|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Tréninková časomíra pro hasiče** | | |
| Jaromír Wysoglad | | |
| [Vložím obrázek terče, až si vzpomenu, že ho mám vyfotit v nějakém pěkném světle] | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  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, který pomocí wifi komunikuje s aplikací naprogramovanou v jazyce C++ s pomocí knihovny SDL2 na notebooku nebo na mobilu poblíž startu. 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 6](#_Toc471681665)

[1 Teoretická a metodická východiska 7](#_Toc471681666)

[1.1 SDL 7](#_Toc471681667)

[1.1.1 SDL\_net 8](#_Toc471681668)

[1.1.2 SDL\_image 9](#_Toc471681669)

[1.1.3 SDL\_mixer 10](#_Toc471681670)

[1.1.4 SDL\_ttf 10](#_Toc471681671)

[2 Využité technologie 11](#_Toc471681672)

[2.1 ESP8266 – 201 11](#_Toc471681673)

[2.2 C++ 11](#_Toc471681674)

[2.2.1 Sming 11](#_Toc471681675)

[2.2.2 SDL 12](#_Toc471681676)

[2.3 Využité programy 12](#_Toc471681677)

[2.3.1 NetBeans 12](#_Toc471681678)

[2.3.2 Sublime text 12](#_Toc471681679)

[3 Způsoby řešení a použité postupy 13](#_Toc471681680)

[3.1 Část časomíry u terčů 13](#_Toc471681681)

[3.2 Část časomíry u startu 13](#_Toc471681682)

[3.3 Webová část aplikace 13](#_Toc471681683)

[3.3.1 Grafy v JavaScriptu 14](#_Toc471681684)

[4 Výsledky řešení, výstupy, uživatelský manuál 17](#_Toc471681685)

[4.1 Ovládání 17](#_Toc471681686)

[4.1.1 Konzole 17](#_Toc471681687)

[4.1.2 GUI 17](#_Toc471681688)

[4.1.3 Web 17](#_Toc471681689)

[Závěr 19](#_Toc471681690)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 20](#_Toc471681691)

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 je jeden ze sto metrových drátů naší staré časomíry někde přerušen, a proto časomíra nefunguje.

Cílem tedy bylo navrhnout časomíru tak, aby se dala co nejsnadněji a nejrychleji nachystat a uklidit. Toho jsem docílil tak, že jsem využil čip ESP8266, který umožňuje komunikaci pomocí Wi-Fi, čímž jsem nahradil porušený sto metrový kabel, který se vždy na začátku tréninku zbytečně dlouho rozmotával a na jeho konci zase smotával. Použití ESP mi také dalo možnost vytvořit k časomíře také webové stránky, které umožňují zobrazení již dosažených časů v podobě grafu, jejich přidávání, mazání a export v podobě CSV.

V této dokumentaci popisuji použití čipu ESP8266, jeho programování pomocí jazyka C++, použití jazyka C++ a knihoven SDL, SDL\_net, SDL\_mixer, SDL\_image, SDL\_ttf pro naprogramování aplikace pro počítač i pro mobil komunikující s aplikací na ESP8266 a webovou aplikaci, která využívá JavaScript a HTML5 Canvas pro zobrazování grafů.

# Teoretická a metodická východiska

## 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.

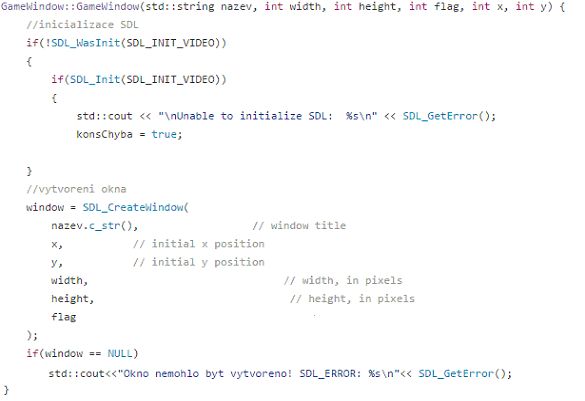
SDL podporuje Windows, Mac OS X, Linux, iOS a Android.

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#.

Ukázka zachycení stisku klávesy v okně pomocí SDL:



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.

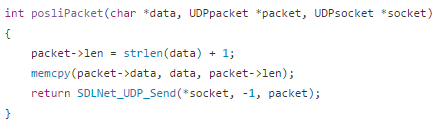
### SDL\_net

SDL\_net je jednoduchá knihovna, která se spolu s SDL používá pro síťovou komunikaci

Přijetí UDP datagramu:



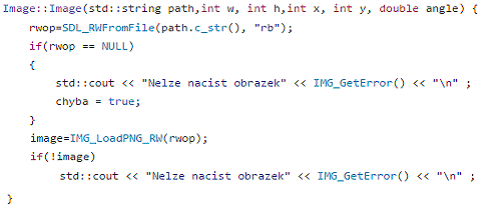
Odeslání UDP datagramu:



### 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.

Načtení png obrázku:



### 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.

### SDL\_ttf

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

# Využité technologie

## 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 Jde 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.

## 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.

### 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.

### 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 C nebo C++ pro mobily.

## Využité programy

### NetBeans

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

### 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.

# Způsoby řešení a použité postupy

## Část časomíry u terčů

Pro detekci stavu terčů jsem použil čip ESP8266 a vodotěsný vypínač na světla. Při sestříknutí terče se zmáčkne vypínač, který rozsvítí žárovku a přes tranzistor připojí jeden z pinů ESP8266 pod přerušením na zem. Při sestříknutí některého z terčů pošle ESP UDP packet s informací, který terč byl sestříknut, pomocí wifi na start. 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.

## 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 zmáčknout enter a zároveň nezastavuje běh programu, pokud nedošlo k žádnému uživatelskému vstupu. Dále se dá program ovládat i pomocí myši, nebo klávesnicí v okně s grafickým uživatelským rozhraním, k čemuž jsem využil knihovny SDL2, SDL\_image pro načítání obrázků a SDL\_ttf pro práci s textem. Pro GUI jsem využil 4 třídy z jednoho mého starého projektu. Aplikace je responzivní (pozice a velikosti tlačítek jsou zadány ve zlomcích aktuální velikosti okna).

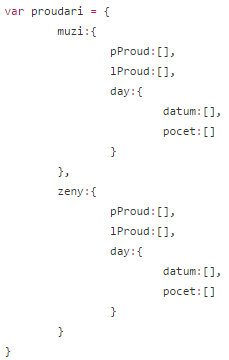
## Webová část aplikace

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.

### Grafy v JavaScriptu

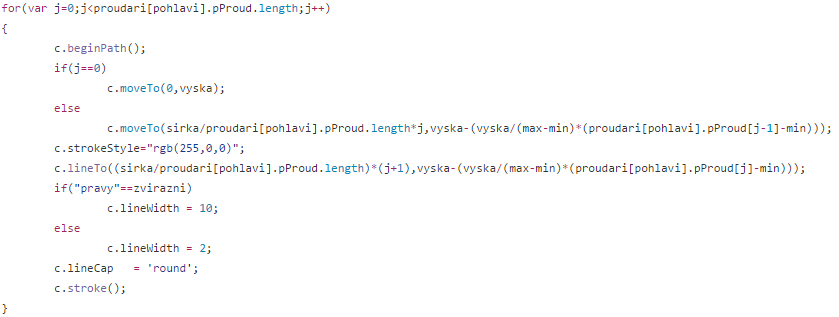
Pro grafy jsem použil část jednoho mého staršího projektu, ve kterém jsem pomocí HTML5 Canvasu a JavaScriptu v grafu zobrazoval výsledky všech týmů z Hasičské ligy Praděd.

Jako zdroj dat pro graf používám JSON s touto strukturou:



Protože se budou webové stránky zobrazovat na zařízeních s různou velikostí displeje, tak musí být responzivní. Dále jsem také chtěl, aby graf vždy zabíral celý canvas, pokud jsou tedy dosaženy například pouze 2 časy, tak graf začne prvním časem úplně vlevo a skončí druhým úplně vpravo. Když se poté přidá 3. čas, tak se původní graf smrští a vpravo se přidá 3. čas. To samé se děje i při dosažení nového nejhoršího, nebo nového nejlepšího času. Díky tomu vznikly při vykreslování 2 docela nepřehledné řádky, které lze vidět na dalším obrázku.

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 myší na jednu z křivek grafu daná křivka zvýrazní (zvýší se její tloušťka), a všechny časy se vypíšou do canvasu pod grafem. Detekci toho, jestli je myš nad křivkou grafu jsem vyřešil tak, že pomocí asynchronní funkce neustále získávám bravu pixelu pod myší a kontroluji jej, 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 dlaší křivku stejné barvy, 20 pixelů širokou s alfa kanálem nastaveným tak, aby pro uživatele nebyla vidět, ale dala se v JavaScriptu detekovat.

Funkce pro detekci pixelu pod myší:



# Výsledky řešení, výstupy, uživatelský manuál

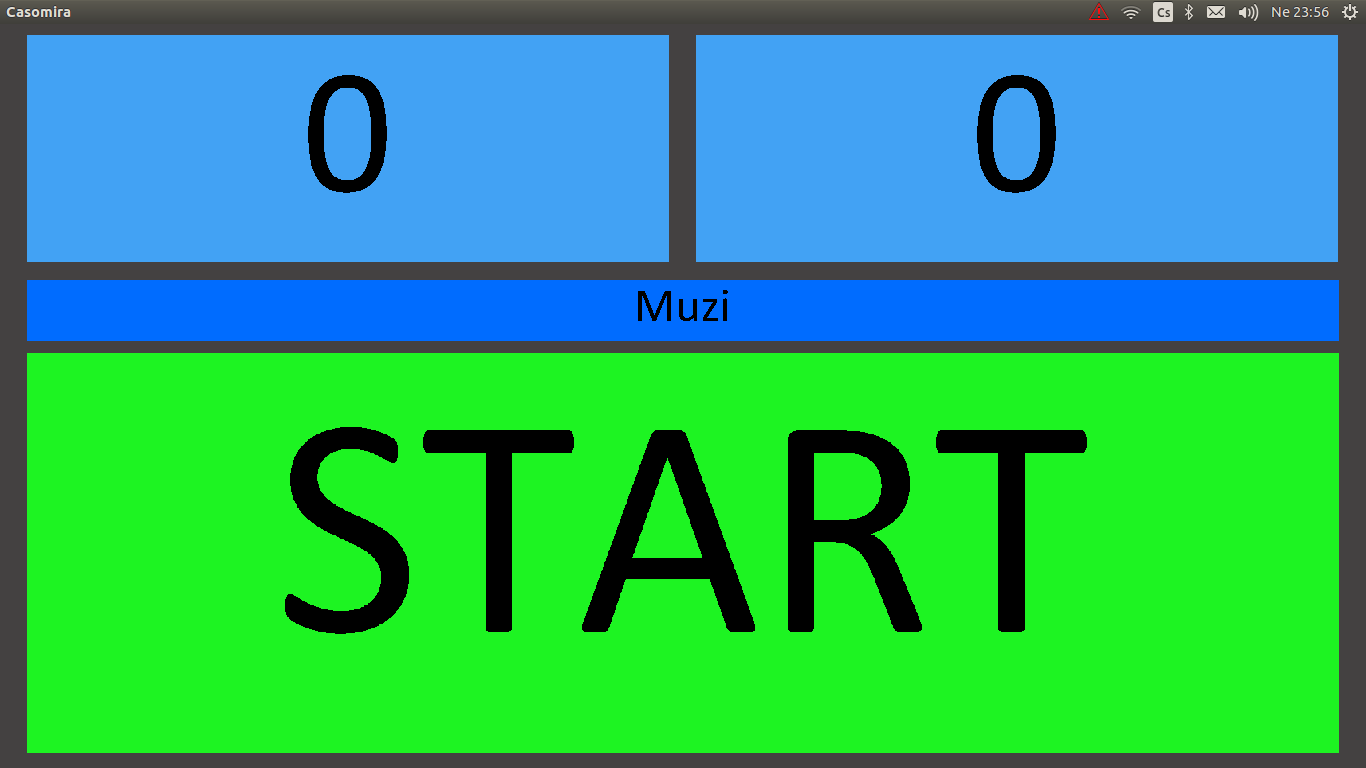
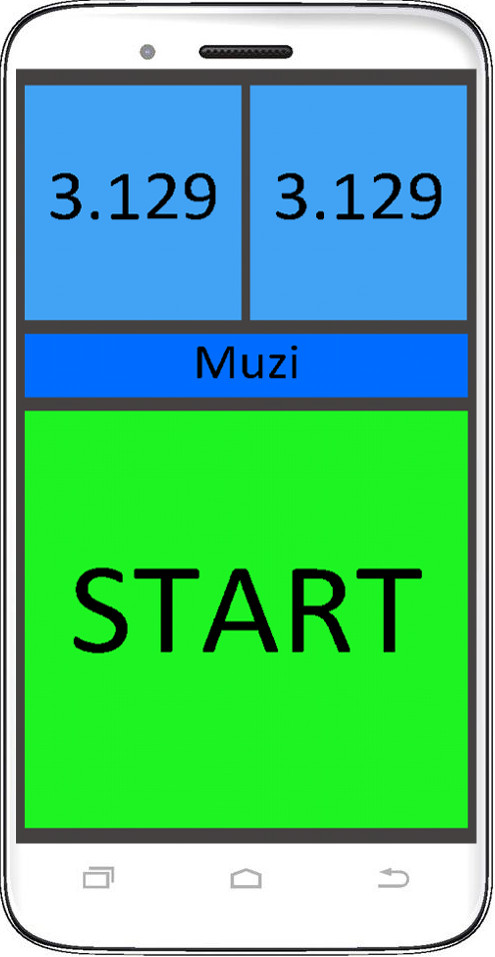
## Ovládání

### 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.

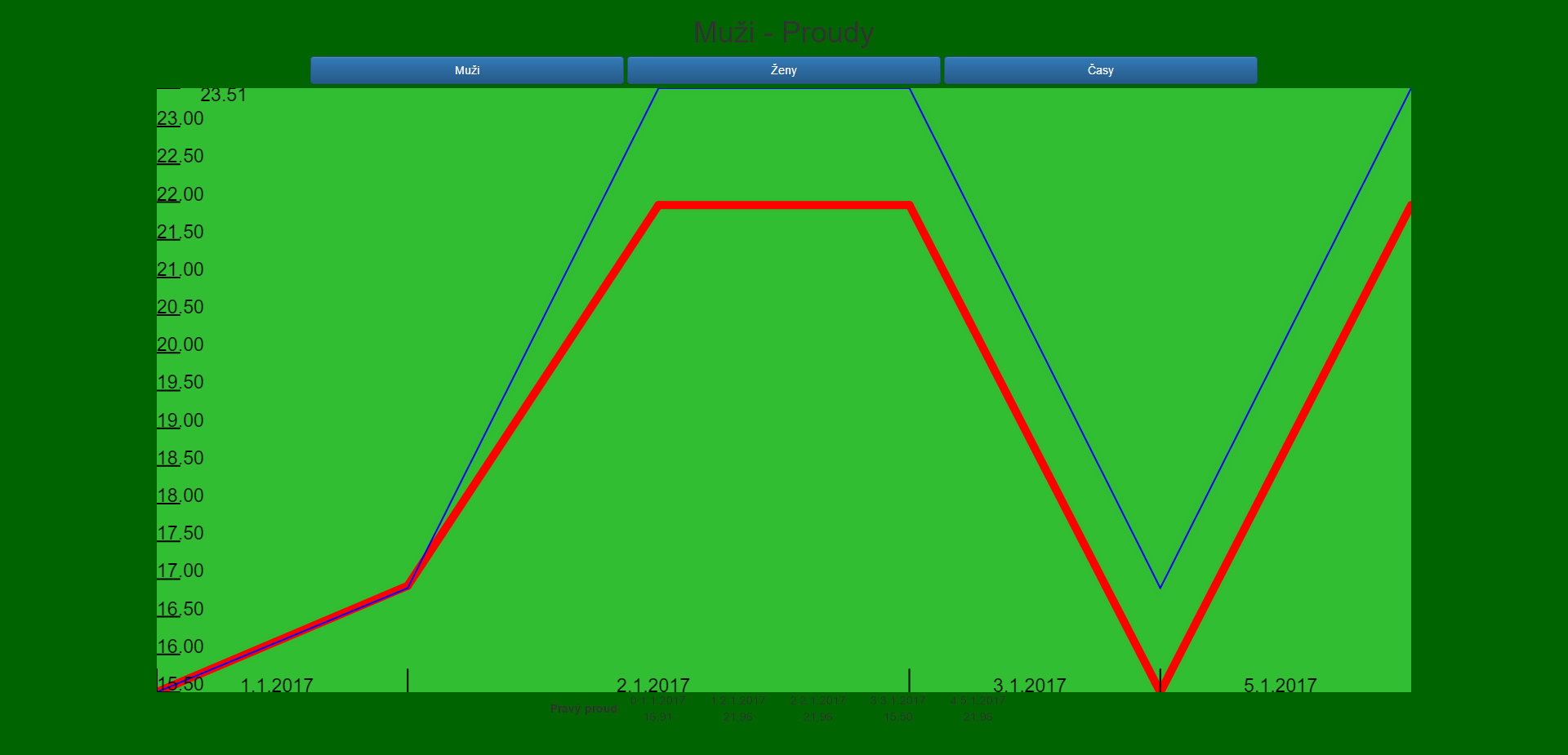
### GUI

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.

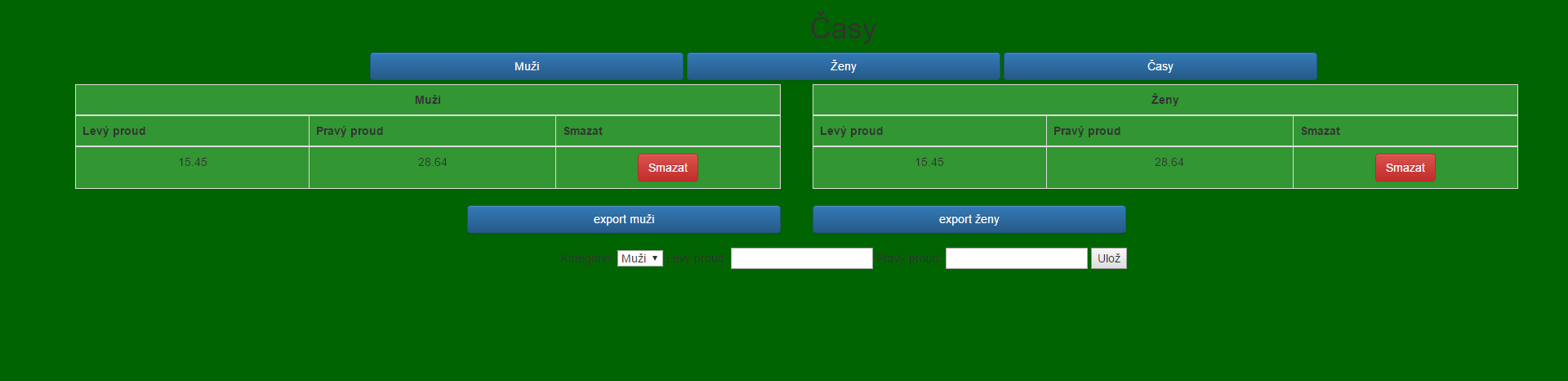


### Web

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.



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.



# Závěr

Cílem práce bylo vytvořit časomíru, která by byla snadnější a rychlejší k použití, než ta stávající. Tohoto zjednodušení jsem dosáhl tak, 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 úspěšně zjišťuje stav terčů. Časomíra je jednodušší a rychlejší na nachystání a prozatím nemá žádný problém se špatným kontaktem, tak jako původní časomíra. Prozatím tedy vše funguje tak jak má, ten pravý test při opravdovém útoku však přijde až na jaře.

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>.