Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Bezdrátová časomíra pro požární útok

Karel Wiedermann

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Karel Wiedermann

Informatika

Název práce

Bezdrátová časomíra pro požární útok

Název anglicky

Wireless timer for fire attack

Cíle práce

Cílem bakalářské práce je vytvoření bezdrátové časomíry pro potřeby požárního sportu za použití dvou plošných spojů arduino uno, mezi kterými bude vytvořena bezdrátová komunikace.

Metodika

Prvním krokem bude vytvoření návrhu zapojení pomocí sotwaru Fritizing. V druhém kroku budou desky zapojeny s komponentami pomocí nepájivého pole. Následují kroku budou vytvořeny řídící programy. Před spájením elektroniky a umístěním do pvc boxů, proběhnout funkční testy. Po u místění elektroniky do pvc boxů, budou provedeny akceptační testy.

Doporučený rozsah práce

30-40 stran

Klíčová slova

požární sport, časomíra, hasiči, Arduino, bezdrátová komunikace

---

Doporučené zdroje informací

BANZI, Massimo. Getting started with Arduino. 2nd ed. Farnham: O'Reilly, 2011. ISBN 9781449309879.

SELECKÝ, Matúš. Arduino: uživatelská příručka. Přeložil Martin HERODEK. Brno: Computer Press, 2016. ISBN 9788025148402.

TONY OLSSON .. [ET AL.]. Open + software: fashionable prototyping and wearable computing using the Arduino. 2nd ed., revised & expanded. S.l.: Blushing Boy, 2011. ISBN 9789197955409.

VODA, Zbyšek. Průvodce světem Arduina. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

Předběžný termín obhajoby

2017/18 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 11. 1. 2018

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 11. 1. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 13. 03. 2018

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Bezdrátová časomíra pro požární útok" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2018 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Markovi Píckovi, Ph.D. za odborné vedení práce, za praktické rady a za možnost využít jeho zkušeností v této problematice. Dále bych rád poděkoval Bc. Janu Čáslavovi ze Sboru dobrovolných hasičů Ratenice za praktické konzultace problematiky časomíry a terčů.

Bezdrátová časomíra pro požární sport

Abstrakt

Tato práce řeší vytvoření bezdrátové časomíry pro požární sport. Cílem je použít dvou desek Arduino, jedna bude server a druhá klient. Zaměřil jsem se hlavně na bezdrátovou komunikaci mezi deskami. Zvolený problém jsem vyřešil pomocí dvou desek Arduino a přídavných modulů nRF24l01 pracujících v 2.4G ISM pásmu. Pro svou práci jsem použil nástroj Fritizing. Podařilo se dosáhnout 100% úspěšnosti pro přenášená data na volném prostranství. V práci jsem vytvořil systém, který není dodnes v prodeji na českém trhu. Přínosem této práce je vytvoření jednoduchého řešení časomíry pro požární sport s možným modulárním rozšířením pro jednotlivé disciplíny požárního sportu.

Klíčová slova: požární sport, časomíra, hasiči, Arduino, bezdrátová komunikace

Wireless timer for fire attack

Abstract

This bachelor thesis deals the creation of wireless timer for fire sport. The goal is to use two Arduino boards, one server and one client. I focused mainly on wireless communication between boards. I have solved the problem by using two Arduino boards and additional modules nRF24l01 working in the 2.4G ISM band. I used the Fritizing tool for my work. We managed to achieve 100% success rate for transmitted data in the open air. At work, I created a system that is not currently available on the Czech market. The benefit of this work is the creation of a simple solution for a fire sport timer with a possible modular expansion for the individual disciplines of fire sport.

Keywords: fire sport, timer, firefighters, Arduino, wireless communication

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

HZS ČR - Hasičský záchranný sbor České republiky
PVC - Polyvinylchlorid
USB - Universal Serial Bus
PWM - Pulse Width Modulation
ICSP - In-Circuit Serial Programming
I2C – Internal Integrated Circuit
PC - personal computer
PCB - Printed Circuit Board
ASCII - American Standard Code for Information Interchange
ISM - industrial, scientific and medical
PPR – Polypropylen random

1 Úvod

Hasičský záchranný sbor České republiky (HZS ČR), jednotky sboru dobrovolných hasičů, Policie České republiky a zdravotnická záchranná služba jsou základními složkami Integrovaného záchranného systému. Jejich vzájemná součinnost se odráží v přípravě na mimořádné události, při provádění záchranných a likvidačních prací. Ve své práci jsem se zaměřil na první zmiňovanou složku – HZS ČR, konkrétně na její podporu při přípravě, pro požární soutěže.

Jelikož jsem členem Sboru dobrovolných hasičů Sendražice a aktivně se věnuji požárnímu sportu, vybral jsem si téma, kterým chci zvýšit prestiž sborem pořádaných závodů a rozšířit tréninkové vybavení sboru.

Tato práce se zabývá problematikou časomíry v požárním sportu. Časomíra je v dnešní době nezbytnou součástí nejen v požárním sportu, ale i ve většině ostatních sportů. Neustálé zlepšování soutěžících a stále větší nároky na přesnost měřených výsledků vytváří neustálý tlak na vývoj stále dokonalejších způsobů, jak zaznamenávat, kontrolovat a vyhodnocovat co nejpřesněji časové výsledky. Je požadována naprostá přesnost, spolehlivost, jednoduchost ovládání, jednoduchá manipulace, v neposlední řadě vysoká mechanická odolnost, voděodolnost a práce při extrémních klimatických podmínkách.

Je zde důležitá vlastnost celého systému, který by měl být jednoduše rozkládatelný. Další důležitou vlastností je ovladatelnost, aby i člověk bez elektrotechnického vzdělání dokázal bezproblémově manipulovat a ovládat časomíru.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem bakalářské práce je vytvoření bezdrátové časomíry pro požární sport za použití desek Arduino, mezi kterými bude vytvořena komunikace přes NRF24L01+ moduly.

2.2 Metodika

- Vytvoření návrhu zapojení pomocí softwaru Fritizing
- Zapojení desek pomocí nepájivého pole
- Vytvoření řídících programů
- Funkční testy

- Spájení elektroniky
- Umístění do PVC boxů
- Vytvoření modelů terčů v Autocad Autodesk
- Akceptační testy

3 Teoretická východiska

3.1 Arduino UNO R3

Arduino UNO je mikrokontrolérová vývojová deska používající čip ATmega328. Deska obsahuje 14 digitálních vstupních a výstupních pinů (z toho může být 6 použito jako výstupy PWM), 6 analogových vstupů, 16 MHz krystal, připojení pomocí USB, napájecí konektor, ICSP rozhraní a resetovací tlačítko. Obsahuje vše potřebné k provozu mikrokontroléru. S deskou se lze naučit pracovat s mikroprocesory za použití speciálního jazyku Wiring. [1], [2]

3.1.1 Specifikace

- MCU: ATmega328
- Pracovní napětí: 5V
- Vstupní napětí: 7-12V
- Vstupní napětí max.: 6-20V
- I/O Piny: 14 (6 použitelných jako PWM výstup)
- Analogové vstupy: 6
- DC Proud na pin: 40 mA
- Flash: 32 KB (ATmega328) 0.5 KB použito pro bootloader
- SRAM 2 KB (ATmega328)
- EEPROM 1 KB (ATmega328)
- Krystal: 16 MHz [3]

3.2 Arduino MEGA 2560

Arduino MEGA 2560 je větším klonem klasického Arduina UNO. Oproti klasickému Arduinu UNO má Arduino MEGA 2560 rychlejší procesor ATMega2560 a také více vstupních a výstupních pinů.

Arduino MEGA je určeno pro náročné uživatele, kteří potřebují ovládat velké množství periferních zařízení. Arduino MEGA má 53 digitálních vstupních a výstupních pinů. Mezi nimi jsou 4 hardwarové sériové porty, 14 pulzně šířkových pinů a I2C rozhraní. Navíc k těmto 53 portům Arduino MEGA nabízí 16 vstupních analogových portů. Navíc

proti klasickému Arduinu obsahuje procesor Arduina MEGA větší množství operační paměti a paměti pro kód. [1], [4]

3.2.1 Specifikace

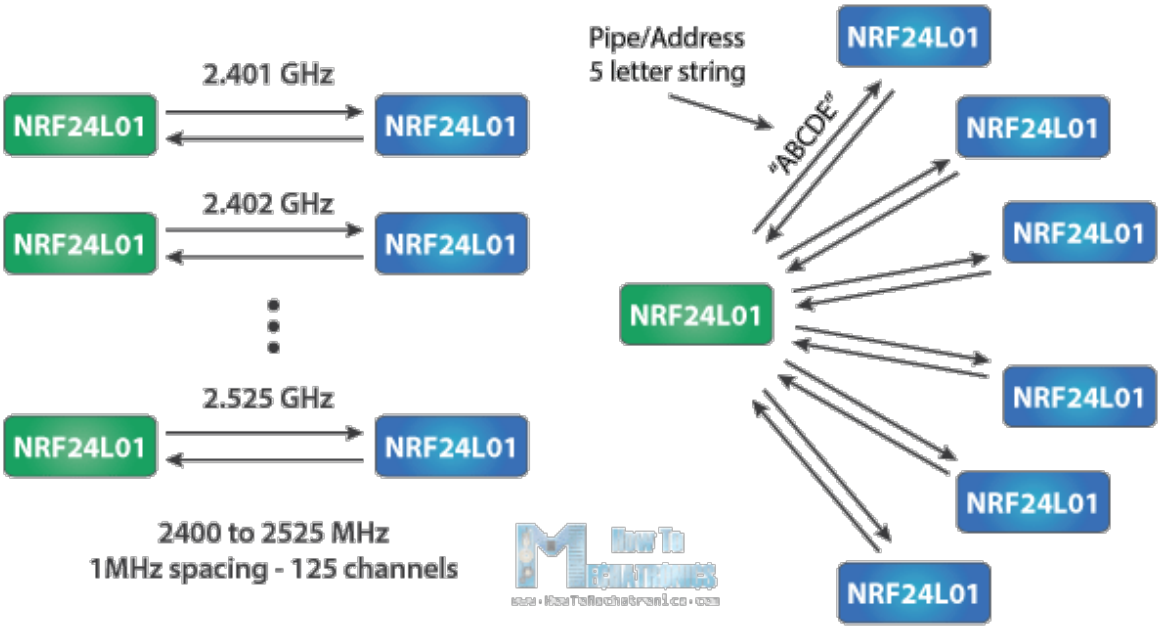
- Mikrokontrolér: ATmega2560
- Taktovací frekvence: 16 MHz
- EEPROM: 4 KB
- SRAM: 8 KB
- Flash paměť: 256 KB z toho 8 KB používá zavaděč
- Provozní napětí: 5V
- Vstupní napětí: 7-12V
- Analogové piny: 16
- Digitální I / O Piny: 54 (z toho 14 PWM)[5]

3.3 Arduino WiFi modul nRF24L01

Arduino WiFi modul nRF24L01 je bezdrátový modul, který umožňuje komunikaci mezi Arduino deskami. WiFi v názvu modulu označuje pracovní frekvenci, která je 2,4 GHz a právě tato frekvence je běžně využívána pro propojování počítačů či telefonů s bezdrátovými routery. Je ale nutné zdůraznit, že není možné se pomocí modulu nRF24L01 připojit na domácí síť WiFi a potažmo internet. Pro tyto účely slouží například ESP8266. Co se týká napájení, bezdrátový modul nRF24L01 vyžaduje napětí 3,3 V, ale datové piny jsou schopné pracovat s 5 V logikou Arduino desky a není tedy nutné používat žádné převody napájecích úrovní. Vysílací výkon nRF24L01 lze nastavit ve čtyřech úrovních od MIN do MAX (viz ukázkový kód), ale pro úrovně HIGH a MAX je doporučeno použít externí zdroj napětí 3,3 V, protože pro tyto vysílací výkony už není dostačující maximální proud, který dokáže dát stabilizátor na Arduino deskách. Co se týká proudového odběru, v Power down režimu má nRF24L01 udávanou spotřebu jen 1 mikroAmpér. [1], [6]

3.3.1 PiPe

Modul nRF24l01 využívá 125 různých kanálů, které umožňují vytvořit na jednom místě síť 125 nezávisle pracujících modemů. Každý může obsahovat až 6 adres nebo každá jednotka může komunikovat současně až s 6 dalšími jednotkami. Pro deklaraci adresy kanálu se používá libovolný textový řetěz o délce pěti znaků. [1], [6]



Obrázek 1 - Popis Pipe [7]

3.4 Jazyk Wiring

Wiring je programovací jazyk vytvořený pro programování mikrokontroléru bez specifických znalostí hardware. V současné době je nejznámější jako součást open-source platformy Arduino, kde má podobu frameworku v jazyce C++. Pro programování v jazyce Wiring se nejčastěji používá integrované vývojové prostředí Arduino IDE, k dispozici jsou ale i další vývojová prostředí například Arduino Eclipse plugin. Wiring vyžaduje mikrokontrolér se zaváděcím programem, typicky desku Arduino osazenou čipy ATmega. Prvotním autorem jazyka je Hernando Barragán, který ho definoval ve své diplomové práci na italském institutu IDII (Interaction Design Institute Ivrea) jako součást prototypovacích nástrojů pro elektroniku a programování. [1], [8]

Program v jazyce Wiring se nazývá sketch a typicky má dvě hlavní části:

setup – funkce, která se spustí jednou na začátku programu a zpravidla obsahuje počáteční nastavení. [8]

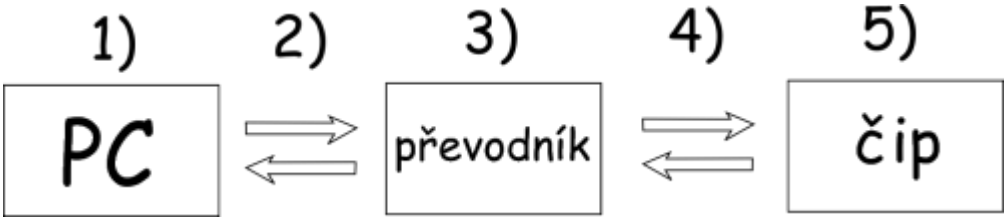
loop – funkce volaná v nekonečné smyčce, když je deska Arduino připojena k napájení. [8]

Pro pokročilejší programování Arduina se používá jazyk C++. Jeho kód se zapisuje běžným způsobem přímo do kódu ve Wiring, protože Arduino IDE používá překladač C++. [9]

3.4.1 Sériová komunikace

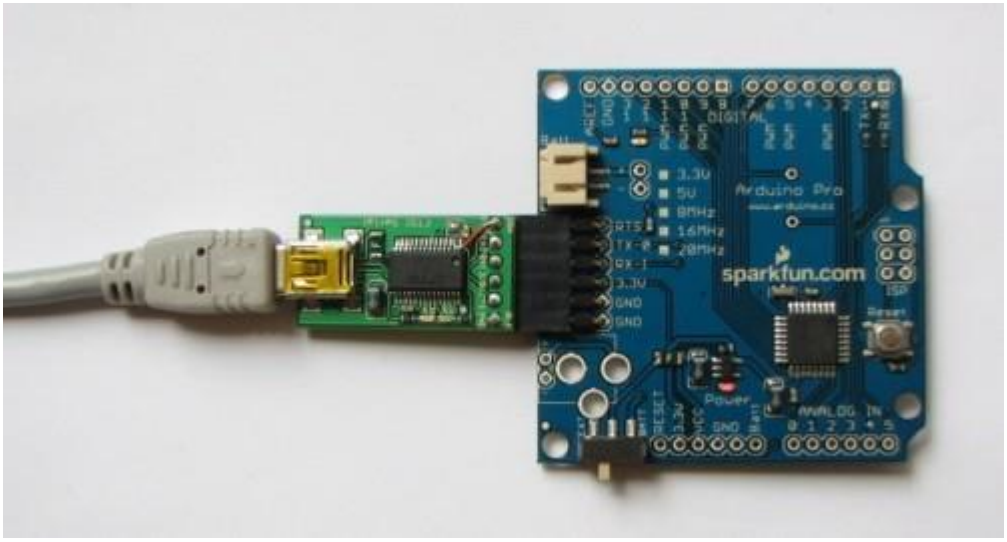
Aby mohlo Arduino správně komunikovat s PC, musí mít několik základních součástí. Následující popis postihuje většinu desek Arduino. Najdou se však i speciální desky, které podporují jiný způsob programování (BT, Ethernet, Wifi…). Proces programování většiny Arduino desek je následující:

1. Základním předpokladem je mít PC s USB portem.
2. USB kabel
3. S převodníkem se nám schéma trochu komplikuje. Můžeme se totiž setkat se třemi základními typy převodníku. Všechny tři převodníky fungují stejně. Liší se pouze způsobem připojení.
    a. Převodník, který je na pevno připájený k základní desce Arduina.
    b. Převodník, který mají některé čipy (ATmega32u4…) přímo v sobě.
    c. Externí převodník, který musíme při programování Arduina připojit.
4. Připojení převodníku a čipu. Při použití externího převodníku se většinou jedná o šest vodičů, které vystupují z desky. U zbylých dvou typů převodníku jsou to pouze kontakty na plošném spoji, nebo propojení uvnitř čipu.
5. Mozek, který přijímá přeložené instrukce od převodníku. [9]



Obrázek 2 - Nahrání kódu na čip [9]

Na obrázku 3 je zobrazeno Arduino s externím převodníkem kódu. [9]



Obrázek 3 - Externí převodník [9]

Sériová komunikace se ale dá využít k více věcem než jen k programování. Pomocí ní totiž můžeme komunikovat s Arduinem, i když už na něm běží náš program. Poté můžeme například číst hodnoty ze senzorů a posílat je do PC, nebo můžeme ovládat Arduino jednoduchými textovými příkazy. [9]

3.4.2 Digitální vstup a výstup

Jelikož je Arduino určeno k dalšímu rozšiřování, obsahuje vstupy a výstupy (nazývané piny), ke kterým se dají vodičem připojit další obvody, čipy, relé, paměti. K práci s těmito piny má Arduino k dispozici jednoduché funkce. Nejdříve ze všeho je však potřeba programu říci, jestli s pinem pracujeme jako se vstupem, nebo výstupem. [1], [9]

3.4.3 Vstup nebo výstup?

K nastavení pinu slouží funkce pinMode. Ta pro svoji správnou činnost potřebuje dva vstupní parametry – pinMode (číslo_pinu, INPUT/OUTPUT). Pokud chceme daný pin používat jako vstup, bude druhý parametr INPUT, pokud jako výstup, bude to OUTPUT. Číslo pinu je většinou natištěno na desce Arduina. Ve verzi Arduino UNO tedy můžeme používat piny 0 až 13. Tím to ale nekončí, protože můžeme k digitálním operacím používat i piny označené jako ANALOG IN, jenom místo samotného čísla musíme před číslo napsat A. U Arduina UNO jsou to tedy A0 až A5. [1], [9]

```
int cislo = 13;
pinMode(cislo, OUTPUT); //nastavení pinu 13 na výstup
```

```
pinMode(12, INPUT); //a pinu 12 na vstup
```

3.4.4 Ovládání výstupu

K ovládání výstupu se používá funkce digitalWrite. Stejně jako pinMode potřebuje i tato funkce dva parametry – číslo pinu a informaci o proudu. Pokud proud teče, je to HIGH, pokud ne, tak LOW. [1], [9]

```
digitalWrite(13, HIGH);
digitalWrite(12, LOW);
```

3.4.5 Čtení vstupu

Ke zjištění, zda proud do vstupu teče, nebo ne, se používá funkce digitalRead. Funkce potřebuje pouze jeden parametr, kterým je číslo pinu. Tato funkce navíc vrací hodnotu. Když proud teče, vrátí hodnotu HIGH, když ne, tak LOW. [1], [9]

```
int cteni;
int vstup = 13;
cteni = digitalRead(vstup); //pokud proud teče, do proměnné cteni se
uloží hodnota HIGH
//pokud ne, tak LOW
```

3.5 Pointer

Některé úlohy programování v jazyce C se provádějí snadněji pomocí ukazatelů a jiné úkoly jako je dynamická alokace paměti, nelze provést bez použití ukazatelů. Každá proměnná je paměťové místo a každá paměťová poloha má definovanou adresu, ke které lze přistupovat pomocí operátoru ampersand (&), který označuje adresu v paměti.

3.5.1 Co jsou ukazatele?

Ukazatel je proměnná, jejíž hodnota je adresa další proměnné, tedy přímá adresa místa v paměti. Stejně jako libovolná proměnná nebo konstanta, musíte před použitím použít ukazatel pro uložení libovolné proměnné adresy. Obecná forma prohlášení proměnné ukazatele je:

```
type *var-name;
```

3.6 Požární útok

Požární útok je mezi hasiči označován jako „královská disciplína“. Je to z toho důvodu, že se musí skloubit mnoho faktorů, vnitřních i vnějších, kterých je zapotřebí, aby se útok vydařil.

Na soutěžích v požárním útoku se snaží sedmičlenná družstva co nejrychleji sepnou spínač na terčích. Voda nabraná z kádě putuje savicemi přes přenosnou hasičskou stříkačku do hadic přes rozdělovač až do proudnic.

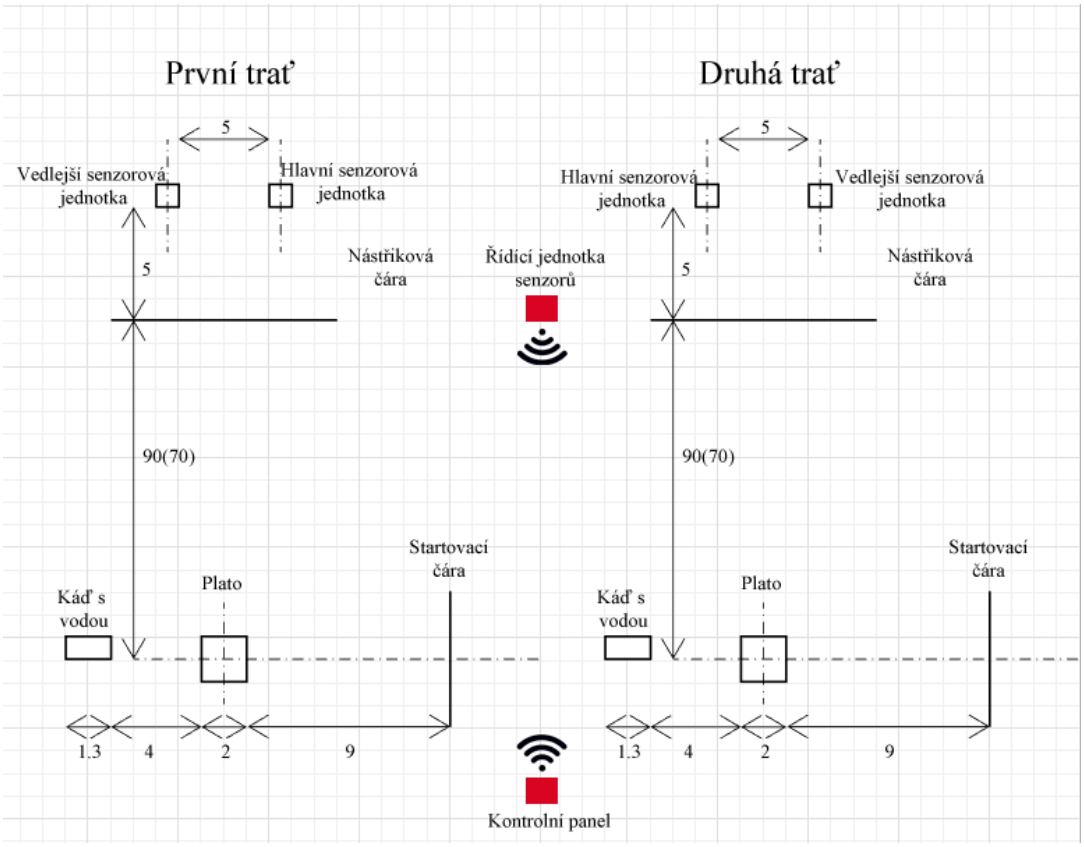
Při požárním sportu se používají dva typy terčů. Oba terče mají ve výšce 160 cm nad zemí umístěný otvor o průměru 5 cm. Prvním typem terčů jsou nástřikové, které jsou používány v postupových soutěžích. Proudaři při tomto typu terčů mají za úkol nastříkat 10 litrů vody do nástřikové nádrže a tím sepnout plovací spínač. Nejlepší proudaři stříkají poslepu a jen poslouchají zvuk terče, jelikož při správném úhlu vody k terči, kde voda přímo proudí do nástřikové nádoby, terč vydává dunivý zvuk. Druhý typ terčů používaný při hasičský soutěží je terč sklopný, kde má proudař své snažení velmi zjednodušené. Stačí jen proudem vody převrátit závažný válec sklopného terče a tím sepnout spínač. Voda ze stříkačky putuje hadicemi přes rozdělovač až k proudnicím.

3.6.1 Složení družstva

- Košař
- Spojař (savičář)
- Strojník
- Béčkař
- Rozdělovač
- Levý proudař
- Pravý proudař

3.6.2 Využití

Časomíra bude vytvořena pro měření dvou nezávislých tratí požárního útoku podle pravidel požárního sportu. Rozmístění jednotlivých prvků pro požární útok znázorňuje obrázek 4.



Obrázek 4 - Teoretický návrh práce

4 Vlastní práce

4.1.1 Použité komponenty

Vývojové desky Arduino v mé práci jsou napájeny z 12 V autobaterií. Pro správné napájení jsem mezi desku a zdroj umístil napěťový regulátor nastavený na 9 V. Výrobce desek udává vstupní napětí v rozmezí 7-12 V, ale veškeré napájecí moduly jsou prodávané 9 V.

- Napěťový regulátor
- Rezistor
- LCD Display
- Relé modul
- Plovací spínače
- Tlačítka
- Deska Arduino UNO
- Deska Arduino mega
- Diodová světla
- Konektory

4.2 Návrh zapojení v programu Fritizing

4.2.1 O programu

Pro grafický návrh ve zjednodušené formě jsem použil software Fritizing. V současné době je na internetu ke stažení verze 0.9.3b. Můj veškerý vývoj byl prováděn ve verzi 0.9.3. Program lze využít nejen k vývoji pro platformu Arduino, ale i pro Picaxe. Vývojové prostředí je rozděleno do 5 základních vývojových záložek (Welcom, BreadBord, Schematic, PCB, Code).

4.2.1.1 BreadBord

Z mého pohledu jedna z nejdůležitějších pracovních záložek v programu. V rámci této záložky program umožňuje grafické vytváření zapojení jednotlivých komponent. V nákresu je jako hlavní objekt připraveno nepájivé pole pro zapojení. V pravé části okna je umístěn panel s výběrem možných komponent. Program všechny dostupné komponenty neobsahuje, ale lze si komponentu vytvořit pomocí nástrojů.

4.2.1.2 Nevýhody

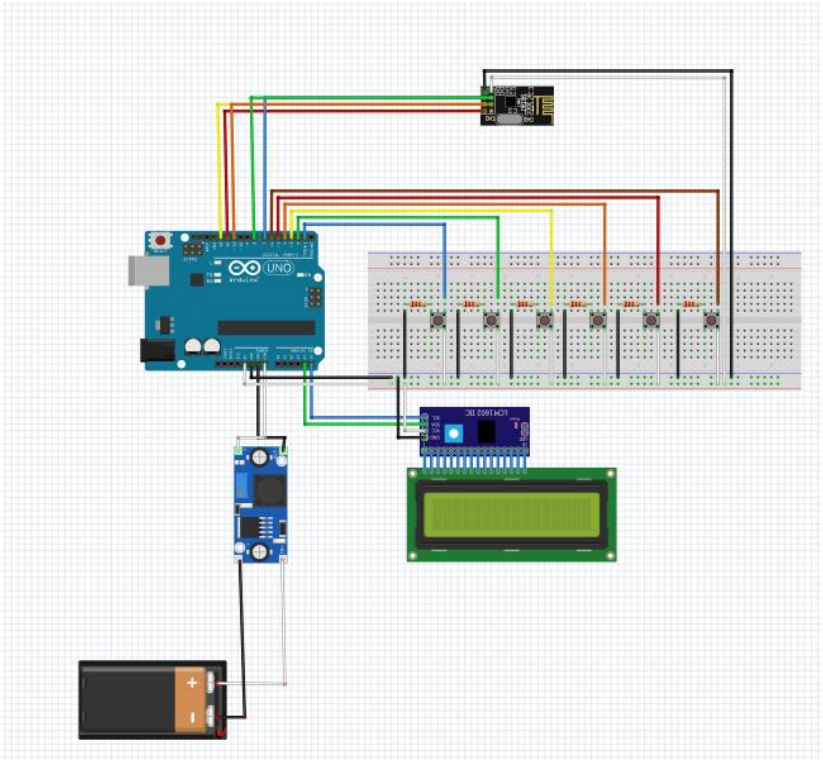
Vývojové prostředí bohužel v sobě nemá implementovaný kompilátor, proto je potřeba mít pro vývoj instalovaný i program Arduino IDE.

4.3 Grafický návrh

Vytvořený grafický návrh je zjednodušené zobrazení celého řešeného případu. Pro zjednodušení jsou senzory nahrazeny jen tlačítky.

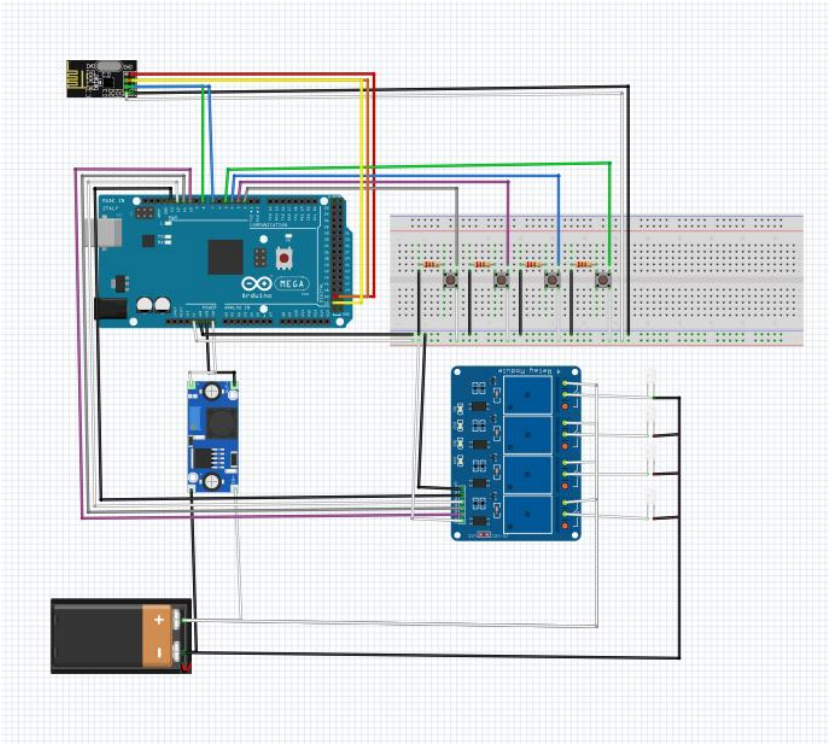
Prvotním krokem práce bylo zjištění technických parametrů jednotlivých elektronických komponent. Dalším krokem bylo vytvoření schématického zobrazení zapojení pomocí programu Fritzing viz obrázek 5 a obrázek 6.

4.3.1 Kontrolní panel



Obrázek 5 - Grafický návrh kontrolního panelu

4.3.2 Senzory



Obrázek 6 - Grafický návrh řídící jednotky senzorů

4.4 Zapojení komponent pomocí nepájivého pole

V rámci složitosti celého produktu jsem nevytvořil celé zapojení pomocí nepájivého pole, ale vždy jen konkrétní segment k odladění chyb a zjištění správné funkčnosti. Následně jsem jednotlivé segmenty umísťoval do PVC boxu a propojoval kabely s konektory.

4.5 Použité knihovny

4.5.1 SPI.h

Knihovna umožňuje komunikovat se zařízeními přes sériové periferní rozhraní a vývojovými deskami Arduino.

SPI je synchronní sériový datový protokol používaný mikrokontroléry pro komunikaci s jedním nebo více periferními zařízeními rychle na krátké vzdálenosti. Může být také použit pro komunikaci mezi dvěma mikrokontroléry.

Při připojení SPI je vždy jedno hlavní zařízení (obvykle mikrokontrolér), které ovládá periferní zařízení.

4.5.2 RF24.h

Knihovna je vytvořena pomocí knihovny SPI.h a slouží k jednoduchému ovládání modulů nRF24L01. Ve své práci jsem použil tyto metody:

4.5.2.1 begin

Funkce pro zahájení komunikace NRF modulu.

4.5.2.2 openWritingPipe

Funkce vytvoří kanál pro zápis modulu nRF24L01. Z kanálu, vytvořeného v zařízení pro zápis, je druhé zařízení schopno odeslaná data číst.

4.5.2.3 openReadingPipe

Funkce pro vytvoření kanálu pro čtení přijatých dat modulem nRF24L01.

4.5.2.4 setPALevel

Funkce pro nastavení režimu rychlosti přenosu (RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH, RF24_PA_MAX). [10]

4.5.2.5 startListening

Funkce pro změnu režimu modulu nRF24L01. Při zavolání této funkce modul začne na vytvořeném kanálu pro čtení poslouchat, zda neobdržel data.

4.5.2.6 stopListening

Funkce pro změnu režimu modulu nRF24L01. Při zavolání této funkce modul přestane na vytvořeném kanálu pro čtení poslouchat, zda neobdržel data.

4.5.2.7 available

Zkontroluje dostupná příchozí data z vysílače.

4.5.2.8 read

Funkce pro čtení obdržených dat z kanálu pro čtení.

4.5.2.9 write

Funkce pro zápis dat, které mají být odeslány do druhého zařízení. Data jsou zapsána do kanálu pro zápis.

4.5.3 Wire.h

Tato knihovna umožňuje komunikaci s I2C.

4.5.4 LiquidCrystal_I2C.h

Knihovna používá funkce knihovny Wire.h, pro svou komunikace s displejem za pomoci obvodu PCF8574, který funguje jako 8 bitový převodník na I2C sběrnici.

Ve své práci jsem použil z knihovny tyto následující metody:

4.5.4.1 begin

Funkce smaže všechny znaky na displeji a nastaví kursor na pozici 0,0.

4.5.4.2 backlight

Funkce zapne podsvícení celého displeje. Opakem funkce je funkce noBacklight, která podsvícení celého displeje vypne.

4.5.4.3 createChar

Funkce pro vytvoření speciálního znaku v paměti. Každý znak je definován maticí o 8 řádcích a 5 sloupcích, tvorba znaku je popsána v kapitole 2.12.3.3. Standardně knihovna umí volat pro vypsání znaky, které má displej uloženy ve své paměti a které jsou obsažené v ASCII tabulce. Znaky s českou diakritikou a speciální znaky musí být vytvořeny jako list bytů.

4.5.4.4 setCursor

Funkce nastavuje list pro výpis znaku na displeji. První parametr určuje řádek, druhý parametr určuje sloupec. Standardně jsou řádky a sloupce indexovány od nuly.

4.5.4.5 print

Funkce pro výpis textu nebo čísla na displej.

4.5.4.6 write

Funkce vypíše na displej znak vytvořený funkcí createChar.

4.6 Použité kódy pro komunikaci

Pro komunikaci mezi zařízeními jsem si vytvořil kódy posílané mezi zařízeními. Pro každý senzor je vytvořen jedinečný identifikátor (kód), který je posílán v obou směrech. Posílané kódy jsou zobrazeny u jednotlivých jednotek v závorkách červeným písmem, viz obrázek 7.

4.6.1 Kódy kontrolního panelu

Kódy jsou odeslány z kontrolního panelu do řídící jednotky senzorů, kde se zpracují.

Kód	Popis	Rozšíření
1	Reset vedlejší senzorové jednotky první tratě	V případě rozšíření časomíry o funkcionalitu měření běhu na sto metrů překážek. Kód bude odpovídat resetu první trati.
2	Reset hlavní senzorové jednotky první tratě	V případě rozšíření časomíry o funkcionalitu měření běhu na sto metrů překážek. Kód bude odpovídat resetu druhé trati.
3	Reset hlavní senzorové jednotky druhé tratě	V případě rozšíření časomíry o funkcionalitu měření běhu na sto metrů překážek. Kód bude odpovídat resetu třetí trati.
4	Reset vedlejší senzorové jednotky druhé tratě	V případě rozšíření časomíry o funkcionalitu měření běhu na sto metrů překážek. Kód bude odpovídat resetu čtvrté trati.

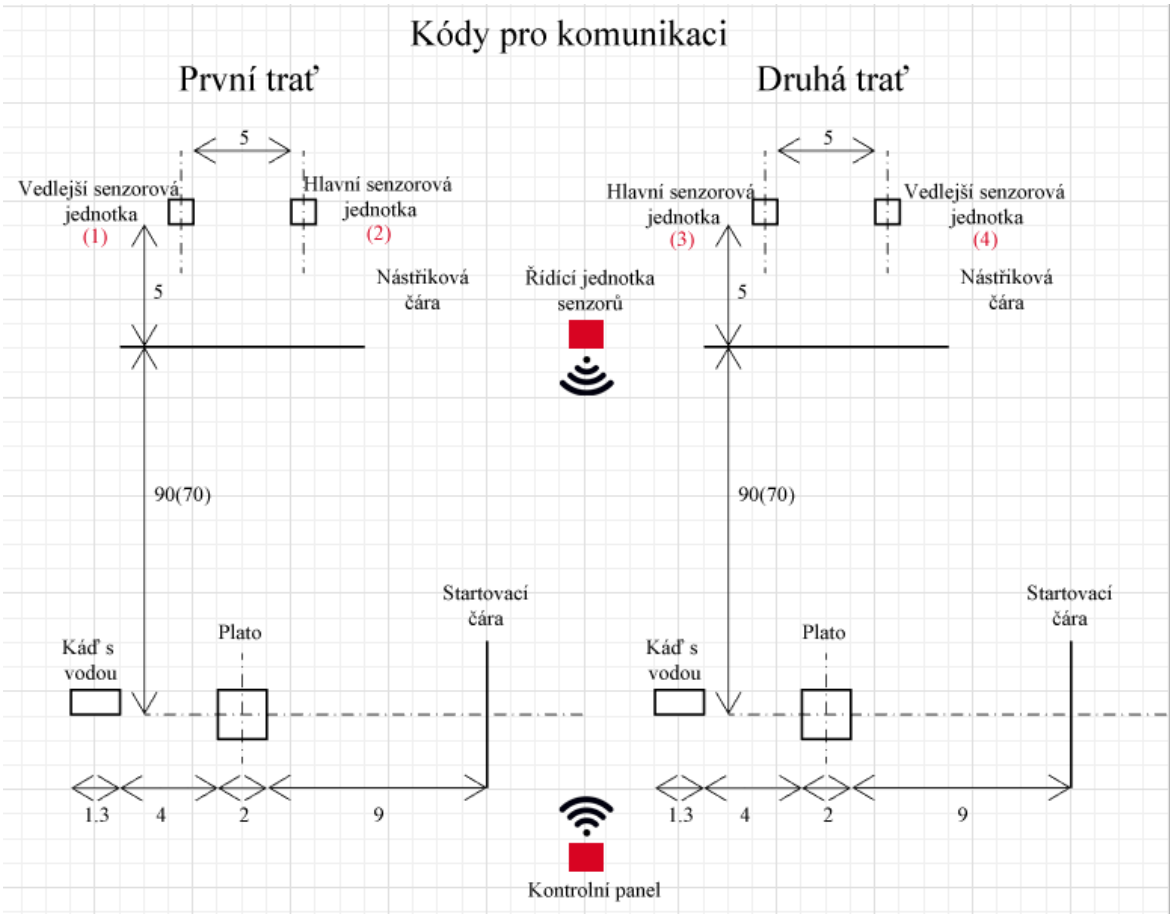
Tabulka 1 - Kódy kontrolního panelu

4.6.2 Kódy řídící jednotky senzorů

Kódy jsou odeslány z řídící jednotky senzorů do kontrolního panelu, kde jsou zpracovány.

Kód	Popis	Rozšíření
1	Senzor vedlejší senzorové jednotky první tratě	V případě rozšíření časomíry o funkcionalitu měření běhu na sto metrů překážek. Kód bude odpovídat první trati.
2	Senzor hlavní senzorové jednotky první tratě	V případě rozšíření časomíry o funkcionalitu měření běhu na sto metrů překážek. Kód bude odpovídat druhé trati.
3	Senzor hlavní senzorové jednotky druhé tratě	V případě rozšíření časomíry o funkcionalitu měření běhu na sto metrů překážek. Kód bude odpovídat třetí trati.
4	Senzor vedlejší senzorové jednotky druhé tratě	V případě rozšíření časomíry o funkcionalitu měření běhu na sto metrů překážek. Kód bude odpovídat čtvrté trati.

Tabulka 2 - Kódy řídící jednotky senzorů



Obrázek 7 - Teoretický návrh práce s kódy pro komunikaci

4.7 Kontrolní panel

Kontrolní panel je hlavní elektronickou jednotkou v mé práci. Pomocí ní se ovládá celá vytvořená struktura ze strany uživatele. Hlavními rysy jsou tlačítka pro její ovládání Start a Reset jednotlivých tratí. Dále je možné připojení startovací pistole. Součástí kontrolního panelu je předpřipravený konektor pro sériový port pro připojení k počítači a následnému rozšíření celého produktu o aplikaci v počítači.

Obrázek 8 - Kontrolní panel

4.7.1 Tvorba modulů pro elektroniku

K zapojení kontrolního panelu jsem použil PVC krabici KP 15 zakoupenou u společnosti GES. Krabice je prodávaná jako celek. Všechny výřezy pro umístění konektorů, tlačítek, antény a displeje, jsem musel sám vytvořit pomocí ruční brusky Dremel a vrtačky se stupňovitým vrtákem.

4.7.2 Struktury

4.7.2.1 Sensor

```
struct sensor
{
   int id;
   unsigned long long stopTime=0;
}
```

4.7.2.1.1 id

Proměnná obsahuje jedinečný identifikátor sensoru. Jeho hodnota se používá jako kód pro komunikaci mezi zařízeními.

4.7.2.1.2 stopTime

Proměnná obsahuje čas, ve kterém kontrolní panel obdržel kód s informací od řídící jednotky sensorů s tím, že aktuální senzor sepnul.

4.7.2.2 ControlPanel

Struktura deklaruje měřenou trať a její ovládací prvky.

```
struct controlPanel
{
   int gStart;
   int bStart;
   int bReset;
   unsigned long long startTime = 0;
   int countSensors = 2;
   sensor sensors[2];
   int cursorLine;
}
```

4.7.2.2.1 gStart

Proměnná určuje, kterému pinu desky Arduino je pro konkrétní trať připojena startovací pistole.

4.7.2.3 bStart

Proměnná určuje, kterému pinu desky Arduino je pro konkrétní trať připojeno startovací tlačítko.

4.7.2.4 bReset

Proměnná určuje, kterému pinu desky Arduino je pro konkrétní trať připojeno resetovací tlačítko.

4.7.2.5 startTime

Proměnná, do které je zapsán čas při zmáčknutí startovacího tlačítka nebo startovací pistole. Proměnná je v kódu taky používána jako ukazatel, zda daná trať byla odstartovaná či ne.

4.7.2.6 countSensors

Proměnná určuje počet senzorů, které daný kontrolní panel sleduje. Proměnná je předpřipravena pro následné rozšíření pro další disciplíny.

4.7.2.7 sensors

List struktur senzorů, které daný kontrolní panel sleduje a ovládá.

4.7.2.8 cursorLine

Proměnná určuje pozici kurzoru řádku, na kterém mají být vypsány informace pro daný kontrolní panel.

4.7.3 Globální proměnné

Pro práci v kódu jsem si definoval řadu globální proměnných, které používám ve funkcích popsaných v další kapitole. Globální proměnou lze použít napříč programem na rozdíl od lokálních proměnných, které lze používat jen v místě jejich deklarace.

4.7.3.1 nRF

Proměnná slouží k inicializaci modulu pro bezdrátovou komunikaci pomocí CE a CS pinu umístěných na deskách Arduino.

```
#define CE 7
#define CS 8
RF24 nRF(CE, CS);
```

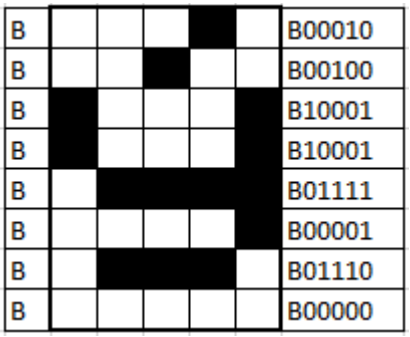
4.7.3.2 addresses

Proměnná je deklarována jako dvourozměrný list bytů z důvodu definování adres kanálů pro komunikaci modulu nRF24L01.

```
const byte addresses[][6] = {"00001", "00002"};
```

4.7.3.3 y

Funkce createChar pracuje s proměnnou y, která je deklarována z důvodu tisknutí dlouhého ypsilon na displej listem bytů.



Obrázek 9 - Bitmapa vytvořeného znaku

```
byte y[8] = {B00010, B00100, B10001, B10001, B01111, B00001, B01110,
B00000};
```

4.7.3.4 lcd

Proměnná představuje objekt displeje o 4 řádcích a 20 sloupcích.

```
LiquidCrystal_I2C lcd(0x27, 20, 4);
```

4.7.3.5 firstControlPanel

Proměnná deklaruje první kontrolní panel se všemi jeho dílčími prvky jako je tlačítko pro ruční start, tlačítko pro reset a konektor pro připojení startovací pistole. Podrobný popis struktury je v kapitole 3.12.2.2. Pro požární útok je nutné, aby každá trať měla vytvořené samostatné ovládání. V mém případě byly tvořeny dvě nezávislé tratě včetně všech řídících prvků a senzorů. V případě rozšíření časomíry pro disciplíny sto metrů překážek a štafetu bude kontrolní panel ovládat všechny senzory.

```
struct controlPanel firstControlPanel;
```

4.7.3.6 secondControlPanel

Proměnná deklaruje druhý kontrolní panel. Panel bude používán jen pro požární útok. V případě rozšíření o další disciplíny požárního sportu bude vždy použita jen proměnná firstControlPanel a její hardware.

```
struct controlPanel secondControlPanel;
```

4.7.3.7 codeDisciplie

Proměnná určuje disciplínu, kterou časomíra měří. Proměnná je vytvořena z důvodu zjednodušení možnosti rozšíření pro další disciplíny.

Kód	Význam
11	Měření požárních útoků
22	Měření disciplín sto metrů překážek nebo štafety

Tabulka 3 - Kódy pro nastavení (případné rozšíření)

```
int codeDisciplie=11;
```

4.7.4 Funkce

4.7.4.1 Setup

Funkce setup je volána jen jednou při startu Arduina. Slouží k nastavení všech potřebných proměnných. Z funkce jsou volány dvě mnou vytvořené funkce writeText a setControlPanels. Funkce jsou vytvořeny z důvodu přehlednosti kódu a strukturování kódu do bloků.

```
void setup()
{
   Serial.begin(9600);
   nRF.begin();
   nRF.openWritingPipe(addresses[1]);
   nRF.openReadingPipe(1, addresses[0]);
   nRF.setPALevel(RF24_PA_LOW);
   nRF.startListening();
   lcd.begin();
   lcd.backlight();
   lcd.createChar(0, y);
   writeText();
   setControlPanels(codeDisciplie);
}
```

4.7.4.2 loop

Po dokončení funkce setup se začne neustále dokola provádět funkce loop, jak její název (loop = smyčka) ostatně napovídá. Příkazy, obsažené v těle této funkce, jsou určeny k provádění veškeré činnosti Arduina. Ve funkci je vytvořená celá řídící logika časomíry rozdělená do tří části. [11]

4.7.4.2.1 Ovládání vstupy od uživatele

Kód ovládá všechny vstupy od uživatele. Jedná se o startovací tlačítka, startovací pistole a resetovací tlačítka.

```
  if ((digitalRead(firstControlPanel.bStart) ||
digitalRead(firstControlPanel.gStart)) && firstControlPanel.startTime ==
0)
    start(&firstControlPanel);
  else  if (digitalRead(firstControlPanel.bReset) &&
firstControlPanel.startTime != 0)
    resetSensors(&firstControlPanel);
  if ((digitalRead(secondControlPanel.bStart) ||
digitalRead(secondControlPanel.gStart)) && secondControlPanel.startTime
== 0)
    start(&secondControlPanel);
  else  if (digitalRead(secondControlPanel.bReset) &&
secondControlPanel.startTime != 0)
    resetSensors(&secondControlPanel);
```

4.7.4.2.2 Výpis informací

Kód obstarává výpis běžícího času na displej. Čas se na displeji rozběhne jen v případě, že došlo uživatelem ke stratu tratě zmáčknutím tlačítka nebo sepnutím startovací pistolí.

```
  if (firstControlPanel.startTime != 0)
    writeTime(&firstControlPanel);
  if (secondControlPanel.startTime != 0)
    writeTime(&secondControlPanel);
```

4.7.4.2.3 Příjem kódů

Kód zajišťuje příjem odeslaných dat z řídící jednotky senzorů. Přijatá data jsou definované kódy v kapitole 2.11.2.

```
  int request;
  if ( nRF.available()) {
    while (nRF.available()) {
      nRF.read( &request, sizeof(request) );
    }
    if (request != 0)
    {
      if (codeDisciplie == 11)
      {
        if (request < 3)
          stopSensor(&firstControlPanel.sensors[request - 1]);
        else
          stopSensor(&secondControlPanel.sensors[request - 2]);
      }
    }
  }
}
```

4.7.4.3 writeText

Slouží k prvotnímu vypsání informací na displej. Je vypsán popis prvotní informace o měřené trati a defaultní zobrazovaný čas. Funkce využívá globální proměnou lcd a její metody k vypsání informací.

```
void writeText()
{
```

```
    lcd.setCursor(0, 0);
    lcd.print("Trat1 Lev");
    lcd.write(byte(0));
    lcd.print(":00:00:00");
…
}
```

4.7.4.4 sendCode

Funkce slouží k odeslání kódu do druhého zařízení. Vstupní parametr funkce je definovaný kód pro komunikaci. Ve funkce pracuji s metodami proměnné nRF.

```
void sendCode(int code)
{
    nRF.stopListening();
    nRF.write( &code, sizeof(code) );
    nRF.startListening();
}
```

4.7.4.5 resetSensors

Funkce má vstupní parametr pointer na strukturu kontrolního panelu. Jednou z proměnných ve struktuře kontrolního panelu je list struktur sensorů, který ovládá kontrolní panel. Funkce, podle kódů popsaných v kapitole 2.11.1, odešle patřičný kód do druhého zařízení. Dále funkce provede reset uložených informací o aktuální trati v proměnných. Čas sepnutí senzoru je nastaven na 0.

```
void resetSensors(controlPanel *c)
{
    (*c).startTime = 0;
    for (int i = 0; i < (*c).countSensors; i++)
    {
        sendCode((*c).sensors[i].id);
        (*c).sensors[i].StopTime = 0;
    }
}
```

4.7.4.6 stopSensor

Funkce má vstupní parametr pointer na sensor deklarovaný ve struktuře kontrolního panelu jako list struktur sensorů. Funkce u předaného sensoru nastaví funkcí millis čas od zapnutí desky Arduino. Nastavený čas je používán při výpočtu zobrazeného času na displeji.

```
void stopSensor(sensor *s)
{
    (*s).StopTime = millis();
}
```

4.7.4.7 start

Funkce má vstupní parametr pointer na kontrolní panel. Funkce při startu tlačítkem nebo startovací pistolí nastaví u kontrolního panelu proměnou startTime, která je použita při výpočtu zobrazovaného času na displeji.

```
void start(controlPanel *c)
{
    (*c).startTime = millis();
}
```

4.7.4.8 setControlPanels

Funkce jako vstupní parametr bere kód nastavení časomíry. Kód 11 říká, že časomíra měří dvě nezávislé tratě požárního útoku. Kód 22 je předpřipraven pro rozšíření na další disciplíny požárního sportu jako je sto metrů překážek nebo štafeta, kde jsou měřeny jednotlivé trati v rámci jednoho kontrolního panelu.

```
void setControlPanels(int code)
```

4.7.4.9 writeTime

Funkce vypisuje na displej běžící čas senzorů aktuální tratě. Ve funkci je vytvořena smyčka nad listem senzorů, které zastavují čas pro daný kontrolní panel. Výpočet času probíhá jako rozdíl času vráceného funkcí millis a času zapsaného stejnou funkcí do

proměnné startTime při startu trati. Pokud daný senzor změnil stav, tak je na displej vždy vypsán čas, kdy došlo ke změně stavu senzoru.

```
void writeTime(controlPanel *c)
{
  unsigned long long now = millis() - (*c).startTime;
  String stringTime = createStringTime( millis() - (*c).startTime);
  for (int i = 0; i < (*c).countSensors; i++)
  {
    if ((*c).sensors[(*c).cursorLine + i].stopTime == 0)
    {
      Serial.println(stringTime);
      lcd.setCursor(11, (*c).cursorLine + i);
      lcd.print(stringTime);
    }
    else
    {
      stringTime = createStringTime((*c).sensors[(*c).cursorLine +
i].stopTime);
      lcd.setCursor(11, (*c).cursorLine + i);
      lcd.print(stringTime);
    }
  }
}
```

4.7.4.10 createStringTime

Funkce vytváří textový řetězec (String), který je použit jako návratová hodnota. Formát textového řetězce je vždy m:ss:ms (m: minuty, s: sekundy, ms: milisekundy). Pro dodržení formátu sekund a milisekund na dvě desetinná místa je volána funkce valueNumber.

```
String createStringTime(unsigned long long timeInLong)
{
  unsigned short  m, s, ms;

  String stringTime = "";
  ms = timeInLong % 1000;
  timeInLong /= 1000;
  s = timeInLong % 60;
  timeInLong /= 60;
  m = timeInLong % 60;
```

```
  timeInLong /= 60;

  if (valueNumber(m))
    stringTime = "0";
  stringTime = String(stringTime + String(m) + ":");
  if (valueNumber(s))
    stringTime = String(stringTime + "0");
  stringTime = String(stringTime + String(s) + ":");
  if (valueNumber(ms))
    stringTime = String(stringTime + "0");
  stringTime = String(stringTime + String(ms));

  return stringTime;
}
```

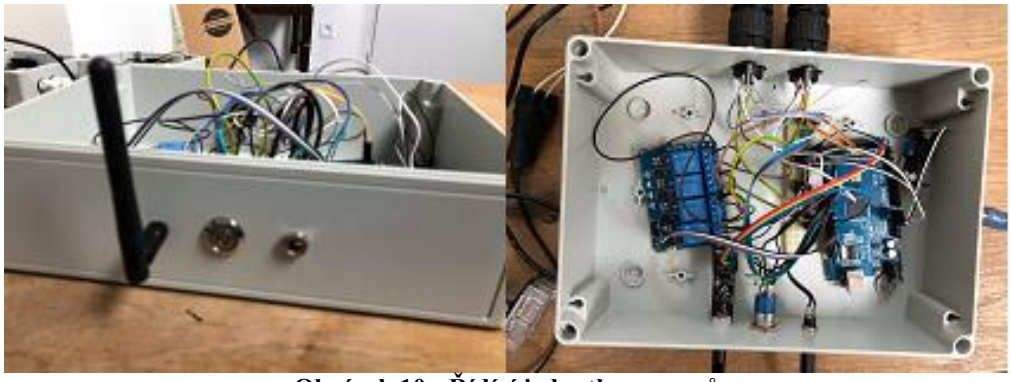
4.7.4.11 valueNumber

Funkce vrací hodnotu true pro čísla, které jsou menší než deset a tím pádem se skládají jen z jednoho znaku a musí být doplněny o úvodní nulu.

```
bool valueNumber(int n)
{
  if (n < 10)
    return true;
  return false;}
```

4.8 Řídící jednotka senzorů

Řídící jednotka je vytvořena tak, aby byla nezávislá na připojených senzorech. Má bakalářská práce se zabývá především vytvořením časomíry pro požární útok pro nástřikové terče. V budoucnu je plánováno vytvořit senzory na fotobuňku nebo tlakové desky. Jednotka je napájena přídavnou autobaterii. Dále se na řídící jednotce nachází tlačítko pro zapnutí a anténa pro přenos dat. Z jednotky vedou dva 7 pinové konektory k propojení hlavních senzorových jednotek popsaných v následující kapitole.



Obrázek 10 - Řídící jednotka senzorů

4.8.1 Struktury

4.8.1.1 Sensor

Struktura sensor je vytvořena k řízení fyzického rozhraní, na rozdíl od struktury senzorů používané v jednotce kontrolního panelu popsaného v kapitole 2.12.2.1.

```
struct sensor
{
   int id;
   int pinSensor;
   int pinLed;
   boolean switchOn = false;
};
```

4.8.1.1.1 Id

Proměnná obsahuje jedinečný identifikátor sensoru. Jeho hodnota se používá jako kód pro komunikaci mezi zařízeními.

4.8.1.1.2 pinSensor

Proměnná obsahuje číslo pinu, ke kterému je připojen senzor.

4.8.1.1.3 pinLed

Proměnná obsahuje číslo pinu, kterým je řízen relé modul pro rozsvícení světel.

4.8.1.1.4 switchOn¨

Proměnná indikuje, zda daný sensor sepnul, aby nedošlo k zacyklení ve funkci loop.

4.8.2 Globální proměnné

Pro práci v kódu jsem si definoval řadu globálních proměnných, které mají stejnou deklaraci a význam, jako v případě globálních proměnných kontrolního panelu popsaných v kapitole 2.12.3. Panel senzorů používá stejné globální proměnné nRF a adresses.

4.8.2.1 countSensors

Proměnná určuje, kolik je možné připojit senzorů.

4.8.2.2 Sensors

Proměnná sensors je deklarována jako list sensorů řídících jednotky senzorů.

```
sensor sensors[countSensors];
```

4.8.3 Funkce

4.8.3.1 setup

Funkce setup je volána jen jednou při startu Arduina. Slouží k nastavení všech potřebných proměnných. Z funkce je volána funkce testLight, která slouží k otestování správného připojení světel.

```
void setup() {
  Serial.begin(9600);
  nRF.begin();
  nRF.openWritingPipe(addresses[0]);
  nRF.openReadingPipe(1, addresses[1]);
  nRF.setPALevel(RF24_PA_LOW);
  nRF.startListening();

  for (int i = 0; i < countSensors; i++)
  {
  sensors[i].id=1+i;
  sensors[i].pinSensor = 2+i;
  sensors[i].pinLed = 9+i;
    pinMode(sensors[i].pinSensor, INPUT);
    digitalWrite(sensors[i].pinLed, LOW);
    pinMode(sensors[i].pinLed, OUTPUT);
  }
  testLight();}
```

4.8.3.2 loop

Kód tvoří smyčku nad všemi nastavenými senzory a kontroluje změnu stavu senzoru.

```
  for (int i = 1; i < countSensors; i++)
  {
    if (!digitalRead(sensors[i].pinSensor) && ! sensors[i].switchOn)
    {
      switchSensor(i);
    }
  }
```

4.8.3.2.1 Příjem kódů

Kód zajišťuje příjem odeslaných kódů z kontrolního panelu. Přijatá data jsou definované kódy v kapitole 2.11.1.

```
int code;
  if ( nRF.available()) {
    while (nRF.available()) {
      nRF.read( &code, sizeof(code) );
    }
    resetSensor(code);
  }
```

4.8.3.3 resetSensor

Funkce slouží k resetu jednotlivých senzorů podle komunikačního kódu.

```
void resetSensor(int i)
{
  sensors[i].pinLed = LOW;
  sensors[i].switchOn = false;
}
```

4.8.3.4 switchSensor¨

Funkce zajištuje při jejím zavolání zapnutí světela a nastavení senzoru do stavu sepnutí. Z funkce je volána funkce sendCode pro odeslání codu s informací sepnutého senzoru.

```
void switchSensor(int i)
```

```
{
  sendCode(sensors[i].id);
  sensors[i].switchOn = HIGH;
  digitalWrite(sensors[i].pinLed, HIGH);
}
```

4.8.3.5 sendCode

Funkce pro zaslání dat. Tato funkce je stejná jako funkce používaná v kontrolním panelu. Její popis je v kapitole 2.12.4.4.

4.8.3.6 testLight

Funkce slouží k otestování, zda jsou všechny jednotky do sebe propojeny správně. Při správném zapojení se dvakrát rozsvítí a zhasnou světla jednotlivých senzorů.

```
void testLight()
{
  switchLight(HIGH);
  switchLight(LOW);
  switchLight(HIGH);
  switchLight(LOW);
}
```

4.8.3.6.1 switchLight

Funkce má vstupní parametr, který určuje, zda mají být světla zapnuta (HIGH) nebo vypnuta (LOW).

```
void switchLight(boolean value )
{
  for (int i = 0; i < countSensors; i++)
  {
    digitalWrite(sensors[i].pinLed, value);
  }
  delay(500);
}
```

4.9 Hlavní senzorová jednotka

Prvotní myšlenkou hlavní senzorové jednotky je propojení řídící jednotky senzorů a jednotek umístěných na terčích s minimalizací potřebné kabeláže. Hlavní senzorová

jednotka obstarává sepnutí spínačů k ní připojených a předání informací z vedlejší senzorové jednotky. Hlavní senzorová jednotka se skládá z 7 pinového konektoru, který slouží k propojení s řídící jednotkou senzorů, z 4 pinového konektoru k připojení vedlejší senzorové jednotky, z tlačítka pro otestování funkčnosti připojení, z vývodu pro připojení plovacího nebo polohového spínače, ze světla pro signalizaci sepnutí spínačů. Pro tlačítko a spínač je použito sériové zapojení.

Pin hlavní jednotka	**Pin vedlejší jednotka**	**Barva vodiče**	**Účel**
1		žlutozelený	Napájení světel hlavní senzorové jednotky
2		černý	Návratový vodič informace pro spínač hlavní senzorové jednotky
3	1	žlutý	Napájení spínačů hlavní i vedlejší senzorové jednotky
4	2	zelený	Propojení do vedlejší senzorové jednotky
5	3	hnědý	Propojení do vedlejší senzorové jednotky
6	4	bílý	Zem napájení světel hlavní i vedlejší senzorové jednotky
7			Není využit

Tabulka 4 - Popis významu konektorů hlavní senzorové jednotky

4.10 Vedlejší senzorová jednotka

Vedlejší senzorová jednotka je zjednodušená hlavní senzorová jednotka. Vedlejší senzorová jednotka se skládá z 4 pinového konektoru k připojení vedlejší senzorové jednotky k hlavní senzorové jednotce, z tlačítka pro otestování funkčnosti připojení, z vývodu pro připojení spínačů, ze světla pro signalizaci sepnutí spínačů. Pro tlačítko a spínač je použito sériové zapojení.

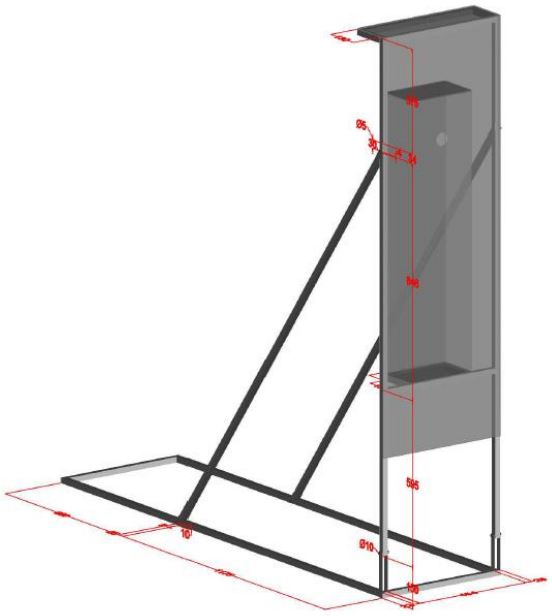
Pin	**Barva vodiče**	**Účel**
1	Červenočerný	Napájení plovacího spínače
2	Černý	Návratový vodič informace pro plovací spínač
3	Žlutý a červený	Napájení světel
4	Bílý a černý	Zem napájení světel

Tabulka 5 - Popis konektorů vedlejší senzorové jednotky

Obrázek 11 - Vedlejší senzorová jednotka

4.11 Vytvoření modelů terčů v AutoCad

Pro svůj návrh modelů nástřikových terčů pro požární sport jsem použil program AutoCad. Při vytváření modelů nástřikových terčů jsem vycházel z pravidel požárního sportu tak, aby byly dodrženy všechny rozměry. Na terči ve výšce 160 cm je kruhový otvor o průměru 5 cm, který slouží k nastříkávání vody do nástřikové nádoby. Předpokládané sestavení terče je plánováno z pěti různých částí. Finančně je nejnáročnější konstrukce, pro kterou byl vybrán nerezový jekl 25 mm x 25 mm x 3 mm. Materiál byl vybrán z důvodu vysoké voděodolnosti a pevnosti. Další částí terče je nástřiková nádoba, pro kterou je pro výrobu navrhnut též nerezový materiál. Pro přední stranu terče byl vybrán plný polykarbonát z důvodu jeho průhlednosti a vysoké odolnosti vůči nárazům. Předposlední částí je PPR potrubí ve tvaru U. Potrubí bude ze strany připevněno na nástřikovou nádobu a do jedné větve bude umístěn plovací spínač a druhá větev bude sloužit k odchodu vzduchu z nástřikové nádoby. Plovací spínač nemůže být přímo umístěn v nástřikové nádobě z důvodu čeření vody při nastříkávání, protože by mohlo dojít k předčasnému sepnutí spínače. Z tohoto důvodu je umístěn mimo nástřikovou nádobu. Poslední částí terče je vytvořená elektronika hlavní senzorové jednotky nebo vedlejší senzorové jednotky.



Obrázek 11 - Model terče

4.12 Akceptační testy

V akceptačních testech jsem vycházel z teoretických východisek uvedených v kapitole 3.5. Jednotlivé prvky jsem rozmístil na sportovní hrací ploše fotbalového oddílu TJ Sokol Sendražice podle obrázku 4 zobrazeného na straně 21. Výsledný produkt se skládá z kontrolního panelu, ke kterému je připojena startovací pistole. Kontrolní panel je napájen autobaterií.

Druhá část produktu se skládá z řídící senzorové jednotky, která je napájena druhou autobaterií. K řídící senzorové jednotce je připojena hlavní senzorová jednotka, do které je připojena vedlejší senzorová jednotka. Veškeré díly jsem zapojil dle popisu a postupoval v testech.

4.12.1 Start tlačítky

Pro obě trati jsem zmáčkl startovací tlačítka v rozdílném časovém intervalu. Stisknutím tlačítek došlo k rozběhnutí času na displeji. Na druhé straně hrací plochy, kde byly umístěny senzorové jednotky, postupně můj asistent zmáčkl testovací tlačítka na jednotlivých senzorových jednotkách, tím došlo k rozsvícení světel a odeslání patřičného kódu do kontrolního panelu, ve kterém daný sensor zastavil čas.

Obrázek 12 - Test startu

4.12.2 Start pistolí

Test probíhal stejným způsobem jako při startu startovacími tlačítky s tím rozdílem, že časomíra byla zpuštěna startovací pistolí. Tento test měl ověřit, jestli se zařízení bude chovat stejným způsobem jako při testu startu tlačítky. Výsledek obou testů byl rovnocenný.

4.12.3 Reset tlačítky po ukončeném útoku

Test probíhal tak, že po dokončení testu startovacími tlačítky bylo zmáčknuto tlačítko pro reset. Došlo k vynulování času na displeji a k odeslání kódu do řídící jednotky senzorů, která vypnula světla pro danou trať. Tím byla potvrzena správnost chování systému.

4.12.4 Reset tlačítky po nedokončeném útoku

Test měl simulovat takzvaný ulitý start. Po startu časomíry nebylo dokončeno sepnutí senzorů a nedošlo tím k zastavení časomíry. Reset byl proveden za běhu časomíry stisknutím červeného tlačítka. Při tomto testu nedošlo k vzájemnému ovlivnění tratí či dalších pokusů.

5 Závěr

Cílem této práce bylo navrhnout a zkonstruovat časomíru pro požární sport, která by byla schopná měřit dvě nezávislé trati. Byl kladen velký důraz na zvýšení spolehlivosti a komfortu obsluhy. Při návrhu bylo počítáno s vytvořením bezdrátové komunikace mezi zařízeními. Dosavadní zkoušky potvrdily správnost návrhů jednotlivých komponent i jejich spojení do funkčního celku. V současné době bylo zařízení předáno Sboru dobrovolných hasičů Sendražice k ladění elektroniky na nástřikových terčích, kdy je třeba odstranit všechny možné problémy a závady, před použitím produktu pro tréninkovou činnost. Díky použití základních desek Arduino a přídavných modulů vyšlo ve výsledném produktu relativně jednoduché zapojení. I v reálných podmínkách se ukázalo, že navržené řešení je funkční. Další úpravy budou probíhat na vytvoření dalších možných měřících senzorů a úpravě softwaru. Uvedení časomíry do provozu je v plánu koncem dubna při zahájení venkovní tréninkové činnosti sboru. Ostré nasazení časomíry pro hasičské soutěže je v plánu v roce 2019 pro všechna kola Polabské hasičské ligy. Tato práce vedla k vytvoření funkčního produktu časomíry a k prohloubení znalostí, jak návrhnout hardware tak i software pro jednotlivé moduly. Do budoucna je v plánu rozšíření, které bylo zmíněno v kapitolách této práce.

6 Seznam použitých zdrojů

1. VODA, Zbyšek. Průvodce světem Arduina. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.
2. Klon Arduino UNO R3 ATmega328P CH340 mini USB. Arduino-shop.cz [online]. [cit. 2018-03-06]. Dostupné z: https://arduino-shop.cz/arduino/1353-klon-arduino-uno-r3-atmega328p-ch340-mini-usb-1466635561.html?gclid=CjwKCAiAqIHTBRAVEiwA6TgJwxp7oix0OcEJzhwmuODRbPooo4O03fnY1d8dMAE_3TWdgjzdhe7mdBoC0HsQAvD_BwE
3. Klon Arduino UNO R3 ATmega328P CH340G USB typ-B kabel. Arduino-shop.cz [online]. [cit. 2018-03-07]. Dostupné z: https://arduino-shop.cz/arduino/1258-klon-arduino-uno-r3-atmega328p-ch340g-usb-typ-b-kabel-1459967190.html
4. Arduino Mega 2560. Czechduino.cz [online]. [cit. 2018-03-07]. Dostupné z: http://www.czechduino.cz/?18,arduino-mega-2560
5. Arduino Mega 2560 + kabel Precizní klon s potiskem. Arduino-shop.cz [online]. [cit. 2018-03-07]. Dostupné z: https://arduino-shop.cz/arduino/946-arduino-atmega2560-kabel-1423598706.html
6. Arduino WiFi modul NRF24L01. Arduino návody [online]. 2016 [cit. 2018-03-07]. Dostupné z: http://navody.arduino-shop.cz/navody-k-produktum/arduino-wifi-modul-nrf24l01.html
7. Arduino Wireless Communication – NRF24L01 Tutorial. *How To Mechatronics* [online]. 2017 [cit. 2018-03-07]. Dostupné z: https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/
8. Wiring (programovací jazyk) [online]. [cit. 2018-03-07]. Dostupné z: https://cs.wikipedia.org/wiki/Wiring_(programovac%C3%AD_jazyk)
9. ZÁKLADNÍ STRUKTURY JAZYKA WIRING. Arduino.cz [online]. 2014 [cit. 2018-03-07]. Dostupné z: https://arduino.cz/zakladni-struktury-jazyka-wiring/
10. RF24. Maniacbug.github.io [online]. [cit. 2018-03-13]. Dostupné z: https://maniacbug.github.io/RF24/classRF24.html
11. ARDUINO – příručka programátora. *Hobbyrobot.cz* [online]. [cit. 2018-03-13]. Dostupné z: http://www.hobbyrobot.cz/wp-content/uploads/ArduinoPriruckaProgramatora.pdf

CD s zdrojovým kódem a návrhem v programu Fritzing.

7 Přílohy

CD s zdrojovým kódem a návrhem v programu Fritzing.