ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Remeš** Jméno: **Kryštof** Osobní číslo: **439133**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídicí techniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Softwarový systém pro rehabilitační pomůcku

Název bakalářské práce anglicky:

Software system for rehabilitation aids

Pokyny pro vypracování:

1. Seznamte se s požadavky na rehabilitaci pacientů s motorickým postižením a s navrženým vývojovým vzorkem trénovacího panelu.
2. Proveďte analýzu potřeb SW vybavení pro ovládání tréningového panelu pro pacienty s motorickým postižením.
3. Na základě analýzy dle bodu 2 zvolte vhodný řídicí systém pro rehabilitační pomůcku a vytvořte odpovídající základní SW.
4. Realizovaný SW vyzkoušejte v praxi.

Seznam doporučené literatury:

Vondra, Jakub. Rehabilitační přístroj. Praha, 2016. Diplomová práce (Ing.). České vysoké učení technické v Praze. Fakulta strojní, ústav přístrojové a řídicí techniky. Vedoucí práce Šárka Němcová.
dle pokynů vedoucího práce

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Jan Chyský, CSc., U12110.1

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

Datum zadání bakalářské práce: **18.04.2018** Termín odevzdání bakalářské práce: **15.06.2018**

Platnost zadání bakalářské práce: _____

doc. Ing. Jan Chyský, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

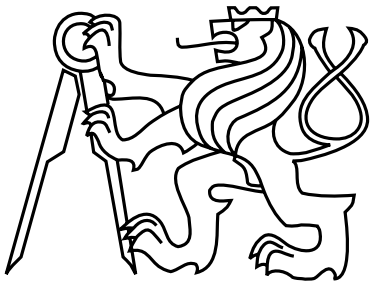
III. PŘEVZETÍ ZADÁNÍ

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

18. 4. 2018 _____
Datum převzetí zadání

_____
Podpis studenta

České vysoké učení technické v Praze
Fakulta strojní
Ústav přístrojové a řídící techniky



Bakalářská práce

Softwarový systém pro rehabilitační pomůcku

Kryštof Remeš

Vedoucí práce: doc. Ing. Jan Chyský, CSc.

Studijní program: Teoretický základ strojního inženýrství, Bakalářský

Obor: bezoborový

15. srpna 2018

Poděkování

Děkuji vedoucímu mé bakalářské práce panu doc. Ing. Janu Chyskému, CSc. za vstřícnost, cenné rady a věcné přípomínky. Děkuji všem konzultantům za rady při řešení specifických problémů, na které jsem narazil při řešení této práce.

Nejvíce bych chtěl poděkovat mámě za poskytnutí kvalitního zázemí pro studium, podporu při studiu a pevné nervy.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 15. 8. 2018 .....

Anotační list

Jméno autora	Kryštof Remeš	
Název práce	Softwarový systém pro rehabilitační pomůcku	
Anglický název	Software system for rehabilitation aids	
Vedoucí práce	doc. Ing. Jan Chyský, CSc.	
Rok	2018	
Studijní program	Teoretický základ strojního inženýrství	
Ústav	Ústav přístrojové a řídicí techniky	
Bibliografické údaje	počet stran	49
	počet obrázků	21
	počet příloh	3
Klíčová slova	Arduino, roztroušená skleróza, Parkinsonova choroba, rehabilitační zařízení, rehabilitace	
Keywords	Arduino, multiple sclerosis, Parkinson's disease, rehabilitation device, rehabilitation	

Abstract

The subject of this thesis is a development of a control software for a rehabilitation device for patients suffering from a multiple sclerosis and a Parkinson's disease. For acomplishing this purpose a research was conducted concerning a past development of the rehabilitation device alongside with a research of symptoms of these diseases and means of rehabilitation, and an analysis of requirements of the control software. As a result the control software was created for a microcontroller board Arduino MEGA, including two exercise modes, loading configuration specifications from a configuration file saved on a SD memory card and storing gathered data into an output file located on the SD card.

Abstrakt

Předmětem práce je vývoj řídicího softwaru rehabilitační pomůcky pro pacienty s roztroušenou sklerózou a Parkinsonovou chorobou. Za tímto účelem byla provedena rešerše vývoje rehabilitační pomůcky, seznámení se se symptomy onemocnění a způsoby jejich rehabilitace, a analýza požadavků na řídicí software. Výsledkem je vytvořený řídicí software pro mikrokontroler Arduino MEGA, se dvěma cvičebními módy, načítající konfigurační soubor z SD paměťové karty a zapisující data do výstupního souboru umístěného na SD kartě.

Obsah

Seznam obrázků

Seznam zkratek a výrazů

avr-g++ název typu kompilátoru používaného pro kompilaci programů pro Arduino

C/C++ programovací jazyky

CNS Centrální nervový systém

CR2032 Velikost/typ knoflíkových baterií

CSV Comma-separated Values, Data oddělená čárkou

Datasheet Technický list, Soubor vytvořený výrobcem obsahující technické informace o elektronické součástce

DATA datový výstup na přijímači přesného času DCF

DCF-77 viz. DCF; 77 - vysílací frekvence (77,5kHz)

DCF D - Deutschland (Německo), C - označení pásma dlouhých vln, F - frankfurtský region

DS3231 čip použitý v bateriovém modulu reálného času

EBV vir Epstein-Barrové

FAT32 32 bitová verze souborového systému FAT

FAT File Allocation Table, je druh souborového systému používaný k alokaci dat v paměti

FOR druh cyklu

GND Ground, označení zemnícího vodiče

GUI Graphical User Interface, neboli grafické uživatelské rozhraní

I^2C Inter-Integrated Circuit, sériová sběrnice vyvinutá firmou Philips

I/O Input/Output, Vstup/Výstup

IDE Integrated Development Environment, neboli vývojové prostředí pro platformu Arduino, používané ke tvorbě programů

IF funkce KDYŽ

int integer - celočíselný formát proměnné

LED Light Emiting Diode, neboli dioda vyzařující světlo ve viditelném vlnovém spektru

MASTER Řídicí zařízení z dvojice řídicí zařízení-řízený modul

MISO Master In Slave Out, datový vstup do ovládacího prvku, výstup z ovládaného prvku používaný pro SPI

MOSI Master Out Slave In, datový výstup z ovládacího prvku, vstup do ovládaného prvku používaný pro SPI

NDT neurovývojová terapie

PON výstup pro ovládání vypnutí/zapnutí na přijímači přesného času DCF

ppm parts per million, jednotka používaná k popisu odchylek

RS Rozstroušená skleróza

RTC Real Time Clock, neboli hodiny reálného času

SCK Serial Clock, signál časování vysílaný ovládacím prvkem

SCL Serial Clock, signál časování pro I^2C sběrnici

SCS / SS Slave Select, výstup ovládajícího prvku, používaný k vypínaní/zapínání vládaného prvku

SD karta Secure Digital paměťová karta, specifický formát paměťové karty vyvinutý společností SD Card Association

SDA Serial Data, signál datového přenosu

Sketch programový skript pro vývojovou desku Arduino

SLAVE Řízený modul z dvojice řídicí zařízení-řízený modul

SPI Serial-Parallel Interface, sériově paralelní rozhraní

SWITCH přepínač, který porovnáním hodnoty v proměnné spustí část kódu, která odpovídá příslušné hodnotě

TSV Tab-separated Values, Data oddělená tabulátorem

USB Universal Serial Bus, neboli univerzální sériová sběrnice

VCC Voltage Common Collector, napájení integrovaných obvodů

Kapitola 1

Úvod

1.1 Návaznost[1]

Tato bakalářská práce navazuje na *VONDRA, Jakub. Rehabilitační přístroj. Praha, 2016. Diplomová práce. České vysoké učení technické v Praze. Vedoucí práce Ing. Bc. Šárka Němcová, Ph.D.*. Zmíněná diplomová práce se zabývala návrhem a konstrukcí rehabilitačního přístroje primárně určeného pro pacienty trpící chorobami postihující pohybový aparát, přesněji jeho senzomotorické funkce, jako jsou roztroušená skleróza a Parkinsonova choroba.

1.2 Motivace práce

Přístroj by měl sloužit jako prostředek k udržování senzomotorických funkcí a pravidelným používáním by měl napomáhat ke zpomalení postupu nemoci. Vlivem fyzického pohybu s důrazem na rychlost reakce se udržuje a obohacuje myelinový obal neuronů, který zajišťuje průchod elektrických signálů nervovými vlákny. Základem účinnosti jakéhokoliv fyzického rehabilitačního procesu je určitá pravidelnost a četnost opakování. Právě tato podstata skýtá pro pacienty s tímto typem onemocnění mnoho problémů. Jednak nedostatek klinických zařízení a personálu, který by se mohl této formě rehabilitace aktivně věnovat s dostatečným nasazení a efektivitou. Zároveň symptomy těchto onemocnění zahrnují zhoršenou a progresivně se zhoršující mobilitu, zejména u pacientů s roztroušenou sklerózou. U Parkinsonovy choroby je na vině ve většině případů vyšší věk pacientů, který tyto problémy přirozeně přináší. Kombinace těchto dvou komplikací dala vzniknout myšlence lehce přemístitelného přístroje, který by si pacienti mohli zapůjčit a bez větších potíží jej používat v pohodlí svého domova. Existence takového přístroje by zpřístupnila možnost rehabilitací většímu počtu pacientů a mnohým by je ulehčila.

1.3 Cíl práce

V diplomové práci[1], na kterou navazuji, byla primárně vyřešena funkční konstrukce rehabilitačního přístroje s velice jednoduchým řídicím softwarem, který ve své podstatě sloužil pouze k ověření funkčnosti konstrukčního řešení. Cílem této bakalářské práce je vytvořit funkční řídicí software pro zmíněnou rehabilitační pomůcku. Pro splnění cíle je tedy třeba podrobněji zanalyzovat požadavky na řídicí software a to nejen z pohledu pacienta, ale i lékařů a firmy, která by tyto přístroje vyráběla.

Kapitola 2

Rehabilitační přístroj

2.1 Specifikace přístroje

2.1.1 Konstrukce

Přístroj byl navržen ve dvou koncepčních řešeních.

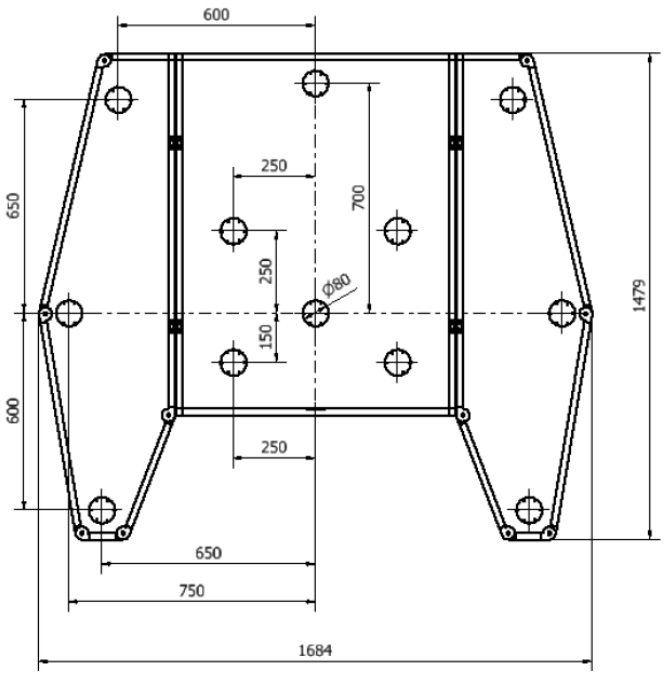
2.1.1.1 Koncept s kovovou konstrukcí[1]

První koncept je tvořen třídílnou kovovou konstrukcí - střední částí a dvěma křídly, která jsou ke střední části připevněna pomocí pantů. To umožňuje složení křídel přes střední část, zvyšuje kompaktnost celého zařízení a usnadňuje přemisťování zařízení. Složitelnost ve dvou bodech však zdaleka neuspokojuje požadavky na mobilitu celého zařízení.

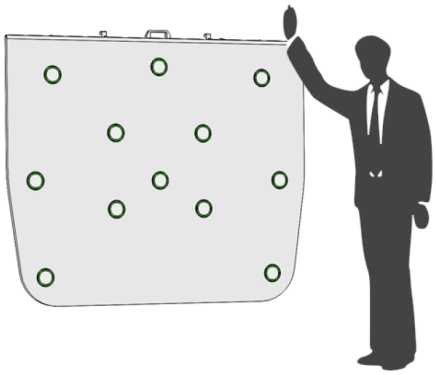
Přístroj obsahuje 12 tlačítek ve formě kapacitních senzorů, které jsou osazeny dvoubarevnými LED diodami červené a zelené barvy, kde střední část nese 6 tlačítek a křídla obsahují po 3 tlačítkách. Kovová konstrukce je kryta z přední a zadní strany deskami z čirého plexiskla. Tlačítka a elektronika jsou umístěny mezi těmito deskami.

2.1.1.2 Koncept z ohebného materiálu[2]

Druhý koncept je tvořen plastovým U profilem na vrchní části, povrchem ze snadno omyvatelného ubrusu z PVC na textilním základu a molitanovou výplní, v níž je uložena všechna elektronika a vedeny všechny potřebné vodiče. Počet tlačítek je stejný jako u předchozí varianty, tj. 12. Tento koncept byl vyvíjen z důrazem na výrazné zlepšení mobility zařízení a s myšlenkou domácího užívání.



Obrázek 2.1: Pevná varianta rehabilitační pomůcky



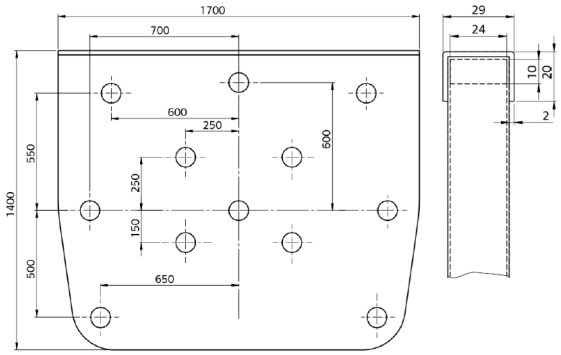
Obrázek 2.2: Vizualizace svinutelné varianty rehabilitační pomůcky

2.1.2 Elektronika

2.1.2.1 Tlačítka na bázi kapacitních snímačů

Při návrhu prototypu kovové konstrukce Jakub Vondra testoval různé varianty tlačítek. Prozkoumány byly následující varianty:

- Kapacitní softwarový senzor na Arduinu

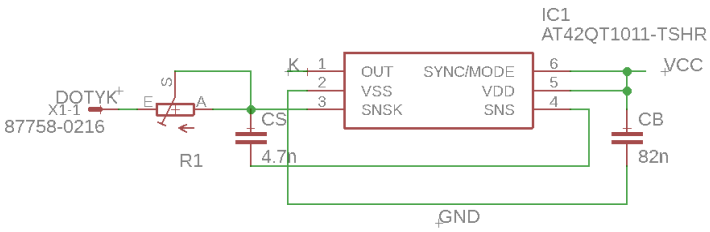


Obrázek 2.3: Schéma svinutelné varianty rehabilitační pomůcky

- Kapacitní senzor s dedikovanou elektronikou
- Mechanický menbránový senzor s mikrospínačem

Jako výsledné tlačítko byl zvolen kapacitní senzor s dedikovanou elektronikou.[1] Tento senzor byl později přepracován studenty magisterského studia v rámci předmětu Projekt II.[3]

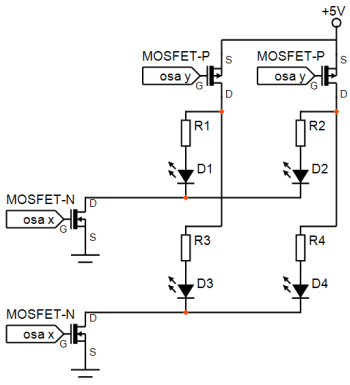
Tlačítka byla osazena čipy, který vyhodnocují signál ze snímače a dále posílají pouze logickou 1 nebo 0.



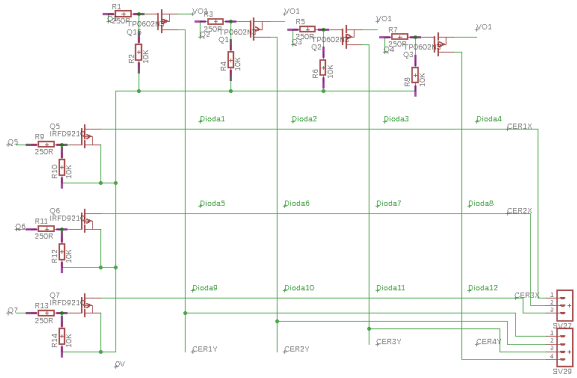
Obrázek 2.4: Schéma použitého tlačítka

2.1.2.2 Maticové zapojení

Přepracování elektronického zapojení bylo uskutečněno studenty magisterského studia v rámci školního projektu. Ke snížení množství vodičů, využili maticového zapojení LED diod do mřížky o velikosti 3x4. "Na obrázku dole je princip maticového uspořádání spínacích tranzistorů. Když chci například sepnout diodu D2, tak musím na druhý tranzistor MOSFET-P přivést logickou 0 a na první tranzistor MOSFET-N přivést logickou 1. Vždy v průniku (sepnutí) obou os se rozsvítí dioda."[3] Z toho vyplývá, že pro vypnutý stav jsou nutné logické 1 na tranzistorech MOSFET-P (výstupy 1-4) a logické 0 na tranzistorech MOSFET-N (výstupy 5-7).



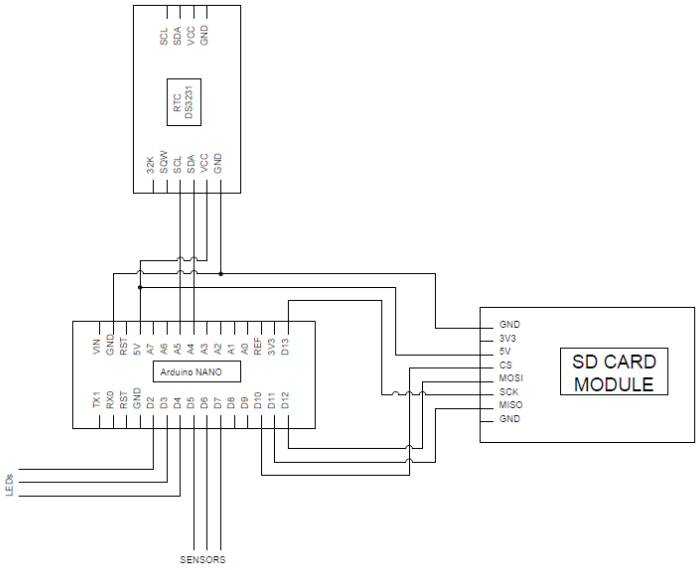
Obrázek 2.5: Princip maticového uspořádání spínacích tranzistorů pro ovládání LED



Obrázek 2.6: Schéma maticového uspořádání pro ovládání LED

2.1.2.3 Řídicí jednotka

Vytvořením maticového zapojení se otevřela možnost použít rozměrově menší mikrokontroler Arduino NANO, oproti původní verzi s deskou Arduino MEGA. Za současného použití programovatelného čipu, který využíval binárního adresování kapacitních snímačů pro další výrazné snížení počtu potřebných vodičů a vstupů/výstupů mikrokontroleru, viz 2.7. Tato varianta však byla následně zavržena po konzultaci se zástupci zadávající firmy a to z důvodu nutnosti stisku více tlačítek zároveň. Svinutelná varianta tedy počítá s použitím desky Arduino MEGA, viz 2.8.



Obrázek 2.7: Schéma vývojového vzorku použitého k ověření funkčnosti programu na základu Arduino NANO

2.2 Varianty konfigurací

2.2.1 Původní verze s Arduino MEGA [1]

Původní verze byla úplně první verzí rehabilitačního přístroje. Byla vyřešena zejména konstrukce a funkčnost celého přístroje. Výsledkem byl skládací koncept s kovovým rámem, řídící jednotkou bylo Arduino MEGA, a kapacitními snímači osazenými LED diodami. Vytvořený software zahrnoval 3 módy: ladění citlivosti kapacitních snímačů, cvičení s 5 tlačítky a cvičení se 12 tlačítky. Cvičící módy byly pouze formou smyčky s náhodnou aktivací tlačítek a zhasnutím po dotyku.

2.2.2 Verze Raspberry Pi + Python [4]

Od prvotního navržení a zkonstruování přístroje Jakubem Vondrou byla elektronická část zařízení upravena minimálně jednou, a to na verzi využívající mikropočítače Raspberry Pi jako řídicí jednotky. Tato úprava byla provedena dvěma zahraničními studenty, Shahirem Akuji a Felixem Hromatka, v rámci projektu v předmětu "Python for Scientific Computations and Control". Cílem projektu bylo vytvoření řídicího programu a grafického uživatelského rozhraní v jazyce Python. Z důvodu použití Raspberry Pi byla původní elektronika přístroje trochu upravena.

2.2.2.1 Řídicí program [4]

Program se skládal ze dvou skriptů:

AutoStart.py: Sloužil k bootování Raspberry Pi a načtení dat z konfiguračního souboru umístěného na flash disku.

Main.py: Obsahoval hlavní řídicí algoritmus cvičení. Program vybral náhodný senzor ze seznamu požadovaných senzorů, aktivoval jeho LED diody a čekal na stisk tlačítka. Při stisku tlačítka zaznamenal dobu mezi aktivací a stiskem tlačítka a zapsal ji do souboru. Pokud reakční doba byla kratší než zadaná doba odměny, uživatel byl odměněn rychlým zablikáním LED diod.

2.2.2.2 Grafické uživatelské rozhraní (GUI) [4]

GUI sloužilo primárně ke tvorbě konfiguračního souboru na stolním počítači, který by měl k dispozici lékař. Pro vytvoření konfiguračního souboru se do GUI daly zadat následující parametry: celé jméno pacienta, počet opakování (ve smyslu počtu zapnutí senzorů, tzn. specifikace délky cvičení), čekací doba (časové okno pro reakci uživatele na aktivaci tlačítka), čas odměny, seznam aktivních senzorů.

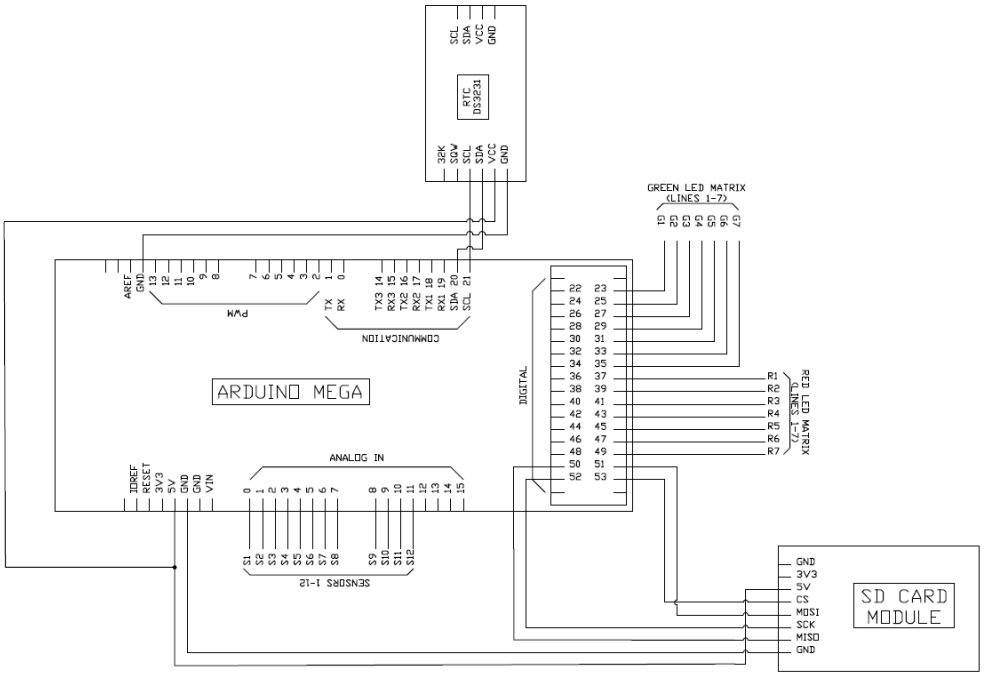
2.2.3 Zamýšlená verze s Arduino NANO

Při použití maticových zapojení pro obě barvy LED diod a programovatelného čipu pro binární adresování použitých snímačů se snížil počet potřebných vstupů/výstupů na 18 (2x7 pro LED diody a 4 pro snímače). To otevřelo možnost použití menší vývojové desky Arduino NANO, které disponuje 22 digitálními I/O a 8 analogovými vstupy. Bohužel v počtu digitálních I/O jsou započítány i analogové vstupy, které dají přenastavit a používat stejně jako vstupy digitální, a dva vstupy/výstupy používané pro sériovou komunikaci, které pro tuto funkci musí zůstat vyhrazeny. Výsledkem tedy je volných 12 čistě digitálních I/O a 8 analogových vstupů (po přenastavení digitálních I/O). Programovatelný čip produkuje výstupní signál ve formě logických 1 a 0. Pro snímače by tedy byly použity analogové vstupy používané jako digitální. Zbyly by tedy dva volné analogové vstupy.

Bohužel celý tento koncept nepočítal s nutností použití rozšiřujících modulů, které vyžadují další vstupy/výstupy. Nedostatek vstupů/výstupů by se ale dal obejít připojením rozšiřujícího modulu přes I^2C sběrnici, která komunikuje přes jediné 2 I/O, které zbyly volné. Zda by se tímto způsobem daly připojit paměťové moduly pro SD kartu nebo USB, si ale nejsem jist, a to z důvodu relativně nízkých přenosových rychlostí této sběrnice.

2.2.4 Aktuální verze s Arduino MEGA [2][3]

Aktuální svinutelná verze byla vytvořena ve spolupráci Jana Soukala, Jana Riedla a Martina Cahyny, kde Jan Soukal navrhl a vytvořil konstrukci svinutelného konceptu, který je osazen elektronikou vytvořenou v rámci Projektu II již zmíněnými Janem Riedlem a Martinem Cahynou. Tato verze tedy funguje na desce Arduino MEGA s maticovým zapojením LED diod a přímým zapojením snímačů na digitální I/O desky Arduino MEGA.



Obrázek 2.8: Schéma aktuálního zapojení řídicí jednotky s Arduino MEGA

Kapitola 3

Onemocnění nervové soustavy

3.1 Roztroušená skleróza [5]

3.1.1 Patologie RS

"Podstatou nemoci je zánět s autoimunitními rysy. Cílem autoimunitního útoku je myelin, obalující některé nervové dráhy v centrálním nervovém systému (CNS), tedy mozku a míše. V zánětlivých ložiscích jsou kromě myelinu ničena v různé míře nervová vlákna. Ztráta nervových vláken je u roztroušené sklerózy podstatou invalidity."

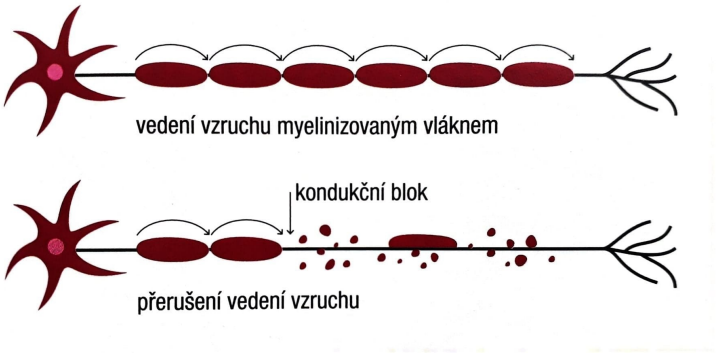
"Imunitní systém zajišťuje své funkce pomocí mnoha nespecifických i specifických mechanismů. Pro přehlednost dělíme specifickou imunitu na buněčnou a humorální. Buněčná je zajišťována především T lymfocyty, humorální je zajišťována protilátkami. Ty jsou tvořeny plazmocyty, což jsou vyzrálé B lymfocyty.

Během vývoje jedince je zajištěno, že většina lymfocytů, které by rozpoznaly velmi agresivně naše vlastní tkáně, je zlikvidována v brzlíku. Ty, které bz to udělaly s menší razancí. jsou uvedeny do stavu spánku, aby se nemohly množit. Za určitých okolností jsou však tyto lymfocyty probuzeny."

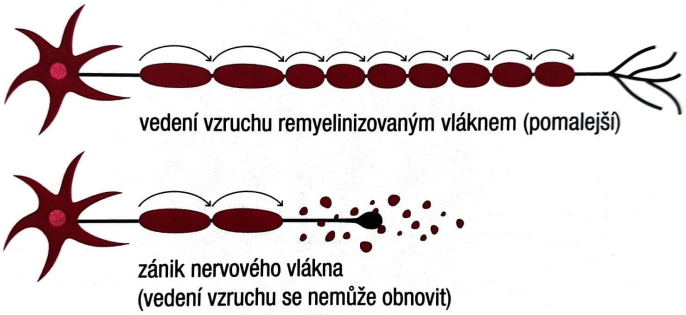
V případech, kdy dojde k oslabení imunitního systému například stresem, bojem s jinými infekčními chorobami, kouřením anebo třeba nedostatkem vitamínu D, může dojít k aktivaci spících lymfocytů a tím ke spuštění onemocnění.

V zánětlivých ložiscích jsou lymfocyty aktivovány antigeny myelinu a začnou jej aktivně napadat a ničit. To vede rozpadu myelinového obalu axonů nervů, viz obrázek 3.1. CNS je schopna tento myelinový obal částečně opravit, avšak takto opravené nervové vlákno vede vzruchy, ve srovnáním s původními vlákny, pomaleji, viz obrázek 3.2 . Výsledná myelinová vrstva je tenčí než původní. U

onemocnění v pokročilém stádiu k remyelinizaci již nedochází. V mnoha případech jsou také nervová vlákna zničena úplně a jejich obnovení není možné.



Obrázek 3.1: Vedení vzruchu myelinizovaným nervem a blokáda vedení (kondukční blok) při demyelinizaci[5]



Obrázek 3.2: Remyelinizace a zánik axonu[5]

3.1.2 Klinické příznaky

3.1.2.1 Optická neuritida

Jedná se o zánět očního nervu, který se projevuje bolestí při pohybu oka a poruchami zraku ve formě mlhavého vidění, výpadku zorného pole, změny barevného vidění (barvy zešednou, obzvláště červená). "Rozvoj obtíží trvá hodiny až dny, není náhlý. Málokdy dojde k úplné ztrátě zraku. Pro postižení optického nervu u RS je typický jednostranný zánět."

3.1.2.2 Senzitivní poruchy

"Pocity snížené, zvýšené anebo jiné citlivosti (hypestezie, hyperstezie anebo parastezie) na různých částech těla jsou velmi časté, a to i na začátku nemoci. Časté jsou pocity mravenčení, změna vnímání tepla, ale i nepříjemné pocity jako pálení a bodání."

3.1.2.3 Motorické poruchy

"Vznik motorických příznaků je vždy prognosticky horším znamením než výskyt senzitivních poruch."Motorické poruchy zasahují jak dolní tak horní končetiny. U horních končetin se zpravidla jedná o projevy, které na první pohled působí jako neobratnost a zpomalení pohybu. "Může však jít i o vznik jednostranné hemiparézy (jednostranné postižení horní a dolní končetiny)."U dolních končetin se nemoc projevuje spastickou paraparézou, který je nejčastějším projevem v pozdějších stádiích nemoci. Jedná se o poruchu chůze omezující pacienta v dosahu a jistotě chůze, neschopnost popoběhnout či poposkočit na jedné nebo i na obou nohách. Jelikož jsou tyto poruchy snadno popsatelné a měřitelná, využívá se jich při neurologických vyšetřeních.

Tyto poruchy jsou hlavní předmětem zaměření rehabilitačního přístroje.

3.1.2.4 Ostatní

Mezi další příznaky patří:

Mozečkové poruchy: zhoršení jemných pohybů končetin, rovnováhy a koordinace

Sfinkterové (svěračové) poruchy: se projevují například jako urgence močení a pocity neúplného vymočení.

Kmenové syndromy: možnost vzniku paréz, poruch citlivosti, okohybných poruch, obrny lícního nervu

Únava, deprese a kognice

Vzácné příznaky: epilepsie, afázie (porucha řeči)

3.1.3 Výskyt

"Roztroušená skleróza (RS) je v rozvinutých zemích nejčastější příčinou progresivní neurologické invalidity u mladých nemocných. Pacienti mají akutně vzniklé neurologické obtíže nebo pozvolný rozvoj neurologické disability."

"Onemocnění se vyskytuje častěji u žen a u indoevropské populace. Prevalence v populaci (výskyt na 100 000 obyvatel) je kolem 1-2 ‰., pokud se roztroušená skleróza vyskytuje v příbuzenstvu, zvyšuje se prevalence na 3-4%"

"Za prokázané rizikové faktory se považuje infekce virem Epsteina-Barrové (EBV), kouření a nedostatek vitaminu D. Poslední faktor vysvětluje, proč onemocnění přibývá směrem k pólům a proč nejnižší prevalence je na rovníku."

3.1.4 Metody fyzioterapie

3.1.4.1 Specifické postupu - neurorehabilitace

"Terapie na neurofyziologickém podkladě má z hlediska fyzioterapie pro pacienta s RS velký význam a je potřeba ji aplikovat ihned od počátku onemocnění. Využívá se při ní významné vlastnosti CNS - neuroplasticity. Je to schopnost CNS reagovat a přizpůsobit se novým podnětům funkční i strukturální přestavbou. Vhodnou a opakovanou stimulací pomáháme najít nepoškozené mozkové oblasti a využít je pro částečnou opravu porušené funkce."

Nejčastěji používanými technikami jsou například: neurovývojová terapie (NDT), senzomotorická stimulace, reflexní lokomoce (Vojtův princip). Právě na principu posledních dvou zmíněných je postavena funkce rehabilitačního přístroje.

3.1.4.2 Vlastní pohybová akivita

Vlastní pohybová aktivita zahrnuje aerobní aktivity, posilovací cvičení, kombinovaný trénink, zdravotně-rehabilitační cvičení, plavání a cvičení ve vodě. Fyzická aktivita se dá realizovat individuálně i skupinově. Skupinová cvičení však prokazují lepší účinnost vlivem kolektivní motivace, sociálních vazeb a vzájemné psychické podpory.

3.1.4.3 Intenzita cvičení

Pacienti, jenž byli v minulosti zvyklí na pravidelnou fyzickou aktivitu, by měli své návyky udržovat, ale neměli by se zbytečně přetěžovat a je potřeba se naučit vnímat hranice svých schopností.

Dle obecného zjednodušeného doporučení by pravidelná pohybová aktivita měla být provozována přibližně dvakrát až čtyřikrát týdně v délce 30 až 90 minut.

3.2 Parkinsonova choroba [6]

3.2.1 Patologie

"Parkinsonova nemoc je neurodegenerativní onemocnění, jehož příčiny nejsou dosud jasné. Souvisí s postupným zánikem neuronů, který začíná v dospělém věku a pomalu se rozšiřuje a je dílčí, protože s týká jen některých populací neuronů, konkrétně neuronů struktury zvané substantia nigra, nacházející se v mozkovém kmeni ve středním mozku."

Tato struktura zajišťuje vstup nervového vzruchu do těchto nervových seskupení a zásobuje dopaminem část bazálních ganglií zvanou striatum. Neurony této části mozku jsou dopaminergní, tzn. přenašečem nervového vzruchu je dopamin. Bazální ganglia kontrolují motoriku.

"Parkinsonova nemoc je tedy definována jako ztráta dopaminergních neuronů v oblasti bazálních ganglií."

3.2.2 Klinické příznaky

3.2.2.1 Akineze

Akineze je hlavním avšak nejméně známým projevem nemoci. Vyskytuje se v různých úrovních. Projevuje se zpožďováním při zahájení a vykonávání pohybu, tzv. bradykineze a hypokineze a může vést až k celkovému snížení hybnosti, např. při gestikulaci. Pacient má problém s koordinací a s přecházením z jednoho pohybu k jinému.

"Postižená osoba popisuje akinezi jako slabost, únavu, pomalost nebo neschopnost vykonat nějaký pohyb, udělat dvě věci zároveň. Postižený má občas pocit ztuhlosti, jako by mu "odumřela"končetina. Z hlediska vnějšího pozorovatele působí akineze jako nehybnost a ztuhlost - ty jsou pro nemoc velmi typické."

Postiženy jsou také jemné motorické funkce a chůze. Pacienti mají problémy s úkony jako jsou zapínání knoflíků, zavazování tkaniček, uvazování kravaty nebo hledání drobných mincí v peněžence.

3.2.2.2 Rigidita (svalová ztuhlost)

Zvýšená svalová ztuhlost se dá pozorovat u pacienta v uvolněném stavu, jehož končetinami hýbe druhá osoba. Projevuje se zvýšením klidového napětí svalů, které kladou odpor v celém rozsahu aktivního i pasivního pohybu příslušné části těla. Pacienti ji popisují jako strnulost a svalové nebo šlachové bolesti.

3.2.2.3 Třes

Třes nemusí být nutně hlavním projevem nemoci. Nejčastěji se vyskytuje v horních končetinách, hlavně rukou a prstech za uvolněného stavu nebo při vysokém soustředění. Může se však objevit i v končetinách dolních (chodidlech). Při pohybu převážné mizí.

3.2.2.4 Psychické poruchy

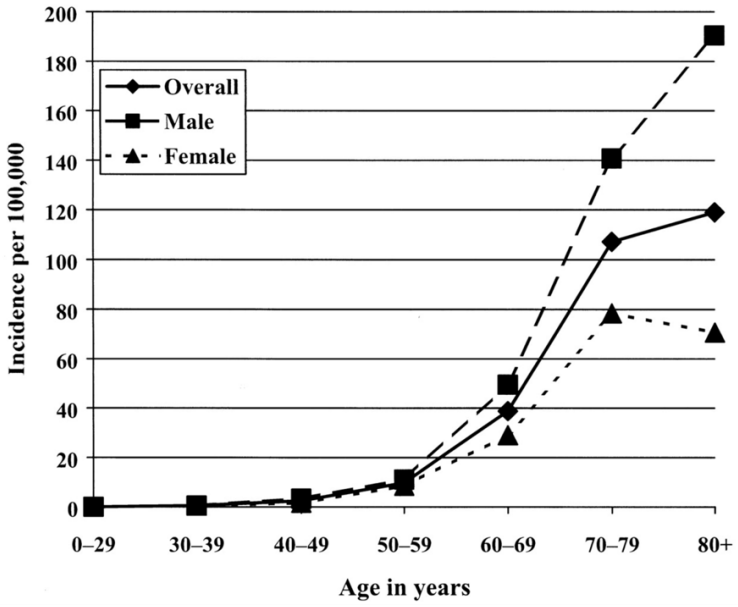
Nemoc je často doprovázena psychickými poruchami ve formě depresí, úzkostných stavů a dokonce i stavů psychotické povahy, jako jsou halucinace, bludy a zmatenost. Výjimkou nejsou ani spánkové a sexuální poruchy.

3.2.3 Výskyt

"Parkinsonova nemoc postihuej 84-187 osob ze 100 000 obyvatel, tedy přibližně každý tisící čověk trpí touto chorobou.

Hlavním rizikovým faktorem je věk, zánik dopaminergních neuronů však není způsobe stárnutím. Takže u osob nad 65 let postihuje 1,4 osoby ze 100."

Incidence, neboli počet nových případů onemocnění, je 10-15 osob ze 100 000 za rok. Z grafu 3.3 zjišťujeme, že výše zmíněné číslo přibližně odpovídá věkové skupině 50-59 let. Pro věkovou skupinu 60-69 let se ale incidence zvyšuje na přibližně 40 osob ze 100 000, a pro kategorii 70-79 dokonce na 105-110 osob ze 100 000. Je třeba si povšimnout rozdílu mezi muži a ženami. U mužů se onemocnění vyskytuje téměř dvakrát častěji než u žen.



Obrázek 3.3: Výskyt Parkinsonovy choroby v závislosti na věku a pohlaví[6]

3.2.4 Fyzioterapie

"Během tohoto *vyváženého období*[1] se doporučuje, aby nemocný udržoval, případně i zlepšoval fyzickou kondici, sportoval (chůze. jóga, plavání atd.) a začal s fyzioterapií, s tělesnými cvičeními individuálně nebo ve skupině. Rehabilitace pomáhá kompenzovat pohybové poruchy."

"Cvičení se zaměřuje na cviky pro udržování svalové aktivity a kloubních pohybů. Dechová cvičení zase pomáhají při relaxaci."

Rehabilitace může mít také analgetický vliv na krční a bederní bolesti a na bolesti sedací kosti.

[1] Vyvážené období - období, kdy je nemoc stabilizována a nedochází k výrazné progresi, tj. málo motorických a psychologických změn

Kapitola 4

Analýza požadavků

4.1 Koncepce softwaru

Softwarový balíček má v konečné verzi vypadat podobně jako již existující verze projektu vytvořeného zahraničními studenty pro Raspberry Pi. Bude obsahovat program na stolní počítač, který by byl k dispozici lékaři (dále jen GUI), a řídicí program nahraný v mikrokontroleru ARDUINO.

GUI bude sloužit k přístupu do databáze pacientů, jako generátor konfiguračního souboru pro řídicí program mikrokontroleru a zároveň zprostředkuje vyhodnocování výstupních dat z mikrokontroleru. Po vyhledání pacienta se zobrazí konfigurace cvičení, které pacient v minulosti absolvoval a přístup ke zpracovaným a nasbíraným datům, jako např. průběh (změna) reakčních dob pacienta v závislosti na čase a srovnání reakčních dob pacienta vzhledem k jednotlivým tlačítkům. Takto zpracovaná data poskytnou lékaři potřebné informace k posouzení stavu pacienta, rychlost postupu nemoci, nejvíce postižené oblasti a míru funkčnosti motorických funkcí. V záložce pro vytvoření konfiguračního souboru se objeví možnost upravovat tyto parametry:

- Požadované senzory
- Časová délka cvičení
- Časové okno odměny (odměnou bude rychlé zablikání stisknutého tlačítka, při splnění požadovaného časového limitu)
- Časové okno pro reakci na aktivovaný senzor
- Přednastavené módy (např. aktivaci pouze středního panelu, pouze pravého křídla, rychlost cvičení atd.)

Řídicí program bude tvořen následující segmenty:

- Inicializace datového nosiče (SD karty, USB flash disku)
- Načtení proměnných z konfiguračního souboru
- Chybová signalizace při nepovedeném načtení konfiguračního souboru
- Signalizace úspěšného načtení konfiguračního souboru
- Signalizace, že zařízení je připraveno ke cvičení
- Start cvičení aktivací jednoho senzoru anebo kombinací více senzorů
- Signalizace přijetí signálu o startu cvičení a spuštění cvičení
- Spuštění algoritmu podle zvoleného módu cvičení
- Záznam doby reakce a čísla senzoru na nějž bylo reagováno
- Zápis zaznamenaných dat do souboru na datovém nosiči
- Signalizace konce cvičení

4.2 Požadované algoritmy (cvičební módy)

4.2.1 Zhasínání s měřením doby reakce

Tento základní cvičební algoritmus náhodně vybírá ze seznamu zvolených senzorů. Příslušné tlačítko aktivuje rozsvícením LED diod a očekává reakci od uživatele ve formě stisknutí tlačítka. Při stisknutí nesprávného snímače správné tlačítko varovně zabliká a zhasne. Při stisknutí správného tlačítka ve stanoveném časovém intervalu pro odměnu bude uživatel odměněn zablikáním diod tlačítka. Při stisknutí v intervalu pro reakci tlačítko zhasne. Ve všech třech případech následuje opakování cyklu, tj. aktivace jiného náhodného tlačítka ze seznamu. Na konci každého průběhu cyklem program zaznamená reakční dobu pacienta a tu s příslušným číslem snímače uloží do textového souboru na datový nosič pro účely pozdějšího vyhodnocení lékařem. Zároveň porovná aktuální délku cvičení s požadovanou maximální dobou cvičení. Pokud je délka cvičení rovna nebo větší než doba požadovaná, cvičební algoritmus se vrátí na začátek, kde očekává stisknutí tlačítka pro zahájení cvičebního programu.

4.2.2 Zhasínání s držením jiného tlačítka

Jelikož se předpokládá použití pro pacienty v různých stádiích onemocnění, je třeba vytvořit i náročnější cvičební program pro pacienty v méně pokročilých fázích onemocnění. Druhým cvičebním módem tedy je "Zhasínání s držením jiného tlačítka". Při tomto cvičení pacient jednou rukou přidržuje tlačítko rozsvícené např. červenou barvou, které při aktivaci a čekání na dotyk bliká, při dotyku svítí, a mezitím druhou rukou reaguje na rozsvěcení jiných tlačítek zelenou barvou. Po uplynutí určité doby červené tlačítko zhasne a rozsvítí se červeně jiné. Pro rozsvěcování zelených tlačítek se dá použít modifikovaného základního algoritmu včetně principu odměn.

4.2.3 Zhasínání jedné barvy a počítání druhé

Třetí cvičebním módem je varianta, kdy uživatel zhasíná jednu ze dvou barev a počítá kolikrát se během procesu rozsvítila barva druhá. Tato varianta je poněkud náročnější na provedení. A to kvůli nalezení efektivního způsobu kontroly počtu rozsvícení druhé barvy, kterou uživatel napočítal.

Nabízí se následující možnosti:

1. Využít tohoto cvičení pouze za dozoru druhé osoby která zná správný (nastavený) počet rozsvěcení, což ale znemožňuje náhodně proměnný počet rozsvěcení, který by si vygeneroval sám program.

2. Po skončení cvičení na jeden ze senzorů "vyťukat"napočítané číslo, jiným senzorem tento počet odeslat/potvrdit a následně by program ověřil, zda se zadané číslo shoduje. Pokud by se počty shodovaly, mohlo by to být potvrzeno zeleným zablikáním středového snímače. Pokud by se počet lišil mohlo by se rozsvítit tlačítka jejichž počet by znázorňoval rozdíl mezi zadanou a skutečnou hodnotou, kde barva by znázorňovala znaménko. Tři červené snímače by tedy znamenaly, že zadaná hodnota byla o 3 menší než skutečná. Toto provedení se však zdá pro uživatele náročné na ovládání a zapamatování v jaké fázi se algoritmus nachází. Orientace uživatele ve fázi algoritmu by se však dal vyřešit připojením jednoduchého displeje, který by uživateli poskytoval potřebné instrukce. K displeji je možnost připojit jednoduchou klávesnici, která by zadávání napočítaného počtu značně ulehčila.

4.3 Zhodnocení a volba řídicího systému

Vzhledem k požadované ceně zařízení a náročnosti řídicího programu bylo rozhodnuto, že pro konstrukci vývojového vzorku bude nejvhodnější platforma Arduino, která je spíše mikrokontroler,

na rozdíl od Raspberry Pi, které je ve své podstatě sice kompaktní ale plnohodnotný počítač. Značnou výhodou je také opensource licence této platformy, a tudíž veřejná přístupnost ke všem datasheetům potřebným k výrobě vlastních desek. Dostupnost těchto informací otevírá možnosti k vytvoření nové speciální desky na základě Arduino, do níž by se integrovaly všechny - jinak externí - moduly potřebné k požadované funkčnosti celého rehabilitačního přístroje.

Kapitola 5

Realizace

5.1 Jazyk a prostředí

5.1.1 Arduino IDE[7]

Vývojové prostředí Arduino IDE je velice jednoduché na používání a velice přehledné. Jeho výhodou oproti jiným dostupným vývojovým prostředím je databáze vzorových sketchů, díky nimž může nový uživatel snadno začít a pochopit základní principy ovládání vývojových desek Arduino. Tyto vzorové příklady obsahují témata od jednoduchého blikání vestavěné LED diody, přes použití jednoduchých tlačítek až ke vzorům práce s moduly SD karet anebo moduly reálného času (RTC). Dále disponuje možností volby typu vývojové desky Arduino a při kompilaci zobrazuje procentuální využití pamětí po nahrání sketche do zvolené vývojové desky Arduino.

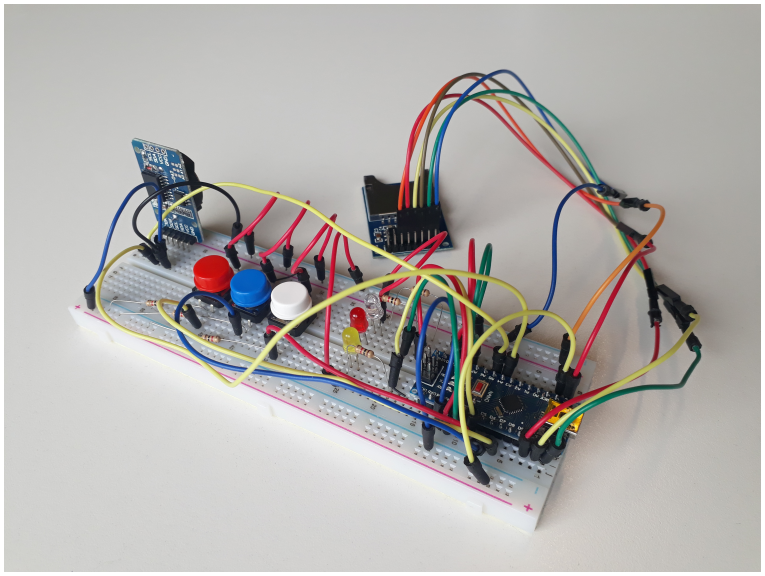
5.1.2 Arduino jazyk[8]

Pro programovací jazyk Arduino je použita kombinace programovacích jazyků C/C++. V podstatě je tvořen souborem C/C++ funkcí, které jsou zavolány vytvořeným kódem. Napsaný sketch (jak se říká programům v Arduino jazyce) je mírně upraven a poté přímo kompilován C/C++ kompilátorem avr-g++. Všechny standardní C a C++ formulace, které jsou podporovány avr-g++ kompilátorem, by měly v Arduinu fungovat.

5.2 Testovací hardware

V průběhu tvorby jednotlivých funkčních částí řídicího programu, bylo zapotřebí ověřovat jejich funkčnost. Pro tento účel jsem využíval nepájivé kontaktní pole společně se základními komponenty, jako jsou LED diody a tlačítkové spínače. Zapojením všech komponentů, tak aby odpovídali struktuře rehabilitačního přístroje, jsem vytvořil vývojový vzorek, který funkčně odpovídal velice zjednodušenému modelu rehabilitačního přístroje. Relativně komplikovaná struktura kapacitních tlačítek byla nahrazena jednoduchými spínacími tlačítky a dvoubarevné LED diody byly nahrazeny jednobarevnými LED diodami s barvami odpovídajícími barvám tlačítek.

Jako řídící jednotku jsem použil Arduino NANO, k níž jsem přes nepájivé kontaktní pole připojil 3 tlačítka a k nim příslušné diody. Později byl tento hardware obohacen o rozšiřující modul SD karet a modul reálného času RTC s čipem DS3237.



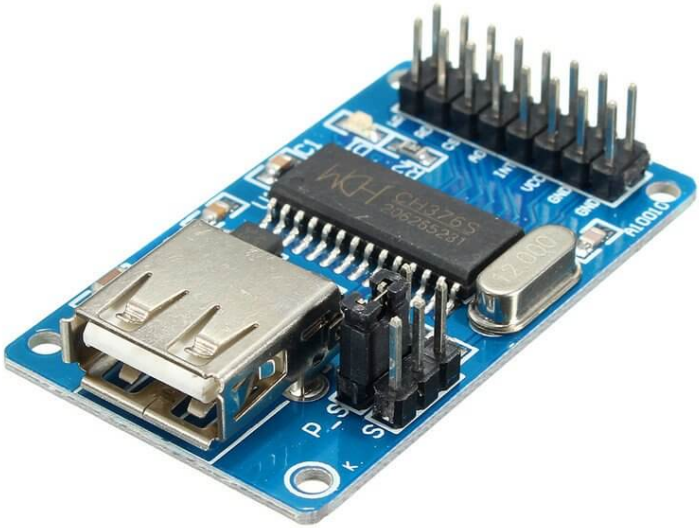
Obrázek 5.1: Použitý testovací hardware

5.2.1 Moduly paměťového úložiště

5.2.1.1 USB Modul CH376S[9]

USB modul osazený čipem CH376s poskytuje rozšíření o USB port, tudíž poskytuje možnost použití USB flash disku jako úložiště pro konfigurační a výstupní datový soubor. Použití USB flash disku jako paměťového úložiště je v naší aplikaci žádanější, a to hlavně z důvodu snadnější manipulace se samotným médiem, jehož rozměry obvykle bývají větší než u paměťových karet typu SD. Zároveň je

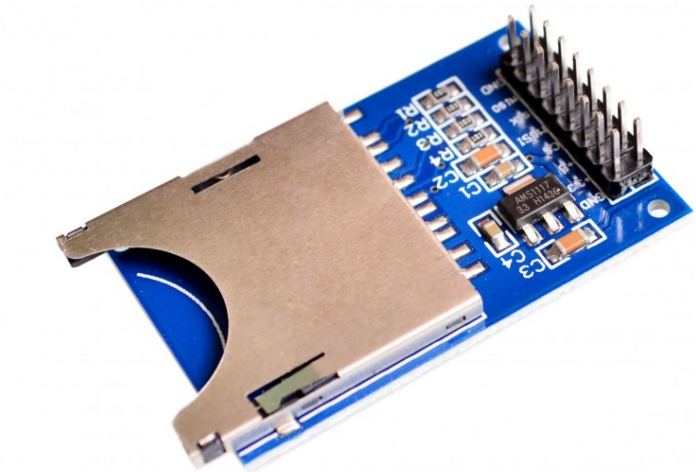
to i neznámější a nejpoužívanější prostředek pro fyzický přenos a uchovávání dat. Dle datasheetu[9] k čipu CH376S je možné pro komunikaci se zařízení v USB portu použít sériové (UART), seriově-paralelní (SPI) i paralelní komunikace. USB flash disk musí být ve formátu FAT12, FAT16 nebo FAT32. Při pokusech o zprovoznění jsem používal komunikačního protokolu UART a to z důvodu dostupnosti neoficiálních knihoven/návodů. Tento protokol využívá pro komunikaci dvou I/O: TXD - datový výstup a RXD - datový vstup. Dále pak je potřeba pouze napájení (VCC) a uzemnění (GND). Další informace o zprovozňování tohoto modulu jsou uvedeni v sekci 5.4.1.



Obrázek 5.2: USB modul CH376S[10]

5.2.1.2 Modul pro paměťové SD karty

Modul pro SD paměťové karty využívá sériově-paralelního (SPI) rozhraní pro komunikaci s mikrokontrolerem. SPI rozhraní využívá 4 I/O: MISO, MOSI, SS, SCK. MISO a MOSI jsou využívány pro přenos dat, jako třeba čtení/zápis, mezi MASTER (obvykle mikrokontroler) a SLAVE (v tomto případě SD modul). SS, neboli Slave Select, je využíván k vybírání momentálně aktivních SLAVE zařízení, což umožňuje zapojení více zařízení přes stejné rozhraní. SCK, neboli Serial Clock, slouží k synchronizaci datových přenosů.[11] SD pamětová karta musí být ve formátu FAT/FAT32.



Obrázek 5.3: Modul paměťových karet SD[12]

5.2.2 Moduly reálného času

5.2.2.1 DCF77 - Příjímač přesného času z Německa

Modul DCF-1 slouží k přijímání času vysílaného z vysílače v Německu, přesněji ve Frankfurtu. Čas je vysílán na dlouhých vlnách o frekvenci 77,5 kHz. Stanice, z níž se vysílá, disponuje trojicí atomových hodin. Čas těchto hodin se relativně odchyluje o méně než $2 \cdot 10^{-12}$ za den. Informace jsou vysílány v průběhu minutových intervalů, jak napovídá obrázek 5.5.[13]

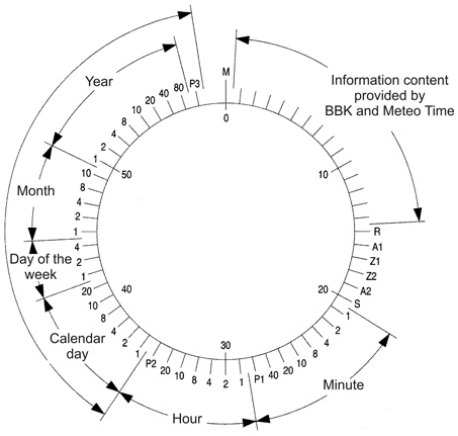
Obrázek 5.5 popisuje, které informace jsou posílány v daný moment. Všechny časové údaje, počínaje vteřinami konče roky, jsou vysílány ve formátu BCD (Binary Coded Decimal). Kružnice je rozdělena na 60 dílů, kde každý dílek odpovídá jedné sekundě. V intervalu 15 až 20 jsou vysílány informace o nepravidelnostech, jako třeba letní/zimní čas, přestupná sekunda. V intervalu 21 až 28 jsou vysílány sekundy, 29 až 35 je vyhrazeno pro minuty, 36 až 41 pro kalendářní dny, 42 až 44 pro den v týdnu, 45 až 49 pro kalendářní měsíc a nakonec 50 až 57 pro poslední dvojčíslí kalendářního roku.[14]

Samotný modul je spojen s Arduinem pomocí 4 konektorů (GND - uzemnění, VCC - napětí, DATA - přenos dat, PON - zapnuto/vypnuto). Modul využívá napětí v rozmezí 1,2-3,3V (obvykle

3V), odebíraný proud je menší než $90\mu A$ a je schopen pracovat při teplotách od -40°C do 85°C.[15]

Velikou nevýhodou tohoto modulu je však jeho inicializace, při které navození spojení s vysílačem přesného času obvyklé trvá 2-5 minut, ale při špatném signálu se může jednat až o 30 minut.[16] Pro udržení časové informace je zapotřebí smyčkového algoritmu, který bude kontinuálně přijímat informace vysílané vysílačem. Tato přijímací smyčka by značně zkomplikovala tvorbu a výslednou strukturu řídicího algoritmu mikrokontroleru, proto jsem variantu jeho použití zavrhl.

Obrázek 5.4: Přijímač přesného času z Německa - DCF77[17]



Obrázek 5.5: Sekvence datového toku vysílání[14]

5.2.2.2 Modul RTC (Real time clock) - DS3231

Rozměrově malý a kompaktní modul reálného času, s vlastním bateriovým napájením (baterie typu CR2032), se velice často využívá v aplikacích jako třeba domácí meteostanice a v jiných zařízeních, jenž pracují s časovými/datovými informacemi. Jakmile je čas u tohoto modulu jednou nastaven při prvotním sestavení zařízení, na bateriovém napájení je schopen udržet nastavený čas až 2 roky. Odchylka kmitů oscilátoru je <2 ppm (parts per million), což je ekvivalentní 0,17 sec/den resp. 1,05 min/rok při teplotě 25°C.[15][18] Odchylka v řádech desetin sekund na den je v naší aplikaci kompletně zanedbatelná, jelikož pro indexování výstupních souborů nám stačí přesné datum a čas na minuty. Z kódovací perspektivy je tento modul také velice výhodný, jelikož nepotřebuje smyčkový algoritmus v samotném mikrokontroleru, nýbrž pouze příkaz kterým se modulu zeptáme na aktuální časový údaj. Modul podporuje datové rozhraní I^2C, které využívá dvou datových vodičů (SCL a SDA) a je připojen k mikrokontroleru ještě vodiči VCC (napájení) a GND (uzemnění).[15]

Obrázek 5.6: Bateriový modul reálného času[19]

5.2.3 Zhodnocení - Arduino NANO vs. Arduino MEGA

Při původním návrhu byla snaha minimalizovat výsledné rozměry řídicí jednotky, tudíž byla zvolena vývojová deska Arduino NANO, která disponuje 32 piny z nichž je 12 digitálních a 8 analogových.[20] Teoreticky měl počet potřebných pinů pro vstupy a výstupy vystačit požadavkům ze strany rehabilitačního přístroje. Při použití maticového zapojení LED diod by na ovládání všech dvoubarevných diod připadlo 2x7 pinů a pro signály snímačů bylo zapotřebí 4 pinů, jejichž kombinace by reprezentovala číslo snímače v binární soustavě. Dohromady tedy 18 pinů. V průběhu vývojového procesu se ale ukázalo, že kvůli přidaným modulům počet pinů Arduina NANO nestačí. Modul SD karet využívá pro komunikaci s Arduinem sériového periferního rozhraní (SPI), která vyžaduje 4 datové vodiče (MISO, MOSI, SCK, SCS).[11] Bateriově napájený modul reálného času DS3231 zase využívá I^2C sběrnice, která vyžaduje pouze 2 datové vodiče (SCL a SDA).[15] Tato sběrnice by umožňovala připojit více zařízení, bohužel však není podporována modulem SD karet a to z důvodu nízké přenosové rychlosti.

Při vývoji rolovatelné varianty byla použita deska Arduino MEGA, která má pro naše použití více než dostatek pinů (54 digitálních I/O a 16 vstupních analogových pinů).[21] Jedním z důvodů výběru varianty MEGA, byl požadavek výrobce na možnost stisku více tlačítek zároveň, což znamenalo nemožnost použití adresování snímačů jako binární kombinace a tudíž nutnosti většího počtu výstupních pinů. Proto je výsledný program upraven pro použití s deskou Arduino MEGA.

5.3 Knihovny a sketche

5.3.1 Sketche

5.3.1.1 Správa SD karty

Značnou výhodou platformy Arduino je velice snadná dostupnost ukázkových příkladů funkčních kódů pro různé aplikace.[22] Tím je každému novému uživateli znatelně ulehčeno počáteční učení správné manipulace s Arduino platformou. Vzorový kód pro práci s SD kartami je samozřejmostí, jelikož se jedná o nejčastější datový nosič používaný s deskami Arduino. [23] Části výše uvedeného sketche byly použity ve výsledném řídicím programu.

5.3.1.2 Hodiny reálného času RTC

Oficiální podklady k modulům reálného času na webu Arduino se zabývaly pouze aplikací pro u desek s jiným druhem procesoru.[24] Po chvilce hledání se mi podařilo najít tutorial k RTC modulu, který

obsahoval i použitý kód. [25] Nalezené podklady jsem upravil tak, aby získané informace z modulu byly zapisovány přímo do výstupního souboru DATALOG. Úprava byla velice jednoduchá, poněvadž zahrnovala pouze redukci použitého kódu a změnu odkazu k výstupu ze sériové komunikace na výstup do textového souboru. Použitý sketch z tutorialu bude sloužit pro budoucí nastavování aktuálního času RTC modulu. Nastavování nebylo zahrnuto do výsledného řídícího programu. Důvodem je, že v budoucnu budou tyto časy nastavovat technici. Usoudil jsem, že je praktičtější, aby se do hlavního řídícího programu nijak nezasahovalo, a reálný čas se nastavoval, jednoduchým, mnohonásobně přehlednějším sketchem určeným pouze k ovládání RTC modulu.

Pro nastavení aktuálního času je potřebná malá úprava samotného kódu. Přesněji je zapotřebí pouze zadat argumenty do funkce `setDS3231time()`, a to ve formátu "sekundy, minuty, hodiny, pořadové číslo dne v týdnu, den, měsíc, rok". Týden začíná nedělí (1) a končí sobotou (7). Úryvek níže je vyňat ze zdrojového kódu získaného z návodu.[25] Lze v něm vidět nastavení pro pátek, 20.7.2018, 20:05. Po nastavení času ve zdrojovém kódu, je potřeba tento kód nahrát do Arduina se zapojeným RTC modulem. To může být uskutečněno pomocí programu Arduino IDE. Dále už je jen zapotřebí nahrát řídicí program pro rehabilitační pomůcku. RTC modul bude pracovat s časem nastaveným při nahrání předchozího programu.

```
void setup(){
    Wire.begin();
    Serial.begin(9600);
    // set the initial time here:
    // DS3231 seconds, minutes, hours, day, date, month, year
    setDS3231time(00,05,20,6,20,7,18);
}
```

5.3.1.3 Přijímač přesného času z Německa DCF77

Sketch zajišťující správnou funkci modulu přesného času nebyl ve finální verzi programu použit, jak již bylo zmíněno v sekci 5.2.2.1.[16] Jelikož přijímač vyžaduje neustále běžící přijímací cyklus, nebylo jeho použití vhodné pro aplikaci v rehabilitačním přístroji.

5.3.1.4 USB modul CH376S

Snaha najít oficiální podklady od výrobce, na anglické verzi jejich webové stránky, byla neúspěšná. [26] V závěru psaní této bakalářské práce se mi povedlo na čínské jazykové verzi webových stránek

výrobce, najít archiv, který obsahoval oficiální datasheet (v čínštině) i knihovnu pro práci s tímto modulem.[27] [28] Oficiální knihovny však nakonec nebyly ke zprovoznění použity. Ve snaze zprovoznit modul USB jsem nalezl tři ruzné zdroje používající sériové komunikace UART. [29] [30] [31] Tutorial [29] obsahuje kompletní návod na zprovoznění, včetně ukázkového videa. Program z tohoto tutorialu je napsán tak, že uživatel musí manuálně přes konzoli sériové komunikace na počítači zadávat čísla, které odpovídají příkazům z kódu.

- **Příkaz „1“** - kontroluje připojení USB modulu.
- **Příkaz „2“** - kontroluje přítomnost USB flash disku.
- **Příkaz „3“** - resetuje všechna nastavení.
- **Příkaz „4“** - provede příkazy 1 a 2, aktivuje komunikaci mezi USB a Arduinem, vytvoří textový soubor TEST4.txt, zapíše do něj, vypíše informace o zápisu a zavře soubor.
- **Příkaz „5“** - je shodný s příkazem 4, akorát místo zápisu do souboru z něj čte.
- **Příkaz „6“** - upravuje obsah souboru TEST4.txt.
- **Příkaz „7“** - smaže soubor TEST4.txt.
- **Příkaz „8“** - přečte soubor s názvem TEST2.txt.
- **Příkaz „9“** - přečte soubor s názvem TEST3.txt.

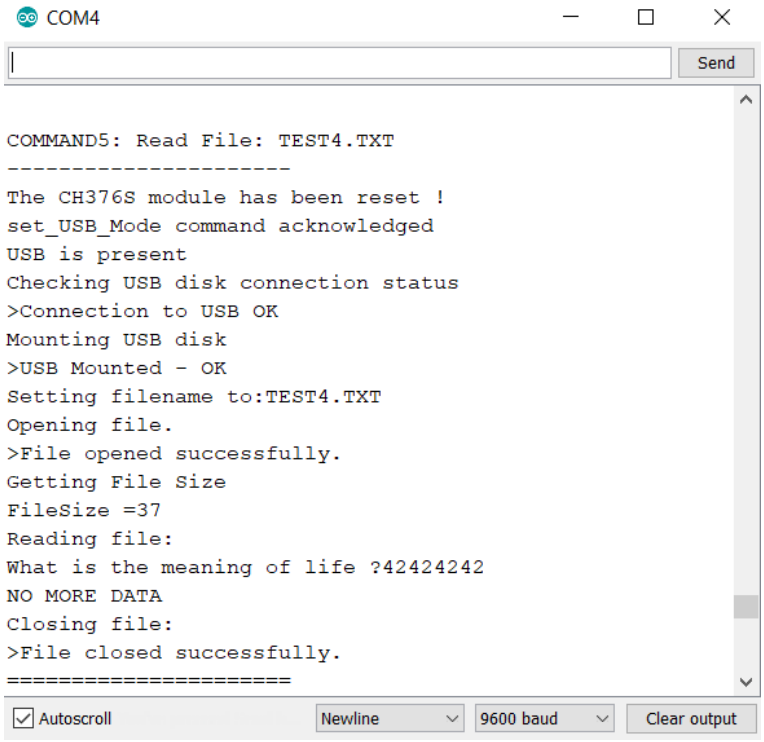
Program [30] vychází z výše zmíněného tutorialu. Zdrojový kód je však jiný. Program provede kompletní inicializaci v jednom cyklu. Z tohoto programu by se dalo vycházet pro implementaci USB úložiště do řídicího programu.

Program [31] je naprogramován pro použití přímo s Arduinem MEGA. Využívá nativní I/O pro sériovou komunikaci (Serial3), které nejsou využívány pro komunikaci s PC konzolí. Tento program nebyl otestován z důvodu dostupnosti pouze desky Arduino NANO.

5.3.2 Knihovny

5.3.2.1 wire.h

Knihovna wire.h je nutná pro použití a správnou funkci sběrnice I^2C. Přes sběrnici I^2C komunikuje Arduino s RTC modulem reálného času.



Obrázek 5.7: Výstup v konzoli počítače vytvořený programem z tutorialu [29]

5.3.2.2 SoftwareSerial.h

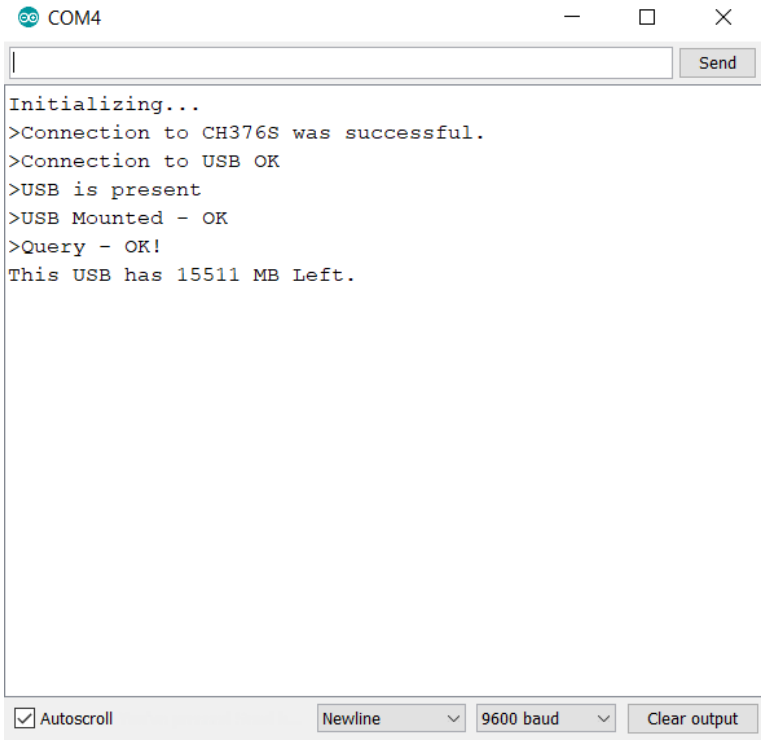
SoftwareSerial.h je knihovna, která umožňuje použití téměř libovolného digitální I/O pro sériovou komunikaci. Používá se, pokud potřebujeme komunikovat po UART sériovém rozhraní s více různými zařízeními, za předpokladu dostatečné paměti v bufferu. V každý daný moment může však po SoftwareSerial komunikovat pouze jedno zařízení.[32] V našem případě je zahrnuta v programu, který ovládá USB modul CH376s.[30]

5.3.2.3 SPI.h a SD.h

Obě tyto knihovny společně zajišťují práci s modulem SD paměťových karet. SPI.h poskytuje potřebné funkce pro sériově-paralelní komunikaci (SPI) a SD.h zprostředkovává správné čtení souborů a zapisování do paměti SD karty.

5.3.2.4 CH375.h

Oficiální knihovna pro USB modul napsaná pro čip CH375, jejíž kód je kompatibilní s čipem CH376s. Podle datasheetu[9] je čipem CH376s pouze novější verzí čipu CH375 a disponuje širší



Obrázek 5.8: Výstup v konzoli počítače vytvořený programem [30]

škálou aplikací. Knihovna pro čip CH375 by měla být kompatibilní s čipem CH376s.

5.4 Problémy v procesu

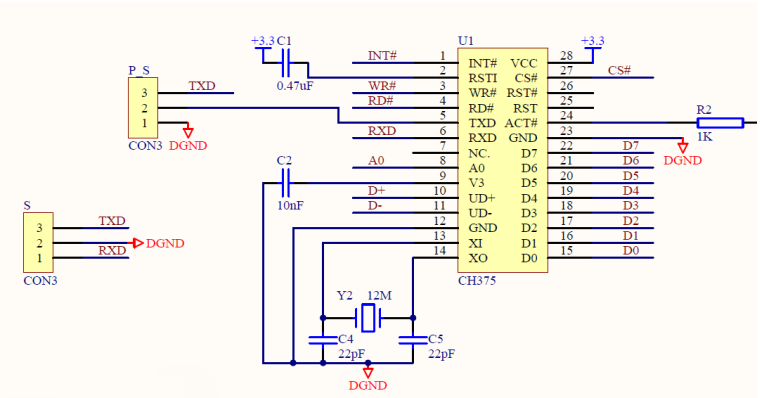
5.4.1 USB modul CH376S

Zakoupený modul byl vyroben jiným výrobcem, než modul použitý v tutorialu[29], což znamenalo odlišné označení a rozložení vstupu/výstupu. Některé výstupy a označení dokonce chyběly. Tato skutečnost značně komplikovala pokusy o zprovoznění USB modulu a způsobila, že jsem od pokusů na dlouho upustil. Anglická verze webové stránky výrobce sice obsahovala odkazy na všechny vyráběné součástky, jejich popis byl však mnohdy velice strohý.[26] Oficiální datasheet od výrobce se mi podařilo dohledat až při pozdějších opětovných pokusech o zprovoznění tohoto modulu, a to na originálních stránkách v čínském jazyce.[27]

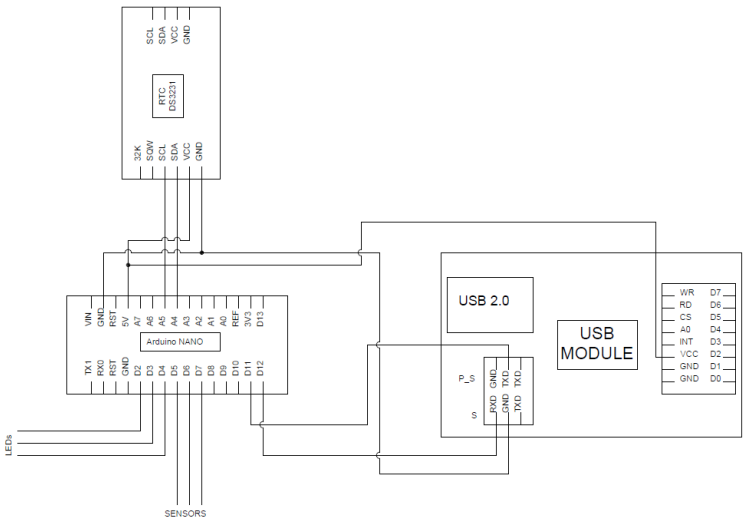
Bližším prozkoumáním integrovaného obvodu modulu jsem nakonec našel příslušné I/O, které jsou zapotřebí ke správné funkci modulu. Po pečlivém zapojení a vložení flash disku, se mi s použitím programu[29] podařilo úspěšně vytvořit, upravit i přečíst jednoduchý textový soubor na flash disku.

Použitý program však občas nedokázal navázat spojení s flash diskem. Nejčastěji se tento problém objevil při novém nahrání programu do Arduina, kdy flash disk zůstal zastrčen do USB portu modulu. Vytažením a opětovným zasunutím se tento problém vyřešil. Prvotní neúspěšné pokusy o zprovoznění byly dle mého názoru způsobeny několika skutečnostmi: nedokonalým zapojením vodičů, zapojením ke špatným I/O na USB modulu a použitím flash disku, který z neznámých důvodů nebyl kompatibilní s USB modulem.

Ikdyž ze schémata poskytnutého výrobcem (obr. 5.9) vyplývá možnost zapojení pomocí tří v řadě za sebou jdoucích I/O označených písmenem "S", při otestování tohoto zapojení modul nefungoval. Jediné nalezené funkční zapojení bylo dle schématu 5.10.



Obrázek 5.9: Výstupy USB modulu dle výrobce[28]



Obrázek 5.10: Schéma funkčního zapojení USB modulu

Úspěšným zprovozněním tohoto modulu se otevírá možnost použití USB flash disku jako datového nosiče. Na úpravu zdrojového kódu řídícího programu, pro práci s USB modulem, mi bohužel nezbyl čas.

5.4.2 Časové indexování souborů a vlastní formát souboru DATALOG.txt

Problém se správným zápisem času a data vznikl domněnkou, že by tento údaj měl být v souboru uveden pouze jednou. V průběhu zkoumání možných formátů výstupního datového souboru, jsem si uvědomil, že je možné časově a datově indexovat samotné reakce, a to díky struktuře jazyka XML, kde bylo nutné tyto informace uvést při každém zápisu. Tento princip jsem poté uplatnil při použití formátu CSV. Podrobný popis zamýšlených formátů naleznete v sekci 5.6.2.

5.4.3 Analýza textového souboru CONFIG.txt

Vyhodnocování konfiguračního souboru a přiřazování dat v něm obsažených ke správným proměnným, považuji za složitou operaci. Problémem pro mě bylo vymyslet vhodný formát dat v souboru tak, aby bylo načtení obsahu tohoto souboru co nejjednodušší. Řešil jsem způsob oddělení číselných hodnot od názvu, oddělení jednoho názvu s příslušnou hodnotou od následujícího, formu jejich přiřazení ke správným proměnným a vlastní pojmenování jednotlivých proměnných. Dále bylo problémem definování aktivních snímačů pro dané cvičení. Jelikož mnou vytvořený čtecí algoritmus jakoukoliv přečtenou číslici přiřazuje k číselné hodnotě proměnné, bylo zapotřebí vymyslet jiný způsob. Nakonec jsem nahradil číselné označení snímačů abecedním, viz 5.6.1. Čtení souboru je provedeno načítáním jednotlivých znaků a následným přidáním přečteného znaku do proměnné formátu `STRING`. Takto jsou načteny i číselné hodnoty, které pak jsou pomocí funkce `toInt()` převedeny do formátu `INTEGER`,která se pak dá jednoduše přiřadit k příslušné proměnné.

```
while (myFile.available()){
    temp = myFile.read();
    if(isAlpha(temp) == true){              //čtení názvu proměnné
        variableName += (char)temp;
    }
    if(temp == '='){
    }
    if(isDigit(temp) == true){              //čtení číselné hodnoty
        numberValue += (char)temp;
    }
```

```
        if(temp == '\n'){                       //konec řádku v souboru
            if(variableName == "waitTime"){     //porovnávání názvu proměnné
                waitTime = numberValue.toInt(); //pro přiřazení
            }
        }
    }
```

5.5 Ladění kódu

5.5.1 Metodika určení aktivovaného senzoru

Jelikož původně k určení aktivovaného senzoru mělo být použito čipu s čtyřmi výstupy o hodnotách 1 nebo 0, jejichž kombinace měla odpovídat číslu snímače v binární soustavě, bylo zapotřebí vyřešit rozpoznávání aktivovaného snímače. Nabízelo se několik možností:

1. Porovnávání pomocí funkce `IF`, zkoumající, zda aktuální kombinace odpovídá jednotlivým kombinacím pro jednotlivé snímače, je varianta myšlenkově nejjednodušší, avšak z programovacího hlediska velice neefektivní a nevzhledná. Pro zjištění aktivovaného snímače by bylo v nejhorším případě zapotřebí 12 porovnání - pro každý snímač jedno.

2. Vnoření funkce `IF` - postupně se porovnávají hodnoty na jednotlivých místech. Po každém porovnání nasleduje větvení dle zjištěné hodnoty. Tato metoda přináší redukci na "pouze"4 porovnání. Stále však není ideální.

3. Matematický převod binární hodnoty pomocí cyklu FOR v kombinaci s funkcí `SWITCH` - cyklus převede kombinaci na číslo v desítkové soustavě.

```
activeSens[5]={1,0,1,0}     \\příklad analyzované kombinace
sensorNum=0;                \\proměnná aktivovaného snímače
power=1;                    \\mocnina
for (i=0; i<4; i++){        \\přepočet
   sensorNum += activeSens[3-i]*power;
   power *= 2;
}
```

Poslední varianta se zdá být nejelegantnějším řešením tohoto problému. Ve finální verzi programu se však nevyskytuje, protože navržený elektronický systém s kombinačním výstupem znemožňuje aktivaci více tlačítek současně. Stisk více tlačítek by byl algoritmem vyhodnocen jako stisk jednoho jiného snímače, kterému by odpovídala výstupní kombinace. Jelikož možnost stiknutí více tlačítek zároveň je nutností, bylo rozhodnuto o použití větší řídicí desky Arduino MEGA, která disponuje dostatečným počtem I/O. Každý snímač je tedy ve finále připojen na vlastní I/O.

5.5.2 Ovládání LED

V průběhu programovacího procesu jsem využil a prozkoumal několik kódových metod ovládání LED diod.

Příkaz na ovládání výstupních pinů je ve formě: `digitalWrite(pin, HODNOTA);`, kde `pin` je číslo ovládaného pinu ve formátu integer a `HODNOTA` může nabývat dvou případů: `LOW` (vypnuto) a `HIGH` (zapnuto). Základní možností je na pozici pin manuálně vložit číslo pinu, což však znemožňuje další manipulaci, resp. záměnu za jiný pin, proto je vhodné si stanovit proměnnou ve stejném formátu, se kterou se dá již snadněji manipulovat. V naší aplikaci ale máme LED diod dvanáct, což by při použití tohoto formátu mělo za následek značné navýšení délky kódu, jelikož potřebujeme diody velice často vypínat a zapínat. Proto by bylo nasnadě použití jiného formátu.

Pro zjednodušení se nabízejí varianty, které využívají formátu `ARRAY` a cyklu `FOR`. Následující úryvky neobsahují všechny nutné části pro správnou funkci v Arduinu, slouží pouze jako ukázka malého zlomku kódu.

1. Dvě proměnné formátu `ARRAY` - jedna obsahující piny k zapnutí (ledON) a druhá obsahující piny k vypnutí (ledOFF). Výsledný cyklus by vypadal následovně:

```
int ledON[3]={1,5};             \\aktivace pinů 1 a 5 (LED1)
int ledOFF[6]={2,3,4,6,7};      \\piny určené k deaktivaci
int i=0;

for(i=0; i<2; i++){              \\zapínání pinů z array ledON
    digitalWrite(ledON[i], HIGH);
}
for(i=0; i<5; i++){              \\vypínání pinů z array ledOFF
    digitalWrite(ledOFF[i], LOW);
}
```

Tato varianta však uvažuje, že pro sepnutí diody je potřeba na oba spínací tranzistory přivést logické 0, což ale není pravda. Pokud ve vypnutém stavu je zapotřebí přivést na P-MOSFET (výstupy 1-4) tranzistory logické 1 a na N-MOSFET (výstupy 5-7) logické 0. Pro spuštění diody 1 musí být na N-MOSFET při výstupu 5 logická 1 a na P-MOSFET při výstupu 1 logická 0:

```
int pinON[5]={2,3,4,5};
int pinOFF[4]={1,6,7};
int i=0;

for(i=0; i<(sizeof(pinON)-1); i++){         \\zapínání pinů z array pinON
    digitalWrite(pinON[i], HIGH);
}
for(i=0; i<(sizeof(pinOFF)-1); i++){        \\vypínání pinů z array pinOFF
    digitalWrite(pinOFF[i], LOW);
}
```

2. Jediná proměnná formátu array, která obsahuje pouze informaci o zapnutí a vypnutí příslušných pinů, jejichž pozice odpovídá číslu pinu. Pro tuto variantu je zapotřebí zapojit všech sedm vodičů ve správném pořadí do pinů Arduina s po sobě následujícími číselnými hodnotami.

```
int led[8]={0,1,1,1,1,0,0};     \\piny 1 a 5 jsou ve spínací kombinaci
int i=0;

for(i=0; i<7; i++){
    if(led[i]==1){                  \\zapne pin pokud se na jeho pozici
      digitalWrite(i+1, HIGH);      \\nachází 1
    }
    else{                           \\vypne pin pokud se na jeho pozici
      digitalWrite(i+1, LOW);       \\nachází hodnota různá od 1
    }
}
```

Zde je bohužel vyžadováno, aby vodiče byly zapojeny na výstupy s po sobě jdoucími číselnými označeními. To by znamenalo možné komplikace při budoucích úpravách zapojení zařízení. Možným řešením je použít `SWITCH...CASE` funkci, která by k pozici v `ARRAY`, neboli hodnotě i, přiřadila správný pin:

```
int led[8]={0,1,1,1,1,0,0};     \\piny 1 a 5 jsou ve spínací kombinaci
int i=0;

for(i=0; i<7; i++){
    switch(i){
        case 0:
            pin = 12;
            break;
        case 1:
            pin = 34;
            break;
        .
        .
        case 6:
            pin = 22;
            break;
        default:
            break;
      }
    if(led[i]==1){                  \\zapne pin pokud se na jeho pozici
      digitalWrite(pin, HIGH);      \\nachází 1
    }
    else{                           \\vypne pin pokud se na jeho pozici
      digitalWrite(pin, LOW);       \\nachází hodnota různá od 1
    }
}
```

Využívat funkce `SWITCH...CASE` pro ovládání LED diod se však zdá spíše komplikující než zjednodušující řešení, které navíc zpomalí provádění jednotlivých částí kódu a zvýší prodlevy.

3. Deklarace nové funkce mimo řídící smyčku:

```
const int redLed1[3]={1,5}              //spínací piny diody

void ledON(int ledNumber[]){            //deklarace funkce
    digitalWrite(ledNumber[0], LOW);
    digitalWrite(ledNumber[1], HIGH);
            }

ledON(redLed1);                         //použití nové funkce
```

Deklarování nové funkce pomůže s přehledností kódu, při opakovaném používání různých kódových sekvencí. Zdá se býti nejlepším řešením pro zjednodušení a zpřehlednění kódu programu. S využitím myšlenek v bodě 1, jsem vytvořil i funkci pro zhasnutí všech diod.

```
const int pLines[5]={1,2,3,4}           //výstupy pro tranzistory P-MOSFET
const int nLines[4]={5,6,7}             //výstupy pro tranzistory N-MOSFET

void allLedsOff(){
  for(i=0;i<(sizeof(pLines)-1);i++){
    digitalWrite(pLines[i],LOW);
  }
  for(i=0;i<(sizeof(nLines)-1);i++){
    digitalWrite(nLines[i],HIGH);
  }
}
```

Pro snížení doby reakce zařízení z důvodu právě probíhajícího kódu jsem vytvořil funkce pro blikání diody, která se přeruší stiskem tlačítka:

```
void interruptLedBlink(int led[], int blinkDelay, int button){
    ledOn(led);                         //zapnutí diody
    for(i=0; i<11; i++){                //cyklus pro celkovou dobu prodlevy
      if(digitalRead(button)==HIGH){    //zrušení cyklu při stisku tlačítka
        break;
      }
```

```
      delay(blinkDelay/10);              //prodleva rozdělená na menší části
    }
    ledOff(led);                         //vypnutí diody
    for(i=0; i<11; i++){                 //cyklus identický s předchozím
      if(digitalRead(button)==HIGH){
        break;
      }
      delay(blinkDelay/10);
    }
}
```

Ve výsledném programu byla spousta opakovaně používaných příkazů nahrazena funkcemi. Díky tomu se počet řádků řídícího algoritmu snížil, ikdyž všechny funkce musely být deklarovány mimo hlavní řídící smyčku, a zároveň se zlepšila čitelnost celého programu.

5.6 Kompatibilita a konfigurovatelnost pomocí GUI

5.6.1 Konfigurační soubor CONFIG.txt

Jako formát konfiguračního souboru jsem zvolil tu nejjednodušší možnou variantu a to je jednoduchý textový soubor. Vzhledem ke struktuře mnou vytvořeného čtecího algoritmu bohužel není možné do názvu uvedených hodnot vpisovat čísla. Konfigurační soubor je napsán ve formátu jedné nastavované proměnné na jeden řádek, např.: *JmenoPromenne=99999*. Z tohoto důvodu bylo nutné vymyslet způsob, jakým zapsat jednotlivá tlačítka, která se mohou v průběhu algoritmu aktivovat. Tento problém byl vyřešen použitím velkých písmen na konci proměnných pro tlačítka namísto čísel, tzn. *butA = tlačítko 1, butB = tlačítko 2, ... , butL = tlačítko 12*. Na pořadí zapsání jednotlivých proměnných do souboru nezáleží. Formát tohoto souboru považuju za jednu ze slabin výsledného programu. Dovedu si představit, že existuje vhodnější formát pro takovýto konfigurační soubor. Obsah použitého konfiguračního souboru vypadá třeba následovně:

```
pacientNO=001109
waitTime=4500
maxExerciseTime=21000
idleTime=1500
butA=1
```

```
butD=1
butK=1
```

V této ukázce `pacienNO` obsahuje číslo přiřazené pacientovi, `waitTime` je časové okno pro stisk aktivovaného tlačítka, `maxExerciseTime` je požadovaná maximální délka cvičení a `idleTime` je čas mezi stisknutím tlačítka a aktivací dalšího tlačítka. Proměnné `butX` říkají, která tlačítka mají být ve cvičení aktivní. V tomto případě to jsou tlačítka 1, 4 a 11 (A=1, D=4, K=11).

5.6.2 Výstupní data (soubor DATALOG)

5.6.2.1 Formát .txt

Formát .txt je velmi jednoduchým a základním formátem, byl tedy mým prvním kandidátem na vhodný formát výstupního souboru. Struktura zapsaných dat vycházela ze stejné myšlenky, jakou se řídí konfigurační soubor, a dal by se tedy použít podobný čtecí algoritmus pro následný import a vyhodnocení dat v počítačovém programu. Všechna data se zapisují do jednoho souboru na jednotlivé řádky. Nová cvičení jsou oddělena dvěma volnými řádky a dvojicí "##", následuje číslo pacienta, datum a čas, pole aktivních tlačítek, reakční čas přiřazený k jednotlivým tlačítkům a celková délka cvičení, viz následující ukázka:

```
##
Exercise details:
Pacient:1109
DATE:26/7/18-14:42
buttons=1,1,1,0,0,0,0,0,0,0,0,0
butB=740
butC=511
butA=570
butC=487
butB=489
butA=413
butA=392
butC=443
exerciseTime=21748
```

Tento formát však není příliš kompatibilní pro použití a zpracování v jiných programech.

5.6.2.2 Formát .xml[33] [34]

XML, neboli eXtensible Markup Language (česky rozšiřitelný značkovací jazyk), je jazyk vytvořený se záměrem kompatibility s jakýmkoliv softwarem, bez ohledu na rok jeho vytvoření, a je tedy univerzálním datovým formátem zpracovatelným spoustou programů jako například tabulkový editor Microsoft Excel. Strukturově je velmi podobný jazyku HTML, který se používá ke tvorbě webových stránek. Z důvodů výše zmíněné kompatibility by bylo velice příhodné vytvářet výstupní datové soubory v tomto formátu.

Problém však u této aplikace nastává v tvorbě úvodní a uzavírací značky. Tyto značky, známé pod označením "root", mohou být v souboru uvedeny pouze jednou. Nastává tedy komplikace při zapisování do jednoho souboru, s definováním okamžiku vytvoření této uzavírající značky, jelikož není přesně stanoveno, které cvičení bude poslední před odevzdáním dat pacientem terapeutovi ke zpracování. Tento problém se však dá obejít tvorbou individuálních souborů pro jednotlivá cvičení. Vzniká tím však komplikace ve způsobu originálního pojmenovávání souborů, tak aby se nepřepisovaly, a samotnou prací s nimi.

```
<?xml version="1.0" encoding="UTF-8"?>          //definování souboru
<datalog>                                       //počáteční značka "root"

<exercise Pacient="1109" Datum_cviceni="26/7/18" Hodina_cviceni="14:42"
Mod="0" Doba_cviceni="21000" Reakcni_okno="1500" Delka_cviceni="180000">
<Snimac>2</Snimac><Reakcni_doba>396</Reakcni_doba>
</exercise>
<exercise Pacient="1109" Datum_cviceni="27/7/18" Hodina_cviceni="14:42"
Mod="0" Doba_cviceni="21000" Reakcni_okno="1500" Delka_cviceni="180000">
<Snimac>2</Snimac><Reakcni_doba>368</Reakcni_doba>
</exercise>

</datalog>                                      //uzavírací značka "root"
```

5.6.2.3 Formát .txt s obsahem ve formátu CSV

Kompromisem, s ohledem na kompatibilitu a přenositelnost dat, by mohl být formát .txt, s daty oddělenými čárkou (CSV). Tento formát využívá jednoduchého formátovaní dat do sloupců pomocí jejich oddělení čárkami anebo tabulátory (TSV). Umožnil by zápis do jediného souboru a zároveň

i jednoduché importování dat do tabulkového softwaru jako je třeba již zmíněný a nejpoužívanější Microsoft Excel. Možný formát obsahu souboru:

```
Pacient,Datum cvičení,Hodina cvičení,Mod,Doba cvičení,Reakční okno,
Délka cvičení,Snímač,Reakční doba
1109,26/7/18,14:42,0,21000,1500,180000,2,143
1109,26/7/18,14:42,0,21000,1500,180000,3,656
1109,26/7/18,14:42,0,21000,1500,180000,7,334
1109,27/7/18,10:31,0,21000,1500,180000,1,342
1109,27/7/18,10:31,0,21000,1500,180000,6,234
```

Ikdyž je tento formát méně přehledný pro čtení člověkem, zdá se momentálně jako nejvhodnější formát výstupu pro pozdější analytické zpracování dat. Při importu do tabulkového procesoru se orientace v datech znatelně zlepší. Cvičební módy zmíněné v sekci 4.2 vyžadují různá výstupní data a proto je v programu zahrnuta tvorba dvou různých výstupních souborů, DATALOG.txt pro mód 1 a DATALOG2.txt pro mód 2. Zjednodušují se tím budoucí zpracování nasbíraných dat.

Kapitola 6

Testování

Všechen software byl testován pouze na vývojovém vzorku z podkapitoly 5.2. Jelikož elektronika svinutelné verze rehabilitačního zařízení není kompletně sestavená a nainstalovaná do zařízení. Testovací možnosti tedy byly omezené.

K dispozici jsem měl pouze Arduino NANO, které ve pozdních fázích tvorby softwaru vykazovalo nedostatečnou velikost operační paměti. Výsledný řídící program tedy nebyl otestován na desce Arduino MEGA, která má být řídící jednotkou rehabilitačního přístroje. Omezená paměť desky Arduino NANO mě omezovala v počtu možných sériových textových výstupů do konzole, které se zdály být největší zátěží na velikost volné paměti. Mnoho sériových výstupů a chybových hlášení tedy bylo třeba zahrnout pouze ve formě komentářů. Nedostatečná paměť se projevovala zejména chybovým hlášení při otevírání konfiguračního souboru z SD karty. Z důvodu omezené paměti, nebylo ani možné vyzkoušet oba vytvořené algoritmy v jednom programu. Musely tedy být testovány zvlášť.

Kvůli nekompletnosti elektroniky rehabilitačního přístroje, nebyla ověřena funkčnost ovládání LED diod při maticovém zapojení. Zároveň se mi ani nepodařilo sehnat tranzistory stejného typu, který byl uveden v projektové zprávě, abych sám sestavil testovací obvod. Za předpokladu funkčnosti navrhnuté elektroniky dle projektu [3], by však ovládání mělo fungovat.

Kapitola 7

Závěr

Podařilo se mi vytvořit dva ze tří módů základního řídicího softwaru pro rehabilitační přístroj na platformě Arduino. Na poslední chvíli se povedlo zprovoznit i rozšiřující USB modul, který umožňuje použití flash disku jako úložiště pro konfigurační a výstupní soubory. Použití USB flash disků je vzhledem k onemocněním pacientů vhodnější a to hlavně z důvodu jednodušší manipulace. Bohužel mi nezbyl čas na implementaci potřebného kódu do řídícího programu pro rehabilitační pomůcku. Výsledný program tedy funguje s použitím modulu pro SD paměťové karty. Paměťové karty SD svými velice malými rozměry, a tudíž i složitější manipulací, výrazně ztěžují pacientům užívání přístroje. Z tohoto důvodu by bylo vhodné upravit zdrojový kód řídicího programu pro použití s USB modulem.

Rehabilitační přístroj disponuje značným potenciálem pro další rozvoj, ať už ze softwarové nebo hardwarové stránky.

Z hardwarového hlediska se nabízí další úprava elektroniky za účelem minimalizace celkového objemu kabeláže. Té by se dalo docílit osazením jednotlivých tlačítek vlastními procesory a připojením všech tlačítek pomocí jediné sběrnice I^2C, popřípadě radikálnějším řešením které by využívalo bezdrátové komunikace. Dále se nabízí osazení přístroje LCD displejem, který by uživateli poskytoval instrukce pro složitější úkony.

Ze softwarového hlediska je třeba vytvořit funkční a příjemné grafické rozhraní (GUI) pro lékaře a terapeuty, které bude kompatibilní s vytvořeným řídicím softwarem, a bude umožňovat snadnou manipulaci s daty, přístup k nim, a poskytne přehledně zobrazené analýzy importovaných dat. Dále se nabízí úprava datového výstupu řídícího softwaru na potenciálně vhodnější a univerzálnější datový formát, a celková optimalizace řídicího softwaru pro efektivnější manipulaci s pamětí a výpočetním výkonem.

Prostor pro rozvoj se nachází i v úpravě stávajícího zařízení na dvě rozdílná zařízení: jedno, v aktuálním konceptu, určené pro pacienty k domácímu použití a rehabilitaci, a druhé určené pro ordinace lékařů. Zařízení pro lékaře by umožňovalo přímé přípojení přes počítač a nastavení parametrů cvičení v reálném čase, bez nutnosti tvorby konfiguračních souborů, jejich kopírování na datový nosič a následné umístění do přístroje, což by značně ulehčilo hledání správného nastavení cvičebních módů pro nové pacienty.

Možným vylepšení je také použití přijímače přesného času DCF-77. Tato úprava by vyžadovala osazení modulu vlastním procesorem, který by vykonával potřebný cyklus pro přijímání a zpracování datového signálu. Připojení k řídicí desce by se realizovalo po sběrnici I^2C, která by se pouze dotazovala na potřebné informace z modulu v požadovanou chvíli.

Literatura

[1] VONDRA, Jakub. *Rehabilitační přístroj*. Praha, 2016. 67 str. Diplomová práce. České vysoké učení technické v Praze. Vedoucí práce Ing. Bc. Šárka Němcová, Ph.D.

[2] SOUKAL, Jan. *Cvičební zdravotnická pomůcka*. Praha, 2018. 40 str. Bakalářská práce. České vysoké učení technické v Praze. Vedoucí práce Ing. Bc. Šárka Němcová, Ph.D.

[3] CAHYNA, Martin a Jan RIEDL. *Projekt II - Rehabilitační pomůcka* Praha, 2018. 13 str. Projektová zpráva. České vysoké učení technické v Praze.

[4] AKUJI, Shahir a Felix HROMATKA. *Modernization of a Medical Device*. Praha, 2017. 14 str. Projektová zpráva. České vysoké učení technické v Praze.

[5] HAVRDOVÁ, Eva. *Rozstroušená skleróza v praxi.* Praha: Galén, 2015. ISBN 978-80-7492-189-6.

[6] BONNET, Anne-Marie a Thierry HERGUETA. *Parkinsonova choroba: rady pro nemocné a jejich blízké*. Praha: Portál, 2012. Rádci pro zdraví. ISBN 978-80-262-0155-7

[7] Arduino - Software. *Arduino* [online]. [cit. 2018-08-11]. Dostupné z: https://www.arduino.cc/en/Main/Software

[8] Arduino - FAQ. *Arduino* [online]. [cit. 2018-08-11]. Volně přeloženo z: https://www.arduino.cc/en/Main/FAQ#toc13

[9] *Technický list: File manage and control chip CH376* [online]. [cit. 2018-08-11]. Dostupné z: https://www.mpja.com/download/ch376ds1.pdf

[10] Ch376-ch376s-u-disk-read-write-module.jpg (800×800). *Robocraze - The World of Embedded Systems | Automation | IoT* [online]. [cit. 2018-08-11]. Dostupné z: https://www.robocraze.com/media/catalog/product/optimized/4/1/41a9ee85281392e3557830b466c9dfc4/ch376-ch376s-u-disk-read-write-module.jpg

[11] Arduino - SPI. *Arduino - Home* [online]. [cit. 2018-08-11]. Dostupné z: https://www.arduino.cc/en/Reference/SPI

[12] Sd-card-modul-spi.jpg (800×800). Mikropočítače a příslušenství - od bastlířů bastlířům - laskarduino [online]. [cit. 2018-08-11]. Dostupné z: https://laskarduino.cz/4343-thickbox_default/sd-card-modul-spi.jpg

[13] DCF 77 - PTB.de. *Physikalisch-Technische Bundesanstalt* [online]. [cit. 2018-08-11]. Dostupné z: https://www.ptb.de/cms/en/ptb/fachabteilungen/abt4/fb-44/ag-442/dissemination-of-legal-time/dcf77.html.

[14] DCF77 time code - PTB.de. Physikalisch-Technische Bundesanstalt [online]. [cit. 2018-08-11]. Dostupné z: https://www.ptb.de/cms/en/ptb/fachabteilungen/abt4/fb-44/ag-442/dissemination-of-legal-time/dcf77/dcf77-time-code.html

[15] *Technický list: RTC Hodiny reálného času* [online]. [cit. 2018-08-11]. Dostupné z: https://arduino-shop.cz/docs/produkty/0/74/1459971336.pdf.

[16] Přesný čas z Německa DCF1 | Arduino návody. *Webový magazín o ARDUINU | Arduino návody* [online]. [cit. 2018-08-11]. Dostupné z: http://navody.arduino-shop.cz/navody-k-produktum/presny-cas-z-nemecka-dcf1.html

[17] 3963-thickbox.jpg (436×356). *Webový magazín o ARDUINU | Arduino návody* [online]. [cit. 2018-08-11]. Dostupné z: http://navody.arduino-shop.cz/images/obr_clanky/58_dcf1/3963-thickbox.jpg

[18] Real-Time Clock Calculator - Maxim. *Maxim Integrated - Analog, linear, mixed-signal devices* [online]. [cit. 2018-08-11]. Dostupné z: https://www.maximintegrated.com/en/design/tools/calculators/product-design/rtc.cfm

[19] 109_rtc_ds3231.jpg (800×800). *Webový magazín o ARDUINU | Arduino návody* [online]. [cit. 2018-08-11]. Dostupné z: http://navody.arduino-shop.cz/images/obr_clanky/109_rtc_ds3231/109_rtc_ds3231.jpg

[20] Arduino Nano. *Arduino* [online]. [cit. 2018-08-11]. Dostupné z: https://store.arduino.cc/arduino-nano

[21] Arduino Mega 2560 Rev3. *Arduino* [online]. [cit. 2018-08-11]. Dostupné z: https://store.arduino.cc/arduino-mega-2560-rev3

[22] Arduino - LibraryExamples. *Arduino* [online]. [cit. 2018-08-11]. Dostupné z: https://www.arduino.cc/en/Tutorial/LibraryExamples

[23] Arduino - ReadWrite. *Arduino* [online]. [cit. 2018-08-11]. Dostupné z: https://www.arduino.cc/en/Tutorial/ReadWrite

[24] Arduino - RTC. *Arduino* [online]. [cit. 2018-08-11]. Dostupné z: https://www.arduino.cc/en/Reference/RTC

[25] Tutoriál – užívání hodin reálného času DS1307 a DS3231 s Arduinem | Arduino.cz. *Arduino.cz - Webový magazín o Arduinu a elektronice* [online]. [cit. 2018-08-11]. Dostupné z: https://arduino.cz/tutorial-uzivani-hodin-realneho-casu-ds1307-a-ds3231-s-arduinem/

[26] CH376 USB Module U Disk Module–Shenzhen LC Technology Co.,Ltd. *Home–Shenzhen LC Technology Co.,Ltd.* [online]. [cit. 2018-08-11]. Dostupné z: http://www.chinalctech.com/index.php?_m=mod_product&_a=view&p_id=649

[27] CH376 USB Module. *Shenzhen LC Technology Co.,Ltd.* [online]. [cit. 2018-08-11]. Dostupné z: http://www.lctech-inc.com/plus/view.php?aid=66

[28] CH375_CH376_U_module.zip | Baidu. *Baidu.com* [online]. [cit. 2018-08-11]. Dostupné z: https://pan.baidu.com/s/1dFOCxPZ

[29] Arduino Basics: CH376S USB Read/Write module. *Arduino Basics* [online]. [cit. 2018-08-11]. Dostupné z: https://arduinobasics.blogspot.com/2015/05/ch376s-usb-readwrite-module.html

[30] CH376S USB Chip with UART Communication. · GitHub. *Discover gists · GitHub* [online]. [cit. 2018-08-11]. Dostupné z: https://gist.github.com/K-ways/52f3e212883f1f5a8076d1efe13a18c4

[31] GitHub - foxik0070/arduino_CH376s. *Discover gists · GitHub* [online]. [cit. 2018-08-11]. Dostupné z: https://github.com/foxik0070/arduino_CH376s

[32] Arduino - SoftwareSerial. *Arduino* [online]. [cit. 2018-08-11]. Dostupné z: https://www.arduino.cc/en/Reference/SoftwareSerial

[33] *Extensible Markup Language (XML)* [online]. [cit. 2018-08-11]. Dostupné z: https://www.w3.org/XML/

[34] XML Syntax. *W3Schools Online Web Tutorials* [online]. [cit. 2018-08-11]. Dostupné z: https://www.w3schools.com/xml/xml_syntax.asp

[35] Arduino - Wire. *Arduino* [online]. [cit. 2018-08-11]. Dostupné z: https://www.arduino.cc/en/Reference/Wire