Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Návrh a implementace GUI pro ovládání robotického vozítka

Plzeň 2012 Václav Běle

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 9. května 2012

Václav Běle

Poděkování

Chtěl bych poděkovat mému vedoucímu Bakalářské práce Ing. Pavlu Mautnerovi Ph.D. za skvělé vedení a rady, které mi pomohli práci dokončit do zdárného konce.

Abstract

The purpose of this thesis is to provide to the user a graphical user interface (GUI), which will controll the movement of the robotic vehicle based on evoked potentials. The most important requirements are communication with a measuring computer, providing flashing patterns, communication with the controlled vehicle and displaying visual feedback from an IP camera located on the vehicle. The communication with measuring computer runs over TCP/IP protocol and communication with the controlled vehicle runs over serial port emulated by Bluetooth.

Obsah

1 Úvod

Lidé již po mnohá desetiletí touží ovládat věci pouhými myšlenkami. Touto myšlenkou se zabývá i tento projekt. Možnost ovládat robotické vozítko pomocí mozkových vln není úplně nereálná. Když člověk sleduje obrazce, které blikají zadanou frekvencí, jsou u něj vyvolávány evokované potenciály. Z tohoto signálu lze matematickými metodami poznat, jakou frekvencí blikají obrazce, které měřená osoba sleduje. Když tedy budeme mít čtyři takovéto obrazce, které budou blikat různými frekvenemi a jsme schopni rozpoznat, který z nich meřená osoba sleduje, můžeme podle toho posílat řídící signály ovládanému objektu.

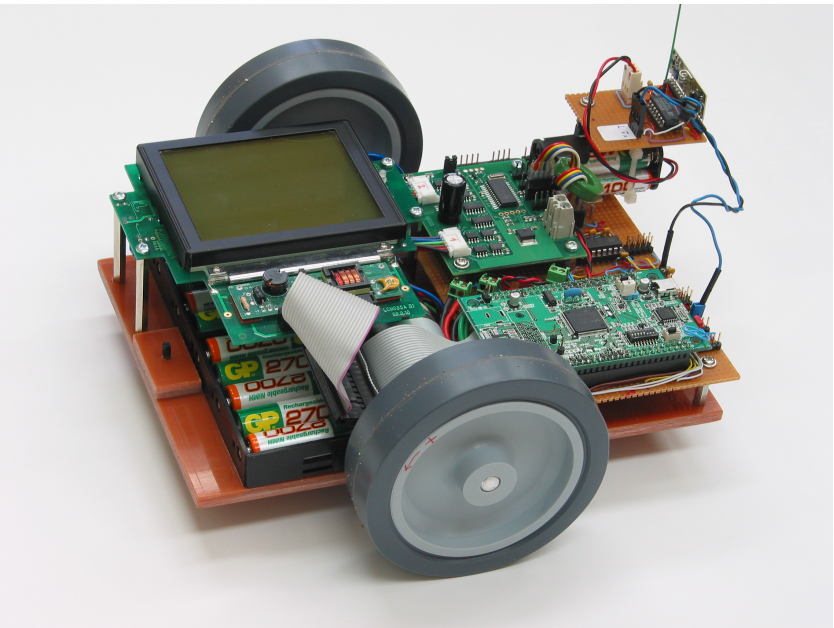
Cílem této pracé je poskytnout uživateli grafické uživatelské prostředí, kterým bude možné ovládat pohyb robotického vozítka evokovanými potenciály. Pro zajištění veškeré funkčnosti je zapotřebí zajistit několik důležitých požadavků. Nejdůležitější z nich jsou komunikace s měřicím počítačem a s ovládaným vozítkem, poskytnutí blikajících obrazců a zobrazení vizuální zpětné vazby z kamery umístěné na vozítku. Propojení této aplikace s vozítkem bude sprostředkováno pomocí bezdrátové technologie Bluetooth, aby nebylo vozítko při svém pohybu omezováno délkou komunikačního kabelu.

Celý projekt má výzkumný charakter a měl by pomoci k usnadnění výzkumu v oblasti Brain-Computer Interface na Západočeské Univerzitě v Plzni a jeho využití pro možné ovládání různých zařízení.

2 Robotické vozítko

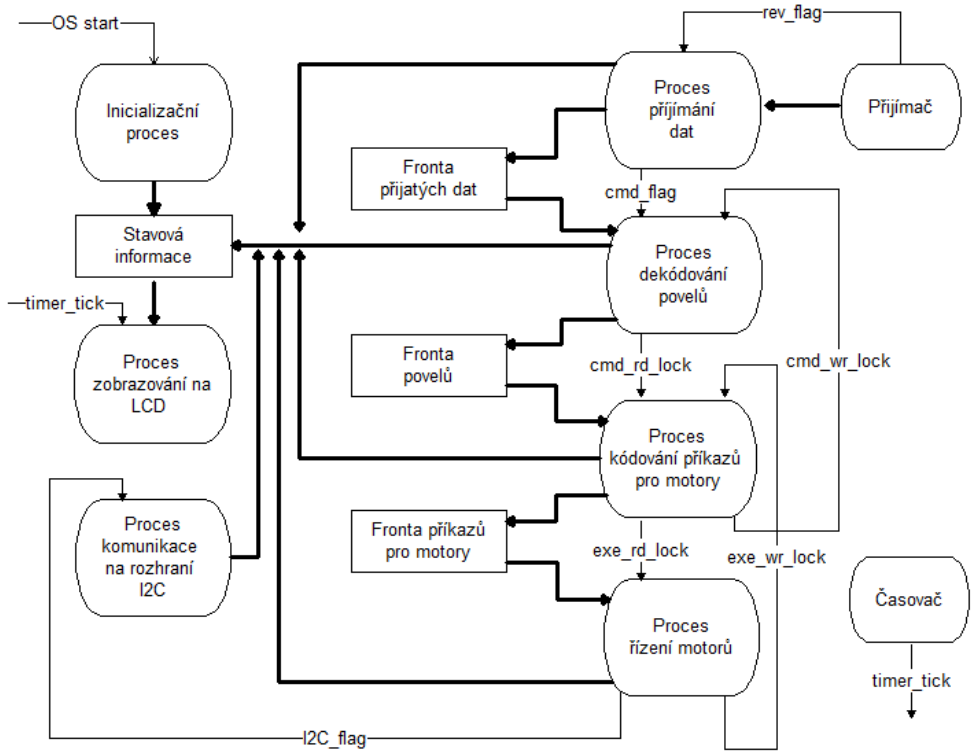
2.1 Architektura robotického vozítka

Robotické vozítko, navržené na KIV, je složeno z 32-bitového procesoru NEC V850ES/HG2, LCD displeje, obvodu pro zajištění komunikace pomocí technologie Bluetooth a modulu, který zajišťuje řízení motorů. Přijímání signálů probíhá pomocí radiofrekvenční komunikace RFCOMM, která poskytuje binární přenos dat a emuluje řídící signály protokolu RS-232 ve vrstvě Bluetooth pásma. Přijímání a zpracovávání řídících signálů zajišťuje software bežící pod operačním systémem MicroC/OS-II.



Obrázek 2.1: Robotické vozítko.

Nejprve přijme data, která uloží do fronty přijatých dat. Poté je dekóduje a uloží do fronty povelů. Tato data se následně zakódují na příkazy pro motory a uloží se do další fronty. Z té si je vybírá proces, který řídí motory vozítka. Podrobné schéma všech procesů a řídících povelů viz Obrázek 2.2.



Obrázek 2.2: Schéma procesů a povelů operačního systému MicroC/OS-II.

2.2 Řídící povely robotického vozítka

Přijímač na vozítku je schopný rozpoznat pouze řídící povely zadané v předepsaném formátu.

2.2.1 Formát povelu

Příklad povelu:
`Cxxxx, xxxx, ..., xxxx, S<CR><LF>`

C	kód povelu
xxxx	parametry
S	kontrolní součet včetně kódu povelu

2.2.2 Kontrolní součet

Pole pro výpočet kontrolního součtu:
`Cxxxx, xxxx, ..., xxxx,` S<CR><LF>

Výpočet kontrolního součtu:
`S = byte_1` $\oplus$ `byte_2` $\oplus$ `byte_3` $\oplus$ `...` $\oplus$ `byte_n`

2.3 Řídící povely

2.3.1 Přehled povelů

Kód	Formát	Význam
P	PS	Stop
D	D±lll, ±rrr, S	Přímé řízení motorů
G	G±fff, ±ddd, S	Přímá jízda se zadáním relativního úhlu
F	F±ddddd, S	Přímá jízda bez změny směru
T	T±fff, S	Otočení o zadaný úhel
W	W±nnnnn, S	Čekání nnnnn ms
C	C±n, S	Zapnutí vypnutí kontroly kontrolního součtu

2.3.2 Popis povelů

U většiny povelů je nutné zadat parametr, který je povolen vždy jen v určitém rozsahu. Z tohoto důvodu je níže uveden podrobnější popis řídících povelů.

P **Stop**

Parametry -

Zastavení a čekání na další povel.

D **Přímé řízení motorů**

Parametry	±lll	rychlost levého kola	rozsah: <-128; +127>
	±rrr	rychlost pravého kola	rozsah: <-128; +127>

Přímo ovládá rychlost levého a pravého kola. - 128 = maximální rychlost v záporném směru, +127 = maximální rychlost v kladném směru, 0 = stop.

G **Přímá jízda se zadáním relativního úhlu**

Parametry	±lll	relativní úhel otočení	rozsah: <-180; +180>
	±ddddd	vzdálenost v mm	rozsah: <0; 65535>

Natočí vozítko o zadaný úhel a ujede zadanou vzdálenost.

F **Přímá jízda bez změny směru**

Parametry	±ddddd	vzdálenost v mm	rozsah: <0; 65535>

Ujede zadanou vzdálenost.

T **Otočení o zadaný úhel**

Parametry	±fff	úhel otočení	rozsah: <-180; +180>

Otočení na místě o zadaný úhel.

W **Čekání**

Parametry	±nnnnn	doba čekání v ms	rozsah: <0; 65535>

Čeká zadanou dobu před provedením dalšího povelu.

C **Zapnutí / vypnutí kontrolního součtu**

Parametry	±n	0 = vypnuto, 1 = zapnuto	rozsah: 0, 1

Zapne / vypne kontrolu kontrolního součtu.

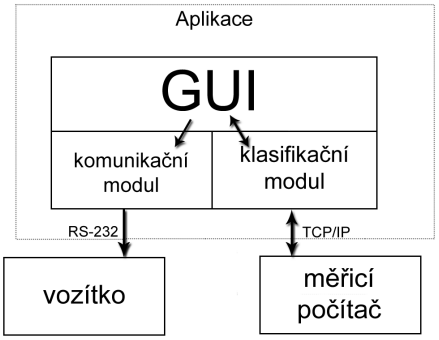
3 Grafické uživatelské rozhraní

Účelem tohoto systému je poskytnutí grafického uživatelského rozhrani (dále jen GUI), které umožňuje ovládání pohybu vozítka pomocí mozkových vln. Aplikace bude pracovat ve čtyřech uživatelských režimech. Komunikace s přístrojem pro měření mozkové aktivity bude zajištěna pomocí protokolu TCP/IP. Samotná komunikace s vozítkem bude probíhat přes rozhraní sériového portu (dále jen RS-232) podle definovaných signálů. Aplikace bude zajišťovat vizuální zpětnou vazbu, v podobě obrazu z IP kamery umístěné na vozítku.

3.1 Obecný popis

3.1.1 Kontext systému

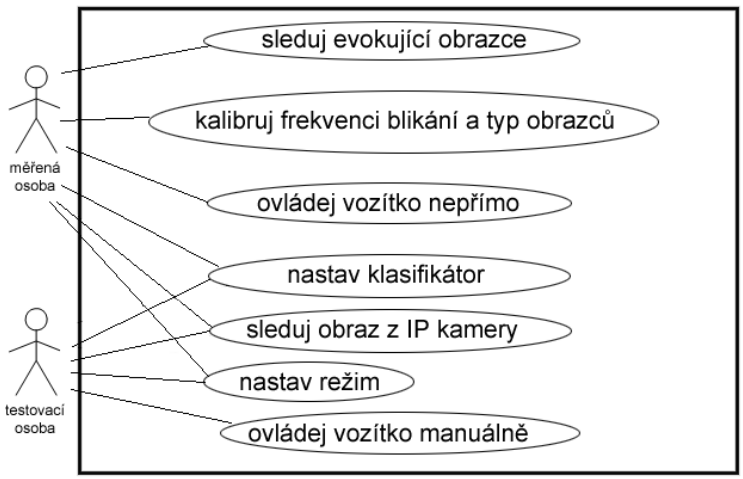
V současné době se vozítko ovládá za pomoci helmy s akcelerometry. Tato aplikace má do budoucna za úkol zcela nahradit stávající ovládání za ovládání pomocí mozkových vln (BCI) založeném na ustálených evokovaných potenciálech. Na níže uvedeném obrázku je zobrazen blokový návrh aplikace. Bude obsahovat komunikační modul, který zajistí komunikaci pomocí RS-232 s ovládaným vozítkem a bude mu zasílat ovládací signály (viz kapitola 2.3.1 Přehled povelů). Klasifikační modul komunikuje s měřicím počítačem pomocí protokolu TCP/IP. Tento modul byl vyvinut již dříve a do této aplikace bude přidán jako hotový a fungující prvek.



Obrázek 3.1: Blokový diagram navrhované aplikace.

Úkolem aplikace je poskytnout uživateli GUI s požadovanými vlastnostmi, pracující ve čtyřech různých módech (viz kapitola 3.2 Funkce systému). Dále by měla zajistit vyvolání vizuálních evokovaných potenciálů u měřené osoby za pomocí kontinuálně se střídajících obrazců danou frekvencí. Podle výsledků, vyhodnocených klasifikátorem, bude ovládat robotické vozítko přes RS-232 pomocí definovaných řídících signálů (viz kapitola 2.3.1 Přehled povelů). Během řízení vozítka bude zobrazovat uživateli obraz z IP kamery umístěné na vozítku. S přístrojem pro záznam mozkové aktivity naváže komunikaci pomocí protokolu TCP/IP a bude od něho přijímat naměřená data.

Use-case diagram této aplikace ukazuje, že jsou dva druhy uživatelů. Jeden druh je **testovací osoba**, která může ovládat vozítko manuálně (klavesnicí nebo myší v Manual módu) a zároveň sledovat obraz z IP kamery umístěné na vozítku. Druhý druh uživatele je **měřená osoba**. U této osoby budou při sledování blikajících obrazců vyvolány evokované potenciály, kterými bude moci ovládat vozítko "nepřímo"tzn. pomocí mozkových vln a zároveň sledovat obraz z IP kamery umístěné na vozítku (Master mód). Měřená osoba bude moci také kalibrovat frekvenci a typ blikajících obrazců na základě vizuální zpětné vazby v podobě pohybující se "kuličky"(Marble mód). Při použití přídavných blikajících LED diod k vyvolání evokovaných potenciálů bude moci měřená osoba použít ke sledovaní obrazu z IP kamery čtvrtý mód aplikace a tím je OnlyVideo mód. Samotný use-case diagram navrhované aplikace je znázorněn na Obr. 3.2 .



Obrázek 3.2: Use-case diagram navrhované aplikace.

Aplikace poběží na počítači umístěném v měřicí laboratoři (UU-403), na operačním systému MS Windows XP, Vista či 7. Aby aplikace běžela plynule a správně musí tento počítač splňovat minimální požadavky:

- Procesor 1,5 GHz.
- 1024 MB operační paměti.
- Monitor s minimálním rozlišením 800x600 pixelů.
- Přístup do lokální sítě pro komunikaci s měřicím počítačem.
- USB port pro komunikaci s vozítkem pomocí emulovaného sériového portu.

Aplikace dále vyžaduje pro svůj běh komponenty, které je nutné stáhnout a nainstalovat do počítače, na kterém tato aplikace poběží. Jsou to **Java Runtime Enviroment** (JRE) verze 6 [5] a knihovnu **Java Media Framework** (JMF) verze 2.1.1e [6] nebo její alternativu **Freedom for Multimedia Framework** (FMJ) minimální verze 20070928-0938 [7].

3.2 Funkce systému

3.2.1 Pracovní režimy

Aplikace umožní uživateli pracovat ve 4 různých režimech (Master mód, Marble mód, Manual mód a OnlyVideo mód) a umožní mu mezi těmito režimy volit. Uživatel si zvolí pomocí ovládacích prvků GUI v jakém režimu chce pracovat a aplikace se poté přepne do příslušného režimu.

- **Master Mode** - Tento režim umožní uživateli ovládat vozítko pomocí jeho změřených reakcí na blikající obrazce. Uživatel sleduje blikající obrazce a jeho reakce jsou přijímány pomocí TCP/IP z měřicího přístroje a vyhodnocovány klasifikátorem. Na základě těchto výsledků jsou zasílany ovládací signály vozítku pomocí RS-232. Dále je uživateli zprostředkován obraz z IP kamery umístěné na vozítku.

- **Marble Mode** - Režim umožňující uživateli kalibraci frekvence a typu blikajících obrazců na základě pohybujícího se kurzoru (kuličky). Uživatel sleduje blikající obrazce. Jeho reakce jsou přijímány pomocí TCP/IP z měřicího přístroje a vyhodnocovány klasifikátorem. Výsledky klasifikace jsou uživateli prezentovány pohybem kurzoru (kuličky) v GUI.

- **Manual Mode** - V tomto režimu bude moci uživatel přímo ovládat pomocí šipek na klávesnici nebo tlačítek v GUI pohyb vozítka. Dále mu bude zprostředkován obraz z IP kamery umístěné na vozítku. Vozítku budou na základě vstupu od uživatele zasílány ovládací signály (viz Kapitola 2.3 Řídící povely) přes RS-232.

- **OnlyVideo Mode** - Tento režim bude sloužit pro sledování videa z IP kamery na vozítku při použití externích blikajících segmentů. Tyto segmenty nejsou součástí aplikace a jsou nezávislé na počítači, na kterém je tato aplikace spuštěna.

3.2.2 Další funkce aplikace

- **Zobrazení obrazu z IP kamery** - Aplikace bude schopna přijímat data z IP kamery umístěné vozítku a zobrazovat je uživateli (Master mód, Manual mód). Uživatel si v okně konfigurace zvolí IP adresu kamery a protokol přenosu. V panelu aplikace, který bude mít volitelnou velikost, mu bude zobrazen obraz z kamery.

- **Komunikace s vozítkem přes RS-232** - Aplikace bude schopna zasílat vozítku ovládací příkazy přes rozhraní sériového portu RS-232. Na základě výsledků klasifikátoru(Master mód) nebo vstupu od uživatele (Manual mód) aplikace zašle předem definované signály (viz Kapitola 2.3 Řídící povely) vozítku.

- **Evokující obrazce** - Uživatel si vybere počet obrázků, jejich typ a frekvenci jejich blikání. Tyto obrazce budou zobrazeny uživateli v definovaných umístěních. Každý obrazec bude mít možnost vlastní konfigurace zobrazovaného obrázku, frekvence problikávání, pevné velikosti

a typu problikávání (s bílým pozadím / inverzní obrazec). Uživatel si bude moci rozhodnout o počtu zobrazených obrazců (horní + dolní / horní + dolní + levý + pravý). Aplikace bude zobrazovat 2 nebo 4 evokující obrazce blikající zvolenou frekvencí umístěné v levé, pravé, horní a dolní části GUI.

- **TCP/IP komunikace s měřicím přístrojem** - Aplikace bude umožňovat komunikaci s měřicím počítačem pomocí TCP/IP. Serverová část komunikačního rozhraní bude umístěna v měřicím počítači a klientská část bude zakomponována v aplikaci. V parametrech klasifikátoru uživatel nastaví IP adresu serveru a port, na kterém bude přijímat data od měřicího počítače.

- **Fullscreen režim aplikace** - Aplikace bude schopna běžet v celoobrazovkovém režimu (fullscreen). Uživatel si bude moci ovládacím prvkem v GUI přepnout aplikaci do fullscreen režimu nebo naopak.

3.3 Požadavky na vnější rozhraní

3.3.1 Uživatelská rozhraní

GUI bude vytvořeno za pomoci knihovny SWING a bude mít defaultní LOOK & Feel této knihovny, který je platformově nezávislý. GUI bude mít obvyklou strukturu - bude obsahovat:

- Hlavní menu obsahující všechny volby ovládání aplikace.
- Nástrojovou lištu (Toolbar), na které budou nejpoužívanější volby aplikace.
- Konfigurační dialog umožňující konfiguraci aplikace.

Nejpoužívanější volby aplikace budou zpřístupněny pomocí klávesových zkratek. Aplikace umožní přepnutí do režimu fullscreen. Nejpoužívanější volby budou zvoleny ve spolupráci se zadavatelem v průběhu vývoje aplikace.

3.3.2 Hardwarová rozhraní

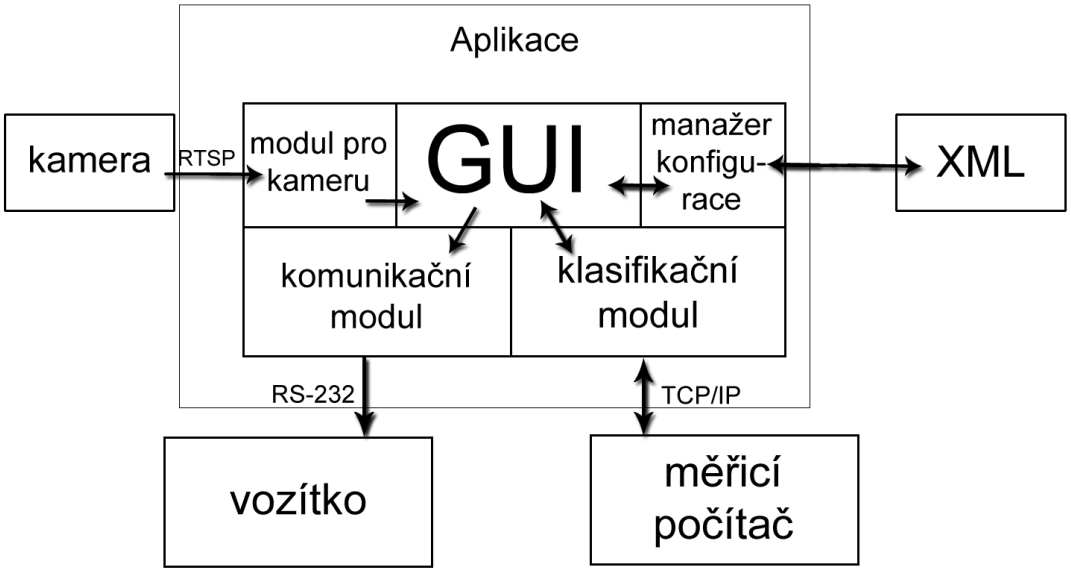
Aplikace bude používat následující rozhraní a standardy:

- IEEE 802.11 (Wi-Fi) nebo IEEE 802.3 (Ethernet) – pro komunikaci mezi aplikací a měřicím počítačem pomocí protokolu TCP/IP.
- RS-232 (Seriový port) – pro komunikaci aplikace s ovládaným vozítkem. Toto rozhraní bude emulované a realizované pomocí technologie BlueTooth.
- IEEE 802.11 (Wi-Fi) – pro bezdrátový přenos video signálu z IP kamery na ovládaném vozítku do aplikace pomocí protokolů RTP/RTSP.

4 Implementace

4.1 Architektura systému, přehled podsystémů

Jako architektura aplikace byla zvolena MVC architektura (Model - View - Controller). Celá aplikace byla rozdělena do několika samostatných podsystémů, které mají svůj specifický účel. Tyto podsystémy jsou centrálně ovládány z grafického uživatelského okna aplikace.



Obrázek 4.1: Blokový diagram aplikace.

Podsystém pro zasílání řídících signálů vozítku

- Umožňuje posílání řídících signálů vozítku přes protokol RS-232.
- Komunikace podsystému s vozítkem je jednosměrná - od aplikace k vozítku.

Podsystém pro zobrazení obrazu z IP kamery

- Zajišťuje příjem videosignálu z IP kamery pomocí protokolu RTP/RTSP.

- Zprostředkovává uživateli vizuální informaci z kamery.

Podsystém pro ukládání konfigurace aplikace

- Poskytuje datové struktury pro uchování aktuální konfigurace aplikace.
- Zajišťuje ukládání a načítání konfiguračních dat z XML souboru.
- Umožňuje konfigurační data zobrazit v dialogovém okně a nechat je změnit.

Podsystém klasifikace a TCP/IP komunikace

- Pomocí Fourierovy transformace zjišťuje četnost výskytu jednotlivých frekvencí.
- Vyhodnocuje výsledky Fourierovy transformace.
- Zajišťuje komunikace aplikace s měřicím přístrojem pomocí protokolu TCP/IP.

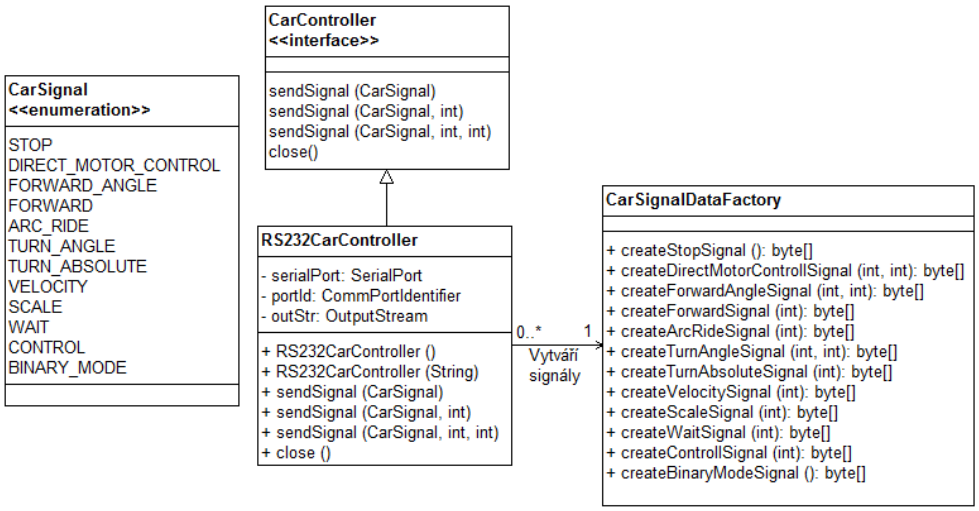
4.2 Zvolené technologie a programovací jazyk

Jako hlavni programovací jazyk je zvolena Java z důvodu rychlého a efektivního vývoje a možnosti programovat objektově. Kvůli jednoduchému zpracování XML souboru a jeho propojení s třídami jazyka Java je k ukládání konfigurace aplikace použita technologie JAXB. Knihovna JMF (FMJ) se využívá pro příjem a zobrazování videa z IP kamery pomocí protokolu RTP/RTSP. K zajištění komunikace s vozítkem pomocí protokolu RS-232 je využita knihovna RXTX. Je to knihovna jazyka Java implementující Java Communication API. Pro rychlé, přesné a plynulé překreslování blikajících obrazců je použita knihovna OpenGL. Jedná se o knihovnu pro zobrazování akcelerované počítačové grafiky.

4.3 Podsystémy

4.3.1 Podsystém pro zasílání ovládacích signálů vozítku

Účelem podsystému je zajistit odesílání ovládacích signálů vozítku pomocí sériového portu (pomocí protokolu RS-232). Tato komunikace je jednosměrná, podsystém nemá od vozítka žádnou zpětnou vazbu.



Obrázek 4.2: UML diagram podsystému pro zasílání ovládacích signálů vozítku.

Výčtový typ `CarSignal` udává množinu všech typů signálů, které lze vozítku poslat. Obsahuje i signály, které prozatím vozítko nepodporuje, protože jejich přidání do základní sady povelů je ve fázi vývoje. Tyto, prozatím nefunkční signály, jsou v níže uvedeném seznamu zvýrazněny kurzívou.

Hodnoty:

STOP - Zastavení vozítka.
DIRECT_MOTOR_CONTROL - Přímé řízení motorů.
FORWARD_ANGLE - Přímá jízda se zadáním relativního úhlu směru.
FORWARD - Přímá jízda bez změny směru.
TURN_ANGLE - Otočení o zadaný úhel.
WAIT - Čekání x milisekund.
CONTROL - Zapnutí / vypnutí kontroly kontrolního součtu.

ARC_RIDE - Jízda v oblouku.

TURN_ABSOLUTE - Otočení do absolutní polohy.
VELOCITY - nastavení rychlosti pohybu.
SCALE - Nastavení měřítka vzdálenosti.
BINARY_MODE - Přepnutí do binárního módu.

Rozhraní `CarController` obsahuje množinu metod pro zasílání ovládacích signálů vozítku spolu s různými daty. Konkrétně se jedná o metodu `sendSignal(CarSignal)`, která odešle vozítku signál předaný v parametru. Spolu s ním lze odeslat jednu nebo dvě číselné hodnoty. Tuto vlastnost zajišťují metody `sendSignal(CarSignal, int)` a `sendSignal(CarSignal, int, int)`. Rozhraní obsahuje také metodu `close()` pro ukončení spojení s vozítkem.

Třída `RS232CarController` implementuje rozhraní CarController. Zajišťuje komunikaci s vozítkem přes sériový port a ke své funkčnosti využívá knihovnu RXTX. Třída obsahuje dva konstruktory. Jeden implicitní `RS232CarController()`, kterým se připojí na port označený "COM1" (v GNU/Linuxu na "/dev/ttyS0") a druhý `RS232CarController(String)`, kterým se připojí na port zadaný v parametru. Seznam veřejných metod a jejich účel je stejný jako u rozhraní `CarController`.

Třída `CarSignalDataFactory` představuje tovární objekt pro vytváření signálů v podobě sekvence bytu v definovaném formátu. Tato třída odstiňuje logiku vytváření binárních signálů od třídy RS232CarController. Tyto signály vytváří pomocí metod:

`createStopSignal (): byte[]` - Vytvoří signál pro zastavení vozítka.
`createDirectMotorControllSignal (int, int): byte[]` - Vytvoří signál pro přímé řízení motorů v podobě sekvence bytů.
`createForwardAngleSignal (int, int): byte[]` - Vytvoří signál pro přímou jízdu se zadáním relativního úhlu v podobě sekvence bytů.
`createForwardSignal (int): byte[]` - Vytvoří signál pro přímou jízdu beze změny směru v podobě sekvence bytů.
`createArcRideSignal (int): byte[]` - Vytvoří signál pro jízdu v oblouku v podobě sekvence bytů.
`createTurnAngleSignal (int, int): byte[]` - Vytvoří signál pro otočení vozítka o zadaný úhel v podobě sekvence bytů.
`createTurnAbsoluteSignal (int): byte[]` - Vytvoří signál pro otočení vozítka do absolutní polohy v podobě sekvence bytů.
`createVelocitySignal (int): byte[]` - Vytvoří signál pro nastavení rychlosti vozítka v podobě sekvence bytů.

`createScaleSignal (int): byte[]` - Vytvoří signál pro nastavení měřítka vzdálenosti v podobě sekvence bytů.

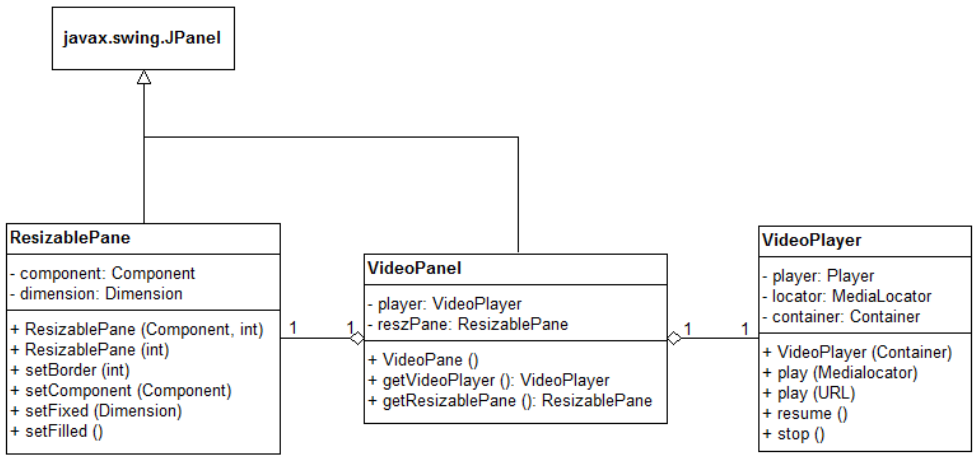
`createWaitSignal (int): byte[]` - Vytvoří signál pro čekání vozítka x milisekund v podobě sekvence bytů.

`createControllSignal (int): byte[]` - Vytvoří signál pro zapnutí/vypnutí kontroly k.s. v podobě sekvence bytů.

`createBinaryModeSignal (): byte[]` - Vytvoří signál pro přepnutí do binárního módu v podobě sekvence bytů.

4.3.2 Podsystém pro zobrazení obrazu z IP kamery

Účelem podsystému je získat data z IP kamery pomocí protokolu RTP/RTSP a zobrazit je uživateli. Podsystém také umožňoje nastavení velikosti obrazu videa a výběr jeho zdroje (adresy IP kamery).



Obrázek 4.3: UML diagram podsystému pro zobrazení obrazu z IP kamery.

Třída `VideoPlayer` je určená pro získání dat z IP kamery a jejich zobrazení na příslušné komponentě z knihovny Java Swing. V době, kdy není video přehráváno zobrazuje černou obrazovku. Třída využívá pro práci s videem knihovnu JMF (FMJ). Konstruktor `VideoPlayer(Container)` vytvoří instanci přehrávače. Parametr udává kontejner (z knihovny Java AWT), do kterého bude vložena komponenta pro zobrazení videa. Instance používá metody:

`play (MediaLocator)` - Spustí přehrávání videa ze zdroje udaném pomocí instance MediaLocator v parametru.

`play (URL)` - Spustí přehrávání videa ze zdroje udaném pomocí url v parametru.
`resume ( )` - Obnoví přehrávání videa z posledního dostupného zdroje.
`stop ( )` - Zastaví přehrávání videa.

Třída `ResizablePane` představuje grafický kontejner pro jednu komponentu. Umožňuje nastavovat buď pevnou velikost této vnitřní komponenty nebo ji nechat vyplnit celý prostor kontejneru. Třída obsahuje konstruktory `ResizablePane(Component, int)` a `ResizablePane(int)`. První z nich vytvoří kontejner a umístí dovnitř komponentu, předanou v parametru. Druhý parametr udává velikost mezery mezi komponentou uvnitř a okrajem (v režimu vyplnění). Druhy konstruktor vytvoří prázdný kontejner. Parametr udává velikost mezery mezi komponentou uvnitř a okrajem (v režimu vyplnění). Tato třída obsahuje metody:

`setBorder (int)` - Nastaví velikost mezery mezi komponentou uvnitř a okrajem (v režimu vyplnění).
`setComponent (Component)` - Nastaví komponentu uvnitř kontejneru.
`setFixed (Dimension dimension)` - Přepne do režimu zobrazení vnitřní komponenty, tak aby měla pevnou velikost udanou v parametru.
`setFilled ()` - Přepne do režimu zobrazení vnitřní komponenty, tak aby vyplňovala celý prostor kontejneru.

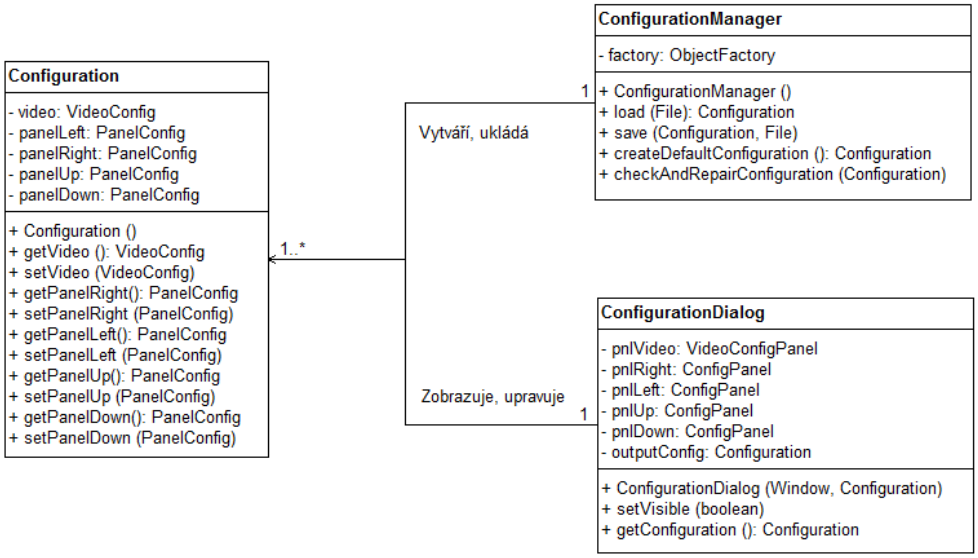
Třída `VideoPanel` slouží jako hlavní modul zajišťující přehrávání videa, určený pro vložení do hlavního okna aplikace. Obsahuje komponentu, na kterou zobrazuje třída VideoPlayer obraz videa a umožňuje nastavení její velikosti pomocí třídy ResizablePane. Obsahuje metody:

`getVideoPlayer (): VideoPlayer` - Vrátí objekt přehrávače určený pro přehrávání videa na panelu.
`getResizablePane (): ResizablePane` - Vrátí objekt určený ke změně velikosti obrazu přehrávání videa.

4.3.3 Podsystém pro ukládání konfigurace aplikace

Účelem podsystému je zajistit ukádání a načítání údajů z konfiguračního XML souboru a jejich uchování ve formě objektů pro zpracování jinými pod-

systémy aplikace. Další úlohou podsystému je umožnit zobrazení této konfigurace uživateli v podobě dialogového okna a umožnit i změnu konfigurace. Součástí podsystému je i množina automaticky vygenerovaných tříd pomocí technologie JAXB ze schéma XSD, popisující XML konfigurační soubor (patří k nim i třída `Configuration`).



Obrázek 4.4: UML diagram podsystému pro ukládání konfigurace aplikace.

Třída `Configuration` je určená pro uchování konfigurace aplikace. Obsahuje v sobě jako atributy další třídy uchovávající konkrétní části konfigurace. Tyto ostatní třídy zde nejsou kvůli rozsahu a přehlednosti uvedeny. Slouží pouze k uchování dat a jejich nastavení nebo vrácení. Tato třída obsahuje metody:

`getVideo (): VideoConfig` - Vrátí objekt konfigurace videa.
`setVideo (VideoConfig)` - Nastaví objekt konfigurace videa.
`getPanelLeft (): PanelConfig` - Vrátí objekt konfigurace levého panelu.
`setPanelLeft (PanelConfig)` - Nastaví objekt konfigurace levého panelu.

Obdobně jako předchozí 2 metody obsahuje třída také metody pro konfiguraci pravého, horního a dolního panelu. (viz Obrázek 4.4)

Třída `ConfigurationManager` slouží pro načítání konfigurace z XML souboru a jejímu převodu na instanci třídy Configuration, či uložení tohoto konfiguračního objektu zpět do souboru. Třída umí vytvořit také objekt standardní konfigurace nebo opravit špatná konfigurační data. K provádění těchto operací využívá metody:

`load (File): Configuration` - Načte konfiguraci ze souboru, udaném v parametru a převede jí na objekt konfigurace.
`save (Configuration, File)` - Načte konfiguraci ze souboru, udaném v parametru a převede jí na objekt konfigurace.
`createDefaultConfiguration (): Configuration` - Vytvoří konfigurační objekt obsahující standardní nastavení aplikace.
`checkAndReapirConfiguration (Configuration)` - Opraví chybějící údaje nebo špatně nastavené hodnoty v konfiguračním objektu.

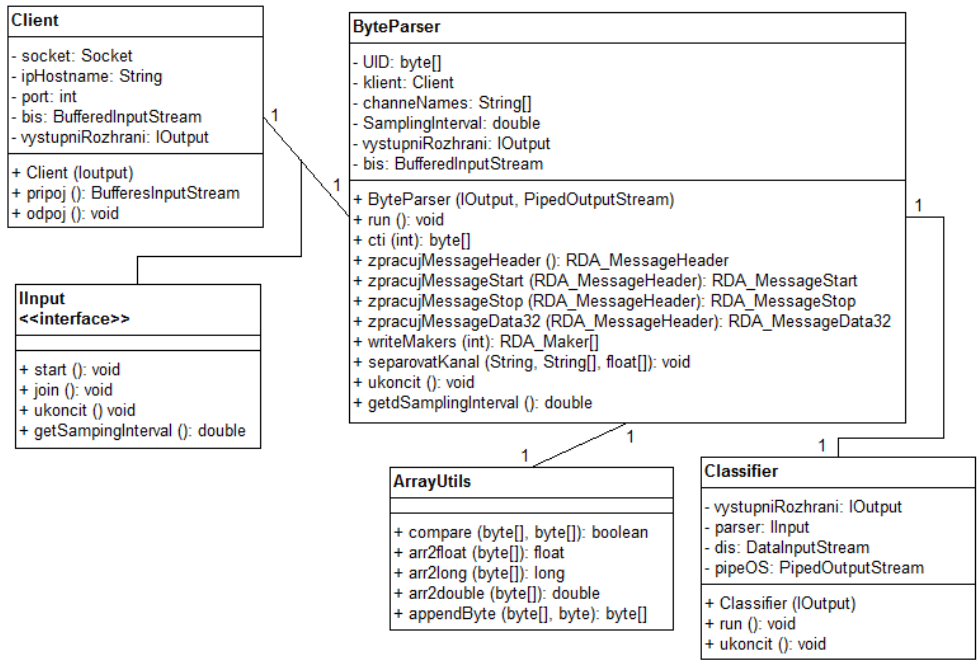
Třída `ConfigurationDialog` slouží pro zobrazení konfigurační třídy Configuration v podobně grafického dialogového okna. Toto dialogové okno se skládá z dalších tříd (přibližně 8), sloužících pro zobrazení jednotlivých konfiguračních dat. Tyto třídy zde nejsou z důvodu rozsahu a přehlednosti uvedeny. Obsahuje jeden konstruktor `ConfigurationDialog (Window, Configuration)`, který vytvoří konfigurační dialog příslušící danému oknu aplikace (1. parametr) zobrazující danou konfiguraci (2. parametr). Tato třída obsahuje metody:

`setVisible(boolean)` - Nastaví zda bude dialogové okno viditelné.
`getConfiguration ( ): Configuration` - Vrátí objekt konfigurace nastavené na prvcích dialogového okna.

4.3.4 Podsystém klasifikace a komunikace TCP/IP

Podsystem nevyužívá žádných vstupních souborů, ale zpracovává data zasílána pomocí protokolu TCP/IP aplikací Brain Vision Recorder. Tato data jsou z proudu bytů transformována do objeků RDA_MessageHeader, RDA_MessageStart, RDA_MessageData32, RDA_MessageStop popř. RDA_Marker. Zpracované objekty obsahují informace o hodnotách naměřených na jednotlivých elektrodách měřicího zařízení, počet a názvy kanálů, použitou vzorkovací frekvenci,... Samotné naměřené hodnoty z jednotlivých

elektrod jsou obsaženy v objektu RDA_MessageData v poli fData[ ] typu float.



Obrázek 4.5: UML diagram podsystému klasifikace a komunikace TCP/IP.

Třída `Client` naváže spojení se serverem a předá proud se vstupními daty (byty) instanci třídy ByteParser. Její konstruktor vytvoří instanci klienta, jejímž vstupním parametrem je výchozí rozhraní pro získávání informací od uživatele a výpis stavových informacích do konzole. Instance této třídy používá metody:

`pripoj ()` - Naváže spojení se serverem na adrese a portu získaného z výstupního rozhraní.
`odpoj ()` - Bezpečně ukončí spojení se serverem.

Třída `ByteParser` slouží ke zpracování bytů přicházejících ze serveru. Zpracovaní bytů se provádí vyhledáváním unikátní posloupnosti 16 bytů (konstanta UID), která označuje hlavičku objektu ve vstupním streamu. Následující 4 byty (načítané do proměnné nSize objektu MessageHeader) představují celkovou velikost daného bloku a další 4 byty (načítané do proměnné nType objektu MessageHeader) identifikují jeden ze 4 možných druhů příchozích zpráv. Na základě údajů v hlavičce je následně zpracován celý příchozí

objekt. Třída implementuje rozhraní IInput. Konstruktor vytvoří instanci ByteParseru, která zajistí zdroj vstupních dat (pomocí třídy Client). Výstup ByteParseru (=naměřené hodnoty z určitého kanálu) je zapisován do výstupního streamu (pipe), který je předán jako parametr. Dalším parametrem konstruktoru je odkaz na výstupní rozhraní (např. pro výpisy do konzole). Třída obsahuje metody:

`run ()` - Slouží ke spuštění vlákna = zahájení zpracování vstupních dat (bytů).
`ukoncit ()` - Slouží k ukončení zpracovávání vstupních dat (bytů).
`cti (int)` - Slouží k načtení požadovaného počtu bytů ze vstupního streamu.
`getdSamplingInterval ()` - Vrací vzorkovací interval (frekvenci) získanou zpracováním objektu RDA_MessageStart.
`zpracujMessageHeader ()` - Zpracuje aktuálně příchozí data ze vstupního streamu do objektu.
`zpracujMessageStart (RDA_MessageHeader)` - Zpracuje aktuálně příchozí data ze vstupního streamu do objektu.
`zpracujMessageData32 (RDA_MessageHeader)` - Zpracuje aktuálně příchozí data ze vstupního streamu v objektu MessageData.
`preskocMarkery (int)` - Přeskočí všechny byty související s objektem typu RDA_Marker - pro tuto úlohu nepotřebné.
`separovatKanal (string, string[])` - Vyseparuje data jen z jednoho kanálu a zapíše je do výstupního streamu.

Třída `ArrayUtils` slouží pro práci s datovými typy. Převádí pole bytů na float, long nebo double a porovnává shodnost dvou takových polí. Třída nemá konstruktor, protože obsahuje pouze statické metody:

`compare (byte[], byte[])` - Zjišťuje, jestli jsou dvě pole bytů shodná.
`arr2float (byte[])` - Převádí pole čtyř bytů na float.
`arr2long (byte[])` - Převádí pole čtyř bytů na long.
`arr2double (byte[])` - Převádí pole osmi bytů na double.
`appendByte (byte[], byte)` - Na konec pole přidá nový byte a posune index o + 1.

Třída `Classiffier` rozpoznává frekvence zadané ve vstupním signálu. Do pole se načítají jednotlivé hodnoty signálu, přetypované na typ double. Jakmile se pole naplní, použije se Fourierova transformace a vyhodnotí se výskyt

jednotlivých frekvencí. Konstruktor vytvoří instanci klasifikátoru, vytvoří vstupní a výstupní data stream a zavolá konstruktor ByteParseru. Výstup klasifikátoru, naměřené hodnoty z určitého kanálu spolu s příznakem rozpoznané frekvence, je zapisován do výstupního streamu. Třída obsahuje metody:

`run ()` - Spustí vlákno pro zahájení rozpoznávání frekvencí v signálu.
`ukoncit ()` - Slouží k ukončení rozpoznávání frekvencí v signálu.
`zavriVystupniStream ()` - Ukončí zápis do výstupního souboru.
`getIndexOfMaxValue (double[])` - Vrací index, na kterém se nachází prvek s maximální hodnotou.
`getMaxValue (double[])` - Vyhledává maximum v poli na daném indexu +- dané okolí.
`setKlasifikovat (boolean)` - Spustí klasifikaci.

5 Testování

Testování aplikace probíhalo ve dvou fázích. Nejprve probíhala řada testů se zkušebními daty, které byly naměřeny v průběhu vývoje klasifikátoru EEG signálů. Měření bylo provedeno na třech studentech, z nichž dvě byly ženy a jeden muž. Při těchto měřeních byly k vyvolání evokovaných potenciálů použity blikající LED diody. Každý člověk reaguje na dané frekvence jinak, ale po několika měřeních se podařilo najít u každého frekvence, na které reaguje nejlépe.

Druhá fáze testování byla napojení náhodně vybrané studentky přímo na tuto aplikaci, aby ovládala vozítko v reálném čase. K vyvolání evokovaných potenciálů byly použity blikající obrazce, které poskytuje tato aplikace. Během krátké doby byly zjišťeny frekvence, na které reagovala velmi dobře i ty, na které byly odezvy nedostatečné.

Při testování se ani v jedné fázi nepoužíval Master mód, tedy reálné ovládání vozítka, ale Marble mód, ve kterém bylo na pohybu "kuličky"dobře vidět, že klasifikátor není na příliš vysoké úrovni. Proto se kvůli obavám o poškození vozítka nepoužil Master mód. Další důvod k nepoužití Master módu byl že k vyvolání evokovaných potenciálů byla použita tato aplikace a blikající obrazce se navzájem interferovaly. Z tohoto důvody se snižil počet blikajících obrazců ze čtyř na dva (horní a dolní) ale stejně nebyl výsledný pohyb kuličky příliš uspokojivý. Použití blikajících LED diod nebylo možné, protože uzavřená kabina pro měřenou osobu není prozatím uzpůsobena k vložení diodových blikačů dovnitř této kabiny. Jako poslední věc se otestovala komunikace a zasílání řídících signálů vozítku. Použitím Manual módu proběhlo bezchybné ovládání vozítka.

Co se týká aplikace, tak při obou fázích testování pracovala spolehlivě a bez chyb. Nebyly problémy s komunikacemi (TCP/IP, RS-232), zobrazováním videa z IP kamery, nastavováním aplikace ani se zobrazováním evokujících obrazců.

6 Závěr

Cílem této práce bylo navrhnout a implementovat grafické uživatelské prostředí pro ovládání robotického vozítka mozkovými vlnami. Výsledkem je aplikace s názvem WH Smart Controller. Aplikace je funkční a byla testována, jak zkušebními daty, tak i přímým připojením s relativně dobrými výsledky. Více informaci ohledně testování obsahuje kapitola 5 zabývající se tímto tématem.

Nevýhodou této aplikace je, že poskytuje problikávající obrazce o frekvencích 5 - 50 Hz pro vyvolání vizuálních evokovaných potenciálů. Tyto stroboskopické efekty mohou vyvolat záchvaty u osob trpících epilepsií či jinou podobnou chorobou. Těmto osobám je doporučeno nesledovat grafický výstup aplikace. Uvedené efekty by mohli vyvolat bolest hlavy, pocit nevolnosti či emoční změny osoby. V takovém případě je doporučeno okamžitě přestat používat aplikaci.

Výsledky této práce dokazují, že je v rámci umělé inteligence možné ovládat předměty protřednictvím lidského mozku. Tento princip ovládání má velmi dobré vyhlídky do budoucnosti. Tento systém snímání mozkových vln je teoreticky schopný odhalit mikrospánek řidiče a v tu chvíli by vyslání zvukového signálu v kabině vozu mohlo zabránit autonehodě.

Aplikace by se měla do budoucna vylepšit o kvalitnější a přesnější klasifikátor naměřených hodnot, jehož implementace nebyla úkolem této práce, ale byl vložen již vyvinutý. Případně by se do ní mohl doimplementovat tzv. Trial Mode, který by před samotným ovládáním vozítka, otestoval uživatele a jako výsledek by mu doporučil frekvence, na které má nejlepší odezvy. Další vlastnost, o kterou by se mohla aplikace vylepšit, je možnost ovládání vozítka pomocí šipek na klávesnici i v případě, když je aktivní Master mód. Tato vlastnost by měla využití jako ochraný prvek vozítka v okamžiku blížící se nehody vozítka při ovládání mozkovými vlnami.

Přehled zkratek

- **GUI**, *Graphical User Interface* - Grafické uživatelské rozhraní.
- **BCI**, *Brain Computer Interface* - Komunikační rozhraní mezi mozkem a strojem.
- **JRE**, *Java Runtime Enviroment* - Aplikační prostředí pro spuštění Java aplikací.
- **JMF**, *Java Multimedia Framework* - Knihovna pro práci s multimediálním obsahem v Javě.
- **FMJ**, *Freedom for Multimedia in Java* - Knihovna implementující funkcionalitu JMF.
- **TCP/IP**, *Transmition Control Protocol* - Síťový protokol pro komunikaci po síti Ethernet.
- **RTP/RTSP**, *Real Time Protocol / Real Time Streaming Protocol* - Protokoly pro přenos videa v reálném čase.
- **JAXB**, *Java Architecture for XML Binding* - Technologie pro zpracování XML souborů a propojení jejich struktury s třídami jazyka Java.
- **XML**, *Extensible Markup Language* - Značkovací jazyk určený pro ukládání strukturovaných dat.
- **OpenGL** - Knihovna pro tvorbu počítačové grafiky.
- **JOGL**, *Java OpenGL* - Java binding na knihovnu OpenGL.

Literatura

[1] Herout Pavel *Učebnice jazyka Java - rozšířené vydání, zahrnuje změny od Java 5.* České Budějovice: Kopp, 2007. ISBN 978-80-7232-323-4.

[2] Herout Pavel *Java - grafické uživatelské prostředí a čeština* Kopp, 2006. ISBN 80-7232-237-0.

[3] Steven J. Luck *An Introduction to the Event-Related Potential Technique* The MIT Press, 2005. ISBN 0-262-62196-7.

[4] Desney S. Tan, Anton Nijholt (Eds.) *Brain-Computer Interfaces - Applying our Minds to Human-Computer Interaction* Springer, 2010. ISBN 978-1-84996-271-1.

[5] *Java Runtime Enviroment* Dostupné z `http://www.oracle.com/technetwork/java/javase/downloads/jre-7u4-download-1591157.html`

[6] *Java Multimedia Framework* Dostupné z `http://www.oracle.com/technetwork/java/javase/download-142937.html`

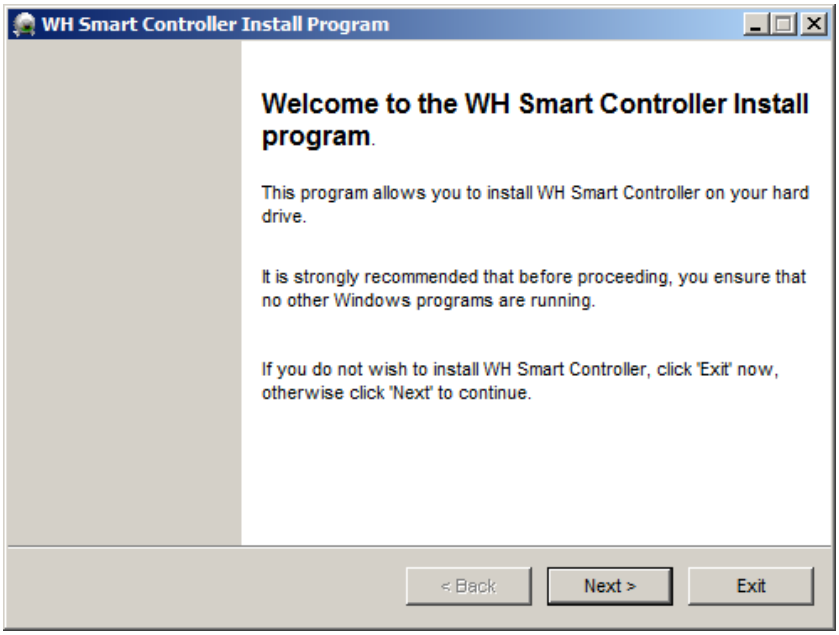
[7] *Freedom for Multimedia in Java* Dostupné z `http://fmj-sf.net/`

A Příloha

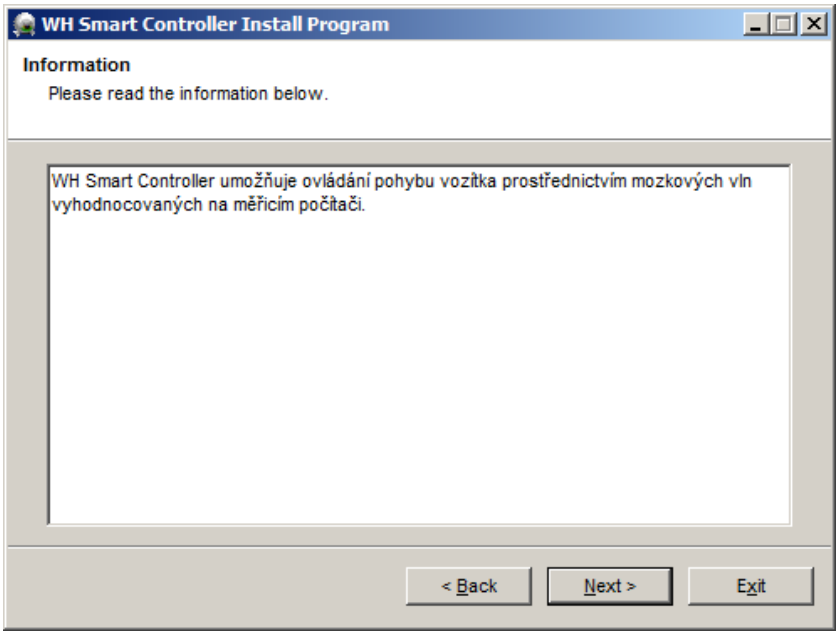
A.1 Uživatelská příručka

A.1.1 Instalace

Kliknutím na setup.exe se spustí instalace aplikace WH Smart Controller. Dále se řiďte pokyny instalátoru.

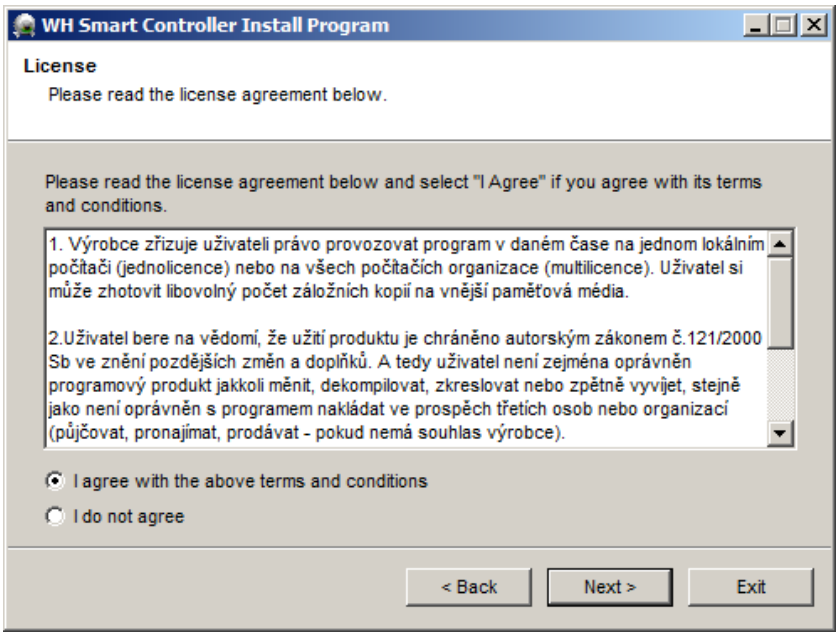


Obrázek A.1: Průběh instalace - krok 1



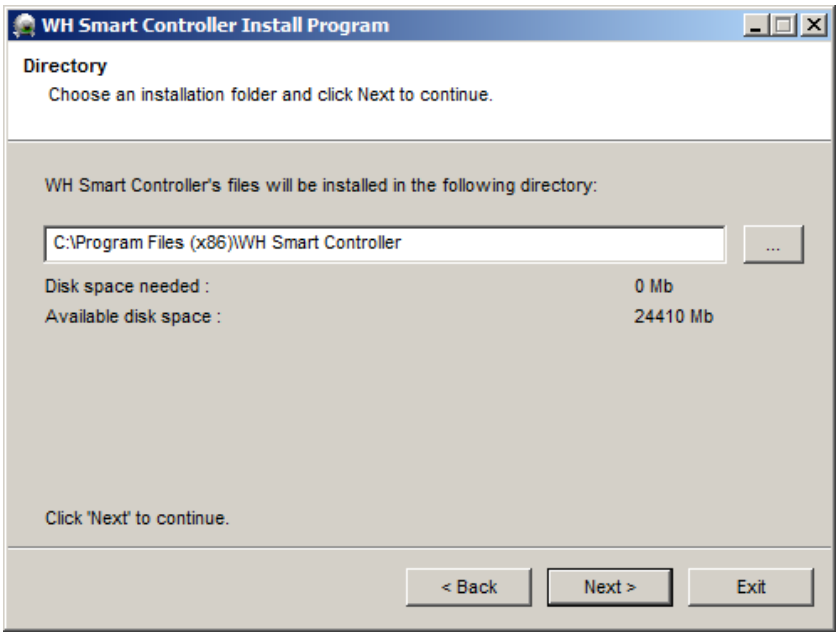
Obrázek A.2: Průběh instalace - krok 2

Pro pokračování instalace musíte v tomto kroku souhlasit s licenčními podmínkami.



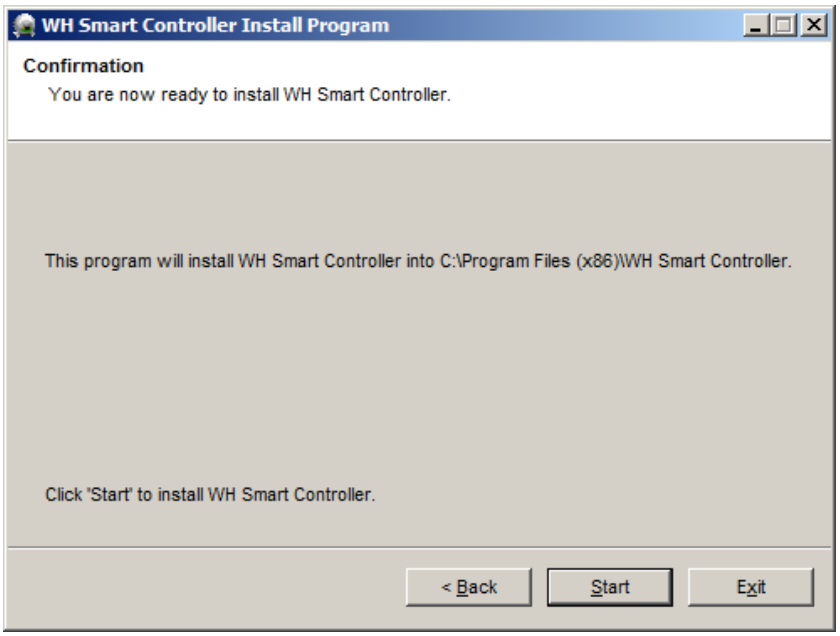
Obrázek A.3: Průběh instalace - krok 3

Zde vyberte umístění na disku, kam chcete, aby se aplikace nainstalovala.



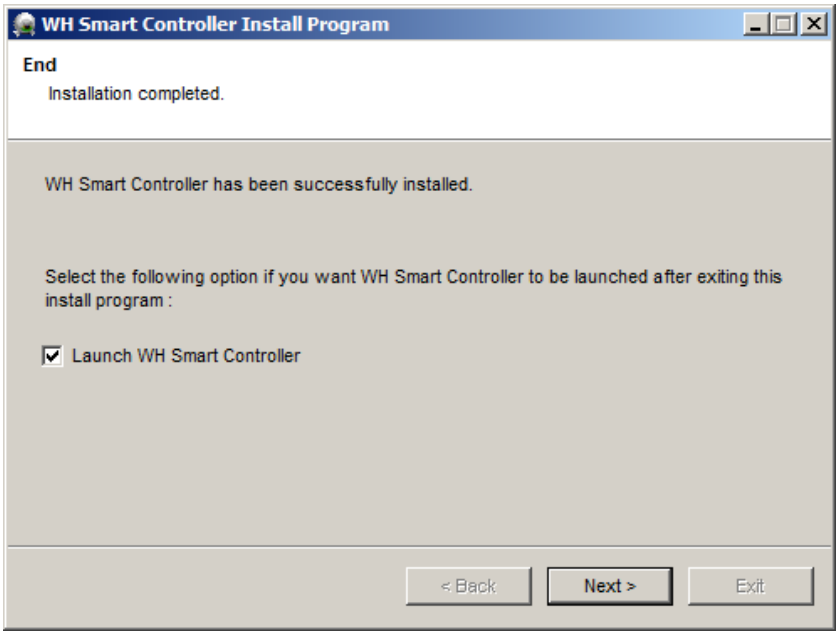
Obrázek A.4: Průběh instalace - krok 4

Stisknutím tlačítka "Start"se spustí instalace.



Obrázek A.5: Průběh instalace - krok 5

V posledním kroku instalace si můžete vybrat, zda se při ukončení instalátoru automaticky spustí aplikace WH Smart Controller.



Obrázek A.6: Průběh instalace - krok 6

Po kompletním dokončení instalace se vytvoří složka v nabídce Start a na ploše se Vám vytvoří ikona WH Smart Controller.

A.1.2 WH Smart Controller

Kliknutí na ikonu whsc.exe, která se nachází na ploše nebo v nabídce Start, slouží ke spuštění aplikace pro ovládání vozítka pomocí evokovaných potenciálů.

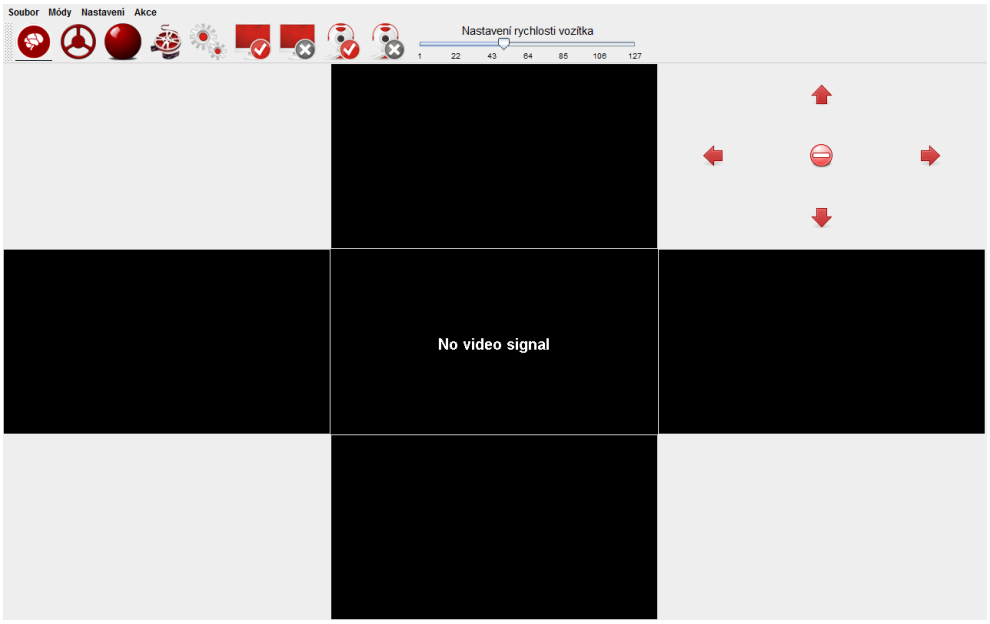
Obrázek A.7: Úvodní obrazovka aplikace WH Smart Controller.

Na úvodní obrazovce programu si můžete vybrat ze tří módů WH Smart Controlleru. Čtvrtý mód "OnlyVideo Mode"je k dispozici spolu s ostatními v Toolbaru nebo v "Módy -> OnlyVideo mód".

A.1.3 Master mód

K dispozici je okno pro zobrazení videa z kamery přidělané na vozítku, čtyři pozice pro evokující obrazce a indikátory komunikace s měřicím počítačem (viz Obrázek A.8). V případě úspěšného navázaní spojení s klientem se prostřední červený indikátor změní na zelený. Šipky mění barvu z červené na zelenou v případě odeslání signálu vozítku pro pohyb daným směrem.

Toolbar v Master módu obsahuje skupinu řídících prvků. První čtyři slouží k přepínání mezi módy (Master, Manual, Marble a OnlyVideo). Další ikona zajišťuje otevření okna s nastavením. Další dvě tlačítka zapínají a vypínají



Obrázek A.8: Náhled aktivního Master módu.

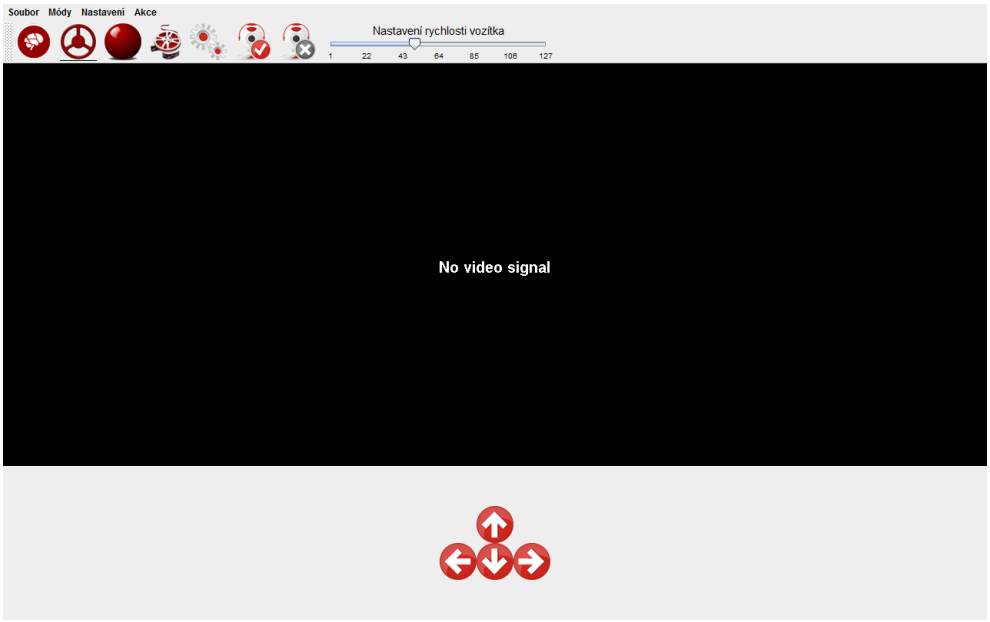
blikání evokujících obrazců a poslední dvě tlačítka slouží k zapnutí a vypnutí zobrazovaní videa z kamery. Úplně vpravo je posuvný regulátor pro nastavení rychlosti vozítka. Náhled tohoto toolbaru viz Obrázek A.9 .

Obrázek A.9: Náhled toolbaru v aktivním Master módu.

A.1.4 Manual mód

Obrazovka Manaul módu obsahuje okno pro zobrazení videa z kamery přidělané na vozítku a prvky pro ovládání vozítka (viz Obrázek A.10). Vozítko lze ovládat šipkami na klávesnici nebo klikáním na směrové šipky v aplikaci.

Toolbar v Manual módu také obsahuje několik ikon. První čtyři slouží k přepínání mezi módy (Master, Manual, Marble a OnlyVideo). Další ikona zajišťuje otevření okna s nastavením. Další dvě tlačítka slouží k zapnutí a vypnutí zobrazování videa z kamery. Úplně vpravo je posuvný regulátor pro nastavení



Obrázek A.10: Náhled aktivního Manual módu.

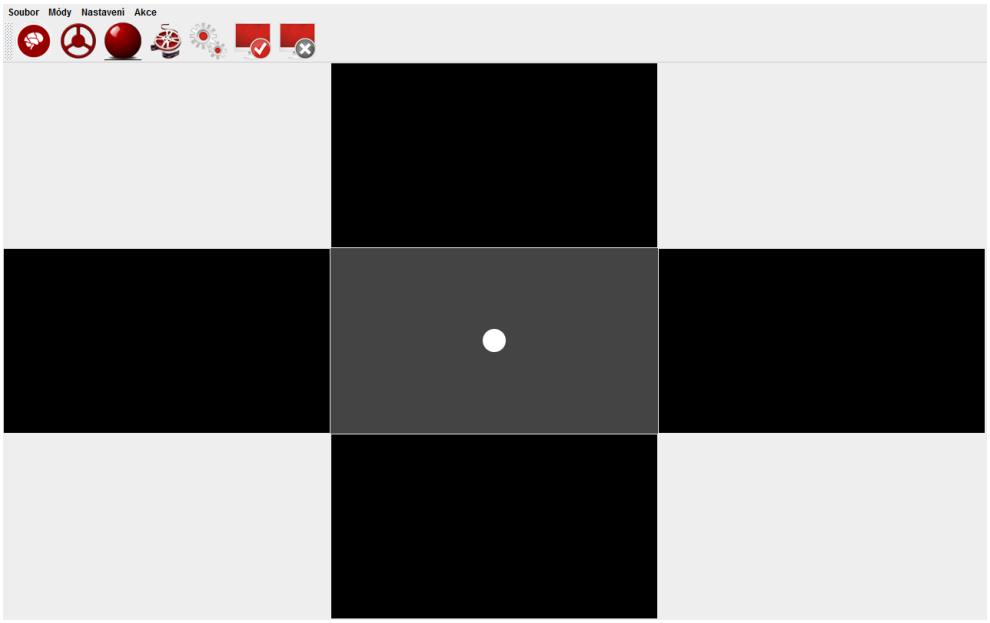
rychlosti vozítka. Náhled tohoto toolbaru viz Obrázek A.11 .

Obrázek A.11: Náhled toolbaru v aktivním Manual módu.

A.1.5 Marble mód

Obrazovka Marble módu obsahuje čtyři pozice pro evokující obrazce a okno, ve kterém je „kulička“, která se pohybuje podle výsledků vyhodnocených klasifikátorem (viz Obrázek A.12).

Toolbar v Marble módu obsahuje několik ikon. První čtyři slouží k přepínání mezi módy (Master, Manual, Marble a OnlyVideo). Další ikona otevře okno s nastavením. Další dvě tlačítka zapínají a vypínají blikání evokujících obrazců. Náhled tohoto toolbaru viz Obrázek A.13 .



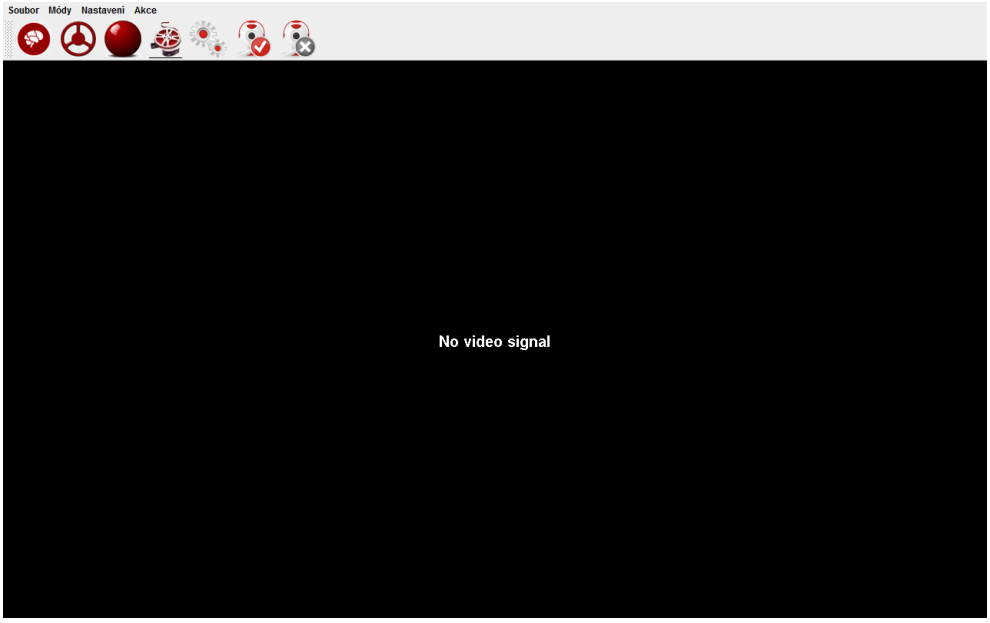
Obrázek A.12: Náhled aktivního Marble módu.

Obrázek A.13: Náhled toolbaru v aktivním Marble módu.

A.1.6 OnlyVideo mód

Obrazovka OnlyVideo módu obsahuje pouze okno pro zobrazení videa z kamery přidělané na vozítku (viz Obrázek A.14).

Toolbar v OnlyVideo módu také obsahuje několik ikon. První čtyři slouží k přepínání mezi módy (Master, Manual, Marble a OnlyVideo). Další ikona zajišťuje otevření okna s nastavením. Další dvě tlačítka slouží k zapnutí a vypnutí zobrazování videa z kamery. Náhled tohoto toolbaru viz Obrázek A.15 .



Obrázek A.14: Náhled aktivního OnlyVideo módu.

Obrázek A.15: Náhled toolbaru v aktivním Onlyvideo módu.

A.1.7 Nabídka menu

Po kliknutí na položku „**Soubor**“ se rozevře nabídka s příslušnými položkami.
Domů – zobrazí úvodní obrazovku.
Ukončit aplikaci – ukončí celý program.
Vedle každé položky je uvedena klávesová zkratka, která také vykoná výše popsanou akci.

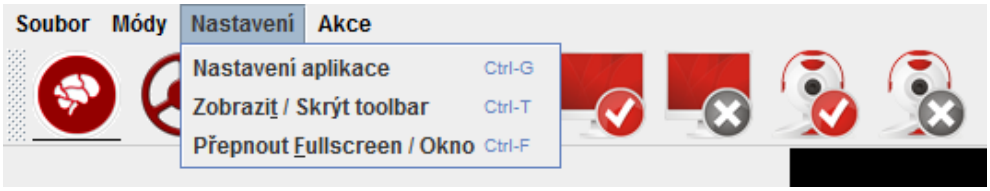
Obrázek A.16: Nabídka menu po kliknutí na položku Soubor.

Po kliknutí na položku „**Módy**“ se rozevře nabídka s příslušnými položkami.

Master mód – přepne aplikaci do Master módu.
Manual mód – přepne aplikaci do Manual módu.
Marble mód – přepne aplikaci do Marble módu.
OnlyVideo mód – přepne aplikaci do OnlyVideo módu.
Vedle každé položky je uvedena klávesová zkratka, která také vykoná výše popsanou akci.

Obrázek A.17: Nabídka menu po kliknutí na položku Módy.

Po kliknutí na položku „**Nastavení**“ se rozevře nabídka s příslušnými položkami.
Nastavení aplikace – otevře okno s nastavením aplikace.
Zobrazit / Skrýt toolbar – zobrazuje nebo skrývá toolbar.
Přepnout Fullscreen / Okno – přepíná mezi zobrazením v okně a fullscreen režimem.
Vedle každé položky je uvedena klávesová zkratka, která také vykoná výše popsanou akci.



Obrázek A.18: Nabídka menu po kliknutí na položku Nastavení.

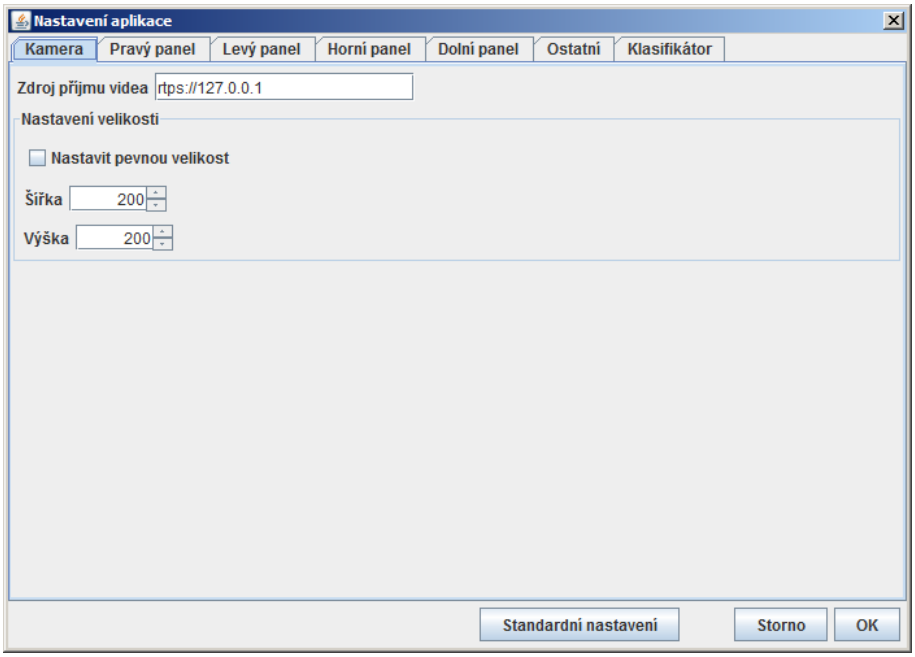
Po kliknutí na položku **"Akce"** se rozevře nabídka s příslušnými položkami. Tato nabídka se liší podle toho, jaký je vybrán mód, protože jsou v ní položky specifické pro vybraný mód. Tyto položky mají stejné funkce jako tlačítka na toolbaru, které se nacházejí za tlačítkem "Nastavení aplikace". Jedná se tedy o funkce (Spustit evokující obrazce, Zastavit evokující obrazce, Spustit video a Zastavit video).

A.1.8 Nastavení aplikace

V okně s nastavením je na výběr z několika záložek (Kamera, Pravý panel, Levý panel, Horní panel, Dolní panel, Ostatní a Klasifikátor).

Kamera:
Zde je možnost nastavit:

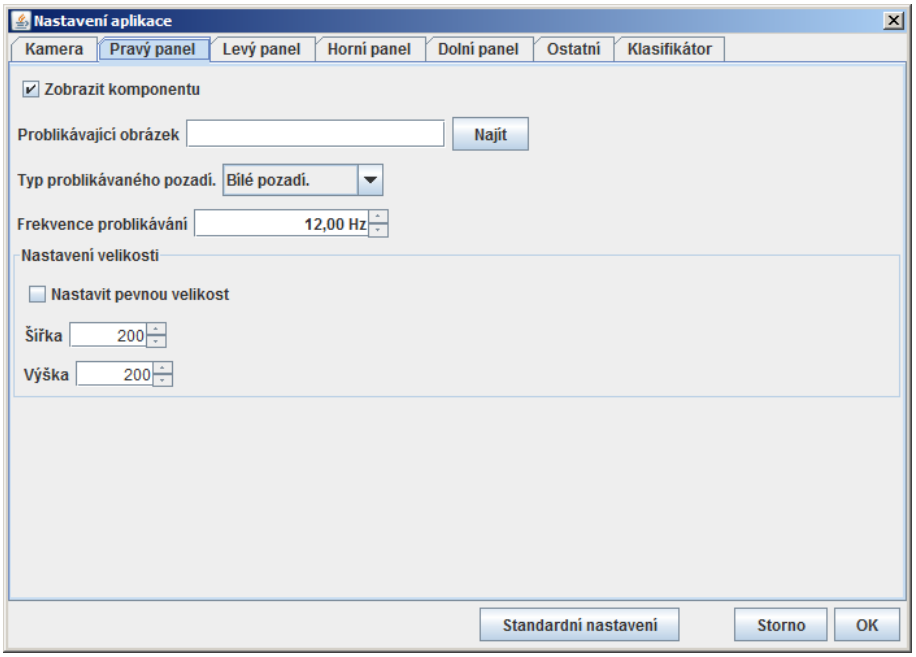
Zdroj přijmu videa - IP adresu zdroje videa (IP kamery).
Nastavit pevnou velikost – Panel pro zobrazování videa bude mít pevnou velikost.
Šířka, Výška – Velikost panelu pro zobrazování videa.



Obrázek A.19: Záložka "Kamera"v okně nastavení aplikace.

Pravý / levý / horní / dolní panel:
Všechny čtyři panely pro blikající obrazce mají totožné nastavení. Pro každý z nich je možnost nastavit:

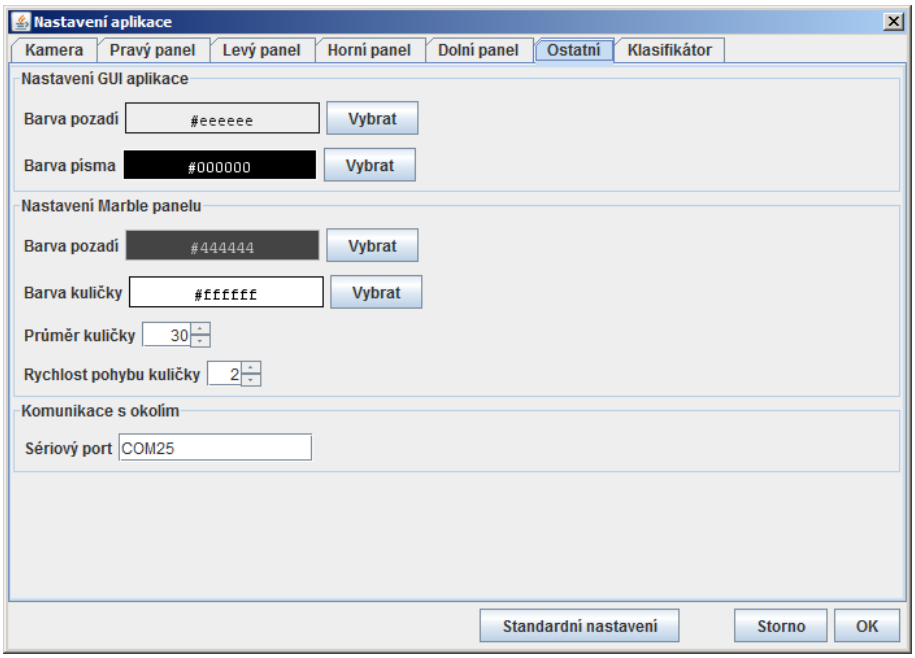
Zobrazit komponentu - Zobrazit / nezobrazit komponentu s blikajícím obrazcem.
Problikávající obrázek - Výběr vlastního zobrazovaného obrázku.
Typ problikávaného pozadí - Obrázek bude problikávat s bílým pozadím nebo s inverzí sebe samotného.
Frekvence blikání - Frekvence blikání obrazce.
Nastavit pevnou velikost - Zobrazovaná komponenta bude mít pevnou velikost.
Šířka, výška - Velikost zobrazované komponenty.



Obrázek A.20: Záložka "Pravý panel"v okně nastavení aplikace.

Ostatní:
Zde je možnost nastavit:

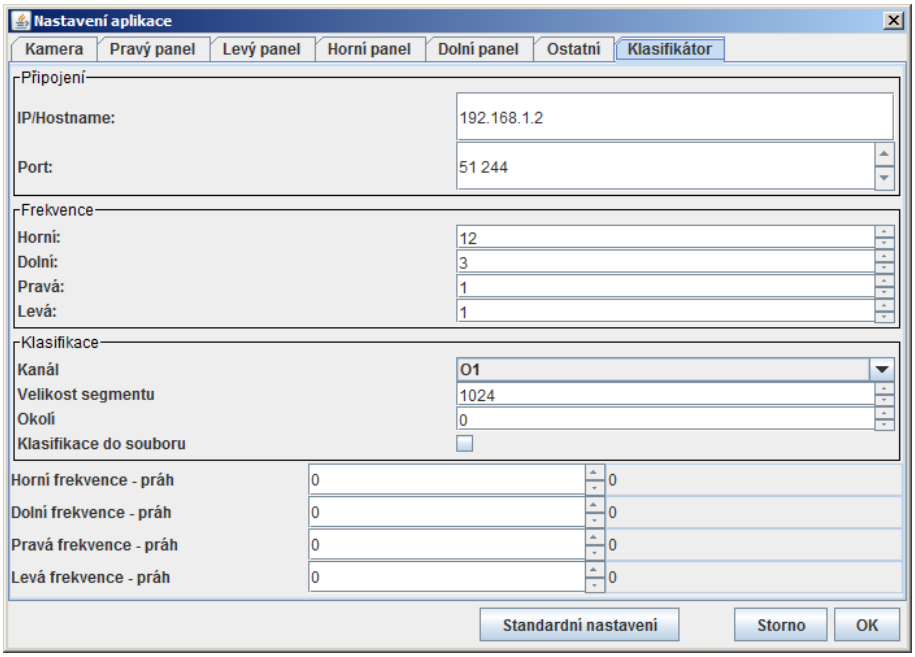
GUI barva pozadí - Barva pozadí celé aplikace.
GUI barva písma - Barva písma v celé aplikaci.
Marble - Barva pozadí - Barva pozadí v Marble módu v panelu s "kuličkou".
Marble - Barva kuličky - Barva "kuličky"v Marble módu.
Průměr kuličky - Průměr kuličky v Marble módu.
Rychlost pohybu kuličky - Rychlost pohybu kuličky v Marble módu.
Sériový port - COM port, na kterém je připojeno vozítko pomocí Bluetooth.



Obrázek A.21: Záložka "Ostatní"v okně nastavení aplikace.

Klasifikátor:
Zde je možnost nastavit:

IP/Hostname - IP adresa měřicího počítače, od kterého přijímá data.
Port - Port, na kterém přijímá data.
Frekvence - Frekvence, kterými blikají obrazce.
Kanál - Kanál, na kterém se měří mozková aktivita.
Velikost segmentu - Velikost segmentu, který se klasifikuje.
IP/Hostname - Ukládat / neukládat výsledky klasifikace do souboru.
Horní / dolní / pravá / levá frekvence - práh - Velikost práhu pro vybranou frekvenci.



Obrázek A.22: Záložka "Klasifikátor"v okně nastavení aplikace.

Všechna změněná nastavení se potvrzují tlačítkem „**OK**“ nebo naopak ignorují tlačítkem „**Storno**“. Pro standardní nastavení barev, zobrazení komponent, velikost komponent, zdroje videa, zdroje blikajících obrázků, druh blikání a komunikačního portu lze použít tlačítko "**Standardní nastavení**".