České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Diplomová práce

Software stimulačního systému pro funkční MR zobrazování

Bc. Pavel Dvořák

Vedoucí práce: doc. Ing. Jaroslav Tintěra CSc.

Studijní program: Otevřená informatika, magisterský
Obor: Softwarové inženýrství
12. května 2014

Poděkování

Děkuji vedoucímu diplomové práce panu doc. Ing. Jaroslavu Tintěrovi CSc., Ing. Janu Rydlovi a jejich kolegům ze Základny radiodiagnostiky a intervenční radiologie Institutu klinické a experimentální medicíny v Praze za poskytnutí motivace, konzultací, cenných rad a za zájem, s jakým sledovali a usměrňovali postup mé práce.

Prohlášení

Prohlašuji, že jsem svou diplomovou 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 12. 5. 2014 .....

Abstract

The goal of this master degree project is to introduce the issue of Functional Magnetic Resonance (fMRI), an evaluation of the currently used stimulating systems, a suggestion and an implementation of the new stimulating system for fMRI. All available technologies were compared and Java was chosen for the new software. Java uses Arduino board for communication with external hardware (joystick and buttons). The final solution was verified by performing clinical and research measurements.

Abstrakt

Cílem této diplomové práce je seznámení se s problematikou vyšetření na funkční magnetické rezonanci, zhodnocení v současnosti používaných stimulačních systémů, návrh a implementace nového stimulačního systému pro funkční MR zobrazování. Na základě zhodnocení dostupných technologií byl vytvořen software v jazyce Java, který pro komunikaci s externím hardwarem (tlačítka a joystick) využívá desky Arduino. Výsledné řešení bylo ověřeno provedením klinického i výzkumného měření.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V roce 1988 byl v Československu uveden do provozu první přístroj, díky kterému bylo možné vyšetřovat pacienty magnetickou rezonancí (MR). Ve druhé polovině 90. let začala být MR využívána i pro funkční zobrazení mozku (fMRI), při kterém je zjišťována aktivita mozku v závislosti na stimulačních podnětech. Aby bylo možné zodpovědně a správně vyhodnocovat naměřená data, je nutné provádět stimulace synchronně s probíhajícím měřením obrazů. I proto tato vědní disciplína vždy vyžadoval spolupráci několika oborů včetně informatiky a statistiky.

Tato práce si klade za cíl zhodnotit řešen stimulace fMRI, která se používají pro funkční MR zobrazování a navrhnout pro potřeby pracoviště v IKEM softwarovou část stimulačního systému, která rozšíří současné možnosti návrhu, provádění a vyhodnocování stimulací. V současnosti používaný systém již nevyhovuje požadavkům, a to ani po softwarové, ani po hardwarové stránce. Nová aplikace má za úkol rozšířit tyto možnosti pro ovládání stimulací, zpřesnit prováděná měření a zároveň umožnit další rozvoj celého systému.

Pro návrh takové aplikace je třeba se seznámit s principy magnetické rezonance a funkčního MR zobrazování. Vývoj je třeba přizpůsobit hardwarovým zařízením, která musí splňovat specifické požadavky vyplývající z přítomnosti silného magnetického pole a snímání relativně slabého vysokofrekvenčního signálu. Pro zajištění kvality výsledných obrazů musí být použita technologie, která nebude způsobovat rušení nežádoucími signály. Úspěšný návrh takové aplikace může zásadně rozšířit možnosti provádění fMRI na pracovišti v IKEM i jinde v České republice.

2 Magnetická rezonance

Nukleární magnetická rezonance (NMR) je jev umožňující vyšetřování magnetických vlastností atomových jader prvků. Tohoto jevu se využívá v mnoha vědeckých disciplínách včetně medicínských oborů. Metoda nejprve využívaná ke studiu magnetických momentů a energetických přechodů jader se osvědčila i v analytické chemii k určování struktury a chemického složení látek (MR spektroskopie). Rozvoj výpočetní techniky v posledních desetiletích umožnil využití jevu magnetické rezonance v medicínských oborech jako tomografické zobrazovací metody (Magnetic Resonance Imaging – MRI).[1] Výhodou nukleární magnetické rezonance je to, že vyšetřovaný pacient není vystaven ionizujícímu záření, a proto oproti výpočetní tomografii (CT) může podstupovat dlouhá nebo opakovaná vyšetření bez rizika vlivu radioaktivity. Naopak je nutné se při rekonstrukci dat naměřených pomocí MR vypořádat s mnohem nižším poměrem signál/šum.

2.1 Funkční MR zobrazování

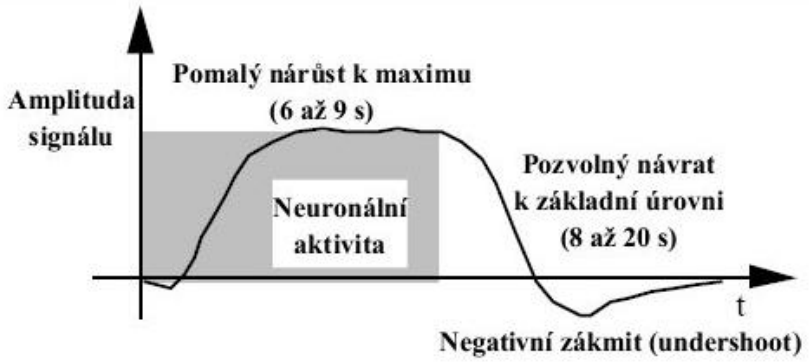
Funkční zobrazování pomocí magnetické rezonance (fMRI) se zjednodušeně zabývá sledováním mozkové aktivity. Pro tento účel se užívají dva fyziologické postupy. Prvním z nich je perfuzní fMRI, kdy se provádí mapování mozkových funkcí na základě změn průtoku krve v místech neuronální aktivity.

Druhý způsob využívá tzv. Blood oxygeantion level dependent (BOLD) efekt. Při zvýšené energetické poptávce aktivních neuronů je zvýšena metabolizace glukózy, která vyžaduje kyslík. Ten je dopravován krví ve vazbě s hemoglobinem[2]. Zde se tedy sleduje poměr průtoku okysličené a neokysličené krve v mozku. To je možné díky jejich rozdílným magnetickým vlastnostem. Odkysličený hemoglobin (deoxy-hemoglobin, dHb) má krevní sloučeniny železa ve stavu se čtyřmi nespárovanými (volnými) elektrony, které tak tvoří značný magnetický moment. Důsledkem je paramagnetické chování dHb na rozdíl odokysličeného hemoglobinu (Hb), který nemá žádný magnetický moment a je diamagnetický. Rozdíl v magnetických vlastnostech látek je vyjádřen rozdílnou susceptibilitou. Relativní rozdíl v susceptibilitě paramagnetického dHb a okolní tkáně tvoří lokální nehomogenity magnetického pole. Rozdíly v susceptibilitě zobrazovaných struktur a jimi vyvolané lokální nehomogenity magnetického pole ovlivňují chování MR signálu.[3]

Neuronální aktivita tedy není měřena přímo. Při neuronální aktivitě pozorujeme zpočátku deoxygenaci hemoglobinu (cca 1s po začátku stimulu) a poté se zpožděním asi 2 – 5 s zvýšení krevního průtoku. Jelikož intenzita průtoku vzroste výrazněji, než spotřeba

[1] RYDLO, Jan. *Periferie stimulačního systému pro funkční MR zobrazování*. Praha: ČVUT 2011. Diplomová práce, ČVUT, Fakulta elektrotechnická, Katedra teorie obvodů.
[2] Viz 1
[3] TINTĚRA, Jaroslav; VYMAZAL, Josef: *Funkční a metabolické MR zobrazení mozku*. Česká radiologie: Časopis radiologické společnosti. Únor 2005

kyslíku, pozorujeme změny v poměru oxy a deoxyhemoglobinu. Naopak po skončení stimulace se poměr postupně vrací do normálu (viz obr. 1).[4]



Obr. 1: Závislost BOLD efektu na čase.[5]

2.1.1 Využití fMRI

Využití fMRI je orientováno do dvou hlavních směrů. Jedním je klinické využití zejména v přípravě pacientů na neurochirurgický výkon, druhým směrem jsou neurofyziologické či psychiatrické studie zdravých dobrovolníků nebo skupin pacientů (např. schizofreniků). Dalším, méně častým, uplatněním fMRI je testování účinků nových farmakologických přípravků nebo vlivu drog na reakce mozku.[6]

2.1.1.1 Klinické využití

Při zasažení mozkových oblastí patologickým procesem (tumor, arteriovenózní malformace ap.) je pro chirurgické řešení důležité znát vzájemnou polohu patologického ložiska a blízkých funkčních center, zodpovědných za základní mozkové funkce, například motorický, optický, akustický, paměťový či verbální processing.[7]

2.1.1.2 Psychiatrické studie

Při psychiatrických studiích je zkoumána především odezva vyšetřovaného pacienta a jeho reakce na stimulace. Zpracovávají se jak obrazy mozku, tak i smyslové odezvy. Ty může vyšetřovaný subjekt realizovat například stiskem tlačítka nebo pohybem joysticku. Může nás např. zajímat správnost, rychlost a intenzita reakce. Podněty mohou být různého typu, nejčastěji vizuální nebo akustické, lze ale pracovat například i s čichovými vjemy.

2.1.2 Design fMRI experimentů

Hodnoty BOLD efektu jsou individuální a neexistuje standardizovaná klidová úroveň. Klidová úroveň se může lišit i u různých center u jednoho pacienta. Proto musí být získány

[4] Dle http://fmri.mchmi.com/
[5] Viz 4
[6] Viz 3
[7] Viz 3

obrazy jak v době, kdy byl pacient v klidu, tak v době stimulací. Při vyhodnocování dat je nutné vědět, jaký byl stav mozku v době jejich snímání. Dle uspořádání stimulačních podnětů rozlišujeme blokový a event related design. Při blokovém designu se pravidelně střídají stejně dlouhé epochy stimulace a klidu. Event related design se skládá z nepravidelně rozmístěných krátkých podnětů. Změna BOLD efektu není v tomto případě tak výrazná, jako u blokového designu, proto je třeba získat větší množství odpovědí. Mezi jednotlivými stimuly je třeba dodržet určitý časový odstup, protože ustálení BOLD efektu může trvat až 20 sekund (viz obr. 1).[8]

2.2 Princip vzniku MR signálu

2.2.1 Jaderný spin a magnetický moment

Z hlediska NMR je důležitou vlastností jaderný spin (I). Ten souvisí s vlastnostmi atomových jader a může nabývat hodnot celého nebo polocelého násobku Planckovy konstanty. Pro zobrazování MR jsou vhodná atomová jádra s hodnotou spinu ½.

Veličina magnetický moment jádra μ vyjadřuje magnetické vlastnosti a popisuje, jak jádro reaguje na přítomnost vnějšího magnetického pole. Je dána vztahem

$$\mu = \gamma \cdot I, \quad (1)$$

kde γ je gyromagnetická konstanta, která je charakteristická pro každé atomové jádro.

Výsledný magnetický moment jádra atomu je vytvořen nespárovaným protonem nebo neutronem. Magnetickou rezonanci můžeme pozorovat jen u jader s lichým nukleonovým číslem, např. ^{1}H, ^{13}C, ^{19}F, ^{23}Na nebo ^{31}P. Hodnoty gyromagnetické konstanty jsou uvedeny v tabulce 1.

Izotop	spin jádra I_q	γ [MHz/T]
^{1}H	½	42,58
^{13}C	½	10,71
^{19}F	½	40,08
^{23}Na	3/2	11,27
^{31}P	½	17,25

Tab. 1: Hodnota spinu jádra a gyromagnetické konstanty.[9]

Jádro vodíku ^{1}H má nejvyšší hodnotu gyromagnetické konstanty, protože jej tvoří pouze jediný proton. Toto jádro je nejcitlivější vůči MR. Vodík je navíc zároveň obsažen v molekulách vody, která tvoří dvě třetiny hmotnosti lidského těla. Zároveň se izotop ^{1}H

[8] Viz 4
[9] http://cw.felk.cvut.cz/wiki/_media/courses/a6m33zsl/mri1.pdf

vyskytuje téměř 100%. Jiný příklad je izotop ^{13}C se přirozeně jen vyskytuje jen v 1 % izotopů, neboť zbytek tvoří izotop ^{12}C.

2.2.2 Vektor magnetizace ve vnějším magnetickém poli

Vektor magnetizace M, který popisuje magnetické vlastnosti mnohaspinového systému, je tvořen vektorovým součtem jednotlivých magnetických momentů jader atomů. Pokud nejsou atomy umístěné v magnetickém poli, nejsou jejich spiny nijak uspořádané a kvůli tepelnému pohybu jsou orientované náhodným směrem. Proto výsledná magnetizace nabývá nulové hodnoty. Poté, co atomy umístíme do silného magnetického pole B_0, přestane být jejich spin orientován libovolně. Za přítomnosti vnějšího magnetického pole může magnetický moment dosahovat pouze konečného počtu orientací, které se navzájem liší energií. Počet orientací je ovlivněn spinem jádra I_q a platí pro něj vzorec

$$k = 2 \cdot I_q + 1. \quad (2)$$

Vodíkové jádro má spin ½, a proto při působení vnějšího magnetického pole může vektor μzaujmout pouze dvě orientace – paralelní a opačnou antiparalelní. Paralelní orientace spinu směřuje souhlasně se směrem vnějšího magnetického pole, antiparalelní proti ní. K tomu, aby byl spin v antiparalelní orientaci, potřebuje větší množství energie, a proto je takových spinů méně. Poměr počtu paralelně (N^-) a antiparalelně (N^+) orientovaných spinů je dán tzv. Boltzmanovým rozdělením:

$$\frac{N^-}{N^+} = e^{-\frac{E}{kT}}, \quad (3)$$

kde k je Boltzmanova konstanta a T je teplota (v Kelvinech) a E je rozdíl energie stavů, který je přímo úměrný intenzitě magnetického pole.

Každý magnetický dipól začne v magnetickém poli vykonávat precesní pohyb okolo osy z (rovnoběžné se směrem vnějšího magnetického pole, viz obr 2). Frekvence tohoto pohybu se nazývá Larmorova frekvence a platí pro ni vztah

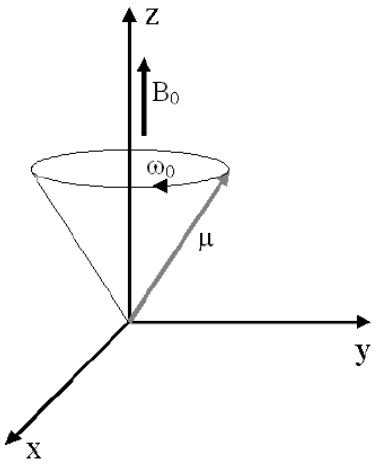
$$\omega_0 = \gamma \cdot B_0, \quad (4)$$

kde B_0 je intenzita vnějšího magnetického pole (T) a γ gyromagnetická konstanta (MHz/T).

Směr magnetického momentu je u všech jader náhodný. Proto v rovině kolmé ke směru magnetického pole B_0 dojde ke vzájemnému vyrušení jejich vlivu na vektor magnetizace tkáně. Výsledný vektor magnetizace M_0 potom má totožný směr jako vnější magnetické pole B_0. [10, 11]

[10] HUETTEL, Scott A.; SONG, Allen W.; MCCARTHY, Gregory. *Functional Magnetic Resonance Imaging.* Second Edition. Sunderland (MA.): Sinauer Associates, Inc., 2009. 542 s.

[11] REIMER, Peter; PARIZEL, Paul M.; STICHNOTH, Falko-Alexander; MEANEY, James F.M. *Clinical MR Imaging : A Practical Approach.* 2nd edition. Berlin: Springer-Verlag, 2003. 597 s. ISBN 3-540-64098-3.



Obr. 2: Precese protonu v magnetickém poli B_0.[12]

2.2.3 Radiofrekvenční pulz, detekce signálu

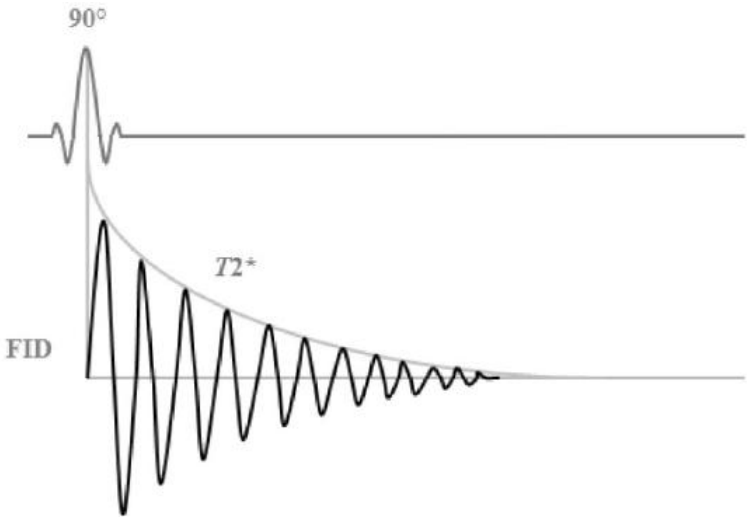
Pro detekci signálu je třeba vytvořit příčnou složku magnetizace, která nebude nulová. To zajistíme vychýlením vektoru magnetizace M_0 ze směru osy *z* směrem k rovině určené osami *x* a *y*. Pomocí radiofrekvenčních (RF) pulzů o frekvenci ω_0 dodáme precedujícím protonům energii. Ty pak s elektromagnetickým impulzem na dané frekvenci rezonují. Dodáním energie přejde část spinů do energeticky náročnější antiparalelní orientace a také dojde ke sfázování magnetických momentů spinů.

Ve výsledku dojde ke sklopení vektoru magnetizace požadovaným směrem. Nenulovou příčnou složku magnetizace již můžeme měřit. Víme, že úhel sklopení magnetizace je přímo úměrný množství dodané energie, tedy velikosti amplitudy a délce RF pulzu.[13]

2.2.4 Signál volné precese

RF pulzy se označují pomocí úhlových stupňů dle toho, jak výrazně dojde po jejich aplikaci ke sklopení vektoru magnetizace M_0. Pokud dodáme soustavě energii 90° RF pulzu, tak je vektor M_0 sklopen do roviny určené osami *x* a *y*, kde rotuje s Larmorovou frekvencí ω_0. V přijímací cívce MR tomografu v této rovině se indukuje napětí. Takto získáme harmonický signál. Během doby po dodání RF impulzu ale dochází k návratu spinů do původní paralelní orientace, čímž postupně klesá amplituda signálu a vektor magnetizace M_0 se vrací do rovnovážné polohy. Rychlost poklesu amplitudy je exponenciální (viz obr. 3). Získaný signál označujeme zkratkou FID (Free Induction Decay). Proces návratu vektoru magnetizace a orientace spinů se nazývá relaxace. [14]

[12] Viz 4
[13] Viz 10 a 11
[14] Viz 10 a 11

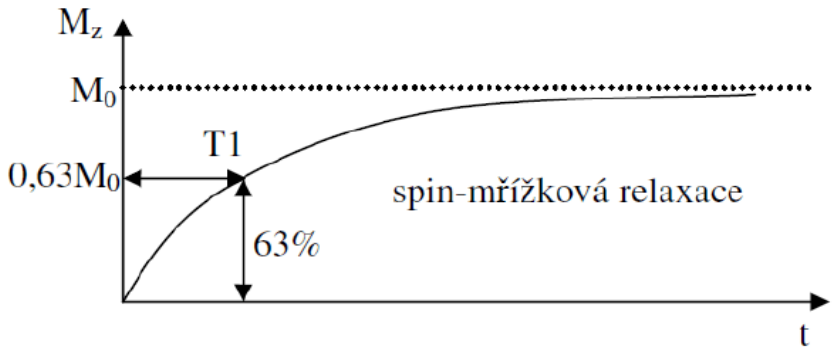


Obr. 3: Signál volné precese. [15]

2.2.5 Relaxace spinového systému

Po excitaci atomových jader nejprve dojde k vychýlení vektoru magnetizace M_0 z rovnovážné polohy, do jiného směru, než má vnější magnetické pole B_0. Následuje ovšem popsaný děj, kdy s exponenciálním růstem rychlosti poklesu amplitudy dojde k návratu do původního stavu. Rychlost relaxace ovlivňují fyzikální vlastnosti tkáně. Relaxace probíhá jak v příčné, tak v podélné složce vektoru magnetizace M_0. Velmi důležité je, že rychlosti relaxací obou složek jsou na sobě nezávislé.

Relaxace podélné složky vektoru M_0 probíhá rychlostí, kterou charakterizuje časová konstanta T_1. Tato relaxace bývá označována jako spin-mřížková relaxace a přebytečná energie je uvolňována jako teplo. Hodnota T_1 udává dobu, za kterou se podélná magnetizace M_z obnoví na 63 % své původní velikosti před excitací RF pulzem (viz obr. 4).



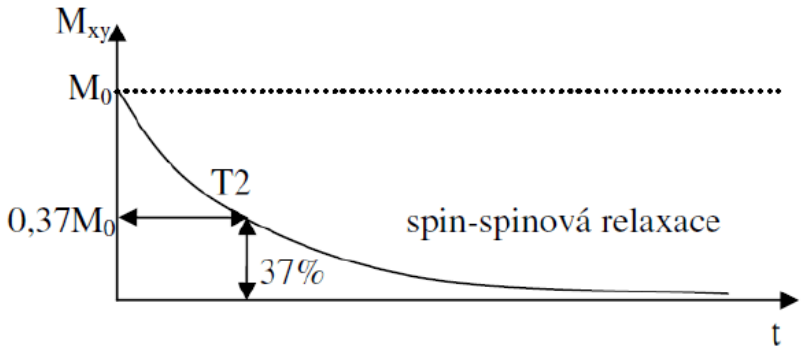
Obr. 4: Průběh T_1 relaxace. [16]

[15] Viz 1

[16] BUREŠ, Martin. *Porovnání MR obrazů ve vybraných oblastech mezi 1,5 T a 3 T.* Praha: ČVUT 2009. Bakalářská práce. Fakulta biomedicínského inženýrství, Katedra biomedicínské techniky.

Při relaxaci příčné složky dochází k jiným dějům. Dodání energie způsobilo, že všechny spiny rotují se stejným fázovým posunem, a proto došlo ke vzniku příčné magnetizace. Relaxace probíhá přesně opačně, dojde tedy k jejich opětovnému rozfázování, a proto vymizí měřitelná příčná složka vektoru magnetizace.

Ztrátu fázové koherence magnetických momentů jednotlivých spinů způsobuje nehomogenita magnetického pole a jejich vzájemná interakce. Časová konstanta T_2, jinak také spin-spinová relaxace, je charakteristickou vlastností každé látky. Hodnota T_2 udává dobu, během které se sníží příčná magnetizace M_{xy} na 37 % své počáteční velikosti, viz obr. 5.



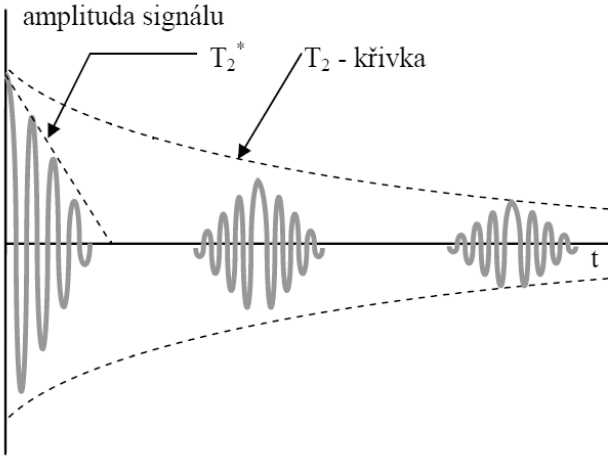
Obr. 5: Průběh T2 relaxace. [17]

Protože ale vnější magnetické pole B_0 není dokonale homogenní, dochází k rozfázování rychleji, než udává konstanta T_2. Dobu poklesu příčné magnetizace tak charakterizuje konstanta T_2*. Rozdíl mezi rychlostmi relaxace dle konstant T_2 a T_2* je vidět na obr. 6.

K oběma relaxačním dějům dochází současně. Konstanta T_1 nabývá hodnot stovek až tisíc milisekund, T_2 hodnot v řádu desítek až stovek milisekund. Hodnota T_2 je pro všechny tkáně nižší, než T_1.

Rozdílné hodnoty konstant T_1 a T_2 u různých typů tkání jsou využity ke konstrukci MR obrazů. K tomuto účelu jsou však nutné složitější sekvence RF pulzů, než popisovaná excitace zapříčiněná pouze jedním 90° pulzem. Jedná se o Sekvence Saturation Recovery, Sekvence Spinového Echa, Sekvence Inversion Recovery a Sekvence Gradientního Echa. [18]

[17] Viz 16
[18] Viz 10 a 11



Obr. 6: Rozdíl mezi konstantami T2 a T2*. [19]

2.3 Vznik MR obrazu

Abychom získali výsledný MR obraz, potřebujeme určit skutečnou polohu jednotlivých protonů, a také to, jak každý přispívá k matici měřeného signálu.

Pomocí sekvence RF pulzů excitujeme celý spinový systém, abychom během jeho relaxace mohli naměřit signál v přijímacích cívkách. K vytvoření prostorového kódování signálu se využívají gradienty magnetického pole ve všech třech směrech. Tyto gradienty jsou tvořeny pomocí gradientními cívkami umístěných ve třech základních směrech – osách *x*, *y* a *z*. Hodnota gradientu se udává v mT/m.

Kombinací gradientních cívek lze vytvořit prostorový gradient směřující libovolným směrem, a tím i snímat jakkoli skloněnou vrstvu obrazu. Jeden gradient působí v rovině kolmé k požadované vrstvě, dva zbývající gradienty (směr fázového a frekvenčního kódování) jsou vzájemně kolmé v rovině prvního gradientu.

Zapnutím gradientu (G_z) získáme lineární nárůst základní magnetické indukce o hodnotu tohoto gradientu podél zvoleného směru, pro příklad podél směru osy *z* (směr je identický s osou magnetu). Pro hodnotu magnetického pole v tomto směru platí následující rovnice

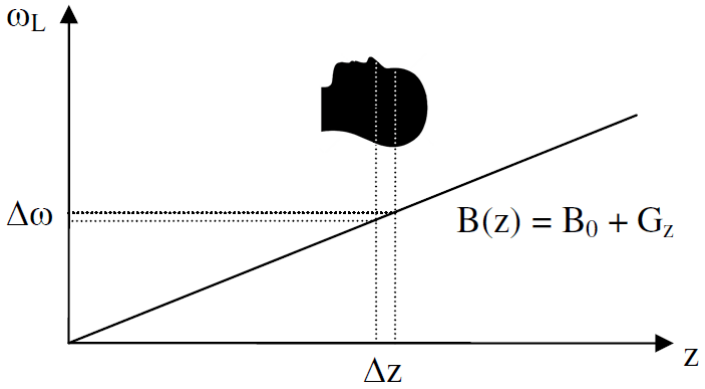
$$B(z) = B_0 + G_z. \quad (5)$$

Spolu se změnou magnetického pole podél tohoto směru se také mění precesní Larmorovy frekvence. [20, 21]

[19] Viz 10 a 11
[20] Viz 10 a 11
[21] Viz 4

2.3.1 Výběr vrstvy

Pokud chceme vybrat jednu vrstvu z celého prostoru, použijeme tzv. selektivní RF pulz, tj. současně zapneme gradient pole a vyšleme excitační RF pulz. Polohu a šířku vrstvy (z) ovlivňuje šířka pásma ω RF pulzu a strmost gradientu (viz obr. 7). Dosáhneme excitace spinů ležících ve vrstvě. U jiných spinů nedojde k rezonanci. [22]

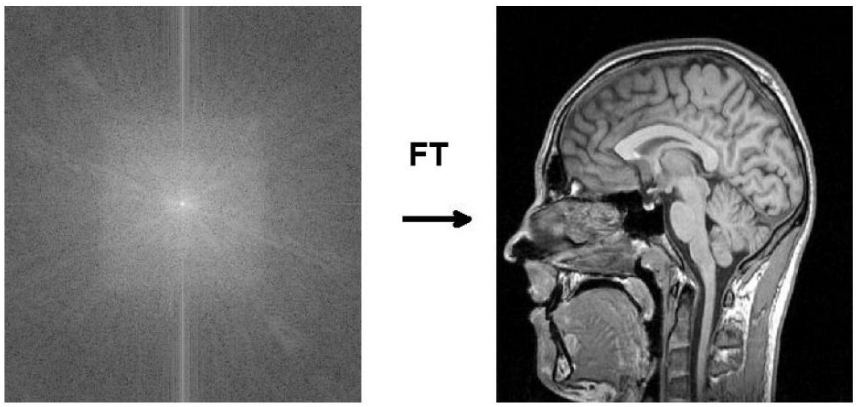


Obr. 7: Výběr vrstvy. [23]

2.3.2 Fourierova transformace k-prostoru

Každý pixel obrazu je ve výsledné matici určen fází (podél směru fázového kódování) a frekvencí (ve směru frekvenčního kódování). Při vlastním měření MRI dochází k opakování sekvence RF pulzů podle počtu řádků matice. Časový průběh gradientů je stejný kromě hodnoty gradientu fázového kódování. Tato hodnota se mění v závislosti na řádku.

Tyto signály jsou ukládány do matice k-prostoru (viz obr. 8). Z tohoto k-prostoru lze 2D Fourierovou transformací získat obraz.[24]



Obr. 8: Fourierova transformace k-prostoru. [25]

[22] Viz 4
[23] Viz 1
[24] Viz 9
[25] Viz 4

2.3.2 Vlastnosti MR obrazu

Výsledná kvalita MR obrazu je ovlivněna několika faktory. Základním je poměr signálu a šumu, další jsou kontrast obrazu, prostorové rozlišení a přítomnost rušivých artefaktů v obraze.

2.3.2.1 Poměr signálu a šumu

Abychom získali lepší poměr signálu a šumu, můžeme např. zvýšit magnetickou indukci. Při rostoucí magnetické indukci roste úroveň signálu kvadraticky a množství šumu lineárně. Proto poměr signál/šum roste s hodnotou magnetické indukce lineárně.[26]

2.3.2.2 Kontrast obrazu

Kontrast obrazu ovlivňuje mnoho faktorů. Jedná se například o relaxační konstanty, perfuze spinů, proudění krve a jiné. Pro strukturální MR obrazy se využívá nejčastěji kontrast získaný z rozdílu relaxačních časů, u měření fMRI se nejčastěji využívá tzv. BOLD efekt, tedy poměr průtoku okysličené a neokysličené krve.[27]

2.3.2.3 Prostorové rozlišení obrazu

3D obraz získaný pomocí MR je složený z prostorových bodů – voxelů. Pokud při stejné hodnotě magnetické indukce zmenšíme voxel, snížíme poměr signál/šum (signál klesá, šum zůstává na stejné úrovni).[28]

2.3.2.4 Artefakty

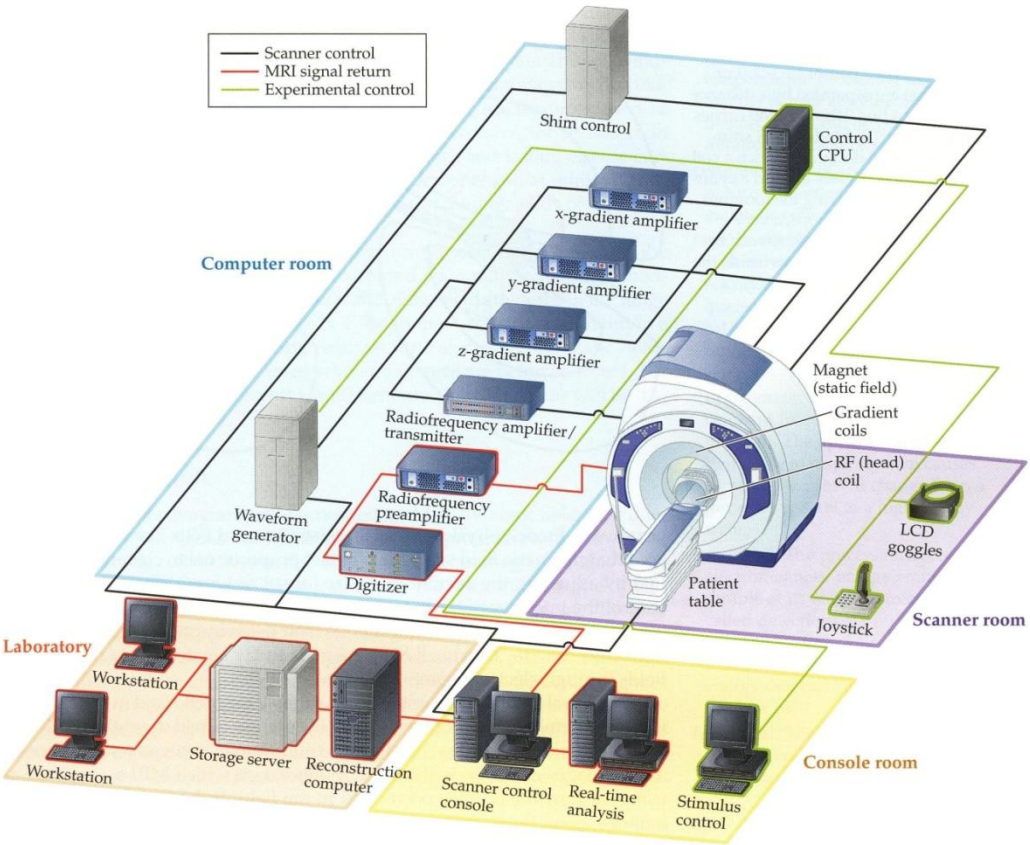
Ve výsledném MR obrazu také můžeme nalézt artefakty, které jsou odlišné od skutečného prostorového rozložení snímaných tkání. Mezi artefakty, které můžeme potlačit, patří např. i rušení jiným vysokofrekrevenčním signálem. Takový signál se do stíněné místnosti může dostat jakýmkoli kovovým vodičem. [29]

2.4 Uspořádání pracoviště MR

Běžné uspořádání MR tomografu a souvisejících zařízení zachycuje obr. 9. Fialově je na něm vyznačena místnost s vlastním tomografem, která musí být stíněná od RF pásma, které by mohly způsobit artefakty v obraze. Žlutě označená oblast obsahuje ovládací počítače. Na pracovišti Základny radiodiagnostiky a intervenční radiologie (ZRIR) Institutu klinické a experimentální medicíny (IKEM) se pro funkční MR využívají dva počítače. Z prvního se ovládá vlastní snímání obrazů (nastavení parametrů sekvence), druhý slouží pro spouštění stimulace systémem SySy, který je podrobněji popsán v kapitole 2.5.

[26] Viz 1
[27] Viz 1
[28] Viz 1
[29] Viz 1

Při spouštění vlastní stimulační série je nutné nejprve nastavit MR tomograf do požadované konfigurace a vyčkat okamžiku, kdy bude připraven k odstartování měření. Poté se spouští vlastní stimulační série na druhém počítači, aby vyčkala synchronizačních impulzů, které ovšem přijdou až poté, co je měření definitivně odstartováno z prvního počítače.



Obr. 9: Uspořádání MR tomografu a ovládácích zařízení. [30]

2.5 Současný stav

V současnosti se na ZRIR IKEM pracovišti používá pro funkční MR zobrazování tomograf o magnetické indukci 3 T Siemens Trio Tim. Pro promítání stimulačních sérií se nyní využívá stimulační systém SySy, který byl vytvořen Milanem Váchou a Pavlem Vachem v rámci diplomových prací na elektrotechnické fakultě ČVUT v roce 2000. [31] Systém SySy byl napsán v programovacím prostředí Delphi. Program je v současnosti používán na počítači s operačním systémem Windows XP.

Systém SySy se skládá ze softwarové a hardwarové části. Softwarovou část tvoří ovládací program, ve kterém se nastavují různé průběhy stimulací (paradigmata) včetně jejich parametrů a který také zajišťuje ukládání potřebných dat o reakcích vyšetřované osoby.

[30] Viz 10
[31] Viz 1

Hardwarová část je tvořena měřicí kartou AD14DSP v PC a kartou čtyřkanálového optického přijímače, postupně doplněným o obvod prodlužující synchronizační impuls, přepínač akustických kanálů a dalších modulů, které byly vyrobeny v průběhu používání zařízení podle měnících se požadavků jednotlivých měření. [32]

Systém SySy je schopen stimulovat zrakové a sluchové vjemy prostřednictvím promítání obrázků, resp. přehrávání zvuku. Systém je dále schopen registrovat externí digitální signál ze čtyř tlačítek s frekvencí 100 Hz, tedy s přesností 10 ms. Není ovšem možné registrovat stavy, kdy bylo stisknuto více než jedno tlačítko zároveň. Pomocí logických obvodů je zajištěno, že z pohledu měřicí karty je stisknuto pouze to tlačítko, které bylo stisknuto jako první. Při sledování stisků tlačítek je totiž měřeno napětí na vstupu karty AD14DSP, kdy první tlačítko poznáme díky vstupnímu napětí 2,5 V, druhé se projeví napětím 5 V, třetí 7,5 V a čtvrté 10 V. V případě, že je stisknuto první a druhé tlačítko zároveň, bylo by na vstupu karty napětí 7,5 V, indikující stisk třetího tlačítka.

Při promítání lze rozdělit interval TR pouze na dva podněty, tedy grafický výstup lze změnit nejvýše dvakrát častěji než s každým TR. Interval TR je možné rozdělit pouze jedním poměrem délky jednotlivých částí. Je-li ovšem nutné registrovat vstup z tlačítek, není z technických důvodů možné rozdělit interval TR na více částí. U zvukových souborů, které mají být spuštěny se zpožděním, lze tento nedostatek obejít jejich slučováním nebo vložením ticha na začátek souboru.

Po skončení měření je vytvořen textový soubor se záznamem událostí v průběhu měření. V tomto souboru každý řádek reprezentuje stav v čase (jeden sample), tedy napětí odpovídající různým tlačítkům.[33] Tyto soubory je nutné strojově zpracovat (z napětí na vstupu zjistit, které tlačítko bylo stisknuto), a až poté přejít k vyhodnocení výsledků měření, kdy lze zjišťovat správnost, rychlost a délku odezvy nebo naopak chybnou reakci. Stimulace přehráváním videí není v tomto softwaru možná, stejně jako registrace analogových signálů z joysticku nebo odesílání informací na digitální výstupy.

Aby bylo možné provádět vyšetření, kde se využívá joystick nebo stimulace prostřednictvím videí, vzniklo několik jednoúčelových programů pro konkrétní měření. Tyto programy nevyužívají měřicí kartu AD14DSP, ale A/D převodník Arduino Uno. Současný software tak zaostává za aktuálně používaným hardwarovým vybavením i za požadavky na provádění vyšetření.

Jednotlivá paradigmata se ukládají do textových souborů, aby bylo možné je měnit i v jednoduchých textových editorech bez nutnosti spouštět program. Tento způsob ukládání také umožňuje strojové generování těchto souborů. Je pevně stanoveno, ve kterém adresáři se musejí nacházet soubory používané pro stimulace.

[32] Viz 1
[33] TINTĚRA, Jaroslav; RYDLO, Jan: *Stimulační systém pro fMRI – návod k použití*

2.6 Jiná řešení

V této kapitole je uváděn krátký přehled řešení stimulačních systémů, které se používají pro vyšetřování funkční magnetické rezonance na jiných pracovištích v České republice a na Slovensku. Informace v kapitolách 2.6.1 až 2.6.5 pocházejí od vedoucího práce doc. Ing. Jaroslava Tintěry, CSc.

2.6.1 Klinika v Nových Butovicích

Na pracovišti v Nových Butovicích využívají systém od společnosti InVivo, což je dceřiná společnost firmy Philips, která dodala MR tomograf. Velkou výhodou tohoto systému (v kombinaci s tomografem prakticky od stejné firmy) je snadné spouštění, kdy stačí potvrdit běh pouze jedinkrát a oba počítače se samy synchronizují. Značnou nevýhodou je však vysoká cena zařízení (řádově přes dva miliony Kč)

Nutnost synchronizace je zároveň i komplikací v případě i malých změn existujícího schématu stimulace. Programování nových schémat je velmi komplikované.

K promítání obrázků stimulace slouží LCD displej umístěný na pojízdném stojanu. Jelikož se displej nachází uvnitř stíněné místnosti, je nutné stínit přívodní kabely. Zároveň je nutné umístit displej vždy na stejné místo.

I pro vlastní stimulační série má popisovaný systém mnoho restrikcí, například vyžaduje pouze jeden rozměr obrazové matice. Videa přehrávat v tomto systému nelze. Výhodou naopak je možnost nastavit různé atributy obrázku – dobu zobrazení nebo například blikání.

2.6.2 Nemocnice Na Homolce

Pro funkční MR zobrazování v Nemocnici Na Homolce využívají svůj vlastní software, který pro svůj běh vyžaduje operační systémem DOS. Tento operační systém nemá plánovač procesů, a tak výhodou je nepřerušitelnost běhu takto spuštěného programu. Pro vlastní stimulaci ovšem využívá pouze nápisy, nelze zobrazovat ani obrázky, ani videa.

2.6.3 Fakultní nemocnice v Ostravě

V Ostravě je pro měření stimulací využíván systém od společnosti Nordic Neurolab. Pro vlastní zobrazování promítané série tento systém využívá brýlí se dvěma LCD displeji. Díky tomu pacient není rušen vnějšími vlivy, naopak je nutné brýle individuálně seřizovat pro každou vyšetřovanou osobu. Softwarová část je schopna provádět audio-visuální stimuly, dává informace o průchodu stimulací a poskytuje nástroje pro analýzu naměřených dat.

2.6.4 FN Motol, FN Hradec Králové, FN Brno

V těchto pracovištích se využívají stimulační systémy, které různou měrou vycházejí ze systému SySy, používaného na pracovišti ZRIR IKEM. V nemocnici v Motole a Hradci

Králové je systém využíván i pro klinická vyšetření. Další šíření a významnější rozvoj tohoto systému není možné očekávat, neboť je pevně vázán na převodní kartu AD14DSP od firmy JanasCard, která se již nevyrábí.

2.6.5 Nemocnice na Kramároch, MR Doktor (Bratislava)

Na pracovišti MR Doktor v Bratislavě je pro měření funkční MR využívána kombinace A/D převodníku Arduino, který sériově komunikuje s jednoduchými programy vytořenými v prostředí Processing. V aktuálně používané konfiguraci nejsou využívána tlačítka pro registraci odezvy vyšetřovaného pacienta.

2.6.6 Psychopy

Jedním z možných řešení pro realizaci stimulačního software pro fMRI je využití open-source aplikace Psychopy. Program využívá knihovnu OpenGL a byl vytvořen v jazyce Python. Aplikace je platformě-nezávislá a lze s ní zobrazovat jakékoli myslitelné druhy stimulace. Pomineme-li běžně používané přehrávání zvuku či videí, jedná se například o zobrazování bodů, textu a lineárních nebo kruhových barevných přechodů. Aplikace podporuje použití více monitorů. Komunikaci s jinými zařízeními lze z Psychopy provádět mnoha různými způsoby, jedná se například o sériové nebo paralelní porty.[34]

2.7 Cílový stav

Nový software by měl dle požadavků zadavatele nahradit stávající systém SySy i jednoúčelové programy, vytvořené pro provádění konkrétních měření. Zároveň má rozšířit možnosti stávajících stimulací prostřednictvím obrazů a zvuků do oblastí, které v současné době není možné provádět.

Jedná se zejména o rozdělení intervalu TR na libovolný počet různě dlouhých částí, jejichž délka nemusí být stejná v průběhu celého měření. Druhou důležitou vlastností bude možnost zvolit zdrojové složky souborů tak, aby bylo snadné pro stejné stimulační série zaměnit zdrojové soubory (pokud tyto jsou stejně pojmenované), například dle konkrétního vyšetřovaného pacienta. Další rozšíření bude spočívat v plnohodnotném zařazení stimulace přehráváním videí. Tento druh podnětů je v současnosti možné provádět pouze za pomoci jiných programů nekompatibilních se současným systémem, což ztěžuje jak přípravu měření, tak i jeho praktické provádění.

V oblasti registrace signálů bude nejvýraznějším přínosem nové aplikace možnost registrovat analogové signály, zejména z joysticku. Tento signál může být podle potřeby ovlivněn a s minimální prodlevou zobrazen na obrazovém výstupu. Software také bude schopen zasílat nastavitelné digitální signály pro přepínání relé. Tato relé mohou být dále

[34] http://psychopy.org/

využita při konkrétních měřeních k různým účelům (například spínání ventilů pro stimulaci čichovými vjemy). Dle předpokladů bude stačit odesílat osm binárních hodnot.

Mezi vlastnosti, které současný software zvládá, patří registrace synchronizačních pulzů, registrace signálu digitálních tlačítek, stimulace promítáním obrazů i promítání zvuku, a to i kombinovaná. Současný systém SySy umožňuje také stimulaci prostřednictvím promítnutí nápisu. Tato možnost ovšem již není v současné době využívána a lze ji snadno nahradit stimulací obrazem, která nabízí větší možnosti.

Jinou požadovanou schopností je i umožnění jednosměrné zvukové komunikace s vyšetřovanou osobou pomocí mikrofonu, dále zobrazování různých výstupů na jednotlivá zobrazovací zařízení (konkrétně stimulaci na projektor a hlavní okno na monitor ovládacího počítače) a ukládání souborů s popisem stimulační série v textovém formátu, aby bylo možné je i nadále upravovat jak v aplikaci, tak bez použití programu v textovém editoru nebo je strojově generovat. Vedlejším požadavkem je například možnost kalibrace středu stimulační obrazovky nebo přeložitelnost aplikace do jiných jazyků.

3 Analýza

3.1 Zhodnocení problému

Pro vlastní měření je nutné registrovat synchronizační impulz co nejpřesněji. Tento impulz přichází z MR tomografu a lze jej získat dvěma způsoby – buď jako elektrický nebo jako optický signál. Je požadováno, aby bylo možné rozdělit interval mezi dvěma synchronizačními pulzy na libovolný počet částí. Start stimulace tedy má být určován časovým odstupem od posledního synchronizačního impulzu.

Aby bylo možné vyhodnocovat aktivace a klidové stavy jednotlivých center mozku, potřebujeme zajistit, aby stimulační podněty byly prováděny vždy ve stejný okamžik a stejným způsobem.

Jedním z úkolů, který aplikace musí zvládat, je registrace digitálních a analogových vstupů. Přímá registrace těchto tlačítek, stejně jako synchronizačního impulzu, bude prováděna na A/D převodníku, se kterým bude aplikace svázána a bez kterého nebude možné ji plnohodnotně používat. Není preferován konkrétní způsob komunikace s A/D převodníkem, jediným požadavkem je její dostatečná četnost, aby zpoždění vzniklé zpracováním a přenosem dat bylo blízké nule. Výběr A/D převodníku bude popsán v kapitolách 3.3 a 4.2.

Dalším požadavkem zadavatele je možnost jednosměrné komunikace s vyšetřovanou osobou. Tento požadavek přímo souvisí s přehráváním zvuků. Spolu s MR tomografem jsou dodávána vzduchová sluchátka. Do těchto sluchátek (a také do reproduktoru ve stropu místnosti) je zaveden intercom, pomocí kterého již lze jednosměrně komunikovat s pacientem. Nevýhodou vzduchových sluchátek je jejich nízký frekvenční rozsah, proto se na pracovišti ZRIR IKEM používají pro jiná, než funkční měření. Pro vyšetření fMRI jsou používána speciální sluchátka, která nabízejí větší frekvenční rozsah zvuku. Tato sluchátka nejsou propojena s intercomem a má-li je pacient na uších, nelze mu sdělit různé informace a pokyny, např. zbývající čas měření. Je třeba zajistit, aby zvuková komunikace nemohla probíhat v době, kdy probíhá stimulační série, protože by sdělováním informací mohla být ovlivněna naměřená data.

Výstup z jednosměrné komunikace má být posílán na běžný zvukový výstup počítače, odkud pokračuje přes filtr elektrickým kabelem do stíněné místnosti. Stejným způsobem je do stíněné místnosti přiváděn zvukový signál při měřeních, kdy se využívá zvukových stimulů.

3.2 Uživatelé aplikace

Mezi uživatele aplikace budou na pracovišti ZRIR IKEM patřit osoby, které mají zkušenost se současným softwarem SySy. Jedná se o vysokoškolsky vzdělané osoby technického zaměření. Jde o osoby, které výpočetní techniku používají pravidelně a na

pokročilé úrovni. Zároveň často používají radiologické termíny související s měřením na magnetické rezonanci. Na jiných pracovištích ovládají podobné systémy i laboranti.

Způsob používání aplikace bude dvojí. První, méně častý, zahrnuje tvorbu a úpravu nastavení jednotlivých měření a jeho stimulačních sérií. Druhou variantou používání aplikace je spouštění požadovaných měření, jako i ukázky, kdy je před vlastním měřením ukazováno vyšetřovaným osobám, co se po nich požaduje pro zdárný průběh vyšetření.

3.3 A/D převodník

Na pracovišti ZRIR IKEM se pro měření fMRI v současnosti využívá měřicí karta AD14PCI s DSP (AD14DSP) a A/D převodník Arduino Uno. AD14PCI slouží pro příjem synchronizačních impulzů a registraci signálů z tlačítek, pro snímání dat z MR-kompatibilního joysticku je využíváno Arduino spolu s programem vytvořeným v prostředí Processing.

Při tvorbě nového SW bude zvolen nový převodník. V úvahu přicházejí buď modul AD14ETH (výrobce JanasCard), nebo některý výrobek z rodiny Arduin. Podmínkou je přesné a rychlé zpracování synchronizačních impulzů a rychlá komunikace s počítačem. Signál z tlačítek je ze stíněné místnosti přenášen opticky, poté je převáděn na napěťové úrovně TTL logiky. Proto i vybraný převodník musí na vstupu pracovat s TTL logikou.

3.3.1 AD14ETH

Modul AD14ETH je měřicí modul komunikující přes Ethernet 10/100 Mbit určený pro rychlá a přesná měření v laboratoři a průmyslu. Ethernet rozhraní zajišťuje galvanické oddělení pro stejnosměrné signály, což výrazně snižuje problémy s rušením a rozšiřuje možnosti modulu. Modul slouží jako 14 bitový A/D převodník. Komunikuje přes rozhraní Ethernet rychlostí 10/100 Mbit/s protokolem TCP/IP.[35] V síti pracuje jako server s nastavitelnou IP adresou. Modul je napájen střídavám napětím 12 V / 50 Hz.

Na tomto modulu nalezneme sedm TTL kompatibilních digitálních vstupů a stejný počet digitálních výstupů. U výstupů má logická 1 pouze úroveň 3,3 V. Pro ovládání modulu je spolu s kartou dodávána i DLL knihovna. Parametry modulu jsou uvedeny v tabulce 2.

rozlišení	14 bitů, odstup signál-šum 74 dB
vzorkovací rychlost	400 kHz
vstupní rozsah	+/-5 V / 0 - 10 V, zesílení volitelné 1, 2, 4, 8 x
počet vstupů	16 (rozšiřitelné na 32)
časovač	rozsah 16 bitů, krok časovače 0,1 ms
počet dig. vstupů	7 (TTL/HCT kompatibilní)
počet dig. výstupů	7 (úroveň logické 1 je 3,3 V)

Tab. 2: Parametry modulu AD14ETH. [36]

[35] JanasCard. *AD14ETH – návod k obsluze*, Praha, 2012
[36] Viz 35

3.3.2 Arduino

Arduino je open-source elektronická prototypovací platforma založená na flexibilním a snadno použitelném hardware a software.[37] Deska Arduina obsahuje několik digitálních I/O pinů a počet analogových vstupů, všechny typy desek obsahují i analogové výstupy. Na desce se především nachází mikroprocesor, který umožňuje vytvořit program definující prováděné činnosti se vstupy, včetně jednoduchých výpočetních operací. S počítačem komunikuje pomocí USB rozhraní. Na něm je vytvořena virtuální sériová linka, která umožňuje rychlou obousměrnou komunikaci.

Na pracovišti ZRIR IKEM jsou k dispozici dvě desky, jedná se o Arduino Uno a Arduino Mega. Arduino Uno je deska s mikroprocesorem od firmy Atmel o frekvenci 16 MHz, flash pamětí o velikosti 32 KiB, šesti analogovými vstupy a 14 digitálními I/O piny. Oproti tomu Arduino Mega2560 nabízí větší množství vstupních i výstupních pinů a obsahuje větší paměť (např. flash o velikosti 256 KiB). Analogové vstupy obou desek pracují s TTL logikou.[38] Parametry desek jsou uvedeny v tabulce 3.

Obě desky mohou být napájeny z počítače pomocí USB portu, jehož prostřednictvím také probíhá sériová komunikace. Možné je i napájení z externího zdroje. Druh napájení je rozpoznán automaticky.

	Arduino Uno	**Arduino Mega**
Microprocesor	ATmega328	ATmega2560
Frekvence procesoru	16 MHz	16 MHz
Provozní napětí	5 V	5 V
Doporučené napájecí napětí	7 – 12 V	7 – 12 V
Limit napájecího napětí	6 – 20 V	6 – 20 V
Počet digitální vstupů a výstupů	14	54
Počet analogových vstupů	6	16
Stejnosměrný proud digitálního I/O pinu	40 mA	40 mA
Stejnosměrný proud pro 3.3V výstup	50 mA	50 mA
Flash paměť	32 KB, z toho 0,5 KB využívá bootloader	256 KB, z toho 8 KB využívá bootloader
SRAM paměť	2 KB	8 KB
EEPROM paměť	1 KB	4 KB

Tab. 3: Srovnání parametrů desek Arduino Uno a Arduino Mega.[39]

3.4 Schéma zapojení

Z předchozích kapitol je zřejmé, že vlastní softwarová aplikace je pouze jedním z dílů řetězce celého stimulačního systému, který tvoří ještě několik dalších hardwarových součástí. Klíčovým bodem systému bude A/D převodník, který zastane a rozšíří funkce poskytované

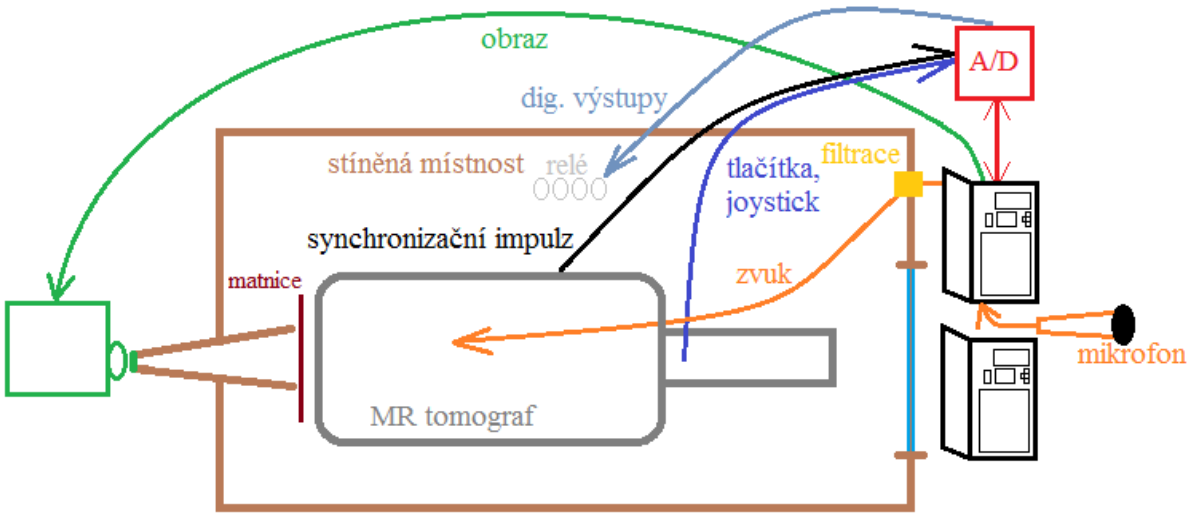
[37] http://arduino.cc
[38] Viz 37
[39] Viz 37

v současnosti již nevyhovující kartou AD14DSP (viz kapitola 2.4). A/D převodník zastane, kromě registrace synchronizačního impulzu zmiňovaného v předchozí kapitole, také funkci prostředníka mezi signálem z tlačítek nebo joystiku a počítačem přijímajícím naměřená data. A/D převodník by také měl ovládat několik relé umístěných ve stíněné místnosti. Signál z tlačítek a joysticku a pro relé je do stíněné místnosti veden optickými kabely, do A/D převodníku budou zapojena prostřednictvím fototranzistorů nebo optických příjmačů.

Aby nebylo nutné stínit elektrický kabel od tlačítek a joysticku, bude A/D převodník přijímat signály z joysticku prostřednictvím fototranzistorů, z tlačítek pomocí optických komponent HFBR od firmy Avago Technologies. Stejně tak synchronizační impulz je přijímán opticky.

Je také nutné pomocí astabilního klopného obvodu prodlužovat délku synchronizačního impulzu z původní jedné mikrosekundy na několik desítek milisekund, aby bylo zajištěno, že nedojde k jeho přehlédnutí.

Mezi vstupní zařízení, která budou připojena k ovládacímu počítači, patří také mikrofon. Signál pro projektor a sluchátka může být z počítače vyveden pomocí standardních výstupů. Zvukový výstup, který bude veden elektrickými vodiči do speciálních sluchátek do stíněné místnosti, je nutné filtrovat, aby nevznikaly nežádoucí artefakty ve výsledném obraze (viz kapitola 2.3.2.4). Schéma všech vstupně-výstupních zařízení a jejich vzájemné komunikace je na obr. 10. Vlastní měření MR tomografu se nastavuje z jiného počítače, než ze kterého se spouštějí stimulace.



Obr. 10: Hardwarová zařízení v systému.

3.5 Dostupné knihovny, výběr platformy

Pro tvorbu aplikace byl zvolen jazyk Java. K této volbě došlo kvůli množství dostupných knihoven, pozitivních zkušeností s javovskými aplikacemi na pracovišti ZRIR IKEM a případně snazší přenositelnosti aplikace na jinou platformu. Operace v programech vytvořených v jazyce Java nemusejí být nutně pomalejší, než u programů v C nebo C++. [40]

Proto budou v následujících odstavcích představeny knihovny, které se mohou při tvorbě aplikace pomoci řešit očekávatelné problémy. Důležitým hlediskem výběru knihovny může být i licence, pod kterou je šířena. Vyvíjený software může například využít externí knihovny pro přehrávání audio a video souborů v různých formátech či pro promítání obrázků. V případě využití Arduina jako A/D převodníku bude využita také knihovna implementující komunikaci po sériovém portu.

3.5.1 JNA

Knihovna Java Native Access (JNA) je určena pro volání DLL metod a funkcí z DLL knihoven v aplikaci napsané v jazyce Java. Pro volání takových funkcí stačí správně definovat v Javě jejich interface a určit zdrojový DLL soubor. Při volání funkcí se pak postupuje stejně, jako při volání jakékoli jiné funkce. Při použití této knihovny pak programátor nemusí psát žádný další kód určený k propojení s DLL souborem. [41] Drobný rozdíl nastává při použití pointerů u primitivních typů. Jazyk Java umožňuje předávat primitivní typy jako hodnotu. Proto knihovna nabízí třídy *Pointer* a *Memory*, díky kterým mj. lze předat primitivní typ jako referenci a získat zpět funkcí zapsanou hodnotu.

Licence: LGPL a Apache Licence

3.5.2 RXTX

Knihovna RXTX rozšiřuje možnosti standardního Java Development Kitu (JDK) o nepřetržitou a event-based sériovou a paralelní komunikaci.[42] Knihovna umožňuje vyhledat dostupné porty a po otevření komunikace lze přijímat a odesílat data pomocí standardních tříd InputStream a OutputStream. Knihovna RXTX se skládá ze dvou částí: Nativního driveru pro sériovou komunikaci (dll) a vlastní javovské knihovny RXTXcomm.jar.

Licence: LGPL

3.5.3 jSSC

Java Simple Serial Connector (jSSC) je další knihovna, která umožňuje snadno realizovat běžnou sériovou komunikaci. jSSC nabízí snadno uchopitelné API a je stále vyvíjena. Současná verze (2.8.0) byla vydána v březnu 2014. Obě popisované knihovny lze

[40] A4M33PAL, zimní semestr 2012, https://cw.felk.cvut.cz/wiki/courses/a4m33pal/cviceni/java_io
[41] https://github.com/twall/jna
[42] http://rxtx.qbang.org/wiki/index.php/FAQ

využít jak v aplikacích využívajících 32bitovou, tak 64bitovou architekturu. Zásadním rozdílem této knihovny oproti RXTX je čtení dat, které neprobíhá pomocí Input a OutputStreamu, ale pomocí vlastního API.

Licence: LGPL

3.5.4 Java Media Framework

JMF umožňuje přidávat do aplikací v jazyce Java podporu pro přehrávání, transformace a streamování zvukových a video souborů.[43] Nevýhodou JMF je, že neumí přehrávat některé typy médií, například MPEG-2[44] a pro přehrávání MP3 souborů je nutné nainstalovat další plugin.

Významnou výhodou JMF je možnost přehrávat zvuky z instance jakéhokoli potomka třídy InputStream, tedy i z instancí ByteArrayInputStream, které budou obsahovat data načtená v paměti aplikace. Obdobně při přehrávání videa obsahuje metodu prefetch(), která zajistí načtení a dekódování úvodního bloku dat videa.

Licence: Sun Community Source Licence

3.5.5 JLayer

Knihovna JLayer je určena pro dekódování, konvertování a přehrávání MP3 souborů v reálném čase.[45] Pro svůj běh nevyžaduje přítomnost JMF. Nevýhodou je, že její aktuální verze vyšla na konci roku 2008.

Licence: LGPL

3.5.6 MP3Transform

Knihovna MP3 Transform obsahuje dekodér a přehrávač MP3 souborů. Pomocí této knihovny je také možné převádět MP3 soubory na WAV. Její zdrojový kód vychází z knihovny JLayer.[46] Nejnovější verze knihovny (0.81) vyšla v dubnu 2010.

Licence: LGPL

3.5.7 VLCJ

Projekt VLCJ umožňuje tvorbu přehrávačů zvuků a videa v javovských programech. Při použití této knihovny je do okna aplikace vložena instance VLC přehrávače. Taková aplikace pak podporuje všechny operace, stejně jako VLC. Ke svému běhu vyžaduje kromě zmiňovaného přehrávače také knihovnu JNA. Jak VLC přehrávač, tak tato knihovna, jsou šířeny pod copyleftovou GPL, na rozdíl od ostatních knihoven šířených převážně pod LGPL.

---

[43] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html
[44] https://community.oracle.com/message/5419391
[45] http://www.javazoom.net/javalayer/about.html
[46] http://code.google.com/p/mp3transform/

S touto knihovnou lze přehrát velké množství audio a video formátů. Nevýhodou při použití knihovny VLCJ v časově synchronizované aplikaci je, že neumožňuje přehrávat video z RAM paměti.[47]

Licence: GPL

3.5.8 JFFMPEG

JFFMPEG je balík kodeků pro JMF. Umožňuje přehrávat velké množství běžně používaných audio a video formátů. Je založen na knihovně FFMPEG.[48] Nevýhodou je, že již není vyvíjen.

Licence: GPL nebo LGPL nebo BSD licence.

3.5.9 Xuggler

Knihovna Xuggler slouží k práci s videem v jazyce Java. Umožňuje dekódvat téměř jakýkoli formát videa.[49] Poskytuje velmi nízkoúrovňový přístup k videu. Jeho nevýhodou je, že již není vyvíjen.

Licence: GLP nebo LGPL

3.6 Používané názvy

Před zahájením vývoje vlastní aplikace a jeho popisem je nutné si ujasnit používané názvosloví a vyjmenovat používané odborné termíny. Některé názvy vycházejí z jejich použití v původním systému SySy (viz kapitola 2.6) a pro zachování kontinuity budou použity i v nové aplikaci.

V systému SySy se pro jedno měření, které obsahuje stimulační podněty a má určité vlastnosti, používaly výrazy *nastavení* nebo *nastavení měření*. Výraz *nastavení* je dle vedoucího práce výstižný a je dobré jej zachovat. Série promítaných podnětů se nazývá *stimulační série*. Aby nedocházelo k záměnám názvů, bude sada vlastností každého *nastavení* nazývána jeho *parametry*.

V parametrech je obsažen údaj o počtu *epoch* a počtu *dynamik* v době *klidu* a *stimulace*. Celé měření se skládá z určitého počtu *epoch*, každá *epocha* obsahuje dobu *klidu* a dobu *stimulace*. Délky těchto dob jsou v celém *nastavení* stejné. Jedna *dynamika* je jedno nabrání obrazu hlavy MR tomografem, po kterém je odeslán *synchronizační impulz*. Každé nastavení se skládá z *klidu* a *stimulace*, neboť je měřen rozdíl v aktivacích během těchto dvou stavů. *Počet dynamik v době klidu* tedy reprezentuje počet synchronizačních impulzů, které

[47] http://www.capricasoftware.co.uk/projects/vlcj/faq.html
[48] http://jffmpeg.sourceforge.net/index.html
[49] http://www.xuggle.com/xuggler/documentation

jsou zaregistrovány v době, kdy vyšetřovaný pacient nemá vyvíjet mozkovou činnost. Doba mezi dvěma *synchronizačními impulzy* se nazývá *repetiční čas* a označuje se zkratkou TR.

Slovo *nastavení* již je používáno, proto nastavení parametrů celého programu bude označováno jako *konfigurace*. V textu se píše o *vstupech* a *výstupech.* Za *vstup* jsou považovány vstupní hodnoty reprezentující stisk tlačítek a souřadnice joysticku, tedy údaje, které vstupují do počítače z A/D převodníku. Naopak *výstupy* jsou posílány A/D převodníku, který může jejich prostřednictvím spínat externí zařízení.

4 Návrh řešení

Kapitola Návrh řešení obsahuje plánované rozdělení funkčností aplikace na menší logické celky. Další část kapitoly tvoří výběr vhodného A/D převodníku, na něj navazuje výběr konkrétních externích knihoven použitých jak pro provádění stimulací, tak pro implementaci komunikace se zvoleným A/D převodníkem. Zaměřím se i na popis způsobu, jakým budou ukládána data programu.

4.1 Koncepce programu

Jakoukoli větší aplikaci je vhodné rozdělit na několik menších celků, které spolu budou komunikovat, ale jinak na sobě nebudou závislé. Tyto celky se později dají snadněji udržovat nebo rozšiřovat, výhodou také může být jejich snadné znovupoužití v jiné aplikaci.[50] V případě potřeby také lze tyto celky nahradit jinými se stejnou nebo podobnou funkčností a podobným rozhraním. V ostatních souborech pak pouze stačí pozměnit několik řádků kódu.

Popisovanou aplikaci tvořící softwarovou část stimulačního systému pro fMRI lze rozdělit do šesti převážně samostatných částí. Jedná se zejména o nastavení stimulace, komunikaci se zvoleným A/D převodníkem, důležité je vlastní provádění stimulace a zpracovávání digitálních a analogových vstupů. Za úplně samostatnou část aplikace lze bezesporu prohlásit oblast zabývající se jednosměrnou komunikací k vyšetřované osobě. Za dobu používání aplikace bude potřeba alespoň jedenkrát provést základní nastavení a kalibraci zobrazovacího projektoru. Nakonec bude věnována pozornost i na návrhu programu pro zvolený A/D převodník.

Zdrojové kódy aplikace je také vhodné pro větší přehlednost dělit do bloků dle návrhového vzoru Model View Controller, čímž se dále zvýší modularita programu.

4.1.1 Nastavení stimulace

Stimulační software pro fMRI musí především zajišťovat možnost nastavení, ukládání a provádění stimulací. U jednotlivých částí série se eviduje čas začátku a typ stimulace, případně také její zdrojový soubor. Jedinou výjimkou může být měření, při kterém se zobrazují na obrazový výstup aktuální vstupní data z joysticku. Zde bude potřebné stanovit i další údaje. Pokud bude aplikace poskytovat samostatný nástroj pro tvorbu modelů sítí využívaných jako pozadí pro kurzor joysticku, bude zdrojovým souborem stimulace právě soubor s údaji o zobrazené síti.

Kromě nastavení stimulační série bude nutné nastavit i některé obecné parametry měření, jako například počet dummy pulzů (tj. počet synchronizačních impulzů, které přijdou

[50] Přednáška č. 10 z předmětu A7B36USI, zimní semestr 2011

ještě před zahájením snímání hlavy) nebo určit počet sledovaných digitálních a analogových vstupů.

Data stimulační série budou ukládána do souborů, ukládání a načítání dat ovšem bude za rozhraním, které v případě potřeby umožní s minimálním úsilím doplnit i jiný způsob ukládání dat. Podrobněji jsou důvody ukládání nastavení stimulací do souborů a formátu těchto souborů uvedeny v kapitole 4.4.

Pro stimulace je nutné zabývat se kromě zpoždění a typem simulace také jejím provedením, které je specifické pro různé typy stimulací. Před zahájením celé stimulační série musí dojít k přípravě dat, aby poté docházelo ke stimulacím s minimálním možným zpožděním od zadaného času startu. Zároveň je třeba některé druhy stimulací nejen startovat, ale i ukončovat.

Z předchozích odstavců vyplývá, že nastavení stimulační série je vhodné rozdělit na dvě samostatné části. Jedna má na starost uchování údajů série a umožňuje jejich úpravy. V této části mohou být všechny druhy reprezentovány jednou obecnou třídou držící společné vlastnosti, a která dále drží informaci, který druh stimulace konkrétní instance reprezentuje. Druhá část má za úkol vlastní provádění nastavených stimulačních podnětů a je vysoce specializovaná dle druhu stimulace. Proto je vhodné zde využít polymorfismu tříd, které budou implementovat nějaké společné rozhraní.

Tyto dvě části by měla spojovat Factory třída, která v okamžiku potřeby transformuje obecné instance sloužící pro práci s daty na konkrétní instance specializovaných tříd, které budou využity pro vlastní stimulaci.

4.1.2 Komunikace s převodníkem

Velmi důležitou částí aplikace bude oblast, která má na starost sériovou komunikaci (či obecně jakoukoli komunikaci) se zvoleným A/D převodníkem. Tato oblast má za úkol:

1) poskytovat rozhraní, pomocí kterého lze iniciovat komunikaci a zpětně se dozvědět o úspěšnosti navázání spojení,
2) rozhraní pro získání přijatých dat v tom pořadí, v jakém byla přijata,
3) rozhraní pro předání dat určených k odeslání A/D převodníku.

Uvedená rozhraní by měla být použitelná i při případné změně A/D převodníku za jiný. Tato část aplikace musí zajistit, aby bylo otevřeno právě jedno spojení, proto je vhodné zde využít návrhového vzoru Singleton. Pro počáteční nastavení je také třeba zjistit, na kterých sériových portech lze komunikovat a mít na výběr z dostupných rychlostí komunikace.

4.1.3 Stimulace a zpracování vstupů

Klíčovým úkolem aplikace je provádění stimulací. K zahájení stimulace musí docházet v co nejkratší době od přijetí dat a zjištění startu. Proto tyto úkony bude provádět

samostatné vlákno s maximální prioritou běhu. Při provádění stimulace se předpokládá, že vlákno má k dispozici uspořádaný seznam částí stimulační série dle jejich pořadí a navázané komunikační spojení, které mu umožní zpracovávat příchozí data nebo odesílat údaje o požadovaném výstupu do A/D převodníku.

Vlastní provádění stimulací je závislé primárně na délce zobrazované série, nikoliv na počtu příchozích synchronizačních impulzů. Tyto impulzy slouží jen jako podnět k provádění stanovených akcí. Tato vlastnost je důležitá zejména na konci měření, kdy po promítnutí posledního stimulačního obrazu není třeba očekávat další synchronizační impulz.

Důležité je, aby bylo možné provádět některé druhy stimulací zároveň, například promítnout obraz a k němu přehrát zvukový soubor. Tento zvukový soubor ovšem může hrát i nadále, během zobrazování dalších obrazů. Zároveň je nutné zajistit konec přehrávání zvuku v okamžiku startu jiné stopy, aby nedošlo k jejich mísení a tím i k jinému než zamýšlenému účinku zvolené stimulace.

Stimulační vlákno přijímá z příchozích dat pouze informace o čase příchodu synchronizačního impulzu. Ostatní informace nezpracovává, a tak se jimi zbytečně nezatěžuje. Tyto informace ale není možné zahodit, je třeba je předat dále. Zpracování údajů z digitálních ani analogových vstupů zpravidla není nutné provádět přímo při běhu stimulační série. Jedinou výjimku představuje měření, při kterém dochází k zobrazování hodnot analogových vstupů jako bod na obrazovce. U analogových vstupních zařízení může docházet k šumu, kdy po sobě jdoucí hodnoty jsou rozkmitané. Proto je třeba brát hodnotu jako průměr několika předchozích hodnot. Výstupní formát zpracovaných příchozích dat bude popsán v kapitole 4.5.

4.1.4 Komunikace s vyšetřovanou osobou

Jednou z autonomních částí aplikace bude možnost jednosměrně hlasově komunikovat s vyšetřovanou osobou. Tato komunikace bude zaznamenávána prostřednictvím mikrofonu a reprodukována standardním audio výstupem počítače. Jelikož musí docházet zároveň k zaznamenávání i reprodukování komunikace, bude vhodné použít více vláken. "Nahrávací" vlákno bude v pravidelných časových intervalech vkládat data do fronty "přehrávacího" vlákna. Opožděním přehrávače zajistíme, že přehrávání bude plynulé.

4.1.5 Kalibrace

V aplikaci bude možné nastavit její základní parametry. Bude se jednat o údaje pro sériovou komunikaci (viz kapitola 4.1.2), složky pro ukládání souborů (viz kapitola 4.6), nastavení monitorů a zejména o kalibraci projektoru využitého pro vlastní zobrazování stimulací. Nebude se jednat o kalibraci barev displeje, ale o kalibraci polohy plochy pro zobrazování stimulačních podnětů. Tuto kalibraci bude možné ovládat jak pomocí klávesnice, tak pomocí čtyř připojených tlačítek primárně určených pro zjišťování reakcí od pacientů.

Pro vyšetření, kde se využívá analogových signálů, bude třeba provádět takovou kalibraci, při které se zjistí střední hodnota těchto signálů. Na rozdíl od kalibrace polohy stimulační plochy se tato kalibrace bude provádět před každým měřením, například z důvodu možnosti výměny zdroje analogových signálů.

4.1.6 Program pro převodník

Zvolený A/D převodník může provádět kromě načítání dat i jejich základní zpracování. Mohlo by se například jednat o průměrování příchozích dat. Zde ale můžeme narazit na problém rychlosti procesoru převodníku, kdy by výpočty, které může provádět i počítač, mohly zpomalit odezvu desky a tím i snížit přesnost měření. Proto je vhodné, aby zvolený A/D převodník pracoval pouze takovým způsobem, že načte data na definovaných vstupních pinech, odešle je komunikačním kanálem, a pokud tímto kanálem přijal nějaká data, podle nich logické nastaví hodnoty na výstupních pinech.

4.2 Volba A/D převodníku

Jedním z důležitých bodů tvorby softwaru synchronizačního systému byl výběr zařízení, které bude využito jako A/D převodník. Primárně byla otestována možnost použití zařízení, která jsou v současnosti dostupná na pracovišti ZRIR IKEM. Jedná se o A/D převodníky Arduino Uno, Arduino Mega 2560 a modul AD14ETH. Funkce zmíněných zařízení byla podrobněji popsána v kapitole 3.3.

Aby bylo při používání nového softwaru dosaženo co nejlepších výsledků, je nutné minimalizovat zpoždění načítání synchronizačního impulzu ve všech hardwarových i softwarových částech řetězce. Na vstup A/D převodníku může být impulz přiveden buď optickým, nebo elektrickým signálem.

4.2.1 Arduino

Programy vytvořené v jazyce Java pro komunikaci s Arduinem využijí například knihovny RXTX nebo jSSC, viz kapitoly 3.5.2 a 3.5.3. Tyto knihovny poskytují API pro nepřetržitou obousměrnou sériovou komunikaci. Jelikož cílem testu bylo zjistit zejména zpoždění vzniklé během čtení a zpracování dat na Arduinu, aplikace v Javě se starala primárně o načtení dat na daném sériovém portu a zapsání času, kdy dorazil synchronizační impulz. Po skončení měření byla tato data vyhodnocena.

Na desku Arduina byl nahrán program, který měl obdobně jednoduchý princip – na konkrétním vstupním pinu registroval jeho hodnotu, kterou poté odesílal. Aby bylo testování prováděno v reálných podmínkách, byla odesílána ještě další data, ta však nenesla žádnou konkrétní informaci, sloužila pouze pro simulaci potřebné šířky pásma.

Arduino umožňuje odesílat data buď jako řetězec, nebo jako pole bytů. Při posílání numerických hodnot jako řetězec je nutné převést vše na string a zpět. Další komplikací je

vyšší režie v případě, že jsou posílány hodnoty delší než jeden znak, kdy je nutné čísla oddělovat například mezerou. Výhodou je přirozená signalizace konce bloku dat pomocí zakončení řádku.

Pokud dojde k přenosu dat pomocí pole bytů, může být lépe využito přenosové pásmo, neboť je možné posílat více binárních informací v jednom bytu. Naopak je nutné vyčlenit jednu hodnotu (nebo jeden bit v každém bytu) reprezentující ukončení bloku dat a při návrhu aplikace zajistit, aby daná hodnota byla odeslána právě v daný moment.

Při posílání dat z Arduina jako řetězce je nutné jej na straně přijímacího programu nejprve rozdělit na jednotlivé tokeny a poté převést na číslo. Datový objem přenosu jednoho bloku je dán jak počtem hodnot, tak jejich aktuální hodnotou. Zatímco odeslání hodnoty „0“ zabere 1 byte, odeslání hodnoty „1023“ z analogového vstupu obsadí 4 byty. Celý blok dat, který bude obsahovat informaci o synchronizačním impulzu, stisku některých ze čtyř tlačítek a dvě analogové hodnoty, tak zabere nejméně 14 bytů (pokud budou obě analogové hodnoty menší jak deset), nejvíce 20 bytů. Délku bloku je možné zmenšit umístěním hodnot synchronizačního impulzu a tlačítek za sebe bez mezery (neboť se jedná o binární hodnoty). Složitější (a mírně delší) bude také zpracování takových dat na straně přijímací aplikace.

Pokud jsou data posílána jako nekonečný proud bytů, není nutné je převádět zpět na čísla, pouze je třeba kontrolovat přijetí hodnoty signalizující konec bloku dat. Vyhradíme-li hodnotu 0 jako ukončení bloku dat. Pro zajištění její exkluzivitu v bloku, stačí všechny ostatní byty zvýšit o jedničku. Tím sice ztratíme jeden bit v každém bytu, ale máme zajištěn exkluzivní výskyt nuly. Na straně přijímače dat dokážeme zpracovat tato data za pomoci rychlých bitových operací (posunů a bitového AND a bitového OR).

Pokud vyhradíme jeden byte pro hodnotu synchronizačního impulzu (který je nejdůležitější), jeden byte pro hodnoty čtyř tlačítek (teoreticky lze v jednom bytu přenášet až sedm tlačítek), tři byty pro přenos dvou analogových hodnot (tedy 24 bitů pro přenos 20 bitů dat) a jeden ukončovací byte s hodnotou nula, přeneseme stejnou informaci v pouhých šesti bytech bez ohledu na velikost hodnot analogových vstupů. Význam jednotlivých bitů je tedy následující:

```
0000 0000
0000 00s1
tttt ttt1
xxxx xxx1
xxx0 yyy1
yyyy yyy1
```

s = synchronizační impulz (0/1), t = hodnota tlačítka (0/1), x = hodnota prvního analogového vstupu (0 – 1023), y = hodnota druhého analogového vstupu (0 – 1023)

Pokud budeme chtít ušetřit jeden byte, je možné poslat informaci o příchodu synchronizačního impulzu spolu s informací o konci bloku, tj. tento byte by nabýval decimální hodnoty 0 nebo 2 dle toho, zda dorazil synchronizační impulz. Zároveň by tento byte byl jediný, pro který by platila rovnost byte % 2 = 0. Pro praktické použití na pracovišti ZRIR IKEM budou využívána nejvýše čtyři tlačítka, tudíž by bylo možné přenášet hodnotu synchronizačního impulzu ve stejném bytu, jako hodnoty digitálních tlačítek. V tomto případě by tak pro přenos všech požadovaných dat postačovaly pouze 4 byty.

4.2.2 AD14ETH

Modul AD14ETH je ovládán prostřednictvím rozhraní, které poskytuje přiložený DLL soubor. Jeho funkcemi a metodami je nutné obsluhovat veškerou komunikaci a nelze využít standardní rozhraní pro Sockety v jazyce Java.

Modul byl nastaven podobnými parametry, jako Arduino, stejně tak přijímací program byl pro účely ověření řešení co nejjednodušší.

4.2.3 Ověření řešení

Popsané programy a A/D převodníky byly otestovány na pracovišti ZRIR IKEM. Test každého z nich trval přibližně 30 minut a za tu dobu bylo zareagováno na 750 synchronizačních impulzů, které přicházely přesně každé 2,5 sekundy.

4.2.3.1 AD14ETH

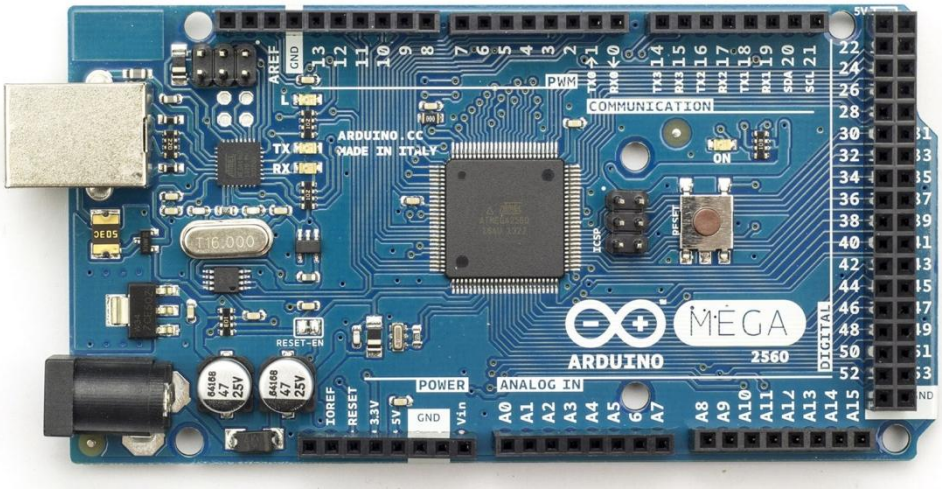
Při použití modulu AD14ETH jsme získali velmi přesné informace, včetně času, kdy byly naměřeny. Abychom data získávali co nejčastěji, byla nastavena nízká velikost TCP paketu. Pro synchronizovanou aplikaci dosahovalo i tak zpoždění při komunikaci hodnot v řádech stovek milisekund, což je pro účely stimulačního software nepoužitelné.

4.2.3.2 Arduino

Při použití desky Arduino Uno bylo měření výrazně nepřesné, délka mezi synchronizačními impulzy kolísala od 2,3 do 2,7 sekundy. Při použití desky Arduino Mega 2560 byly dosaženy lepší výsledky. Po přijetí 750 synchronizačních impulzů byla spočítána průměrná doba mezi dvěma impulzy 2500,008 ms, dva synchronizační impulzy dělilo maximálně 2512 ms a minimálně 2496 ms. Mezi každými dvěma synchronizačními impulzy pokaždé odešlo z Arduina do počítače 1049 nebo 1050 bloků dat.

Naměřené hodnoty lze považovat za dostatečně přesné a desku Arduino Mega 2560 (viz obr. 11) lze proto využít pro komunikaci se vznikající synchronizační aplikací. Přesnost lze ještě mírně zvýšit, pokud na stejném vstupním pinu, jako registrujeme synchronizační impulz, registrujeme přerušení vzestupnou hranou. Byl proveden také test, kdy bylo za zdroj synchronizačního impulzu považováno pouze přerušení, a nebyla načítána hodnota z digitálního vstupu na daném pinu. Primárním výstupem tohoto testu je skutečnost, že na

A/D převodníku ani v počítači nebyl zmeškán žádný synchronizační impulz, přestože jej tvořil právě jeden řádek odesílaných dat.



Obr. 11: Arduino Mega 2560.[51] Vlevo nahoře USB konektor, vlevo dole nápájecí konektor.

4.3 Využité knihovny a technologie

4.3.1 Sériová komunikace

V předcházející kapitole byl popsán způsob volby A/D převodníku. Veškerá Arduina se připojují k počítači prostřednictvím USB portu a komunikují přes virtuální sériový port linkou RS232.

Vlastní komunikaci přes (virtuální nebo skutečný) sériový port není třeba implementovat od začátku, neboť lze využít jednu z běžně používaných knihoven. V jazyce Java jsou k dispozici knihovny RXTX a jSSC. Podrobnější popis těchto knihoven je v kapitolách 3.5.2 a 3.5.3.

Komunikační rozhraní sériového portu RS232 je již několik desetiletí ustálené a neměnné. Přesto nelze říci, že z tohoto důvodu lze ignorovat skutečnost, že knihovna již není vyvíjena. Dalším faktorem ovlivňujícím výběr knihovny může být například čas, za jak dlouho po otevření komunikace dojde ke skutečnému zahájení přenosu dat.

Tento faktor jasně zvýhodňuje knihovnu jSSC, neboť při provedených zkouškách spojení byla sériová komunikace ustavena ihned, zatímco při kombinaci knihovny RXTX a desky Arduino Mega bylo nutné přibližně 10 sekund čekat, než začala proudit data. Tato

[51] Viz 37

prodleva by mohla zbytečně znejistět uživatele aplikace. Knihovna jSSC je navíc, jak již bylo zmíněno, stále vyvíjena.

4.3.2 Přehrávání audio a video souborů

Pro vlastní provádění audiovizuálních stimulací byl vyhodnocen jako nejlepší JMF (viz kapitola 3.5.4). Primárním důvodem výběru této knihovny je skutečnost, že při jejím používání bylo dosaženo nejmenších zpoždění při přehrávání, a to jak zvukových stop, tak videa. Jeho nevýhodou je morální zastaralost a omezené množství přehrávaných formátů. Tato omezení nejsou zanedbatelná, ale pro stimulační systém je daleko důležitější zmíněné minimální zpoždění startu.

Pro přehrávání zvuků jsou využívány třídy z balíku *javax.sound.sampled*. Nejprve dojde k načtení celého souboru do paměti, poté je z načteného pole bytů vytvořen *ByteArrayInputStream*, který slouží jako vstupní data pro instanci třídy *AudioInputStream*. Při tomto postupu bylo dosaženo zpoždění startu zvuku okolo 20 až 30 milisekund, při použití jiných knihoven byly hodnoty zpoždění v řádech stovek milisekund. Je nutné zmínit, že takto lze přehrávat pouze WAV soubory.

Rozhraní *javax.sound.sampled* lze využívat i pro nahrávání signálu z mikrofonu. Nabízí se tedy využití JMF i pro jednosměrnou komunikaci s pacientem (viz kapitola 3.1). Hlavním úkolem je zde zajistit, aby přehrávání nahraného zvuku probíhalo bez přerušení.

Při přehrávání videa dochází k využití kodeků z balíku *jFFMPEG* (viz 3.5.8), které rozšiřují možnosti standardního JMF. Díky této knihovně lze přehrávat AVI soubory, které využívají video kodek XVID a zvukový kodek PCM. Rozhraní Java Media Frameworku pro přehrávání videí obsahuje metodu *prefetch*, která po svém zavolání načítá a dekóduje první sekundy videa. Tím byla snížena prodleva na startu videa na zanedbatelnou hodnotu. Při přehrávání videí dochází ke zpoždění startu v rozmezí pouze 50 až 100 mikrosekund, to je při rychlosti videa 24 snímků za sekundu zpoždění o dva nebo tři snímky. Zmíněná zpoždění jsou statisticky nevýznamná vůči měřeným odezvám, zejména BOLD efekt dosahuje svého maxima po 6 až 9 sekundách, viz kapitola 2.1. Při používání jiných knihoven nebo při nevyužití metody *prefetch* bylo naměřeno zpoždění v řádech několika stovek milisekund až půl sekundy. Jiné, modernější, knihovny nenabízejí obdobnou metodu, ani žádnou jinou možnost načíst a dekódovat data v předstihu.

4.3.3 Ověření návrhu

Byla ověřena i funkčnost navrhovaných řešení v praxi. Sériová komunikace byla otestována v rámci výběru A/D převodníku (viz kapitola 4.2.3.2). Jak pro zvuky, tak pro videa byl proveden stejný test. Jeden soubor byl rozdělen za pomoci skriptu na několik desítek stejně dlouhých částí. Délka každé části odpovídala nastavenému intervalu mezi synchronizačními impulzy z MR tomografu a každý příchozí impulz spouštěl další část hudební skladby nebo videa.

Během přehrávání zvukové skladby nebyly při běžném poslechu patrné žádné nežádoucí efekty, několik na sobě nezávislých osob se dokonce původně domnívalo, že přehrávaná skladba není rozdělena na části po dvou sekundách. Při přehrávání videa již byly místy zaznamenány drobné náznaky nesouladu obrazu na hraně dvou částí, přesto byly pro osoby, které nevěděly, ve kterých místech došlo k rozstříhání záznamu, pouhým okem těžko odhalitelné.

Další zkoušky poskytly porovnání v současnosti používaného softwaru a vyvíjené aplikace. Jejich cílem bylo zjistit, která aplikace spouští videa s menší prodlevou. Po zhlédnutí několika desítek videí došli pozorovatelé k závěru, že vyvíjená aplikace dosahuje přinejmenším stejných, spíše však lepších výsledků, než v současnosti používaný software.

4.4 Ukládání dat

Zadavatel požaduje ukládání měřených stimulačních sérií do souborů, které bude možné upravovat v běžných textových editorech nebo generovat externími skripty. Ze zkušeností se stávajícími programy vyplývá, že je vhodné ukládat parametry nastavení také do souborů, neboť ty lze snadno duplikovat a využít pro jiné měření. Na rozdíl od stimulační série není potřeba upravovat tyto soubory ručně. Komplikací ovšem může být nutnost uložit výsledný soubor v očekávaném kódování, tj. v UTF-8.

Aby bylo možné upravovat data stimulační série mimo program, je třeba stanovit formát textového souboru. Víme, že u každého prvku série může být definováno jeho zpoždění vůči synchronizačnímu impulzu, dále musí být stanoven typ stimulace a u některých typů také potřebný soubor. Zároveň může nastat i situace, kdy v daný čas nenastává žádná stimulace řízená programem, pouze dochází k odeslání dat do A/D převodníku.

U některých typů měření je požadováno, aby byl příchozí analogový signál souřadnic joysticku v danou chvíli upraven o stanovený úhel a násobek vzdálenosti bodu od středu v polárním systému souřadnic. Tato změna příchozího signálu může být buď skoková v daný moment, nebo postupně nabíhat po dobu trvání následujícího TR. V souboru se stimulační sérií proto musí být možnost nastavovat i tyto ostatní údaje, přestože se nevyužijí při všech běžně používaných měřeních.

Aby byl formát souborů jednotný, byl stanoven následující formát. Každá položka stimulační série je právě na jednom řádku souboru. V každém řádku jsou jednotlivé položky odděleny tabulátorem (znakem \t). Položky v hranatých závorkách není nutné uvádět. Celý řádek vypadá takto:

```
Zpoždění (tabulátor) typ_stimulace [(tabulátor) odesílaná_data [(tabulátor) deviace_signálu]]
```

Zpoždění obsahuje čas, kdy má dojít k provedení stimulace (nebo obecně akci) definované na aktuálně čteném řádku.

Typ stimulace obsahuje druh zvolené stimulace nebo zdrojový soubor stimulace. Pokud je ke stimulaci využit soubor z počítače, lze z jeho typu snadno určit druh stimulace.

Následující dva parametry, tedy *odesílaná data* a *deviace signálu* jsou volitelné. Pokud nejsou zadány, předpokládá se, že se neodesílají žádná výstupní data a analogový signál se nemění. Je možné zadat oba parametry, žádný parametr, nebo pouze odesílaná data. Pokud je třeba definovat změnu analogového signálu, je třeba určit i odesílaná data.

Parametr *odesílaná data* obsahuje hodnoty dat, které se odesílají na výstupní piny Arduina. Tyto hodnoty lze zadat buď v decimální, nebo v binární soustavě. Číselná soustava je určena prvním znakem – buď písmenem *d* pro desítkovou soustavu, nebo *b* pro binární soustavu.

Poslední parametr reprezentuje *deviaci analogového signálu*. Tato změna je zadávána v polárních souřadnicích, tj. násobek změny délky a hodnota, o kterou se změní úhel. Zadává se jako tři mezerou oddělená čísla, např. 2.5 30 1. První číslo může být desetinné a představuje násobek změny vzdálenosti. Druhé číslo je změna úhlu a poslední číslice reprezentuje informaci, zda změna signálu má nastat okamžitě nebo lineárně narůstat během doby následujícího TR, aby v okamžiku příchodu synchronizačního impulzu dosáhla zadaných hodnot. Ukázka možných řádků v souboru se stimulační sérií:

```
0     img9.jpg
```
v čase příchodu synchronizačního pulzu je promítnut obrázek, odesílaná data jsou nulová (8x 0), signál se nemění

```
1820 img10.jpg       d128       1.0 0 0
```
1820 ms po příchodu synchronizačního pulzu je promítnut obrázek, sériovou komunikací jsou odeslána data 1 0 0 0 0 0 0 0, signál se nemění

```
0     img11.jpg      b10000000 1.0 0 0
```
v čase příchodu synchronizačního pulzu je promítnut obrázek, sériovou komunikací jsou odeslána data 1 0 0 0 0 0 0 0, signál se nemění

```
0     nothing        d0
```
v čase příchodu synchronizačního pulzu nedojde k žádné akci, odeslány jsou samé nuly, signál se nemění

```
0     grid1.grid     d0         1.0 0 1
```
v čase příchodu synchronizačního pulzu je zobrazen model (mříže) ze souboru grid1.grid, odeslány jsou samé nuly, signál se nemění

```
0     nothing        d0         0.25 20 1
```
v čase příchodu synchronizačního pulzu nedojde k žádné akci, odeslány jsou samé nuly, příchozí analogový signál je v polárních souřadnicích zkrácen na čtvrtinu, k úhlu je přičteno 20 stupňů. Změna roste lineárně, až do doby příchodu příštího synchronizačního impulzu. V ten okamžik dosáhne definovaných hodnot.

4.5 Vstupní data

V současné době lze pomocí programu SySy zaznamenávat pouze vstupní data ze čtyř tlačítek. Tyto údaje jsou ukládány do souborů ve formátu, kdy na jednom řádku souboru je buď znak #, který vyjadřuje okamžik, kdy přišel synchronizační impulz, nebo číslice. Tato číslice vyjadřuje hodnotu napětí na analogovém vstupu karty AD14PCI**.** Umisťování dat do tohoto formátu je sice snadné, o to náročnější je ovšem jejich analýza.

Pro analýzu vstupních dat z různých typů měření lze jen obtížně definovat jednotný výstupní formát, aby nebylo nutné další zpracování. Proto byl stanoven takový výstupní formát, aby bylo další zpracování naměřených údajů pokud možno co nejsnazší. Vyhodnocování je nejčastěji prováděno skripty pro program Matlab. Zároveň je třeba zachovat kontinuitu s již proběhlými měřeními pokračujících studií.

U dat z digitálních vstupů (tedy tlačítek) je nejčastěji využívaný formát, který je tvořený časem stisku tlačítka (*on set*) a jeho délka (*duration*). Hodnota *on set* je vztahována k času prvního synchronizačního impulzu, který spustil stimulační sérii. Textový soubor, který obsahuje naměřené údaje, bude potom tvořit jeden řádek hodnot *on set* navzájem oddělených středníkem, druhý řádek souboru by obsahoval stejně oddělené hodnoty *duration* s tím, že první hodnota *on set* tvoří pár s první hodnotou *duration* atd. Pokud je sledováno více tlačítek, následuje volný řádek a hodnoty z dalšího tlačítka ve stejném formátu. Na následující ukázce je vidět, jak vypadá výstupní soubor, pokud jsou registrována dvě tlačítka. První tlačítko bylo stlačeno 11,238 sekundy po startu stimulace po dobu 3,748 sekundy a 26,233 sekundy po startu na 3,757 sekundy. Druhé tlačítko bylo stlačeno třikrát: 159, 14994 a 29990 tisícin sekundy po startu měření, pokaždé přibližně na 3,75 sekundy.

```
11238;26233
3748;3757

159;14994;29990
3755;3743;3740
```

Údaje z analogových vstupů (zejména joysticku) není možné snadno převést do podobného formátu, jako digitální vstupy, aby nedošlo ke ztrátě informační hodnoty. Proto byl zvolen výstupní formát, kdy jeden řádek souboru obsahuje čas od prvního synchronizačního impulzu (stejná hodnota, jako údaj on set v předchozím odstavci), binární informaci o tom, zda přišel nebo nepřišel synchronizační impulz a poté hodnoty všech načítaných analogových vstupů. Jednotlivé hodnoty v řádku jsou odděleny tabulátorem (znakem \t).

V případě, že dochází k ovlivnění hodnot příchozího analogového signálu (viz kapitola 2.6), panuje přesvědčení, že je lepší ukládat do souboru s výstupními daty již ovlivněné hodnoty souřadnic. Jednoduchým algoritmem je pak snadné dopočítat skutečné hodnoty, pokud známe prodlevu od synchronizačního impulzu a víme, že ovlivnění nastává buď

skokově, nebo lineárně. Následující ukázka přibližuje formát souboru, do kterého jsou zaznamenávány příchozí hodnoty analogových signálů. První sloupec tvoří čas od začátku stimulace, druhý informace o synchronizačním impulzu, třetí a čtvrtý sloupec tvoří hodnoty naměřené v daný čas.

```
88   1    141  327
104  1    150  349
309  1    219  342
698  0    282  405
713  0    283  406
```

4.6 Složky souborů

V současné době jsou soubory, které se využívají pro stimulace při měření fMRI, umisťovány na základě jejich typu v příslušných složkách. Konkrétně se jedná o adresáře *obrazky* a *zvuky*, které musí být v kořenovém adresáři aplikace[52].

Toto rozdělení odpovídá potřebě uživatelů programu, proto je vhodné jej zachovat. Přibude ještě složka, do které se budou ukládat videa. Není nutné nadále omezovat umístění nebo název zdrojových složek programu. Cesta k těmto složkám může být libovolně změnitelná, například dle jazykové mutace souborů. Zároveň bude doplněna možnost určit zdrojovou složku souborů jednotlivě pro každé měření. Pokud tato složka se specifickými daty nebude zadána, použijí se obecné složky.

Podobný stav je i u souborů, které obsahují data nastavení jednotlivých měření a stimulační série pro jejich běh. Údaje o nastaveních jsou umisťovány do adresáře *Nastaveni*, samotné série pak do složky *Serie*.[53] Zde by bylo možné uvažovat o sloučení adresářů do jednoho.

Pokud je prováděna registrace externích signálů z tlačítek, jsou tato data v současném systému SySy uložena do obyčejného textového souboru s názvem *reg_yyyymmdd_hhmm.txt* do kořenové složky programu.[54] Zde ovšem může dojít k přehlédnutí nebo záměně souboru s jiným, proto je vhodné tyto údaje ukládat do zvláštní složky, nejlépe ještě dále data roztřídit dle dne provedení měření. Výhodné je také mít možnost doplnit do názvu souborů jméno vyšetřovaného pacienta v aplikaci. Současný i navrhovaný formát těchto souborů byl popsán v předchozí kapitole 4.5.

[52] Viz 33
[53] Viz 33
[54] Viz 33

5 Implementace

5.1 Arduino

Deska Arduino obsahuje svůj vlastní mikroprocesor, jehož činnost lze ovlivnit dle konkrétní potřeby vytvořením a nahráním specifického programu. Tyto programy jsou tvořeny nejméně dvěma metodami: *setup*() a *loop*(). Zatímco metoda *setup*() je zavolána jen jednou při startu a používá se pro inicializaci proměnných, nastavení vstupních a výstupních pinů nebo načítání knihoven, metoda *loop*() je volána poté stále dokola ve smyčce a slouží k vlastnímu běhu programu a umožňuje komunikaci s jinými aplikacemi.

Pokud je Arduino využito pouze jako A/D převodník, obsahuje metoda *setup*() pouze nastavení rychlosti sériové komunikace a nastavení, které piny jsou vstupní a které výstupní. Jelikož je pro zrychlení načítání synchronizačního impulzu využíváno přerušení na vstupním pinu, obsahuje metoda *setup*() také jeho registraci. Pro inicializaci přerušení je použito volání metody *attachInterrupt* (viz http://arduino.cc/en/Reference/attachInterrupt). Konkrétně je registrováno přerušení č. 0, které je řízeno vstupním napětím na pinu č. 2. Dále je definováno, že přerušení je vyvoláno vzestupnou hranou napětí, která nastává právě v okamžiku příchodu synchronizačního impulzu.

Pro obsluhu přerušení slouží metoda *interrupt*(). Tato metoda pouze nastaví proměnnou synchro na číselnou hodnotu 3 a odešle data. Hodnota 3 vznikla načtením digitálního vstupu na příslušném pinu (který má logickou hodnotu 1), jeho bitovým posunutím o jedno místo doleva a provedením bitového OR s číslem jedna.

Metoda *loop*() volaná v nekonečné smyčce, obsluhuje načítání dat ze vstupů a jejich zpracování před odesláním sériovou komunikací. Nejprve jsou načtena data z analogových vstupů, poté z tlačítek a nakonec je načtena hodnota synchronizačního impulzu. Zpracování dat zahrnuje především jejich bitový posun o jedno místo doleva a aplikaci bitového OR s hodnotou 1. U 10 bitových hodnot analogových vstupů dochází za pomoci bitových posunů k rozdělení dvou desetibitových hodnot do tří bytů. Poté jsou data odeslána. Podrobnější popis odesílaných dat je v kapitole 4.2.1. K přenosu dat je vždy využíváno šesti bytů, i když některá nastavení vyžadují pouze synchronizační impulz.

Po odeslání dat do počítače dochází ke kontrole, zda jsou na sériové lince dostupná data k přečtení. Na základě načtených dat jsou poté nastaveny hodnoty na příslušných výstupních pinech.

Program pro Arduino může obsluhovat až sedm digitálních vstupů, je ovšem nutné rozhodnout, zda při nižším počtu tlačítek budou informace ukládány do prvních nebo posledních několika bitů. Bylo rozhodnuto, že je-li načítáno méně tlačítek než 7, budou

informace uloženy v bitech, které se nacházejí nejvíce vpravo. Proto při registraci hodnot čtyř tlačítek bude přenášen druhý byte ve formátu

```
tttt0001
```

kde t je hodnota jednoho tlačítka.

Aby bylo možné snadno měnit nebo kontrolovat nastavení jednotlivých vstupních a výstupních pinů, jsou jejich čísla definována v globálních proměnných uvedených na začátku programu. Podobně je definován počet načítaných digitálních vstupů (tlačítek). Na pracovišti ZRIR IKEM bude využíváno následující nastavení pinů zachycené v tabulce 4.

Název/typ hodnoty	Číslo pinu	Poznámka
Synchronizační impulz	2	Též přerušení č. 0
Digitální vstup č. 1	6	Tlačítko 1
Digitální vstup č. 2	5	Tlačítko 2
Digitální vstup č. 3	4	Tlačítko 3
Digitální vstup č. 4	3	Tlačítko 4
Analogový vstup č. 1	A6	X souřadnice joysticku
Analogový vstup č. 2	A14	Y souřadnice joysticku
Digitální výstup č. 1	22	-
Digitální výstup č. 2	24	-
Digitální výstup č. 3	26	-
Digitální výstup č. 4	28	-
Digitální výstup č. 5	30	-
Digitální výstup č. 6	32	-
Digitální výstup č. 7	34	-
Digitální výstup č. 8	36	-

Tab. 4: Přiřazení jednotlivých vstupních a výstupních pinů.

5.2 Balík Globals

Balík *Globals* obsahuje tři třídy a jeden interface. Jedná se o třídy používané napříč celým programem, které se nehodilo umístit do jiných balíků. Ve třídách *Globals* a *GlobalConfiguration* jsou uchovávány globální proměnné. Třída *ApplikacionException* slouží k propagaci specifických výjimek vzniklých v programu. Interface *SettingProducer* je využíván pro poskytnutí načteného nastavení stimulace třídě *Globals*.

5.2.1 Třída GlobalConfiguration

Rozdíl tříd *Globals* a *GlobalConfiguration* je především v trvalosti uchovávaných dat. Zatímco třída *Globals* slouží pro uchovávání proměnných za běhu aplikace a při startu se nastavují dané inicializační hodnoty, třída *GlobalConfiguration* slouží pro uchování základního nastavení programu pro všechna spuštění. Při startu programu je načtena poslední uložená konfigurace. V této třídě se také nacházejí základní neměnné konstanty, které jsou

závislé na připojeném hardwaru. Jedná se zejména o omezení počtu digitálních a analogových vstupů a digitálních výstupů. Mimo to je ve třídě *GlobalConfiguration* uchováváno vše podstatné: jméno sériového portu pro komunikaci s Arduinem, rychlost sériové komunikace, čísla obrazovek pro zobrazování stimulace a ovládacího okna, nastavení používaných složek (viz kapitola 4.6) nebo také způsob a nastavení pozice zobrazení obrazových stimulací.

5.2.2 Třída Globals a rozhraní SettingProducer

Třída *Globals* drží otevřené nastavení, načtenou konfiguraci programu a informaci o běhu stimulace. Dále také obsahuje metody a funkce pro nastavování nebo získávání uchovávaných informací. Poskytování nastavení třídě *Globals* není možné přímo, ale pomocí interface *SettingProducer*. Toto rozhraní implementuje metodu, pomocí které si třída *Globals* převezme nově otevřené nebo vytvořené nastavení. Po předání neprázdného nastavení dojde ke kontrole fyzické přítomnosti částí stimulační série na disku počítače. Všechny proměnné, funkce a metody ve třídě *Globals* jsou definovány jako statické.

5.3 Balík Serial

Balík tříd *Serial* obsahuje třídy využívané pro sériovou komunikaci. Tento balík také zapouzdřuje použití knihovny implementující vlastní komunikaci, aby bylo možné v případě potřeby tuto knihovnu snadno zaměnit bez nutnosti měnit mnoho tříd napříč celou aplikací.

5.3.1 Třídy Arduino, InputDataIterator, rozhraní InitializedListener

Hlavní třída balíku, která zajišťuje sériovou komunikaci s Arduinem a zároveň poskytuje dovnitř programu rozhraní pro komunikaci, je nazvána *Arduino*. Tato třída je implementována dle návrhového vzoru Singleton. Obsahuje metody a funkce pro zahájení nebo ukončení komunikace, odeslání dat nebo přidání (a odebrání) pozorovatelů, čekajících na vlastní zahájení přenosu dat. Tito pozorovatelé musí implementovat rozhraní *InitializedListener*, které je také součástí balíku tříd serial. Pro vlastní čtení a odeslání dat jsou vytvořeny instance privátních tříd *ArduinoReader* a *ArduinoWriter*. Instance těchto tříd poté běží ve vlastních vláknech.

Jiným třídám aplikace jsou příchozí data poskytována pomocí rozhraní *Iterator*, které lze získat zavoláním funkce *getInputDataIterator*(). Tato funkce vrátí instanci třídy *InputDataIterator* (z balíku serial), která má za úkol zapouzdřit přístup k datům, aby nebylo nutné v aplikaci znát třídu *ArduinoReader* sloužící pro sériovou komunikaci. Vlastní příchozí data jsou ukládána do FIFO fronty.

5.3.2 Třídy InputData a OutputData

Pro práci s přijatými nebo odesílanými daty slouží třídy *InputData* a *OutputData*. Třída *OutputData* slouží k pouhému ukládání binárních hodnot (jako booleanů) do pole, které je pro odeslání převáděno do číselné hodnoty jednoho bytu.

Třída *InputData* slouží pro držení právě jedné sekvence příchozích bytů. Tato sekvence má konstantní délku, proto je ukládána do pole. Pomocí přetížené metody put jsou do instance třídy postupně vkládány další byty, přičemž je kontrolován počet již vložených hodnot. Pokud byl do instance vložen jiný počet hodnot, než byl očekáván (viz popis dat odesílaných z Arduina v kapitole 4.2.1), bude instance zahozena.

Druhým úkolem této třídy je poskytovat rozhraní k příchozím datům. K tomu slouží funkce *isSynchro, getButtons, isButton, getXCoordinate, getYCoordinate* a *getAnalogInput*. Pro zrekonstruování vlastních příchozích dat je nutné použít bitové posuny, podobně jako v programu pro Arduino, a znát skutečnost, že poslední bit má vždy hodnotu 1.

Při vlastním zpracování dat je také třeba znát čas jejich přijetí. Proto je při zavolání konstruktoru uložen aktuální systémový čas.

5.3.3 Třída SerialUtils

Třída *SerialUtils* obsahuje statické funkce, které poskytují informace související se sériovou komunikací. Jedná se o poskytnutí seznamu dostupných sériových portů a možných rychlostí sériové komunikace. Tato třída také poskytuje instanci třídy *SerialPort* (součást knihovny *jSSC*), zajišťující spojení na zvoleném portu.

5.4 Balík Setting

Balík *Setting* obsahuje třídy, jejichž instance drží informace o vlastním nastavení stimulace a prováděné stimulační sérii. Jedná se o třídy *Setting*, *StimulationPart* a *AnalogInputChange*, ke kterým ještě patří výčet *StimulationPartType*. Instance třídy *AnalogInputChange* nese informaci o velikosti změny analogového signálu. Tato změna je zadávána v polárních souřadnicích, tedy jako násobek vzdálenosti bodu od počátku a velikost změny úhlu. Vztahy tříd v balíku zachycuje obr. 12.

5.4.1 Třída Setting

Třída *Setting* tvoří střed celé aplikace, neboť obsahuje údaje potřebné pro provádění konkrétní stimulace. Jedná se například o určení zdrojové složky souborů nebo stanovení počtu sledovaných digitálních a analogových vstupů. Třída implementuje dvě rozhraní: *Serializable* pro umožnění serializovaného ukládání na disk a *Comparable* pro seřazení měření dle názvu. Setry třídních proměnných kontrolují, zda skutečně dochází ke změně hodnot a zda se má aplikace při svém konci nebo při otevření jiného nastavení dotazovat na uložení.

Mezi neserializovaná pole třídy patří také seznam jednotlivých částí stimulační série. Části jsou na základě požadavku zadavatele ukládány v textovém formátu, který je možné upravit nebo nechat vygenerovat libovolným skriptem (viz kapitola 4.4).

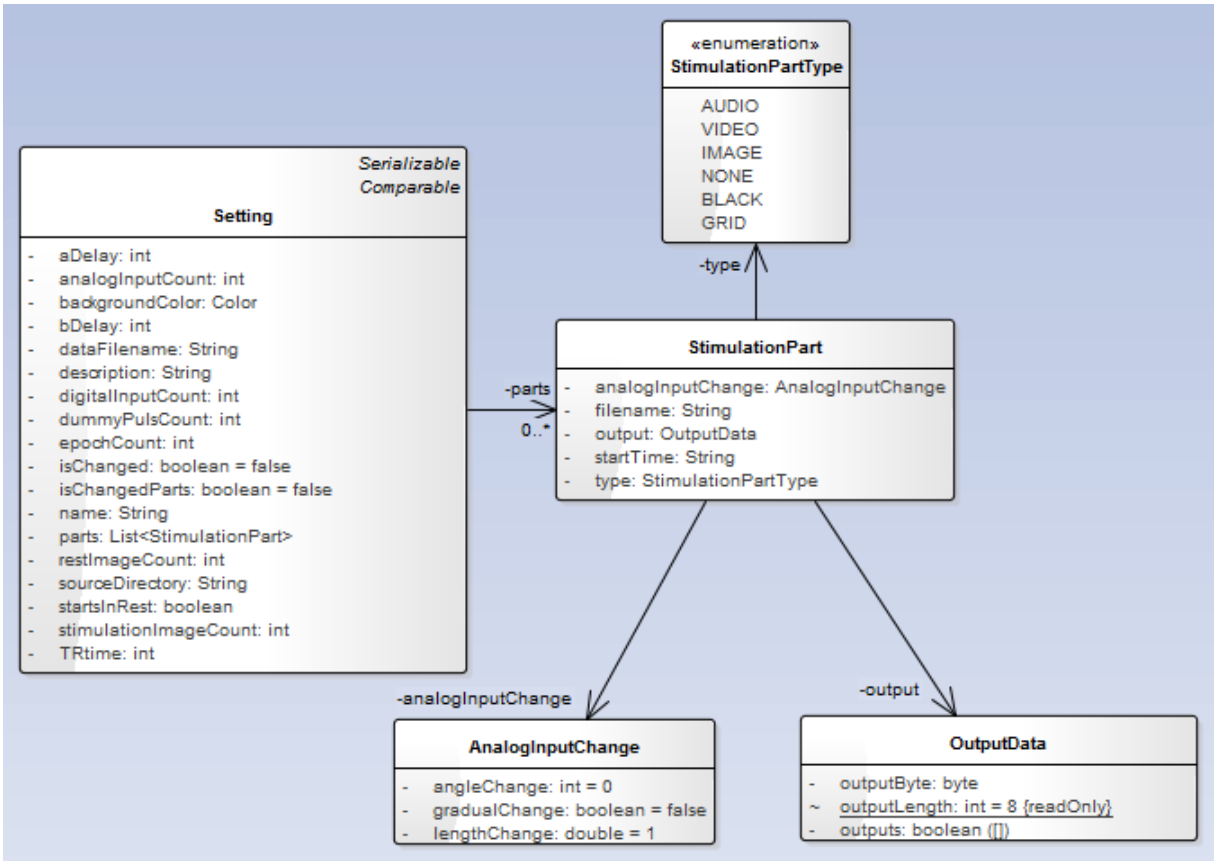
5.4.2 Třída StimulationPart, výčet StimulationPartType

Každou součást stimulační série reprezentuje instance třídy *StimulationPart*. Obsahuje nejen informace o souboru využitém pro vlastní stimulaci, ale také data odesílaná A/D převodníku a definici úpravy příchozích analogových signálů (viz kapitola 2.6). Pro převedení instance třídy do textového formátu je využito přepsání funkce *toString*. Formát textových souborů byl popsán v kapitole 4.4.

Vlastní typ stimulace je určen výčtovým typem *StimulationPartType*. Jedná se o následující druhy stimulace:

- AUDIO – stimulace přehráním zvukového souboru
- VIDEO – stimulace přehráním videa
- IMAGE – stimulace zobrazením obrazu
- NONE – žádná změna oproti předchozí stimulaci
- BLACK – zobrazení jednobarevného pozadí
- GRID – zobrazení mříže a kurzoru joysticku podle hodnot analogového vstupu

Počet stimulací může být v budoucnu rozšířen v závislosti na potřebách pracovišť, kde se bude vyvíjená aplikace využívat. V současné době se uvažuje o zobrazování množiny bodů evokujících lidskou postavu v pohybu.



Obr. 12: Balík Setting.

5.4.3 Balík Setting.model

Třída *GridModel* v balíku *setting.model* slouží pro ukládání informací o mříži používané pro stimulační sérii. Mříž je tak z hlediska tvorby skladby stimulační série obyčejný soubor, stejně jako obrázek nebo video. Modely mříží jsou serializovatelné, aby bylo možné je ukládat na disk.

V budoucnu se předpokládá doplnění dalších modelů (například bludiště), třídy reprezentující jejich vlastnosti budou umístěny také v tomto balíku.

5.5 Balík Utils

Třídy, které jsou obsaženy v balíku *Utils*, obsahují nesourodou sadu funkcí, které jsou využívány na různých místech programu. Velká většina funkcí těchto tříd je statická.

Třída *AudioUtils* obsahuje přejatou funkci pro zjištění délky WAV souboru a funkci poskytující formát zvuku pro nahrávání a následnou reprodukci, který využívají třídy z balíku *communication* (viz kapitola 5.6). Obdobně třída *VideoUtils* obsahuje funkci pro zjištění délky videa.

Třída *ByteUtils* nabízí funkce pro převody čísel na pole booleanů nebo naopak. Při převodu na pole pravdivostních hodnot je nutné kromě čísla zadat i požadovanou délku výstupního pole. Dále v této třídě nalézáme funkci, která převádí proměnnou typu byte do kladných hodnot.

Třída *DisplayUtils* spravuje informace o dostupných zobrazovacích zařízeních. Poskytuje jejich počet, informace o jejich rozlišeních a také souřadnice počátku každého připojeného zařízení. Dále obsahuje statickou funkci, která po startu programu zkontroluje dostupnost nastavených zobrazovacích zařízení.

Třída *FileUtils* obsahuje funkce, jejichž náplň činnosti souvisí se soubory a souborovým systémem. Jedná se o implementaci zápisu a čtení jak do binárních, tak do textových souborů, tvorby složek, přejmenovávání, kopírování nebo mazání souborů nebo například zjištění absolutní cesty k souboru.

Pomocí třídy *IconProducer* lze získat obrazové ikony využívané pro tlačítka v grafickém rozhraní programu. Soubory s ikonami jsou zkompilovány spolu se zdrojovými kódy programu v balíku *utils.icons*.

Třída *StringUtils* obsahuje některé funkce pro práci s řetězci, například převod seznamu na text. Další funkce slouží pro grafické rozhraní, jemuž poskytují název otevřeného nastavení buď v plné, nebo zkrácené délce.

Třídy v balíku *Utils.transform* slouží k převodu souřadnic z kartézského systému do polárního a naopak.

5.5.1 Třída StringProducer a přeložitelnost aplikace

Třída *StringProducer* zapouzdřuje poskytování textových řetězců, aby mohl být v případě potřeby změněn jejich jazyk. K tomu byly využity standardní prostředky: objekty typu *Locale* a *ResourceBundle*. Díky těmto dvěma instancím dochází k načítání textových popisků ze souborů messages_(jazyk)_(země).properties. Tyto soubory jsou zkompilované spolu s programem, díky čemuž není třeba kontrolovat jejich přítomnost. Nastavený jazyk je kontrolován pouze po startu aplikace, proto je po jeho změně nutné restartovat program. Jako výchozí jazyk je použita čeština. Třída *FileChooserTranlator* slouží k lokalizaci dialogu pro výběr nebo uložení souboru.

5.5.2 Balík Utils.filters

Třídy v balíku *Utils.filters* obsahují třídy rozšiřující základní javovské třídy využívané pro filtrování. Třída *ExtensionFilter* je využívána pro filtrování souborů zobrazených v dialogu výběru souboru dle typu. Třída *FiletypeFilter* slouží také k filtrování souborů, ovšem je předávána jako parametr funkci *listFiles* třídy *File*.

5.6 Balík Communication

Třídy v balíku *communication* obstarávají jednosměrnou komunikaci s měřenou osobou, popsanou v kapitolách 4.1.4 a 3.1. Skládá se ze tříd *Recorder, Player* a *PlayerPreparer*. Instance těchto tříd běží ve svých vlastních vláknech. Třída *Recorder* má za úkol nahrávat zvukový záznam, ukládat jej do paměti a každých 250 milisekund poskytnout surová data instanci *PlayerPreparer*. Toto se děje pomocí předávání instancí třídy *ByteArrayOutputStream*.

PlayerPreparer poté z poskytnutých dat vytvoří *AudioInputStream* a vloží je do své fronty, kterou poskytuje instanci třídy *Player*. Z hlediska *Playeru* nabízí pouze implementaci rozhraní *Iterator*. Přehrávač záznamu již pouze získá připravená zvuková data, která předá zvukovým výstupním zařízením. Popsaná komunikace je spuštěna kliknutím myši na tlačítko v grafickém rozhraní a probíhá pouze po dobu stisku myši.

5.7 Grafické rozhraní

Pro grafické rozhraní aplikace byl využit Swing, který je běžnou součástí Javy. V balíku *gui* se nacházejí tři hlavní okna: hlavní, stimulační a kalibrační. Třída *GUI* poskytuje statické funkce využité napříč celým grafickým rozhraním, například tvorbu tlačítek nápovědy nebo zajišťuje vykreslování oken na správný monitor.

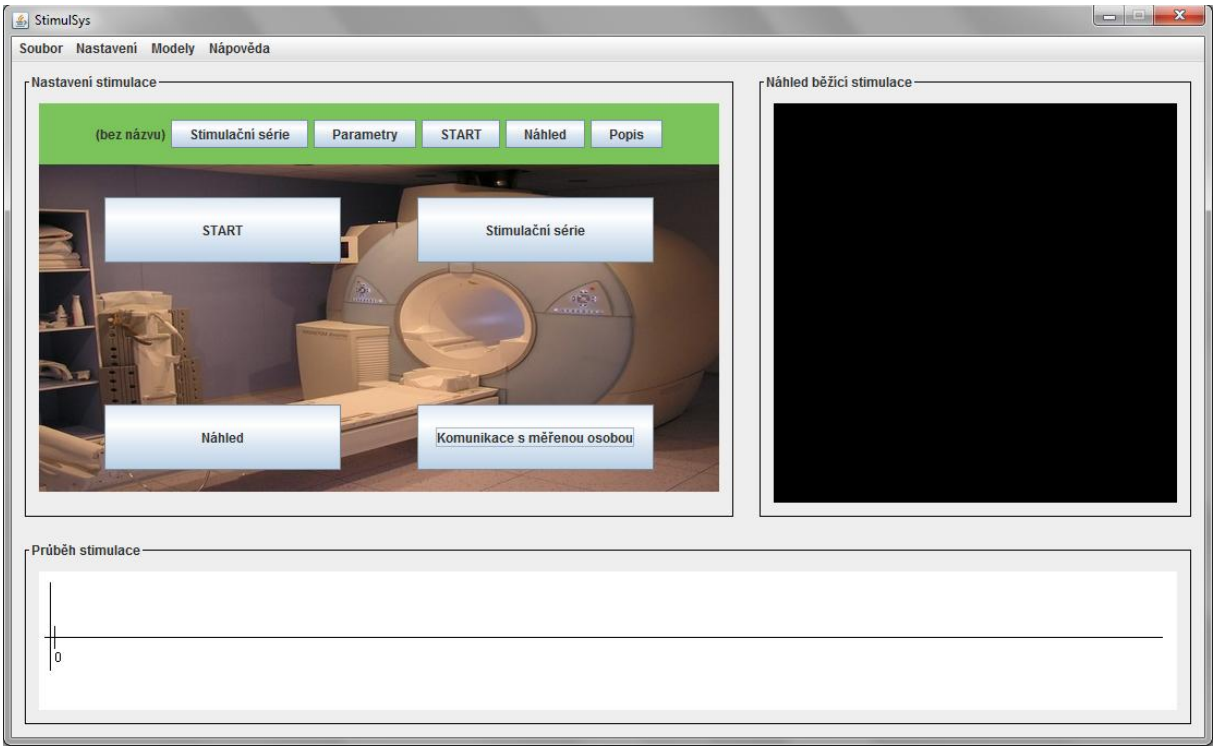
5.7.1 Okna aplikace

Kalibrační okno je využito pro určení pozice středu nebo levého horního rohu zobrazovaných stimulačních podnětů. Obsahuje modrý obdélník, který je možné posunovat šipkami po obrazovce. Aby bylo možné provádět kalibraci přímo z MR tomografu, lze ovlivnit pozici obdélníku i pomocí tlačítek připojených do Arduina.

5.7.1.1 Třída MainWindow

Hlavní okno slouží k zobrazení jednotlivých panelů využívaných pro ovládání aplikace. Toto okno má pevně daný rozměr. Skládá se ze tří základních bloků: Nastavovacího panelu, náhledového panelu a spodního panelu (viz obr. 13). Nastavovací panel slouží k vlastnímu ovládání aplikace. Náhledový panel obstarává zobrazení náhledu probíhající stimulace na zobrazovacím monitoru. Ve spodním panelu je zobrazena časová osa stimulační série otevřeného nastavení. V ní je patrný počet a délka jednotlivých bloků klidu a stimulace. Pokud právě probíhá stimulace, je na časové ose vyznačena již uběhlá doba běhu měření.

Základní funkce hlavního okna lze ovládat buď pomocí zobrazovaných panelů, nebo pomocí klasického kaskádového menu. Toto menu je definováno ve třídě *MainMenu*. V závislosti na běhu stimulace jsou některé položky tohoto menu zakázané. Hlavní okno aplikace dále poskytuje rozhraní pro zobrazování informačních, chybových nebo dotazových dialogů.



Obr. 13: Hlavní okno programu.

5.7.1.2 Třída StimulationWindow

Stimulační okno má dvě využití, která si jsou velmi podobná. První z nich zajišťuje vlastní provádění stimulačních podnětů. Druhý způsob využití je pro zobrazení náhledu stimulační série. Oba způsoby využití se liší ovládáním. Při provádění stimulací je zobrazování řízeno časem a příchozími synchronizačními impulzy. Pokud je zobrazen náhled, jsou jednotlivé části stimulační série zobrazovány po stisku klávesy nebo kolečkem myši. V obou případech slouží stimulační okno jako zobrazovač podnětů, které jsou řízeny příslušnými controllery.

5.7.2 Dialogová okna

Dialogová okna aplikace jsou uložena v balíku *gui.dialogs*. Jedná se o dva dialogy. První z nich, *TextAreaDialog*, slouží pro zobrazení velkého textového pole a je využit pouze v případě, že se nezdařil zápis naměřených dat do souboru. Druhý dialog je využit pro nastavení výstupních bitů. Tento dialog má více forem, které dodržují společné rozhraní *OutputDialogInterface*. Formy se využívají v závislosti na možnostech připojeného hardwaru a parametrech otevřeného nastavení.

Pro zobrazení dialogových oken s informací nebo popisem chyby je využíváno statických metod a funkcí třídy *JOptionPane*.

5.7.3 Ovládací panely

Všechny ovládací panely se nacházejí v balíku *gui.panels.impl*. Jejich společná rozhraní jsou v balíku *gui.panels*. Zde se nacházejí jak panely sloužící pro vlastní zobrazování v hlavním okně, tak panely, které mají jiný předmět využití. Druhou jmenovanou skupinu tvoří třídy *StatePanel*, *SettingTopPanel* a *StimulationPanelLine*. Třída *StatePanel* reprezentuje stavový panel měření, který se zobrazuje ve spodní části hlavního okna, viz kapitola 5.7.1.1. Třída *SettingTopPanel* zajišťujte zobrazení tlačítek pro přepínání mezi jednotlivými panely otevřeného nastavení. Třída *StimulationPanelLine* je využívána pro zobrazení jedné linky v panelu *StimulationPanel*.

Ostatní třídy balíku *gui.panels.impl* implementují rozhraní *PanelInterface*. Toto rozhraní je předáváno hlavnímu oknu aplikace, které prostřednictvím jeho funkcí získá potřebné informace pro zobrazení panelu. Zároveň rozhraní obsahuje funkce, které chrání uživatele před ztrátou neuložených dat při přepnutí na jiný panel. Některé panely také implementují rozhraní *BackgroundColorChanger*. Toto rozhraní definuje funkci na předání nastavené barvy pozadí, aby mohl být zobrazen náhled zvoleného odstínu na celé obrazovce.

Panely, v nichž se tvoří modely, implementují rozhraní *ModelPanel*. To umožňuje předat otevřený model (přesněji jím implementované rozhraní) instanci třídy kontrolující změnu upravovaného modelu. Dalším rozhraním je *ThumbStarter* pro panely, z nichž je spouštěn náhled stimulace.

Mezi panely, které jsou nezávislé na otevřeném nastavení, patří *TitlePanel*, *GlobalConfigurationPanel*, *InputCheckPanel*, *HelpDescriptionsPanel* a *GridPanel*.

- *TitlePanel* tvoří úvodní okno aplikace.
- *GlobalConfigurationPanel* zobrazí formulář, kde se zadává nastavení celého programu.
- *InputCheckPanel* zobrazuje panel sloužící pro náhled aktuálně čtených hodnot z Arduina.
- *HelpDescriptionsPanel* zobrazuje přehled (název a popis) všech uložených nastavení stimulace.
- *GridPanel* je využit pro nastavení parametrů mříží.

Obsah ostatních panelů závislý na otevřeném nastavení.

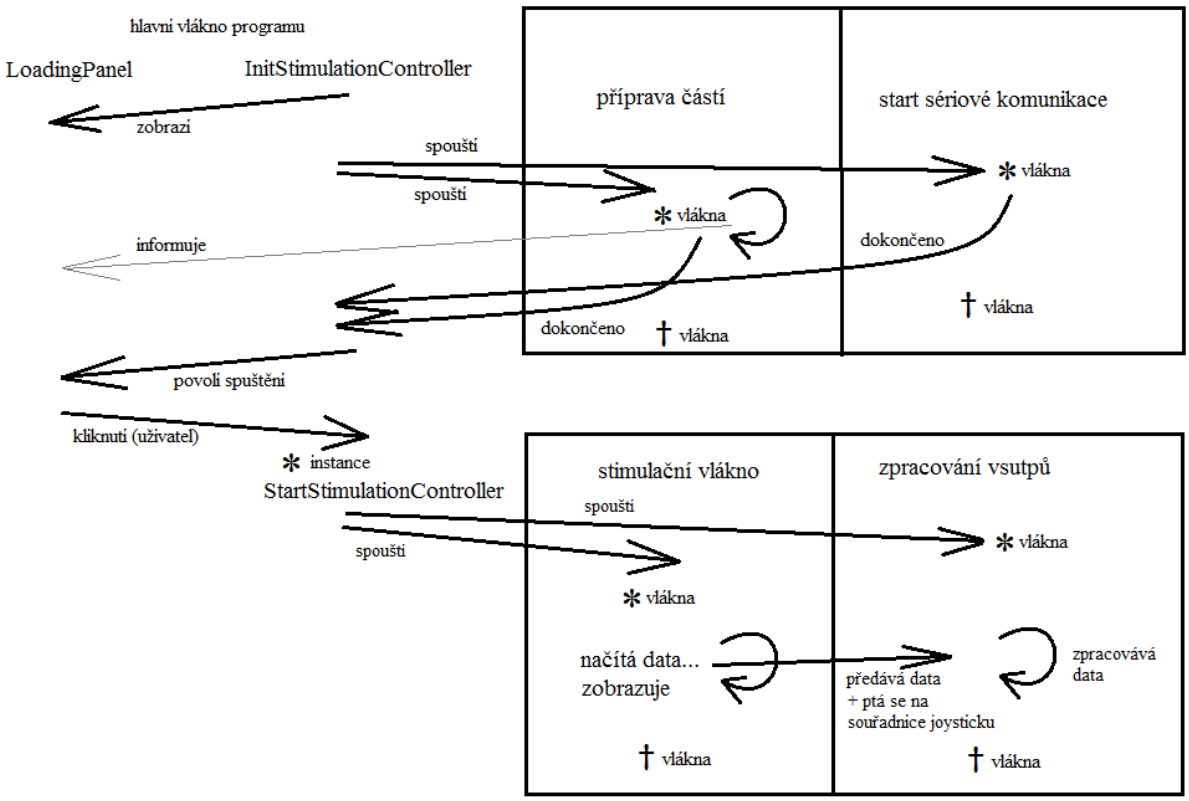
- *SettingPanel* tvoří úvodní panel zobrazeného měření.
- *LoadingPanel* je zobrazen, pokud probíhá příprava dat pro vlastní stimulaci nebo její náhled.
- *DescriptionPanel* umožňuje nastavit název a popis měření.
- *ParametersPanel* zobrazuje formulář, sloužící pro úpravu základních údajů nastavení stimulace.
- *StimulationPanel* je panel, ve kterém se nastavují jednotlivé body stimulační série.

5.7.4 Stimulační panely

Panely využívané pro provádění stimulací implementují společné rozhraní definované abstraktní třídou *StimulationViewPart*. Toto rozhraní definuje funkce pro přípravu dat, zahájení či ukončení stimulace, vložení do zobrazovacího okna nebo vložení náhledu do hlavního okna. Tyto panely vznikají až v okamžiku zahájení přípravy stimulace nebo náhledu. Vytváří je instance třídy *StimulationViewPartFactory*. Kromě panelů reprezentujících používané druhy stimulace (vyjmenované v kapitole 5.4.2), nalezneme v balíku třídu *DotPanel*, která zobrazí kurzor joysticku.

5.8 Provádění stimulací

Před vlastním startem stimulace musí proběhnout její příprava, tj. vytvoření stimulačních panelů (viz kapitola 5.7.4) a nastartování sériové komunikace. Tyto dva úkoly má za úkol instance třídy *InitStimulationController*. V případě, že je možné odstartovat stimulaci, informuje tato třída grafické rozhraní. Vlastní start je pak závislý na akci uživatele. V okamžiku kliknutí je předáno řízení třídě *StartStimulationController*, která odstartuje instance tříd *StimulationController* a *InputController*. Obě instance běží ve svém vlastním vlákně. Vzájemné volání tříd a okamžiky vzniku a zániku vláken zachycuje obr. 14.



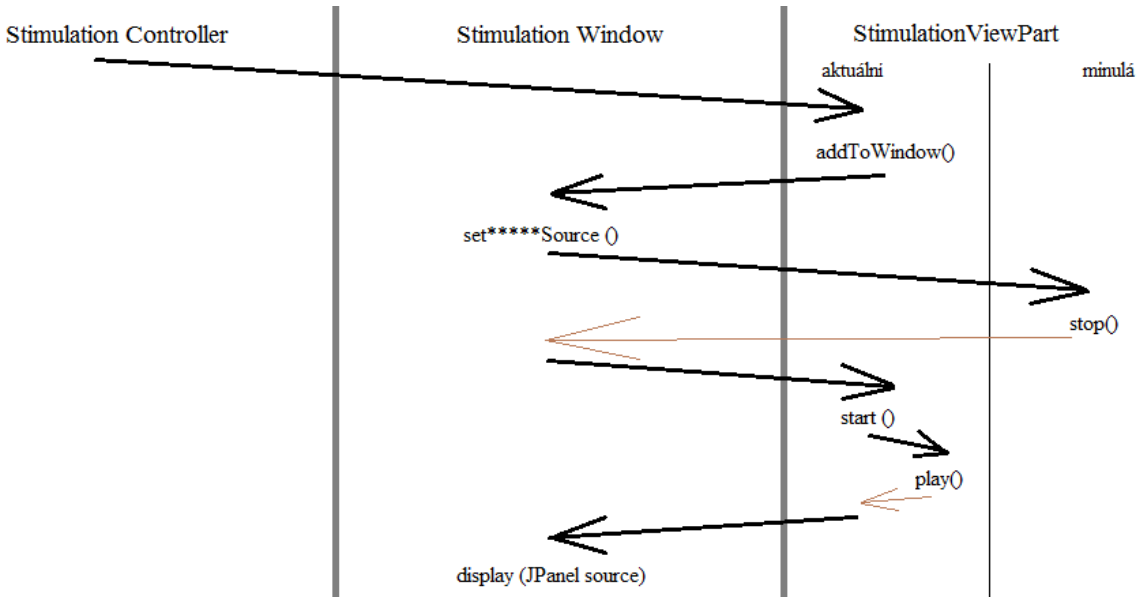
Obr. 14: Vlákna na startu stimulace.

Vlastní provádění stimulací má za úkol třída *StimulationCotntroller*. Tato třída přijímá data z A/D převodníku prostřednictvím poskytovaného rozhraní, tedy *Iteratoru* (viz kapitola 5.3.1). Zobrazování stimulační série je závislé na čase a částech stimulační série. To znamená, že k jeho ukončení dojde po provedení posledního stimulačního podnětu. Z přijatých dat si tato třída vybere pouze informaci o příchodu synchronizačního impulzu a čase jeho příchodu (viz kapitola 5.3.2). Poté jsou přijatá data předaná instanci *InputController*.

Při vlastním odstartování stimulačního podnětu je na jeho instanci zavolána metoda *addToWindow*. Tím je stimulačnímu oknu nastaven zdroj dat a jeho typ (obraz, zvuk nebo video). Zároveň musí dojít k ukončení některých předchozích stimulací. Tím je zaručeno, že jedna zvuková stopa může být přehrávána během zobrazení několika obrazů, ale je ukončena startem videa. Po ukončení předchozí stimulace může být odstartována nová. Vzájemná komunikace tříd v tomto okamžiku je zachycena na obr. 15. Po provedení poslední stimulace jsou stále načítána data z A/D převodníku, už na ně ale není reagováno. Příchozí data jsou nyní pouze předávána vedle běžící instanci třídy *InputController*, která je průběžně zpracovává.

Instance třídy *InputController* běží ve smyčce až do doby, kdy obdrží informaci o ukončení měření a dojdou jí data ke zpracování. Jinak jsou zpracovávána analogová data a jinak digitální. Pokud jsou příchozí digitální data stejná, jako v minulém případě, jsou zahozena. V případě, že se data liší, hledá se, stiskem kterého tlačítka nastala změna, a tato je zaznamenána. Data jsou zpracovávána do formátu *on set, duration* (viz kapitola 4.5). Je-li to

nutné, dochází při zpracování analogových dat k výpočtu velikosti změny příchozího signálu a jeho následné úpravě. Jinak jsou analogová data pouze ukládána do množiny, která bude na konci měření zapsána do souboru. Pro uchování informací o příchozích datech a jejich změnách jsou využity třídy z balíku *controller.stimulation.input*.



Obr. 15: Start stimulačního podnětu.

Třída ThumbController slouží pro náhled stimulace. Je využívána jak pro náhled celé stimulační série otevřeného nastavení, tak pro náhled vytvářených modelů mříží. Proto panely, ze kterých je spouštěn náhled, musí implementovat rozhraní ThumbStarter. Toto rozhraní zajišťuje předání informace, zda během náhledu přeskakovat části stimulace, které nemají obrazový výstup. Instance třídy ThumbController zároveň slouží jako event listener pro stisky kláves a rotace kolečkem myši, pomocí čehož dochází k přesunům na následující nebo předchozí části stimulační série.

5.9 Ukládání nastavení

Nastavení měření jsou ukládána do souborů. Toto byl jeden z požadavků zadavatele práce. Formát souborů je popsán v kapitole 4.4. Ukládání a načítání dat je skryto za rozhraním, tudíž bude v případě potřeby možné nahradit ukládání do souborů například databází. Jedná se o tato rozhraní:

- *SettingLoader* – funkce load pro načítání nastavení
- *SettingSaver* – funkce save pro ukládání nastavení
- *SettingFinder* – funkce findAll pro vyhledání všech nastavení

V budoucnu může být toto rozhraní (zejména *SettingFinder*) rozšířeno dle potřeb používání aplikace. Třídy související s ukládáním nastavení (včetně listenerů reagujících na kliknutí na tlačítka) jsou v balíku *controller.persistence*, implementace pro práci se soubory je v balíku *controller.persistence.impl*.

5.10 Ostatní controllery

Ostatní třídy, které jsou umístěné v balících contr*oller.setting, controller.configuration. controller.model* a *controller*, si jsou velmi podobné. Balík controller.configuration obsahuje listenery, které obsluhují nastavení konfigurace programu. Třídy, jejichž název začíná slovem Set, mají na starost přečtení dat z formulářů a jejich nastavení. Balík *controller.setting* obsahuje obdobně strukturované třídy pro úpravu parametrů nastavení jednotlivých měření. Balík *controller.model* obsahuje listenery pro tlačítka související se správou modelů. V názvech tříd je vždy na prvním místě typ modelu (zatím pouze Grid).

Třídy v balíku *controller* obsluhují reakce na tlačítka, která nejsou zařazena do již dříve popisovaných oblastí aplikace. Třída *AbstractController* je předkem většiny tříd z balíků controller a zajišťuje referenci na zobrazovací okna.

6 Ověření řešení

Pro ověření funkčnosti a kvality řešení byly průběžně prováděny experimenty, které potvrzovaly nebo vyvracely předpokládané závěry. Zjišťovala se například rychlost načítání synchronizačních impulzů při výběru vhodného A/D převodníku (viz kapitola 4.2) nebo prodlevy startu při použití různých knihoven pro přehrávání audio a video souborů (viz kapitola 4.3). Tyto experimenty byly prováděny iterativně po dobu několika měsíců během dokončování jednotlivých částí aplikace tak, aby nalezené nedostatky mohly být průběžně odstraňovány. Zároveň docházelo k drobným testům uživatelského rozhraní, při kterých pod dohledem autora pracovala s aplikací osoba, která ji neznala. Poznatky z těchto pozorování byly v další tvorbě aplikace zužitkovány.

Dále byl proveden experiment, při kterém byla testována funkčnost a rychlost zpracování analogových hodnot při snímání joysticku. Testovala se zejména správnost a rychlost výpočtů, které provádějí lineárně narůstající ovlivnění příchozích dat. Během testu byl joystick zapojen do dvou Arduin, každé z nich bylo připojeno do jiného počítače. V prvním byly příchozí hodnoty zpracovány již používaným algoritmem, druhý počítač obsahoval vyvíjenou aplikaci. Cílem testu bylo zjistit, zda je obrazový výstup na obou počítačích shodný, dále byly porovnány oba soubory se zaznamenanými příchozími daty.

Bylo zjištěno, že obrazový výstup se na obou počítačích téměř shodoval, na počítači s vyvíjeným softwarem byl poněkud více rozkmitaný. To je ovlivněno častějším načítáním dat, neboť původní software načítá data každých 40 milisekund, zatímco nový software zpracovává data tak často, jak mu je Arduino posílá, tj. při rychlosti 9600 bitů/s průměrně každých 5 milisekund.

Větší rozdíl byl zaznamenán při porovnání výsledných výstupních souborů. Soubory pořízené novým programem byly z důvodu častější registrace dat mnohonásobně větší a data z obou souborů se navzájem lišila. Příchozí data v souboru vytvořeném novým softwarem byla ovlivněna pouze změnou úhlu a délky v polárních souřadnicích. Data z původního programu obsahují přímo souřadnice zobrazeného bodu, tzn., že byly oříznuty krajní hodnoty dat z okamžiků, kdy byl kurzor vykreslen na okraji pole, přestože mohl být více vzdálen od jeho středu.

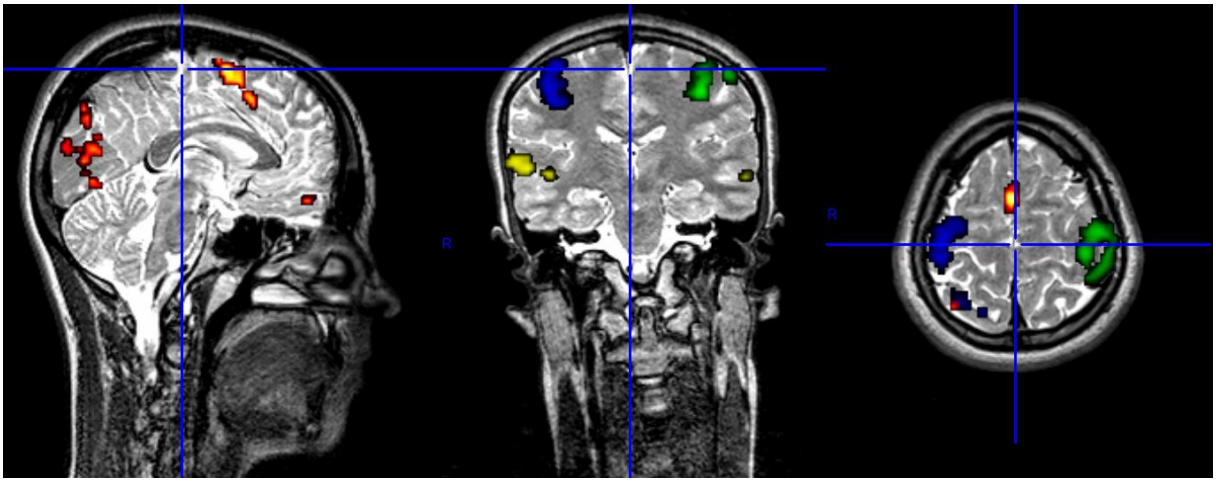
Další rozdíl oproti souboru dat z původního programu byl způsoben násobením dat dvěma, které nový program neprováděl. Jiná odlišnost v datech obou souborů byla způsobena jejich četnějším přijímáním, kdy data v novém softwaru byla vypočítána jako průměr z 20 posledních záznamů, které dorazily během několika desítek milisekund, zatímco starý program průměroval 20 posledních vzorků z posledních 800 milisekund.

6.1 Klinické měření

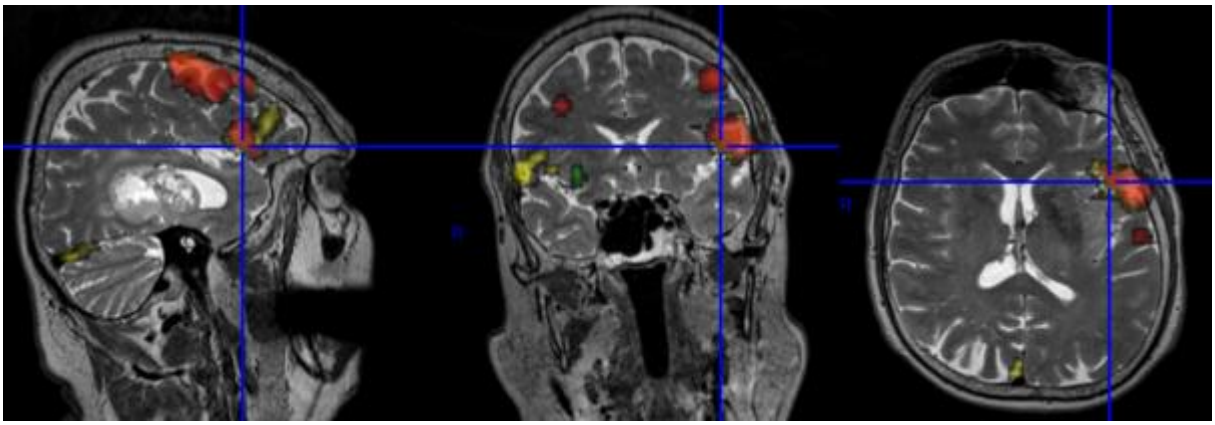
Software byl zkušebně využit pro naměření dat motoriky horních končetin. Toto měření je typický příklad blokového návrhu měření (viz kapitola 2.1.2), neboť se skládá z deseti dynamik v době stimulace, následovaných deseti dynamikami v době klidu. Na začátku každé doby je promítnut jeden stimulační obraz. Zobrazení center motoriky je na obrázku 15. Dále byl proveden test verbální fluence, jehož výsledky jsou zachyceny na obrázku 16.

6.2 Výzkumné měření

Aplikace byla také využita pro stimulaci přehráváním videí při testu pacientů s roztroušenou sklerózou. Bylo přehráno 10 videí o délce 30 sekund postupně třem pacientům a jednomu dobrovolníkovi. Jednalo se o měření v dlouhodobé studii, která již byla započata za pomoci původního jednoúčelového programu.



Obr. 16: Centra motoriky. Levá horní končetina modře, pravá horní končetina zeleně.



Obr. 17: Verbální fluence.

7 Závěr

Práce zhodnotila v současnosti používaná řešení pro stimulační podněty pro fMRI a potřeby pracovitě Základny radiodiagnostiky a intervenční radiologie (ZRIR) Institutu klinické a experimentální medicíny (IKEM) v Praze. Na základě těchto poznatků byla navržena a vytvořena nová aplikace pokrývající veškeré současné, i některé budoucí potřeby svých uživatelů. Díky integraci přehrávání videí a zobrazování signálů z joysticku na mříži je možné opustit skupinu doplňkových skriptů, které suplují nedostatky předchozího systému SySy.

Vytvořená aplikace již je v současnosti používána pro měření funkčního MR zobrazování na pracovišti v IKEMu. V následujících měsících se předpokládá její instalace na Slovensko pro potřeby Nemocnice na Kramároch a do vznikajícího Národního ústavu duševního zdraví v Klecanech. Možné je i rozšíření do jiných pracovišť v České republice.

Vývoj aplikace dokončením a odevzdáním této práce nekončí, v budoucnu bude možné se například zaměřit na úpravu některých open-source knihoven, díky čemuž by byl nahrazen v současnosti používaný Java Media Framework. Spolu s tím také bude možné aplikaci provozovat na 64bitové platformě. Joystick lze využít i pro jiné typy měření, například pro hledání průchodu bludištěm. V brzké době bude realizováno zobrazování skupiny bodů, které tvoří lidské postavy. O této možnosti se zmiňuje MUDr. Filip Španiel, Ph.D.:

Předkládané řešení představuje půdorys pro studium funkčního korelátu sociální kognice. Aktivační paradigmata použitelná v rámci fMRI vyšetření představují v této oblasti obtížnou výzvu a jejich výskyt je v literatuře celkem sporadický.

V našem případě bude vyšetřované osobě prezentována animace vzájemně sociálně interagujících humánních modelů, které budou zcela očištěny od potenciálních distraktorů, jako jsou individuální rysy, pohlaví, habituální typologie.

Podkladem jsou pohybová 3D základová data snímána pomocí Kinectu, který je vybaven infračervenou kamerou. Údaje o pohybu snímaného figuranta jsou v první fázi převáděny pomocí speciálního software do prostorového vzorce lidského těla, zakódovaného do celkem patnácti bodů, označujících uzlové prvky lidské postavy (hlava, ramena, paže, pánev, velké klouby, kotníky).

V krátkých spotech vstupují dvojice snímaných figurantů do vzájemné sociální interakce (například různé formy formálního a neformálního pozdravu) případně jeden snímaný figurant v modelové situaci oslovuje tímto způsobem přímo vyšetřovaného. Jsou vytvářeny 8-10 sec trvající spoty, které jsou koregistrovány s fMRI akvizicí. Jako kontrolní kondice slouží otáčející se Neckerova krychle, složená z bodů označujících vrcholy tohoto geometrického útvaru.

Vyšetřovaná osoba ve scanneru určuje, zda jsou jednající animované postavy promítané dataprojektorem do zrcátka, umístněného na sekundární hlavové cívce, ve formálním či neformálním vztahu, popř. v jakém vztahu je samostatně vystupující postava

k samotnému vyšetřovanému. Dešifrování gestické komponenty, která nese specifický, sociálně relevantní význam, vyžaduje od vyšetřovaného u takto abstrahovaných postav složených ze soustavy bodů, vyšší míru mentálního úsilí, což zprostředkuje vyšší odstup signálu od šumu v rámci fMRI vyšetření.

Použijeme i modifikaci s animací paže a prstů. V této větvi budeme studovat neurální koreláty při observaci smysluplného motorického aktu. Použijeme sérii animovaných motorických vzorců (zatloukání hřebíku, řezání pilou, uchopení šálku apod.) V naší soustavě bude mít každý bod definující uzlovou struktur paže autonomní souřadnice, které bude možné úhlově deviovat. Postupně se bude během prezentace úhlová deviace redukovat. Z chaosu náhodně se pohybujících bodů náhle vyvstane vlastní tvar, tedy gestalt významotvorného pohybu. Úlohou vyšetřovaného bude koregistrovat pomocí stisknutého tlačítka emergentní stav, ve kterém si subjekt náhle uvědomí smysluplnost pozorovaného motorického aktu a zařadí ho do správné kategorie. Tento přístup umožní sledovat aktivaci předpokládaných zrcadlových neuronů (mirror neuron system) u člověka.

Oba přístupy budou použity u zdravých kontrol a u pacientů se schizofrenií.

Seznam literatury

[1] TINTĚRA, Jaroslav; VYMAZAL, Josef: *Funkční a metabolické MR zobrazení mozku*. Česká radiologie: Časopis radiologické společnosti. Únor 2005

[2] HUETTEL, Scott A.; SONG, Allen W.; MCCARTHY, Gregory. *Functional Magnetic Resonance Imaging.* Second Edition. Sunderland (MA.): Sinauer Associates, Inc., 2009. 542 s.

[3] REIMER, Peter; PARIZEL, Paul M.; STICHNOTH, Falko-Alexander; MEANEY, James F.M. *Clinical MR Imaging : A Practical Approach.* 2nd edition. Berlin: Springer-Verlag, 2003. 597 s. ISBN 3-540-64098-3.

[4] RYDLO, Jan. *Periferie stimulačního systému pro funkční MR zobrazování*. Praha: ČVUT 2011. Diplomová práce, ČVUT, Fakulta elektrotechnická, Katedra teorie obvodů.

[5] TINTĚRA, Jaroslav; RYDLO, Jan: *Stimulační systém pro fMRI – návod k použití*, Praha, 2013

[6] BUREŠ, Martin. *Porovnání MR obrazů ve vybraných oblastech mezi 1,5 T a 3 T.* Praha: ČVUT 2009. Bakalářská práce. Fakulta biomedicínského inženýrství, Katedra biomedicínské techniky.

[7] JanasCard. *AD14ETH – návod k obsluze*. Praha, 2012

[8] Funkční magnetická rezonance / fMRI Brno, *Popis principu fMRI.* [online], 2004 – 2013 [cit. 20. 4. 2014]. http://fmri.mchmi.com/main_index.php

[9] KYBIC, Jan; HORŇÁK, Juraj; BOCK, Michael; HOZMAN, J., DOUBEK, Pavel. *Magnetická rezonance* [online], 2008 – 2013 [cit 25. 4. 2014]. http://cw.felk.cvut.cz/wiki/_media/courses/a6m33zsl/mri1.pdf

[10] PIERCE, Jonathan. *PsychoPy v1.80* [online], 2014 [cit. 15. 1. 2014]. http://psychopy.org/

[11] *Arduino Home* [online], 2014 [cit. 20. 4. 2014]. http://arduino.cc

[12] *Rychlost I/O operací v Javě* [online], 2013 [cit. 20. 4. 2014]. https://cw.felk.cvut.cz/wiki/courses/a4m33pal/cviceni/java_io

[13] *Java Native Access* [online], 2014 [cit. 15. 1. 2014]. https://github.com/twall/jna

[14] *RXTX* [online], 2013 [cit. 25. 10. 2013]. http://rxtx.qbang.org/wiki/index.php/FAQ

[15] *Java SE Desktop Technologies* [online], 2014 [cit. 15. 1. 2014]. http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html

[16] *JMF and MPEG2* [online], 2009 [cit. 15. 1. 2014].
https://community.oracle.com/message/5419391

[17] *Java MP3 Library* [online], 2004 [cit. 15. 1. 2014].
http://www.javazoom.net/javalayer/about.html

[18] *MP3Transform* [online], 2014 [cit. 15. 1. 2014] http://code.google.com/p/mp3transform/

[19] Caprica Software Limited, *vlcj FAQ* [online], 2014 [cit. 15. 1. 2014].
http://www.capricasoftware.co.uk/projects/vlcj/faq.html

[20] *Jffmpeg* [online], 2014 [cit. 15. 1. 2014] http://jffmpeg.sourceforge.net/index.html

[21] ConnectSolutions, LLC, Xuggler [online], 2011 [cit. 15. 1. 2014].
http://www.xuggle.com/xuggler/documentation

Příloha 1 Použité zkratky

A/D	analogově digitální
API	Application Programming Interface
BOLD	Blood oxygeantion level dependent
BSD	Berkeley Software Distribution
CT	výpočetní tomografie
dHB	odkysličený hemoglobin
DLL	dynamicky linkovaná knihovna
FID	Free Induction Decay
FIFO	first in, first out
fMRI	functional Magnetic Resonance Imaging
GPL	General Public License
Hb	odokysličený hemoglobin
HCT	high-speed CMOS s TTL napětím
I/O	vstup/výstup
IKEM	Institut klinické a experimentální medicíny
IP	Internet Protocol
JDK	Java Development Kit
JMF	Java Media Framework
JNA	Java Native Access
jSSC	Java Simple Serial Connector
LGPL	Lesser General Public License
MP3	MPEG-2 Audio Layer III
MPEG-2	Motion Picture Experts Group 2
MR	magnetická rezonance

NMR	nukleární magnetická rezonance
MRI	Magnetic Resonance Imaging
PCM	pulzně kódová modulace
RF	radiofrekvenční
TCP	Transmission Control Protocol
TR	repetiční čas
TTL	tranzistorově-tranzistorová logika
UTF-8	UCS Transformation Format
VLC	VideoLAN Client
VLCJ	VideoLAN Client for Java
WAV	Waveform audio file format
ZRIR	Základna radiodiagnostiky a intervenční radiologie

Příloha 2 Instalační příručka

Pro instalaci aplikace je třeba nejprve nainstalovat do počítače všechny podpůrné její součásti. Aplikace je určena pro operační systém Windows 7.

Jedná se o tyto aplikace:

- Java 7, 32 bitová verze, http://download.oracle.com/otn-pub/java/jdk/7u55-b13/jre-7u55-windows-i586.exe
- JMF, Windows Performance Pack, http://download.oracle.com/otn-pub/java/jmf/2.1.1e/jmf-2_1_1e-windows-i586.exe
- Arduino, tutoriál k instalaci na http://arduino.cc/en/Guide/Windows

Ostatní knihovny (*jffmpeg-1.1.0.jar, jmf.jar a jssc-2.8.0.jar*) jsou přiloženy ve složce lib. V případě problémů zkontrolujte jejich přítomnost. Program spustíte pomocí souboru *StimulSys.jar*.

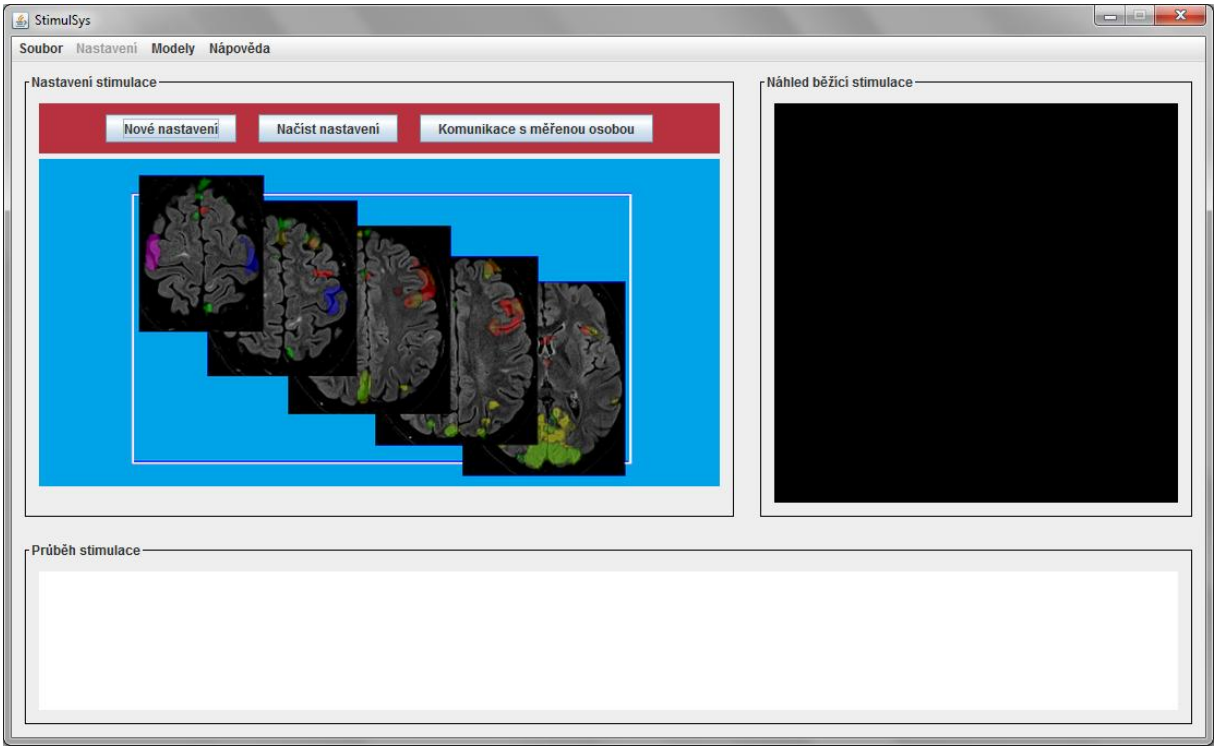
Pro plnohodnotné používání aplikace je nutné mít připojeny i další hardwarové součásti.

Příloha 3 Uživatelská příručka

Uživatelská příručka se skládá z těchto bodů:

1) Konfigurace programu
2) Tvorba nového nastavení
3) Tvorba stimulační série
4) Tvorba modelů
5) Náhled stimulační série
6) Běh stimulační série
7) Registrace vstupů
8) Formát ukládaných souborů
9) Přehled dostupných nastavení
10) Jednosměrná komunikace s pacientem

Po spuštění aplikace je otevřeno základní okno. Z něj je možné vytvořit či otevřít nastavení, přejít na tvorbu modelu nebo do konfigurace programu. Při prvním spuštění aplikace je vhodné přejít ke konfiguraci programu.



Obr. 18: Úvodní okno aplikace.

P3.1 Konfigurace programu

Panel konfigurace programu nabízí nastavení veškerých aspektů aplikace. Před vstupem do konfigurace jste varováni, neboť změny provedené v této části aplikace mohou způsobit její praktickou nepoužitelnost. Je nutné nastavit port Arduina a rychlost sériové komunikace. Rychlost komunikace je třeba zvolit stejnou, jako v programu nahraném do desky. Doporučená rychlost komunikace je 9600 bitů za sekundu.

Pokud počítač nemá standardní sériový port (ve Windows označeny COM1 až COM3) a k počítači je připojeno pouze jedno Arduino, je k dispozici pouze jedna možnost výběru. Pokud není vybrán žádný sériový port, konfiguraci programu není možné uložit, tudíž je zbytečné ji provádět bez připojeného sériového portu. U některých možností konfigurace je k dispozici nápověda, ta je zobrazena po kliknutí nebo najetí na oranžový otazník u příslušných řádků.

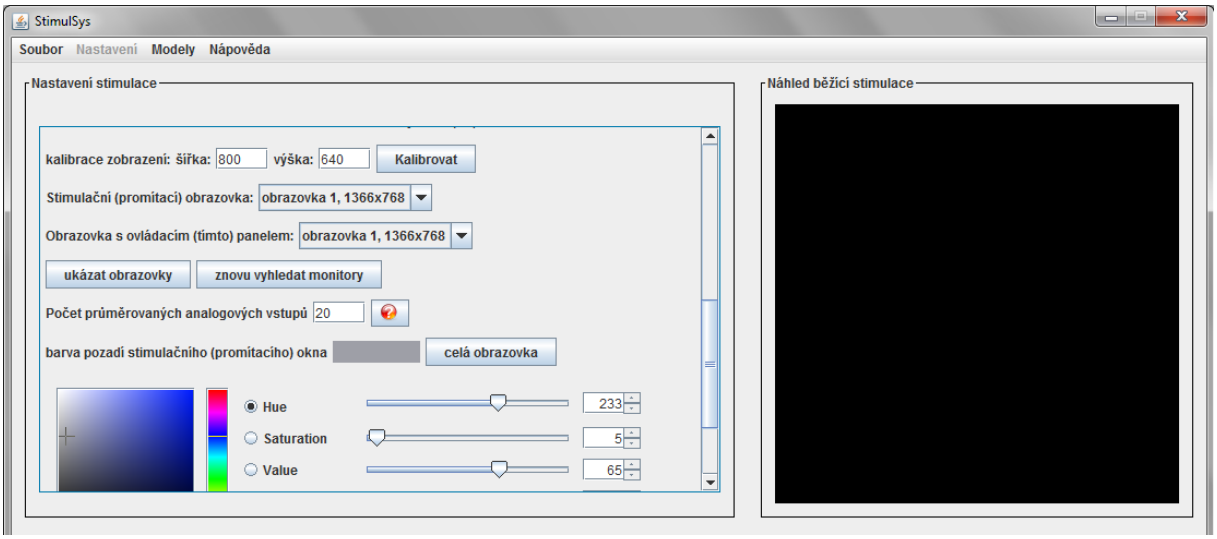
Kalibrace zobrazení určuje pozici, kam se zobrazují obrazové stimulační podněty (tj. obrazy, videa a mříže). Přesná pozice je určena také zobrazovacím módem, kdy se zobrazí buď střed na pozici středu (bez ohledu na velikost obdélníku použitého při kalibraci) nebo levý horní roh do levého horního rohu (bez ohledu na velikost obdélníku). Kalibraci lze ovládat šipkami nebo tlačítky.

Pokud chcete na jednotlivá obrazová výstupní zařízení (monitory, projektor) zobrazovat různé části aplikace, nastavte tuto možnost v části monitorů. Stimulační obrazovka je typicky projektor, ovládací obrazovka by měla být jiná. Je možné, že bude třeba změnit nastavení Vašeho operačního systému tak, aby bylo možné zobrazovat různé výstupy na jednotlivá zařízení. Ve Windows 7 klikněte pravým tlačítkem myši na plochu, vyberte *Rozlišení obrazovky*. V nabídce *Více monitorů* pak vyberte volbu *Rozšířit tato zobrazení*. (viz obr. 20). Po změně nastavení operačního systému bude nutné ještě kliknout na tlačítko Znovu vyhledat monitory.

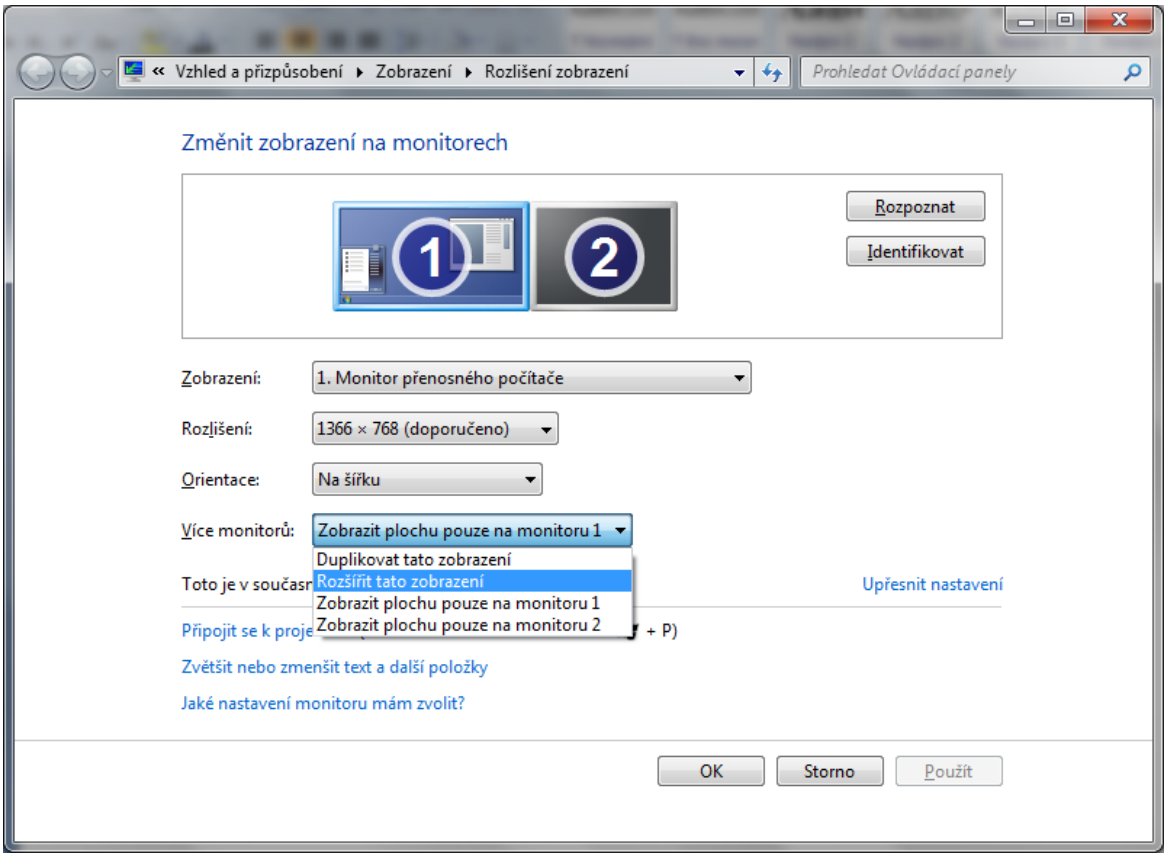
Barva pozadí stimulačního okna je použita jako základní barva pro pozadí jednotlivých nastavení, dále je tímto odstínem vybarveno okno stimulační obrazovky v době, kdy neběží žádná stimulační série.

Konfigurace programu je uložena do souboru *configuration.dat* v kořenové složce aplikace. Pokud tento soubor není nalezen, je použita základní konfigurace platná v době prvního spuštění programu.

Pro kontrolu vstupních dat z Arduina můžete využít položku Kontrola vstupů z menu Soubor. V ní se zobrazují aktuální příchozí hodnoty analogových a digitálních dat.



Obr. 19: Konfigurace programu.



Obr. 20: Nastavení monitorů ve Windows 7.

P3.2 Tvorba nového nastavení

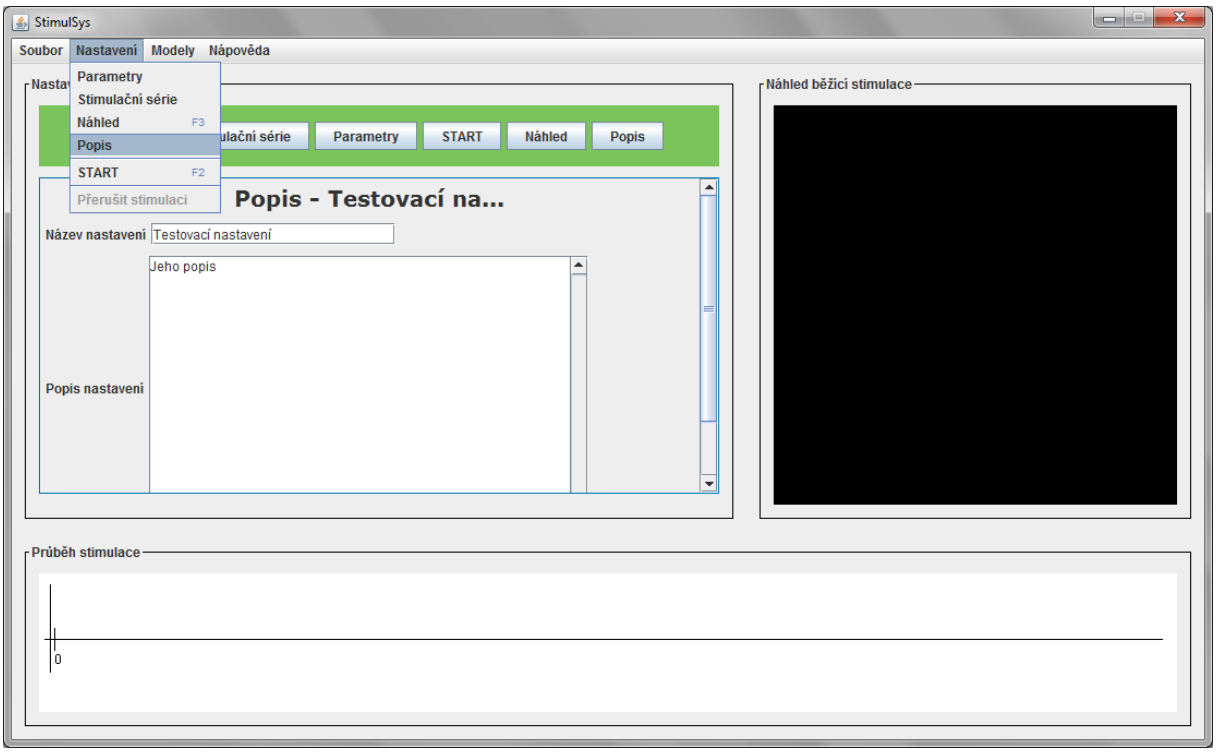
Pro vytvoření nového nastavení vyberte v menu Soubor možnost *Nové nastavení*, použijte klávesovou zkratku *CTRL + N* nebo na úvodní obrazovce programu klikněte na tlačítko *Nové nastavení*. Po vytvoření bude zobrazen panel, kde je potřeba zadat název a nepovinně i popis tvořeného nastavení. Název a popis nastavení je využit při výpisu všech nastavení v menu Nápověda → Popisy nastavení.

Pokračujte zadáním parametrů nastavení. Vyberte buď v menu Nastavení možnost *Parametry*, nebo klikněte v horní zelené lince na možnost *Parametry*. Zde je nutné zadat počty epoch a počty dynamik v jednotlivých epochách. Je možné nastavit počet dynamik buď v době klidu, nebo v době stimulace na nulu.

- Počet epoch reprezentuje počet úseků klid + stimulace v měření.
- Počet dynamik reprezentuje počet skenů.

V nastavení parametrů je dále možné zadat zdrojovou složku souborů. Stimulační soubory jsou pak načítány z ní a nikoliv ze standardních složek, nastavených v konfiguraci programu. Pokud zadáte hodnoty zpoždění stimulace A a B, můžete pak v nastavení stimulační série použít pro určení startu podnětu kromě číselných údajů také tato písmena.

Zde také můžete určit, kolik bude během měření registrováno digitálních a analogových vstupů.



Obr. 21: Nastavení popisu.

P3.3 Stimulační série

Nastavení stimulační série je možné až v okamžiku, kdy je určen počet dynamik v měření. Do této nabídky lze vstoupit kliknutím na Stimulační série v horní zelené lince nebo vybráním položky Stimulační série v menu Nastavení.

Každý řádek reprezentuje jednu součást série. Číslice úplně vlevo představuje pořadí synchronizačního impulzu (tedy sejmuté dynamiky). Je podbarvena dle toho, zda v té době má probíhat stimulace nebo klid. Stimulaci znázorňuje oranžové podbarvení, klid světle zelené. Stejné barvy jsou použity i ve spodním přehledu průběhu stimulace. Druhá číslice určuje start stimulace. Zadává se v milisekundách od posledního synchronizačního impulzu. Jinou možností je zadat písmena „a“ nebo „b“, je však třeba předem definovat jejich hodnoty v nastavení parametrů. Start stimulace není možné měnit u řádků reprezentujících synchronizační impulzy.

Další položka určuje typ stimulace. Jsou dostupné tyto možnosti:

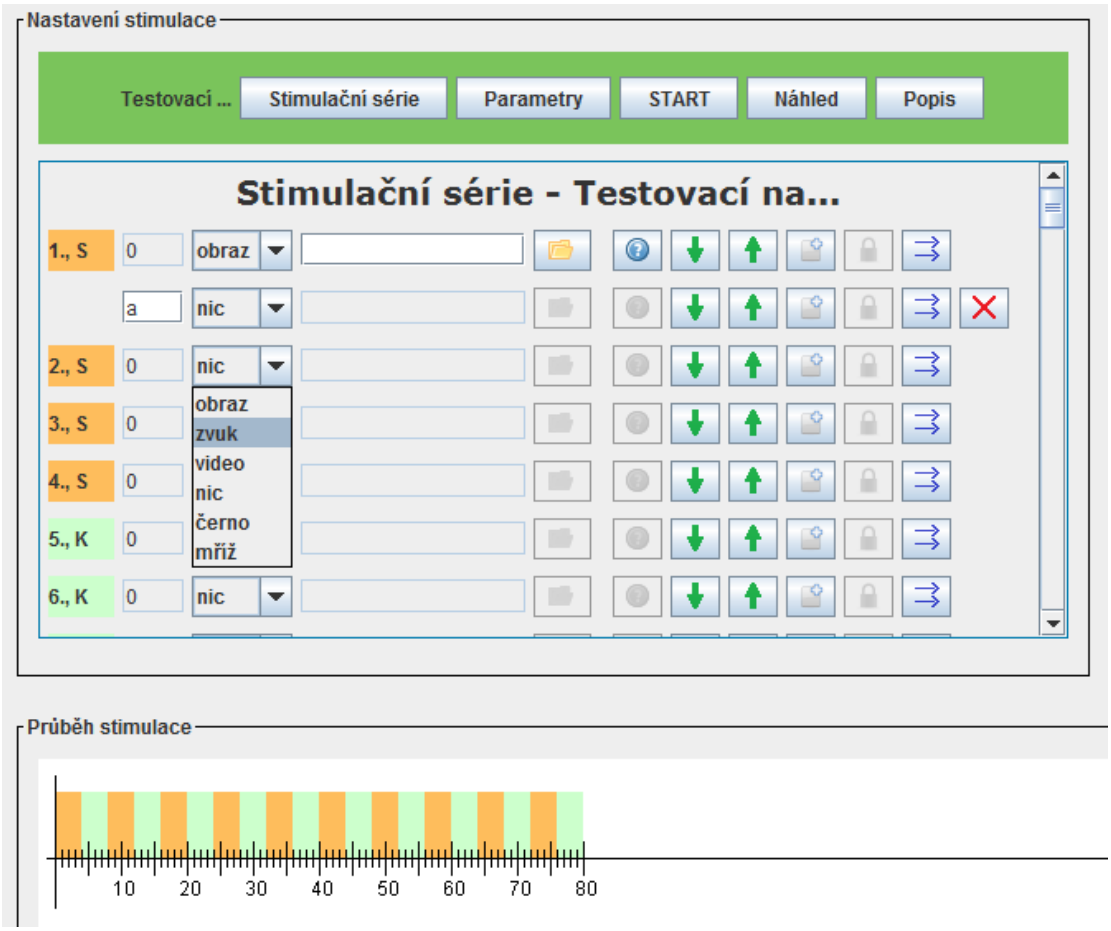
- **Obraz**: Dojde k promítnutí obrazu
- **Zvuk**: V zadaném okamžiku dojde ke startu přehrání zvuku
- **Video**: V zadaném okamžiku je spuštěno video
- **Mříž**: V zadaný čas je zobrazen model mříže
- **Černo**: V zadaném čase je na obrazovce zobrazena barva pozadí (typicky černá)
- **Nic**: Nedojde k žádné akci na obrazovce, běžící stimulace pokračují

Možnost „nic“ můžete zvolit, pokud například chcete zobrazit jeden obraz přes několik dynamik nebo při přehrávání videí delších než jeden repetiční interval (TR). Při výběru některé z prvních čtyř možností (obraz, zvuk, video, mříž) je nutné pokračovat výběrem zdrojového souboru stimulace. U ostatních možností není tato položka dostupná.

Modrý otazník za výběrem souboru provede kontrolu, zda daný stimulační soubor existuje. Dle toho po kliknutí změní svoji podobu, buď na červený vykřičník, nebo na zelenou fajfku. Dvě zelené šipky slouží ke změně pořadí řádků. Čtvrté tlačítko zajistí vložení nového řádku pod stávající. U vložených řádků lze nastavit zpoždění startu stimulace. Tyto vložené řádky lze také zrušit prostřednictvím kříže umístěného vpravo.

Zámek lze využít pouze u zvuků a videí. Zároveň je v *parametrech* nutné nastavit délku intervalu TR. Po kliknutí na toto tlačítko dojde k zablokování následujících řádků, jejichž akce nastanou v době trvání zvuku nebo videa. Automaticky bude vložen nový řádek s nastaveným zpoždění startu na okamžik konce zvuku nebo videa. Opětovným kliknutím na toto tlačítko se zámek následujících řádků zruší.

Poslední tlačítko slouží k nastavení osmi digitálních výstupů nebo také velikosti změny příchozího analogového signálu (pokud ten je registrován).



Obr. 22: Nastavení stimulační série.

P3.4 Tvorba modelů

Pokud chcete používat ve stimulační sérii modely mříží, je nutné je nejprve vytvořit ve správě modelů (menu Modely → Mříž). Pří tvorbě mříží můžete určit velkou část jejich parametrů včetně barev. Barvy je nutné zadávat pomocí hexadecimálních kódů. Pro zobrazení mříže můžete použít tlačítko náhled. Spustí se běžný náhled stimulační série, proto je nutné pomocí šipky doprava nebo dolů přejít na mříž. Okolí mříže bude vykresleno černě. Tlačítko storno vrátí provedené změny na poslední uložené nastavení otevřeného modelu nebo na základní nastavení, pokud žádný model nebyl otevřen. Pro ukončení editace modelů slouží tlačítko zavřít editor v horní části obrazovky programu.

P3.5 Náhled stimulační série

Po nastavení stimulační série můžete zkontrolovat, zda je vše nastaveno správně. K tomu slouží náhled, který můžete spustit z menu (Nastavení → Náhled), stiskem klávesy F3 nebo z horní zelené linky. Náhled není možné spustit za běhu jiné stimulační série. Při startu náhledu máte k dispozici možnost volby, zda ignorovat ty části stimulační série, ve kterých

byl nastaven typ „nic“. Náhled je zobrazen na stejné obrazovce, jako hlavní okno programu. K pohybu v náhledu lze využít klávesnice nebo myši:

Pohyb vzad: šipka dolu, šipka doleva, klávesa Page Down, rolování kolečkem myši dolu.

Pohyb vpřed: šipka nahoru, šipka doprava, klávesa Page Up, klávesa Backspace, rolování kolečkem myši nahoru.

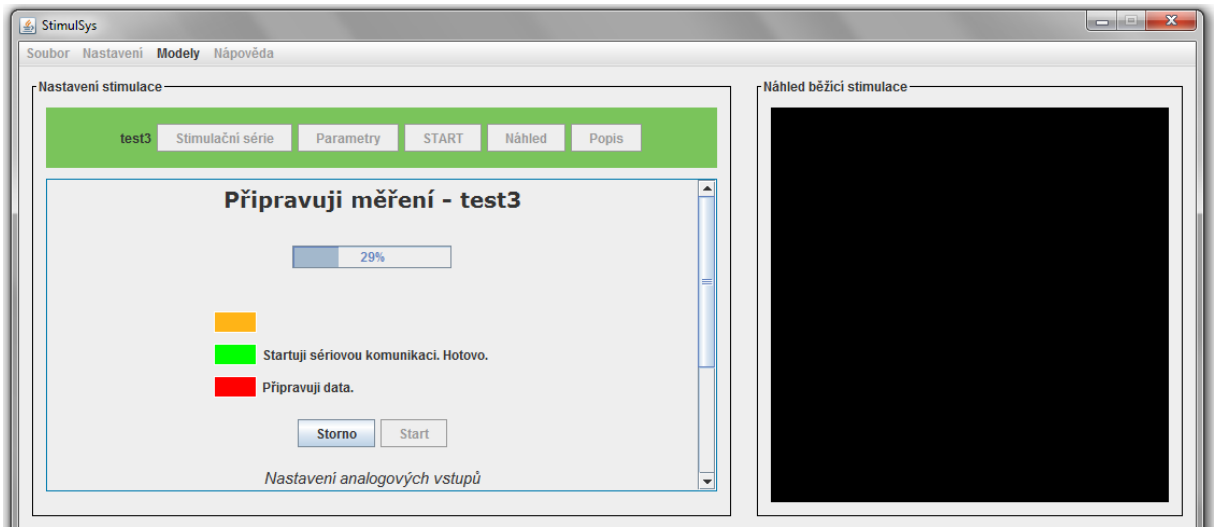
Ukončení náhledu: klávesa ESC.

P3.6 Start stimulační série

Stimulační sérii odstartujete ze stejného panelu, jako náhled. Na tento panel se můžete dostat buď prostřednictvím položky START v menu Nastavení, klávesové zkratky F2, nebo tlačítkem na zelené lince. Během doby načítání jednotlivých částí stimulační série můžete určit počet vzorků, ze kterých se průměruje poslední hodnota analogového vstupu.

Průběh přípravy můžete sledovat dle zobrazených popisků i podle zobrazených semaforových barev. Nejvýše umístěná barva signalizuje celkovou připravenost ke startu. Pod ní naleznete ukazatel stavu inicializace komunikace s Arduinem. Třetí řádek slouží k zobrazení stavu přípravy stimulační série. Můžete také sledovat, kolik procent série je již připraveno.

Poté, co je MR tomograf připraven spustit vlastní měření, klikněte na tlačítko Start a spusťte tomograf. **V průběhu stimulační série můžete počítač používat, mějte však na zřeteli, že jakoukoli prováděnou činností můžete narušit synchronizovaný běh jednotlivých podnětů stimulační série.** Za běhu stimulační série se v hlavním okně zobrazuje náhled a informace o právě probíhající stimulaci.



Obr. 23: Start stimulační série.

P3.7 Registrace vstupů

Informace o stisknutích vstupních tlačítek jsou ukládány do složky určené v konfiguraci programu. V základním nastavení programu je tato složka pojmenovaná *inputs*. Dále jsou soubory členěny do složek podle data, kdy probíhalo měření. Na konci každého měření můžete zadat název souborů. Základní formát tvoří datum a čas měření. Pro každé měření mohou být vytvořeny dva soubory: První s analogovými vstupy, druhý s digitálními vstupy.

U dat z digitálních vstupů (tedy tlačítek) je nejčastěji využívaný formát, který je tvořený časem stisku tlačítka (*on set*) a jeho délka (*duration*). Hodnota *on set* je vztahována k času prvního synchronizačního impulzu, který spustil stimulační sérii. Textový soubor, který obsahuje naměřené údaje, bude potom tvořit jeden řádek hodnot *on set* navzájem oddělených středníkem, druhý řádek souboru by obsahoval stejně oddělené hodnoty *duration* s tím, že první hodnota *on set* tvoří pár s první hodnotou *duration* atd. Pokud je sledováno více tlačítek, následuje volný řádek a hodnoty z dalšího tlačítka ve stejném formátu. Na následující ukázce je vidět, jak vypadá výstupní soubor, pokud jsou registrována dvě tlačítka. První tlačítko bylo stisknuto 11238 milisekund po startu stimulace po dobu 3748 milisekund a 26233 milisekund po startu na 3757 milisekund. Druhé tlačítko bylo stisknuto třikrát: 159, 14994 a 29990 milisekund po startu měření, pokaždé přibližně na 3750 milisekund.

```
11238;26233
3748;3757

159;14994;29990
3755;3743;3740
```

Údaje z analogových vstupů nejsou převáděny do podobného formátu, jsou uloženy v podobě *čas, synchronizační impulz ano/ne* (1/0), *hodnoty 1. – X. analogového vstupu*. Jednotlivé hodnoty v řádku jsou odděleny tabulátorem (znakem \t).

Jestliže dochází k ovlivnění hodnot příchozího analogového signálu, jsou do souboru s výstupními daty ukládány již ovlivněné hodnoty souřadnic. Následující ukázka znázorňuje formát souboru, do kterého jsou zaznamenávány příchozí hodnoty analogových signálů. První sloupec tvoří čas od začátku stimulace, druhý informace o synchronizačním impulzu, třetí a čtvrtý sloupec tvoří hodnoty naměřené v daném čase.

```
88   1    141  327
104  1    150  349
309  1    219  342
698  0    282  405
713  0    283  406
```

P3.8 Formát ukládaných souborů

Data o jednotlivých *nastaveních* jsou ukládána do souborů. Tyto soubory můžete libovolně upravovat v jiných editorech, pokud dodržíte stanovený formát souboru. Celý soubor také musí být uložen v kódování UTF-8. Každá položka stimulační série je právě na jednom řádku souboru. V řádcích jsou jednotlivé položky odděleny tabulátorem (znakem \t). Položky v hranatých závorkách není nutné uvádět. Celý řádek vypadá takto:

```
Zpoždění (tabulátor) typ_stimulace [(tabulátor) odesílaná_data
[(tabulátor) deviace_signálu]]
```

Zpoždění obsahuje čas, kdy má dojít k provedení stimulace definované na aktuálním řádku. Je třeba zadávat i řádky s nulovým zpožděním, tyto pak značí akci v době příchodu synchronizačního impulzu. Zároveň slouží pro průběžnou kontrolu synchronizace aplikace. Pokud tyto řádky vynecháte, stimulační série nemusí proběhnout korektně.

Typ stimulace obsahuje druh zvolené stimulace nebo zdrojový soubor stimulace. Zdrojový soubor je zadáván bez cesty k němu a musí být umístěn ve správné složce. Pokud chcete zobrazit černou obrazovku, zadejte hodnotu *black*, pro žádnou akci *nothing*.

Parametry *odesílaná data* a *deviace signálu* jsou volitelné. Pokud nejsou zadány, předpokládá se, že se neodesílají žádná výstupní data (resp. výstupní data jsou samé nuly) a analogový signál se nemění. Je možné zadat oba parametry, žádný parametr, nebo pouze odesílaná data. Pokud je třeba definovat změnu analogového signálu, je třeba určit i odesílaná data.

Parametr *odesílaná data* obsahuje hodnoty bitů, které se odesílají na výstupní piny Arduina. Tyto hodnoty lze zadat buď v decimální, nebo v binární soustavě. Číselná soustava je určena prvním znakem – buď písmenem *d* pro desítkovou soustavu, nebo *b* pro binární soustavu.

Poslední parametr reprezentuje *deviaci analogového signálu*. Tato změna je zadávána v polárních souřadnicích, tj. jako násobek změny délky a hodnota, o kterou se změní úhel. Zadává se jako tři mezerou oddělená čísla, např. 2.5 30 1. První číslo může být desetinné a představuje násobek změny vzdálenosti. Druhé číslo je změna úhlu a poslední číslice reprezentuje informaci, zda změna signálu má nastat okamžitě nebo lineárně narůstat během doby následujícího TR, aby v okamžiku příchodu synchronizačního impulzu dosáhla zadaných hodnot. Ukázka možných řádků v souboru se stimulační sérií:

```
0     img9.jpg
```
v čase příchodu synchronizačního pulzu je promítnut obrázek, odesílaná data jsou nulová (8x 0), signál se nemění

```
1820 img10.jpg       d128       1.0 0 0
```
1820 ms po příchodu synchronizačního pulzu je promítnut obrázek, sériovou komunikací jsou odeslána data 1 0 0 0 0 0 0 0, signál se nemění, případně je skokově změněn na příchozí hodnoty

```
0     img11.jpg       b10000000 1.0 0 0
```
v čase příchodu synchronizačního pulzu je promítnut obrázek, sériovou komunikací jsou odeslána data 1 0 0 0 0 0 0 0, signál se nemění

```
0     nothing         d0
```
v čase příchodu synchronizačního pulzu nedojde k žádné akci, odeslány jsou samé nuly, signál se nemění

```
0     grid1.grid      d0          1.0 0 1
```
v čase příchodu synchronizačního pulzu je zobrazen model (mříže) ze souboru grid1.grid, odeslány jsou samé nuly, signál se nemění

```
0     nothing         d0          0.25 20 1
```
v čase příchodu synchronizačního pulzu nedojde k žádné akci, odeslány jsou samé nuly, příchozí analogový signál je v polárních souřadnicích zkrácen na čtvrtinu, k úhlu je přičteno 20 stupňů. Změna roste lineárně, až do doby příchodu příštího synchronizačního impulzu. V ten okamžik dosáhne definovaných hodnot.

P3.9 Přehled dostupných nastavení

Pokud chcete zjistit, jaká nastavení měření jsou aktuálně dostupná, vyberte v menu Nápověda položku Popisy nastavení. Zde v levé části obrazovky můžete vybrat nastavení dle jejich názvu, vpravo se pak zobrazí popis daného měření. V tomto přehledu jsou zobrazena pouze nastavení, která jsou uložena ve složce určené v konfiguraci programu.

P3.10 Jednosměrná komunikace

Jednosměrná komunikace s vyšetřovanou osobou probíhá pouze v době, kdy je stisknuto komunikační tlačítko. Tato spouštěcí tlačítka se nacházejí na úvodní obrazovce programu (viz obr. 18) a na obrazovce s otevřeným nastavením (viz obr. 13).

Příloha 4 Příklad fMRI studie

ODCHYLNÁ AKTIVACE V RÁMCI SELF-REFERENČNÍ SÍTĚ U PRVNÍ EPIZODY SCHIZOFRENIE. STUDIE S FMRI.

J. Rydlo[1], F. Španiel[2], I. Ibrahim[1], J. Tintěra[1]

[1] *Institut klinické a experimentální medicíny, Vídeňská 1958/9, Praha 4, 140 21*

[2] *Psychiatrické centrum Praha, Ústavní 91, Praha 8, 181 03*

Cíl

Rozpoznání volní aktivity vlastní od vnějších původců je podle metaanalýz fMRI zdravých subjektů zprostředkováno předním cingulem a frontomediálními oblastmi.

Paradigma mapující aktivační vzorce self-reference je využitelné ve výzkumu schizofrenie. Jádrové příznaky onemocnění jsou charakterizovány ztrátou schopnosti rozlišovat vlastní a cizí, což se na klinické úrovni projevuje příznaky bludů pasivity, prožitků ovlivňování, vkládání či odnímání myšlenek.

Cílem práce bylo zjistit rozdíly v aktivaci self-referenčních oblastí mezi pacienty s první epizodou schizofrenie a kontrolami. Bylo vyvinuto paradigma umožňující pomocí distorze zpětné vizuální vazby manipulovat prožitkem vlastní volní kontroly nad jednoduchým pohybem. Proto bylo potřeba realizovat MR kompatibilní joystick a software umožňující pohybovat kurzorem po obrazovce.

Materiál a metodika

Ke konstrukci joysticku byla použita mechanická část běžného joysticku, doplněná o vlastní elektroniku pro snímání polohy a optický přenos signálu. Software zobrazuje plochu pro pohyb kurzorem a načítá souřadnice polohy joysticku.

Během měření se střídají 20s bloky. V prvním bloku pohyb kurzoru neodpovídá pohybu joysticku, protože souřadnice polohy kurzoru jsou lineárně transformovány (souřadnice r, φ). V druhém bloku pohyb kurzoru odpovídá pohybu joysticku. Vyšetřovaná osoba má rozpoznávat vlastní pohyb kurzoru od ovlineného vnějším původcem a podle instrukcí pohybovat kurzorem v určených částech plochy. Software ukládá souřadnice aktuální polohy kurzoru do textového souboru.

Data jsou analyzována pro koregistraci s fMRI.

Byla vyšetřena skupina 35 zdravých dobrovolníků a 35 pacientů.

Výsledky

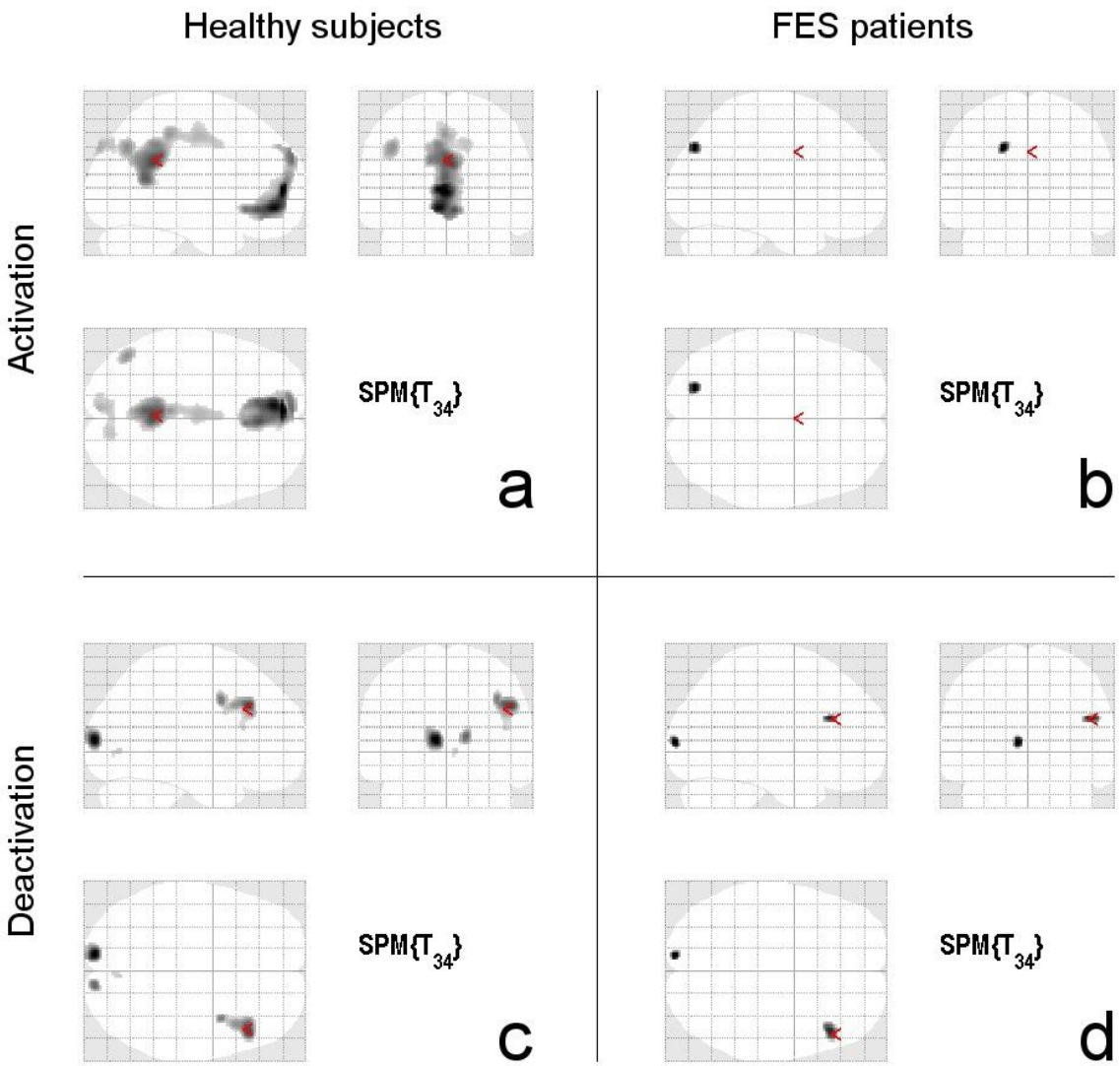
Úspěšnost rozpoznání vlastní volní aktivity od aktivity vnějších původců se u zdravých kontrol a pacientů lišila. V blocích bez vnějšího ovlivňování pohybu dobrovolníci správně přiřazovali pohyb sobě v 89% a pacienti v 76% trvání tohoto bloku. Proto bylo nutné při

vyhodnocování fMRI dat použít koregistraci. Aktivace byly nalezeny v předním a zadním cingulu a deaktivace v insule.

Závěr

Pacienti v nejčasnějších stadiích schizofrenie neaktivují při self-referenčním paradigmatu oblast předního cingula. Funkční narušení této oblasti může být patofyziologickým podkladem schizofrenního onemocnění.

Institucionální podpora: MZ ČR 00023001IKEM



Obr. 24: Aktivace u zdravých subjektů (a, c) a pacientů (b,d)

Příloha 5 Obsah CD

Zkompilovaná aplikace v *.JAR souboru, složka /lib s uloženými knihovnami, soubor configuration.dat s konfigurací programu.

Projekt pro NetBeans 7.3.1.

Program pro desku Arduino 2560.

Text práce ve formátech docx a pdf.