|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Bezpečnostní systém** | | |
| Jakub Feret | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2017/2018 | |

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

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

V Opavě 31. 12. 2017

*podpis autora práce*

###### Poděkování

*Děkuji panu učiteli Ing. Petru Grussmannovi za cenné rady, panu učiteli Mgr. Marcelu Godovskému za pomoc se součástkami a panu učiteli Mgr. Markovi Lučnému za rady a pomoc při nahrání aplikace na školní hosting.*

**Anotace**

Cílem mého projektu bylo vytvořit bezpečnostní systém s možností bezdrátového ovládání přes WiFi. Projekt se skládá z hardwarové a softwarové části. Základ hardwarové části tvoří hlavně vývojová deska NodeMcu. Zařízení pomocí ultrazvukového senzoru snímá vzdálenost k nejbližší překážce, a pokud se tato vzdálenost změní, spustí se program. Ten po uplynutí aktivační doby upozorní uživatele SMS zprávou na možné narušení bezpečnosti. Zařízení je možné aktivovat i deaktivovat zadáním hesla z klávesnice, popřípadě pomocí webové aplikace. Programová část bezpečnostního systému je řešena v jazyce Arduino, což je kombinace jazyků C a C++. Druhou část projektu tvoří webová aplikace vyvinutá v jazyce PHP s využitím Nette Frameworku. Tato aplikace umožňuje po přihlášení vzdáleně ovládat bezpečnostní zařízení. Administrátor může kromě aktivace nebo deaktivace alarmu také upravovat časové prodlevy před aktivací nebo spravovat uživatelské účty.

**Klíčová slova**

Bezpečnost; NodeMcu; ESP8266; Nette; WiFi; bezdrátová komunikace; webová aplikace; Arduino; PHP

OBSAH

[Úvod 6](#_Toc502497209)

[1 Výroba bezpečnostního systému 7](#_Toc502497210)

[2 Princip fungování bezpečnostního systému 8](#_Toc502497211)

[3 Využité technologie 9](#_Toc502497212)

[3.1 Hardware 9](#_Toc502497213)

[3.1.1 Seznam součástek 9](#_Toc502497214)

[3.1.2 NodeMcu 10](#_Toc502497215)

[3.1.3 Ultrazvukový senzor HC-SR04 10](#_Toc502497216)

[3.1.4 LCD displej 1602 11](#_Toc502497217)

[3.1.5 Membránová klávesnice 4x4 11](#_Toc502497218)

[3.1.6 Bzučák a LED 13](#_Toc502497219)

[3.2 Napájení 13](#_Toc502497220)

[3.3 Software 14](#_Toc502497221)

[3.3.1 Jazyk Arduino 14](#_Toc502497222)

[3.3.2 Webová aplikace 14](#_Toc502497223)

[3.3.3 Arduino IDE 1.8.5 14](#_Toc502497224)

[3.3.4 NetBeans 8.2 14](#_Toc502497225)

[3.3.5 XAMPP 5.6.32 15](#_Toc502497226)

[3.3.6 Fritzing 0.9.3b 15](#_Toc502497227)

[4 Způsoby řešení a použité postupy 16](#_Toc502497228)

[4.1 Hardwarové zařízení 16](#_Toc502497229)

[4.1.1 Ověření hesla 16](#_Toc502497230)

[4.1.2 Změna hesla 16](#_Toc502497231)

[4.1.3 Popis fungování bezpečnostního systému 16](#_Toc502497232)

[4.1.4 Připojení k WiFi 17](#_Toc502497233)

[4.1.5 Komunikace se serverem 18](#_Toc502497234)

[Získávání hodnot 18](#_Toc502497235)

[Odesílání hodnot 18](#_Toc502497236)

[4.2 Webová aplikace 19](#_Toc502497237)

[4.2.1 Přihlášení 19](#_Toc502497238)

[4.2.2 Nastavení 20](#_Toc502497239)

[4.2.3 Bezdtrátová komunikace 21](#_Toc502497240)

[5 Výsledky řešení 22](#_Toc502497241)

[5.1 Podoba hardwarového zařízení 22](#_Toc502497242)

[5.2 Podoba webové aplikace 22](#_Toc502497243)

[5.3 Bezdrátová komunikace 24](#_Toc502497244)

[Závěr 25](#_Toc502497245)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 26](#_Toc502497246)

[Seznam příloh 27](#_Toc502497247)

Úvod

Pro vytvoření bezpečnostního systému s možností ovládat jej vzdáleně pomocí webové aplikace jsem se rozhodl, protože mi problematika Internetu věcí, tedy propojení hardwarových zařízení s Internetem, připadala velmi zajímavá a sám jsem si chtěl něco takového vyzkoušet. Na podobném principu jako můj bezpečnostní systém totiž může fungovat například ovládání celých chytrých domácností, kde je možné třeba z mobilního telefonu zapnout světla, nebo postavit na čaj. Bezpečnostní systém upozorní uživatele SMS zprávou, světelným a zvukovým signálem na pohyb před senzorem. Zařízení lze využít ke kontrole nemovitostí či osobních předmětů.

Hlavním cílem projektu však bylo propojit zařízení se zabezpečenou webovou aplikací a umožnit tím jeho vzdálené ovládání. Uživatel se k aplikaci připojí pomocí uživatelského jména a hesla. Běžní uživatelé pak mohou systém bezpečně aktivovat nebo deaktivovat. Administrátor má navíc možnost změnit v aplikaci dobu do aktivace a do spuštění alarmu. Zároveň může vytvářet nové uživatelské účty, popřípadě upravovat nebo mazat současné.

Protože se jednalo o můj první projekt v této oblasti, zvolil jsem pro jeho realizace známé a dobře zdokumentované technologie. S některými z nich, jako například s Frameworkem Nette, jsem se již setkal ve škole. Hardwarovou část tvoří hlavně vývojová deska NodeMcu s WiFi modulem ESP8266 a k němu připojené další komponenty, o kterých se ještě zmíním v dalších částech dokumentace. Bezpečnostní zařízení je naprogramováno v jazyce Arduino, jedná se o kombinaci jazyků C a C++. Webovou aplikaci řeším pomocí PHP Frameworku Nette. Komunikace mezi aplikací a bezpečnostním zařízením probíhá přes WiFi na principu REST API.

V této dokumentaci podrobně popisuji výrobu bezpečnostního systému a princip jeho fungování. Na začátku se zmiňuji o problémech, kterými jsem se zabýval při výrobě bezpečnostního zařízení, pokračuji popisem technologií nezbytných k jeho výrobě i k jeho současné funkčnosti a rozebírám, jak funguje nejen samotné bezpečnostní zařízení, ale i webová aplikace potřebná k jeho vzdálenému ovládání. V další části vysvětluji, na jakých principech funguje jejich vzájemná komunikace, a popisuji jednotlivé úkony obou částí systému. V závěru se zabývám současnou podobou bezpečnostního systému a hodnotím odvedenou práci.

# Výroba bezpečnostního systému

První část mého projektu představovalo sestavení samotného zařízení. Protože se jednalo o mou první zkušenost se sestavováním integrovaných obvodů, výběru a nákupu součástek předcházel zdlouhavý výzkum dané problematiky. Nejdříve jsem se rozhodl jako základ mého zařízení použít vývojovou desku Arduino UNO, založenou na mikrokontroléru ATmega328 od firmy Atmel. Pro komunikaci s webovou aplikací jsem vybral WiFi modul ESP8266-01. Toto spojení se však časem ukázalo jako nepraktické a neefektivní, proto jsem se rozhodl nahradit jak Arduino UNO, tak ESP8266-01 vývojovou deskou NodeMcu, která již obsahuje WiFi modul ESP8266.

Vzhledem k tomu, že jsem s nakoupenými součástkami nikdy dříve nepracoval, rozhodl jsem se postupovat postupně. Zapojoval jsem jednu součástku po druhé do nepájivého pole a hned jsem zkoušel její funkčnost pomocí jednoduchých příkladů v Arduino IDE. Finální verzi projektu jsem nejdříve navrhnul v programu Fritzing, kde jsem si odzkoušel rozvržení součástek na nepájivém poli.

Po zapojení a odzkoušení všech součástek jsem přešel k programování samotného bezpečnostního systému. Pro tento účel jsem zvolil programovací jazyk přímo pro Arduino, což je kombinace jazyků C a C++. Když byla první část projektu plně funkční, čekala mě ta pravá výzva – spojit zařízení přes WiFi s webovou aplikací a umožnit tak jeho ovládání na dálku, což byl, jak jsem se již zmiňoval, hlavní cíl mého projektu.

Webovou aplikaci jsem se rozhodl vyřešit pomocí PHP Framework Nette, se kterým jsem již měl zkušenost ze školy, spolu s databázovým systémem MySQL. Protože se jedná o aplikaci určenou pro ovládání bezpečnostního systému, snažil jsem se ji co nejvíce zabezpečit. Každý uživatel aplikace se musí nejprve přihlásit svým uživatelským jménem a heslem. Administrátor pak může vytvářet, upravovat či mazat uživatele a má také plnou kontrolu nad samotným bezpečnostním zařízením – může měnit intervaly před aktivací a před spuštěním alarmu a samozřejmě také zařízení na dálku zapnout nebo vypnout. Ostatní uživatelé pak mají možnost změnit pouze vlastní přihlašovací údaje a také mohou zapnout nebo vypnout alarm.

# Princip fungování bezpečnostního systému

Propojení ESP8266 a webové aplikace jsem nejdříve řešil tak, že jsem v aplikaci měnil hodnoty v MySQL databázi a k nim jsem pak přistupoval z ESP8266 pomocí knihovny MySQL connector. To se ukázalo jako nevyhovující, jednak kvůli bezpečnosti a také kvůli tomu, že by toto řešení spolehlivě fungovalo pouze na lokální úrovni.

Rozhodl jsem se raději vytvořit jednoduché REST API. V aplikaci vypisuji hodnoty do JSON výstupu a k tomu přistupuji z ESP8266 pomocí GET požadavku. Přijatá data pak pomocí knihovny ArduinoJson dekóduji a ukládám do proměnných, se kterými pak můžu dále pracovat. Díky tomu jsem schopen alarm na dálku zapnout, popřípadě změnit jeho nastavení. Bylo však také potřeba zajistit, aby se server dozvěděl o případném manuálním vypnutí alarmu z klávesnice a změnil pak příslušnou hodnotu v databázi. Tento problém jsem vyřešil pomocí požadavku POST. Opět vytvářím JSON výstup, tentokrát ho však posílám do Nette webové aplikace, kde ho dekóduji a ukládám do databáze. Ani tento postup však není příliš vhodný, zejména kvůli potřebě posílat v pravidelných intervalech požadavky na server (například každých deset vteřin, nejlépe ještě častěji) a zjišťovat, zda se hodnoty nezměnily. To vede k velkému odběru elektrické energie, což mi znemožňuje napájet zařízení z baterie, jak jsem původně plánoval.

To by bylo možné, pokud bych zařízení propojil se serverem bez nutnosti posílat na něj každou chvíli požadavky. Mohl bych tak uvést zařízení do režimu spánku, dokud by nedostalo impuls k probuzení, například změněním určité hodnoty. Tím bych výrazně snížil spotřebu elektrické energie. Chtěl jsem proto opět změnit princip komunikace mezi webovou aplikací a ESP8266, konkrétně jsem měl v plánu použít komunikační protokol MQTT, který umožňuje propojení se serverem bez nutnosti neustálého posílání požadavků. Protože však Nette MQTT nijak nepodporuje, bylo by nutné kompletně přepsat webovou aplikaci, nejlépe do jiného jazyka (například Pythonu), což by se mi bohužel nepodařilo stihnout před termínem odevzdání projektu, a tak jsem zařízení ponechal v současném funkčním, avšak ne zcela vyhovujícím stavu.

# Využité technologie

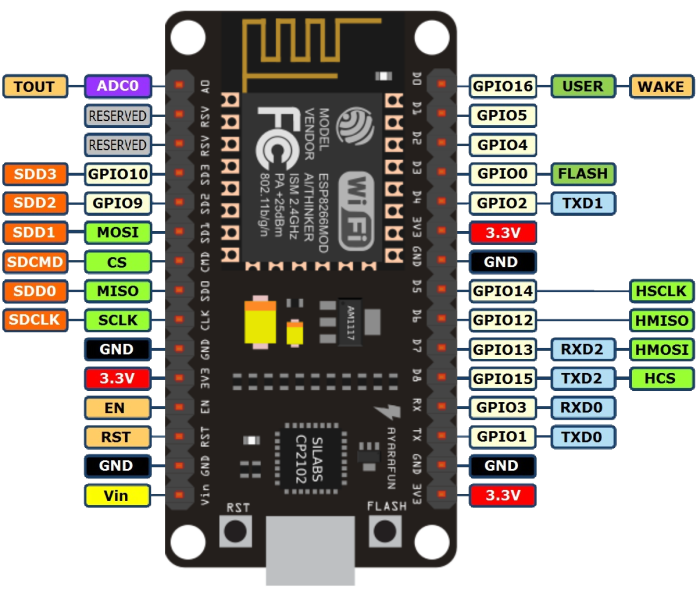
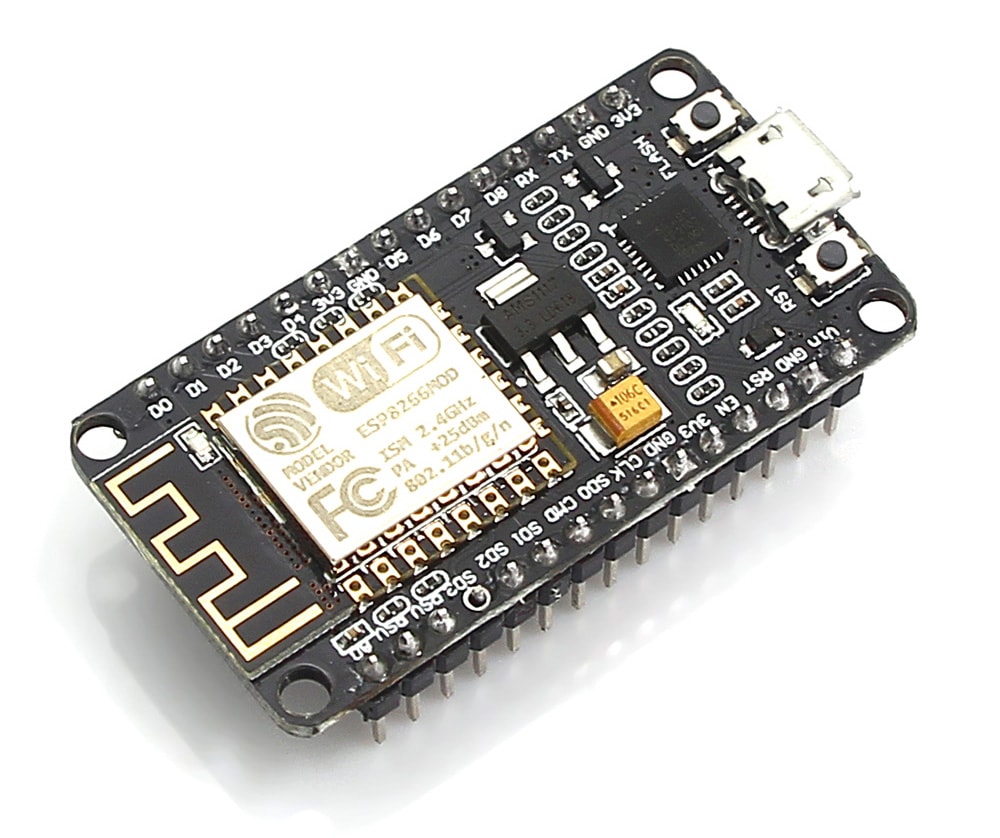
## Hardware

### Seznam součástek

* Vývojová deska NodeMcu,
* ultrazvukový senzor HC-SR04,
* LCD displej 1602,
* LCD I2C sériové rozhraní
* membránová klávesnice 4x4,
* I2C sběrnice PCF8574P,
* 5 V bzučák,
* červená LED,
* rezistor 330 Ω,
* obousměrný osmi kanálový převodník logických úrovní,
* napájecí modul nepájivého pole,
* 2 nepájivé pole se 400 kontakty.

### NodeMcu

Základ projektu tvoří vývojová deska NodeMcu (obrázek č. 1). Firmware obstarává WiFi modul ESP8266 od společnosti Espressif Systems, hardware je založen na modulu ESP-12E, ovšem s šesti GPIO piny navíc. NodeMcu tedy obsahuje 12 digitálních GPIO pinů a jeden analogový pin (obrázek č. 2). Pro program je k dispozici paměť o velikosti 128 kB. Na rozdíl od většiny ostatních komponentů, NodeMcu pracuje s napětím 3,3 V.

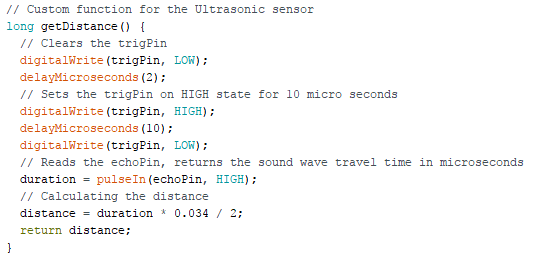


*Obrázek č. 2 Piny NodeMcu*

*Obrázek č. 1 NodeMcu*

### Ultrazvukový senzor HC-SR04

Bezpečnostní systém funguje na principu ultrazvukových vln, které měří vzdálenost k nejbližší překážce. To zajišťuje ultrazvukový senzor HC-SR04, který je schopný pomocí ultrazvukových vln změřit vzdálenost od dvou do 400 cm s přesností plus mínus tři milimetry. Úhel měření je 15 stupňů. Senzor po aktivaci automaticky pošle osm 40 kHz vln a čeká na odražený signál. Pokud odražený signál dorazí zpět, senzor zaznamená čas od vyslání vlny k jejímu návratu. Vzdálenost se pak vypočítá jako doba od vyslání vlny k jejímu návratu \* rychlost zvuku / 2 (obrázek č. 3). Při vymýšlení principu fungování bezpečnostního zařízení jsem se rozhodoval mezi ultrazvukovým senzorem a pohybovým čidlem. Pro ultrazvukový senzor jsem se rozhodl hlavně proto, že pohybové čidlo po spuštění snímá celou místnost a zaznamená jakýkoliv pohyb v dané místnosti. Nehodí se proto do domácností s volně se pohybujícími zvířaty. Ultrazvukový senzor naproti tomu snímá pouze určitou oblast přímo před sebou, takže pokud se nainstaluje na správné místo, je senzor schopen zaznamenat například otevření dveří, avšak zvíře procházející kolem dveří už nezaznamená.



*Obrázek č. 3 Funkce pro získání vzdálenosti z ultrazvukového senzoru*

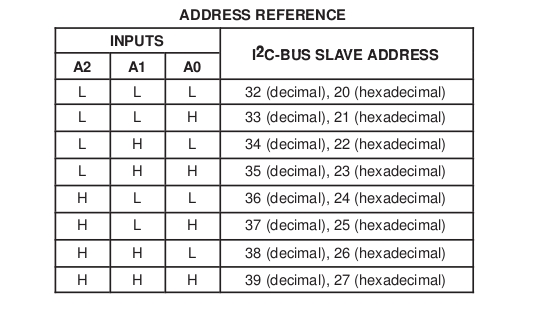
### LCD displej 1602

Jako výstupní zařízení jsem použil LCD displej 1602, který dokáže zobrazit 32 znaků na dvou řádcích (16 x 2). Kvůli omezenému počtu pinů na NodeMcu jsem pro připojení využil I2C sériové rozhraní, které mi umožnilo připojit displej pouze dvěma vodiči. Funkčnost displeje obstarává knihovna LiquidCrystal\_I2C.

### Membránová klávesnice 4x4

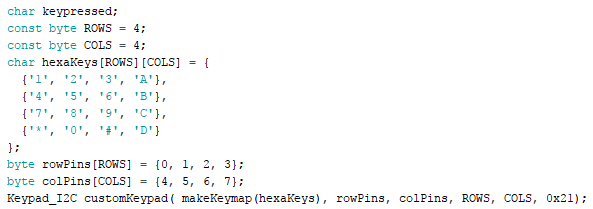
Uživatelský vstup řeším pomocí membránové klávesnice o šestnácti klávesách (4 řádky a 4 sloupce). Pro ovládání klávesnice je potřeba osm vodičů, pro které jsem však neměl k dispozici dostatek pinů. Nejdříve jsem se proto rozhodl zredukovat osm digitálních pinů na jeden analogový, a to pomocí pole rezistorů. Využil jsem toho, že funkce analogRead() dokáže přiřadit vstupní napětí od 0 do 3,3 V číselným hodnotám od 0 do 1024. Díky rezistorovému poli bude po zmáčknutí každé klávesy tato hodnota vždy jiná. Napsal jsem si tedy jednoduchou knihovnu v jazyce C++, která ke každé hodnotě, kterou získám funkcí analogRead() z analogového pinu, přiřadí danou klávesu. Bohužel po přechodu z Arduino Uno na NodeMcu toto řešení vyústilo ve spoustu problémů, zejména kvůli nedokonalému vyvedení analogového pinu NodeMcu, na kterém jsem naměřil četné výkyvy signálu, které způsobovaly samovolné „mačkání“ kláves. Rozhodl jsem se proto vyřešit zapojení klávesnice pomocí I2C sběrnice PCF8574P, podobně jako u LCD displeje.

Protože je díky I2C sběrnici klávesnice připojená ke stejným pinům jako LCD displej, zbývá mi dostatek pinů pro ostatní komponenty. Aby to však bylo možné, nesmí mít LCD displej a klávesnice stejnou I2C adresu. Pomocí jednoduchého Arduino programu jsem zjistil, že displej používá adresu 0x27, stačilo tak nastavit pro klávesnici jakoukoliv jinou adresu, s čímž mi pomohla tabulka na obrázku č. 4. Zvolil jsem adresu 0x21, takže bylo nutné u I2C sběrnice PCF8574P uzemnit piny A2 a A1 a pin A0 připojit k napětí 3,3 V.



*Obrázek č. 4 Tabulka pro určení I2C adresy*

Funkčnost klávesnice obstarává knihovna Keypad\_I2C. Hodnoty kláves se zapisují do dvourozměrného pole (obrázek č. 5).



*Obrázek č. 5 Deklarace proměnných pro klávesnici*

Dále je potřeba klávesnici inicializovat ve funkci setup() (obrázek č. 6).



*Obrázek č. 6 Inicializace klávesnice*

Vždy, když je potřeba v nějaké funkci zjišťovat, zda byla stisknuta nějaká klávesa, musí se z knihovny Keypad\_I2C zavolat funkce getKey() (obrázek č. 7).



*Obrázek č. 7 Zapsání stisknuté klávesy do proměnné keypressed*

### Bzučák a LED

Součástí zařízení je také 5 V bzučák o frekvenci 2,3 kHz, který slouží k upozornění na zmáčknutí tlačítka a také jako bezpečnostní prvek, který má hlasitým zvukem vylekat případného narušitele. Jako další bezpečnostní prvek slouží červená LED, která po spuštění alarmu začne jasně svítit.

## Napájení

Problém nastal při řešení napájení jednotlivých komponentů, zejména proto, že vývojová deska NodeMcu a WiFi modul ESP8266 běží na jiném napětí, než ostatní součástky. Konkrétně NodeMcu vyžaduje napětí 3,3 V, zatímco ostatní komponenty 5 V. Nebylo proto možné připojit komponenty přímo k NodeMcu, protože by se jim nedostávalo dostatek napětí. Stejně tak jsem nemohl NodeMcu napájet 5 V, které by ho pravděpodobně usmažily. Tento problém jsem se rozhodl vyřešit pomocí obousměrného osmi kanálového převodníku logických úrovní, který je schopen převádět 5 V na 3,3 V a naopak.

Samotné napájení je řešeno pomocí napájecího modulu nepájivého pole, který dokáže připojit napájení do dvou napěťových větví. U každé z nich lze jumperem zvolit, jaké napětí do ní bude přivedeno (buď 3,3 V nebo 5 V). Modul vyžaduje vstupní napětí od 6,5 V do 12 V, které lze přivést kabelem z elektrické sítě nebo pomocí USB z počítače. Maximální výstupní proud je 700 mA, což je dostačující pro napájení většiny součástek.

Všechny komponenty jsem připojil na dvě nepájivá pole, každé z nich disponuje 400 kontakty.

## Software

### Jazyk Arduino

Program sloužící k ovládání hardwarových součástek jsem napsal v jazyce Arduino, který je, až na drobné úpravy, velmi podobný jazyku C nebo C++. Jazyk Arduino byl přímo vytvořen k programování integrovaných obvodů.

### Webová aplikace

Webovou aplikaci jsem řešil v jazyce PHP, konkrétně jsem využil PHP Frameworku Nette ve verzi 2.4. Při tvorbě uživatelského rozhraní jsem použil framework Bootstrap v3.3.4, který mi usnadnil práci s HTML5 a CSS, hlavně v oblasti responzibility. Vzhled uživatelského rozhraní jsem řešil pomocí kaskádových stylů CSS3 a také pomocí javascriptové knihovny jQuery. K provozu aplikace využívám databázový systém MySQL, ve kterém ukládám jak uživatelské údaje, tak data nutná pro ovládání samotného bezpečnostního zařízení. Celá aplikace běží na Apache HTTP Serveru.

### Arduino IDE 1.8.5

Jako vývojové prostředí jsem pro Arduino program zvolil Arduino IDE 1.8.5, které je přímo uzpůsobeno pro nahrávání programů do vývojových desek jako Arduino nebo NodeMcu.

### NetBeans 8.2

Pro vývoj PHP webové aplikace jsem zvolil integrované vývojové prostředí NetBeans 8.2, kvůli jeho přehlednému uživatelskému rozhraní a jeho podpoře množství programovacích jazyků.

### XAMPP 5.6.32

Při vývoji webové aplikace byl nezbytný multiplatformní softwarový balíček XAMPP, díky kterému je možné vytvořit lokální webový server. Konkrétně jsem použil verzi 5.6.32. Balíček obsahuje serverovou aplikaci Apache, databázi MariaDB a skriptovací jazyk PHP. XAMPP jsem zvolil kromě jeho jednoduchosti a intuitivního ovládání také proto, že většina webových serverů využívá stejné komponenty jako XAMPP, takže je přechod z lokálního testovacího serveru na živý server velmi jednoduchý.

### Fritzing 0.9.3b

Pro návrh bezpečnostního zařízení jsem použil open-source program Fritzing ve verzi 0.9.3b. Ten umožňuje graficky sestavit zapojení jednotlivých součástek na nepájivém poli, což pomáhá k lepší představě o výsledném rozložení. Program dále usnadňuje tvorbu schématu zařízení, ze kterého lze pak přímo navrhnout desku plošných spojů, jejíž výrobu je možné přímo v programu objednat.

# Způsoby řešení a použité postupy

## Hardwarové zařízení

### Ověření hesla

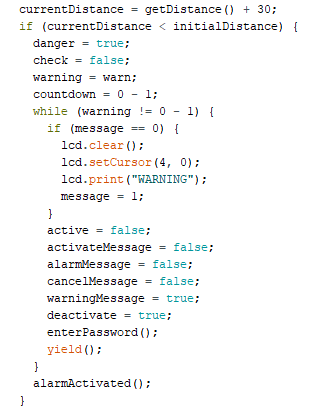
Stěžejní funkcí programu je funkce enterPassword(). Při tvorbě zařízení pro mě byla bezpečnost systému velmi důležitá, proto musí být naprostá většina požadavků ověřená zadáním hesla. Na začátku se zavolá funkce eepromRead(), která zpřístupní současné platné heslo uložené v EEPROM paměti pro porovnání se zadaným heslem. Hlavním úkolem funkce enterPassword() je číst hodnotu stisknuté klávesy, uložit ji do proměnné a po stisknutí klávesy \* tuto proměnnou porovnat s platným heslem uloženým v EEPROM paměti. Následující akce se odvíjejí od současné situace alarmu – pokud je alarm vypnutý a je zadáno platné heslo, spustí se funkce pro aktivaci alarmu activateAlarm(). Pokud je alarm již zapnutý a je zadáno platné heslo, zavolá se naopak funkce pro deaktivaci alarmu deactivateAlarm(). Uživatel má také možnost zrušit probíhající odpočet do aktivace alarmu a i v tomto případě funkce enterPassword() zajistí po zadání hesla deaktivaci alarmu. Při pokusu o deaktivaci alarmu před vypršením času do aktivace má uživatel tři pokusy na zadání platného hesla. Pokud se mu to nepodaří, je okamžitě zavolána funkce alarmActivated() a alarm se spustí.

### Změna hesla

Uživatel má také možnost současné heslo změnit, stará se o to funkce changePassword(). Funkci lze vyvolat, pokud alarm není aktivován, stisknutím klávesy B. Funkce vyžaduje zadání současného a nového hesla. Kvůli zamezení případným překlepům je nutné nové heslo zadat dvakrát. Pokud jsou všechny podmínky splněny, zavolá se funkce eepromWrite(), která nové heslo uloží do EEPROM paměti.

### Popis fungování bezpečnostního systému

Po zapnutí NodeMcu se automaticky spustí funkce loop(). Ta vypíše na displej zprávu a zavolá funkci enterPassword(). Po zadání správného hesla se zavolá funkce activateAlarm(), která odpočítává čas do aktivace systému. Tento čas slouží k opuštění hlídaného prostoru. Uživatel má zároveň možnost odpočet zrušit, a to stisknutím klávesy C a zadáním správného hesla, čímž se opět zavolá funkce loop(). Po uplynutí času se spustí funkce alarmCheck() (obrázek č. 8), která pomocí funkce getDistance() aktivuje ultra-zvukový sensor a začne měřit vzdálenost k nejbližší překážce.



*Obrázek č. 8 Ukázka kódu funkce alarmCheck()*

Pokud se tato vzdálenost změní, aktivuje se odpočet do spuštění alarmu, pomocí funkce ledBlink() se rozbliká LED a zavolá se funkce enterPassword(), což dává uživateli možnost zadat platné heslo a tím odpočet zrušit. Pokud to neudělá, zavolá se po uplynutí odpočtu funkce alarmActivated(). Ta spustí naplno bzučák a jasně rozsvítí červenou LED. Nakonec zavolá funkci enterPassword(), ve které lze zadáním správného hesla alarm deaktivovat.

### Připojení k WiFi

Pro snadnější připojení k WiFi jsem využil knihovnu WiFiManager pro ESP8266. Díky ní se zařízení po zapnutí automaticky pokusí o připojení k již nakonfigurované WiFi síti. Pokud se to nepodaří, nebo pokud ještě žádná síť nakonfigurována nebyla, automaticky vytvoří web server, na kterém uživatel zadá heslo ke své WiFi a ESP8266 se k ní pak bude automaticky připojovat.

### Komunikace se serverem

Pro vzdálené ovládání bezpečnostního zařízení je nutná jeho komunikace se serverem. Tu zajišťuji pomocí GET a POST požadavků z ESP8266 na server.

#### Získávání hodnot

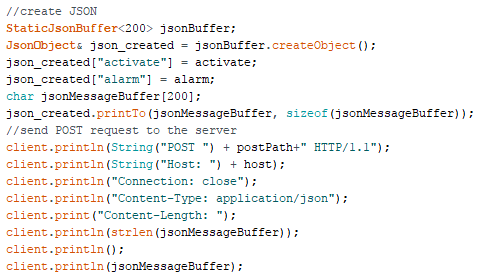
Získávání dat ze serveru řeším pomocí GET požadavku z ESP8266. Funkce getValues() se připojí k serveru a získá HTML stránku, která obsahuje JSON s daty. Následně se zbaví hlavičky stránky a pomocí knihovny ArduinoJson dekóduje JSON data. Ta se pak uloží do proměnných, se kterými lze dále pracovat (obrázek č. 9).



*Obrázek č. 9 Ukázka kódu z funkce pro získání JSON dat ze serveru*

#### Odesílání hodnot

Odesílání dat na server mi zajišťuje funkce postValues(). Ta zakóduje data do JSON výstupu, připojí se na server a pomocí POST požadavku tato data odešle (obrázek č. 10). Zbytek obstará samotná webová aplikace.



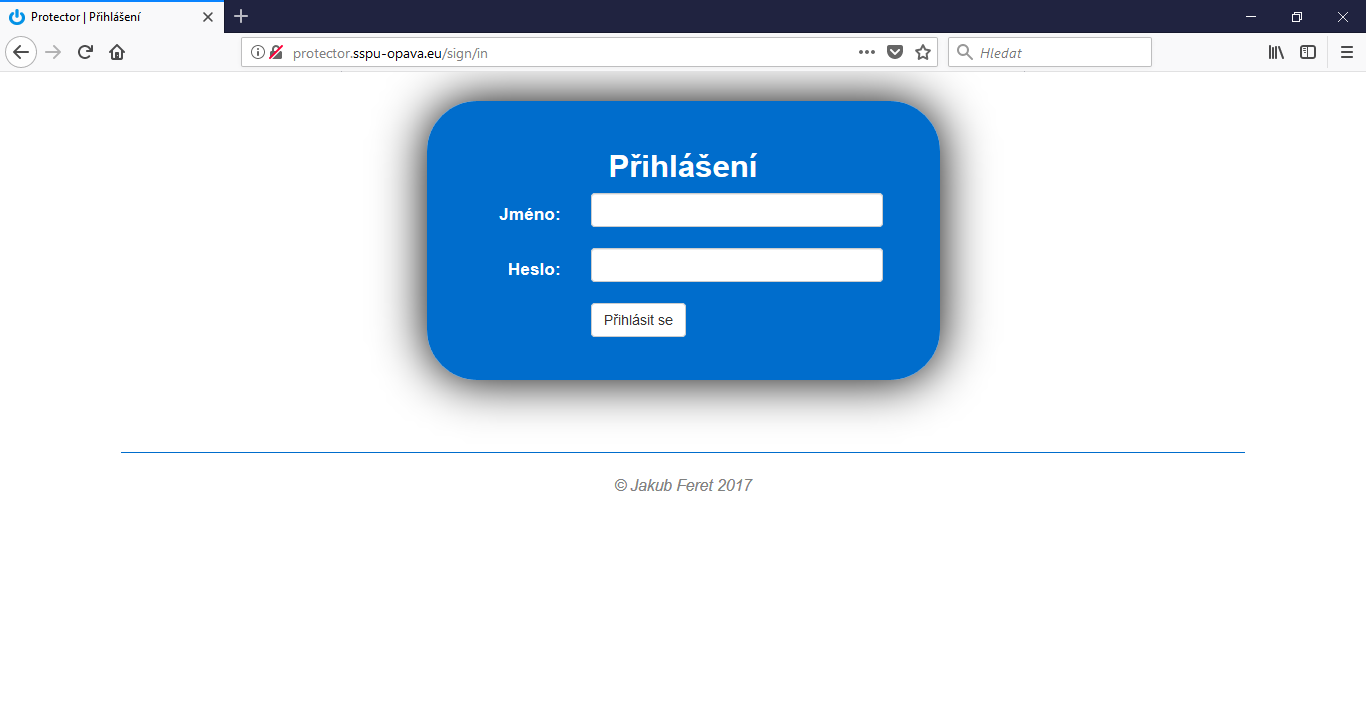
*Obrázek č. 10 Ukázka kódu z funkce pro odesílání JSON dat na server*

## Webová aplikace

Při tvorbě webové aplikace mi značně usnadnil práci Framework Nette, který již obsahuje četná bezpečnostní opatření. Například pro přihlašování uživatelů tedy využívám funkce, které již Nette v základu má. Aplikace je propojená s databázovým systémem MySQL, do kterého prostřednictvím Nette ukládám data, ke kterým pak můžu přistupovat.

### Přihlášení

Do aplikace je nejprve vždy nutné se přihlásit pomocí uživatelského jména a hesla (obrázek č. 11).

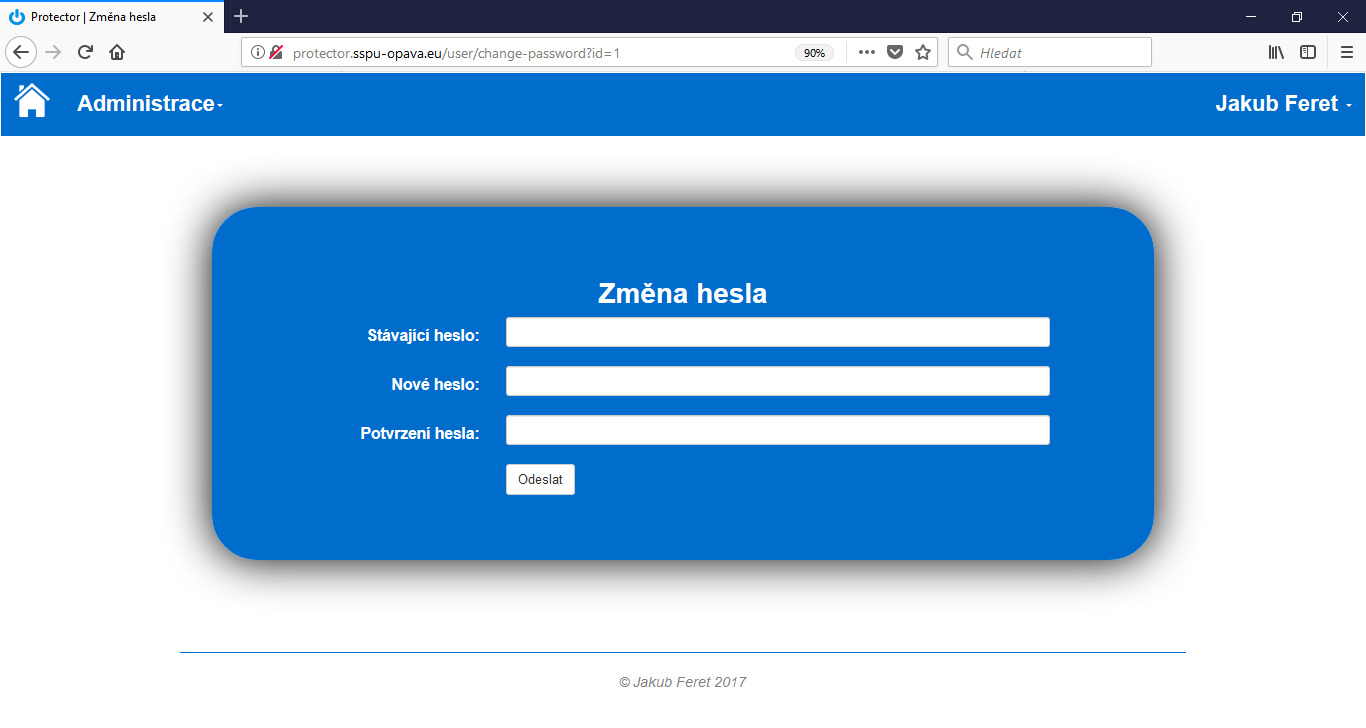


*Obrázek č. 11 Přihlašovací okno do webové aplikace*

Po přihlášení se pomocí údajů v databázi rozhodne, zda se přihlásil běžný uživatel nebo administrátor. Na tom závisí, jaká práva bude daný uživatel v aplikaci mít.

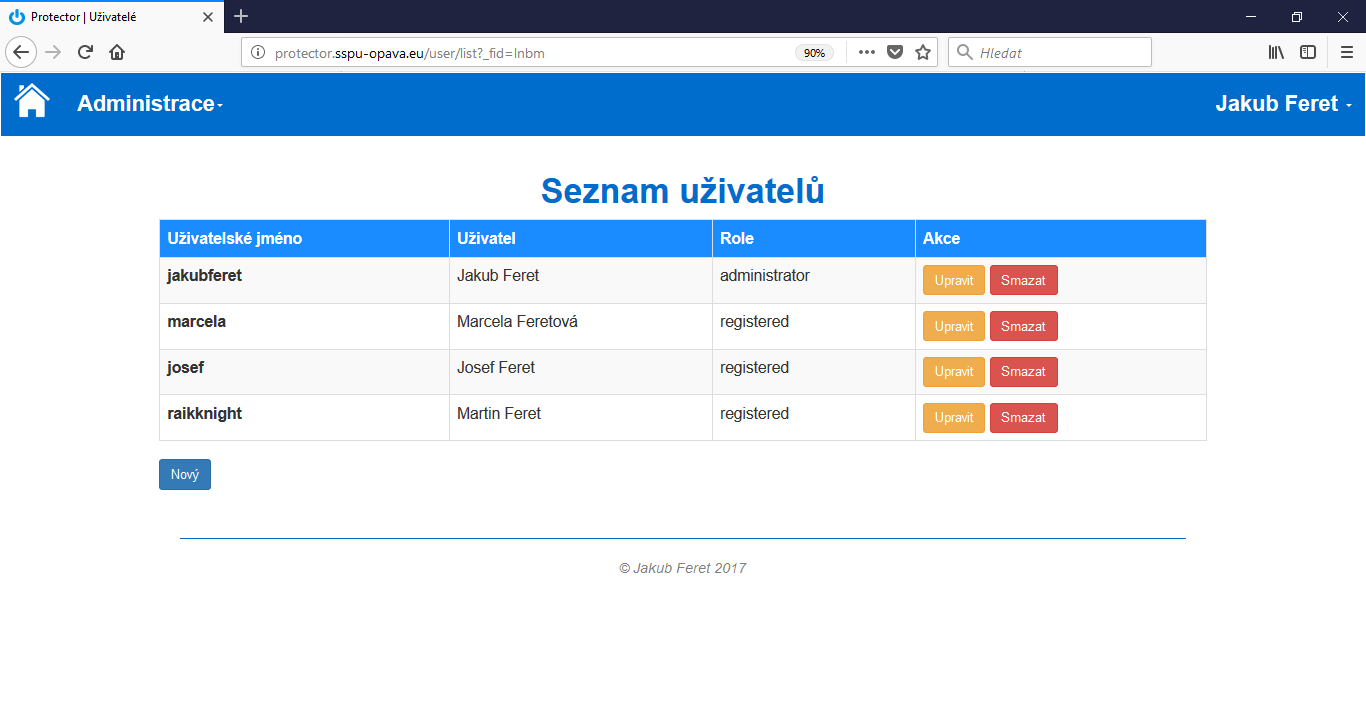
### Nastavení

Běžný uživatel může upravit své osobní údaje jako jméno nebo heslo (obrázek č. 12) a také může aktivovat nebo deaktivovat alarm.



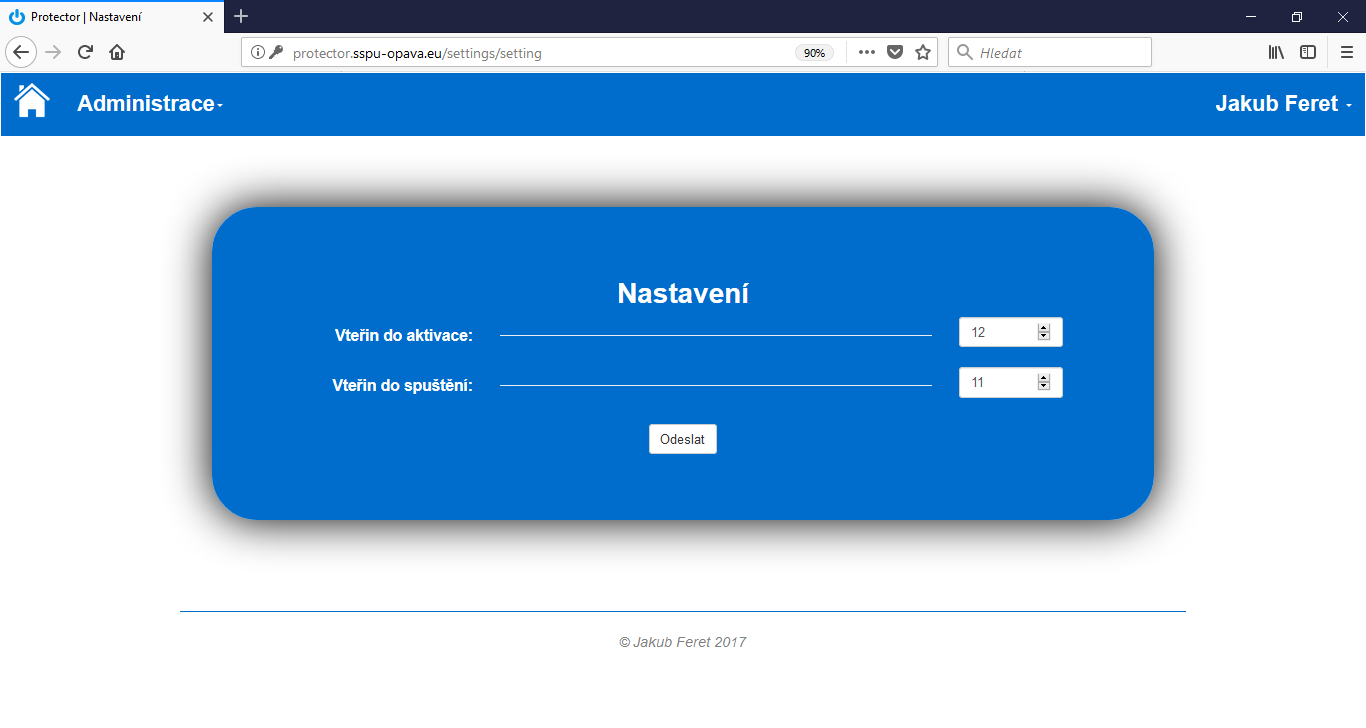
*Obrázek č. 12 Formulář pro změnu hesla*

Administrátor může kromě již zmíněného také upravovat údaje ostatních uživatelů, popřípadě přidávat nové uživatele nebo mazat současné (obrázek č. 13).



*Obrázek č. 13 Administrace uživatelů*

Má právo také upravovat dobu do aktivace a dobu do spuštění alarmu (obrázek č. 14).



*Obrázek č. 14 Nastavení bezpečnostního zařízení*

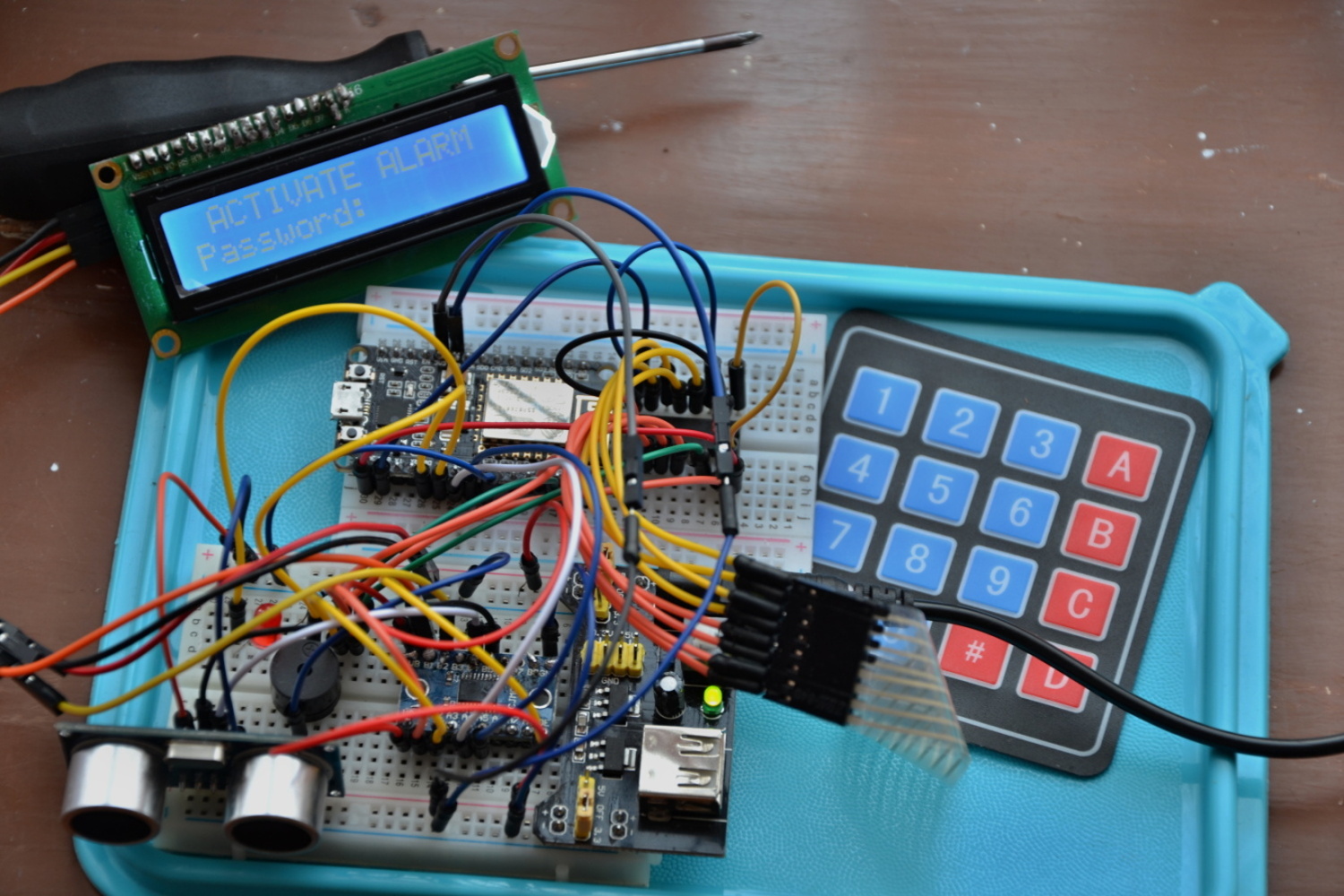
### Bezdtrátová komunikace

Důležitým úkolem webové aplikace je také komunikace s hardwarovým zařízením. Ta je řešena pomocí REST API. Za účelem poskytování dat pro ESP8266 jsem vytvořil funkci, která získá data z databáze a vytvoří z nich JSON výstup. Ten si potom snadno stáhne a dekóduje samotné zařízení pomocí GET požadavku. Získávání dat z ESP8266 jsem vyřešil pomocí POST požadavku ze zařízení, ve kterém posílám JSON výstup. Tato data pak v aplikaci zpracuji, JSON dekóduji a hodnoty uložím do databáze.

# Výsledky řešení

## Podoba hardwarového zařízení

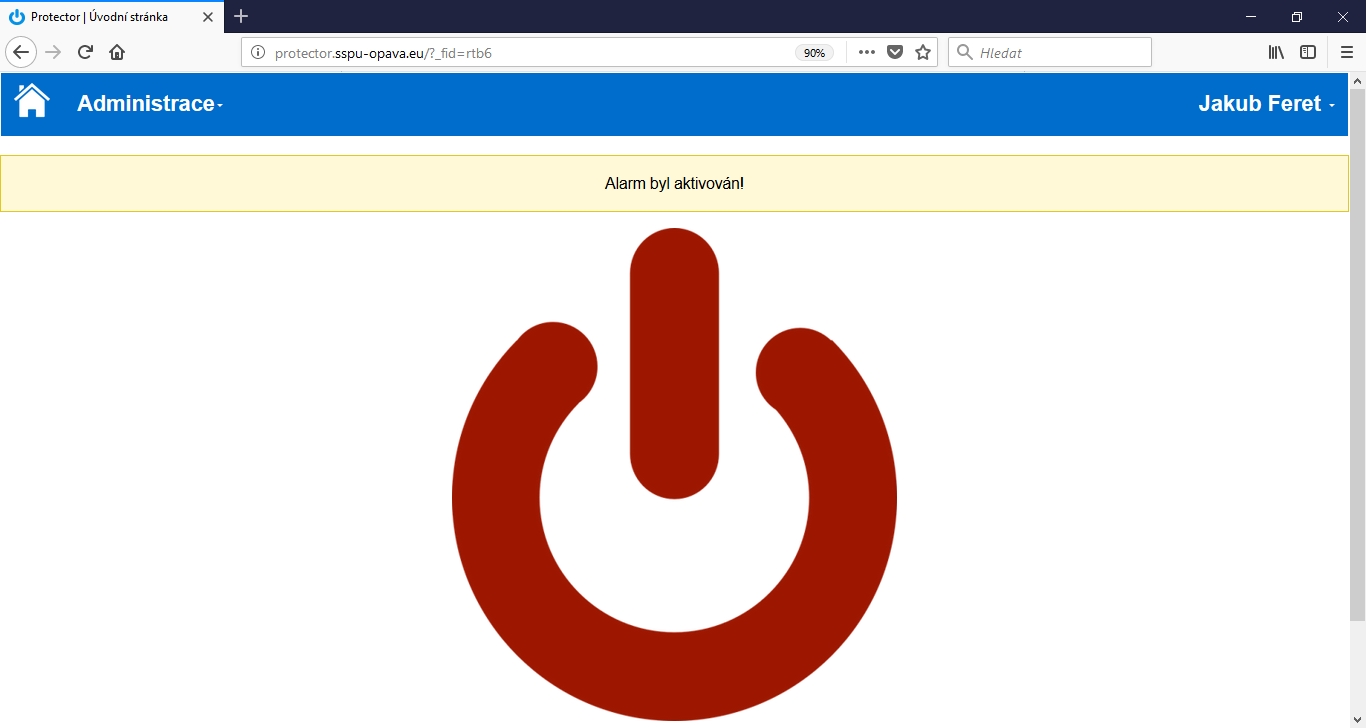
Bezpečnostní zařízení je v současné době plně funkční, po několika denním testování splňuje mé počáteční očekávání a všechny jeho funkce bez problému fungují. Z důvodu častých úprav je zařízení umístěno na dvou nepájivých polích. Do budoucna by bylo vhodné vytvořit pevný plošný spoj na poli pájivém, zejména kvůli křehkosti současného řešení. Systém jsem umístil do jednoduché krabičky s vyvrtanými otvory pro napájecí kabel a pro ultrazvukový senzor. Jak jsem již dříve v dokumentaci zmiňoval, napájení kabelem z elektrické sítě není u bezpečnostního systému vhodné, protože by případnému narušiteli stačilo vytrhnout napájecí kabel z elektrické zásuvky. Lepší by bylo napájet zařízení výkonnou baterií schovanou v krabičce, popřípadě systém umístit napevno do zdi a přímo připojit k elektrické síti.



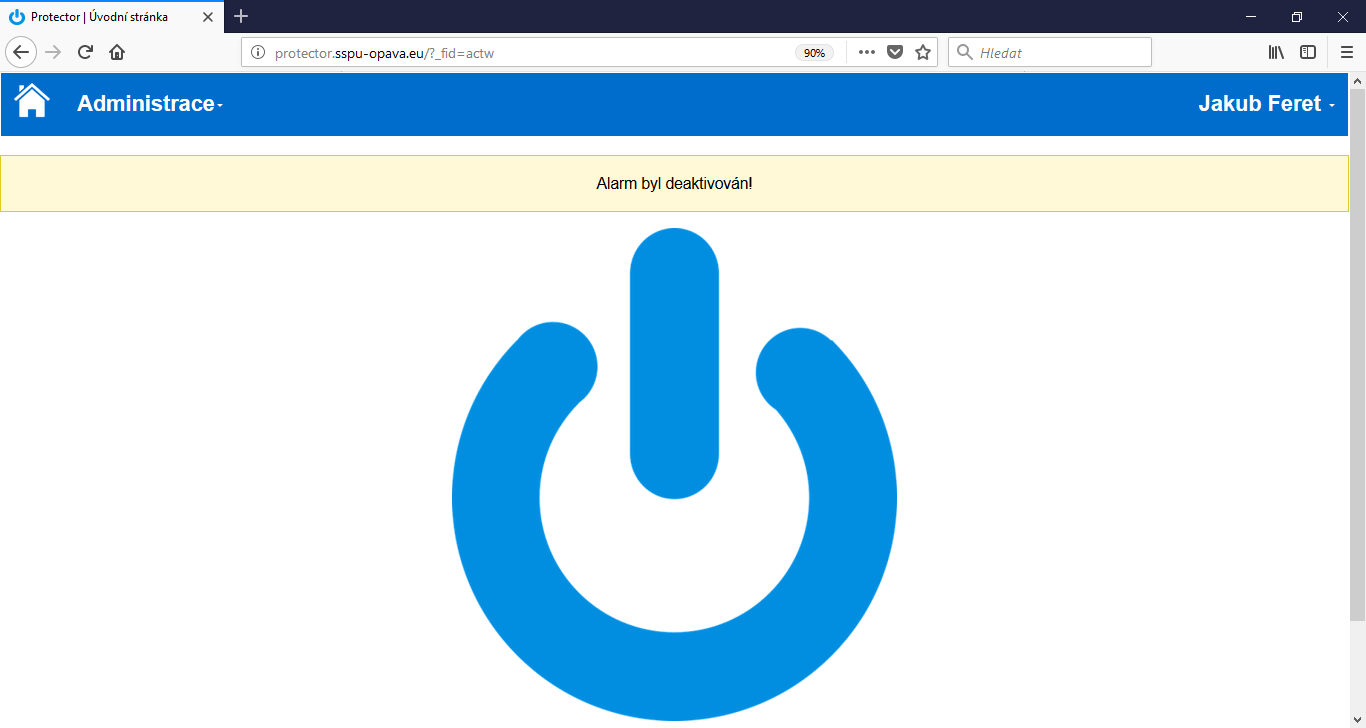
*Obrázek č. 15 Bezpečnostní zařízení ve vývoji*

## Podoba webové aplikace

Webová aplikace spolehlivě splňuje svůj účel, uživatelé se bez problému přihlásí a následně mohou v aplikaci dělat přesně to, co jim dovolují jejich uživatelská práva. Běžní uživatelé se proto nedostanou k administrátorským nastavením nebo k jiným nepřístupným částem aplikace. Administrátor naopak může kdykoliv měnit všechny údaje: své vlastní, údaje ostatních uživatelů a samozřejmě nastavení bezpečnostního zařízení. Do budoucna bych rád přidal více možností nastavení jako například automatickou aktivaci bezpečnostního systému v určité době a podobně.



*Obrázek č. 16 Tlačítko pro deaktivaci alarmu*



*Obrázek č. 17 Tlačítko pro aktivaci alarmu*

## Bezdrátová komunikace

Komunikace mezi webovou aplikací a hardwarovým zařízením na principu REST API funguje spolehlivě. Pro bezpečnostní zařízení však není nejvhodnější, jak z hlediska bezpečnosti, tak samotné funkcionality, jelikož je nezbytné zasílat ze zařízení požadavky na server v pravidelných časových intervalech a mezi těmito intervaly se zařízení nemá jak dozvědět o případných změnách v aplikaci (například o impulsu k aktivaci alarmu). Vhodnější by proto pro tento konkrétní případ bylo využití jiné technologie, například protokolu MQTT, který by zařízení se serverem spojil trvale a reagoval by pouze na změny. Rád bych také komunikaci lépe zabezpečil proti případným pokusům o narušení, například pomocí protokolu HTTPS.

**Závěr**

Cílem projektu bylo vytvořit funkční bezpečnostní systém s možností ovládat jej na dálku přes webovou aplikaci. Vytyčené cíle byly splněny, zařízení i webová aplikace bez problému fungují. Při vývoji mě nicméně napadlo nespočet možných změn, které by zařízení vylepšily, jako například napájení z baterie nebo lepší způsob komunikace mezi aplikací a zařízením. Na tato vylepšení mi již bohužel nestačil čas před termínem odevzdání práce, proto se jim budu věnovat později. Rád bych také přidal více možností nastavení do webové aplikace, jako například možnost automaticky spouštět alarm v určitou dobu a podobně. Samotné součástky bych rád umístil na desku plošných spojů místo dvou nepájivých polí, čímž bych vyřešil problém s křehkostí celého zařízení.

Seznam použitýCH INFORMAČNÍCH ZDROJů

[0] *ESP8266 Quick Start Guide* [online]. [cit. 2017-12-28]. Dostupné z: http://rancidbacon.com/files/kiwicon8/ESP8266\_WiFi\_Module\_Quick\_Start\_Guide\_v\_1.0.4.pdf

[2] *Logic Level Shifting* [online]. [cit. 2017-12-28]. Dostupné z: http://www.instructables.com/id/A-Quick-Guide-on-Logic-Level-Shifting/

[3] *ESP8266 Full Communication From Anywhere in the World* [online]. In: 12. 12. 2015 [cit. 2017-12-28]. Dostupné z: https://www.youtube.com/watch?v=uWbLpMJ8jiA

[4] *Nette autentifikace* [online]. [cit. 2017-12-28]. Dostupné z: https://doc.nette.org/cs/2.4/quickstart/authentication

[5] *Nette změna hesla* [online]. [cit. 2017-12-28]. Dostupné z: <https://github.com/nette/tutorial-todo/blob/master/app/presenters/UserPresenter.php>

[6] *Nette autentizace* [online]. [cit. 2017-12-28]. Dostupné z: <https://doc.nette.org/cs/2.4/access-control>

[7] *Nette Tracy* [online]. [cit. 2017-12-28]. Dostupné z: <https://tracy.nette.org/cs/>

[8] *Drahak REST API, chráněná komunikace* [online]. [cit. 2017-12-28]. Dostupné z: <https://www.itnetwork.cz/php/nette/diskuzni-forum-php-nette-framework/drahak-rest-api-chranena-komunikace-5767f69a2b12b>

[9] *JSON decode* [online]. [cit. 2017-12-28]. Dostupné z: http://php.net/manual/en/function.json-decode.php

[10] *ArduinoJson* [online]. [cit. 2017-12-28]. Dostupné z: https://github.com/bblanchon/ArduinoJson

[11] *ESP8266 Internet Controlled LED* [online]. [cit. 2017-12-28]. Dostupné z: http://blog.nyl.io/esp8266-led-arduino/

[12] *Arduino multitasking* [online]. [cit. 2017-12-28]. Dostupné z: https://learn.adafruit.com/multi-tasking-the-arduino-part-1/using-millis-for-timing

[13] *I2C Keypad* [online]. [cit. 2017-12-28]. Dostupné z: https://www.hackster.io/venkatesh\_rao/i2c-keypad-73a012

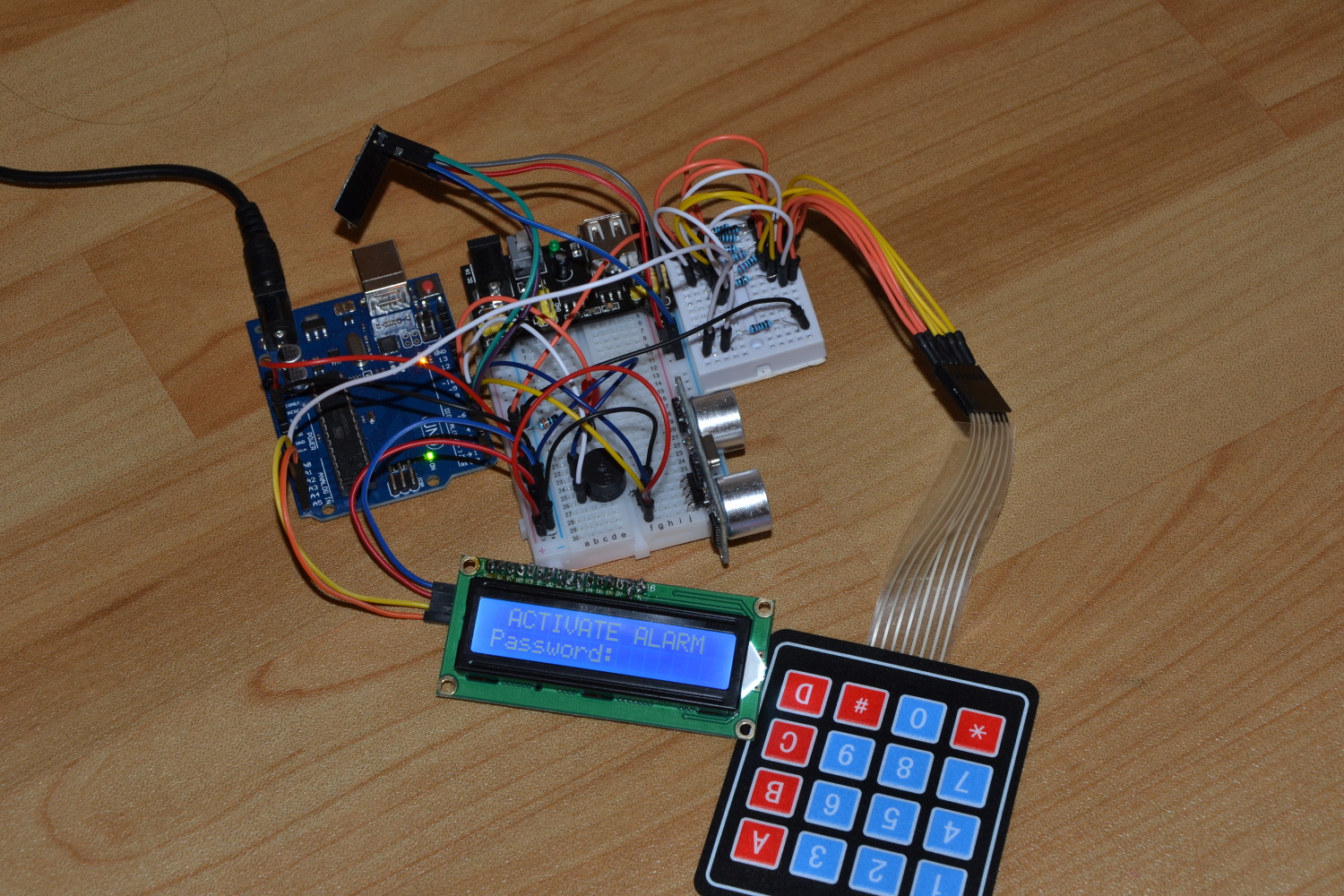
[14] *Insert JSON into database using PHP REST* [online]. In: 18. 1. 2016 [cit. 2017-12-28]. Dostupné z: <https://www.youtube.com/watch?v=4QYt2Yqo5TA>

[05] *ESP8266 EEPROM* [online]. [cit. 2017-12-28]. Dostupné z: https://circuits4you.com/2016/12/16/esp8266-internal-eeprom-arduino/

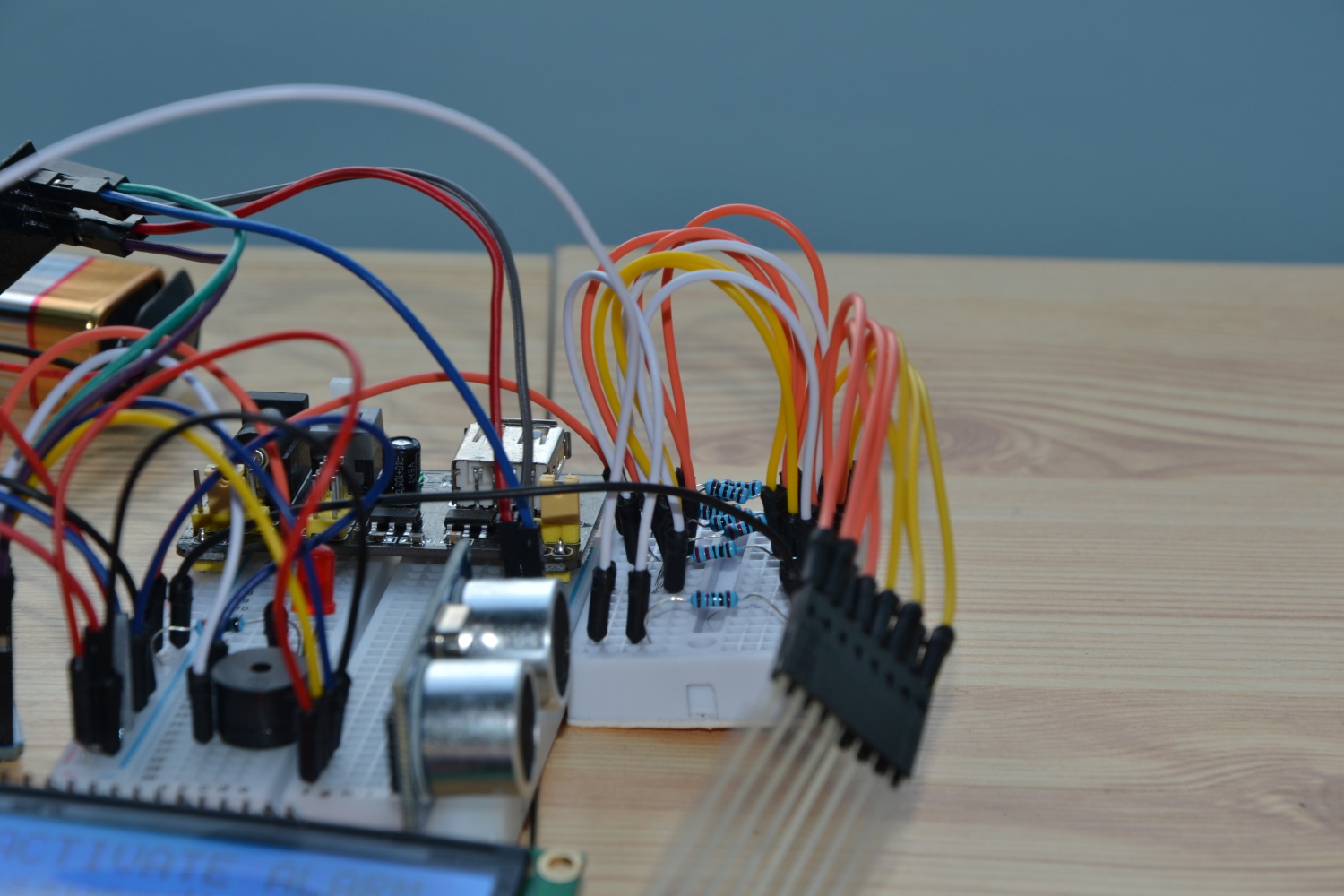
Seznam příloh

č. 1 Fotodokumentace

**Příloha č. 1: Fotodokumentace**



*První verze bezpečnostního zařízení složená z Arduino UNO a ESP8266-01*



*Detail na rezistorové pole sloužící k zapojení klávesnice přes jeden analogový pin, později nahrazené I2C sběrnicí*



*Výroba krabičky na bezpečnostní zařízení*



*Finální podoba bezpečnostního zařízení*