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

Bakalářská práce

Jednoduchý laser game systém

Plzeň 2022 Tomáš Linhart

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš LINHART**
Osobní číslo:	**A18B0255P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Jednoduchý laser game systém**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte problematiku laser game a navrhněte prostředky pro základní hardwarovou a softwarovou realizaci.
2. Vytvořte a naprogramujte zařízení realizující tuto funkci – zařízení umožňující „střelbu“ a personifikovanou detekci zásahů, s možností komunikace, informováním uživatele a řízením hry.
3. Ověřte funkci zařízení na reprezentativním vzorku experimentů a definujte jeho omezení.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Tomáš Mainzer, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

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

V Plzni dne 5. května 2022

Tomáš Linhart

Abstract

Point of this thesis is to design and implement hardware and software used to play LaserGame. In teoretical part of the thesis multiple existing solutions used for playing LaserGame are explored and own solution using mobile data network to transfer data is designed. Optics to create laser beam are designed and it is validated to not cause any harm to players. In second part of the thesis is implemented central server to manage running games and communication, terminal to set up new games, show information about past games and manage connected devices. Hardware based on A9G microcontroler that is used for detecting hits, shooting and interaction with player is designed and developed. In the last part of the thesis, software is tested and test game is played to validate correct functions of the designed solution.

Abstrakt

Tato bakalářská práce řeší návrh a implementaci hardware a software pro hraní hry LaserGame. V teoretické části jsou prozkoumány existující řešení, které se pro hraní LaserGame využívají a je navrženo vlastní řešení s využitím komunikace pomocí mobilní datové sítě pro přenos dat. Je navržena optika tvořící laserový paprsek a je ověřena škodlivost tohoto paprsku na zdraví hráčů. V druhé částí je implementován centrální bod pro řízení hry a komunikace, terminál pro nastavení jednotlivých her, zobrazení výsledků a správu zařízení. Je vytvořen hardware založený na mikrokontroleru A9G, jenž slouží pro detekci zásahů, střelbu a interakci s hráčem. Software je v poslední části práce otestován a je provedena zkušební hra ověřující funkčnost navrženého řešení.

Obsah

1 Úvod

Předmětem této bakalářské práce je prostudovat existující možnosti a způsoby realizace systému pro hru Laser game. Další částí bude prozkoumání možností vlastního HW a SW řešení, prozkoumání limitací, specifikace požadavků na systém a funkční omezení. Systém zvolený v první části práce bude v druhé části implementován. Součástí práce je funkční software a hardware, který bude otestován a připraven ke hraní hry Laser game.

2 Laser game

Laser game (někdy také Laser tag) je hra v reálném světě pro více hráčů. Hráči mají oblečené vesty s elektronikou, která umí pomocí součástek citlivých na světelné záření detekovat zásah zbraní protihráče. Vesty často umí na malém displeji zobrazovat například zbývající čas hry, počet munice v zásobníku, případně koho hráč zasáhl anebo kým byl zasažen. Vesty bývají také doplněny o reproduktor, kterým jsou hráčům sdělovány například informace o zásahu, případně jsou přehrávány různé zvukové efekty, jako například zvuk střelby a jiné. Druhou částí výbavy hráče je zbraň, která "střílí"pomocí laserové diody. Při výstřelu se na malý okamžik rozsvítí laserová dioda ve zbrani a pokud zasáhne součástku citlivou na světelné záření na vestě protihráče, získá hráč bod a zastřelený hráč se musí "dojít oživit", to znamená vyčkat stanovený čas pro oživení anebo splnit jinou podmínku pro oživení. Vzhledem k tomu, že pro detekci zásahů se používá světelné záření, tak se Laser game hraje převážně v uzavřených prostorech s nízkou úrovní osvětlení, kdy dosah laserového paprsku není příliš ovlivněn viditelným světlem. Tyto uzavřené prostory se nazývají laser arény a často se v nich kromě Laser game nacházejí další možnosti vyžití, jako například restaurace a bar, únikové hry, závodní simulátory a další.

Laser game je možné hrát mnoha různými herními režimy. Nejběžnějším je pravděpodobně režim Deathmatch. Tento herní režim si lze představit jako hru "všichni proti všem", kdy cílem je zastřelit nejvíce ostatních hráčů a přitom být zastřelen co nejméně krát. Poté co je hráč zasažen je typicky nutné pár sekund vyčkat dokud se opět automaticky neoživí a nemůže pokračovat ve hře. Týmovou alternativou je Team Deathmatch, kdy jsou hráči rozdělení do dvou, či více, týmů a cílem je zastřelit nejvíce hráčů z ostatních týmů. Dalším herním režimem je Capture the flag, kdy hráči jsou rozděleni na týmy. Každý tým má přidělený předmět, typicky vlajku. Cílem je ukořistit tuto vlajku od nepřítele, přičemž je zároveň nutné ubránit svoji vlajku od ukořistění protivníkem. Po ukořistění vlajky a donesení do vlastní základny hra končí, vítěznému družstvu se připisuje bod, vlajka se vrátí druhému družstvu a hra se opakuje. Herních režimů je samozřejmě více, například Bomba, King of the hill a další. Tyto režimy ovšem nejsou tak populární jako výše zmíněné.

Ve hře se konají i profesionální turnaje. V Česku a na Slovensku se hraje Mistrovství Česka a Slovenska v Laser Game. Mistrovství se mohou zúčastnit čtyřčlenné týmy, které se v kvalifikačních zápasech utkají s ostatními týmy

v jedné z šestnácti Laser game arén, které jsou do mistrovství zapojené. Vítězové z kvalifikačních zápasů se poté utkají na dvoudenním mistrovství, přičemž vítěz mohl v roce 2019 vyhrát až 100 000 Kč [10].

3 Existující řešení

Na internetu je možné najít mnoho již existujících podomácku vytvořených (dále "DIY") řešení, která si jednotlivci vytvářeli doma. Tato řešení často jsou pouze typu "proof of concept"a nejsou použitelná pro produkční prostředí. V jedné z prací dostupných na internetu autor s přezdívkou MacDynamo detailně vysvětluje přípravu elektrického zapojení laseru zbraně a senzoru zásahu, avšak tento systém slouží pouze pro jednoduchou střelbu a detekci zásahu. Systém neumožňuje identifikaci střelce ani komunikaci mezi zbraněmi/vestami a dalšími [18]. Součástí práce není ani například model zbraně a návod na přípravu vesty. Tento návod tedy není vhodný pro tuto práci.

Další prací, která se zabývá vlastním řešením pro hru Laser game je LZRTag dostupná z webových stránek [24]. Původně tato práce využívala vlastních obvodů, založených na mikroprocesoru ATmega328P. Postupně se ale autor rozhodl pro změnu mikrokontroleru na ESP-32 [23]. Funkčním principem je u této práce propojení vest a zbraní pomoci WiFi. Zbraně a vesty mezi s sebou komunikují s využitím protokolu MQTT. Tento protokol je široce rozšířený pro použití v sítích internetu věcí (dále "IoT") a pomocí vytvořených kanálů je možné snadno a s nízkou odezvou odesílat a přijímat data. Zbraně odesílají statistické informace na server, kde je možné je dále zpracovat a zobrazit. Server je připravený v jazyce Ruby a komunikuje také pomocí protokolu MQTT. Toto řešení je mnohem obsáhlejší, obsahuje detekci zásahů s identifikací hráče, audiovizuální zprostředkování informace hráči, centrální bod (server), soubory pro přípravu vlastních desek tištěných spojů, modelů zbraní a dalších.

Poslední prací, kterou zmíním je práce OpenLaserTag [15]. Tato práce jde ještě více do hloubky, kdy pro přenášení dat má připravený vlastní protokol, kde data přenáší s pomocí infračervené diody anebo řešení pomocí různých mikroprocesorů, jako například STM32 anebo procesoru založeném na architektuře ARM. Autorem této práce je Čech Tomáš Krejčí, informace jsou tak dostupné v českém i v anglickém jazyce [16].

4 Porovnání implementace oproti existujícím řešením

Hlavním rozdílem mé implementace je záměr pro využití tohoto systému ve venkovních prostorech. Hlavním problémem tohoto záměru je detekce zásahu a přenos informací mezi zbraněmi a vestami hráčů. Z důvodu nutnosti přesnosti detekce zásahu a ověření, zda se hráč nenachází za překážkou, není možné využít například souřadnic GPS a určení směru pomocí gyroskopu ve zbrani. Jako jediný použitelný způsob se tedy jeví využití světelného paprsku - laseru. Práce využívají buď běžný červený anebo zelený laser, práce OpenLaserTag využívá infračervených diod. Tyto diody se běžně používají například v dálkových ovladačích. Řešení s infračervenými diodami se jeví jako nejvhodnější, vzhledem k tomu, že existují senzory citlivé pouze na infračervené záření. Pro použití infračervené diody je avšak nutné sestrojit optiku, která vytvoří kolimovaný paprsek infračerveného záření. Bez této optiky by se paprsek rozšiřoval, jeho dosah by byl velmi zkrácený a zároveň by zasáhl i hráče, kteří se nacházejí mimo jeho střelnou dráhu. Vzorce pro výpočet ideální vzdálenosti diody od senzoru a návody pro sestavení optiky jsou veřejně dostupné na stránkách OpenLaserTag [17].

Dalším rozdílem vyplývajícím ze hry ve venkovním prostředí je použití mobilní sítě na rozdíl od WiFi používané v ostatních implementacích. Hráči se pravděpodobně budou pohybovat po větším území, které by bylo obtížné pokrýt WiFi signálem. Pasivní komunikace (Bez aktivního připojení - například dekódování jména hráče z dat přenesených při výstřelu) je pro moji implementaci nevhodná z důvodu nutnosti přenosu stavových informací, jako například start hry, ukončení hry a zásah hráče.

Využití komunikace prostřednictvím mobilních sítí přináší další změnu oproti ostatním pracím a to využití hlavního centrálního bodu. Tímto bodem pravděpodobně bude hlavní server (dále "Server"), na který budou připojeny veškeré zbraně a stanice pro nastavení hry (dále "Terminál"). Pro zjednodušení může Server mít pouze funkci směrovače, kdy bude jen předávat data mezi zbraněmi a terminálem, ale nebude s nimi nikterak manipulovat ani je ukládat.

Herní pole mohou být velmi rozsáhlá a proto se jako vhodný nápad jeví využití GPS pro sledování polohy hráčů pomocí GPS. Na konci hry je například možné v terminálu zobrazit mapu, kde se hráči pohybovali anebo například připravit mapu, ve které jsou zvýrazněny oblasti, ve kterých se

hráči pohybovali nejvíce - tzv HeatMapy. Tyto informace je poté možné využít například pro strategické plánování v dalších kolech hry. Vzhledem k tomu, že venkovní pole je těžké ohraničit a tím pádem zabránit hráčům, aby herní pole opouštěli, je možné sledování GPS použít také pro tento účel.

5 Nároky na HW

5.1 Výběr mikrokontroleru

Z hlediska výbavy hráče je nutné vlastního řešení založeného na některém z komerčně dostupných mikrokontrolérů. Nejpopulárnějším mikrokontrolérem mezi komunitou "bastlířů"je bezpochyby Arduino. Jeho nejběžnější verze Arduino Uno je založena na mikroprocesoru ATmega328P. Frekvence tohoto mikroprocesoru je 16 MHz a je doplněný o 32 KB paměti Flash a 2KB paměti SRAM. Data pro uchovávání po odpojení napájení lze uložit do EEPROM o velikosti 1KB. K mikrokontroléru lze připojit periferie pomocí 14 digitálních pinů (z toho 6 podporuje pulzně-šířkovou modulaci, dále "PWM") a 6 analogových pinů s provozním napětím 5V [11]. Vzhledem k nízkému výkonu, malé paměti a tomu, že tento kontrolér neobsahuje žádné bezdrátové komunikační řadiče (Bluetooth, WiFi, GPRS) a GPS není vhodný pro tuto implementaci.

Často používanou alternativou jsou mikrokontroléry založené na platformě ESP32. Tyto mikrokontroléry využívají až dvoujádrové procesory s frekvencí 80 až 240 MHz. Podporují bezdrátový přenos dat přes WiFi a Bluetooth ve verzi 4.2 a Bluetooth Low Energy (dále "BLE"). Piny mikroprocesoru jsou často sdružené pro více funkcí, například pro protokoly I^2C pro přenos dat mezi senzory, I^2S pro přenos zvuku, kapacitní vstupy pro dotykové senzory, PWM a další [22]. Tento kontrolér je vhodnější možností než mikrokontrolér Arduino Uno, přesto však neobsahuje periferie pro GPRS a GPS, proto v této práci nebude využitý.

S vedoucím práce jsme zvolili řešení založené na mikroprocesoru A9G a A9 od společnosti AI-Thinker. Součástí mikroprocesoru A9G je modul pro zjištění lokace pomocí GPS (A také čínské alternativy Beidou) a zároveň slot pro SIM kartu a modem pro komunikaci pomocí sítí GPRS. Na desce pudding board využívané pro tuto práci se nachází 29 pinů pro vstup a výstup. Součástí balení jsou GPRS a GPS antény, které se dají připojit k mikrokontroléru pomocí konektorů. Mikroprocesor běží na frekvenci až 312 MHz, je doplněný 32 Mb Flash paměti a 32 Mb RAM. Součástí je možnost využití mnoho sběrnic - například SPI, l^2C a další. Jediným rozdílem mezi procesory A9G a A9 je absence GPS modulu u procesoru A9 [4].

5.2 Výběr LED diod a ochrana zraku hráčů

Při výběru systému pro detekci zásahů je nutné brát v potaz ochranu zdraví hráčů. Bezpečnosti ohledně používání laserů se věnuje Evropská norma IEC 60825-1, která lasery rozděluje do jednotlivých tříd. Třída 1 je pro lasery s velmi nízkým výkonem, které jsou bezpečné pro pohled okem po dobu 100s i s použitím optických přístrojů (dalekohledu, lupy a dalších). Naopak Třída 4 slouží pro výkonové lasery, které jsou nebezpečné pro oči i při zásahu odraženým paprskem po velmi krátkou dobu a mohou také způsobit popálení kůže[2][20]. Tyto lasery jsou využívány převážně v lékařském a armádním průmyslu. Pro tento projekt ovšem nebude využita optika, která vytvoří laserový paprsek, nýbrž nerozbíhavý (kolimovaný) paprsek světelného záření o průměru dané čočky. Norma IEC 60825-1 stanovuje pojem *Maximum permissible exposure* (volně přeloženo jako "Maximální dovolená hodnota expozice záření", dále "MPE"), která určuje maximální bezpečnou dobu vystavení záření o daném výkonu na cm^2.

Pro detekci zásahů se z důvodu rušení viditelným světlem jeví jako nejvýhodnější využít infračervených světlo-emitujících diod (dále "LED") a fotodiod citlivých na záření dané vlnové délky. Nejvhodnější infračervenou LED je typ TSAL6100 od výrobce Vishay. Tato LED vyzařuje záření o vlnové délce 940nm a o výkonu 130mW. Záření je vyzařováno pod úhlem pouze 10°, což je vhodné pro sestrojení optiky pro zaostření paprsku pomocí spojné čočky. Pokud bychom uvažovali kolimační čočku o průměru 50mm, vypočítáme výkon na cm^2 pomocí vzorce

$$W/cm^2 = 0,130/(\pi * 2,5^2) \tag{5.1}$$

Výsledná hodnota je 0,006 W/cm^2. Při porovnání s tabulkou zjistíme hodnotu MPE přibližně 0.5s. Tento čas by měl být dostatečný pro vyslání krátkého signálu modulovaného na frekvenci 38 kHz.

Nespornou výhodou je její nízká cena za kus v řádu korun českých [13].

Pro detekci bude využit přijímač infračerveného záření TSOP4438. Tento senzor je citlivý na záření o vlnové délce 940nm, což přesně odpovídá záření vyzařovaném LED zmíněnou výše. Cena tohoto senzoru se také pohybuje v řádu korun českých [12].

5.3 Hardware pro komunikaci s uživatelem a napájení

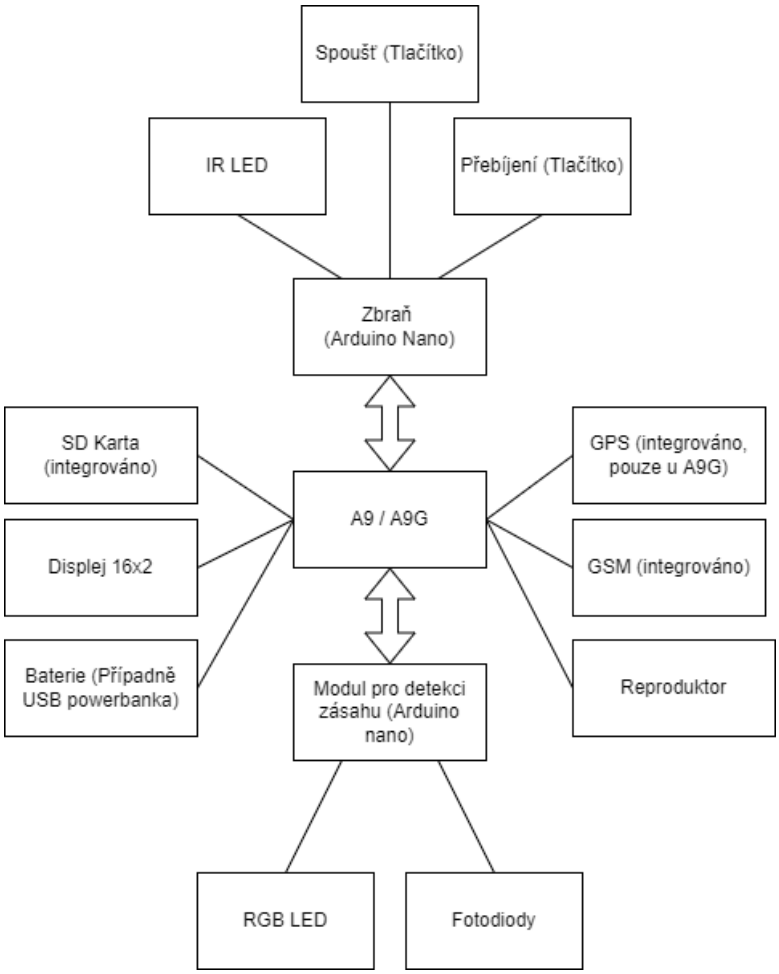
Pro komunikaci s uživatelem bude použitý běžný reproduktor, I2C 16x2, 20x2 anebo 20x4 displej a barevné adresovatelné LED typu WS2812.

Vzhledem k tomu, že se hráči budou pohybovat mimo zdroj energie, je nutné zbraně a vesty napájet pomocí baterií. Nejvhodnější volbou jsou Li-Ion články, které jsou vyráběny s kapacitou okolo 3000 mAh a napětím okolo 3,6V. Články je dle potřeby možné spojit do série pro zvýšení napětí anebo paralelně pro zvýšení kapacity. Je možné je dobíjet a v případě potřeby jsou snadno vyměnitelné. Jejich kapacita je pro několik her dostačující a nízká hmotnost nezpůsobí hráčům problémy při pohybu na herním poli. Další volbou je napájení pomocí USB konektoru a použití USB powerbanky. Toto řešení představuje jednodušší alternativu vůči samostatným článkům, jelikož není třeba navrhovat obvod pro řízení nabíjení a vybíjení baterie. Výhodou také je rychlá výměna powerbanky za jinou, pokud se první mezi hrami vybije.

5.4 Rozdělení hardware na moduly

Hardware bude rozdělený na jednotlivé moduly zajišťující detekci zásahu anebo střelbu. Tyto moduly budou řízeny pomocí mikrokontroleru Arduino Nano a s hlavním mikrokontrolerem A9 budou komunikovat pomocí sběrnice I2C. Většina hardwarových součástek bude umístěna ve vestě, kterou bude mít hráč oblečenou. U zbraně se nachází pouze spoušť, tlačítko pro přebití zásobníku a infračervená LED. Fotodiody jsou umístěny na modulech pro detekci zásahu. Všechny moduly jsou připojeny pomocí I2C sběrnice k mikrokontroleru A9G, který bude umístěna na vestě. Souběžně s I2C sběrnicí vede vodič pro přenos interrupt signálů mezi moduly. Umístění veškerých komponent je znázorněno na obrázku 5.1.

Nevýhodou rozdělení jsou vyšší náklady na vytvoření kompletního hardware. Pro jednoduchost cenu veškerých položek, jejichž cena je v řádu korun, zaokrouhlíme na 150 Korun českých. Uvažujeme-li cenu modulu A9G přibližně 600 Kč, cenu I2C displeje 100 Kč a cenu mikrokontroleru Arduino Nano 250 Kč, pak jedna vesty s dvěma moduly pro detekci zásahu a jedním modulem pro střelbu stojí uživatele 1600 Korun českých. Tuto cenu je možné snížit nákupem součástek z tržiště ebay.com. Přidání každého dalšího modulu pro detekci zásahu stojí přibližně 300 Korun českých.



Obrázek 5.1: Blokové schéma zapojení a umístění HW komponent

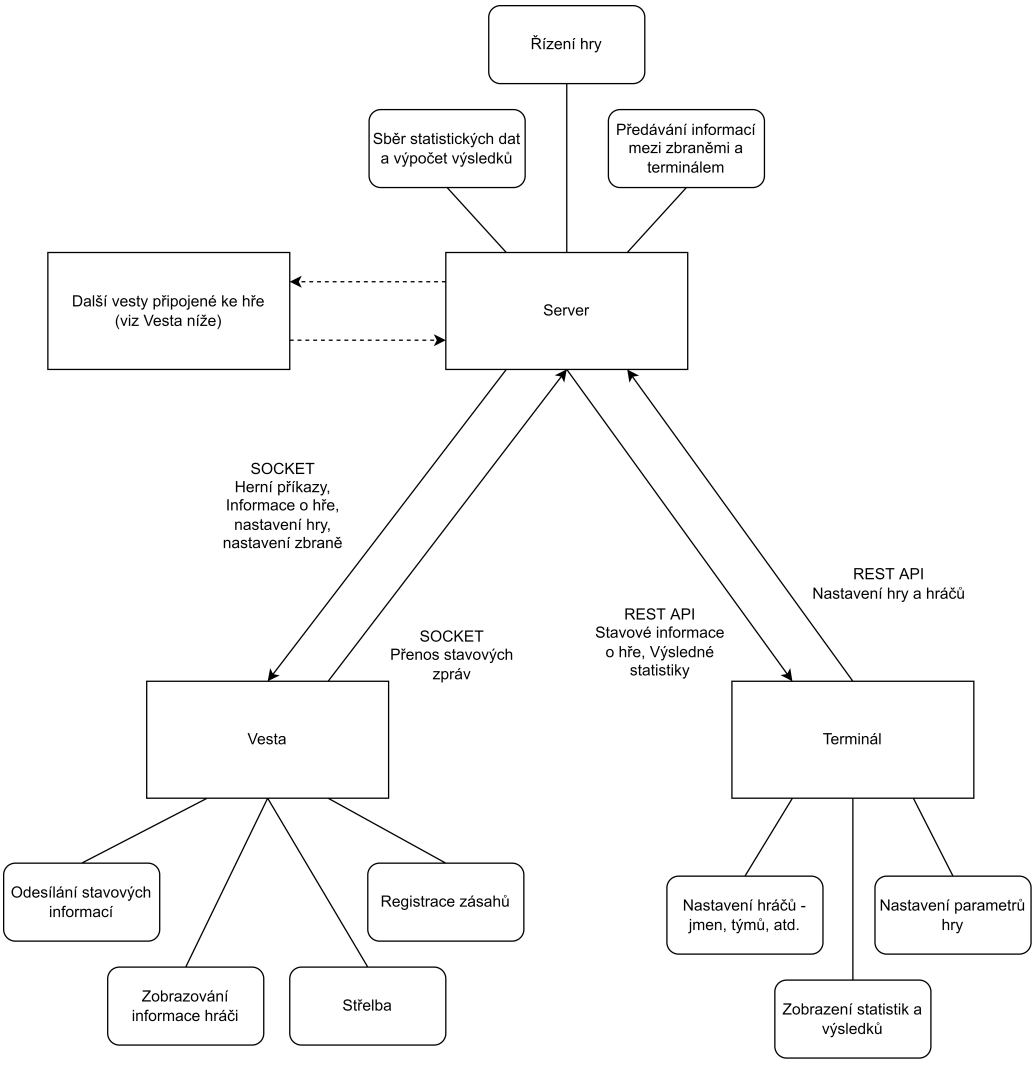
5.5 Hardware požadavky pro Terminál a Server

Pro terminály bude postačující webový prohlížeč v počítači, případně tabletu anebo mobilním telefonu. Terminál bude sloužit pouze pro nastavení parametrů hry, případně zobrazení výsledných statistik vygenerovaných Serverem. Pro jednoduchost nastavení je doporučeno využít notebook, anebo jiný počítač s monitorem, myší a klávesnicí, na kterém bude nainstalována terminálová aplikace. Dalším vhodným řešením je tablet s operačním systémem Android, anebo využití počítače typu All-In-One (dále "AiO") s dotykovou obrazovkou. Vstupní údaje by poté byly zadávány pomocí virtuální klávesnice, která je v systému Windows 10 podporována. Pro komunikaci se serverem bude nutné zajistit internetovou konektivitu.

Pro realizaci této práce budou vytvořeny pouze dva kusy vest se zbraní ve stavu proof-of-concept. Toto množství nepředstavuje velkou zátěž na výpočetní výkon a paměť serveru. Zátěž internetového připojení je minimální, mezi zařízením budou přenášeny pouze krátké stavové a příkazové zprávy v intervalu řádu sekund. Data spotřebovaná režií REST API, anebo využitého protokolu Socket je minimální.

6 Návrh struktury SW

Veškeré zařízení komunikují pouze s centrálním bodem - serverem. Komunikace mezi serverovou částí a vestami probíhá pomocí protokolu Socket, který zajišťuje přenos dat v reálném čase. Terminály se Serverem komunikují pomocí REST API. Vesta musí při připojení být registrována některým z uživatelů. Při přihlášení se zařízení přihlásí pomocí jedinečného identifikátoru. Uživatel poté může do hry vybrat libovolné svoje vesty. Samotné směrování toku dat je úkolem Serveru. Základní úkoly jednotlivých zařízení dat, datový protokol a směr toku dat mezi jednotlivými částmi je znázorněn na obrázku 6.1.



Obrázek 6.1: Základní činnosti jednotlivých zařízení a toky dat mezi nimi

6.1 Zbraně a vesty

Hlavním úkolem softwaru ve zbraních a vestách je detekce zásahů a střelba na protivníky. Při startu hry zbraně obdrží informace o ID hráčů, Jménech a Týmu ze Serveru. V softwaru je nutné zakódovat data přenášená mezi jednotlivými zbraněmi a vestami a přenést je pomocí infračerveného záření. Mezi zařízeními se přenese ID střelce, pomocí kterého vesta z informací přenesených na začátku hry získá jméno a tým protivníka, který daného hráče zasáhl. Vesta může dle nastavení hry v terminálu měnit různé parametry, jako například limitovat počet nábojů a měnit počet životů hráče.

Vesta v periodických intervalech na Server zasílá stavové informace pomocí protokolu Socket. Tato data obsahují primárně polohu hráče zapsanou pomocí GPS souřadnic. Vesta dále zasílá například informace o zásahu jiným hráčem.

Pro pomoc s programováním mikroprocesoru je výrobcem poskytnuta dokumentace, SDK a příklady v jazyce C. Součástí SDK jsou knihovny pro přístup k jednotlivým periferiím a funkcím čipu A9G.

Alternativou k programování v jazyce C je verze programovacího jazyka MicroPython pro čip A9G, která je volně ke stažení ze serveru Github [21]. Tato verze obsahuje již připravené knihovny pro snadné připojení k mobilní síti (cellular), připojení k serveru pomocí socketu (socket) anebo získání GPS souřadnic (gps). Nevýhodou je, že knihovna je delší dobu neudržovaná a spuštění MicroPython kódu na procesoru bude mít dopad na jeho výkon. Z tohoto důvodu je kód vest naprogramovaný v jazyce C s využitím zmíněného SDK.

6.2 Terminál

Terminál slouží primárně pro nastavení parametrů dané hry. Aplikace bude umožňovat nastavení přezdívek hráčů a přidělení týmů, počtu životů, délku hry a dalších možností. Po spuštění hry bude možné sledovat průběh aktuální hry a poslední pozice hráčů. Okno terminálu bude aktualizováno průběžně pomocí metody short-polling. Tato metoda spočívá v periodických dotazech na server, který v odpovědí zasílá aktuální informace. Z terminálu bude možné běžící hru ukončit. Po ukončení hry se objeví statistiky, výsledky poslední hry a kompletní mapa se všemi pozicemi hráčů.

Terminál bude naprogramovaný pomocí HTML a Javascriptu. Pro zvýšení rychlosti a snížení objemu přenesených dat budou asynchronně načítány pouze funkční komponenty pomocí AJAX volání. Externí Javascript knihovny budou načteny pouze jednou při prvním otevření webové stránky.

6.3 Server

Server je hlavním spojovacím bodem celého systému. Server po spuštění vyčkává na připojení zbraní a terminálu. Po připojení předává data mezi terminálem a vestami. Předává stavové zprávy od zbraní zpět k terminálu a uchovává je pro finální statistiky, které po ukončení hry předá terminálu.

Úkolem Serveru je také řízení hry, jako například ukončení hry po uplynutí času, anebo při žádosti z terminálu.

Veškeré akce bude Server schopný provádět paralelně pro více uživatelů nezávisle na sobě. Na Serveru bude ošetřeno, aby nedošlo k ovlivnění jedné hry jinou hrou. Veškerá zařízení budou moci být přihlášeny pouze do jedné hry zároveň a uživatel bude moci mít spuštěnou pouze jednu hru v daný moment.

Historická data a data o uživatelích a stanicích budou uchovávána v databázi Microsoft SQL. Sever bude vytvořený v jazyce Java s pomocí frameworku Spring Boot. Pro komunikaci s databází bude využito knihovny Hibernate, která je součástí frameworku Spring. Pro vytvoření Socket serveru bude využita běžná implementace *java.net.ServerSocket*, dostupná v Java API.

7 Řízení hry, komunikace s uživatelem

Pro řízení hry bude sloužit hlavní Server. Jeho úkolem bude přenášet, validovat a zpracovávat veškerá data, která jsou pro hru potřebná. K Serveru budou zbraně přistupovat pomocí socketů. Terminál bude pro komunikaci využívat REST API. Použitím těchto technologií budou zajištěny reakce téměř v reálném čase. Před hrou bude nutné nastavit parametry hry - hráče a jejich vesty, týmy, počet munice a životů a další. Tyto informace budou přeneseny na Server, který je předá hráčům do jejich vest. Pomocí terminálu je možné spustit a zrušit hru, informace budou opět okamžitě přeneseny do vest, které budou hráče informovat. Při hře budou vesty na Server pravidelně zasílat stavové zprávy, jako například informace o poloze hráče. Aktuální polohu hráčů bude možné v reálném čase pozorovat na terminálu, případně zobrazit kompletní mapu na konci hry.

Vesty budou s uživateli komunikovat pomocí audiovizuálních efektů. Použitý mikroprocesor A9G umožňuje přehrávání zvukových stop z SD karty. Pro signalizaci zásahu, zvuk výstřelu a další efekty lze této funkce využít. Na vestě budou umístěny adresovatelné barevné LED, kterými bude signalizovaný tým, ve kterém se hráč nachází a zda je naživu. Rozsvícené LED také slouží pro snadnější identifikaci cíle. Dalším způsobem předávání informací bude znakový displej o 16 (případně 20) sloupcích a 2 (případně 4) řádcích. Na tomto displeji bude primárně zobrazený stav munice, zásahy soupeřů, případně jiné stavové informace (začátek/ukončení hry, atd..).

8 SDK a podpůrný software

8.1 SDK pro vývoj A9G

Pro vývoj aplikací pro čip A9G je na webu dostupný Software Development Kit (Volně přeloženo jako Sada pro vývoj softwaru, dále SDK) včetně knihovny pro vývoj aplikace a příkladů využití této knihovny. Toto SDK je ke stažení z Github repozitáře výrobce čipu A9G Ai-Thinker [7]. Podrobná dokumentace je dostupná na Github webové stránce vývojáře [6]. Sestavení výsledného souboru, který slouží k nahrání do čipu A9G se provádí pomocí skriptu *build.bat*, který je součástí SDK. Tento skript sestaví výsledný program a uloží do souboru formátu *lod* podle specifikace v souboru *Makefile*, který je součástí projektu a který obsahuje informace o názvu programu, umístění knihoven a další. Sestavený program se do mikroprocesoru nahrává přes USB-to-TTL převodník pomocí aplikace Coolwatcher, která je součástí SDK CSDTK4.2, které je dostupné ke stažení ze služby pro sdílení souborů mega.nz [5]. V době psaní této práce bohužel odkaz pro stažení ze webových stránek vývojáře nebyl funkční.

8.2 Apache Maven

Apache Maven je nástroj pro sestavení aplikací a jednodušší správu závislostí aplikací. Pomocí tohoto nástroje lze automatizovaně podle předpisu v souboru *pom.xml* sestavit, spustit a otestovat danou aplikaci. Tento nástroj je použit v projektu Serveru. Používá se pro sestavení, otestování a pro generování Java struktur z OpenAPI. Proces sestavení se skládá z jednotlivých, předem definovaných kroků. Tyto kroky, jako například generování zdrojů, testování anebo přeložení kódu lze spouštět samostatně, případně je lze z procesu sestavení vynechat.

8.3 Gitlab a Jenkins

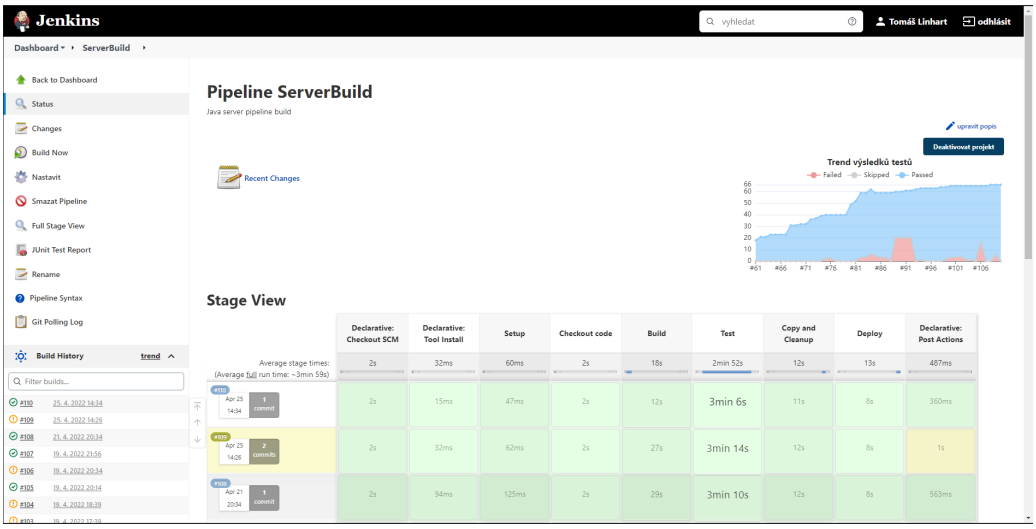
Celkový vývoj je rozdělený do pěti samostatných projektů a to Serveru, Terminálu, programu pro mikrokontroler A9G, programu pro modul pro střelbu a programu pro modul pro detekci zásahu. Kód těchto projektů je uložený na serveru Gitlab, který zároveň zajišťuje uložená historie změn v daném projektu.

Pro jednodušší sestavení a nasazení Serveru a Terminálu je připravena aplikace Jenkins [1]. Ten slouží k automatizaci procesů ve vývoji softwaru. Umí sestavit, spouštět a testovat projekty využívající systémy pro správu projektů jako Apache Ant, Apache Maven a další. Systém je open-source a je dostupný pod licencí MIT. Během vytváření této práce je Jenkins spuštěný na dedikovaném serveru, na kterém je nainstalovaný i Server a webový hosting na platformě XAMPP pro Terminál. Tento server je dostupný pouze v lokální síti vývojáře.

Předpis pro jednotlivé kroky automatizace se nazývá pipeline a je typicky uložen v souboru Jenkinsfile v kmenovém adresáři projektu. Pro potřeby této práce jsou programem Jenkins zpracovávány dva projekty - Terminál a Server. Jenkins periodicky kontroluje Gitlab repozitáře těchto dvou projektů a v případě změny automaticky spustí pipeline daného projektu. Ostatní projekty jsou sestavovány manuálně na počítači vývojáře, jelikož je nutné zajistit nahrání sestaveného kódu do mikrokontrolerů pomocí USB-to-TTL převodníku. Server, na kterém je spuštěna aplikace Jenkins je umístěn v serverovně a připojení USB periferií je nepraktické.

8.3.1 Popis pipeline pro sestavení Serveru

Prvním krokem v pipeline je stažení repozitáře z Gitlabu. To zajistí, že kód, který je právě sestavovaný je nejaktuálnější verze. Druhým krokem je samotné sestavení projektu pomocí Apache Maven. Během tohoto kroku dojde k vygenerování výsledného jar souboru, který lze spustit na jakémkoliv počítači, na kterém je nainstalována Java. V dalším kroku dojde k spuštění jednotkových a integračních testů. Výsledky z testování jsou zobrazené na stránce projektu v přehledném grafu. Detaily pro každý test, který selhal, jsou dostupné v podrobnostech o sestavení projektu. V posledním kroku dojde ke kopírování posledního sestaveného jar souboru do adresáře, ze kterého se spouští služba Serveru. Tato služba je poté restartována a tím je zajištěno, že je právě spuštěna nejaktuálnější verze Serveru.



Obrázek 8.1: Hlavní obrazovka projektu Server na Jenkins

8.3.2 Popis pipeline pro sestavení Terminálu

Pipeline pro sestavení terminálu je oproti Serverové pipeline o poznání jednodušší. Hlavním úkolem této pipeline je připravit ZIP archiv, který slouží pro uložení jednotlivých verzí aplikace. Při přípravě archivu je soubor *config.js* nahrazený souborem *config.prod.js*, který obsahuje konfigurační hodnoty použité v produkčním prostředí. Poslední verze Terminálu je potom překopírována do složky, ze které je Terminál hostovaný a tím pádem je v produkčním prostředí dostupná nejnovější verze.

9 Realizace vest

9.1 Návrh a realizace hardware

Hardware vest je rozdělený do více samostatných modulů. Každý typ modulu slouží k jinému účelu, jako například detekci zásahů anebo střelbu. Výhodou modulárního přístupu je snadná škálovatelnost, kdy je jednoduše možné připojit další moduly, snadná náhrada poškozeného modulu za nový a jednodušší kód v daných modulech. Nevýhodou jsou vyšší pořizovací náklady, mírně vyšší spotřeba elektrické energie a režie při předávání dat po datové sběrnici.

Moduly mezi sebou komunikují pomocí protokolu I2C. Hlavní charakteristikou tohoto protokolu je nutnost hlavního zařízení (dále "master") a vedlejších zařízení (také podřízených, dále "slave"). Hlavním zařízením, které řídí tok dat na sběrnici je modul s mikrokontrolerem A9G. Aby ostatní moduly mohly předávat stavové zprávy do hlavního modulu, souběžně s datovou sběrnicí vede sběrnice pro přerušení. Součástí této sběrnice je vodič pro standardní přerušení a vodič pro přerušení při střelbě. Výchozí napěťová úroveň představuje logickou hodnotu 1 (5V, případně 3V - viz 9.1.4 Datová sběrnice, dále také "HIGH"). Při žádosti o zaslání zprávy modul nastaví logickou hodnotu na 0 (0V, dále také "LOW"). Hlavní modul detekuje změnu úrovně napětí na sběrnici přerušení a spustí obsluhu přerušení. Při vykonávání obsluhy se postupně zjistí od všech modulů, zda mají nějaké informace, které předají hlavnímu modulu. Moduly na dotaz z hlavního modulu odpovídají příkazem (viz 9.2.4 Příkazy předávané mezi moduly), který jednoznačně definuje akci, kterou má hlavní modul vykonat. Součástí příkazu mohou být i další doplňující data.

Na stejné sběrnici je také připojený displej, do kterého jsou pomocí I2C protokolu přenášena data, která se na něm zobrazují. Data zobrazená na displeji jsou periodicky aktualizována. Zvukové informace jsou pro demonstraci funkčnosti přenášeny pouze pomocí malého piezo reproduktoru. Tento reproduktor má velmi malý výkon a tudíž není nutné využití audiozesilovače.

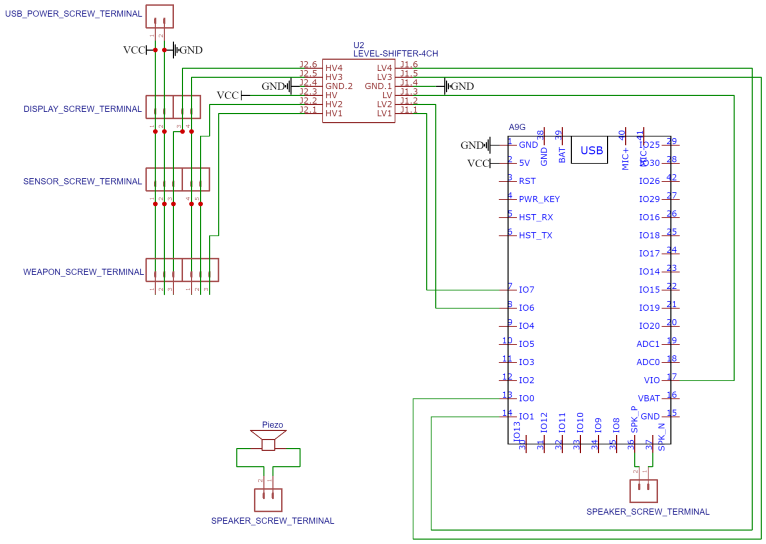
Pro jednodušší možnost opravy a programování mikrokontrolerů v modulech jsou tyto mikrokontrolery a převodníky logických úrovní osazeny v DuPont dutinkových lištách. Konektory typu DuPont jsou široce rozšířeny v oblasti mikrokontrolerů.

Hardware je osazený na pájivých polích, které nahrazují leptané desky tištěných spojů. Tato pole umožňují snadné osazení součástek pomocí propo-

jování jednotlivých bodů pole. V případě nutné změny lze snadno propojení odpájet a změnit tím rozvržení plošného spoje. V případě, že by se jednalo o produkční výrobek jsou ale výhody profesionálního tištěného spoje, jako například čistota spojů, nižší kapacita tras, přesné osazení součástek a další, nesporné.

9.1.1 Návrh hlavního modulu A9G

Nespornou výhodou modulu A9G je integrace velké části komponent, jako je například čtečka SD karet, GPS a GSM, přímo na vývojové desce pudding board. Na desce modulu je poté osazen pouze převodník logických hodnot, a konektory pro připojení napájení, datových sběrnic a reproduktoru.



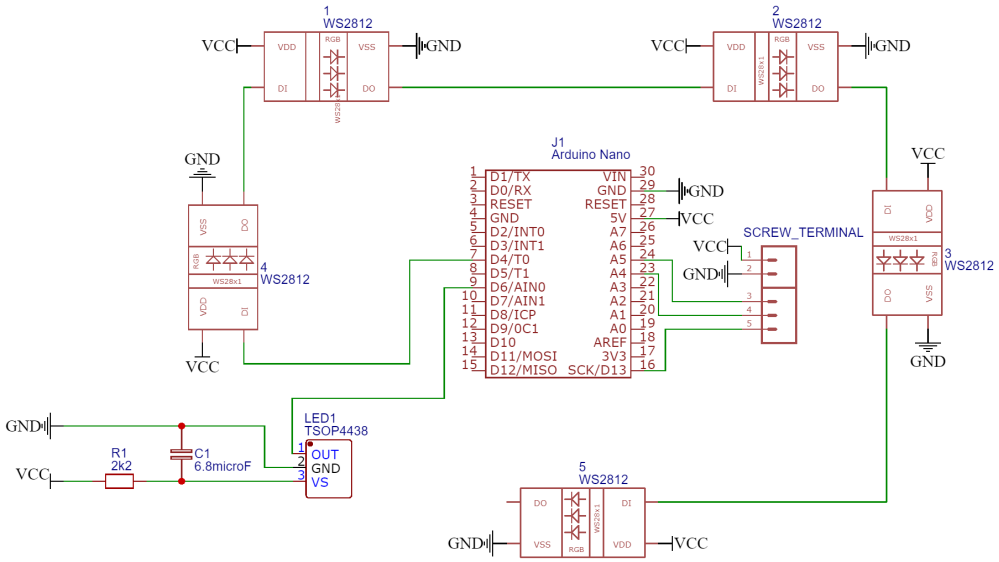
Obrázek 9.1: Schema zapojení hlavního modulu

9.1.2 Návrh modulu pro detekci zásahu

Hlavním úkolem modulu pro detekci zásahu bude přijímat a dekódovat signály pomocí senzoru infračerveného světla. Jak již bylo zmíněno v teoretické části, bude pro tuto detekci využit senzor TSOP4438. Pro odrušení napěťových špiček je k napájecím vodičům připojený rezistor o hodnotě 2200Ω a kondenzátor hodnoty $6.8\mu F$.

Pro vizuální zobrazení, zda je hráč na živu a zobrazení barvy týmu budou využity adresovatelné RGB LED. Výhodou adresovatelných diod je použití pouze jednoho GPIO pinu z mikrokontroleru a odpadá nutnost využití odporů před LED.

Komunikace s hlavním modulem probíhá po datové sběrnici pomocí protokolu I2C. Modul je napájen napětím 5V z datové sběrnice mezi hlavním modulem A9G a tímto modulem. (viz 9.1.4 Datová sběrnice)

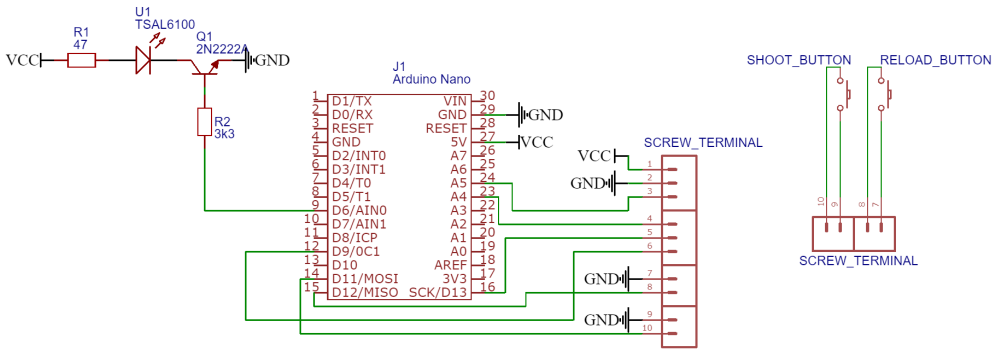


Obrázek 9.2: Schema zapojení modulu pro detekci zásahu

9.1.3 Návrh modulu pro střelbu

Primární funkcí modulu pro střelbu je zakódovat ID hráče a pomocí infračerveného paprsku ho odeslat na vestu protihráče. Pro vyslání paprsku je využita infračervená dioda TSAL6100, která je uložena v optice (viz 9.1.5 Optika). Dioda je řízena pomocí bipolárního NPN tranzistoru 2N2222A, který poskytuje dostatečně rychlé časy náběhu a doběhu pro modulaci signálu pomocí 38kHz frekvence. Pro omezení průchodu proudu je před diodou předřazen rezistor s hodnotou 47Ω. Tato hodnota je mírně vyšší, než je doporučeno výrobcem. Tím je mírně omezen dosah zbraně, avšak je zvýšena životnost této diody.

K modulu jsou připojena dvě tlačítka, která jsou ve výchozím stavu interním pull up rezistorem mikrokontroleru Arduino Nano nastavena na logickou hodnotu HIGH. Tato tlačítka slouží pro střelbu a přebíjení zbraně. Při stisku tlačítka dojde k propojení vstupního pinu kontroleru s GND pinem a tím pádem se logická hodnota změní na LOW. Kontroler detekuje tuto změnu a provede příslušnou akci.



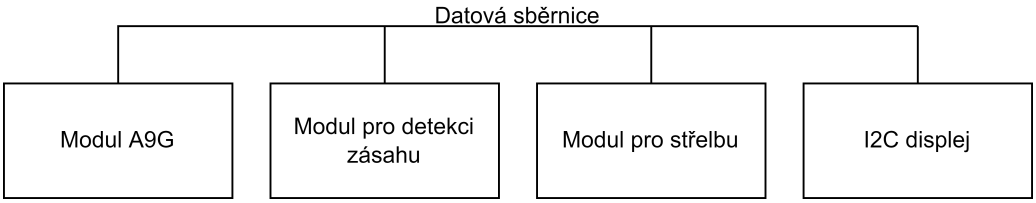
Obrázek 9.3: Schema zapojení modulu pro střelbu

9.1.4 Datová sběrnice

Datová sběrnice slouží pro komunikaci mezi hlavním modulem a ostatními moduly, které vykonávají jejich dané funkce. Datová sběrnice se skládá z šesti hlavních vodičů, viz Tabulka 9.1 níže. Jednotlivé moduly se připojují ke sběrnici paralelně, podle schématu níže (viz Obrázek 9.4). Pro realizaci datové sběrnice je využitý CAT6 UTP kabel. Kabel této specifikace obsahuje osm vodičů, které jsou spleteny do čtyřech kroucených párů. Uvnitř vodiče je plastové jádro. Vliv rušení na sběrnici by vzhledem k typu použitého kabelu měl být minimální. Napájení sběrnice, tedy 5V a GND, je vedeno čtyřmi vodiči (viz Tabulka 9.1).

Barva vodiče	Využití vodiče
Oranžová	Napájení 5V
Oranžovo-bílá	Napájení 5V
Zelená	Napájení GND
Zeleno-bílá	Napájení GND
Hnědá	I2C protokol SDA
Hnědo-bílá	I2C protokol SCL
Modrá	Přerušení - žádost o zprávu
Modro-bílá	Přerušení - střelba

Tabulka 9.1: Barva a využití vodičů v datové sběrnici



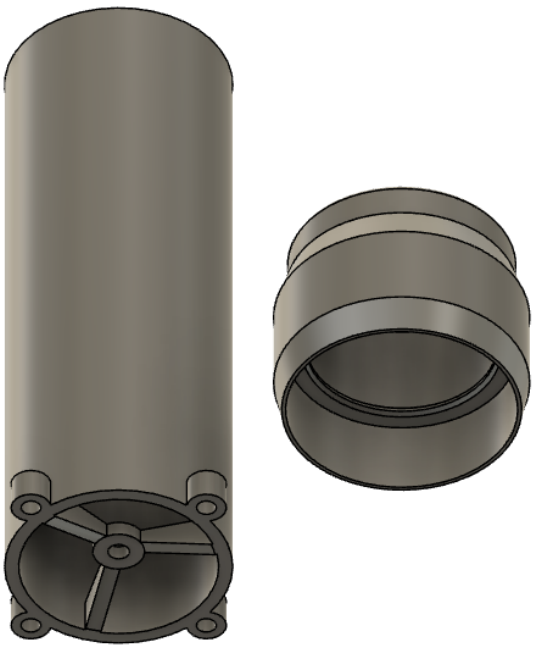
Obrázek 9.4: Jednoduché schéma zapojení datové sběrnice

Vzhledem k tomu, že modul A9G pracuje s napěťovou úrovní 3V a ostatní moduly včetně LED displeje vyžadují napěťovou úroveň 5V, je nutné mezi hlavní modul a datovou sběrnici vřadit převodník logických úrovní. Tento převodník využívající unipolární tranzistory BSS138 umí spolehlivě převést logické úrovně až pro čtyři vodiče. Pomocí tohoto převodníku je převedena logická úroveň pro všechny čtyři datové vodiče - sběrnici I2C a sběrnici přerušení.

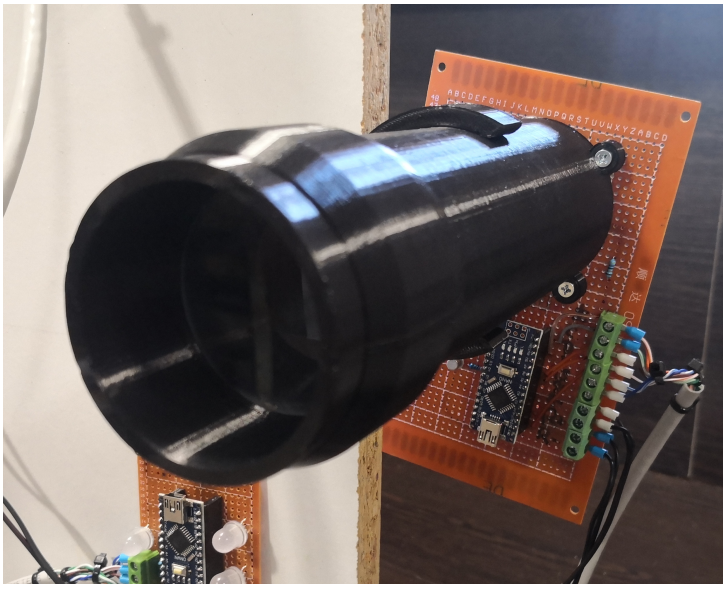
9.1.5 Optika

Základem optiky je tubus o průměru pět centimetrů a délce přibližně šestnáct centimetrů. Uprostřed tubusu je umístěn malý držák pro infračervenou diodu TSAL6100, která je do daného držáku přilepena. Na konci tubusu, ve vzdálenosti 15 centimetrů od LED diody je umístěný malý přesah, do kterého je umístěna spojná čočka o průměru pět centimetrů, jejíž ohnisková vzdálenost je právě patnáct centimetrů. Umístění LED diody přímo do ohniska dané čočky zaručuje vytvoření kolimovaného paprsku. Čočka je poté upevněna převlečným dílem, který je přilepený k tubusu. Na konci optiky u LED diody jsou umístěny čtyři díry sloužící pro připevnění optiky šrouby k desce plošných spojů.

Model optiky je vytvořený v programu Autodesk Fusion 360 (viz Obrázek 9.5). Pro možnost 3D tisku je model převedený na seznam příkazů pro 3D tiskárnu (dále také "G-Code") pomocí softwaru Ultimaker Cura. Optika je vytištěna pomocí 3D tiskárny Creality CR-10 z materiálu PLA (kyselina polymléčná, často využívaný polymer pro 3D tisk) černé barvy.



Obrázek 9.5: Model optiky s převlečným dílem



Obrázek 9.6: Kompletní optika s čočkou osazená na modulu pro střelbu

9.2 Návrh a realizace software

Software modulů je psaný v jazyce C v případě modulu A9G a speciální verzí jazyka C++ upravenou pro mikroprocesory Arduino v případě ostatních modulů. Jednotlivé moduly spolu komunikují pomocí přerušení a datové sběrnice I2C.

9.2.1 Realizace hlavního modulu

Program hlavního modulu je podobně jako program pro Arduino rozdělený do dvou částí. V první části běhu dojde k inicializaci veškerých komponent, jako je například nastavení GPIO přerušení, spuštění GPS, připojení k GSM síti a další. V této fázi také dojde k zobrazení načítací obrazovky, připojení na Socket server a pokus o přihlášení pomocí LOGIN zprávy (viz 10.5.2 Stavové zprávy přenášené pomocí Socket serveru). Po přijetí zprávy LOGINOK je inicializační část ukončena a je spuštěna hlavní smyčka, která reprezentuje herní část.

Při inicializaci dochází také k vyhledání modulů pro detekci zásahu připojených k datové sběrnici. Funkční modul vyšle na adresy v rozsahu od 10 do 15 zprávy HELLO a očekává odpověď. Pokud nedojde k chybě zápisu nebo čtení a v příchozí zprávě se nachází příkaz HELLO_RESPONSE, je adresa modulu uložena do pole. Při odesílání zpráv do modulů se poté zpráva posílá pouze na adresy, které se nachází v tomto poli, čímž dochází ke zrychlení komunikace na sběrnici. U modulu pro střelbu je očekávána pevná I2C adresa hodnoty 18 a proto mu není zasílána zpráva HELLO. Hodnoty rozsahu modulů pro detekci zásahu a adresu modulu pro střelbu je možné měnit pomocí konfiguračního souboru *config.h*.

Začátkem herní části vesta přechází do stavu, kdy je připravena ke hře a očekává příkazy pro inicializaci a start hry. Příchod těchto zpráv potvrdí příslušnou potvrzovací zprávou. Informace ze zprávy pro inicializaci hry, jako například ID hráče a číslo týmu, načte do proměnných a předá zároveň ostatním modulům, které je dále zpracují. Po potvrzení inicializační zprávy obdrží zprávu po spuštění hry, kdy součástí zprávy je takzvaný Epoch čas (počet sekund, které uběhly od 1. ledna 1970) startu a konce hry. Modul odpočítává start hry porovnáním s aktuálním časem, který se automaticky aktualizuje pomocí mobilní sítě. Po startu hry odešle zprávu ACTIVATE (viz 9.2.4 Příkazy předávané mezi moduly) do ostatních modulů, které se tímto aktivují.

Vykonávání herní smyčky je možné narušit pomocí prostřednictvím sběrnice přerušení. Pokud dojde k přerušení střelbou, modul odečte jeden náboj

z celkového počtu nábojů, které má hráč v zásobníku. Pokud hráči dojdou náboje, je do zbraně odeslán příkaz OUTOFAMMO, čímž dojde k blokování střelby zbraně do doby, dokud hráč zbraň nepřebije. Při druhém typu přerušení se hlavní modul postupně dotáže všech funkčních modulů, zda pro něj nemají nějakou zprávu. Při tomto dotazování je možné přijmout pouze jednu zprávu od každého typu. Tím je zaručeno, že například pokud zásah registruje více modulů, hráč je zasažen pouze jednou. Vzhledem k rychlosti komunikace po datové sběrnici se nepředpokládá, že by mohlo dojít k dalšímu zásahu během komunikace. Pokud by ovšem k takovéto události došlo, je druhý příchozí zásah ignorován.

Vzhledem k tomu, že procesor A9G podporuje možnost více úloh, které jsou spouštěny nezávisle na sobě, je hlavní kód rozdělený na dvě úlohy. V první úloze s vyšší prioritou dochází ke zpracování událostí generovaných systémem. Mezi tyto události se řadí příjem zprávy ze socket serveru, příjem informací z GPS anebo GSM čipu a další. V druhé úloze dochází k samotnému vykonání herního kódu. Kvůli běhu více úloh je avšak nutné zajistit zabezpečení kritických sekcí, jako je například příjem a odesílání zpráv pomocí socketu anebo sběrnice I2C. Kritické sekce jsou zabezpečeny jednoduchou implementací semaforu, kdy do kritické sekce může vstoupit pouze jedna úloha. Pokud se v kritické sekci již jiná úloha nachází, tak úloha, která se snaží ke kritické sekci přistoupit vyčkává pomocí API funkce *OS_Sleep*. Ta zajistí předání procesorového času jiné úloze, která může dokončit svoji činnost a opustit kritickou sekci.

Během herní části dochází k periodickému obnovování displeje. Uživatel na displeji vidí nejaktuálnější informace o hře, jako například počet životů a nábojů a zbývající čas hry. Pro snížení objemu přenosu dat je aktualizována pouze ta část, ve které došlo ke změně textu.

9.2.2 Realizace modulu pro detekci zásahu

Hlavní funkcí modulu pro detekci zásahu je číst data z infračerveného senzoru. Tento senzor je připojený k datovému pinu číslo D6. Pokud je modul aktivován, to znamená, že obdržel z mikrokontroleru A9G zprávu ACTIVATE (viz 9.2.4 Příkazy předávané mezi moduly), pak ve smyčce čte hodnoty z infračerveného senzoru. Samotné čtení se provádí pomocí knihovny IRremote, která je veřejně dostupná ke stažení [9].

Po přečtení hodnoty, která je délky 4 byty, se zpráva rozdělí na kontrolní byty a datový byte. Kontrolními byty se rozumí první tři byty z přijatých dat. Tyto tři byty se musí rovnat hodnotě specifikované v konstantě *BYTE_CHECK_MASK*, která je definována v souboru *common.h* a která

musí být součástí modulu pro detekci zásahu a modulu pro střelbu. Pokud se první kontrolní byty nerovnají dané konstantě, je zpráva považována za poškozenou a je zahozena. V případě ověření zprávy je z datového bytu přečteno ID hráče, který zasáhl danou vestu. Toto ID je poté předáno hlavnímu modulu A9G pomocí datové sběrnice.

Vedlejší funkcí modulu je zobrazení týmu daného hráče pomocí adresovatelných LED diod. Datový vodič diod je připojený ke kontroleru pomocí datového pinu číslo D4. Pro nastavení barev je použita knihovna Adafruit_NeoPixel, která je též volně dostupná ke stažení [3]. Barva LED diod je nastavena podle čísla týmu, které modul obdrží součástí zprávy INIT (viz 9.2.4 Příkazy předávané mezi moduly).

Přijímání a zpracování předání dat obsluhuje knihovna Wire, která je součástí vývojové sady Arduino. Tato knihovna zprostředkovává dvě funkce, které lze použít pro přerušení hlavní smyčky kódu. Funkce *onRequest* zprostředkovává obsluhu při žádosti o data z tohoto modulu. Během této obsluhy jsou odeslána data, která se nacházejí ve proměnné *valueToSend* a daná proměnná je po odeslání nastavena na příkaz NOINFO. Druhá funkce *onReceive* zprostředkovává obsluhu přerušení při příjmu dat z hlavního modulu. Během této obsluhy je detekován příkaz, který je tomuto modulu zaslán a je provedena příslušná akce.

9.2.3 Realizace modulu pro střelbu

Vzhledem k jediné hlavní funkci je softwarová realizace modulu pro střelbu poměrně jednoduchá. Pro obsluhu I2C sběrnice je využita knihovna Wire, stejně jako v modulu pro detekci zásahu (viz 9.2.2 Realizace modulu pro detekci zásahu).

Modul periodicky kontroluje logickou úroveň na vstupních pinech, ke kterým jsou připojena tlačítka pro přebití zbraně a pro střelbu. Při stisknutí tlačítka pro střelbu je zasláno přerušení do hlavního modulu a zároveň je pomocí infračervené LED diody TSAL6100 vyslána datová zpráva. O zakódování zprávy do paprsku se stará knihovna IRremote, která byla využita i v modulu pro příjem signálu. První tři byty jsou kontrolní byty, které jsou uložené v souboru *common.h*. Do posledního bytu zprávy je vloženo ID hráče, které modul pro střelbu obdržel ve zprávě INIT (viz 9.2.4 Příkazy předávané mezi moduly).

Pokud hráči dojdou náboje a tím pádem modul obdrží příkaz OUTOFAMMO, je tlačítko pro střelbu deaktivováno do doby, než modul obdrží zprávu RELOAD_COMPLETE, případně ACTIVATE. Při stisknutí tlačítka pro střelbu je na dobu 100ms změněna logická hodnota na datovém vodiči pře-

rušení pro střelbu, které zajistí obsluhu příslušného přerušení v hlavním modulu. Tlačítko pro přebíjení je aktivní pouze pokud modul obdrží zprávu ACTIVATE do doby než obdrží zprávu DEACTIVATE. Při stisknutí tlačítka pro přebíjení je do hlavního modulu odeslána zpráva obsahující příkaz RELOAD.

9.2.4 Příkazy předávané mezi moduly

Příkazem, který se předává mezi moduly se rozumí první byte zprávy přenesený mezi funkčním modulem a hlavním modulem A9G. Tento byte přesně identifikuje typ zprávy, která je zasílána a přijímající modul může podle toho přesně reagovat. Součástí zprávy mohou být další data, která se nachází na dalších pozicích ve zprávě. Seznam předávaných zpráv je dostupný v tabulce níže. Hodnoty příkazů jsou definovány v souboru *commands.h* a musí se shodovat napříč všemi moduly.

Příkazy, které se odesílají z hlavního modulu do ostatních funkčních modulů jsou v rozsahu 0x01 až 0x79. Pro příkazy z funkčních modulů do hlavního modulu je vyhrazený rozsah 0x81 až 0x9F. Speciální příkaz 0xA0 je zaslán, pokud hlavní modul zažádá o informace, ale funkční modul nemá žádnou informaci, kterou by zaslal.

Příkaz	HEX	Akce
HELLO	0x01	Začátek komunikace s modulem
DEACTIVATE	0x02	Deaktivace modulu
ACTIVATE	0x03	Aktivace modulu
OUTOFAMMO	0x04	Zablokování spouště ve zbrani
RELOAD_COMPLETE	0x05	Odblokování spouště ve zbrani
INIT	0x06	Nastavení ID hráče a týmu
HELLO_RESPONSE	0x81	Odpověď na začátek komunikace
SHOT	0x82	Detekce zásahu
RELOAD	0x83	Stisknutí tlačítka přebíjení
NOINFO	0xA0	Modul nemá žádná data k předání

Tabulka 9.2: Zprávy zasílané mezi moduly

Součástí zprávy 0x82 SHOT je ID hráče, který tuto vestu zasáhl. ID je uloženo v druhém bytu zprávy. Ve zprávě 0x06 INIT se nachází ID hráče, který je svázaný s touto vestou a číslo týmu. To se nachází ve druhém bytu zprávy, ID hráče je uloženo ve třetím bytu zprávy.

Maximální délka zprávy, kterou je možné přenést mezi moduly je deset

bytů. Odesílání zpráv této délky příliš nevytěžuje sběrnici, avšak tato délka by měla být dostatečná pro případná rozšíření do budoucna.

10 Realizace serveru

Serverová aplikace je připravena v jazyce Java verze 11. Pro poskytování REST API je využito frameworku Spring Boot z důvodu, že poskytuje funkční a stabilní HTTP server s vícevláknovým zpracováním žádostí a obsáhlou sadou dalších funkcí, které usnadňují vývoj REST API a přístup k databázi. Pro realizace socket serveru je využito standardní implementace java.net.Socket, která je součástí Javy. Té je využito z důvodu absence knihoven, které by pracovaly s čistou implementací Socketů a nevyžadovaly využití WebSocket protokolu. Tento protokol není na mikrokontroleru A9G podporován. Aplikaci je možné konfigurovat pomocí soubor application.properties, anebo parametrů v příkazové řádce.

10.1 Struktura aplikace

Aplikace je rozdělena do několika hlavních částí a to části pro zprostředkování REST API, pro poskytování Socket serveru, pro řízení logiky hry a části pro přístup k databázi a pomocných funkcí.

REST API je dále rozděleno na Kontroler, Službu a Model. Kontroler slouží pro přijetí žádosti na REST API a zpracování vstupních dat do modelových tříd. Tato data pak předá ke zpracování službě, která vykoná veškerý kód, a vrátí výslednou hodnotu. Kontroler poté odešle výsledné hodnoty v odpovědi, případně podle daných hodnot změní například HTTP stavový kód, nebo jiné parametry odpovědi. Kontroler implementuje rozhraní, které je vygenerováno pomocí generátoru z OpenAPI (viz 10.1.1 API-first architektura). Veškeré modelové třídy jsou též vygenerovány pomocí tohoto generátoru. O zabezpečení žádostí se stará součást frameworku Spring Security.

Socket server běží v samostatném vlákně a řídí přidělování dalších vláken nově připojených klientů. Využitím samostatného vlákna pro každého klienta je docíleno plynulé obsluhy několika klientů zároveň. Potvrzení přijetí zprávy kontroluje třída Acknowledger.

Řízení herní logiky hry zabezpečuje instance třídy GamesManager. Tato třída zajišťuje ověření vstupních dat, kontroluje správně předání informací vestám a zajišťuje zasílání veškerých informací. Poskytuje veškeré informace o hrách pro další zpracování pomocí REST API rozhraní. Odpočet konce hry zajišťuje využití instance třídy java.util.Timer, která spustí danou akci za určitý počet sekund. Odpočet je součástí instance Game, která primárně uchovává informace o jednotlivé hře.

Popis, jak získávat data z databáze jsou uchována v rozhraních v balíčku *repository*. Funkce tohoto rozhraní, které rozšiřuje rozhraní JpaRepository poskytované frameworkem Spring, deklarují objekty, které vstupují jako parametry, a objekty, které jsou očekávány na výstupu funkce. Vlastní SQL příkaz dané funkce je možné zadat anotací *@Query*. Jména sloupců jednotlivých objektů jsou popsány anotací *@Column* v modelových třídách. Tyto třídy jsou anotovány pomocí anotace *@Table*, která určuje z jaké tabulky je možné tyto informace získat.

Konstruktory, gettery, settery a další standardní funkce jsou automaticky generovány pomocí knihovny Lombok. To zjednodušuje zápis kódu, avšak je možné, že při generování metod equals a hashCode jsou součástí proměnné, jenž nejsou žádané. Odebrání proměnné z funkce Equals, případně toString je tedy zajištěno anotací *@EqualsAndHashCode.Exclude*, respektive *@ToString.Exclude*.

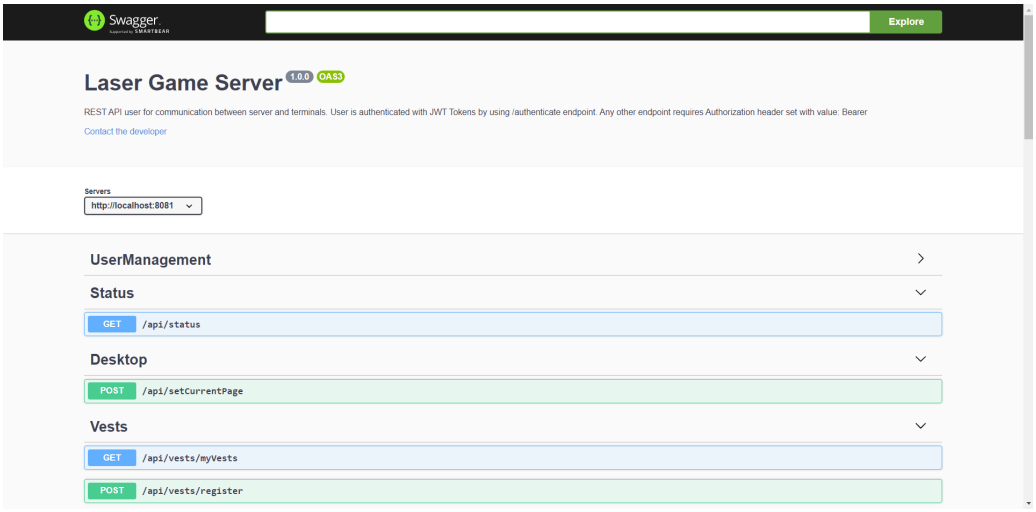
10.1.1 API-first architektura

Využití API-first architektury spočívá v generování kódu podle zadané API definice. Hlavní výhodou tohoto přístupu je to, že vygenerovaný kód přesně odpovídá struktuře v API a eliminuje možnost chyb při psaní kódu. Generování kódu zároveň urychluje vývoj, jelikož není třeba samostatně navrhovat modelové třídy, které budou sloužit pro přenos dat.

Pro vygenerování Java kódu se využívá Apache Maven zásuvného modulu (také plugin) openapi-generator, který vygeneruje zadané modelové třídy při spuštění kroku (viz 8.2 Apache Maven) generate-sources [19]. Tento plugin automaticky umožňuje načíst soubor typu YAML, který obsahuje API definice zapsané ve formátu OpenAPI verze 2 či verze 3. Po načtení API definic vygeneruje kód podle šablony, která se nachází v souboru *api.mustache*. Tato šablona je upravena pro generování kódu pro využití frameworkem Spring. Použitím této šablony odpadá nutnost použití většiny anotací, které framework Spring ve výchozím stavu vyžaduje, jelikož jsou součástí již vygenerovaných rozhraní, které třídy jednotlivých kontrolerů implementují.

Jednotlivé endpointy (koncové body, na které lze zasílat žádosti pomocí HTTP protokolu) a datové třídy jsou definovány v souboru *service-api.yaml* ve formátu OpenAPIv3. Z tohoto souboru lze snadno vygenerovat HTML webovou stránku pomocí nástrojů, které jsou součástí IDE (tzv. vývojové prostředí, anglicky Integrated Development Environment) IntelliJ IDEA, které bylo využito k vývoji. Alternativou je použití webového portálu editor.swagger.io. Z vygenerované webové stránky lze snadno zjistit data, která je nutné zaslat a odpovědi, které lze očekávat z daného endpointu. Webová

stránka zároveň umožňuje zaslání dat na tyto endpointy pomocí vestavěného testovacího nástroje.



Obrázek 10.1: Ukázka vygenerované stránky obsahující REST API použité na Serveru

10.2 Struktura databáze

Microsoft SQL databáze obsahuje celkem pět tabulek, které obsahují informace o uživatelích, vestách a výsledcích. Databázi je možné vytvořit pomocí skriptu *createDatabase.sql*, který je uložen v kořenovém adresáři zdrojového kódu Serveru. Pro jednoznačnou identifikaci v každé tabulce slouží sloupec ID typu bigint, který tvoří primární klíč dané tabulky.

10.2.1 Tabulka Users

Tabulka Users obsahuje data o jménu, heslu a poslední stránce uživatele. Jméno, heslo a poslední stránka jsou datového typu nvarchar maximální délky 255 znaků. V databázi by za každých okolností měl být uložený pouze otisk (také Hash) daného hesla, což zajišťuje Server. Pole poslední stránky může obsahovat hodnotu NULL (to znamená, že hodnota není vyplněna).

10.2.2 Tabulka Vests

Tabulka Vests je jednoduchá tabulka obsahující jméno a kód vesty. Tyto pole jsou typu nvarchar s délkou maximálně 50 znaků. V tabulce se zároveň nachází sloupec Owner, jenž specifikuje ID uživatele z tabulky Users, ke kterému je vesta přiřazena. Sloupci je přiřazen takzvaný cizí klíč, jenž zaručuje

integritu dat. To znamená, že v tabulce Vests se nemůže vyskytovat záznam s hodnotou sloupce Owner takovou, která neexistuje v tabulce Users.

10.2.3 Tabulka Games

V tabulce Games jsou uloženy detailní informace o nastavení dané hry. Sloupec Status typu nvarchar reprezentuje stav hry, ve kterém se aktuálně nachází. Cizí klíč ve sloupci Start_User přiřazuje hru uživateli z tabulky Users, který hru spustil. Tyto hodnoty musí být vždy vyplněny. Další hodnoty, jako je například Start_Date, Game_Duration, Player_Lives, Weapon_Damage, Player_Ammo, Respawn_Period a Start_Timeout vyplněny být nemusí (povolují hodnoty typu NULL). Tyto hodnoty slouží pro nastavení jednotlivých parametrů hry a ve většině případů jsou typu int, případně datetime u sloupce Start_Date.

10.2.4 Tabulka Player_Results

V této tabulce jsou uloženy informace o jméně, týmu a výsledcích daného hráče. Výsledky jsou provázány s konkrétní hrou pomocí cizího klíče ve sloupci Game a s vestou, kterou hráč v dané hře používal, pomocí cizího klíče ve sloupci Vest. Sloupce Kills a Deaths pro ukládání informací o zásazích a smrtích uživatele jsou typu int. Maximální délka jména hráče je 255 znaků. Veškeré hodnoty v této tabulce musí být vyplněny.

10.2.5 Tabulka Player_Position

Tato jednoduchá tabulka slouží pouze k uložení zeměpisné šířky a délky, na které se daný hráč nacházel. S hráčem je tabulka svázána pomocí cizího klíče ve sloupci Player. Součástí tabulky je i datum a čas, kdy došlo k uložení dané pozice.

10.3 Správa uživatelů a jejich přihlašování

Registrace uživatele

Osobní údaje uživatelů jsou uloženy v databázové tabulce. Při registraci dojde k vytvoření nového záznamu v této tabulce, ve kterém je uloženo uživatelské jméno a otisk hesla. Tento otisk využívá funkce BCrypt a je vytvořený pomocí třídy *BCryptPasswordEncoder*, která je součástí frameworku

Spring Boot. Uživatele je možné odstranit pouze odstraněním daného záznamu v databázové tabulce.

Přihlášení uživatele

Při přihlášení dojde k porovnání vloženého hesla a otisku pomocí funkce authenticate v třídě *AuthenticationManager*. Tato třída je též poskytována frameworkem Spring Boot a automaticky zajistí načtení uživatelských dat z databáze a provedení porovnání přihlašovacích údajů. V případě, že uživatel zadá nesprávné jméno či heslo, dojde k vyhození výjimky *AuthenticationException*, která je dále zpracována a v odpovědi dojde k vrácení HTTP chybového kódu 406 NOT ACCEPTABLE. V případě, že je uživatel autentikován, je vygenerován identifikátor sezení (Session ID), který jednoznačně identifikuje toto přihlášení. Tento identifikátor je uložen ve správci sezení (třída *SessionManager*) a zároveň je součástí vygenerovaného JWT tokenu, který uživatel používá při dalších žádostech na Server.

V případě, že ve správci sezení již je uložený daný uživatel, je žádost o přihlášení zamítnuta a je nutné ji opakovat s parametrem *logover* nastaveným na hodnotu true. Touto žádostí dojde k odstranění předchozího sezení a je nahrazeno sezením novým. Pokud se uživatel z předchozího sezení pokusí o provedení jakékoliv jiné žádosti, je tato žádost zamítnuta. Tento mechanismus slouží k tomu, aby se zabránilo kolizím při zasílání různých žádostí.

Ověřování žádostí přihlášených uživatelů

Před zpracováním jakékoliv žádosti (vyjma žádosti na endpoint /authenticate, který slouží pro přihlášení) je ověřována autorizace dané žádosti. Pro ověření slouží třídy, které jsou v balíčku *cz.zcu.tlinhart.lasergame.server.authentication*. Implementace je inspirována příkladem z webových stránek JavaInUse [14]. Proces ověřování žádosti je předřazený jakémukoliv jinému výkonnému kódu. Během tohoto procesu dojde k přečtení a ověření přijatého JWT tokenu pomocí šifrovací funkce. Z toho je také získána doba platnosti daného token a identifikátor sezení, který je ověřen v manažeru sezení. Pokud není splněna některá z podmínek, je token označen za neplatný a žádost je zamítnuta. V odpovědi se poté nachází HTTP stavový kód 401 UNAUTHORIZED.

10.4 Implementace REST API

REST API je implementováno s využitím frameworku Spring Boot. Tento framework byl zvolen jelikož poskytuje plně funkční HTTP server, zajišťuje obsluhu zpracování HTTP žádostí, podporuje snadný přístup do databáze s využitím frameworku Hibernate a snadné předávání instancí třídy mezi jednotlivými endpointy pomocí anotací *@Autowired*. Jednotlivé kontrolery jsou definovány pomocí anotace *@RestController*. Tyto kontrolery zajišťují obsluhu žádostí jednotlivých endpointů, které jsou definovány pomocí OpenAPI specifikace a následně vygenerovány použitím OpenAPI generátoru (viz 10.1.1 API-first architektura). Zprávy jsou zasílány ve formátu JSON použitím třídy ObjectMapper. Pro zaslání data a času je využito vlastní implementace serializátoru, který zasílá hodnotu jako počet sekund od 1. 1. 1970, takzvaného epoch času.

10.4.1 AuthenticationController

AuthenticationController slouží pro přihlašování a odhlašování uživatelů. Data jsou zpracována a předána službě AuthenticationService ke zpracování. Chyby, které se mohou vyskytnout při přihlášení jsou odchytávány pomocí funkce s anotací *@ExceptionHandler*. V případě, že se jedná o výjimku typu NotLogoverException, je vrácena odpověď se stavovým kódem 406 NOT ACCEPTABLE. Ostatní výjimky jsou vypsány do konzole a je vrácena odpověď se stavovým kódem 409 CONFLICT, jelikož se pravděpodobně jedná o nesprávně zadané přihlašovací údaje.

10.4.2 GamesController

Kontroler slouží pro řízení a zobrazení stavu dané hry. Endpoint createGame slouží pro vytvoření, případně zkopírování nové hry. ID hry, která má být zkopírována, je předáno v těle žádosti. Hry je možné i mazat odesláním žádosti na endpoint deleteGame. Po vytvoření nové hry je třeba tuto hru spustit zavoláním endpointu startGame, kdy v těle žádosti se musí nacházet data o parametrech hry a údaje jednotlivých hráčů. Běžící hru je možné zastavit před uplynutím časového limitu voláním endpointu stopGame. Seznam všech her, které uživatel odehrál je možné získat pomocí endpointu myGames. Pro získání konkrétních informací o hře je připravený endpoint getGameInfo. V případě, že součástí žádosti není ID hry, o které chce klient získat informace, pokusí se vrátit informace o právě běžící hře. Pokud nenalezl žádnou hru vrací odpověď s chybovým statusem 409 CONFLICT.

10.4.3 VestsController

Registrovat nové vesty je možné pomocí endpointu register. V těle žádosti se musí nacházet informace o nově registrované vestě a to kód vesty a její pojmenování. Pokud registrace nebyla úspěšná, je vrácen chybový kód 409 CONFLICT. Vesty, které má uživatel zaregistrované, je možné odstranit pomocí endpointu delete. Seznam vest, které má uživatel zaregistrované, poskytuje koncový bod na adrese myVests. V odpovědi se nachází jméno a kód vesty a informace, zda je vesta právě připojená k serveru.

10.4.4 DesktopController a StatusController

Jediným úkolem DesktopControlleru je zprostředkovat endpoint pro uložení poslední stránky, kterou uživatel zobrazil. Data jsou předána službě, která je uloží do databáze k danému hráči. Při přihlášení uživatele je pak poslední stránka předána součástí přihlašovací odpovědi a Terminál tuto stránku automaticky zobrazí. Uživatel je určen pomocí JWT tokenu, který je předán v Authorization parametru žádosti.

StatusController zprostředkovává informace o stavu serveru, jako je využití procesoru, paměti a doba běhu aplikace. Tento endpoint je v rámci této práce využitý především pro testování a ladění aplikace.

10.5 Implementace Socket serveru

Objekty, jenž jsou zasílány pomocí zpráv musí implementovat rozhraní ISendable. Toto rozhraní zajišťuje, že bude implementována funkce *getSocketSendData*. Tato funkce vrací hodnotu typu String, která obsahuje přesný obsah dané zprávy včetně znaků pro oddělení hodnot (podtržítek). ID zprávy je přidáno těsně před odesláním na začátek zprávy a odděleno podtržítkem.

10.5.1 Ověřování přijetí zprávy

Zprávy, které se odesílají ze Serveru na jednotlivá zařízení jsou identifikovány jednoznačným číslem typu Integer, který se nachází jako první parametr dané zprávy. Toto číslo je zvoleno z rostoucí sekvence s výchozí hodnotou 1. Pro každého připojeného klienta je vytvořena nová sekvence.

Očekává se, že klient, který zprávu přijme, potvrdí přijetí této zprávy odesláním zprávy ACK, která obsahuje ID právě přijaté zprávy. V případě, že nedojde k tomuto potvrzení, je po určitém intervalu odeslání zprávy opako-

váno. Tento interval je definovaný hodnotou *app.settings.messageAcknowledgeTimeoutSeconds* v souboru *application.properties*. Dokud není zpráva potvrzena, je tato zpráva uložena v proměnné *sentBuffer*. Po potvrzení je hodnota nastavena na NULL, případně je nahrazena další zprávou z fronty (viz níže).

Do doby, než je poslední odeslaná zpráva potvrzena, tak není možné odeslat žádnou další zpráva. Všechny zprávy, které nebylo možné odeslat, jsou uloženy do fronty a po potvrzení přijetí zprávy je odeslána první zpráva z této fronty. Tím je zaručeno, že nedojde ke změně pořadí odeslaných zpráv, avšak žádná zpráva není zahozena a na klienta jsou postupně doručeny všechny odeslané zprávy.

Kontrolu přijetí zajišťuje nové vlákno třídy Acknowledger. To periodicky kontroluje hodnotu proměnné *sentBuffer* pro každého připojeného klienta. Pokud tato hodnota není NULL a rozdíl aktuálního času a času odeslání konkrétní zprávy je větší než hodnota intervalu opakování zprávy, dojde k opakování odeslání zprávy. Vlákno po vykonání vyčkává počet sekund daný konfigurační hodnotou *app.settings.acknowledgerCheckPeriod*.

10.5.2 Stavové zprávy přenášené pomocí Socket serveru

Stavové zprávy jsou přenášeny jako textové řetězce přesně daného formátu. Jednotlivé zprávy a akce, které se při jejich přijetí vykonávají jsou rozepsány v tabulkách níže.

Zprávy přenášené ze serveru na zařízení

Zprávy, které jsou odesílány ze serveru na zařízení je nutné potvrzovat zprávou ve formátu *ACK_[ID_Zprávy]*. Přijetí zpráv, které se odesílají ze zařízení na server, se neověřuje. ID zprávy je předáváno jako první parametr před danou zprávou. Tato zpráva pak vypadá následovně [ID_Zprávy]_[Obsah_Zprávy], kdy obsah zprávy lze zjistit v tabulce.

Zpráva	Welcome to lasergame
Akce	Informační zpráva o připojení na lasergame server. Využitím této zprávy se ověřuje správnost funkce mechanismu potvrzování přijetí zpráv pomocí ACK zpráv.

Zpráva	PREP_[PlayerID]_[TeamID]_[Player1ID]_[Team1ID] _[Name1]_[Player2ID]_[Team2ID]_[Name2]_...
Akce	Slouží pro předání informací o hráčích, kteří budou hrát v připravované hře. Pole PlayerID a TeamID předávají informace pro aktuálního hráče. Další pole označují data pro ostatní hráče ve hře a mohou předávat informace až o osmi hráčích.
Zpráva	START_[AMMO]_[LIVES]_[SHOTDAMAGE] _[RESPAWNPERIOD]_[EPOCHSTART]_[EPOCHEND]
Akce	Slouží pro předání informací o právě spuštěné hře. Obsahuje parametry hry a čas, kdy hru spustit a ukončit. Při přijetí zprávy dojde k odpočtu startu a následnému spuštění hry.
Zpráva	STOP
Akce	Hra je ukončena, i pokud ještě nedošlo k vypršení časového limitu.
Zpráva	LOGINOK
Akce	Odpověď na zprávu LOGIN. Oznamuje, že vesta je v pořádku přihlášena.
Zpráva	LOGINERR
Akce	Odpověď na zprávu LOGIN. Oznamuje, že při přihlašování vesty došlo k chybě a je nutné pokus o přihlášení opakovat. Nepředává důvod selhání přihlášení.
Zpráva	PING
Akce	Není prováděna žádná akce. Slouží pro detekci uzavření Socketu na straně serveru.

Tabulka 10.1: Socket zprávy zasílané ze serveru na připojená zařízení

Zprávy přenášené ze serveru na zařízení

Na rozdíl od zpráv, které se přenášení do zařízení, není součástí zpráv, které se zasílají na server, ID zprávy. Formát zprávy, která je zaslána tak přesně odpovídá formátu z tabulky níže.

Zpráva	ACK_[MESSAGE_ID]
Akce	Potvrzení poslední přijaté zprávy.
Zpráva	LOGIN_[VEST_CODE]
Akce	Žádost o přihlášení vesty s daným kódem vesty.

Zpráva	PREPAREDONE
Akce	Oznamuje správné přijetí a zpracování dat ze zprávy PREP.
Zpráva	STARTDONE
Akce	Oznamuje správné přijetí a zpracování dat ze zprávy START.
Zpráva	POS_[LATITUDE]_[LONGITUDE]
Akce	Předává na server aktuální pozici hráče.
Zpráva	SHOT_[PLAYER_ID]
Akce	Předává serveru ID hráče, který aktuálního hráče zasáhl.

Tabulka 10.2: Formát Socket zpráv zasílaných ze zařízení na Server

11 Realizace terminálu

Vzhledem k aktuálnímu vývoji v oblasti aplikací je terminálová aplikace připravena pouze jako webová stránka. Rozložení stránky je přizpůsobeno pouze zobrazení na počítačích, případně tabletech s rozložením na šířku (odborně nazýváno "landscape"rozložení). Při vývoji je využito frameworků Bootstrap, jQuery a jQueryUI. Pro zrychlení načítání jednotlivých stránek a snížení přenášeného datového objemu je využito stahování komponent a dat pomocí AJAX (více v 11.1 Návrh a realizace frameworku). Jednotlivé skripty jsou načítány pomocí frameworku RequireJS, který poskytuje asynchronní načítání skriptů pouze v případě potřeby. Pro vygenerování map s přehledem o pozicích hráčů je použita knihovna Leaflet s využitím OpenStreetMap.

11.1 Návrh a realizace frameworku

Frameworkem se rozumí sada funkcí, která slouží pro základní načítání a zpracování dat na obrazovce. Hlavním úkolem tedy je načítat jednotlivé komponenty pomocí AJAX (Asynchronous Javascript and XML) volání. Součástí frameworku jsou pomocné funkce, které mohou jednotlivé komponenty volat. Framework zajišťuje správné načtení dat z dostupného REST API a zpracování chybových kódů na nejnižší úrovni. Při obdržení HTTP status kódu 401, který znamená neplatnost přihlašovacích údajů, je uživatel přesměrován na přihlašovací stránku. V případě nedostupnosti REST API je uživateli zobrazena chybová obrazovka.

Ke každému REST API volání a AJAX načítání privátních komponent je do hlavičky HTTP žádosti přidána hlavička Authorization, která obsahuje Bearer token, který aplikace získá v odpovědi z REST API serveru při přihlášení. Tento token jednoznačně identifikuje daného uživatele a bez tohoto tokenu žádost o data selže. Token je uložený v lokálním úložišti daného prohlížeče pod klíčem bearer.

Veřejné funkce frameworku jsou dostupné v globálním namespace pojmenovaném lasergame. K těmto funkcím lze tedy přistupovat pomocí volání lasergame.[název_funkce]().

Veškerý obsah jednotlivých komponent je zobrazen v hlavním elementu typu DIV. Komponentu, kterou má framework právě vykreslit získá z GET parametru *url*, případně z dat při přihlášení uživatele. I když se jedná o asynchronní načítání komponent, tak framework zajišťuje uložení těchto strá-

nek v historii prohlížeče a umí načítat komponenty i při přechodu historií zpět, či vpřed. Při načítání jednotlivých komponent je zobrazena načítací obrazovka. Skrytí této obrazovky zajišťují přímo jednotlivé komponenty, které frameworku oznámí kompletní načtení pomocí funkce *lasergame.componentLoadComplete()*

Je nutné, aby každá komponenta implementovala globální funkci *loadComponent()*. Tato funkce zajistí načtení Javascriptu v dané komponentě. Tento Javascript může například zavolat REST API pro získání dat, které se v komponentě vykreslí. na konci funkce je nutné zavolat metodu *lasergame.componentLoadComplete()*, jinak nedojde k ukončení načítání stránky a uživateli bude stále zobrazena pouze načítací obrazovka.

Před načtením nové komponenty je zavolána funkce *onBeforeComponentUnload()*, pokud je definována. Pomocí této funkce je možné spustit akce těsně před tím, než je místo komponenty načtena komponenta nová. Tohoto mechanismu se využívá například pro odstranění posluchačů událostí (například na stisknutí tlačítka), nebo na vynulování intervalů.

11.2 Komunikace se serverem

Pro komunikaci se serverem je využito asynchronního volání pomocí s využitím REST API, které poskytuje server (viz 10.4 Implementace REST API). Pro získání dat framework poskytuje funkci *readRESTData*. Tato funkce zajišťuje načtení dat ze serveru, který je specifikovaný v konfiguračním souboru.

V případě, že dojde k chybě načítání dat z důvodu žádného, případně nevalidního, tokenu v hlavičce Authorization, dojde k zobrazení přihlašovací stránky. Uživatel je po přihlášení přesměrován na předchozí stránku, avšak REST žádost není opakována. Například v případě, že uživatel klikne na tlačítko Vytvořit novou hru, je přesměrován na přihlášení a následně zpět na Ovládací panel. Pro vytvoření nové hry musí na tlačítko Vytvořit novou hru kliknout znovu.

Pokud dojde k chybě, na kterou klient neumí zareagovat, je zobrazena chybová obrazovky obsahující detaily chyby. Po zobrazení této je možné pouze stránku aktualizovat, případně přejít na hlavní ovládací panel. Tuto chybovou obrazovku není možné jinak skrýt.

11.3 Realizace komponent

Komponenty jsou rozděleny na dva typy - privátní a veřejné. Veřejné komponenty jsou dostupné bez jakékoliv autorizace. Těmito komponenty jsou úvodní stránka, kterou uživatel vidí při načtení domény a stránka pro přihlášení. Privátní komponenty by měly být dostupné pouze po přihlášení. Vzhledem k tomu, že terminál je psaný pouze v HTML a součástí není například PHP server, je vhodné kontrolu privátních komponent implementovat při použití aplikace. K tomuto lze využít například konfigurace Apache anebo Nginx serveru.

Jednoduchá konfigurace je dostupná v souboru *.htaccess* ve složce *components*. Tato konfigurace zajišťuje, že pokud není hlavička Authorization v žádosti vůbec vyplněna, je vrácen chybový kód 401. V případě, že je hlavička vyplněna, ale není validní, je žádost zpracována. Detailnější kontrolu tokenu je možné provést pomocí PHP skriptu, anebo využitím zásuvných modulů do serveru, například mod_authnz_jwt pro server Apache [8].

Kontrola přihlášení při načítání komponent ovšem není kriticky důležitá. Důvodem je to, že komponenta obsahuje pouze kostru, do které jsou vykreslena data načtená z REST API. Při načítání těchto dat dojde k plnohodnotné validaci JWT tokenu a tudíž není možné, aby došlo k zobrazení soukromých dat.

11.3.1 Veřejné komponenty

Mezi veřejné komponenty se řadí tři komponenty - komponenta pro přihlášení, pro odhlášení a komponenta úvodní stránky. Tyto komponenty se nacházejí ve složce *components/public*. Součástí práce není komponenta pro registraci, jelikož registraci je možné provádět pouze pomocí příkazového řádku na serveru.

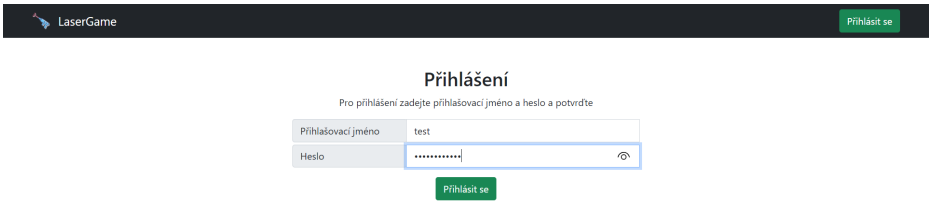
Komponenta pro přihlášení - loginComponent

Komponenta pro přihlášení uživateli zobrazí jednoduchý formulář, do kterého uživatel vloží přihlašovací jméno a heslo, kterým je zaregistrovaný. Po stisknutí tlačítka přihlášení je provedena žádost na REST API endpoint /authenticate, která obsahuje uživatelské jméno a heslo. Pokud je uživatel přihlášen, je vrácen v odpovědi stavový kód 200 OK, a součástí odpovědi je JWT token a poslední stránka, kterou měl uživatel otevřenou. JWT token je uložen do lokálního úložiště prohlížeče a je otevřena stránka, která byla

v odpovědi. Pokud v odpovědi žádná stránka není, je otevřen Ovládací panel.

V případě, že se v odpovědi nachází status 406 NOT ACCEPTABLE, je uživateli zobrazena možnost odhlásit druhý Terminál, na kterém je uživatel přihlášen, a přihlásit se na tomto zařízení. K tomu dojde po odeslání stejné žádosti, ve které je navíc nastavený parametr logover na hodnotu true.

V případě, že uživatel nezadá správné přihlašovací údaje, tak je v odpovědi stavový kód 409 CONFLICT a uživateli je zobrazena informace, že zadal nesprávné přihlašovací jméno anebo heslo.



Obrázek 11.1: Vzhled obrazovky s načtenou komponentou pro přihlášení

Komponenta pro odhlášení - logoffComponent

Komponenta při načtení odešle REST API žádost na endpoint, který odhlásí daného uživatele. JWT token, který je odeslán v této žádosti se stane nevalidní a při jeho dalším použití se vrátí chybový kód 401. Zároveň dojde k jeho odstranění z lokálního úložiště. Po zpracování žádosti se načte komponenta pro přihlášení.

Komponenta úvodní stránky - homepageComponent

Komponenta úvodní stránky je jednoduchá komponenta, která pouze zobrazuje úvodní nadpis a text, oznamující uživateli nutnost přihlášení.

11.3.2 Privátní komponenty

Komponenta ovládacího panelu - dashboardComponent

Pravděpodobně nejvíce využívanou komponentou je komponenta ovládacího panelu. Tato komponenta se primárně zobrazuje po přihlášení a zprostředkovává možnosti pro zobrazení vest (zařízení), které má uživatel zaregistrován a historii her, které má uživatel odehrané, případně které právě běží. Vzhledem k možnému vyššímu počtu odehraných her je tento seznam stránkovaný. Nepředpokládá se, že by uživatel měl zaregistrovaný enormně velký počet vest a proto tato část obrazovky nevyužívá stránkování.

Komponenta herních informací - gameInfoComponent

Informace o hře je možné zobrazit pomocí komponenty herních informací. Pokud je v GET parametru specifikována gameID, jsou zobrazeny informace o hře s daným ID. Pokud toto ID není specifikováno, aplikace se pokusí zobrazit hru, kterou uživatel má právě spuštěnou (to znamená je ve stavu RUNNING).

V levé horní části obrazovky se nachází odpočet zbývajícího času a seznam posledních událostí ve hře. Mezi tyto události se řadí start a konec hry a zásah hráče jiným hráčem. Seznam těchto událostí je dostupný pouze pokud hra právě běží, nebo těsně po ukončení hry. Po opuštění stránky a opětovném otevření detailu je tato část prázdná. Pokud hra je ukončena, místo odpočtu času je zobrazen text *Hra skončila*.

V pravé horní části obrazovky se nachází tabulka, která zobrazuje jména hráčů, jejich přiřazení k týmu, počet zásahů a smrtí a rozdíl mezi počty zásahů a smrtí. Ve spodní části obrazovky se nachází mapa, která zobrazuje aktuální pozici všech hráčů.

Informace jsou pravidelně aktualizovány metodou short-polling. Principem této metody je, že se v zadaném intervalu dotazuje serveru na informace o hře a tyto informace překresluje na obrazovce.

Komponenta pro spuštění hry - startGameComponent

Spuštění nové hry se provádí pomocí komponenty pro spuštění hry. Při vytváření nové hry je nutné, aby už byla založena nová hra, která bude ve stavu CREATED. Při načtení komponenty se načtou informace o právě vytvářené hře. I když se jedná o spuštění nové hry, tyto údaje mohou být předvyplněny, pokud se jedná o kopírování předchozí hry.

Po stisknutí tlačítka pro spuštění hry jsou informace o konfiguraci hry

předány na REST API endpoint pro spuštění hry. Pokud některá hodnota není vyplněna, tak je využita výchozí hodnota, která je specifikována v souboru *constants.js*. Endpoint pro spuštění hry odpovídá až v momentě, kdy má informace o spuštění od všech vest, případně dojde k nějaké chybě při spouštění. Tato chyba se zobrazí v informačním proužku v horní části obrazovky. Uživatel poté musí chybu napravit a spustit hru znovu.

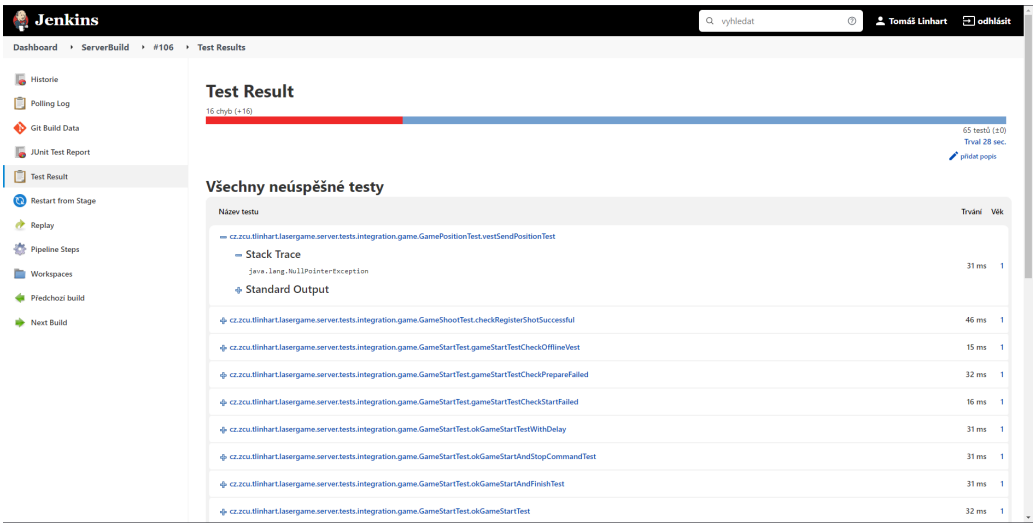
Komponenta pro registraci vesty - registerVestComponent

Novou vestu, kterou uživatel chce zaregistrovat do systému je možné přidat pomocí komponenty pro registraci vesty. Tato komponenta je vzhledem velmi podobná komponentě pro přihlášení uživatele. Nachází se zde vstupní pole, do kterých lze vložit kód vesty, který slouží jako jednoznačný identifikátor dané vesty. Druhým vstupním polem je pole pro vložení názvu vesty.

12 Testování

Pro testování Serverové aplikace je využito testovacího frameworku JUnit, který je běžně využíván pro testování Java kódu. Jednotlivé testy se nacházejí ve složce *src/test*. Tento framework je využitý i pro integrační testování, jenž slouží pro komplexnější ověření funkčnosti aplikace. Automatické spouštění testování zajišťuje krok test, který je spuštěný při sestavení aplikace pomocí frameworku Apache Maven (viz 8.2 Apache Maven). Výsledky automatického testování jsou uloženy v souboru typu XML v adresáři projektu.

Možnosti automatického testování je využito v aplikaci Jenkins, která toto testování spouští automaticky při sestavení a po dokončení běhu pipeline přehledně zobrazuje výsledky testů. Počet procházejících a spadlých testů je viditelný v grafu v pravém horním rohu při zobrazení stránky projektu (viz Obrázek 8.1). Při otevření výsledků testování daného běhu pipeline je možné vidět detailní výsledky včetně chybových zpráv a výpisu z konzole aplikace během testu (viz Obrázek 12.1).



Obrázek 12.1: Ukázka výsledků testování na Jenkins

Aplikace pro Terminál a aplikace pro jednotlivá zařízení jsou testována pouze během reálného testování. Vzhledem ke klíčové úloze Serveru byla jeho testování věnována veškerá pozornost.

12.1 Jednotkové testování

Jednotkové testování bylo využito v rané fázi vývoje aplikace. Je připraveno pouze malé množství jednotkových testů, jenž testují správnou funkčnost třídy *SocketMessage*, jelikož její správná funkčnost je pro kompletní fungování aplikace klíčová. V dalších fázích vývoje bylo využito pouze integrační testování.

12.2 Integrační testování

Integrační testování se používá pro otestování větších celků aplikace. Větším celkem se v kontextu této práce rozumí otestování příslušného REST API endpointu, otestování Socket komunikace a v krajním případě celkové otestování herní logiky Happy Day scénářem.

12.2.1 Testování REST rozhraní

Pro integrační testování jsou připravené testovací třídy v balíčku *cz.zcu.tlinhart.lasergame.server.tests.integration.rest*. Názvy těchto tříd jsou určeny podle kontrolerů, které testují. Pro veškeré integrační testování je připravena třída *GenericIntegrationTest*, jenž poskytuje funkce, které zjednodušují přípravu testů. Využitím této třídy dojde k automatickému spuštění celé aplikace. Jednotlivé kroky testu jsou dokumentovány pomocí komentářů v kódu.

Pro každý endpoint je otestováno, zda nelze údaje získat bez přihlášení. Toto zabezpečení je kritické pro zachování soukromých dat a je proto součástí každého testu. Endpointy jsou poté testovány podle dostupné API dokumentace, kdy jsou testovány scénáře, které mohou nastat. Typicky jsou ověřována data a HTTP stavový kód, který je součástí odpovědi.

Žádosti na endpointy jsou zasílány pomocí frameworku MockMVC, který je součástí Spring Boot. MockMVC nahrazuje odesílání klasické odesílání a přijímání HTTP zprávy, avšak funkcionalita endpointů, včetně ověření autorizace, zůstává zachována. Využitím tohoto testovacího frameworku je dosaženo zrychlení testů a zjednodušení přípravy testovacích scénářů.

12.2.2 Testování Socket Serveru

Socket testy slouží převážně k ověření funkčnosti komunikace mezi Serverem a klientem (vestou), k ověření správného formátu zasílaných dat a k tomu, že jeden klient není ovlivněný jiným klientem. Pro otestování Socket

serveru je připravena třída *GenericSocketTest*, která dědí od třídy *GenericIntegrationTest* a tím rozšiřuje její funkčnost. Při spuštění každého testu dojde k automatickému připojení jednoho Socket klienta na Socket server. Tohoto klienta je možné v testu okamžitě použít pro zasílání a příjem zpráv ze Serveru. V případě nutnosti je možné spustit i klienta druhého, který je nezávislý na prvním klientovi. Vzhledem k tomu, že většina komunikace probíhá až po spuštění hry, je připravena funkce, která zajistí vytvoření nové hry s předem vyplněnými informacemi.

12.2.3 Testování Happy Day scénářem

Testování Happy Day scénářem spočívá v komplexním otestování celého průběhu od spuštění serveru, přes vytvoření a spuštění nové hry, správné zasílání dat z vest až po poskytnutí dat pro zobrazení na Terminálu. Tento test je naprosto klíčový, jelikož jakékoliv jeho selhání signalizuje chybu, která nastane při každém správném běhu aplikace. Během tohoto testu není ověřena reakce serveru na poškození, či zaslání nesprávných dat, reakce na chyby a další události, které typicky nastávají pouze v krajních případech.

12.3 Otestování maximální vzdálenosti pro přenos dat

Maximální možná vzdálenost, po kterou je možné spolehlivě přenést data mezi modulem pro střelbu a modulem pro detekci zásahu, byla změřena pomocí speciálního testovacího scénáře. Modul pro střelbu v periodických intervalech vysílal číslo 15 zakódované ve zprávě, která zároveň obsahovala kontrolní byty, čímž bylo simulováno skutečné přenášení ID hráče. Modul pro detekci zásahu pokaždé, když detekoval zásah a správně přečetl kontrolní byty a datový byte obsahující číslo 15, rozsvítil osvětlení modulu na 500 ms. Tento modul byl postupně posouván směrem od modulu pro střelbu do doby, než přestal správně přijímat data. Tato vzdálenost pak byla změřena.

Test byl proveden za slunečného dne na přímém slunečním osvětlení. Maximální vzdálenost, na které modul správně dekódoval přijatá data byla 48 metrů.

12.4 Reálné ověření funkčnosti

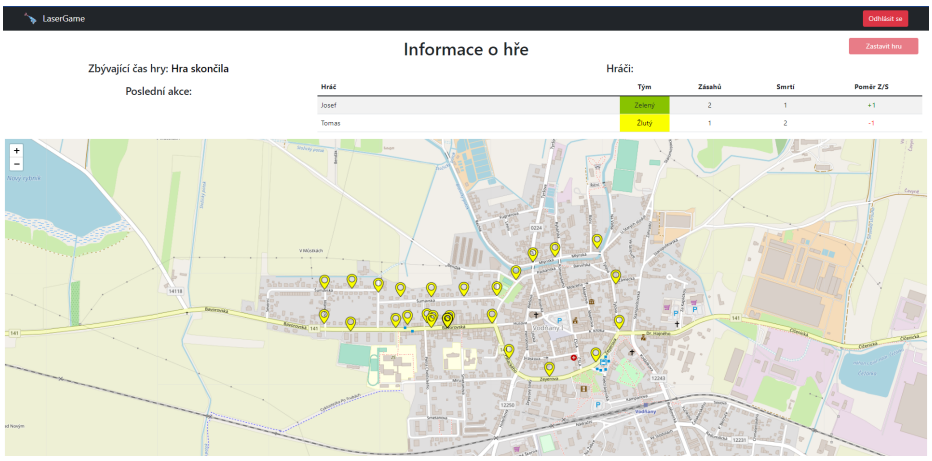
Reálné ověření funkčnosti proběhlo způsobem odehrání několika her s pomocí funkčního Serveru, Terminálu a dvou vest, které byly zaregistrovány

pod jedním uživatelem. Během těchto her byla otestována střelba a detekce zásahů a odesílání GPS souřadnic na server. Odesílání GPS souřadnic bylo ověřeno naložením vesty do auta a krátkou cestou okolo města. Souřadnice byly správně odesílány, odchylka při měření polohy dosahovala zpravidla řádu metrů, avšak odchylka jednoho měření byla cca 8 kilometrů. Součástí testu také byla kontrola správného zobrazení a aktualizace dat na Terminálu. Během tohoto testování se vyskytl malý počet chyb, kdy nedošlo ke správné detekci zásahu a k chybě, kdy byly nové události ve hře zobrazeny na Terminálu vícekrát. Tuto chybu se podařilo lokalizovat a odstranit. Chyba detekce zásahu mohla být způsobena rušením při zasílání dat pomocí sběrnice I2C anebo rušením příchozího signálu vlivem vnějšího světla. Tuto chybu se bohužel nepodařilo odstranit.

Během ověření funkčnosti byla také pozorována spotřeba elektrické energie pomocí jednoduchého měřiče "USB CHARGER Doctor". Vzhledem k metodě pozorování a neuvedené přesnosti měřícího přístroje jsou tyto výsledky pouze orientační. Při inicializaci zařízení dosahuje spotřeby přibližně 2W při napětí 5V. Po úspěšné inicializaci a vyčkávání na spuštění hry spotřeba klesne na přibližně 0.5W. Během hry byla pozorována průměrná spotřeba okolo 1W, při načítání dat z GPS senzoru a jejich odesílání byla spotřeba elektrické energie přibližně 2.5W. V případě, že bychom uvažovali napájení z USB powerbanky o kapacitě 10400 mAh (10.4 Ah) při napětí článků 3.6V a průměrnou spotřebu 1.5W, pak jednoduchým vzorcem zjistíme orientační dobu napájení pomocí této powerbanky.

$$t = (10.4 * 3.6)/1.5 \tag{12.1}$$

Výpočtem je zjištěna doba napájení z této powerbanky přibližně 25 hodin, což je pro hraní dostačující.



Obrázek 12.2: Výsledky poslední testovací hry

13 Uživatelská příručka

13.1 Použití Serveru

Následující kapitoly popisují spuštění a použití aplikace LaserGame Server a možnosti jeho konfigurace.

13.1.1 Spuštění aplikace

Výslednou sestavenou aplikaci ve formátu *.jar* je možné spustit z příkazového řádku na jakémkoliv počítači s nainstalovanou aplikací Java SE verze 11 a vyšší. Po spuštění příkazem *java -jar LaserGameServer.jar* dojde k inicializaci frameworku Spring Boot a načtení hodnot ze souboru application.properties, který je zabalený uvnitř souboru s aplikací. Konfigurační hodnoty je možné změnit i při spuštění aplikace a to přidáním konfiguračního parametru na konec příkazu, například *java -jar LaserGameServer.jar –app.settings.socketServerPort=8080*. Po úspěšném spuštění je zobrazena zpráva *Application is initialized.* a uživateli jsou zobrazeny příkazy, které může použít.

Obrázek 13.1: Vzhled server aplikace po inicializaci.

13.1.2 Možnosti konfigurace

Možnosti konfigurace jsou popsány níže. Konfiguraci je možno měnit pomocí konfiguračního souboru, případně pomocí parametrů při spouštění aplikace.

Konfigurační klíč	app.settings.sessionTimeoutHours
Popis	Doba platnosti JWT tokenu (v hodinách).
Výchozí hodnota	24

Konfigurační klíč	app.settings.passwordHashStrength
Popis	Složitost šifrování hesla. Větší hodnota zvyšuje zabezpečení, avšak vyžaduje více výpočetního výkonu
Výchozí hodnota	10

Konfigurační klíč	app.settings.socketServerPort
Popis	Port na kterém naslouchá Socket server
Výchozí hodnota	8888

Konfigurační klíč	app.settings.socketMessageWarningThreshold
Popis	Počet chybných zpráv, které Socket může přijmout dokud není spojení nuceně ukončeno ze strany Serveru.
Výchozí hodnota	10

Konfigurační klíč	app.settings.socketReadTimeout
Popis	Počet sekund, po které vlákno socketu čeká na přijetí zprávy.
Výchozí hodnota	10

Konfigurační klíč	app.settings.messageAcknowledgeTimeoutSeconds
Popis	Počet sekund, které má klient na potvrzení přijaté zprávy.
Výchozí hodnota	5

Konfigurační klíč	app.settings.acknowledgerCheckPeriod
Popis	Perioda kontroly přijatých potvrzení zpráv.
Výchozí hodnota	1

Konfigurační klíč	app.settings.prepareResponseTimeout
Popis	Časový limit na přijetí všech zpráv PREPARE-DONE po odeslání PREP zprávy klientovi.
Výchozí hodnota	15

Konfigurační klíč	app.settings.startResponseTimeout
Popis	Časový limit na přijetí všech zpráv STARTDONE po odeslání START zprávy klientovi.
Výchozí hodnota	15

Tabulka 13.1: Příkazy, které lze použít v konzoli Serveru

13.1.3 Použití příkazů

Server aplikaci je možné ovládat zadáváním textových příkazů do konzole. Tyto příkazy se uživateli zobrazí po spuštění aplikace, případně při zadání příkazu *help* anebo zadání neznámého příkazu. Příkazy jsou popsány v tabulce níže.

Příkaz	help
Akce	Zobrazí v konzoli tabulku s příkazy a krátkým popisem.

Příkaz	register [username] [password]
Akce	Zaregistruje nového uživatele se zadaným uživatelským jménem a heslem. Informuje uživatele, zda byla registrace úspěšná.

Příkaz	logout [username]
Akce	Odhlásí uživatele se zadaným jménem.

Příkaz	setstatus [vestCode] [ONLINE|OFFLINE]
Akce	Nastaví hodnotu statusu pro danou vestu na ONLINE anebo OFFLINE. Použito pouze pro ladění.

Příkaz	quit
Akce	Ukončí aplikaci.

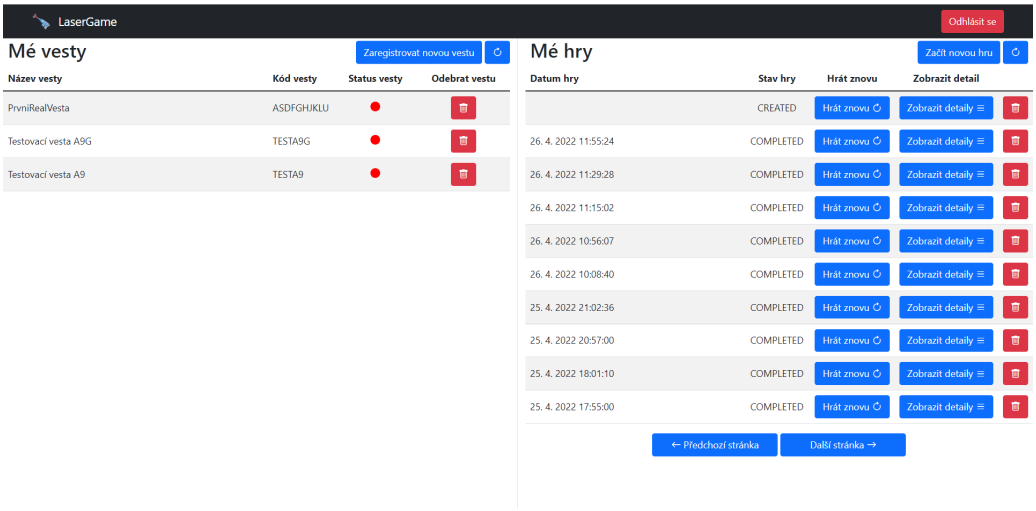
Tabulka 13.2: Příkazy, které lze použít v konzoli Serveru

13.2 Popis obrazovek Terminálu

13.2.1 Obrazovka ovládacího panelu

Obrazovka ovládacího panelu je výchozí obrazovka po přihlášení do aplikace. V levé části obrazovky je umístěna tabulka obsahující vesty, které má uživatel zaregistrované a tlačítko pro registraci nové vesty. V pravé části se nachází tabulka obsahující seznam her, které uživatel odehrál a tlačítko pro spuštění nové hry.

Ze seznamu her je možno daný záznam hry smazat, zobrazit detaily (přesměruje uživatele na danou komponentu) a spustit danou hru znovu. To uživatele přesměruje na komponentu pro spuštění hry, avšak hráči a nastavení jsou předvyplněny ze hry, která se spouští znovu.

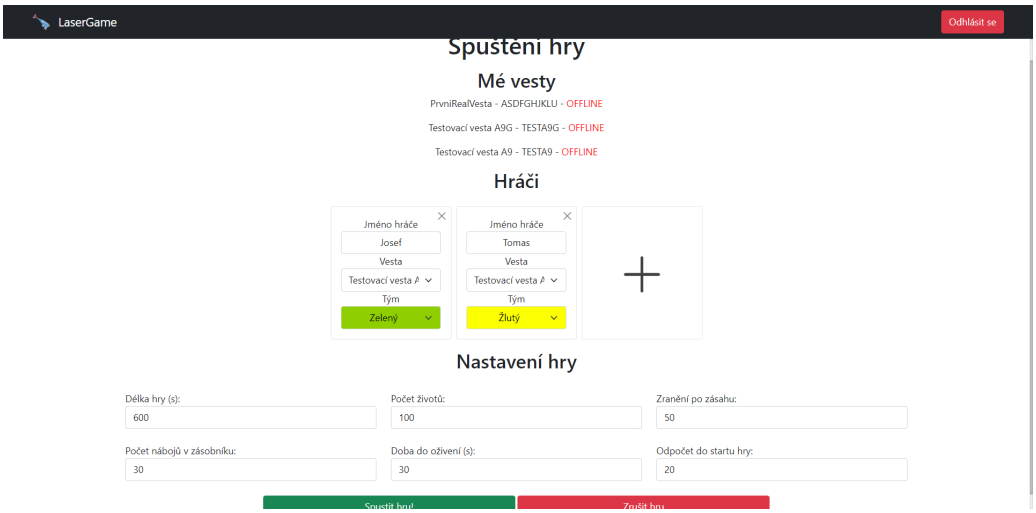


Obrázek 13.2: Vzhled obrazovky s načtenou komponentou ovládacího panelu

13.2.2 Obrazovka spuštění hry

V horní části obrazovky se nachází seznam vest, které lze použít pro hru. Níže se nachází seznam hráčů, kteří budou danou hru hrát. Hráčům je možné přiřadit vestu ze seznamu vest, které má daný hráč zaregistrované. Hráčům lze v daném poli přiřadit tým a jejich jméno. Nového hráče lze přidat pomocí tlačítka, který se nachází na konci seznamu a odstranit pomocí tlačítka, který se nachází v pravém horním rohu okénka daného hráče.

Ve zbylé části obrazovky se nachází vstupní pole pro nastavení parametrů hry, jako například doba hry, počet životů hráčů, množství nábojů v zásobníku a další a tlačítka pro spuštění či odstranění hry.

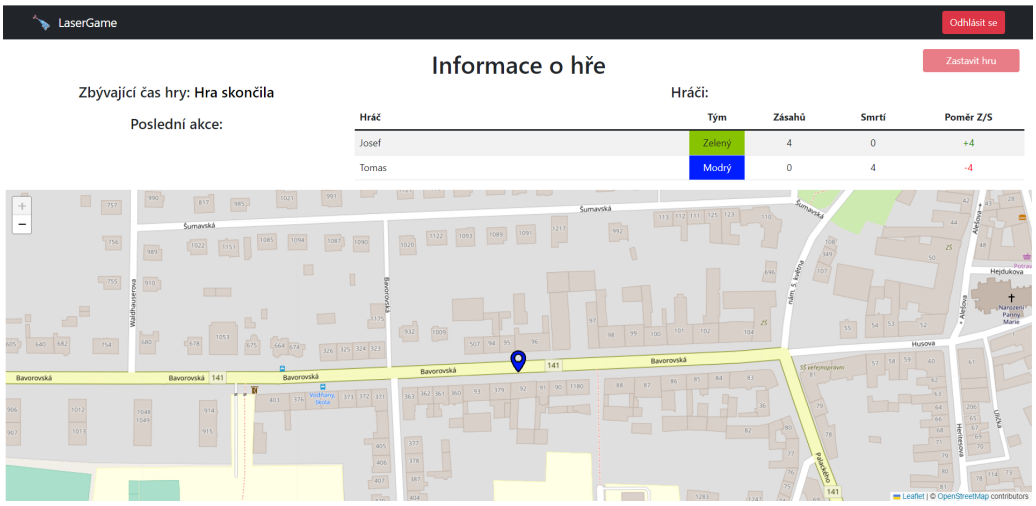


Obrázek 13.3: Vzhled obrazovky s načtenou komponentou pro spuštění hry

13.2.3 Obrazovka informací o hře

V horní části obrazovky se nachází seznam akcí, které se ve hře staly a tabulka obsahující seznam hráčů, jejich tým, počet zásahů, smrtí a jejich poměr. Pokud je hodnota poměru zásahů a smrtí kladná, pak určuje o kolik zásahů má více, než smrtí. Pokud je záporná , tak určuje o kolik smrtí má uživatel více než zásahů.

Po ukončení hry jsou zobrazeny všechny pozice, na kterých se hráči během dané hry nacházeli. Jednotlivé pozice jsou reprezentovány ikonou s barvou týmu daného hráče. Po kliknutí na tuto ikonu se zobrazí dialogové okno, ve kterém se nachází datum a čas, kdy byla pozice uložena a jméno hráče, jenž se na této pozici nacházel.



Obrázek 13.4: Vzhled obrazovky s načtenou komponentou herních informací

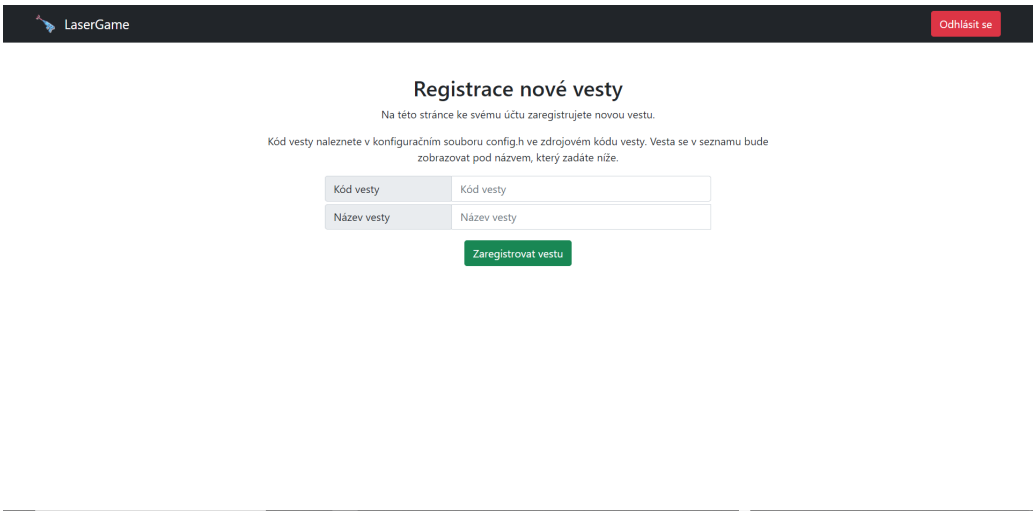
13.2.4 Obrazovka pro přihlášení

Obrazovka pro přihlášení a registraci jsou vzhledem velmi podobné. Na obrazovce pro přihlášení se nachází vstupní pole pro uživatelské jméno a heslo a tlačítko pro přihlášení. V případě nesprávných přihlašovacích údajů je zobrazena chybová hláška.

Pokud je uživatel přihlášený na jiném terminálu, musí přihlášení opakovat, čímž uživatele z prvního terminálu odhlásí.

13.2.5 Obrazovka pro registraci vesty

Místo toho na obrazovce pro registraci vesty se nachází vstupní pole pro kód vesty a název vesty. Tento název se zobrazuje v tabulce na komponentě ovládacího panelu a při spouštění nové hry. Pokud se uživatel pokusí přidat kód vesty, kterou má zaregistrovanou jiný hráč, je zobrazena chybová hláška a uživatel musí kód vesty změnit a registraci opakovat.



Obrázek 13.5: Vzhled obrazovky s načtenou komponentou pro registraci vesty

13.3 Použití kompletní aplikace

Po registraci a přihlášení nového uživatele je nutné nejdříve zaregistrovat nové vesty, se kterými bude uživatel hrát. Tuto registraci provede na obrazovce pro registraci vesty. Kód vesty, který uživatel zaregistruje je poté nutné nahrát do vesty jeho změnou v konfiguračním souboru config.h, zkompilováním kódu hlavního modulu a jeho nahráním do daného modulu.

Uživatel poté může spustit novou hru. Přidá nové hráče, kteří budou hrát danou hru (maximálně 8 hráčů) a přidělí jim tým a vestu z rozbalovacího seznamu. Veškeré vesty musí být spuštěny a přihlášeny k serveru. To lze zjistit na displeji vesty.

Obrázek 13.6: Displej vesty oznamující správné připojení k serveru a připravenost ke hře

Uživatel poté vyplní zbylé hodnoty a stiskne tlačítko Spustit hru, čímž

zahájí odpočet do startu hry. Na Terminálu se objeví obrazovka s informacemi o právě běžící hře.

Obrázek 13.7: Displej odpočítávající start hry a deaktivovaný modul pro detekci zásahu

Odpočet slouží pro rozptýlení hráčů po hřišti. Po uplynutí odpočtu se vesty aktivují a hráč může začít střílet po ostatních hráčích, případně být zasažen jiným hráčem. Při stisknutí tlačítka pro střelbu dojde k vystřelení pouze jednoho náboje. Pro střelbu dalšího náboje je nutné tlačítko uvolnit a stisknout znovu. O střelbě je uživatel informován pomocí zvuku z reproduktoru, který se nachází u hlavního modulu. Pokud hráči dojdou náboje, o čemž je informován také pomocí zvuku, je nutné zbraň přebít stisknutím příslušného tlačítka. Do té doby je zbraň deaktivována, avšak hráč může být zasažený jinými hráči. Zbraň je možné libovolně přebíjet i pokud zásobník není úplně prázdný.

Obrázek 13.8: Celkový pohled na hardware při spuštěné hře

Při zasažení je hráči odečten počet životů zadaný v konfiguraci při spouštění hry. Zároveň je zobrazena informace o tom, kdo hráče zasáhl. Pokud hráči zbývá nula životů, je připsán zásah hráči, který aktuálního hráče zasáhl. Aktuální hráč musí vyčkat dokud není oživen a nemůže pokračovat ve hře. Doba do oživení je nastavena při startu hry.

Obrázek 13.9: Displej zobrazující čas do oživení hráče

Po ukončení hry je hráčům na vestě zobrazena informace o konci hry. Vesta je poté deaktivována a připravena k další hře. Na Terminálu je zob-

razena informace, že hra již skončila a jsou zobrazeny veškeré pozice hráčů, na kterých se během hry nacházeli (viz Obrázek 12.2). Po ukončení hry je možné vytvořit novou hru, případně hru opakovat, čímž dojde k vytvoření nové hry s údaji předvyplněnými ze hry, která se opakuje. Hru je možné také ukončit pomocí tlačítka v pravém horním rohu na obrazovce s informacemi o právě běžící hře.

14 Závěr

Během práce byla prozkoumána existující řešení, které se již využívají pro hru LaserGame a návrh vlastního řešení práce. Výsledkem je funkční Server pro spuštění hry LaserGame, aplikace Terminál pro nastavení jednotlivých her a hardware a software vest a zbraní, které hráči používají při hře.

Server připravený v jazyce Java implementuje REST API rozhraní pro komunikaci s Terminálem a Socket server pro komunikaci s vestami v reálném čase. Data jsou ukládána do relační databáze. Server je částečně otestován jednotkovým a integračním testováním a pomocí automatického happy day scénáře. Součástí je standardní JavaDoc dokumentace a dokumentace REST API rozhraní pomocí OpenAPI specifikace.

Webová stránka Terminál umožňuje snadné nastavení jednotlivých her a snadnou správu vest pro daného uživatele. Aplikace je připravena v jazyce HTML a Javascript.

Hardware vest je rozdělený na jednotlivé moduly. Software pro jednotlivé moduly vesty je připravený v jazyce C. Pro komunikaci mezi zařízeními je využito modulovaného infračerveného paprsku, jenž je kolimován pomocí vlastní navržené optiky. Funkčnost hardwaru a softwaru byla ověřena na dvou testovacích zařízeních. Celé řešení obsahuje několik drobných nedostatků, jako je například občasné selhání detekce zásahu, jenž by bylo možné vyřešit změnou napětí na datové sběrnici na 3.3V a vyřazení logického převodníku hodnot, případně změnou typu kabelu datové sběrnice.

Dalším možným rozvojem práce je návrh a vytvoření tištěných desek plošných spojů, přidání nových herních modů a nastavení, přidání možnosti omezení velikosti herního pole, případně vytvoření vesty a zbraně, na kterých bude umístěný hardware navržený v této práci.

Literatura

[1] Jenkins.io, 2022. Dostupné z: https://www.jenkins.io/.

[2] Dostupné z: https://www.carove-lasery.cz/soubor-maximalni-povoleny-vykon-pro-jednotlive-tridy-v-zavislosti-na-vlnove-delce-10-.pdf.

[3] Adafruit. Adafruit/ADAFRUIT_NEOPIXEL: Arduino Library for controlling single-wire led pixels (Neopixel, WS2812, etc..), 2022. Dostupné z: https://github.com/adafruit/Adafruit_NeoPixel.

[4] Ai-Thinker. A9. Dostupné z: https://ai-thinker-open.github.io/GPRS_C_SDK_DOC/en/hardware/a9.html.

[5] Ai-Thinker-Open. Ai-Thinker CSDTK4.2 SDK kit, Apr 2022. Dostupné z: https://mega.nz/folder/WiZlDSJZ#ftSjd5U5vRBolHCyd_hE3A.

[6] Ai-Thinker-Open. Ai-Thinker GPRS C SDK Documentation, Apr 2022. Dostupné z: https://ai-thinker-open.github.io/GPRS_C_SDK_DOC/en/.

[7] Ai-Thinker-Open. A9G SDK, Apr 2022. Dostupné z: https://github.com/Ai-Thinker-Open/GPRS_C_SDK.

[8] AnthonyDeroche. Anthonyderoche/MOD_AUTHNZ_JWT: An authentication module for Apache httpd using JSON web tokens, 2022. Dostupné z: https://github.com/AnthonyDeroche/mod_authnz_jwt.

[9] Arduino-IRremote. Arduino-irremote/arduino-irremote: Infrared Remote Library for arduino: Send and receive infrared signals with multiple protocols, 2022. Dostupné z: https://github.com/Arduino-IRremote/Arduino-IRremote.

[10] Churý, L. *Mistrovství Česka a Slovenska v Laser Game 2019* [online]. 2019. Dostupné z: https://mcslg.cz/o-turnaji/.

[11] GM Electronic, s. s. Vývojový kit Arduino Uno R3 - klon, 2022. Dostupné z: https://www.gme.cz/klon-arduino-uno-r3-atmega328p-ch340-mini-usb.

[12] GM electronic, s. s. Fotodioda BPV10NF, . Dostupné z: https://www.gme.cz/bpv10nf.

[13] GM ELECTRONIC, s. s. Infra LED 5mm 130mW/10°TSAL6100, . Dostupné z: https://www.gme.cz/infra-led-5mm-tsal6100.

[14] JAVAINUSE. Spring Boot Security + JWT hello world example. Dostupné z: https://www.javainuse.com/spring/boot-jwt.

[15] KREJČÍ, T. LaserTag comunity., Dec 2018. Dostupné z: https://openlasertag.org/language/en/.

[16] KREJČÍ, T. OpenLaserTag IR protocol, Feb 2018. Dostupné z: https://openlasertag.org/language/en/openlasertag-ir-communication-protocol/.

[17] KREJČÍ, T. Optical emmiter tube, Aug 2020. Dostupné z: https://openlasertag.org/language/en/optical-emmiter-tube/.

[18] MACDYNAMO – INSTRUCTABLES. *DIY Laser Tag System (Revamped)* [online]. Instructables, Nov 2017. Dostupné z: https://www.instructables.com/DIY-Laser-tag-system-from-simple-to-complex/.

[19] OPENAPITOOLS. OpenAPITools/openapi-generator: Openapi generator allows generation of API client libraries (SDK Generation), server stubs, documentation and configuration automatically given an openapi Spec (V2, v3), 2022. Dostupné z: https://github.com/OpenAPITools/openapi-generator.

[20] OPSWEB1. Laser Classification Explanation, Jul 2018. Dostupné z: https://ehs.lbl.gov/resource/documents/radiation-protection/laser-safety/laser-classification-explanation/.

[21] PULKIN. pulkin/micropython, Sep 2020. Dostupné z: https://github.com/pulkin/micropython/tree/master/ports/gprs_a9.

[22] S.R.O., C. ESP-32s ESP32 Development Board 2.4GHz dual-mode WIFI+Bluetooth antenna module: Dratek.cz, 2022. Dostupné z: https://dratek.cz/arduino/1581-esp-32s-esp32-esp8266-development-board-2.4ghz-dual-mode-wifi-bluetooth-antenna-module.html.

[23] XASIN. LZRTag - Flexible DIY Lasertag, Aug 2018. Dostupné z: https://hackaday.io/project/160804-lzrtag-flexible-diy-lasertag.

[24] XASWORKS. XasWorks/LZRTag, 2019. Dostupné z: https://github.com/XasWorks/LZRTag.