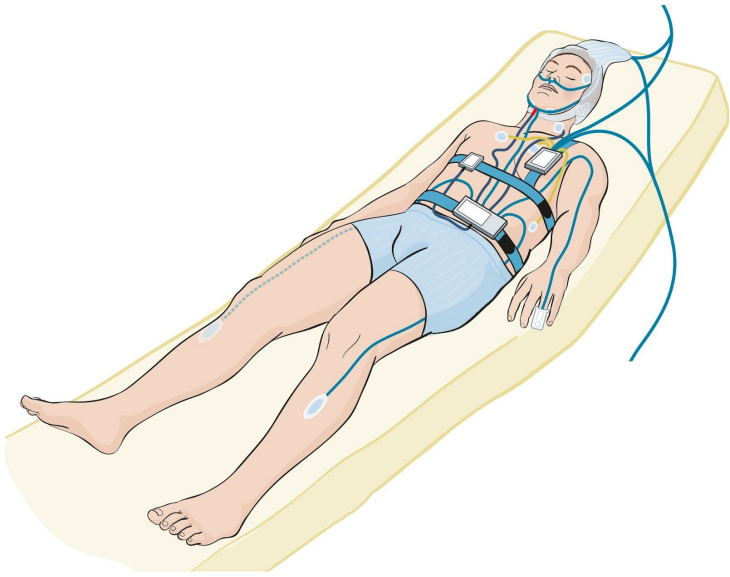
2.5.5. Použití pulsní oxymetrie v praxi

Metoda pulsní oxymetrie má celou řadu využití. Původně se začala využívat na jednotkách intenzivní péče a u perioperačního monitoringu, kde sloužila anesteziologům jako pomůcka pro stanovení kardiorespiračního stavu pacienta, a

kde výrazně zvyšovala ochranu pacientova zdraví. Dnes se pulsní oxymetry používají i na některých běžných odděleních a při intervenční radiologii, při domácím léčení a dokonce i jako podpora při sportu.

Pulsní oxymetrie se používá u celé řady výkonů a vyšetření. Pomáhá stanovit závažnost onemocnění, jehož příznakem může být nedostatečné zásobení kyslíkem, či rozhodnout o nutnosti použití komplexnější analýzy krevních plynů pro stanovení přesnější diagnózy. U domácí kyslíkové terapie představuje důležitou kontrolu pro pacienta, že terapie probíhá v pořádku. Jak již bylo zmíněno výše, používá se rovněž jako nezbytná součást monitoringu pacienta při intenzivní péči. Dále je běžnou součástí polysomnografů (viz obrázek 8) a monitoringu spánku obecně, například při onemocněních jako je spánková apnoe, monitoringu během andoskopického vyšetření a mnoha dalších. [29]

Měření pulsním oxymetrem je ovlivněno celou řadou jevů, mezi které patří vliv okolního světla, mechanické otřesy, abnormality hemoglobinu, vysoká koncentrace karboxyhemoglobinu nebo methemoglobinu, srdeční tep a rytmus, vasokonstrikce a s ní spojená nízká perfúze, samotná činnost srdce a další. Velkou skupinou jevů, které způsobují nízkou hodnotu SaO_2, jsou dechové problémy. Mezi tyto problémy patří nesprávná činnost plic, obstrukce či rezistivita dýchacích cest, nízká difúzní kapacita plic, oslabení dýchacích svalů, hypoventilace, nízký parciální tlak kyslíku a mnoho dalších. [35]



Obrázek 8: Praktické využití pulsního oxymetru v praxi při polysomnografii. [40]

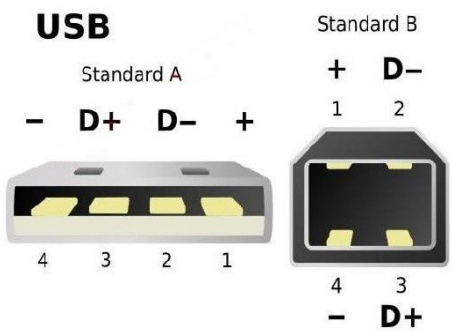
Moderní pulsní oxymetry jsou vybaveny prostředky, které jsou schopny mnoho nepříznivých technických vlivů eliminovat, nicméně je důležité brát tyto vlivy na

vědomí a snažit se jejich zdrojům i přes to předcházet. Například měření vlivu okolního světla a mechanických otřesů ukázalo, že nesprávnou manipulací s pulsním oxymetrem je možné generovat prakticky libovolnou hodnotu SpO_2 a tepové frekvence. [31] Rovněž se zjistilo, že koncentrace karboxyhemoglobinu a methemoglobinu přispívá proporcionálně k rozdílu SpO_2 vůči SaO_2. [32, 33]

S použitím pulsního oxymetru se rovněž váže i celá řada úskalí, na které je potřeba brát zřetel. Obsluha pulsního oxymetru musí projít dostatečným proškolením, neboť jeho chybné použití může v krajním případě mít pro pacienta fatální následky. Například při použití kyslíkové podpory je možné přehlédnout dechové potíže pacienta, které mohou vést k hyperkapnii, která se vyznačuje vzestupem oxidu uhličitého v krvi. Metoda pulsní oxymetrie nám neříká nic o pacientově respiračním stavu, pouze o kyslíkové saturaci, která je dechem podmíněna. Dále je nutné si uvědomit existenci časového zpoždění mezi příčinou poklesu kyslíkové saturace a její detekcí. Při akutní zástavě dechu je pulsní oxymetrie spíše na škodu než k užitku, neboť nám dává falešnou informaci o pacientově skutečném zdravotním stavu. Rovněž je nutné brát v potaz, že pulsní oxymetrie neměří množství kyslíku v krvi, poměr rozpuštěného kyslíku vůči vázanému, frekvenci dýchání, srdeční výdej či krevní tlak. [26]

2.6. Universal Serial Bus

USB (Universal Serial Bus) je universální seriová sběrnice, která byla světu představena v lednu 1996 za účelem zjednodušení a unifikace sběrnicové komunikace pro různé typy zařízení používaných zejména v počítačové technice. Většinou se jedná o zařízení, která nevyžadují sofistikovanou komunikaci, a kde je jednoduché použití a jednoduchá komunikace vítanou výhodou. Kromě komunikace se USB s výhodou používá i pro napájení malých zařízení.



Obrázek 9: Základní USB konektory typu A a typu B. [48]

Ve své klasické podobě využívá USB čtyřvodičové komunikace (viz obrázek 9). Dva vodiče se používají na napájení a dva vodiče se používají jako datové signály. Jako napájení se používá 5V stejnosměrných. Topologie sběrnice je hvězda, kde délka mezi jednotlivými propojenými zařízeními je maximálně 2 až 5 metrů. V centru topologie je zařízení zvané jako host, obvykle osobní počítač, ke kterému jsou připojena periferní zařízení, nazývaná slave. Počet zařízení, která mohou být připojena k jednomu uzlu, je 127. [48]

Mezi výhody USB patří zejména snadné použití, možnost plug & play. Dále velkou výhodu přinesla možnost napájení připojených zařízení v rozsahu od 100 mA do 500 mA, nízká cena a nízká chybovost. Nevýhodou USB specifikace je její složitost pro vývojáře, ať už hardware nebo software a dále zejména možnost využití starých zařízení, pro které musely být vyvinuty speciální integrované obvody a redukce. [49]

Vývoj USB začal v roce 1994, kdy se skupina, té doby na poli výpočetní techniky nejvlivnějších, společností rozhodla vytvořit nový typ sjednocující sběrnice, která by nahradila celou řadu sběrnic, používaných k propojení osobního počítače a jeho periferií. Těmito firmami byly Compaq, DEC, IBM, Intel, Microsoft, NEC a Nortel. Vývoj standardu byl úspěšný a v lednu 1996 byla představena první specifikace USB 1.0. Další specifikace následovaly, stejně tak jako typy konektorů a třídy komunikací. Hlavním cílem každé nové specifikace bylo zejména zvýšení přenosové rychlosti. Tabulka 1 znázorňuje vývoj jednotlivých USB standardů. [48]

Specifikace	Publikováno	Maximální přenosová rychlost
USB 1.0	Leden 1996	1,5 Mbit/s
USB 1.1	Srpen 1998	12 Mbit/s
USB 2.0	Duben 2000	480 Mbit/s
USB 3.0	Listopad 2008	5 Gbit/s
USB 3.1	Červenec 2013	10 Gbit/s

Tabulka 1: Přehled USB standardů [48]

Zařízení používající USB se dělí do mnoha tříd podle typu komunikace. [10] Pro různá zařízení se používají různé třídy komunikací, které si s sebou nesou výhody a nevýhody. Mezi tyto třídy patří například třída pro tiskárny, čtečky chytrých karet, mikrofony, atd. Často používanou třídou je tzv. *Human Interface Device* (HID), který

se používá pro klávesnice, myši či například joysticky. Pomocí tzv. interface může zařízení specifikovat více tříd současně. Typickým zařízením s touto funkcionalitou je mobilní telefon. [7]

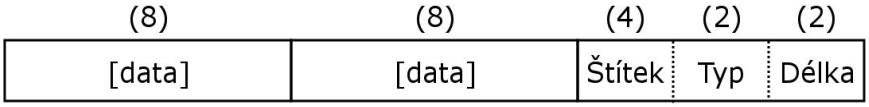
2.7. USB Human Interface Device

USB Human Interface Device je specifikace použití USB pro tzv. "zařízení přímo ovládaná člověkem", to ale není v žádném případě podmínka. Mezi tato zařízení patří například klávesnice, počítačová myš, joystick, herní ovládací zařízení, ale i tlačítka, přepínače, či dokonce měřící zařízení, čtečky čárových kódů apod. Ve zkratce se dá říci, že se jedná o zařízení často generující data, u kterých není nutné potvrzovat příjem. [7]

Informace o USB zařízení jsou uloženy v ROM paměti. Jsou rozděleny na segmenty, které se nazývají descriptory. *Interface descriptor* je důležitým descriptorem, který specifikuje třídu, nebo i více tříd, podle kterých USB zařízení pracuje. Nicméně kvůli velké různorodosti HID zařízení, nedefinuje interface descriptor protokol. Ten se může pro jednotlivá HID zařízení lišit. Pro tyto účely byl zaveden *report descriptor*. Descriptorů USB zařízení je celá řada, jsou uspořádány do hierarchické struktury a tvoří součást firmware zařízení. Kromě těchto dvou descriptorů existuje například *device descriptor*, který reprezentuje celé zařízení a uchovává v sobě informace o zařízení jako je Product ID a Vendor ID, *configuration descriptor*, který udává, jak je zařízení napájené, a jaká používá rozhraní (interface descriptory), a *endpoint descriptor*, který definuje jiný uzel USB než je hlavní uzel. *Device descriptor* může být definován pouze jeden, zbylé descriptory mohou strukturu větvit. [11]

Report descriptor, jak bylo zmíněno výše, sice definuje protokol, ale slouží rovněž jako prostředek pro výměnu dat mezi HID zařízením a HID ovladačem. Je složen z různého počtu menších částí, které se nazývají itemy (*items*). Item je definovaný jednobytovou hlavičkou a volitelně dlouhým blokem dat. Hlavička obsahuje informaci o typu (*type*), štítku (*tag*) a délce (*size*) itemu. Tyto informace dohromady vymezují konkrétní item. Protože se třída HID převážně používá pro zařízení s jasně definovanou funkcionalitou, jako je například klávesnice či myš, je celá řada itemů předdefinovaná. Itemy s předdefinovanou funkcionalitou se v literatuře označují jako *usage*. Existují skupiny itemů zvané kolekce (*collection*), které umí definovat, zda-li se jedná o nějaké předdefinované zařízení, a následně blíže určit, jak se mají jednotlivé části zařízení chovat. Například, jaká data mají být

zahrnuta do zprávy po stisku tlačítka, nebo po pohybu myši. Nicméně, protože se funkcionalita zařízení liší, například funkce myši zabudovaná do klávesnice, lze mezi různými kolekcemi přepínat a používat tak funkce jiných předdefinovaných zařízení[6]. [7, 12, 24]



Obrázek 10: Ukázka struktury itemu.

Komunikace mezi HID zařízením a HID ovladačem (například na straně PC) probíhá dvěma způsoby. Buď přes *kontrolní rouru* (control pipe) nebo přes *přerušení* (interrupt pipe). Nastavení komunikace se provádí pomocí výše zmíněných itemů. Item k tomu určený může být nastaven jako vstupní, a tím definovat vstupní report, nebo jako výstupní, a tím definovat výstupní report. Dále je možné nastavit, zda-li bude komunikace reportu probíhat kontrolní rourou nebo přerušením. Kontrolní roura slouží k odesílání dat vyžádaných ovladačem pomocí příkazu *Get_Report* a k příjmu dat od hostitelského zařízení. Přerušení se používá k asynchronnímu přenosu dat z HID zařízení nebo k přenosu nízkolatentních dat do HID zařízení. Výstupní přerušení je volitelné, a pokud není nastavené, jsou všechna výstupní data posílána kontrolní rourou přes příkaz *Set_Report*. Struktura těchto příkazů je autorovi aplikace pracující s použitým zařízením známá, a je pouze na něm, jak dále s daty naloží.[7]

Obrázek 11: Komunikace mezi HID zařízením a HID ovladačem.

2.8. Komunikace s HID zařízením v Microsoft Windows

Kód v Microsoft Windows může běžet ve dvou módech, v user módu a v kernel módu. Oba módy poskytují různá přístupová práva zejména do paměti či jiných částí systému. Aplikace mohou běžet pouze v user módu, kdežto například ovladače mohou běžet v kernel módu. Vývojářům aplikací komunikujících s externími zařízeními je dán přístup pouze k bezpečnému rozhraní, které ovladač nabízí. Komunikace aplikace s ovladačem se ve Windows provádí přes rozhraní *Windows*

[6] [12] obsahuje hezký návod, jak si definovat vlastní HID report descriptor.
[7] Detailní popis třídy HID, zejména pak nezmíněná témata, je ve specifikaci HID v [7].

API, a to buď přímo voláním funkcí, které nabízí, či nepřímo použitím nějaké, pro tento účel vytvořené, nadstavby. Nadstavbu v tomto případě představuje například rozhraní *.NET Framework*, které je dále v teoretické části diskutováno. *.NET Framework* využívá mezijazyk *Common Intermediate Language* (CIL) a běhové prostředí *Common Language Runtime* (CLR). Veškeré jazyky, které *.NET Framework* zastřešuje, komunikují s rozhraním *Windows API* právě přes CIL a CLR. [23]

Základní knihovnou *Windows API* pro práci s HID zařízením je knihovna *hid.dll*. Obsahuje metody, které se dělí do třech kategorií: vyhledání a nastavení zařízení, přesun dat a tvorba reportů. Metody prvních dvou zmíněných kategorií se označují předponou HidD_ a metody poslední zmíněné kategorie se označují předponou HidP_. Učelem odlišení metod předponami je práce s předdefinovanými reporty, tzv. usage, k čemuž se používají metody kategorie s předponou HidP_. Jejich výhodou je standardizace komunikace s běžně používanými zařízeními jako je například klávesnice či myš. [25]

2.9. C#

C# je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft v roce 2000. Je součástí zastřešující platformy *.NET Framework*, jejíž největší výhodou je specifikace *Common Language Interface* (CLI), která umožňuje několika vysoce úrovňovým programovacím jazykům, mimo jiné například J#, Visual Basic .NET a C++/CLI, kompilaci do jednotného mezikódu (*Common Intermediate Language*), který je následně kompilován do strojového jazyka. Díky tomu se docílí nezávislosti těchto vybraných vysokoúrovňových jazyků na architektuře stroje a operačním systému, pro který mají být využité. [16]

Kód napsaný pod platformou .NET Framework se nazývá řízený kód (*managed code*). Výhodou řízeného kódu je mimo jiné zajištění typové bezpečnosti, ochrany přetékání polí a indexů, zpracovávání výjimek a garbage collector. Nicméně existuje stále mnoho rozhraní a knihoven, které je nutné ve Windows použít. Například se jedná o *Component Object Model* (COM), rozhraní ActiveX a funkce Win32 API, mezi které patří například v této práci použitá knihovna hid.dll. [14, 15]

Syntaxe C# je podobná jazyku Java a původní předlohou obou jazyků jsou jazyky C a C++. Jelikož je podstatou jazyka objektově-orientovaný přístup, je jeho struktura tvořena třídami, ze kterých se vytvářejí instance (objekty). Základními prvky kódu jsou příkazy, které definují chování programu. Posloupnost příkazů, plnících

jako celek nějakou funkci, je umístěna uvnitř metody, jejímž voláním se tato funkce provádí. Skupina metod, která určuje funkcionalitu nějakého složitějšího objektu, se sdružuje do třídy. Nakonec skupina tříd, vztahujících se k nějaké kategorii, se nachází uvnitř jmenného prostoru (*namespace*). Hlavním účelem jmenného prostoru je zabránit konfliktu pojmenovávání tříd různými programátory a tématicky oddělit knihovní metody vztahující se k nějaké vyšší funkcionalitě. Pro představu, jmenným prostorem může být "matematika", která obsahuje třídu "celá čísla", a ta obsahuje metodu "sečti dvě čísla na vstupu", ve které je posloupnost příkazů realizujících tuto metodu.

Pokud je potřeba v kódu používat funkcionalitu z jiného jmenného prostoru, je možné přes tečkovou notaci[8] přistoupit k dané třídě jmenného prostoru, či použít direktivu `using`[9]. Direktivu `using` je nutné použít ještě před deklarací jmenného prostoru.

Proměnné, operátory, primitivní datové typy, základní, rozhodovací a iterační příkazy a metody jsou v zásadě shodné s jazyky Java a C++. Příkazy se ukončují středníkem, při deklaraci proměnných je nutné uvést jejich datový typ, existují zde příkazy `if`, `switch`, `for`, `while` a další, stejně tak, jak je tomu v jazyce Java. Pole a kolekce pracují na stejném principu, ale mají oproti všem výše zmíněným jazykům mírně odlišnou syntaxi. Za zmínku stojí, že C# byl vyvinut mimo jiné jako robustní programovací jazyk a nedovoluje proto kompilaci potencionálně chybového kódu, jako je například použití nepřiřazených lokálních proměnných, či přiřazování hodnot jiných typů než je proměnná. [17]

Nicméně některým chybám zabránit nelze. Jsou to například chybné vstupy od uživatele, či přetečení maximální hodnoty určitého datového typu nebo indexu v poli. Pro tento účel slouží správa chyb a výjimek pomocí příkazů `try, catch` a `finally`, opět velmi podobná jazyku Java. Posloupnost příkazů, která by potencionálně mohla způsobit chybu, se použitím složených závorek obalí do příkazu `try`, a za tento příkaz se použije příkaz `catch` s definovanou třídou výjimky. Pokud v bloku `try` nastane chyba, přeruší se vykonávání tohoto bloku a skočí se do bloku `catch`, kde se provede kód bezpečně ošetřující vzniklou chybu. Z důvodů nutnosti provedení nějakého příkazu, který by případná výjimka přeskočila, se použije příkaz `finally`.

[8] Tečková notace představuje hierarchický přístup do nižších struktur, které jsou veřejně přístupné, od jmenného prosotru, přes třídu, až k metodám či k proměnným. Zapisuje se jako "jmenný prostor".třída.metoda.

[9] Obdoba příkazu `import` programovacího jazyka Java.

Kód uzavřený do bloku `finally` se provede vždy. Používá se například pro uzavření otevřených streamů, či spojení s databází. [17]

Základní syntaxe psaní tříd, jejich metod a konstruktorů, přístupových práv pomocí `public`, `private` a `static`, stejně jako vytváření objektů klíčovým slovem `new`, je téměř shodná s jazykem Java. Přidanou hodnotu zde představuje možnost vytváření částečných tříd, kdy je možné třídu rozdělit do více souborů. Toho je možné docílit označením daných částí tříd jako `partial`. [17]

Referenční typy se od hodnotových typů liší tak, že pracují pouze s adresou paměti, která odkazuje na daný objekt či hodnotu v této paměti uchovanou. Nepracují tedy s hodnotou přímo. Tento přístup je základem objektově orientovaného jazyka. Při předávání referenčního typu metodám používá C# klíčové slovo `ref`, které musí být uvedeno před názvem proměnné v argumentu volané funkce, jak ve funkci volající, tak v její deklaraci. Takto adresované nicméně mohou být i proměnné hodnotových typů, což danou funkcionalitu oproti jiným jazykům rozšiřuje. Mimoto ještě existuje předání pomocí výstupního parametru `out`. Tím je možné referenčním přístupem inicializovat daný parametr až uvnitř metody, což za jiných okolností možné není. [17]

Velmi důležitou kapitolou ve světě objektově orientovaných jazyků je dědičnost. Máme-li několik tříd, které obsahují stejné metody či datové složky, je výhodné, aby všechny tyto třídy dědily od jedné jediné základní třídy. Tím se psaní, údržba i přehlednost kódu výrazně zjednoduší. Oproti jazyku Java má C# několik odlišností. Skutečnost, že třída A (potomek) dědí od třídy B (rodič), se zapisuje pomocí dvojtečky `class A : B`. Dále je nutné, aby se konstruktor potomka odkazoval na konstruktor rodiče přidáním `: base(parametry)`, za jeho deklaraci. Metody v C# je možné skrývat, tj. implementovat v potomkovi metodu se stejným názvem, ale odlišnou funkcionalitou. Skrývající metoda by potom měla mít ve své deklaraci uvedeno klíčové slovo `new`, aby kompilátor věděl, že se nejedná o chybu. Je možné také deklarovat virtuální metody klíčovým slovem `virtual`, které se v rodičovské třídě implementují právě proto, aby byly v dědící třídě pomocí klíčového slova `override` předefinovány. C# rovněž poskytuje úroveň přístupu pomocí klíčového slova `protected`. Metoda rodičovské třídy či její datová složka takto označená, je viděna pouze třídami, které od ní dědí. S pojmem dědičnosti se váže také pojem zapečetěné třídy. V případě, že nechceme, aby bylo možné od nějaké třídy dědit, například proto, že to není logicky či funkčně smysluplné, je možné pomocí klíčového slova `sealed` třídu zapečetit. Tento koncept je možné použít i na metody.

Metoda rodičovské třídy označená jako sealed nemůže být v odvozené třídě předefinována. Nicméně zapečetit je možné pouze předefinovanou metodu. [17]

Rozhraní v C#, stejně jako v mnohých objektově orientovaných jazycích, představuje způsob oddělení definice metod od jejich implementace. Pokud třída dědí od rozhraní, musí implementovat všechny jím definované metody. Rozhraní, jak je ukázáno v praktické části této práce, může například definovat všechny metody, potřebné při použití daného komunikačního protokolu. Rozhraní je definováno klíčovým slovem `interface` a v jazyce C# je dobrým zvykem jako první znak jeho názvu uvádět znak "I". Rozdílem mezi děděním od třídy a děděním od rozhraní je ten, že třída nebo rozhraní může dědit od více rozhraní současně, kdežto dědit od třídy je možné pouze od jedné jediné. Je ale možné dědit od třídy a více rozhraní současně. [17]

Podobně jako rozhraní existují i abstraktní třídy. Narozdíl od rozhraní obsahuje abstraktní třída i implementaci definovaných metod. Nicméně rozdílem od klasické třídy je to, že není možné vytvořit instanci abstraktní třídy. Tím zajistíme implementaci více třídám společných metod. Abstraktní třída je definována klíčovým slovem `abstract`. Uvnitř abstraktní třídy mohou být definované abstraktní metody, které, stejně jako metody uvnitř rozhraní, nemají žádné tělo, a jejich význam je identický. [17]

Během vykonávání programu je nutné pro hodnotové i referenční typy alokovat paměť. Hodnotové typy se ukládájí do zásobníku a pokud skončí jejich obor platnosti, například skončí vykonávání metody obsahující lokální proměnné, jednoduše se na zásobníku uvolní paměť, kterou tyto proměnné používaly. Referenční typy používají haldu a na jeden objekt může ukazovat více referencí. C# stejně jako Java používají automatickou správu paměti zvanou garbage collector. Ta běží samostatně v asynchronním vlákně. Pokud nalezne objekt, na který neodkazuje žádná reference, automatická správa paměti se postará o uvolnění paměti, kterou tento objekt používá. Programátorovi je dána možnost provést vlastní kód během destrukce objektu. K tomuto účelu slouží destruktor, což je funkce se stejným názvem jako její třída, nemá žádné parametry ani návratovou hodnotu a jejímu názvu předchází použití vlnovky. Destruktor není možné zavolat a není ani možné ovlivnit, kdy ho automatická správa paměti vykoná. Důvodem je již zmíněný požadavek jazyka C# na robustnost, která u jazyku C a C++ chybí. V jazyce Java k tomuto účelu slouží funkce `finalize()`, kterou kompilátor C# rovněž používá, ale která je pro použití v kódu jako takovém nedostupná. [17, 18]

Důležitým prvkem syntaxe u třídy, struktury či později uvedeného rozhraní, je implementace vlastností pro přístup k datovým položkám. V C# existuje možnost přistupovat k datovým složkám obdobně, jako jazyk Java používá tzv. gettery a settery, nicméně použití vlastností vede na čistší, jednodušší a obecně bezpečnější kód. Vlastnost vypadá jako datová složka, ale funguje jako metoda. V těle její deklarace se používají dva bloky `get` a `set`. V těle `get` se pomocí návratové hodnoty získá hodnota datové složky a v těle `set` je možné hodnotu datové složky nastavit na požadovanou hodnotu. Vlastnost nemá argument, ale používá vestavěný parametr `value`. Výhodou tohoto přístupu je mimo jiné to, že s vlastností se pracuje jako s klasickou proměnnou. Při čtení vlastnosti se provede kód v těle `get` a při zápisu do vlastnosti se provede kód v těle `set`. Při vynechání deklarace jednoho z těchto klíčových slov potom dostaneme vlastnost pouze pro čtení či vlastnost pouze pro zápis. Vlastnosti je možné kombinovat i s konstruktorem, čímž se docílí výrazně jenoduššího a přehlednějšího kódu. [17]

Vlastnost nabízí elegantní způsob přístupu k datovým složkám třídy. Nicméně je-li použita na pole, její využitelnost se výrazně komplikuje. Indexer je rozšířením vlastnosti o vstupní parametr nazývaný index. Mimoto nabízí možnost pracovat s jakoukoli hodnotovou proměnnou jako s polem, což je výhodné například u bitových operací, kdy můžeme manipuloval s jednotlivými bity reprezentujícími hodnotový typ. [17]

Důležitým konceptem, který u rozsáhlých moderních programovacích jazyků nesmí chybět, je schopnost zpracování událostí. Události nastávají nezávisle na běhu programu a je většinou žádoucí, aby byly obslouženy v čase jejich vzniku. To si s sebou nese nutnost přerušit běh programu, obsloužit vzniklou událost a následně v běhu původního kódu pokračovat. Událost může být vyvolána mnoha faktory jako jsou například stisk tlačítka, uplynutí časovače, či příjem dat po sběrnici USB, jak je uvedeno v praktické části této práce. Základním prvkem, který se s událostmi v C# váže, je delegát. Delegát je ukazatel na metodu, či více metod, a jeho zavoláním se spustí metoda, na kterou ukazuje. Výhodou delegátů je, že je možné je používat jako argumenty metod. Delegát se deklaruje klíčovým slovem `delegate` a metodu je na něj možné navázat přetíženým operátorem +=. Namísto metod je možné k detegátu navázat tzv. lambda výraz, který funguje podobně jako anonymní metoda, avšak je více flexibilní. Lambda výrazy se u delegátů používají zejména v případě, kdy je potřeba na delegát navázat více metod s různými signaturami. Událost se deklaruje

klíčovým slovem `event` a jejím typem musí být detegát, ke kterému se váže. Složí k automatickému spouštění delegátu. [17]

Generické typy představují v programování způsob logického oddělení algoritmu a datových typů. Poskytují nám možnost psát algoritmus bez toho, abychom se starali s jakým datovým typem pracujeme. V C#, stejně jako v jazyce Java, představuje využití generických tříd (struktur, rozhraní, či metod) čistý a typově bezpečný způsob, jak tohoto zobecnění docílit. Generická třída je definována přidáním `<T>` za její název. Symbol `T` zde představuje generický typ. Vytvářením instance dané třídy potom tento generický typ specifikujeme opět přidáním hranatých závorek za název třídy, ale s tím rozdílem, že symbol `T` je nahrazen konkrétním datovým typem. [17]

Posledním prvkem, který je nad rámec čistého C#, a který je dále v této práci používán, je atribut. Atribut je instancí třídy, která dědí od třídy `System.Attribute`. Atribut se vkládá do hranatých závorek a píše se před programový element (třídu, metodu, hodnotový typ atd.). Jeho účelem je modifikovat metadata daného elementu, specifikovat daný element přidanou informací, například označit kód elementu jako zastaralý, což se projeví zahlášením varování při jeho použití, nebo označit, že daná metoda používá části neřizeného kódu, kterým je třeba externí knihovna kernel32.dll, atd. [20]

Jazyk C# nabízí další rozšiřující možnosti jako dotazovací jazyk LINQ, přetěžování operátorů, velmi důležitý a v této práci využívaný subsystém Windows Presentation Foundation a další. Rovněž je možné výše zmíněné prvky probrat více do hloubky a ukázat jejich použití, výhody a omezení. Nicméně pro hrubé pochopení praktické části této práce je zde podaný výklad dostačující. U speciálních případů bude podáno konkrétní vysvětlení. Pro hlubší seznámení se s problematikou C# je možné prostudovat publikace [17, 19, 21].

3. Program

Prvním krokem praktické části projektu byla volba programovacího jazyka. Rozhodnutí padlo na použití C# a to z několika důvodů. Platformou, pod kterou má projekt fungovat, je Microsoft Windows.

Dalším důvodem byla potřeba najít vhodnou knihovnu pro snadnou komunikaci přes USB HID rozhraní. To se ukázalo být jako problém. Windows sice mají hid.dll knihovnu [2], nicméně práce s ní je příliš náročná a komplexní pro použití v tomto projektu. Naštěstí existuje několik volně dostupných knihoven, které práci s hid.dll usnadňují. Po několika neúspěšných pokusech o začlenění těchto knihoven padlo rozhodnutí na použití knihovny C# USB HID Interface napsané Szymonem Roslowskim na webu codeproject.com [3]. Byla snadná na implementaci a celkově funkční pro potřeby projektu. Avšak problémem byla úplná absence dokumentace. Jedním z dílčích cílů této práce bylo připravit popis knihovny C# USB HID Interface tak, aby její uživatel v případě nutnosti zásahu do jejího kódu, nemusel složitě pročítat její implementaci.

Posledním důvodem pro volbu C# byl autorův zájem o seznámení se s touto problematikou. Platforma .NET navíc představuje komplexní a důležitý prostředek pro práci ve Windows a rovněž v komerčním prostředí je hojně využívána a požadována.

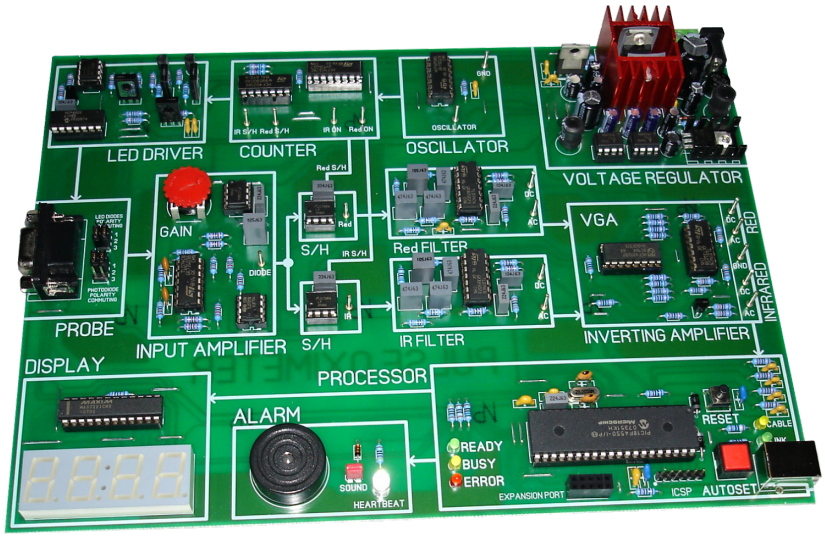
3.1. Použitý pulsní oxymetr

Pulsním oxymetrem ve smyslu této práce je zařízení zobrazené na obrázku 12, které vzniklo jako bakalářská práce Ing. Jana Dvořáka [1], a které je zde použito na seznámení se s problematikou komunikace přes USB HID za využití programovacího jazyka C#.

Motivací pro sestrojení tohoto pulsního oxymetru byl návrh a sestavení laboratorního přípravku umožňujícího záznam hrubých dat, které je možno dále použít při vývoji robustních algoritmů signálového zpracování a při výuce. Tyto signály jsou totiž v klasických komerčních snímačích nedostupné. Autor zařízení laboratorní přípravek rozdělil do funkčních bloků, ze kterých je možné hrubé signály získat a dále zpracovávat.

Modul je schopen z měřeného signálu v reálném čase počítat tepovou frekvenci, krevní kyslíkovou saturaci a přes sběrnici USB odesílat data pomocí 10-bitového A/D

převodníku do počítače. Komunikace s pulsním oxymetrem je popsána v příslušné kapitole praktické části této práce.



Obrázek 12: Pulsní oxymetr Jana Dvořáka.

3.2. C# USB HID Interface

Důvodem, pro který byla tato knihovna vyvinuta bylo jak usnadnění práce s knihovnou hid.dll, tak napsání funkční knihovny pro snadné použití. Struktura knihovny je názorně zobrazena v příloze 2. Za zmínku stojí vysvětlení důležitých částí této knihovny. Hlavní třídou celé knihovny je `UsbHidDevice`, nacházející se ve jmenném prostoru `UsbHid.USB`. Skrze její instanci se provádí veškerá komunikace a obsluha připojeného HID zařízení. Jmenný prostor `UsbHid.USB` se dále dělí na `.Classes`, obsahující veškeré třídy pro práci s HID zařízením, na `.Structures` obsahující struktury pro ucelené uchovávání rozličných informací o HID zařízení a komunikaci s ním. Jmenný prostor `UsbHid.USB.Classes` se potom dělí na `.DllWrappers`, který definuje potřebné metody z Windows API, a na `.Messaging`, který obsahuje třídu reprezentující zprávu protokolu HID. Mimoto jmenný prostor `UsbHid.USB.Classes` obsahuje funkce na obsluhu HID hendleru a na hledání připojených a požadovaných HID zařízení.

V dalším textu bude popsán postup vytvoření instance HID zařízení a práce s ním.

```
Device = new UsbHidDevice(int vendorId, int productId);
```

Nejprve je nutné vytvořit instanci třídy `UsbHidDevice`, do jejíhož konstruktoru vstupuje číslo `vendorId`, které charakterizuje jednotlivého výrobce, a číslo `productId`, které si výrobce zvolí pro odlišení svých produktů. Knihovna předpokládá, že uživatel tato čísla zná, a nenabízí proto žádný mechanismus jejich výběru se seznamu.

Instance `UsbHidDevice` nám zprostředkovává způsob, pomocí kterého můžeme přímo pracovat s připojeným zařízením. Konstruktor zajistí nastavení hodnot připojení do instance struktury `DeviceInformationStructure`, která uchovává informace potřebné pro nalezení požadovaného HID zařízení a cesty k vytvořeným handlerům pro přímou komunikaci s HID zařízením. Dále je konstruktor odpovědný za vytvoření BackgroundWorkera[10], který běží v samostatném vlákně a slouží k zpracovávání událostí jako je například čtení dat ze zařízení. Pro uvolnění paměti používané instancí po jejím zániku, dědí třída od rozhraní `IDisposable` (vynucuje implementaci metody `Dispose()`), což je nutné proto, že HID zařízení je připojené pomocí hendleru z neřízeného kódu Windows API, a automatická správa paměti tento systémový prostředek neumí uvolnit. [21]

Třída dále obsahuje veřejné funkce pro připojení zařízení (`UsbHidDevice.Connect()`), odpojení zařízení (`UsbHidDevice.Disconnect()`), odeslání zprávy do zařízení (`UsbHidDevice.SendMessage(IMessage message)`) a odeslání příkazu do zařízení (`UsbHidDevice.SendCommandMessage(byte command)`). Funkce pro odeslání zprávy a pro odeslání příkazu se liší pouze v tom, že příkaz je zpráva bez parametrů (pole parametrů vrací `NULL`).

```
Device.OnConnected += DeviceOnConnected;
Device.OnDisConnected += DeviceOnDisConnected;
Device.DataReceived += DeviceDataReceived;
```

Třída `UsbHidDevice` definuje tři delegáty na události, jejichž použití je v kódu výše, a to `OnConnected`, `OnDisConnected` a `DataReceived`. Jak název sám napovídá, tyto delegáty odkazují na funkce, které jsou volány při připojení zařízení, odpojení zařízení, či při detekci dat z připojeného zařízení.

Na tyto funkce je možné pomocí delegátu navázat vlastní funkce, v tomto případě `DeviceOnConnected()`, `DeviceOnDisConnected()` a `DeviceDataReceived(byte[] data)`. Při každé události vyvolané čtením dat ze

[10] Viz “BackgroundWorker” ve slovníku pojmů v příloze 2.

vzdáleného zařízení, se tato data posílají jako vstupní parametr do poslední zmíněné metody. Uživatel knihovny si poté s témito daty může naložit podle svého.

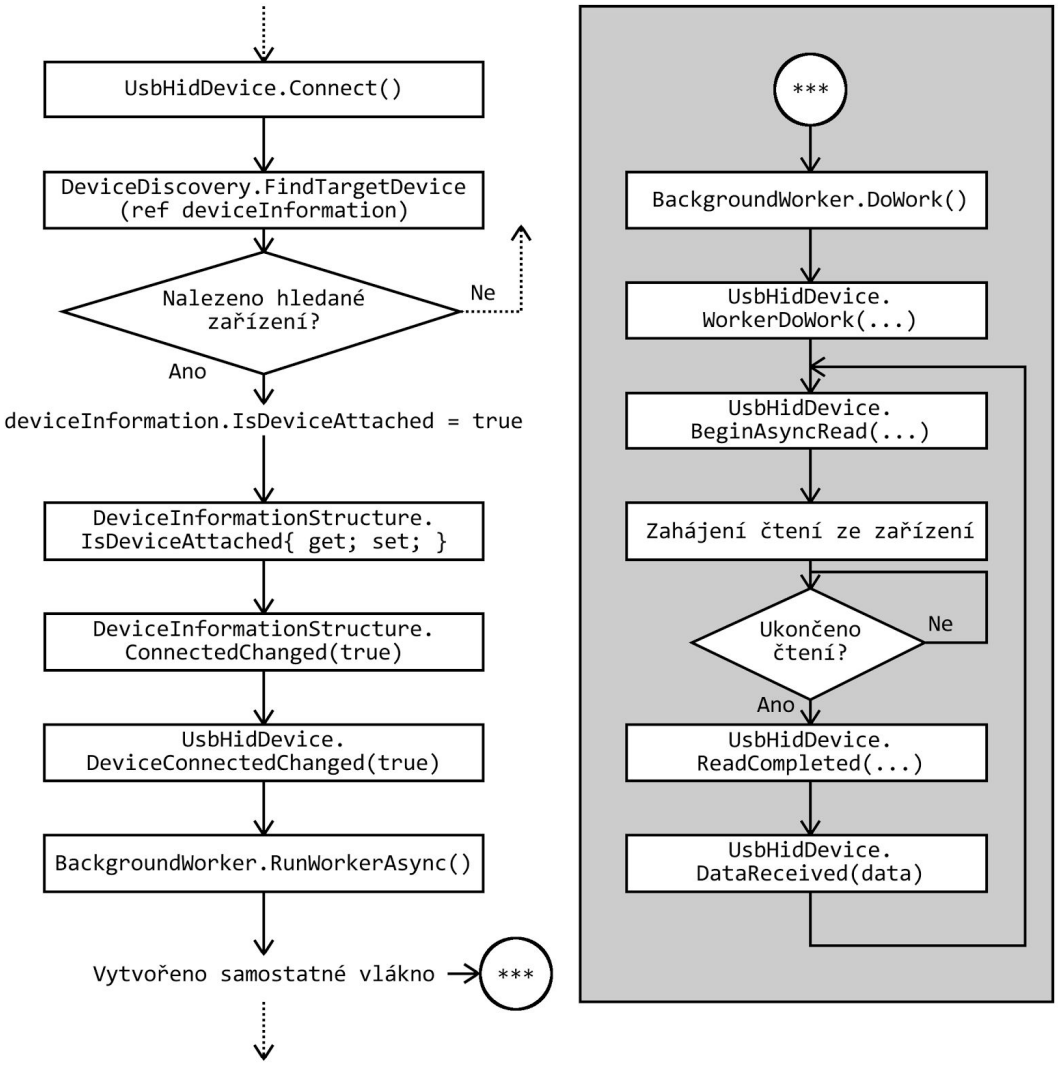
```
Device.Connect();
```

Po inicializaci všech potřebných událostí je možné pomocí metody `UsbHidDevice.Connect()` zařízení přípojit. Tato metoda volá metodu `FindTargetDevice` z třídy `DeviceDiscovery` a vrací příznak typu bool o výsledku připojení. Metoda `FindTargetDevice` probíhá následovně. Nejprve se použitím metody `FindHidDevices` získají adresy všech připojených HID zařízení. Poté se vybere takové, u kterého se shoduje požadované Vendor ID a Product ID. A nakonec se na toto zařízení vytvoří ukazatel, který mohou používat metody pro zápis a čtení dat.

Jak bylo zmíněno výše, k odesílání dat do připojeného HID zařízení slouží dvě metody `SendMessage` a `SendCommandMessage`. Obě metody volají metodu `WriteRawReportToDevice` třídy `DeviceCommunication` a jediný rozdíl mezi nimi je, že uživatel `SendMessage` této metodě posílá instanci třídy `CommandMessage`, která může obsahovat pole parametrů, a uživatel třídy `SendCommandMessage` této metodě posílá pouze číselný příkaz, a metoda si pomocí něho instanci `CommandMessage` sama vytvoří. Tím se zajistí, aby příkaz posílaný do HID zařízení, neměl žádné parametry. Metoda `WriteRawReportToDevice` poté použitím metody `Kernel32.WriteFile` pouze zapíše data do příslušného zapisovacího hendleru připojeného HID zařízení, čímž se na tomto zařízení vyvolá přerušení a případná následná akce vzhledem k použitému protokolu.

Zpráva posílaná do zařízení a zmíněná v předchozím odstavci se sestává z nulového bytu, bytu pro příkaz (`command`) a 63 bytů vyhrazených pro parametry (`parameters`). Její formát je dán protokolem HID rozhraní a její obsah je dán protokolem použitého zařízení a stanovený jeho výrobcem. Formát zprávy je následující

0	command (1)	parameters (63)



Obrázek 13: Schématické znázornění čtení dat ze zařízení.

Čtení dat ze zařízení je vzhledem k zápisu dat do zařízení složitější na pochopení. Pro lepší pochopení této funkcionality slouží obrázek 13. Zápis do zařízení je uživatelem vynucená operace, kdežto čtení dat ze zařízení je vůči vykonávání programu asynchronní, a proto je nutné, aby obslužná metoda běžela v samostatném vlákně. Knihovna C# USB HID tento problém řeší vytvořením tzv. BackgroundWorkera[11]. Ke spuštění BackgroundWorkera je nutné zavolat metodu `RunWorkerAsync`. Ta je volána z metody `DeviceConnectedChanged` napojené na příslušný delegát, při úspěšném nalezení hledaného zařízení z výše uvedené metody `FindTargetDevice`. Voláním metody `RunWorkerAsync` vynutí příslušný delegát metodu `WorkerDoWork`. Následující kód již běží ve svém vlastním vlákně.

[11] Viz “BackgroundWorker” ve slovníku pojmů v příloze 2.

Metoda `WorkerDoWork` otevře `FileStream` pro čtení dat a následná metoda `BeginAsyncRead` spouští metodu `BeginRead`. Výhodou metody `BeginRead` je to, že po skončení čtení je volána metoda předaná v jejím vstupním argumentu. Touto metodou je `ReadCompleted`, jejímž cílem je získat přečtená data a předat je přes delegát `DataReceivedDelegate` uživatelské metodě `DeviceDataReceived`, kde dochází k jejich konečnému zpracování. Blok `finally` v metodě `ReadCompleted` zajistí opětovné volání metody `BeginAsyncRead`, čimž se v paralelním vlákně vytvoří smyčka a celý proces se opakuje až do zániku vlákna. Avšak pro čtení dat ze zařízení je ještě nutné si o ně danému zařízení říci. O tom je pojednáváno v následujícím textu.

3.3. Komunikace s pulsním oxymetrem

Pro připojení pulsního oxymetru k aplikaci musíme znát jeho Vendor ID a Product ID. V případě použitého pulsního oxymetru se jednalo o dvojici “04D8” a “003F”. Následné rozšíření knihovny C# USB HID umožňuje pomocí GUI výběr zařízení ze seznamu a není tedy nutné si hodnoty Vendor ID a Product ID zjišťovat. Po úspěšném připojení je před samotným čtením dat nutné si o ně říci posláním speciálního příkazu do zařízení s hodnotou příkazu 15. Pro zastavení odesílání dat zařízením potom analogicky slouží příkaz 7. K tomu lze použít volání knihovní metody `SendCommandMessage` v následujícím formátu.

```
Device.SendCommandMessage(byte command);
```

Poté začne pulsní oxymetr data posílat do počítače, a ta jsou následně asynchronně čtena a zpracovávána uživatelem knihovny pomocí metody napojené na příslušného delegáta (`DataReceivedDelegate`). Data posílaná ze zařízení[12] představují dva průběhy měření a to hodnotu odpovídající intenzitě infračervené složky a hodnotu odpovídající intenzitě červené složky použitého světla.

3.4. Zpracování dat ze zařízení

Měřená data ze zařízení jsou zpracovávána za účelem stanovení průběhu saturace kyslíkem, stanovení celkové hodnoty kyslíkové saturace a určení tepové frekvence. Datový balík posílaný použitým zařízením pulsního oxymetru se skládá

[12] Datový rámec je popsaný v práci [1], v příloze 2.

mimo jiné ze stejnosměrných složek měřeného signálu pro červenou (DC_{RED}) a infračervenou (DC_{IR}) LED a z třinácti vzorků odpovídajících střídavým hodnotám pro červenou (AC_{RED}) a infračervenou (AC_{IR}) LED. Jeden datový balík byl použit pro výpočet jedné hodnoty SpO_2, a to z toho důvodu, že výpočet SpO_2 je nutné průměrovat přes několik po sobě jdoucích hodnot. Výpočet byl proveden nejprve získáním normovaného poměru

$$R = \frac{\frac{1}{DC_{RED}} \cdot \sum_{i=1}^{13} AC_{REDi}}{\frac{1}{DC_{IR}} \cdot \sum_{i=1}^{13} AC_{IRi}}.$$

A poté následoval výpočet samotného SpO_2 z obecně přijímaného vztahu

$$SpO_2 = 110 - 25R.$$

Takto vypočtená hodnota SpO_2 je vynášena do grafu.

3.5. Rozšíření C# USB HID Interface

Použitá knihovna C# USB HID Interface neumožňuje vyhledat HID zařízení jinak, než ze znalosti hodnot Vendor ID a Product ID, což je z uživatelského hlediska poměrně limitující. Proto byla knihovna rozšířena o metodu `FindHidDeviceDescription` ve třídě `DeviceDiscovery`. Vstupem této metody jsou referencí volaná pole pro výrobce HID zařízení, název zařízení, Vendor ID a Product ID. Připojená HID zařízení jsou postupně v cyklu pomocí volání Windows API čtena a dotazována na svůj název, výrobce a hodnoty ID atributů.

Výše zmíněná volání Windows API nebyla rovněž v knihovně implementována, čímž bylo nutné o tyto volání rozšířit třídu `Hid` ze jmenného prostoru `UsbHid.USB.Classes.DllWrappers`.

Tato funkcionalita byla následně využita v GUI aplikaci k možnosti vylistování všech připojených HID zařízení a poskytnutí uživateli aplikace možnost připojit se k chtěnému HID zařízení.

3.6. GUI aplikace

Pro demonstraci funkčnosti komunikace se zařízením pulsního oxymetru byla napsána GUI aplikace (viz obrázek 14). Pomocí této aplikace je možné přes výše

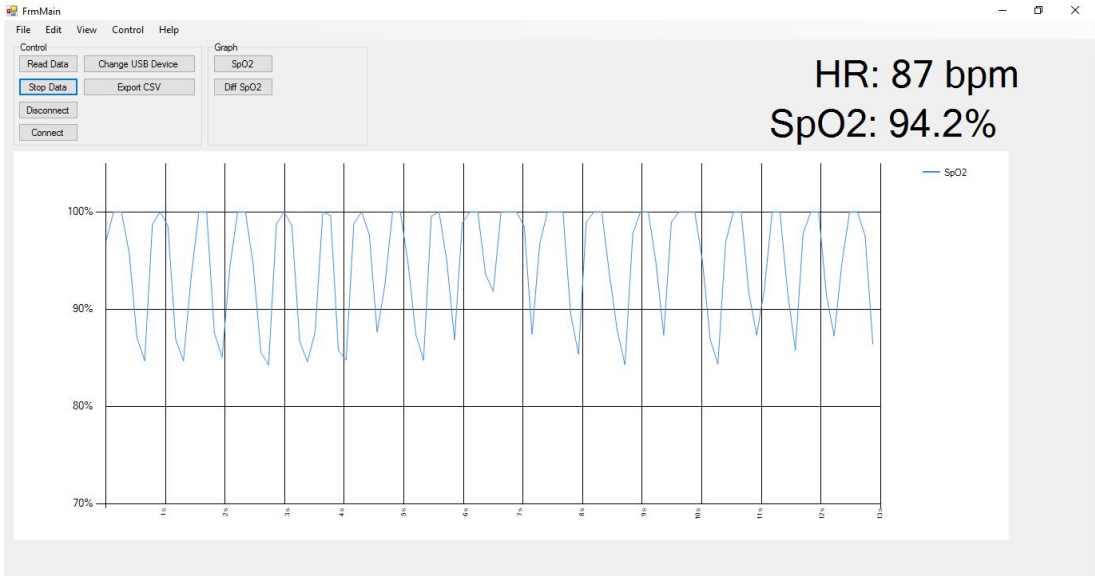
zmíněnou knihovnu C# USB HID zjistit připojená HID zařízení. Pokud je v tomto seznamu zařízení výše zmíněný pulsní oxymetr, je možné oxymetr připojit a zahájit čtení měřených dat. Z měřených dat je vypočtena kyslíková saturace, která je následně zobrazena v grafu. Z hodnot grafu jsou potom vypočteny hodnoty průměrné kyslíkové saturace a tepové frekvence.

V prvním kroku je nutné stisknout tlačítko “Change USB Device”. Tím se otevře okno s možností výběru HID zařízení. Výběrem daného zařízení ze seznamu získáme informaci o jeho výrobci a jeho název. Pokud je zařízení odpojeno je možné ho stiskem tlačítka “Connect selected device” připojit. Pokud zařízení připojeno není, je možné ho stiskem tlačítka “Disconnect current device” odpojit. Pokud je nějaké zařízení již připojeno, je zobrazen jeho název v dolní části okna za textem “Current device: “. Po dokončení fáze připojení nebo odpojení zařízení je možné toto okno stiskem tlačítka “Close” zavřít.

Po úspěšném připojení pulsního oxymetru k aplikaci je možné v hlavním okně v sekci “Control” stiskem tlačítka “Read Data” zahájit čtení dat ze zařízení. Tím je pomocí knihovny C# USB HID voláno přerušení, ve kterém jsou data ze zařízení zpracována tak, jak je popsáno v příslušné kapitole této práce. Zpracovaná data jsou ve formě vypočtené kyslíkové saturace vynášena do grafu. Rozsah časové osy je třináct vteřin a rozsah osy hodnot SpO2 je od 70% do 100%, protože hodnota SpO2 pod 70% je vždy brána za příznak hypoxie. Po třináctivteřinových intervalech jsou stará data překreslována novými. Pro ukončení čtení dat ze zařízení slouží tlačítko “Stop Data”. Tlačítkem “Disconnect” je možné rovnou v hlavním okně GUI připojené zařízení odpojit. Pomocí tlačítka “Export CSV” je možné naměřená data i s jejich časovými hodnotami exportovat do CSV souboru na zvolené místo v počítači.

V hlavním okně aplikace se dále nachází sekce “Graph”, která pouze slouží k demonstraci možného rozšíření aplikace. Z nabídky je možné pomocí tlačítek “SpO2” a “Diff SpO2” přepínat zobrazení grafu. Tlačítko “SpO2” zobrazuje standardní průběh popsaný výše a tlačítko “Diff SpO2” zobrazuje jeho diferenci. Samotný graf diference nemá praktické využití a slouží jako demonstrace možné funkcionality.

Aplikaci je možné uzavřít standardně z menu “File” a “Exit”. Je nutné si uvědomit, že uzavřením aplikace nedojde k odpojení zařízení, či k ukončení čtení dat. To je zejména z toho důvodu, že data jsou zařízením odesílána nezávisle na aplikaci, a uživatelem nevyžádané ukončení čtení dat by mohlo způsobit uživateli problémy, pokud zařízení v daném čase využívá i k jiným účelům.



Obrázek 14: Vzhled GUI aplikace s naměřenými daty.

4. Závěr

Zadání diplomové práce bylo zvoleno velmi široce, čímž vznikla možnost seznámit se s celou řadou pro autora nových pojmů, kterými byly zejména komunikace po sběrnici USB a programovací jazyk C#. Podle toto bylo i voleno vypracování práce a jednotlivá témata byla popisována obecně, aby se s danou problematikou mohl seznámit i čtenář pojmů neznalý. Reference, ze kterých autor práce čerpal, obsahují několik hezkých knih, které všechna popisovaná témata detailněji a celistvě popisují.

Praktická část práce se zaměřila na řešení problému komunikace se zvoleným pulsním oxymetrem. Zde bylo nutné překonat několik obtíží, a to zejména zprostředkovat komunikaci po implementačně složité sběrnici se zařízením, ke kterému chyběla vývojářská dokumentace, a v programovacím jazyce pro autora neznámem. Tento úkol usnadnilo použití již napsané open-source knihovny pro práci s HID zařízením v jazyce C#. Tím se směr práce vydal cestou důkladného prostudování a popisu této knihovny pro případné budoucí uživatele.

Po úspěšném propojení pulsního oxymetru s počítačem a rozšířením použité knihovny o další funkcionalitu byla napsána GUI aplikace, sloužící jak k demonstraci funkcionality, tak k faktickému měření kyslíkové saturace a srdeční frekvence.

Posledním bodem v zadání práce bylo navrhnout nutné změny firmware zvoleného měřicího zařízení. Jelikož byl software přizpůsoben použitému zařízení, nebylo tento bod nutné implementovat. Nicméně jistě by stála za vyzkoušení změna komunikačního protokolu přes nastavení HID descriptorů daného zařízení. Bohužel kvůli autorově pracovnímu vytížení na tuto úlohu nezbyl čas.

Reference

[1] DVOŘÁK, Jan. *Pulsní oxymetr*. Praha, 2010. Bakalářská práce. ČVUT. Vedoucí práce Ing. Jan Havlík, Ph.D.

[2] Introduction to HID Concepts. In: *MSDN* [online]. 2016 [cit. 2016-06-19]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/hardware/jj126202(v=vs.85).aspx

[3] C# USB HID Interface. In: *CodeProject* [online]. 2013 [cit. 2016-06-19]. Dostupné z: http://www.codeproject.com/Tips/530836/Csharp-USB-HID-Interface

[4] System.Management Namespace. In: *MSDN* [online]. 2016 [cit. 2016-06-19]. Dostupné z: https://msdn.microsoft.com/en-us/library/system.management(v=vs.110).aspx

[5] Hemoglobin. In: *Wikipedia* [online]. 2016 [cit. 2016-06-19]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Hemoglobin&oldid=13456661

[6] Methemoglobin. In: *Wikipedia* [online]. 2016 [cit. 2016-06-19]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Methemoglobin&oldid=11224860

[7] *Universal Serial Bus: Device Class Definition for Human Interface Device*. 2001.

[8] Delegate. In: *MSDN* [online]. 2016 [cit. 2016-07-12]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms173171.aspx

[9] How to: Use a Background Worker. In: *MSDN* [online]. 2016 [cit. 2016-07-12]. Dostupné z: https://msdn.microsoft.com/en-us/library/cc221403(v=vs.95).aspx

[10] USB-IF Device Class Documents. In: *Universal Serial Bus* [online]. 2016 [cit. 2016-07-15]. Dostupné z: http://www.usb.org/developers/docs/devclass_docs/

[11] USB Descriptors. In: *USB in a NutShell* [online]. 2016 [cit. 2016-07-16]. Dostupné z: http://www.beyondlogic.org/usbnutshell/usb5.shtml

[12] Tutorial about USB HID Report Descriptors. In: *Eleccelerator* [online]. 2016 [cit. 2016-07-16]. Dostupné z: http://eleccelerator.com/tutorial-about-usb-hid-report-descriptors

[13] SEVERINGHAUS, John W. a Yoshiyuki HONDA. History of Blood Gas Analysis. VII. Pulse Oximetry. *Journal of Clinical Monitoring*. 1987, **3**(2), 135–138. DOI: 10.1007/bf00858362.

[14] What Is Managed Code. In: *MSDN* [online]. 2016 [cit. 2016-07-23]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/bb318664(v=vs.85).aspx

[15] Interoperating with Unmanaged Code. In: *MSDN* [online]. 2016 [cit. 2016-07-23]. Dostupné z: https://msdn.microsoft.com/en-us/library/sd10k43k(v=vs.110).aspx

[16] C Sharp (programming language). In: *Wikipedia* [online]. 2016 [cit. 2016-07-23]. Dostupné z: https://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&oldid=731002010

[17] SHARP, John. *Microsoft Visual C# 2008 Krok za krokem*. Brno: Computer Press, a. s., 2008. ISBN 978-80-251-2027-9.

[18] VENNERS, Bill. Object finalization and cleanup: How to design classes for proper object cleanup. In:*JavaWorld* [online]. 1998 [cit. 2016-08-20]. Dostupné z: http://www.javaworld.com/article/2076697/core-java/object-finalization-and-cleanup.html

[19] C#. In: *MSDN* [online]. 2016 [cit. 2016-08-21]. Dostupné z: https://msdn.microsoft.com/ en-us/library/kx37x362.aspx

[20] Attributes. In: *MSDN* [online]. 2016 [cit. 2016-08-21]. Dostupné z: https://msdn.microsoft. com/en-us/library/mt653979.aspx

[21] KEPRT, Aleš. *Systémové programování v jazyce C#* [online]. Olomouc, 2008 [cit. 2016-08-22].

[22] How to: Run an Operation in the Background. In: *MSDN* [online]. 2016 [cit. 2016-08-22]. Dostupné z: https://msdn.microsoft.com/en-us/library/hybbz6ke(v=vs.110).aspx

[23] AXELSON, Jan. *USB Complete: Everything You Need to Develop USB Peripherials, Third Edition*. 2005.

[24] Universal Serial Bus: HID Usage Tables. In: *www.usb.org* [online]. 2004 [cit. 2016-08-25]. Dostupné z: http://www.usb.org/developers/hidpage/Hut1_12v2.pdf

[25] Introduction to HID Concepts. In: *MSDN* [online]. 2016 [cit. 2016-08-25]. Dostupné z:

https://msdn.microsoft.com/windows/hardware/drivers/hid/introduction-to-hid-concepts

[26] E. HILL, MD Stoneham a FEARNLEY S-J. Practical Applications of Pulse Oximetry. *Update in Anaesthesia*. 2000, **11**, 156–159.

[27] Hemoglobin. In: *Wikipedia* [online]. 2016 [cit. 2016-11-13]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Hemoglobin&oldid=748790555

[28] LOPEZ, Santiago. *Pulse Oximeter Fundamentals and Design.* In: *www.nxp.com* [online]. 2012 [cit. 2016-11-18]. Dostupné z: http://www.nxp.com/files/32bit/doc/app_note/AN4327.pdf?tid=AMdlDR

[29] Konica Minolta Sensing, Inc. *How to Read SpO_2, Third Edition*. 2006.

[30] N. STUBAN, M NIWAYAMA. Optimal filter bandwidth for pulse oximetry. *Review of Scientific Instruments*. 2012, **83**.

[31] NS. Trivedi, AF Ghouri, NK Shah, E Lai, SJ Barker. Effects of motion, Ambient Light, and Hypoperfusion on Pulse Oximeter Function. *J Clin Anesthesia*. 1997, **9**, 179-183 doi:10.1016/S0952-8180(97)00039-1

[32] SJ Barker, KK Tremper, J Hyatt. Effects of Methemoglobinemia on Pulse Oximetry and Mixed Venous Oximetry. *Anesthesiology*. 1989, **70**, 112-117

[33] SJ Barker, KK Tremper. The Effect of Carbon Monoxide Inhalation on Pulse Oximetry and Transcutaneous PO2. *Anesthesiology.* 1987, **66**, 677-679

[34] Pulse oximetry. PROUT, Jeremy, Tanya JONES a Daniel MARTIN. *Advanced Training in Anaesthesia*. Oxford: Oxford University Press, 2014, s. 66-67. ISBN 978-0-19-960995-6.

[35] WEBSTER, J G (ed.). *Design of Pulse Oximeters*. New York: Taylor & Francis Group, 1997. ISBN 978-7503-0467-2.

[36] JN Amoore. Pulse oximetry: an equipment management perspective. *IEE Colloquium on Pulse Oximetry: A Critical Appraisal.* 1996

[37] FONTAINE, Alexandra, Arben KOSHI, Danielle MORABITO a Nicolas RODRIGUEZ. *Reflectance - Based Pulse Oximeter for the Chest and Wrist*. Worcester Polytechnic Institute. Vedoucí práce Yitzhak Mendelson.

[38] *Kofrlab wiki* [online]. Praha, 2016 [cit. 2016-12-03]. Dostupné z: http://patf-biokyb.lf1. cuni.cz/wiki/start

[39] POLE, Yash. Evolution of the pulse oximeter. *International Congress Serie*. 2002, (1242), 137-144.

[40] *Illumedic* [online]. illumedic Design, 2016 [cit. 2016-12-04]. Dostupné z: http://illumedicdesign.se/

[41] OPENSTAX. *Anatomy & Physiology* [online]. OpenStax CNX, 2013 [cit. 2016-12-10]. Dostupné z: https://opentextbc.ca/anatomyandphysiology/

[42] BOOTH, Jeremy. A Short History of Blood Pressure Measurement. *Proc. roy. Soc. Med*. 1977, (70), 793-799.

[43] Stanovení srdečního výdeje. In: *WikiSkripta* [online]. 2016 [cit. 2016-12-14]. Dostupné z: http://www.wikiskripta.eu/index.php/Stanoven%C3%AD_srde%C4%8Dn%C3%ADho_v%C3%BDdeje

[44] GAWLINSKI, Anna. Measuring Cardiac Output: Intermittent Bolus Thermodilution Method. *Critical Care Nurse*. 2004, **24**(5), 74-78.

[45] GARCÍA, X., L. MATEU a J. MAYNAR. Estimating cardiac output. Utility in the clinical practice. Available invasive and non-invasive monitoring. *Med Intensiva*. 2011, **35**(9), 552–561. DOI: 10.1016/j.medine.2012.01.001.

[46] LINTON, R. A. F., D. M. BAND a K. M. HAIRE. A New Method of Measuring Cardiac Output in Man Using Lithium Dilution. *British Journal of Anaesthesia*. 1993, **71**, 262-266.

[47] Calibration of the LiDCOplus System. *LiDCO Group plc* [online]. London, 2016 [cit. 2016-12-15]. Dostupné z: http://www.lidco.com/products/lidcoplus/calibration.php

[48] USB. *Wikipedie* [online]. Wikimedia Foundation [cit. 2016-12-31]. Dostupné z: https://en.wikipedia.org/wiki/USB

[49] *USB in a NutShell* [online]. 2010 [cit. 2016-12-31]. Dostupné z: http://www.beyondlogic.org/usbnutshell/

[50] Hemodynamics. In: *Wikipedia* [online]. 2017 [cit. 2017-01-08]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Hemodynamics&oldid=758475541

[51] Mean arterial pressure. In: *Wikipedia* [online]. 2017 [cit. 2017-01-08]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Mean_arterial_pressure&oldid=757931966

[52] Sphygmomanometer. In: *Wikipedia* [online]. 2017 [cit. 2017-01-08]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Sphygmomanometer&oldid=756170228

[53] STUBÁN, N., N. Masatsugu. Non-invasive calibration method for pulse oximeters. *Periodica Polytechnica*. 2008, **52**, 91-94.

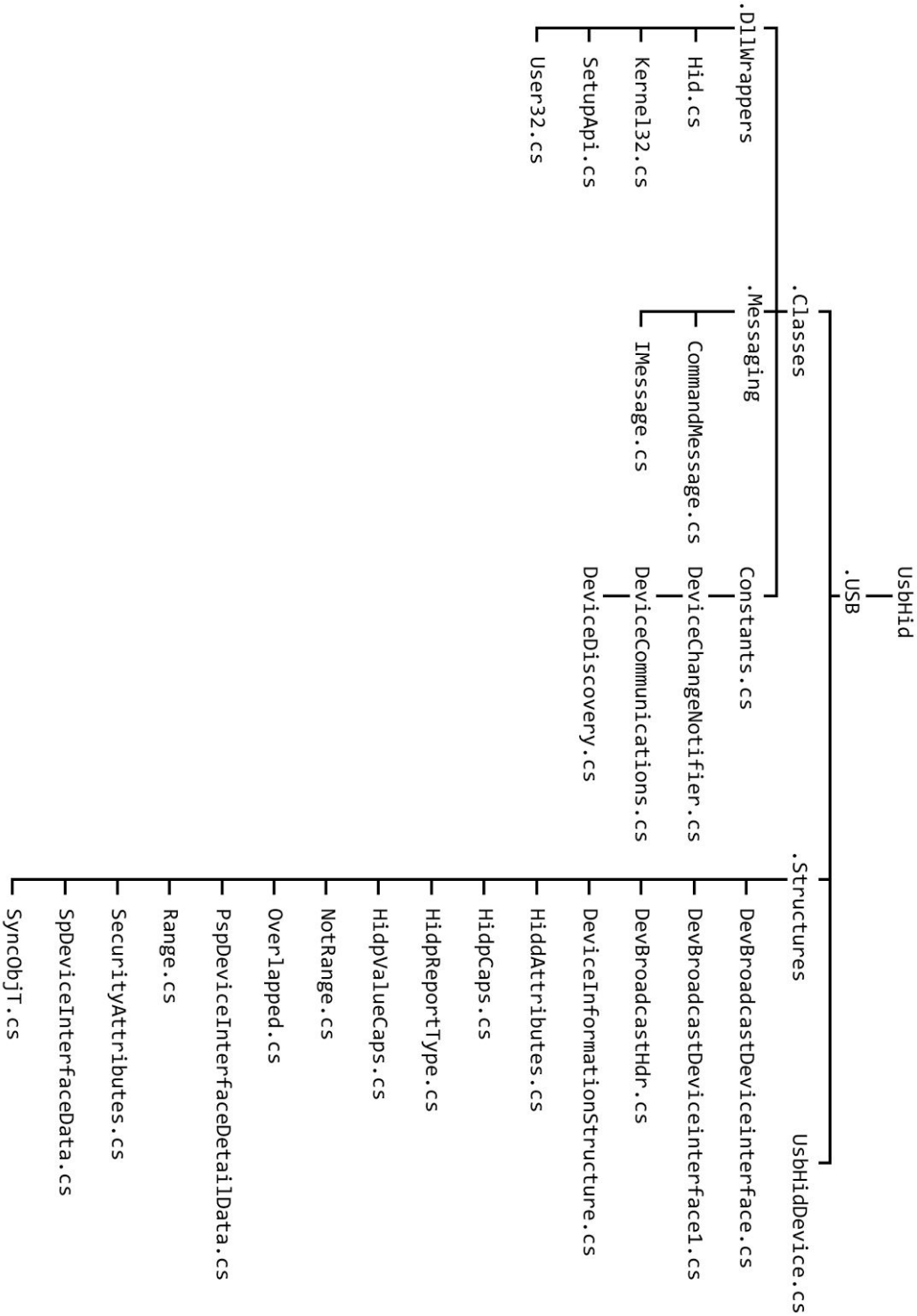
Příloha 1: Seznam vysvětlených pojmů

1. **BackgroundWorker** je třída, která umožňuje provádět operace v samostatném vlákně. Používá se zejména pro časové náročné operace, kdy není žádoucí čekat na dokončení, a znemožnit tak provádění dalších operací. BackgroundWorker se používá například pro asynchronní čtení dat ze vzdáleného zařízení. Na hendler události BackgroundWorker.DoWork je možné navázat metodu, která je při zavolání BackgroundWorker.RunWorkerAsync() spuštěna v samostatném vlákně. Třída BackgroundWorker umožňuje dále zjišťovat stav procesu vlákna, dotazovat se na jeho provedení, či na chybové hlášky v případě neúspěšného ukončení procesu. [22]

2. **Dalegate** (delegát) je datový typ, který reprezentuje referenci na metodu s danými parametry a s danou návratovou hodnotou. Pokud je vytvořena instance delegátu, je ji možné přidružit k metodě s kompatibilním podpisem a návratovou hodnotou. Tyto metodu je potom možné volat zkrze delegátu. Tím je možné, použitím delegátu, posílat metody, přes vstupní argumenty, do jiných metod. [8]

3. **GUID** (nebo UUID) je akronym pro “Globally Unique Identifier” (nebo “Universally Unique Identifier”). Je to 128 bitový integer používaný k identifikaci zdrojů. Termín GUID je obecně používán vývojáři pracujícími pod Windows, zatímco UUID se používá všude jinde. C# GUID reprezentuje strukturou `System.Guid`.

Příloha 2: Struktura knihovny C# USB HID



Příloha 3: Obsah přiloženého DVD

diplomova_prace.pdf
program.zip