SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE
dokumentace

Tréninková časomíra pro hasiče

Jaromír Wysoglad

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2016/2017

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2016

podpis autora práce

ANOTACE

Cílem projektu bylo vytvoření tréninkové časomíry pro hasiče, která by umožnila snadnou a rychlou manipulaci. Pro kontrolu stavu terčů byl použit čip ESP8266 naprogramovaný pomocí jazyka C a frameworku Sming. Čip prostřednictvím wifi komunikuje s aplikací na notebooku nebo na mobilu poblíž startu. Tato aplikace byla vytvořena v jazyce C++ s využitím knihovny SDL2. Součástí časomíry je i webové rozhraní, které pomocí JavaScript a HTML5 Canvas zobrazuje grafy již odběhnutých časů a umožňuje jejich export v podobě CSV souboru nebo jejich vložení či mazání pomocí editačního rozhraní.

OBSAH

ÚVOD

Ve volném čase se už několik let s týmem hasičů z SDH Zátor věnuji soutěžím v požárním útoku. Požární útok je disciplína, při které se závodníci snaží v co nejkratším čase natáhnout celé hadicové vedení od zdroje vody skrz přenosnou stříkačku (čerpadlo) až k terčům a terče shodit, což vyžaduje přesnou časomíru. Navíc mi bylo v únoru minulého roku řečeno, že jeden ze sto metrových drátů naší staré časomíry je přerušen, a proto časomíra nefunguje.

Mým cílem bylo navrhnout takovou časomíru, která by se dala co nejsnadněji a nejrychleji nachystat i uklidit. Toho jsem docílil použitím čipu ESP8266, jenž umožňuje komunikaci pomocí Wi-Fi a elegantně tak řeší věčné problémy s kabelem. Díky pokročilým vlastnostem čipu ESP jsem mohl novou časomíru doplnit i o webové stránky, které poskytují přehledné zobrazení dosažených časů v podobě grafu, přidávání nových i odstraňování starých výsledků a export ve formátu souboru CSV.

V této dokumentaci se zabývám především problematikou programování čipu ESP8266 a využití knihoven SDL (SDL_net, SDL_mixer, SDL_image, SDL_ttf) ve spojení s jazykem C++ k vývoji desktopové i mobilní aplikace komunikující s čipem ESP8266. Naznačuji rovněž princip fungování webové aplikace, která využívá JavaScript a HTML5 Canvas pro zobrazování grafů. Poslední kapitola je věnována především ovládání časomíry a uživatelskému rozhraní aplikací.

1 TEORETICKÁ A METODICKÁ VÝCHODISKA

1.1 SDL

SDL (Simple DirectMedia Layer) je multiplatformní knihovna, která nabízí nízkoúrovňový přístup ke zvuku, vstupním zařízením (myš, klávesnice, joystick) a grafice pomocí OpenGL a Direct3D.

SDL podporuje Windows, Mac OS X, Linux, iOS a Android.

SDL je napsáno v C, funguje proto i ve spolupráci s C++ a jsou k dispozici i verze knihovny pro jiná jazyky včetně Python a C#.

```
bool quit = false;
bool vypis = true;
bool pohlavi = true;
bool started = false;
unsigned int start = 0;
SDL_Event e;
if(SDL_PollEvent(&e))
{
        if(e.type==SDL_QUIT)
            quit=true;
        else if (e.type==SDL_KEYDOWN)
        {
            switch(e.key.keysym.sym)
            {
                case SDLK_SPACE:
                    start = SDL_GetTicks();
                    if(started)
                    {
                        started = false;
                        startButton.setImage(std::string("img/start.png"));
                        startText->setHodnota(std::string("START"));
                        startButton.render();
                    }
                    …
                    break;
                …
                }
        }
}
```

Ukázka zachycení stisku klávesy v okně pomocí SDL

```
GameWindow::GameWindow(std::string nazev, int width, int height, int flag, int x, int y)
{
    //inicializace SDL
    if(!SDL_WasInit(SDL_INIT_VIDEO))
        if(SDL_Init(SDL_INIT_VIDEO))
            std::cout << "\nUnable to initialize SDL:  %s\n" << SDL_GetError();

    //vytvoreni okna
    window = SDL_CreateWindow(
        nazev.c_str(),                  // window title
        x,                              // initial x position
        y,                              // initial y position
        width,                          // width, in pixels
        height,                         // height, in pixels
        flag                            // flags
    );
    if(window == NULL)
        std::cout<<"Okno nemohlo byt vytvoreno! SDL_ERROR: %s\n"<< SDL_GetError();
}
```

Vytvoření okna pomocí SDL

K SDL existují i některé přídavné knihovny, já jsem využil SDL_net pro síťovou komunikaci, SDL_mixer pro zvuk, SDL_image pro načítání obrázků a SDL_ttf pro práci s fonty.

1.1.1 SDL_net

SDL_net je jednoduchá knihovna, která se spolu s SDL používá pro síťovou komunikaci.

```
if(SDLNet_UDP_Recv(socketIn, packetIn)) {
    prijato = packetIn->data[0];
    switch(prijato)
    {
        case 'm':
            pohlavi = true;
            muziZenyButton.setImage(std::string("img/muzi.png"));
            muziZenyText->setHodnota("Muzi");
            muziZenyButton.render();
            break;
        …
    }
}
```

Přijetí UDP datagramu

```
int posliPacket(char *data, UDPpacket *packet, UDPsocket *socket)
{
    packet->len = strlen(data) + 1;
    memcpy(packet->data, data, packet->len);
    return SDLNet_UDP_Send(*socket, -1, packet);
}
```

Odeslání UDP datagramu

1.1.2 SDL_image

SDL_image je knihovna, která se spolu s SDL používá k otevírání obrázků různých formátů bez nutnosti programování různých nekompresních a konverzních algoritmů.

```
Image::Image(std::string path,int w, int h,int x, int y, double angle) {
    rwop=SDL_RWFromFile(path.c_str(), "rb");
    if(rwop == NULL)
        std::cout << "Nelze nacist obrazek" << IMG_GetError() << "\n" ;
    image=IMG_LoadPNG_RW(rwop);
    if(!image)
        std::cout << "Nelze nacist obrazek" << IMG_GetError() << "\n" ;
}
```

Načtení png obrázku

1.1.3 SDL_mixer

SDL_mixer je knihovna, která se spolu s SDL používá pro přehrávání zvuků. Tato knihovna umožňuje snadné načtení zvuků v různých formátech a umožňuje snadné přehrání i několika z nich najednou.

1.1.4 SDL_ttf

SDL_ttf umožňuje načtení námi zvoleného TrueType fontu a následně spolu s SDL vyrenderování programátorem zadaného textu s použitím tohoto fontu.

2 VYUŽITÉ TECHNOLOGIE

2.1 ESP8266 – 201

ESP8266 je levný wifi modul, který se dá použít jak ve spolupráci s Arduinem nebo jiným čipem, tak i samostatně. Tento konkrétní model 201 obsahuje 512 MB flash paměti. Je možné ho programovat v jazyce C, C++, Lua, Python, JavaScript.

ESP jsem zvolil po jeho doporučení p. Grussmanem, a také proto, že jsem potřeboval způsob komunikace mezi startem a terči, který tolik nezdržuje při chystání nebo sklízení jako 100 metrů dlouhý, k přerušení náchylný kabel.

2.2 C++

C++ je programovací jazyk, který vyvinul Bjarne Stroustrup a je rozšířením jazyka C. C++ podporuje několik programovacích stylů jako je procedurální programování, objektově orientované programování a generické programování, není tedy jazykem čistě objektovým. V současné době patří C++ mezi nejrozšířenější programovací jazyky.

Jazyk C++ jsem zvolil, protože se jej učíme ve škole, dobře se mi v něm programuje a je vhodný pro programování mikrokontrolerů i ESP.

C++ jsem tedy s pomocí frameworku Sming použil pro naprogramování ESP a s pomocí knihoven SDL2, SDL2_net, SDL2_image, SDL2_ttf a SDL2_mixer pro naprogramování desktopové části časomíry.

2.2.1 Sming

Sming je open source framework sloužící pro programování ESP v jazyce C++. Výhodou programování v tomto frameworku je velká podobnost s programováním pro Arduino; vývojáři zvyklí na práci s Arduinem tedy nemají problém s přechodem na ESP a Sming a také knihovny používané pro Arduino se dají využít spolu se Smingem.

2.2.2 SDL

Simple DirectMedia Layer je multiplatformní knihovna v jazyce C, díky které se dá programovat grafika, audio a komunikace po síti. Umožňuje dokonce i programování v jazycích C nebo C++ pro mobily.

2.3 Využité programy

2.3.1 NetBeans

NetBeans je open-source IDE, které se používá k programování v jazycích C, C++, Java, PHP, HTML, JavaScript. Použil jsem ho pro programování desktopové části aplikace v jazyce C++ spolu s knihovnou SDL.

2.3.2 Sublime text

Sublime text je multiplatformní textový editor, oproti jiným editorům se liší mnoha užitečnými pokročilými funkcemi, např. editováním několika částí kódu najednou, editování několika souborů najednou, možnost stažení mnoha snipetů, které ulehčují programování. Použil jsem k vývoji webové části aplikace a pro naprogramování firmware do ESP v jazyce C a frameworkem Sming.

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

3.1 Část časomíry u terčů

Pro detekci stavu terčů jsem použil čip ESP8266 a vodotěsný vypínač na světla. Při sestříknutí terče se stiskne vypínač, který rozsvítí žárovku a přes tranzistor připojí jeden z pinů ESP8266 pod přerušením na zem. Při sestříknutí některého z terčů pošle ESP prostřednictvím Wi-Fi aplikaci v notebooku (mobilu) UDP packet s informací, který terč byl sestříknut. Při sestříknutí obou terčů se pošle z notebooku na startu UDP packet zpět na ESP, který obsahuje aktuální datum a dosažený čas, tyto informace se uloží do CSV souboru pro pozdější zobrazení na webu.

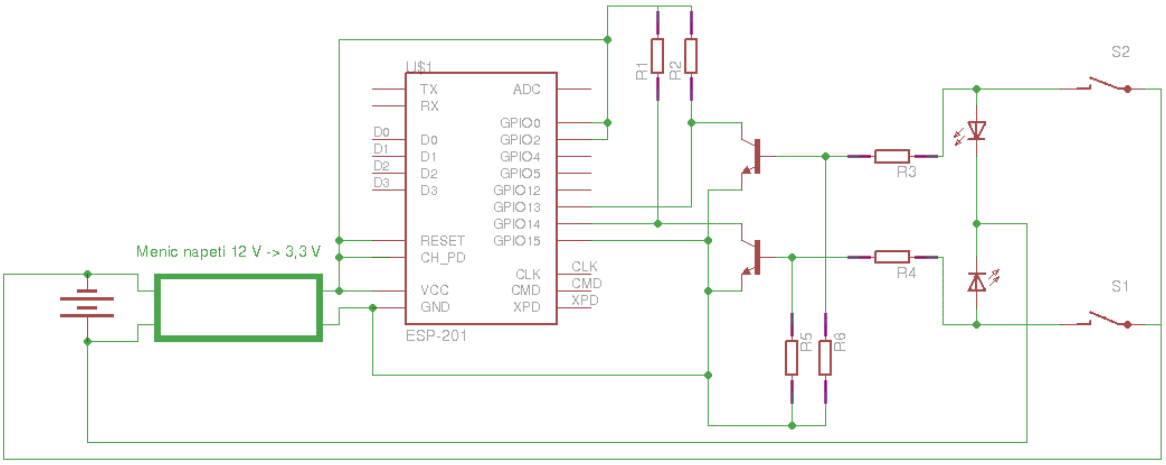


Schéma zapojení

Vzadu za terčem jsem použil již odzkoušený sklápěcí mechanismus ze staré časomíry:

3.2 Část časomíry u startu

K startování a případnému zastavování časomíry při nezdařeném pokusu slouží program napsaný v jazyce C++ s pomocí knihoven SDL2, SDL_net, SDL_mixer, SDL_image, SDL_ttf. Použití knihoven SDL mi umožnilo již napsanou desktopovou aplikaci použít i pro mobily, proto se dá k startování používat buď notebook, nebo mobil s Androidem. Program se dá ovládat pomocí konzole, kde jsem využil funkci kbhit(). Díky této funkci lze zjistit, zda bylo do konzole cokoliv napsáno, aniž by bylo nutné stisknout klávesu Enter a zároveň nezastavuje běh programu, pokud nedošlo k žádnému uživatelskému vstupu. Dále je možné program ovládat i pomocí myši nebo klávesnicí v okně s grafickým uživatelským rozhraním. Při jeho vývoji jsem používal knihovny SDL2, SDL_image pro načítání obrázků a SDL_ttf pro práci s textem. Pro GUI jsem využil 4 třídy z jednoho mého staršího projektu. Aplikace je responzivní (pozice a velikosti tlačítek jsou zadány ve zlomcích aktuální velikosti okna).

3.3 Webová část aplikace

Čip ESP8266 umožňuje i vytvoření webového serveru, proto jsem na terčích vytvořil webové stránky, které pomocí HTML5 Canvasu a JavaScriptu zobrazují grafy dosažných časů pro jednotlivé kategorie. Dále tento web umožňuje i mazání již dosažených časů, přidávání nových časů a jejich export jako soubor CSV.

3.3.1 Grafy v JavaScriptu

Také k vytváření grafů jsem použil část svého staršího projektu, ve kterém jsem pomocí HTML5 Canvasu a JavaScriptu v grafu zobrazoval výsledky všech týmů z Hasičské ligy Praděd.

```
var proudari = {
	muzi:{
		pProud:[],
		lProud:[],
		day:{
			datum:[],
			pocet:[]
		}
	},
	zeny:{
		pProud:[],
		lProud:[],
		day:{
			datum:[],
			pocet:[]
		}
	}
}
```

Jako zdroj dat pro graf používám JSON s touto strukturou:

Protože se budou webové stránky zobrazovat na zařízeních s různou velikostí displeje, musí být responzivní. Dále jsem také chtěl, aby graf vždy zabíral celý canvas: pokud jsou tedy dosaženy například pouze dva časy, v grafu se objeví první čas úplně vlevo a druhý zcela vpravo. Po přidání dalšího času se graf automaticky překreslí, aby zachytil všechny hodnoty. To samé se děje i při dosažení nového nejhoršího, nebo nového nejlepšího času. Díky tomu vznikly při vykreslování dva docela nepřehledné řádky, které lze vidět na dalším obrázku.

```
for(var j=0;j<proudari[pohlavi].pProud.length;j++)
{
	c.beginPath();
	if(j==0)
		c.moveTo(0,vyska);
	else
		c.moveTo(sirka/proudari[pohlavi].pProud.length*j,vyska-(vyska/(max-
			min)*(proudari[pohlavi].pProud[j-1]-min)));
	c.strokeStyle="rgb(255,0,0)";
	c.lineTo((sirka/proudari[pohlavi].pProud.length)*(j+1),vyska-(vyska/(max-
		min)*(proudari[pohlavi].pProud[j]-min)));
	if("pravy"==zvirazni)
		c.lineWidth = 10;
	else
		c.lineWidth = 2;
	c.lineCap   = 'round';
	c.stroke();
}
```

Ukázka vykreslení jedné z křivek grafu

vyska = výška canvasu
sirka = šířka canvasu
max = nejhorší čas (nejvyšší bod grafu)
min = nejlepší čas (nejnižší bod grafu)

Součástí grafu je i stupnice, která je rovněž responzivní. V závislosti na velikosti displeje mění velikost čísel, podle nejlepšího a nejhoršího času mění svůj začátek i konec a podle toho se také mění hodnota u jednotlivých stupňů (počet stupňů je stále stejný).

Protože pouhým pohledem na graf nelze zjistit přesný čas, tak se při najetí kurzoru myši na jednu z křivek grafu daná křivka zvýrazní (změní se její tloušťka), a všechny časy se vypíší do tabulky pod grafem. Detekci toho, jestli je kurzor myši nad křivkou grafu, jsem vyřešil tak, že pomocí asynchronní funkce neustále při pohybu myši získávám barvu pixelu pod kurzorem a kontroluji, jestli není červený, nebo modrý (křivka grafu). Jelikož může být na některých mobilech obtížné dotknout se 2 pixely široké křivky, tak při vykreslování grafu vykresluji pod každou z křivek také další křivku stejné barvy, 20 pixelů širokou, s alfa kanálem nastaveným tak, aby pro uživatele nebyla vidět, ale dala se v JavaScriptu detekovat.

```
function pixelOnMouseOver(canvas,callback)
{
	ctx = canvas.getContext("2d");
	canvas.addEventListener('mousemove',function(e){
		var w = canvas.width, h=canvas.height;
		data= ctx.getImageData(0,0,w,h).data;
		if (e.offsetX === void 0) {
			Object.defineProperties(MouseEvent.prototype, {
				'offsetX': {
					get: function() {
						return this.layerX - this.target.offsetLeft;
					}
				}, 'offsetY': {
					get: function() {
						return this.layerY - this.target.offsetTop;
					}
				}
			});
		}
		var idx = (e.offsetY*w + e.offsetX)*4;
		var parts = Array.prototype.slice.call(data,idx,idx+4);
		mousePos = getMousePos(canvas, e);
		callback.apply(ctx,parts);
	},false);
}


pixelOnMouseOver(document.getElementById("muziProudy"),function(r,g,b){
		if(("("+r+","+g+","+b+")"=="(255,0,0)"||"("+r+","+g+","+b+")"=="(0,0,255)") &&
			document.getElementById("proud").innerHTML !=
			(("("+r+","+g+","+b+")" == "(255,0,0)") ? "Pravý proud" : "Levý proud"))
		{
		    //výpis časů do tabulky, zvýraznění jedné z křivek
		}
}
```

Funkce pro detekci pixelu pod myší

4 VÝSLEDKY ŘEŠENÍ, VÝSTUPY, UŽIVATELSKÝ MANUÁL

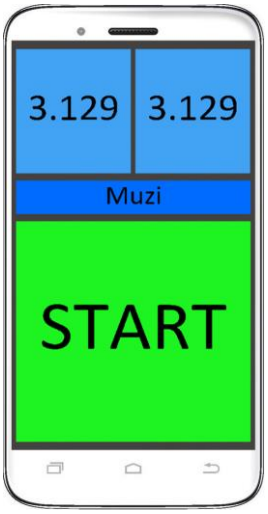
4.1 Ovládání

4.1.1 Konzole

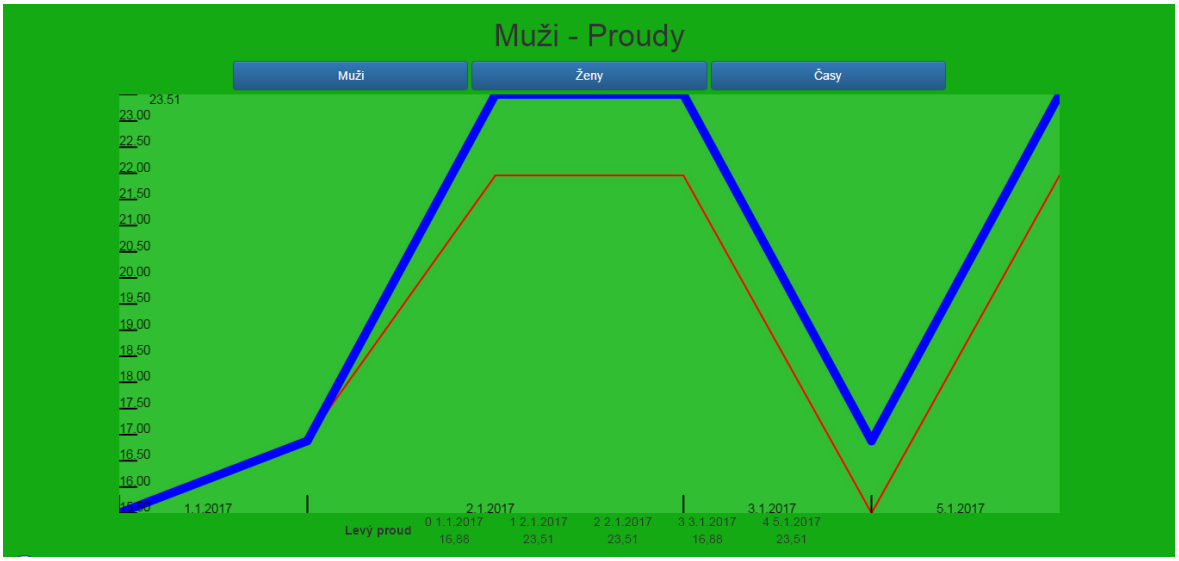
Program po stisku klávesy **`h`** v konzoli vypíše nápovědu, která popisuje veškeré další ovládání. Stiskem klávesy **`m`**, nebo **`z`**, lze přepínat mezi kategoriemi (muži/ženy). Zvolení správné kategorie je důležité kvůli následnému zápisu výsledného času do ESP a poté jeho zobrazení v grafu. Po stisknutí **`mezerníku`** se spustí, nebo zastaví měření času. Stiskem klávesy **`t`** se pošle UDP packet na ESP, které vzápětí odešle packet zpátky, jímž lze zjistit, zda jsou terče zvednuté, nebo je potřeba k nim zajít a zvednout je.

4.1.2 GUI

Stiskem klávesy **`t`**, nebo kliknutím na jedno ze dvou horních tlačítek lze zjistit aktuální stav terčů, který se projeví zbarvením horních dvou tlačítek zelenou barvou (pokud jsou terče zvednuté), nebo červenou barvou (pokud je potřeba je zvednout). Stiskem klávesy **`m`** nebo **`z`**, nebo kliknutím na tlačítko uprostřed s právě zvolenou kategorií lze přepínat mezi kategoriemi. Stiskem **`mezerníku`**, nebo kliknutím na tlačítko s nápisem START nebo STOP lze spustit, nebo zastavit měření času. Jelikož je aplikace pro Android prakticky totožná s aplikací pro notebook, tak se stejně i ovládá.



Na webu lze nahoře v menu vybrat kategorii, pro kterou chceme zobrazit graf s již dosaženými časy. Po najetí myši na křivku v grafu, nebo dotykem v její blízkosti na dotykovém displeji lze o křivce zjistit bližší informace.



Grafy na webu

Kliknutím na tlačítko Časy v menu se lze přesunout do administračního rozhraní webové aplikace, kde jsou ve dvou tabulkách všechny již dosažené časy v obou kategoriích a můžeme je v nich mazat. Dále lze na této stránce časy i přidávat.

Webové administrační rozhraní

ZÁVĚR

Cílem práce bylo vytvořit časomíru, která by byla snadnější a rychlejší k použití než ta stávající. Tohoto zjednodušení jsem dosáhl tím, že jsem nahradil 100 metrů dlouhý kabel, který byl náchylný k přerušení a jeho chystání na začátku tréninku a uklízení na jeho konci trvalo zbytečně dlouho, bezdrátovou Wi-Fi komunikací. Využití Wi-Fi mi také dalo možnost pomocí grafů zobrazovat již dosažené časy na webových stránkách.

Všech vytyčených cílů se mi nakonec povedlo dosáhnout. Aplikace na notebooku i na mobilu úspěšně komunikuje s aplikací na ESP, která zjišťuje stav terčů. Časomíra je jednodušší a rychlejší na přípravu a prozatím nemá žádný problém se špatným kontaktem jako původní časomíra. Dosud tedy vše funguje jak má, ten pravý test však přijde až na jaře při opravdovém útoku.

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] Sming Framework API [online].
[cit. 2016-12-20].
<http://sminghub.github.io/sming-api-develop/index.html>.

[2] SDL 2.0 API by Name [online].
[cit. 2016-12-20].
<https://wiki.libsdl.org/CategoryAPI>.

[3] Lazy Foo, Beginning Game Programming v2.0 [online].
poslední revize 15. 2. 2016 [cit. 2016-12-20].
<http://lazyfoo.net/tutorials/SDL/index.php>.

[4] Sming [online].
poslední revize 15. 2. 2016 [cit. 2016-2-20].
<https://github.com/SmingHub/Sming>.

[5] SDL_net documentation [online].
[cit. 2016-12-20].
<http://jcatki.no-ip.org:8080/SDL_net/SDL_net_frame.html>.

[6] SDL_mixer documentation [online].
[cit. 2016-12-20].
<http://jcatki.no-ip.org:8080/SDL_mixer/SDL_mixer_frame.html>.

[7] SDL_image documentation [online].
[cit. 2016-12-20].
<https://www.libsdl.org/projects/SDL_image/docs/SDL_image_frame.html>.

[8] SDL_ttf documentation [online].
[cit. 2016-12-20].
<https://www.libsdl.org/projects/SDL_ttf/docs/SDL_ttf_frame.html>.