**Univerzita Hradec Králové**
**Fakulta informatiky a managementu**
**Katedra informačních technologií**

# Analýza možností systému iOS pro implementace Smart Home řešení

Diplomová práce

Autor: Bc. Václav Divíšek
Studijní obor: Aplikovaná informatika

Vedoucí práce: prof. Ing. Ondřej Krejcar, Ph.D.

Hradec Králové prosinec 2021

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 3.12.2021 Bc. Václav Divíšek

Poděkování:

Děkuji vedoucímu diplomové prof. Ing. Ondřej Krejcar, Ph.D. za trpělivost a metodické vedení práce.

# Anotace

Cílem diplomové práce je analyzovat možnosti propojení domácích zařízení komunikujících prostřednictvím infračerveného signálu (IR) s aplikací Apple Home. Propojení IR s Apple Home není běžně dostupné a diplomová práce řeší tento problém.

Pro tvorbu propojení IR byl použit kontrolér Arduino Uno R3 s Arduino Ethernet Shield, který je nezbytný pro připojení do domácí sítě. Jako server byl použit Synology DS918+, kam byly nainstalovány všechny potřebné komponenty pro diplomovou práci. Dále bylo vytvořeno API a uživatelské rozhraní umožňující instalaci vytvořeného zařízení do aplikace Home. API bylo autorem naprogramováno a byly využity technologie Blazor Server pro uživatelské rozhraní a ASP NET Core 5 pro API. Uživatelské rozhraní je realizováno prostřednictvím webové aplikace, kde byl především kladen důraz na jednoduchost ovládání. V závěru práce je uveden způsob testování, který byl proveden a zajišťuje správnou funkcionalitu a zabezpečení zařízení.

Výsledkem diplomové práce je vytvořené připojení IR zařízení do Apple Home a jeho ovládání z této aplikace. V aplikaci Home je možné ovládat všechny druhy IR osvětlení (vypnuto/zapnuto, jas, barva osvětlení) a všechny ostatní IR zařízení v režimu vypnuto/zapnuto.

# Annotation

## Title: Analysis of Possibilities for Smart Home Solution on iOS Platform

The master's thesis analyses the possibilities of connecting home devices communicating via infrared light (IR) with Apple Home application. Connecting IR to Apple Home is not generally available and the thesis aims to solve this problem.

IR connection was controlled by Arduino Uno R3 with Ethernet Shield, which is necessary for connecting to the home network. All necessary components fort this thesis were installed in the server Synology DS918+. API and user interface were created to enable the installation of the crafted device into the Home application. The API and the user interface were programmed by the author of the thesis using Blazor Server technology for the user interface and ASP NET Core 5 for the API. The user interface was implemented in the form of a web application where the emphasis was placed mainly on its simplicity. The thesis presents a testing method which has been applied and ensures a proper functionality and security of the created device.

The result of the master's thesis is the connection of an IR device to Apple Home and device control based on its application. All types of IR lights (off / on, brightness, lighting color) and all other IR devices in off / on mode can be used in the Home application using the created device.

# Obsah

# Seznam obrázků

# Seznam tabulek

# 1 Úvod

V dnešní digitální době se lidé snaží vše automatizovat a zefektivnit. Tomuto automatizování se nevyhnuli ani naše domovy. Hlavním důvodem pro automatizování domácnosti může být hned několik. Může se jednat o ušetření nákladů, pohodlnější život nebo třeba vyšší efektivitu. Toho lze docílit např.: zapojením termostatu ke kotli a nastavením, kdy se má kotel sepnout a při jakých teplotách. Dalším příkladem automatizování domácnosti mohou být čidla u světel, nastavením, kdy světla reagují na pohyb. Automatizací to ale neskončilo, bylo totiž žádoucí, aby spolu jednotlivé automatizované prvky mohly komunikovat. S postupným rozšiřováním internetové sítě a domácí sítě, následným používáním bezdrátové sítě se tato komunikace začala realizovat. Připojení těchto prvků (různá čidla, senzory, domácí spotřebiče atd.) se nazývá internet věcí, častěji se ale používá anglické označení Internet of Things, nebo zkráceně IoT. Termín IoT poprvé zmínil Kevin Ashton při prezentaci v Procter & Gamble v roce 1999.

Chytrá domácnost využívá zařízení (např.: čidlo teploty, světla, žaluzie atd.) připojených do sítě a ovládá jednotlivá zařízení. Často je tyto zařízení možné ovládat vzdáleně, jelikož jsou připojeny k internetu. Tím že jsou připojeny k internetu, je možné tato zařízení ovládat i vzdáleně pomocí mobilního telefonu nebo tabletu.

V úvodní části diplomové práce jsou stručně uvedeny dostupné technologie, které jsou k dispozici pro komunikaci chytrých zařízení. Kde jsou uvedeny základní informace o těchto technologiích.

Problém ale nastává, pokud zařízení nepoužívá moderní bezdrátovou komunikaci a používá infraport (tuto technologii používá většina starších ovladačů například televizních zařízení, klimatizací a levnějších světelných zařízení). Pro infraport jsou sice dostupná řešení, ale žádné z nalezených zařízení nepodporuje integraci do HomeKit společnosti Apple.

Proto bylo rozhodnuto navrhnout zařízení, které tento nedostatek propojení IR zařízení do aplikace Home bude odstraňovat. K tomuto zařízení je poskytnuto uživatelské rozhraní pro administraci IR kódů a přiřazování k jednotlivým koncovým IR zařízením.

Úvodní část diplomové práce (kapitola 3) je věnována přehledu nejčastěji používaných druhů bezdrátové komunikace používané v chytré domácnosti. Následující kapitola 4 pojednává o IR komunikaci, která je obsažena ve velkém množství zařízení v domácnosti a také analyzuje již dostupná řešení pro integraci do Apple Home. Po odhalení nedostatků dostupných řešení je navrženo vlastní řešení, které odstraňuje tyto nedostatky. Návrh vlastního řešení lze nalézt v 5. kapitole. V této kapitole se nachází schéma návrhu vlastního řešení, Ir přijímače/vysílače a možnosti komunikace s Apple HomeKit. Na tuto kapitolu navazuje 6. kapitola, která je věnována výběru technologií pro implementaci a jednotlivých částí vlastního řešení. V závěru diplomové práce je obsažen způsob, jakým bylo řešení testováno (kapitola 7) a dosažené výsledky (kapitola 8 a 9)

# 2 Cíle práce

Hlavním cílem diplomové práce je implementace vlastního zařízení, které je schopné připojit nalezené vzdáleně ovládané domácí zařízení do ekosystému HomeKit. V rámci postupu řešení je třeba seznámit se s technologiemi používající HomeKit (ekosystém pro chytré domácnosti vyvinutý značkou Apple Inc.) a pochopit jeho fungování. Dále je třeba provést analýzu trhu s chytrými zařízeními a komunikace která je používána. Po zanalyzování komunikačních kanálů chytrých zařízení jsou v práci popsána dostupná domácí zařízení a především technologie komunikace, které jsou používána na těchto zařízení. Dále bylo nutné seznámit se s dostupnými řešeními na trhu. Návrh obsahuje plnou implementaci ovládání, včetně všech jeho částí a zvolených technologií.

# 3 Chytrá domácnost

Pro chytrou domácnost může být použito v češtině více názvů. Nejčastěji se používají dvě česká slova chytrá domácnost a inteligentní domácnost. Dále se využívají anglické termíny SmartHome nebo Smart home. Dále je potřeba si definovat slovo IoT (Internet of Things – česky Internet věcí). IoT je definováno jako systém vzájemně propojených výpočetních zařízení, mechanických a digitálních strojů, předmětů, zvířat nebo lidí, které mají unikátní identifikátory (UID) a schopnost přenášet data po síti bez interakce z člověka na člověka nebo z člověka na počítač [1].

Z definice výše tedy zjistíme, že chytrá domácnost je pouze výsek IoT.

Co si představit pod pojmem chytrá domácnost? Pod tímto pojmem si můžeme představit skupinu zařízení (např. osvětlení, klimatizace, zámek, topení, …), která spolu mohou komunikovat napřímo nebo přes jiné zařízení. Jelikož spolu mohou komunikovat je možné je dále automatizovat. Příkladem takovéto automatizace je například nastavení odchodu z bytu. Po odchodu poslední osoby z bytu se vypne osvětlení, zamknou dveře a nastaví se udržovaní teploty v místnostech. Toto vše se provede bez jakéhokoliv zásahu uživatele, jelikož zařízení spolu mohou komunikovat.

## 3.1 Komunikace IoT

Pro komunikaci konceptu IoT lze použít buď fyzické připojení (kabel) do počítačové sítě nebo bezdrátové. Při propojením IoT bezdrátově vzniká větší nebezpečí, že bude toto zařízení napadnuto. Problém je v šíření informací rádiovými vlnami, které se snadno odposlouchají, je-li útočník v dosahu. Proto je nutné komunikaci zabezpečit, aby toto nebylo možné. Nejčastěji se používá párování zařízení při inicializaci. Většina certifikovaných IoT v HomeKit používá níže uvedení řešení.

### 3.1.1 ZigBee

ZigBee (logo na Obrázek 1) je jediné kompletní řešení IoT – od počítačové sítě po univerzální jazyk, který umožňuje inteligentním objektům spolupracovat. ZigBee zlepšuje výběr a flexibilitu vývoje IoT pro uživatele a vývojáře. Dodává jim jistotu, že produkty a služby budou spolupracovat prostřednictvím standardizace a testování celé architektury [2].

ZigBee protokol je v IoT nejrozšířenějším druhem komunikace (bylo zaregistrováno přes 3500 zařízeních používajících ZigBee) [3]. Tento druh komunikace používá např. Philips Hue, Amazon Echo, Ikea, Samsung a mnoho dalších.


**Obrázek 1: Logo ZigBee [4]**

### 3.1.2 Z-Wave

Z-Wave (logo na Obrázek 2) je přední bezdrátová technologie používaná mnoha bezpečnými a důvěryhodnými značkami, které se snaží, aby byl každý chytrý dům chytřejší a bezpečnější. Díky Z-Wave mohou produkty pro inteligentní domácnosti navzájem komunikovat bez ohledu na to, na jaké značce nebo platformě jsou postaveny. To vše díky centrálnímu inteligentnímu rozbočovači.[5]

Z-Wave je druhá nejrozšířenější komunikace (zaregistrováno přes 3200 zařízení používajících Z-Wave oproti 3500 používajících ZigBee) [5]. Mezi nejznámější firmy používající Z-Wave je FIBARO.


**Obrázek 2: Logo Z-Wave [5]**

## 3.2 Certifikace HomeKit

Na českém trhu existuje mnoho zařízení, které nativně podporuje Apple HomeKit. Proto aby zařízení dostalo certifikaci HomeKit musí zařízení splňovat podmínky MFi (Made For iPhone) [6] programu. Tento program zaručuje, že při vytvoření nového zařízení nebude zneužito osobních informací koncového uživatele. Zapojením do MFi programu uživatel získá technickou specifikaci, certifikační nástroje a mnoho dalšího. Apple doporučuje zapojení do programu MFi všem vývojářům zařízení, výrobcům zařízení nebo vlastníkům, kteří přebírají zodpovědnost za certifikaci příslušenství MFi (logo certifikací Obrázek 3). Tento MFi program je bezplatný, ale pro zapsání do MFi je nezbytné zadat vlastní firmu. Aby bylo možné používat certifikaci Apple jsou nutné následující kroky:

1. Produktový plán – Odeslání produktového plánu pro příslušenství, která integruje technologii s licencí MFi.
2. Vývoj – Design, vývoj a testování příslušenství. Podle potřeby si uživatel obstará všechny komponenty MFi.
3. Certifikace – Využití certifikačního nástroje společnosti Apple k zajištění optimální uživatelské zkušenosti a interoperability. Odeslání ke kontrole vzorky připravené k výrobě s balením, v kterém bude produkt prodávám.
4. Masová produkce – Po dokončení certifikace schválené společností Apple lze začít s výrobou a prodejem.


**Obrázek 3: Certifikace Apple pro různá zařízení [7]**

# 4 IR pro SmartHome

Infračervené (IR), někdy nazývané infračervené světlo, je elektromagnetické záření (EMR) s vlnovými délkami delšími než viditelné světlo. Pro lidské oko je tedy neviditelné. IR se obecně chápáno, jako vlnové délky od nominální červené hrany viditelného spektra přibližně 700 nanometrů (frekvence 430 THz) až po 300 GHz[8].

Infračervené světlo je použito pro komunikaci mezi zařízeními. Komunikaci si velice zjednodušeně můžeme představit jako světelnou Morseovu abecedu. Přenosová rychlost se pohybuje v rámci desítek kb/s v závislosti na protokolu. Obrovskou nevýhodou tohoto zařízení je, že vysílač a přijímač musí být v přímém dohledu. Ačkoliv se jedná o starou technologii, je neustále používána pro nejrůznější domácí ovladače. Můžeme ji najít například u většiny televizních ovladačů (novější TV přecházejí na Bluetooth), klimatizačních jednotek a levnějších dálkových osvětlení.

## *4.1 Přehled existujících IR ovladačů*

Na trhu existuje velké množství univerzálních IR ovladačů. Lze je rozdělit na dva typy: ruční dálkový ovladač (je malý a bezdrátový) a stolní dálkový ovladač. Ruční dálkový ovladač si můžeme představit, jako klasický ovladač pro televizi, nebo například hifi. Stolní dálkový ovladač je naopak větší. Umístěn bývá většinou na skříňce nebo stolku.

Ruční dálkový ovladač (Obrázek 4) – Jeho výhoda je ve velikosti a skladnosti. Používá se, pokud ztratíte originální televizní ovládání, nebo chcete ovládat více zařízení najednou. Obrovská výhoda v ovládání více zařízení najednou je, pokud máme například televizi a satelitní přijímač. Uživateli stačí jeden univerzální ovladač pro obě zařízení. Nevýhoda je, že tento univerzální ovladač se složitější na nastavení i následující používání. Další nevýhoda je nemožnost ručního dálkového IR ovladače připojení do Smart home.


**Obrázek 4: Ruční univerzální dálkový ovladač (Meliconi CONTROL 2.1) [9]**

Stolní dálkový IR ovladač – Tento typ je opakem ručního dálkového ovladače. Co u ručního dálkového IR ovladače jsou výhody, tak stolní dálkový IR ovladač neposkytuje a naopak, co jsou nevýhody tak poskytuje. Obrovská výhoda je, že tato stolní dálkové IR ovladače jsou připojeny do domácí sítě. Dále je pro ovládání použita mobilní aplikace. Novější modely těchto zařízení už podporují chytré služby jako Google Assistant, Amazon Alexa a IFTTT. Bohužel ale kontabilita s HomeKit neexistuje. Po delším vyhledávání a pochopení ekosystému HomeKit byly nalezeny dva stolní IR ovladače, které nepodporují přímo HomeKit, ale lze doimplementovat tato funkcionalita pomocí instalace pluginu do HAP (viz. kapitola 5.2.) od Homebridge. Jedná se o zařízení BOND [10] (neprodejné v ČR, cena 100$) a BROADLINK RM4C MINI (cena 500 Kč). Druhé zmíněné zařízení BROADLINK RM4C MINI (Obrázek 5) po nainstalování pluginu „Homebridge Broadlink Rm“ [11] poskytuje ovládání jednoduchých vypínačů, závěsů, větráku, světla, garážových vrat, zámku, okenních rolet a klimatizace. Zařízení BOND je v možnosti

ovládání o dost skromnější, lze totiž ovládat pouze krb, větrák, okenní závěsy a všechna zařízení pouze pro vypnutí a zapnutí.

**Obrázek 5: Stolní dálkový IR ovladač BROADLINK RM4C MINI (vlastní zpracování)**

| | BROADLINK RM4C MINI | BOND |
|---|---|---|
| Dostupné v ČR | 0 | X |
| Možnost automatizace | 0 | 0 |
| Počet zařízení | 8 | 4 |
| Google Assistant | 0 | 0 |
| Amazon Alexa | 0 | 0 |
| Cena | 500 Kč | 100 $ (cca 2 050 Kč) |

**Tabulka 1: Porovnání BOND a BROADLINK RM4C MINI (vlastní zpracování)**

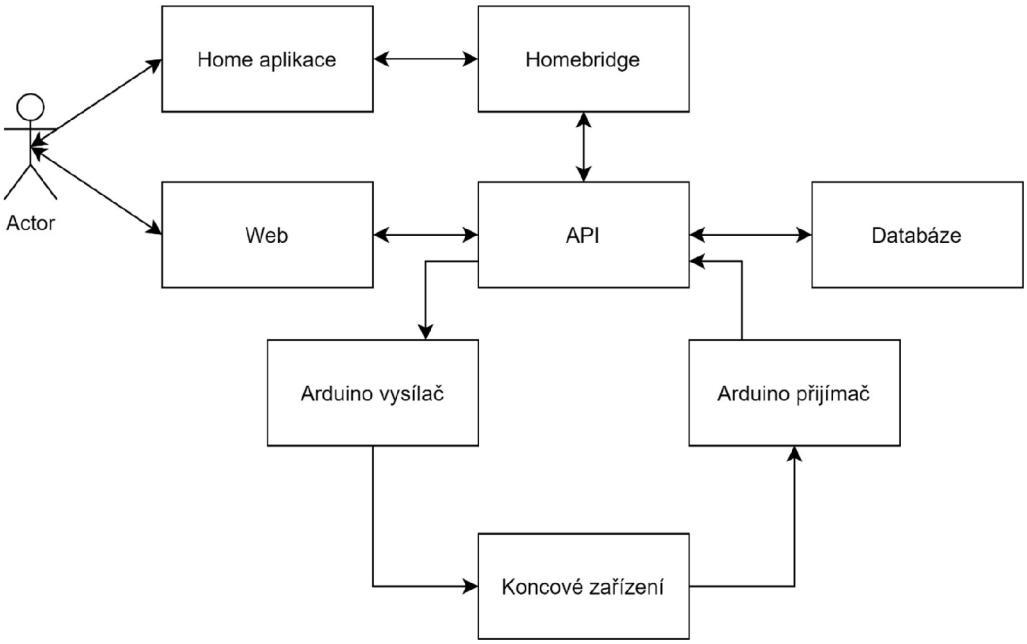
## 4.2 Přehled nekomerčních zařízení

Při hledání v odborných článcích, bakalářských pracích a diplomových pracích nebylo nalezeno žádné řešení, které by implementovalo IR ovládací zařízení do HomeKit. Při hledání bakalářských a diplomových prací bylo prohledáno theses.cz. Pro vyhledávání v odborných článcích jsem prohledával EBSCO, Web of Science, Scopus a SAGE Journals. Při vyhledávání byla použita kombinace klíčových slov IR, SmartHome, HomeKit a Infrared. Nejblíže nalezeným tématem byla práce „Application of universal remote control of non-smart home appliances for smart home concepts“ [12], ve které se autoři Jan Dvořák a Ondřej Krejcar zabývají tvorbou univerzálního IR zařízení, které replikuje IR signál.

# 5 Návrh vlastního zařízení

Jelikož nebylo nalezeno na trhu žádné zařízení, které by umožnilo lehkou implementaci IR ovládání, bylo rozhodnuto o vytvoření vlastního řešení. Pro IR ovládání k chytré domácnosti je potřeba si vytvořit vlastní zařízení, které se bude starat o obsluhu komunikace mezi chytrou domácností a „neinteligentním“ zařízením (televize, klimatizace, …). Navržené řešení počítá s následujícími prvky (vyobrazení komunikace na Obrázku 6).

1) Ovládací zařízení (Home aplikace) – zařízení na kterém je operační systém macOS nebo iOS, které vlastní aplikaci Home.
2) Koncové zařízení – zařízení, které disponuje IR ovládáním a připojujeme do chytré domácnosti
3) Přijímač a vysílač IR – přijímač a vysílač, který bude zpracovávat signál z koncového zařízení
4) Web – uživatelské rozhraní pro přiřazování signálů k zařízením
5) Databáze – uchovávající všechna uživatelská nastavení
6) Homebridge – HAP server obstarávající komunikaci podle HAP specifikace s
7) API – hlavní rozhraní pro výměnu informací od Homebridge, web, přijímače/vysílače IR a komunikaci s Databází




**Obrázek 6: Komunikace napříč vlastním řešením (vlastní zpracování)**

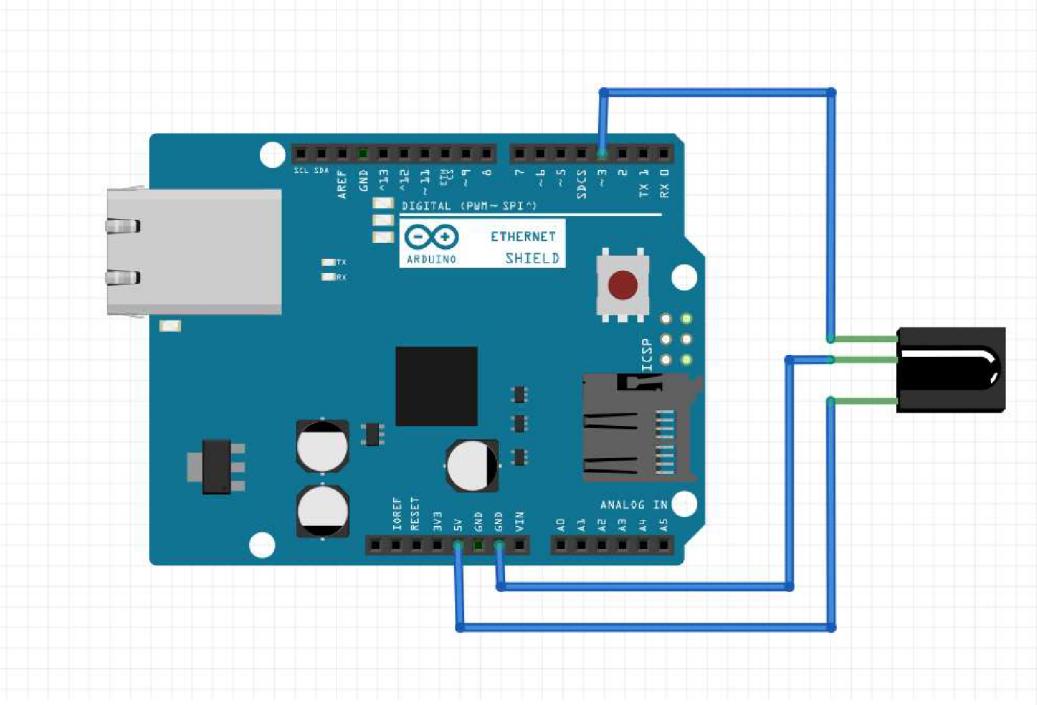
## 5.1 Návrh přijímače a vysílače IR

Pro realizaci přijímače a vysílače bylo rozhodnuto použít Arduino UNO R3. Arduino UNO R3 je mikrokontrolérová vývojová deska, která je založena na čipu ATmega328, která obsahuje 14 digitálních vstupů/výstupů a 6 analogových vstupů. Arduino UNO R3 jsem si vybral pro realizaci vlastního řešení, jelikož je jednoduše dostupné, lehce programovatelné a cenově výhodné (lze zakoupit od 149 Kč na dratek.cz, nebo okolo 4$ - cca 90Kč + clo z Aliexpress.com). Dále byla použita IR dioda, zde jsem použil IR vysílač LED 5 mm 940nm (cena 3Kč na laskarduino.cz), a infračervený přijímač pro dálkové ovládání (VS1838B, cena 5,40Kč na laskarduino.cz). Pro připojení do domácí sítě je potřeba buď Ethernet Shield W5100 (cena 329Kč na laskarduino) nebo Wifi modul (Arduino WiFi Shield, cena 2136 Kč na hwkitchen.cz). Bylo zvoleno použití Ethernet Shield pro simulaci a vývoj, jelikož Arduino WiFi Shield je finančně o dost náročnější.

### 5.1.1 Návrh přijímače IR

Pro přijímač je potřeba Arduino UNO R3, infračervený přijímač pro dálkové ovládání (VS1838B) a samozřejmě Arduino Wifi Shield nebo Arduino Ethernet Shield. Arduino Wifi Shield nebo Arduino Ethernet Shield je lehce zapojitelné do Arduino UNO R3, jelikož stačí nasadit tento Shield na Arduino UNO R3. Schéma zapojení je na Obrázku 7 a součástky s cenou v Tabulce 2.

Přijímač se stará o přijetí signálu z požadovaného zařízení IR zařízení, které připojujeme do chytré domácnosti. Dále je tento signál dešifrován a poslán přes Arduino Ethernet Shield na API.




**Obrázek 7: Zapojení přijímače IR signálu do Arduino Ethernet Shield (vlastní zpracování)**

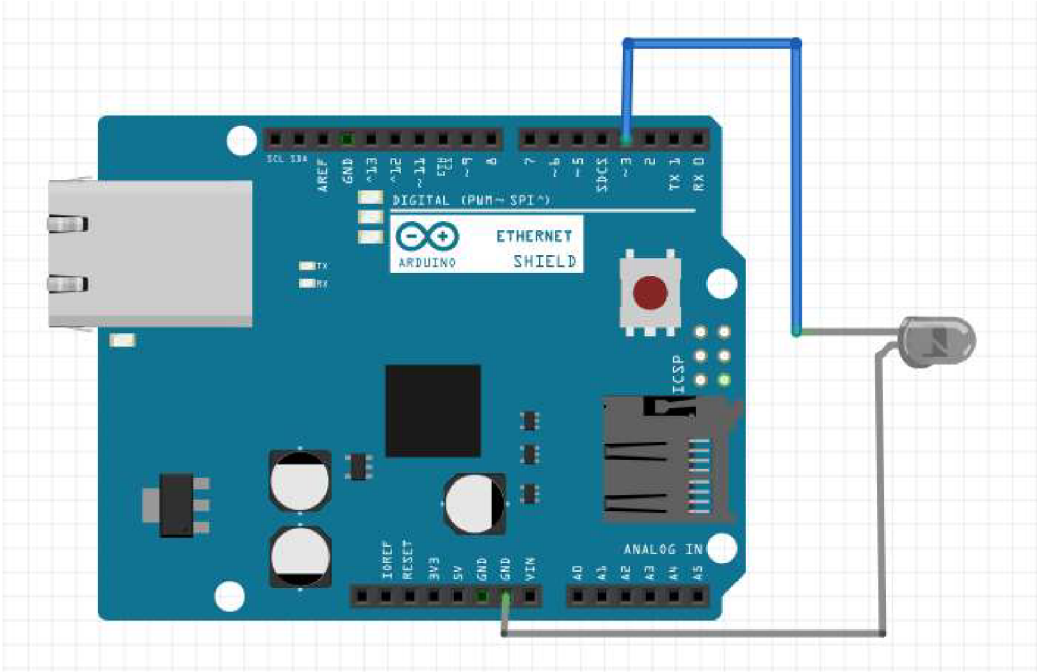
| Výrobek | Cena |
|---|---|
| Arduino UNO R3 | 149 Kč |
| infračervený přijímač (VS1838B) | 5,40Kč |
| Ethernet Shield | 329 Kč |
| Celková cena s Ethernet Shield | 483,40Kč |

**Tabulka 2: Cena Arduino přijímače (vlastní zpracování)**

### 5.1.2 Návrh vysílače IR

Pro vysílač bude potřeba Arduino UNO R3 (149 Kč), IR vysílač LED 5 mm 940nm (3Kč) a Arduino Ethernet Shield nebo Arduino WiFi Shield. Schéma zapojení je na Obrázku 8 a součástky s cenou v Tabulce 3.

Vysílač je připojen do domácí sítě, kde naslouchá na TCP. Pokud je proveden nějaký příkaz z HomeKit, API pošle na toto TCP příkaz, který se má vykonat. Vysílač pomocí knihovny IRremote tento příkaz zakóduje požadovaným protokolem, které používá koncové zařízení a dioda předá pomocí infračerveného světla příkaz zařízení.




**Obrázek 8: Zapojení vysílače IR signálu do Ethernet Shield (vlastní zpracování)**

| Výrobek | Cena |
|---|---|
| Arduino UNO R3 | 149 Kč |
| IR vysílač LED | 3 Kč |
| Ethernet Shield | 329 Kč |
| Celková cena s Ethernet Shield | 481Kč |

**Tabulka 3: Cena Arduino vysílače (vlastní zpracování)**

## 5.2 Komunikace s HomeKit

HomeKit je systém vyvinut společností Apple Inc., který umožňuje ovládat všechna inteligentní kompatibilní zařízení. Tato kompatibilní zařízení jsou označena logem (logo na Obrázku 9).

Pro framework jako je HomeKit (a HealthKit) je potřeba relativně skromná investice do dovednosti a času potřebného k vývoji veškerého příslušenství třetích stran, jenž je integrováno do HomeKit. HomeKit umožňuje zbavení se enormního množství kódu, který by bylo nutné psát pro vytvoření moderní technologie pro chytré domácnosti.[13]

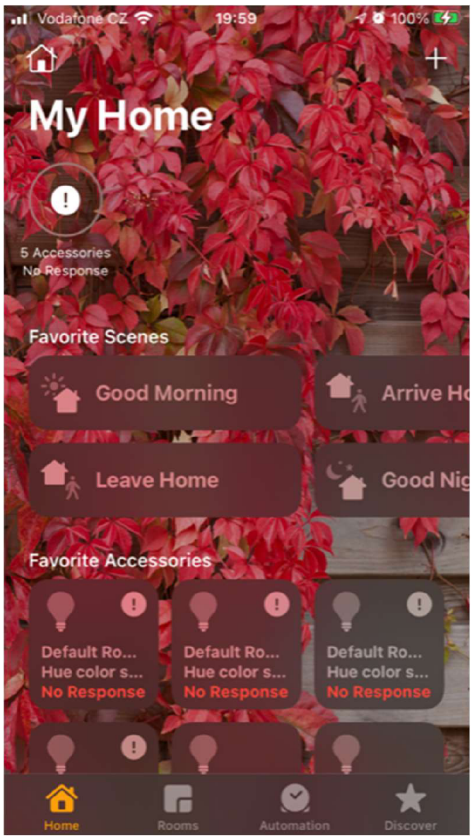

**Obrázek 9: Logo kompatibilního zařízení s HomeKit[14]**

Pro komunikaci HomeKit používá takzvaný HAP (HomeKit Accessory Protocol). K protokolu je dostupná dokumentace (HomeKit Accessory Protocol Specification – HAP Specification), která je rozdělena na dvě části. První část je určena pro komerční použití a není veřejně dostupná. Druhá část je pro nekomerční použití a je volně dostupná na https://developer.apple.com/HomeKit/. HAP Specification definuje všechny zařízení, která je možné ovládat pomocí HomeKit. Nalezneme zde například: zámek dveří, otevírání garážových dveří, zvonek, klimatizaci, větrák, zásuvku, otevírání oken, nejrůznější světla, ad. HAP Specification je ale volně dostupná na odkaze https://developer.apple.com/HomeKit/specification/. Bohužel není možné ukázat ukázku dokumentace, jelikož Apple Inc. to striktně zakazuje. Nicméně je dost podobná konfiguraci pro Homebridge viz. obrázek 32.

Dále HAP Specification definuje základní funkce, bez které HAP není možné provozovat. Jako je například párování zařízení, vyhledávání zařízení v domácí síti, stavové kódy, notifikace, ovládání zařízení a mnoho další.

Jelikož na trhu existuje mnoho hotových řešení pro HAP, bylo rozhodnuto tedy proto použít již hotové řešení. Hlavním důvodem bylo, že hotová řešení jsou velice komplexní a uživatelsky přívětivá. Na základě rešerše bylo zvoleno jako nejvhodnější řešení Homebridge, které bylo založeno Nickem Farinou a Khaos Tianem a je dostupné i jako doplněk do NAS od firmy Synology. Mezi další dostupná řešen jsou například Hoobs, Home Assistant, OpenHab a další.

HAP server komunikuje přímo s aplikací Home (ukázka aplikace na Obrázku 10) od značky Apple Inc., která se stará o vzdálené ovládání zařízení a Homebridge.



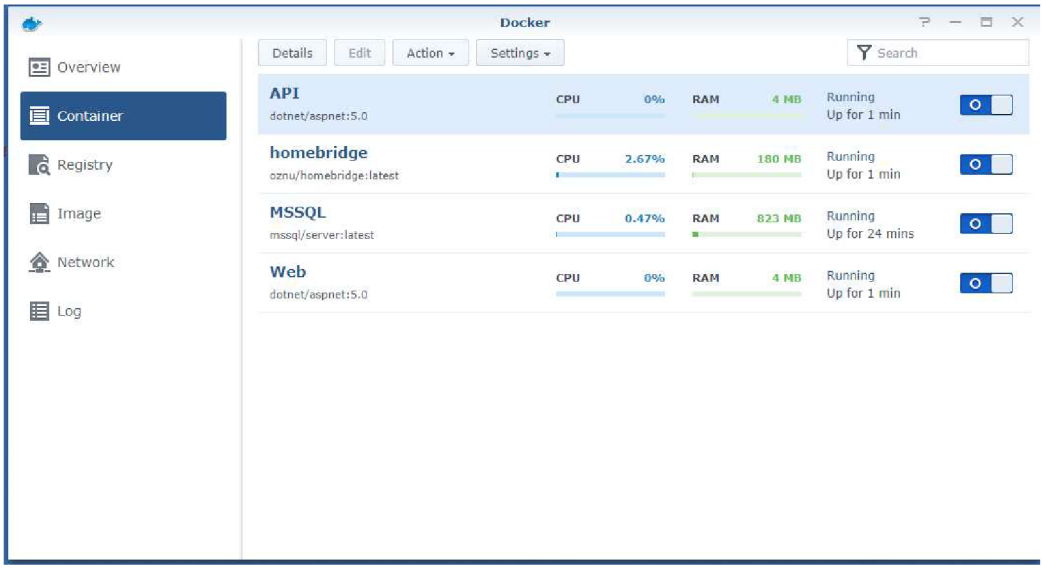

**Obrázek 10: Ukázka aplikace Home na iOS (mobilním telefonu) (vlastní zpracování)**

### 5.2.1 Homebridge

Homebridge je napsaný v systému NodeJS, lze provozovat ve své domácí síti a jenž emuluje rozhraní API HomeKit pro iOS. Podporuje pluginy, což jsou moduly přispívané komunitou, které poskytují základní most z HomeKit k různým API třetích stran poskytovaným výrobci zařízení SmartHome. [15]

Homebridge je napsán v jazyce NodeJS a je tudíž dostupný na všech platformách (Linux, Windows, macOS a dokonce i jako ISO pro Raspberry Pi a různé NAS servery). Pro vlastní vzorové řešení využiji domácí NAS server Synology DS916+.

Homebridge je využito jako rozhraní mezi ovládacím prvkem Home aplikace a API. Homebridge příjme informaci od Home aplikace a předá ji pomocí konfiguračního souboru na API. Konfigurační soubor je formátován ve standartu JSON. V podstatě se jedná o definici všech koncových zařízení stejně jako tomu je v HAP Specifikaci zařízení, pouze v rozšířeném formátu. Tato specifikace je volně dostupná na adrese https://developers.homebridge.io/#/service/AccessControl.




**Obrázek 11: Ukázka Docker kontajnerů (vlastní zpracování)**

# 6 Implementace

V rámci implementace Arduino přijímače a vysílače vyžaduje nakonfigurovat domácí síť a především nastavit statickou IP adresu, na které bude Arduino obsluhováno. Toto lze realizovat v rámci sítě na úrovni domácího routeru, který obsahuje DHPC server. Statickou IP adresu pro Arduino lze nastavit následujícími způsoby:

1. vyčleněním IP adres, které DHCP server nebude přiřazovat a poté nastavit IP adresu z vyčleněných Arduino zařízení
2. přiřazením MAC adresy k IP adrese, poté DHCP server bude přiřazovat tuto IP adresu na základě MAC adresy

## *6.1 Implementace přijímače a vysílače IR*

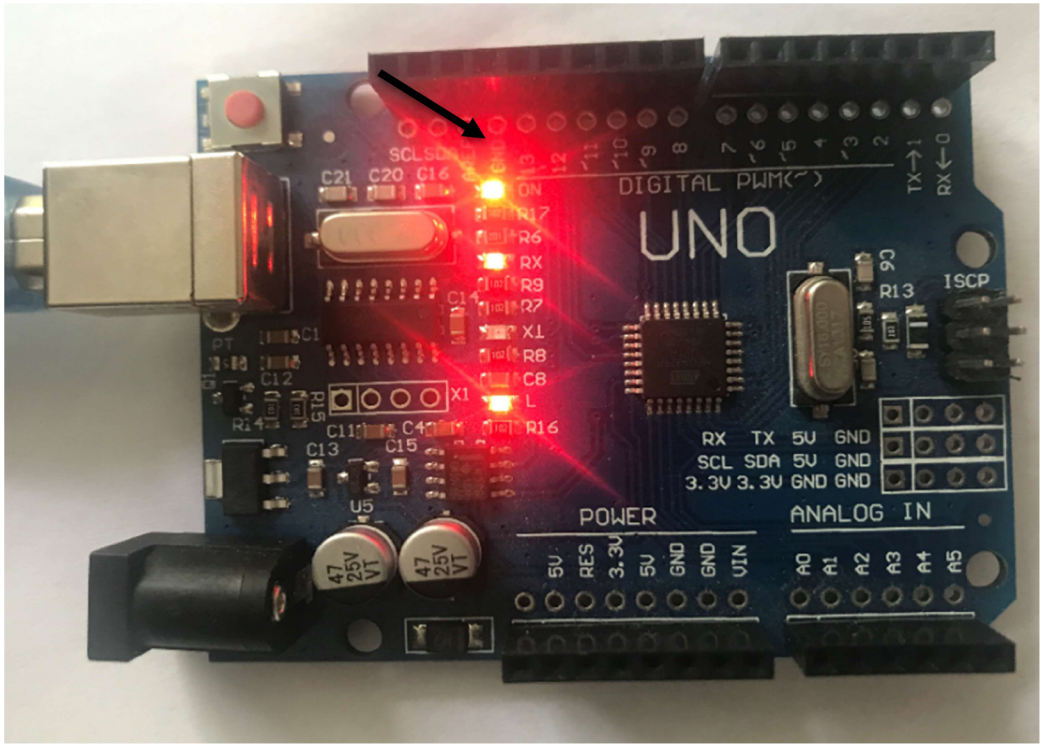
Pro programování Arduino UNO R3 lze použít Arduino IDE, které je volně ke stažitelní bez poplatků nebo Arduino Web Editor, který je zpoplatněn. Nevýhodou Arduino IDE je nutnost stažení a následovná instalace, kdežto Arduino Web Editor není nutné instalovat, jen se registrovat a přihlásit (přehled cen na Obrázku 12). Na základě rešerše bylo zvoleno Arduino IDE verzi 1.8.15 (dostupná na https://www.arduino.cc/en/software).


**Obrázek 12: Ceny Arduino Web Editor [16]**

### 6.1.1 Arduino IDE

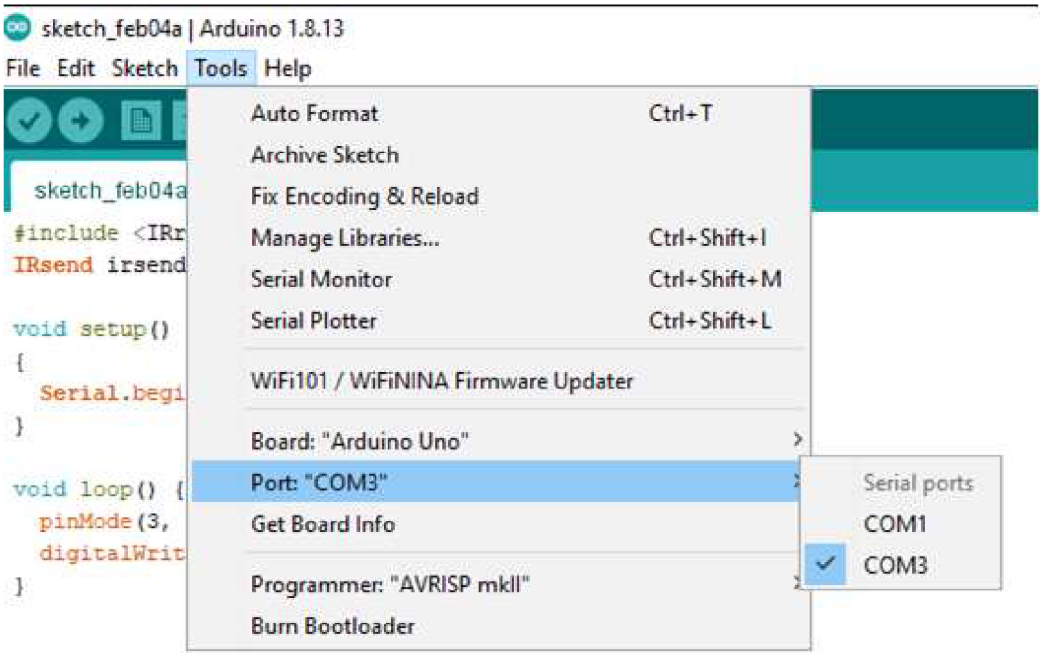
Open-source software Arduino IDE podporuje psaní kódu pro určený typ základní desky. Tento software lze použít s jakoukoli základní deskou Arduino. [17]

Nejdříve je zapojeno Arduino UNO R3 pomocí kabelu USB – USB B k počítači. Na Arduino by se měla rozsvítit červená dioda (obrázek číslo 13 níže). Červená dioda značí, že je Arduino napájeno (ukázka na Obrázku 13).




**Obrázek 13: Připojení Arduino UNO R3 (vlastní zpracování)**

Bylo třeba nastavit Arduino UNO R3 v programu Arduino IDE. K tomuto kroku bylo nutné kliknout na „Tools“ v hlavní nabídce a nastavit „Board“ a „Port“ (ukázka na Obrázku 14). V „Board“ vybereme Arduino Uno a v „Port“ COM port vašeho USB. Pokud není znám COM port, do kterého je připojeno Arduino Uno R3, po vypojení Arduino Uno R3 z USB požadovaný port zmizí. Port se opět objeví po zapojení Arduino UNO R3 do USB. Nebo COM port lze najít ve Správci zařízení.




**Obrázek 14: Propojení Arduino IDE s Arduino UNO R3 (vlastní zpracování)**

### 6.1.2 Arduino knihovny

K dešifrování IR signálu existují knihovny IRMP [18], IRLremote [19], IRLib2 [20], IRremote [21] a Minimal NEC [22]. Všechny tyto knihovny podporují Arduino UNO R3. U IRLib2 skončila podpora v září 2019. U IRLremote skončila podpora v dubnu 2018. Čili jsem se rozhodoval mezi IRMP, IRremote a Minimal NEC. Nevýhoda Minimal NEC je, že podporuje pouze NEC protokol [23]. IRMP na rozdíl od IRremote obsahuje 50 protokolů (IRremote 17 protokolů). Ale nevýhoda IRMP je složitější implementace. Po rešerši a otestování výše zmíněných knihoven bylo po multikriteriální analýze (tabulka číslo 4) zvolena knihovna IRremote, která je volně dostupná na Github (https://github.com/Arduino-IRremote/Arduino-IRremote). Knihovna podporuje dešifrování IR protokolů: Denon (používaný značkou Sharp), JVC, LG, Kaseikyo (Panasonic), Samsung, Sony, RC5, RC6, BoseWave a nejrozšířenějšího protokolu NEC.

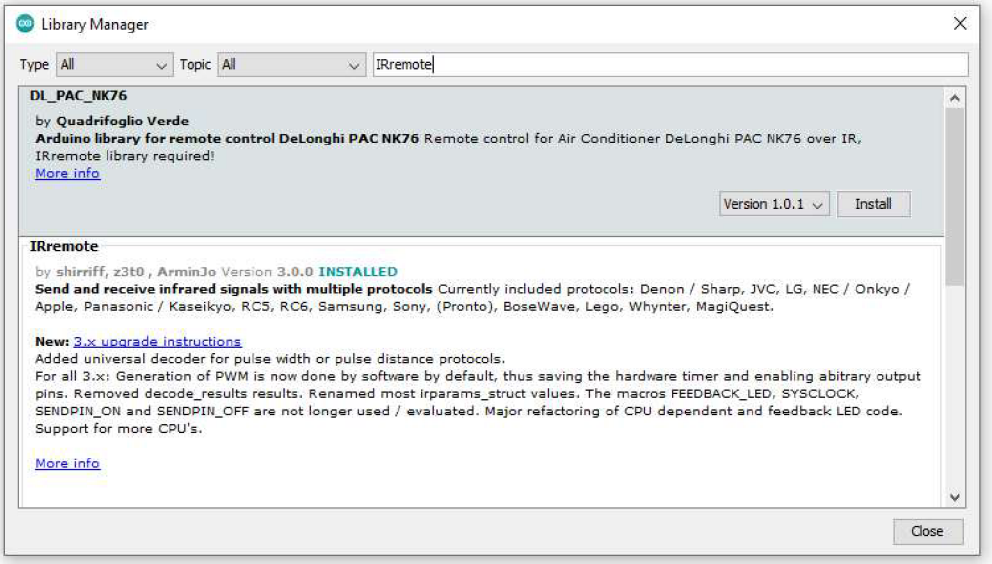
| | IRMP | IRLremote | IRLib2 | IRremote | Minimal NEC |
|---|---|---|---|---|---|
| Počet protokolů | **50** | 3 | 12 + hash* | 17+hash* | NEC |
| Typ dekódování | Za běhu | Za běhu | **RAM** | **RAM** | Za běhu |
| Typ enkódování | Za běhu | Za běhu | Za běhu | **Za běhu nebo RAM** | Za běhu |
| Knihovna je aktualizována | **ANO** | NE | NE | **ANO** | **ANO** |

**Tabulka 4: Porovnání IR knihoven [21]**

Další potřebnou knihovnou je Ethernet. Tato knihovna je navržena pro práci s Arduino Ethernet Shield, Arduino Ethernet Shield 2, Leonardo Ethernet a dalšími zařízeními založenými na W5100 / W5200 / W5500. Knihovna umožňuje desce Arduino připojení k internetu. Deska může sloužit buď v režimu server nebo klient. Knihovna podporuje až osm (W5100 a desky s <= 2 kB SRAM jsou omezeny na čtyři) souběžných připojení (příchozí, odchozí nebo kombinace příchozí/odchozí). [24]

Poslední potřebnou knihovnou je SPI (Serial Peripheral Interface), která zajišťuje komunikaci mezi Arduino UNO R3 a Ethernet Shield. SPI je synchronní sériový datový protokol používaný mikrokontrolery pro rychlou komunikaci s jedním nebo více periferními zařízeními na krátké vzdálenosti, ale může také být použit pro komunikaci mezi dvěma mikrokontrolery. [25]

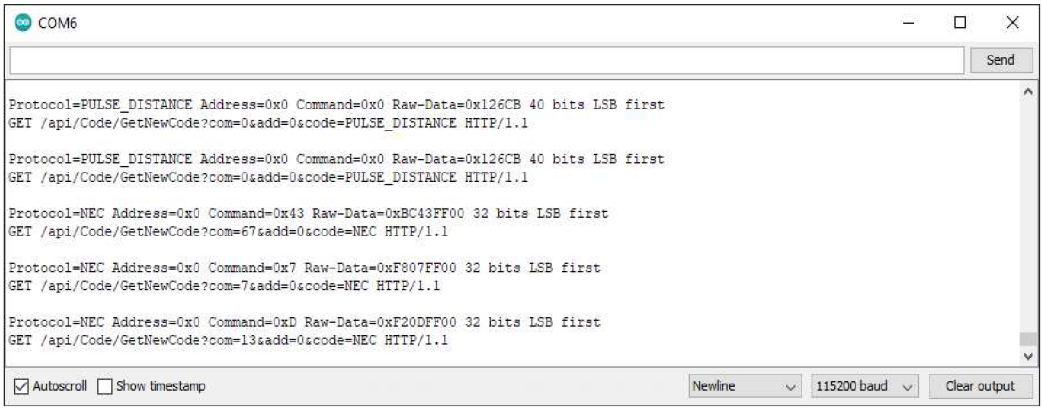
Importování knihovny do Arduino IDE je velmi jednoduché. Stačí kliknout na Sketch -> Include Library -> Manage Libraries. Po kliknutí na Manage Libraries se otevře okno vyhledávání knihoven. Do vyhledávacího okna je třeba zadat potřebnou knihovnu, ukázka (obrázek číslo 15) přidává knihovnu IRremote. Po vyhledání knihovny poté kliknout na Install. Tím je knihovna nainstalována do Arduino IDE a můžeme ji začít používat.




**Obrázek 15: Instalace knihovny IRremote (vlastní zpracování)**

### 6.1.3 Arduino přijímač

Při implementaci Arduino přijímače se objevil jeden hlavní nedostatek. Arduino Ethernet Shield totiž neumožňuje komunikaci se zabezpečeným HTTPS, jelikož nemá dostatečný výpočetní výkon, aby byli schopné v reálném čase vypočítávat šifrování. Při hledání vhodné náhrady bylo objeveno ESP8266 (WiFi). Bohužel ani ESP8266 není bez chyby ohledně HTTPS. I když ESP8266 podporuje HTTPS, požadavky se odesílají v řádu sekund, což eliminuje použití. Čili jsem uznal HTTPS za nevhodný pro odesílání dat z Arduino přijímače na vlastní API. Dalším pokusem bylo použití TCP komunikace. V API jsem vytvořil službu na pozadí, která odchytávala TCP na určeném portu (TcpServer.cs z vlastního API). Při testování na stolním počítači komunikace probíhala v pořádku, ale při nahrání na domácí server nebylo možné se připojit. Z tohoto důvodu bylo rozhodnuto o poslání načteného příkazu z Arduino přijímače použít HTTP.



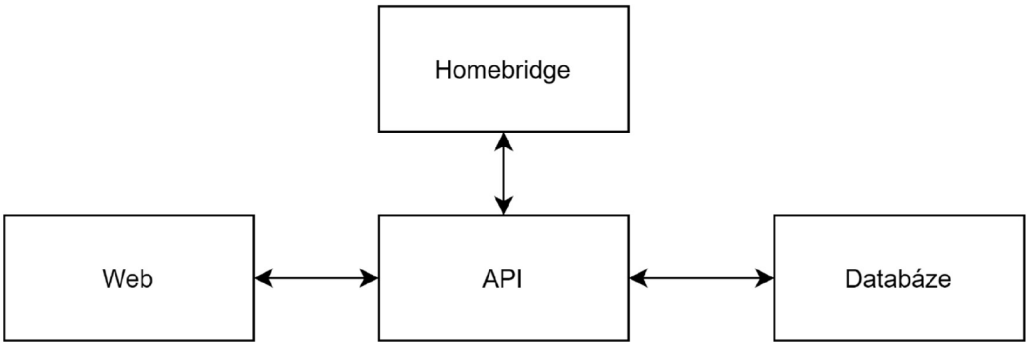

**Obrázek 16: Ukázka dekódované komunikace (vlastní zpracování)**

### 6.1.4 Arduino vysílač

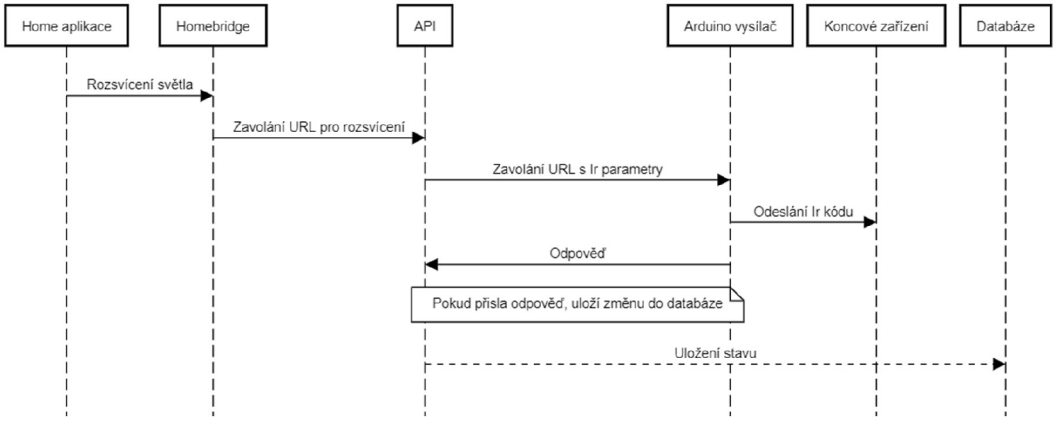
U vytváření Arduino vysílačů je nutné přesně identifikovat o jaké zařízení se jedná. A to z toho důvodu, že pokud budeme mít více vysílačů a více stejných koncových zařízení, ovládali bychom je všechny. Například, pokud bychom měli jeden vysílač v obývacím pokoji a druhý vysílač v ložnici a v obou místnostech jsme měli stejný typ světla, rozsvěceli bychom a zhasínali obě světla najednou. Toto musíme eliminovat natavením Ethernet Shield. Ethernet Shield umožňuje nastavení MAC adresy a statické IP adresy. Ošetření pomocí MAC adresy lze provést nastavením MAC adresy a poté přihlášením do DHCP serveru (u většiny domácností router), kde je nutné vyčlenit IP Adresy, které nebude router přidělovat. Dále je potřeba přiřadit MAC adresu určité vyčleněné IP adrese. Tato IP adresa je nutná zadat do administračního webu k Arduino zařízení. Ošetření IP adresou lze provést tak, že Ethernet Shield přiřadíme rovnou IP adresu a MAC adresu a poté v DHCP serveru vyčleníme tuto IP adresu, nebo rozsah IP adres. Tím Arduino bude mít statickou IP adresu, kterou nebude ovlivňovat DHCP server. V projektu byla zvolena druhá možnost (vyčlenění IP adres). Výhoda ve vyčlenění IP adres je, že se uživatel nemusí pořád přihlašovat do DHCP serveru a zadávat MAC adresy a pro ně IP adresy. Avšak nelze uvést 2 stejné IP adresy.

## 6.2 Server

Pro implementaci databáze, Homebridge, API a webu bylo použito domácí NAS (Network Attached Storage) server od značky Synology DS918+ (jedná se o starší verzi a cena nástupce DS920+ je 14 389 Kč na czc.cz bez disků). Tento NAS byl použit z toho důvodu, jelikož je již dlouhodobě používán a nemusel být tedy server zakupován. Jako vhodnější variantu, pro někoho, kdo nevlastní žádný domácí server bych doporučil Raspberry Pi 4 (cena na czc.cz 1899 Kč a k němu je třeba dokoupit SD kartu).




**Obrázek 17: Komunikační model serveru (vlastní zpracování)**




**Obrázek 18: Ukázka sekvenčního diagramu pro odeslání příkazu z aplikace Home (vlastní zpracování)**

### 6.2.1 Databáze

Další nezbytnou částí je ukládání informací pro následné zpracování. K tomuto slouží databáze, která bude uchovávat informace o přijatých IR signálech (jejich kód a typ kódování). Dále tyto IR signály budou propojeny s jednotlivými zařízeními pro jednodušší spravování. Od verze Microsoft SQL [26] (MS SQL) Server 2017 je možné instalovat do systémů používající Linux. Do této verze musela být MS SQL instalován výhradně na operačním systému Windows.

Při výběru databázového serveru bylo zvažováno mezi variantou MySQL a variantou Microsoft SQL. Obě tyto databáze lze provozovat na Linux serveru, který je nainstalován na použitém NAS serveru. Dalším požadavkem byla rychlost a administrace databáze. Zde drtivě zvítězilo Microsoft SQL (porovnání rychlostí v tabulce 5). Administrace MySQL probíhá pomocí přihlášení skrze správce, ten je nutné stáhnout zvlášť. Jako správce MySQL lze použít například český Adminer (vyvinut jedním z nejznámějších českých PHP programátorů Jakubem Vránou) nebo celosvětově nejrozšířenějším phpMyAdmin. Oba tyto správce jsou napsány v PHP a tudíž není nutné instalovat. Avšak je zapotřebí nainstalovat PHP a ještě Apache nebo NGINX. Poté se Adminer nebo phpMyAdmin může spustit ve webovém prohlížeči. Pro MS SQL administrace probíhá ve velice uživatelsky přívětivém desktopovém programu SQL Server Management Studio (SSMS). Hlavní volbou MS SQL byla plná integrace do programu Visual Studio a především do .NET. Pro správné fungování na Synology NAS serveru (dále jen server) je nutné nainstalovat Docker. V průběhu psaní diplomové práce se přesunul balíček s MSSQL ze standardního Docker registru do MCR (Microsoft Container Registry), kde lze nalézt všechny publikované image pro Docker od společnosti Microsoft. Pro instalaci image MSSQL v Docker je nutné přidat do vyhledávacích registrů Docker tuto URL (http://mcr.microsoft.com).

| Scénář | MySQL | MS SQL | Postgre SQL |
|---|---|---|---|
| Hledání 2 souvisejících tabulek | 114,6 ms | 47,7 ms | 262,7 ms |
| Hledání 4 souvisejících tabulek | 602,8 ms | 39,9 ms | 264,7 ms |
| Hledání s pod dotazem | 285.6 ms | 34,3 ms | 302 ms |
| Hledání s pod dotazem obsahující tabulky | 480,8 ms | 76,1 ms | 306,3 ms |
| Hledání s jiným typem připojení tabulky | 773,7 ms | 27,5 ms | 305,4 ms |
| Hledání pomocí příkazu „is null“ | 35,1 ms | 19,2 ms | 304,9 ms |

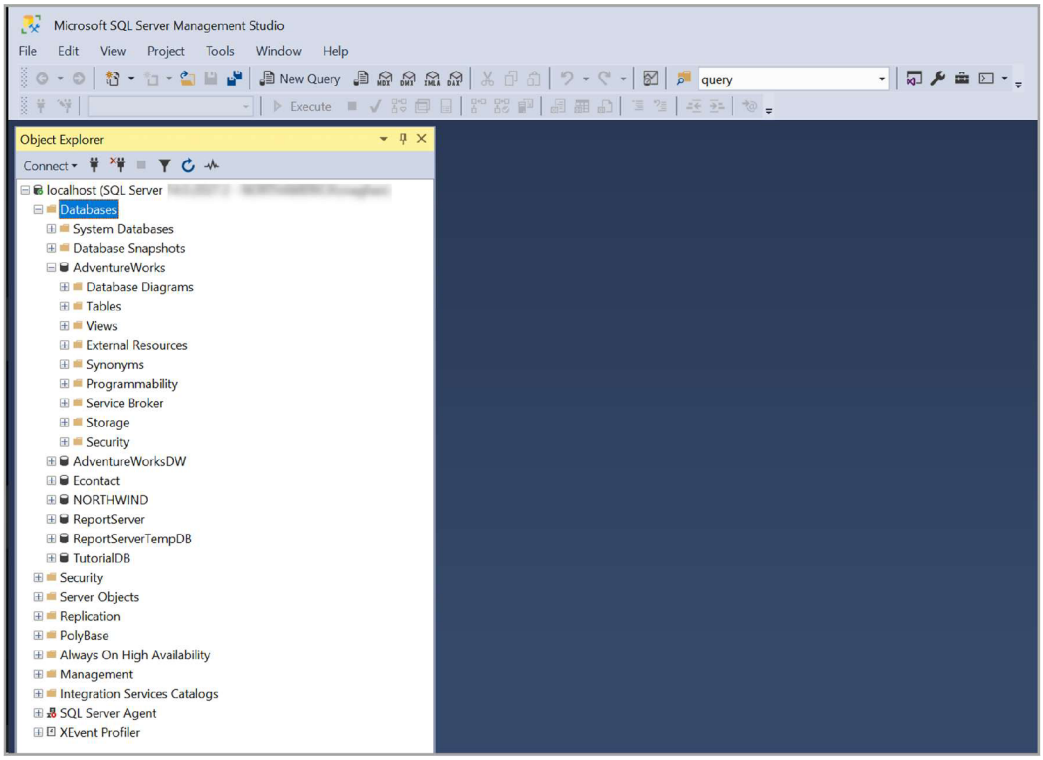
**Tabulka 5: Porovnání rychlosti – Výzkum souvisejících tabulek prováděný přímo z databáze [27]**

#### 6.2.1.1 Microsoft SQL (MS SQL)

„Pod označením SQL Server se neskrývá pouze samotný databázový server, ale komplexní moderní, výkonná, spolehlivá a bezpečná serverová platforma pro ukládání a správu dat v databázi, datových skladů a balíky nástrojů pro Business Intelligence (BI) včetně pokročilého reportování.“ [28]

Pro vlastní řešení bylo rozhodnuto použít nejnovější MS SQL které je označeno jako MS SQL 2019. Toto řešení je dostupné v licencích Express, Web, Developer, Standard a Enterprise. Placené licence jsou Standard (cena 900$ na server cca 19 500Kč) a Enterprise (cena 13 748 cca 300 000 Kč). Pro řešení diplomové práce byla použita verze Express.

Pro uživatelsky přívětivé ovládání je potřeba doinstalovat SSMS (SQL Server Management Studio). Po nainstalování SSMS a přihlášení se do MS SQL lze nastavovat všechny dostupné funkce nainstalované verze.



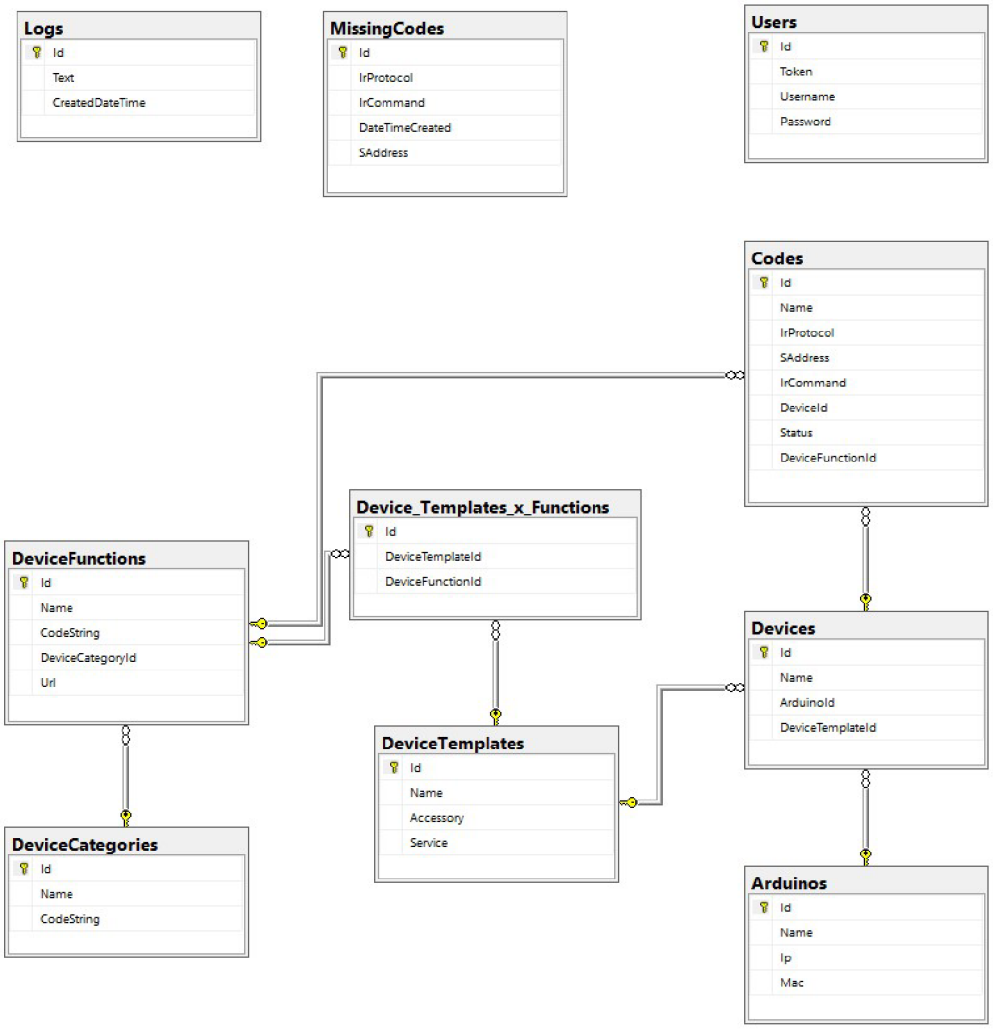

**Obrázek 19: Ukázka SSMS [29]**

### 6.2.1.2 Návrh databáze

Pro vlastní řešení bylo zapotřebí uchovávat informace o přijatých kódech z Arduino přijímače (tabulka „MissingCodes“). Tyto kódy po přiřazení skrze uživatelské rozhraní se přimigrují do tabulky „Codes“, kde bude muset uživatel přidat ke kódu pojmenování pro lepší orientaci a přiřadit k zařízení „Devices“. Tabulka „Codes“ je napojena na pomocnou tabulku „DeviceFunctions“, která uchovává potřebné údaje pro generování JSON konfigurace zařízeních pro import do Homebridge. Zařízení bude mít více kódů a bude dále přiřazeno k jednomu Arduino UNO R3 (tabulka „Arduinos“). Tabulka „Device“ je dále propojena s pomocnou tabulkou „Device Templates“, jenž určuje, jaký typy kódu lze k zařízení přiřadit. Aby se například nestalo, že k osvětlení, které má pouze stavy vypnuto/zapnuto nebude možné přiřadit kód pro barvu osvětlení. Pro zabezpečení

API byla vytvořena tabulka „Users“. Ta slouží k přihlašování do uživatelské administrace. Při chybě volání API (bude podrobněji vysvětleno v kapitole 6.2.2) se chybová hláška zapisuje do tabulky „Logs“.

Data v tabulkách „Device_Templates_x_Functions“, „DeviceTemplates“, „DeviceFunctions“ a „DeviceCategories“ jsou pevně definovány a obsahují pouze ovládání osvětlení. Při definování dalších zařízení je třeba nastavit všechny tyto tabulky.




**Obrázek 20: Entity-relationship model návrhu databáze (vlastní zpracování)**

### 6.2.2 API

Pod pojmem API (Application Programming Interface) se v softwarovém inženýrství rozumí souboru knihoven, funkcí, protokolů a tříd pro rozhraní aplikací. API je vytvořeno ve frameworku ASP.NET 5. Hlavním důvodem byl Microsoft ekosystém, který zaručuje snadný návrh databáze, API a webových stránek. Pro psaní kódu bylo využito software od firmy Microsoft Visual Studio 2019(VS) [30]. VS obsahuje kompletní sadu nástrojů pro celý cyklus programování. Lze v něm psát kód, nahrávat kód na Git, rozdělovat práci skrz Azure DevOps, nasazovat nebo objednávat hosting (bez rozšíření pouze na Azure a s rozšířením AWS Toolkit i na cloudové služby Amazon), spravovat databázi, testování kódu, atd.

ASP .NET 5 API může používat REST (Representational state transfer) API nebo gRPC (Remote Procedure Call). Nativní používání SOAP není podporováno. gRPC je podporováno od verze .NET Core 3.1. V současné době nahrazuje WCF (Windows Communication Foundation).

gRPC je moderní vysoce výkonný framework RPC (Remote Procedure Call) s otevřeným zdrojovým kódem, který lze spustit v jakémkoli prostředí. Může efektivně propojovat služby v datových centrech a mezi nimi s podporou pro vyvažování zátěže, trasování, kontrolu stavu a ověřování. Je také použitelný koncové zařízení distribuovaných počítačů pro připojení zařízení, mobilních aplikací a prohlížečů k back-endovým službám. [31]

REST [32] je založeno na klasické komunikaci požadavek – odpověď. Ke komunikaci může být využito protokolu HTTP nebo HTTPS a metody GET, POST, PUT, PATCH nebo DELETE. REST API je zejména používáno pro webové služby a jejich komunikaci na internetu. REST API je alternativa k XML (Extensible Markup Language) komunikačnímu protokolu SOAP (Simple Object Access Protocol).

Pokud porovnáme výhody gRPC nad REST, tak se především jedná o rychlost přenosu a obousměrné streamování. gRPC používá nový protokol HTTP/2 a pro kontrakt používá formát „.proto“. Tento formát musí být definovaný na obou stranách přenosu a před přenosem je zakódován do Protobuf (optimalizovaný binární kód). Výhoda REST oproti gRPC je, že se pro komunikaci používá pomalejší http, ale je pro člověka čitelný. Z důvodu čitelnosti při debuggování byl zvolen REST.

Na základě autorovy praktické zkušenosti lze doporučit přejít na gRPC v případě pomalé funkčnosti napojení.

Generování klientského napojení na API bylo použito OpenAPI, jež je realizováno nuget balíčkem „Swashbuckle.AspNetCore“. Specifikace OpenAPI (OAS) definuje standardní jazykově agnostické rozhraní pro RESTful API, jenž umožňuje lidem i počítačům objevit a porozumět schopnostem služby bez přístupu ke zdrojovému kódu, dokumentaci nebo prostřednictvím kontroly síťového provozu. Když je správně definován, může uživatel porozumět vzdálené službě a komunikovat s ní s minimálním množstvím implementační logiky. [33] Ve zkratce se jedná o definování rozhraní mezi klientem a API. Balíček „Swashbuckle.AspNetCore“ obsahuje grafické rozhraní, kde je možné API přímo testovat, ale především generuje JSON, který definuje rozhraní mezi klientem a API. Z tohoto JSON rozhraní je možné vygenerovat jak klientskou část, tak API část. Pro toto generování lze použít mnoho nástrojů jako je třeba: Autorest, NSwag, Swagger a OpenAPI. V projektu je použit Autorest verze 2.0.4283. V diplomové práci bylo nejdříve napsáno API a poté byla generována pomocí Autorest klientská část. Nevýhodou generování je, že klientská část vytvoří vlastní modely objektů a tak není možné sdílet modely mezi klientem a API. Dále není možné měnit token po inicializaci klientského API. Z těchto dvou důvodů bylo rozhodnuto vytvořit další projekt (ApiNamespaceGenerator). Ten odstraňuje tyto výše uvedené nedostatky. Jedná se o konzolovou .NET aplikaci. V první řadě bylo nutné udělat v API vlastní schéma pro generování, kde bylo definováno, aby v názvu proměnné byla nahrazena tečka za „AAAIIIIIAAA“. Po spuštění ApiNamespaceGenerator se „AAAIIIIIAAA“ nahradí zpátky na tečky a přepíše se kromě dalších věcí namespace na sdílený projekt (IrController.Model).

```
[ApiController]
    [Route("api/[controller]/[action]")]
    public class UserController : ControllerBase, IUserRepository
    {
        private readonly IUserRepository _userRepository;
        public UserController(IUserRepository userRepository)
        {
            _userRepository = userRepository;
        }

        [HttpPost]
        public LogedUserModel ChangePassword(NewPasswordModel changePassword)
        {
            return _userRepository.ChangePassword(changePassword);
        }

        [HttpPost]
        public LogedUserModel Login(LoginModel loginModel)
        {
            return _userRepository.Login(loginModel);
        }

        [HttpGet]
        public bool LogOut()
        {
            return _userRepository.LogOut();
        }
    }
```

**Obrázek 21: Ukázka API kontroléru (vlastní zpracování)**

Routování v API je realizováno použitím defaultního middleware „EndpointRoutingApplicationBuilderExtensions“ a následovným definováním v každém kontroléru. Ten je lze nalézt v anotaci „Route ()“ nad definováním třídy. Například definice [Route (“api/[controller]/[action]”)] říká, že URL cesta k tomuto API bude “adresa serveru/ api / jméno kontrolérů / metoda“ čili pro tento kontrolér a metodu „ChangePassword“ bude adresa /api/User/ChangePassword. Dalším důležitou anotací je anotace nad jednotlivými metodami, která definuje, o jakou HTTP metodu se jedná. V ASP .NET 5 jsou obsaženy všechny http metody, což jsou:

- POST – používá se pro posílání dat na server. Data jsou obsažena v těle dotazu. Je vhodné použít pro velké objemy dat.
- GET – používá se pro získání dat ze serveru. Dotaz je přidán do URL. URL je omezena na maximální počet znaků, který je zhruba okolo

2000 znaků. [34] Čili při posílání větších objektů není doporučeno používat GET.

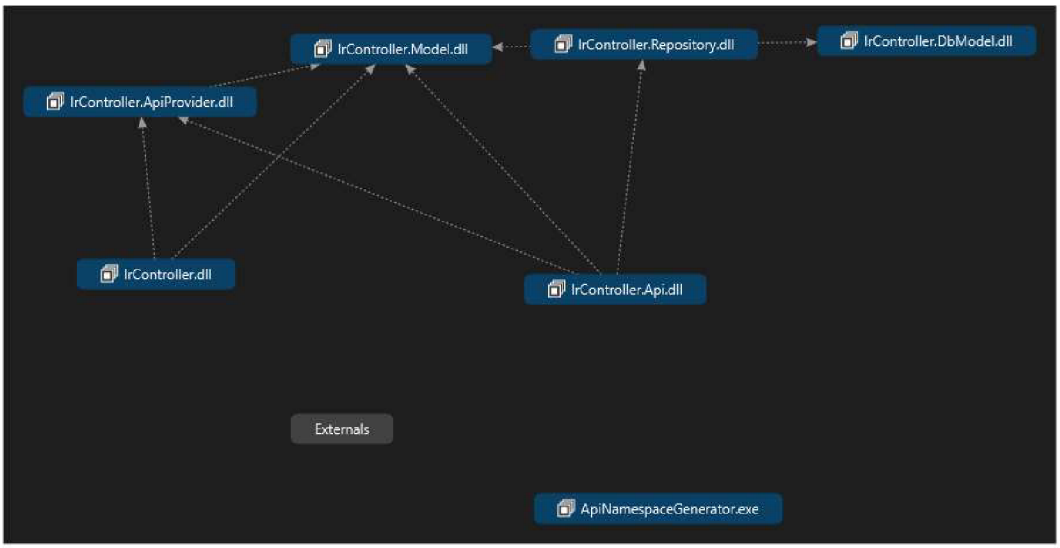
- PUT a PATCH – měla by se používat pro aktualizování a nahrazování prvků.
- DELETE – měla by se používat pro mazání prvků.

V praxi se ale z HTTP metod používají jen GET a POST, jelikož stačí na předání všech druhů informací. Kdežto ostatní HTTP metody (PUT, PATCH a DELETE) nejsou standartně implementovány do Microsoft IIS [35], je nutné přidat rozšíření (.NET Extensibility) a definovat je. Ke komunikaci s Homebridge bylo použito HTTP metody GET, jelikož se jednalo o malé přenosy informací.

#### 6.2.2.1 Struktura API

API bylo rozděleno do 6 projektů (reference projektů na Obrázku 22). Projekt je rozdělen z důvodu znovupoužitelnosti a zapouzdření kódu.

- ApiNamespaceGenerator – upravuje vygenerované klientské API rozhraní (IrController.ApiProvider)
- IrController.Api – definování API ( z tohoto projektu je generován Json s OpenApi specifikací)
- IrController.ApiProvider – knihovna rozhraní pro webové administrační stránky
- IrController.DbModel – databázový model s napojením na databázi
- IrController.Model – sdílený model mezi webovými administračními stránky a API
- IrController.Repository – business logika, která se stará o čtení a ukládání dat do databáze. Dále obsahuje tvorbu generování zdrojového kódu pro Arduino a Homebridge.

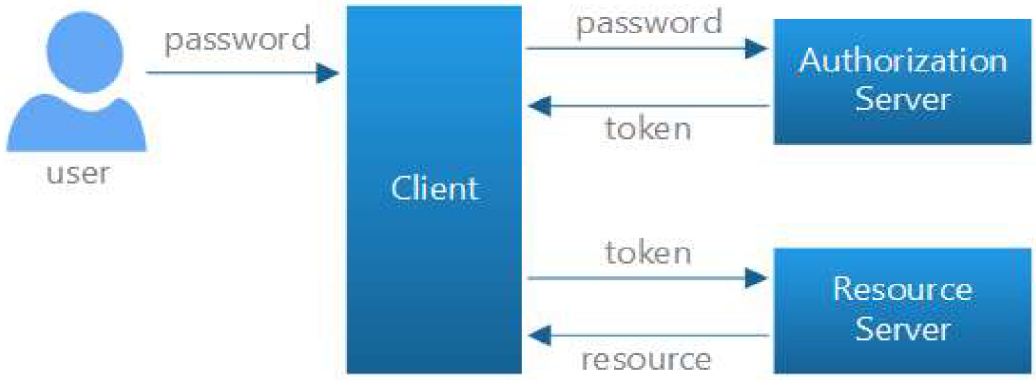



**Obrázek 22: Zobrazení referencí v projektu z Visual Studio 2019 (vlastní zpracování)**

### 6.2.2.2 Zabezpečení

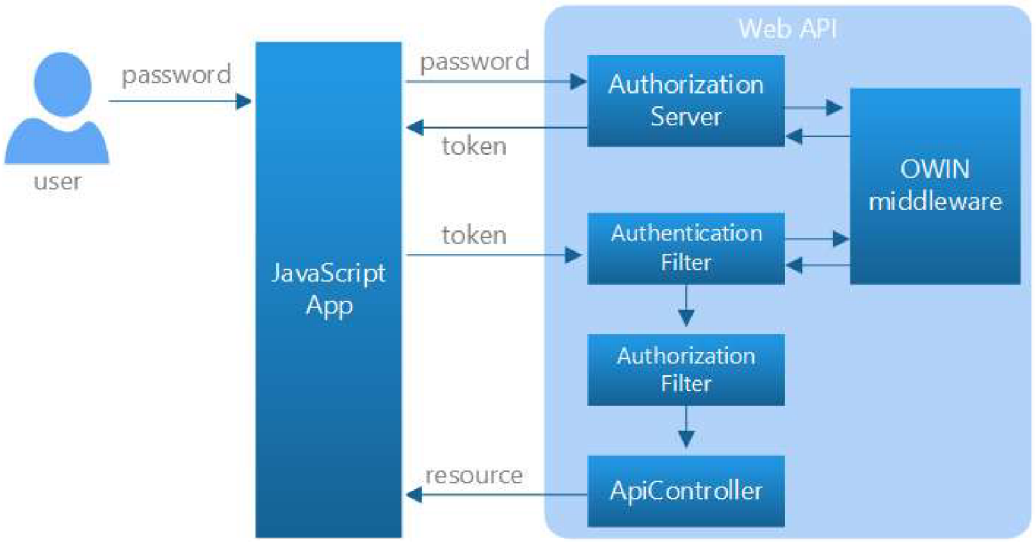
I přesto, že API bude nasazeno na NAS v privátní domácí síti bylo rozhodnuto zvýšení zabezpečení proti nevyžádanému vniknutí. Pro zabezpečení API se nabízejí následující možnosti: API Token nebo OAuth2 protokol.

OAuth 2.0 je průmyslový standardní protokol pro autorizaci. OAuth 2.0 se zaměřuje na jednoduchost vývojářů klientů a zároveň poskytuje specifické autorizační toky pro webové aplikace, desktopové aplikace, mobilní telefony a zařízení v obývacím pokoji. Tato specifikace a její rozšíření jsou vyvíjeny v rámci pracovní skupiny IETF OAuth [36]. [37] V podstatě pro autorizaci používá tzv. autorizační server (ukázka komunikace na obrázku 23). Uživatel zažádá tento autorizační server o token, poté zadá potřebné údaje a při autorizovaném přístupu tento autorizační server vytvoří uživateli token a odešle mu jej. Dále uživatel už může přistupovat tímto tokenem k API. Výhodou toho řešení je, že API neuchovává hesla čili je uživatel chráněn proti ukradnutí hesel.




**Obrázek 23: Ukázka autorizačního serveru [38]**

Zabezpečení pomocí API tokenu používá pouze API. Uživatel zadá heslo a uživatelské jméno, které se odešle na API a z té je provedena autorizace. Při úspěšné autorizaci je uživateli vrácen vygenerovaný token. Tento token uživatel používá pro další volání API (ukázka komunikace na Obrázku 24). Autentizace je prováděna na straně API. U tohoto řešení je nutné uložit heslo na straně serveru.




**Obrázek 24: Ukázka vytvoření API tokenu [38]**

OAuth 2 je bezpečnější z uživatelského hlediska. Ale pro použití chytré domácnosti je bohužel nepraktické. Hlavním nedostatkem je, že pokud se přeruší internetové spojení uživatel nebude moci ovládat koncové zařízení. Tento problém

by mohl být vyřešen implementací OAuth serveru v domácí síti. To je ale velice neefektivní, jelikož by tímto způsobem stejně hrozilo přečtení uživatelských hesel.

Do API je tedy implementováno zabezpečení pomocí API tokenů. V API je povolen přístup na „api/User/Login“, kde proběhne autorizace. Při úspěšné autorizaci je uživateli vygenerován nový token. Tímto tokenem se uživatel může dotazovat na zbylé funkce API. Autorizace je prováděna tzv. middlewarem [39]. Autorizační middleware načte hlavičku „Authorization“ a načtenou hodnotu se pokusí vyhledat v databázi v tabulce „Users“, pokud ji najde dovolí přístup. Pokud ne, vrátí chybový požadavek 401 (neautorizovaný uživatel). Jelikož je token posílán v hlavičce je vhodné použití HTTPS [40] (Hypertext Transfer Protokol Secured) na rozdíl HTTP [41] (Hypertext Transfer Protokol). Oba tyto protokoly slouží k přenosu dat mezi zařízeními a jsou bez stavové. HTTPS je novější protokol a používá šifrování SSL nebo TSL. Pro tento druh šifrování je nezbytné vlastnit certifikát, ten je podepsán certifikační autoritou, která zaručuje pravost certifikátu (RapidSSL, PositiveSSL, DigiCert, atd.). Certifikát lze i vygenerovat vlastní. Pokud je vygenerován vlastní certifikát, tak prohlížeč není schopen určit, jestli se jedná o ověřený a zobrazuje varování, že certifikát není důvěryhodný. To ale v našem případě nevadí. Hlavním důvodem pro použití HTTS a certifikátu je totiž ochrana proti tzv. Man in the Middle [42].

#### 6.2.2.3 Logování chyb

Při vývoji docházelo k nevalidnímu volání API, proto bylo přidáno logování chyb. Toto logování je automaticky prováděno přes middlewarem, pokud dojde k chybě při autorizovaném požadavku. Příkladem je třeba, pokud se na API budeme snažit dotázat na koncové zařízení, které neexistuje. Tento problém bychom mohli řešit ověřením, zda koncové zařízení existuje a pokud ne, vrátit například prázdný objekt. To by ale způsobovalo těžko odhalitelné chyby v komunikaci mezi API – Arduino UNO R3 a API – Homebridge.

Pro odhalení chyb na API lze ověřit log chyb v databázi (tabulka „Logs“). Do této tabulky jsou zapisovány všechny výjimky, které byli vyvolány voláním API. Výjimky jsou odchytávány middlewarem nazvaným „ExceptionMiddleware“. Tento middleware je dále ukládá do databáze i s volaným požadavkem.

```
|URL:/api/Code/ManageCode
BODY:{
  "Id": 2,
  "Name": "Test",
  "IrProtocol": "NEC",
  "Saddress": "156156156",
  "IrCommand": "44165156",
  "DeviceId": 1,
  "FunctionName": ""
}
System.InvalidOperationException: Sequence contains no elements
...
```

**Obrázek 25: Ukázka vyjímky uložené v DB (vlastní zpracování)**

Jak je vidět v ukázce na Obrázku 25, jsou zde všechny potřebné informace pro následovné simulování chyby. V ukázce je v prvním řádku URL, na kterou byl požadavek odeslán a obsah těla ve formátu JSON. Pokud bychom chtěli chybu replikovat, stačí si otevřít v prohlížeči API rozhraní a vložit do požadované funkce API tělo. Simulace lze samozřejmě provést i v jakémkoliv simulátoru (např. Postman [43]).

#### 6.2.2.4 Připojení do databáze

Pro připojení do databáze MS SQL byl zvažován Nuget balíček EF (Entity Framework), EFC (Entity Framework Core) a integrované řešení ADO.NET. ADO.NET používá pro dotazování ODBC čili je nutné všechny dotazy nad databází přímo v SQL a není možné použít ORM (Objektově relační mapování). Proto byl ADO.NET vyřazen z rozhodování, jelikož by musela být každá změna tabulky provedena v SSMS a poté přepsány dotazy. EF a EFC oba používají ORM i proto je možné pro dotazování nad databází používat objekty, což urychluje práci programátora a zvyšuje bezpečí úprav databáze. Výhodou ORM je rychlejší psaní kódu, ochrana před SQL Injection a při úpravě databázových modelů lze snadněji

najít chybu. Obě EF a EFC podporují dva způsoby propojení s databází CF (Code First) nebo DF (Database First). Při tvorbě CF programátor tvoří třídy, které bude chtít uchovávat v databázi a poté provede tzv. migraci ta vytvoří skript, který po provedení upraví databázové modely, tak aby odráželi vytvořené třídy. DF přistupuje propojením opačně. Vychází z databáze a v projektu se generují třídy odrážející databázový model. Jelikož se většinou při programování dříve píší třídy je tedy výhodnější použít CF. Mezi CF a DF lze libovolně přepínat, proto je výhodné, pokud již existuje databáze provést DF a poté EF nebo EFC přepnout do CF. EFC nabízí nové funkce a zároveň implementuje všechny důležité stávající funkce z EF. Dále obsahuje vylepšení algoritmu pro optimalizaci překladu dotazu do SQL jazyka. Jelikož tedy EFC je vylepšené EF, tak bylo pro implementaci použito EFC.

```
String sql = @"SELECT [Id], [Name], [IrProtocol], [SAddress], [IrCommand], [DeviceId],
collation_name FROM [dbo].[Codes]";

using (SqlCommand command = new SqlCommand(sql, connection))
{
    using (SqlDataReader reader = command.ExecuteReader())
    {
        while (reader.Read())
        {
            Console.WriteLine("{0} {1}", reader.GetString(0), reader.GetString(1));
        }
    }
}
```

**Obrázek 26: Ukázka ADO.NET (vlastní zpracování)**

```
using (IrControllerContext _db = _dbContextFactory.CreateDbContext())
            {
                return _db.Codes.Select(x => new CodeModel {
                    Id = x.Id,
                    Name = x.Name,
                    IrCommand = x.IrCommand,
                    IrProtocol = x.IrProtocol,
                    Saddress = x.SAddress,
                    DeviceId = x.DeviceId,
                    DeviceName = x.Device.Name
                }).ToList();
            }}
```

**Obrázek 27: Ukázka stejného kódu (obrázek 26) v EF a EFC (vlastní zpracování)**

Samotné API nemá žádnou referenci na databázový model. Připojení do databáze je realizováno pomocí projektu „IrController.Repository“ (dále jen repositáře). Toto bylo provedeno z důvodu zapouzdření databáze. Repositáře jsou

načteny do API nuget balíčkem „Scruptor“. Scruptor prohledá všechny třídy zadané v projektu a nahraje všechny tyto třídy, mající určitou anotaci (v našem případě „LoadRepositoryAttribute“. Tyto třídy jsou implementovány jako Singleton, Scoped nebo Transient a lze je použít v jednotlivých kontrolérech pomocí Dependency Injection. Jelikož se jedná o repositáře, které mají přístup do databáze je vhodné je nastavit jako Singleton, a to z toho důvodu že EFC má omezení pool přístupů k databázi. Tento pool přístupů k databázi definuje maximální počet souběžných připojení do databáze. Pokud je plný a vytvoříme nové připojení do databáze, toto nové připojení do databáze bude muset čekat, dokud se nějaké připojení neuvolní.

- Singleton – Instance je vytvořena při prvním dotazu na tuto instanci. Při každým dalším dotazu je zvonu použita tato instance.
- Transient – Takzvaně přechodná instance. Po každém zavolání je vytvořena nová instance ať už se jedná o jeden HTTP požadavek nebo různý HTTP požadavek.
- Scoped – Pro jeden HTTP požadavek je vytvořena jedna instance.

### 6.2.3 Klientské rozhraní

Klientské rozhraní slouží pro uživatelské nastavování Arduino zařízení a spravování odposlechnutých IR kódů. Při výběru technologie, ve které bude napsané klientské rozhraní, bylo rozhodnuto využít opět řešení od Microsoftu a to open source [44] platformu ASP.NET. ASP.NET umožňuje vytvářet moderní webové stránky, aplikace a služby nad .NET frameworkem. Od frameworku .NET Core (rok vydání 2016) je dokonce multiplatformní (lze použít se všemi operačními systémy). V podstatě lze říci, že ASP.NET rozšiřuje možnosti použití frameworku .NET o základní zpracování webových požadavků, syntaxi webových stránek (Razor [45]), knihovnu pro webové vzory MVC [46] (Model – View – Controller), základní registraci a přihlašování uživatelů, rozšíření editoru v programu Visual Studio o zvýraznění syntaxe a doplnění kódu. Programovat ASP.NET lze ve 3 programovacích jazycích (C#, F# nebo Visual Basic). Dále si může programátor v ASP.NET vybrat ze dvou technologií, které jsou určeny pro webové aplikace a to v ASP.NET Core MVC a Blazor.

Model architektury MVC (Model-View-Controller) odděluje aplikaci do tří hlavních skupin komponent: modely, zobrazení a kontrolery. Tento model pomáhá dosáhnout oddělení logiky programu. Pomocí tohoto modelu se požadavky uživatelů směrují do kontroleru, který zodpovídá za práci s modelem a provádění uživatelských akcí nebo různé pomocné výpočty. Kontroler zvolí zobrazení, které bude zobrazeno uživateli a poskytne uživateli potřebné data, která jsou vyžadována. [47]

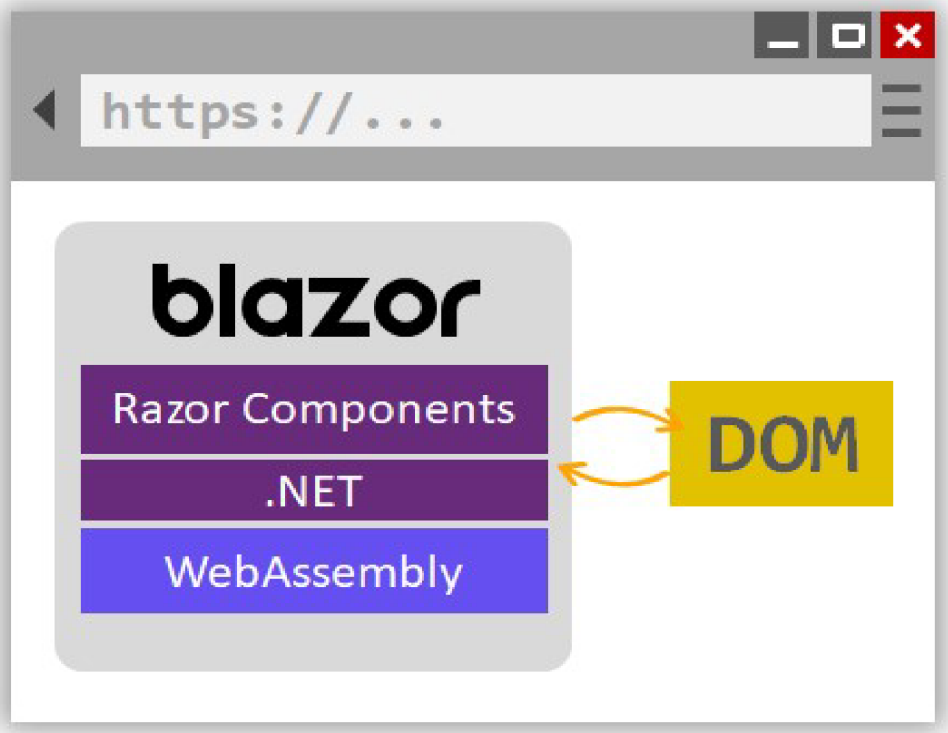
- Model - v MVC architektuře obsahuje jakoukoliv business logiku, operace nad databází a oprerace potřebné pro kontroler.
- Zobrazení – tato vrstva se stará o vykreslení uživatelského rozhraní. Pregenerovává uživatelský kód na HTML kód. V ASP. NET Core MVC se nejčastěji používá syntaxe Razor.
- Kontroler – obluhuje interakci uživatele a je meziprvkem mezi modelem a zobrazením. Mimo jiné se stará i o routování.

Blazor framework je reakce od Microsoft na vytváření jednostránkových aplikací, ale na rozdíl od většiny jenodstránkových aplikací nepoužívá pro klientskou iterakci JavaScript [48] ale WebAssembly.

WebAssembly je formát binární instrukce pro virtuální stroj založený na zásobníku. Wasm je navržen jako cíl přenosné kompilace pro programovací jazyky, což umožňuje nasazení na webu pro klientské a serverové aplikace. [49]

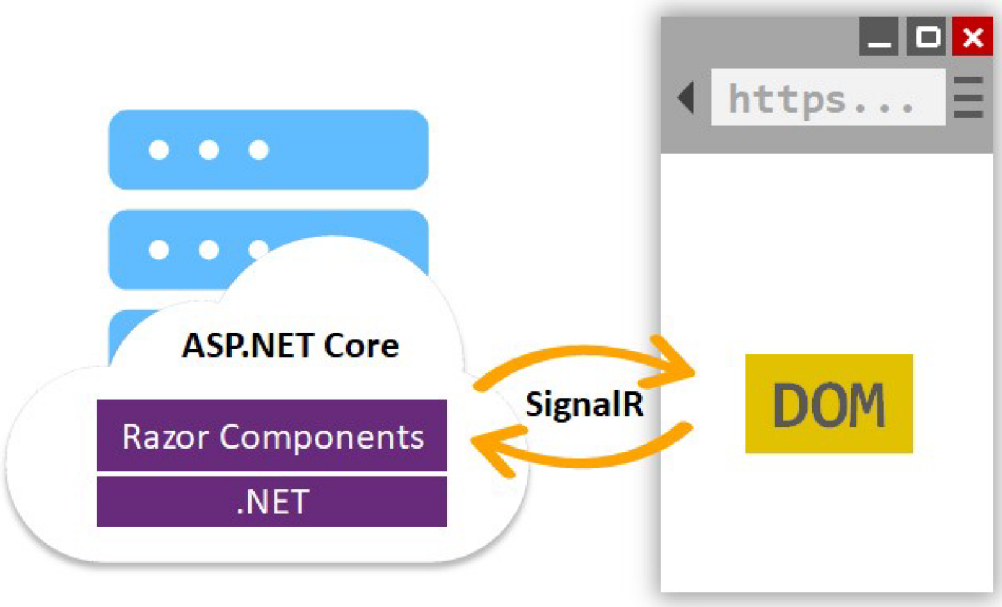
Blazor je dostupný od ASP .NET Core 3 verze uveřejněné v roce 2018. Slovo Blazor vzniklo spojením slov Browser (prohlížeč) a Razor (syntaxe pro psaní zobrazení). Blazor je možné hostovat jak na straně serveru, tak ve webovém prohlížeči.

Hostování na straně klienta v prohlížeči (Blazor WebAssembly – ukázka struktury na Obrázku 28) probíhá tak, že při prvním přístupu se stáhnou do prohlížeče celé stránky s potřebnými balíčky a následně probíhá jen výměna dat s API.




**Obrázek 28: Struktura WebAssembly [50]**

Druhým typem hostování Blazor je na straně serveru (Blazor Server – ukázka struktury na Obrázku 29). „Blazor odpojí logiku vykreslování komponenty od způsobu použití aktualizací uživatelského rozhraní. Blazor Server poskytuje podporu pro hostování Razor komponent na serveru aplikace v ASP.NET Core. Aktualizace uživatelského rozhraní se zpracovávají přes SignalR připojení. „[50]




**Obrázek 29: Struktura Blazor Server [50]**

Blazor má obrovskou výhodu oproti ASP.NET Core MVC a to je eliminace JavaScriptu a použití WebAssembly z internetového prohlížeče. Uživatelská interakce lze přímo napsat v jazyce C# přímo v Razor. Jedinou nevýhodou WebAssembly je nepodporování některých starších prohlížečů. A však ani tento nedostatek není tolik zásadní, aby znemožnil efektivní použití. Takže jeho výhody jednoznačně přesahují. A mezi hlavní výhody patří ladění ve Visual Studio, kde je samozřejmostí používání breakpoint a krokování na rozdíl od Javascriptu. Bylo rozhodováno mezi Blazor Server a Blazor WebAssembly. Blazor Server vyšel jako součást LTS (Long-term support) release (.NET Core 3.1 LTS s podporou do prosince 2022), ale Blazor WebAssembly se zatím nedočkal LTS release, proto jsem zvolil Blazor Server.

### 6.2.3.1 Blazor Server

Při prvním otevření nového projektu se jeví Blazor Server podobný ASP.NET Core MVC, ale při bližším pohledu si lze všimnout, že se všechna logika MVC přesunula do stránek psaných v Razor. Pro routování se používá syntaxe „@page“ která je také obsažena v Razor souboru. Razor kód (Obrázek 30) pro Blazor Server lze v podstatě rozdělit do 3 částí. V první části se definují namespace (jmenný prostor), dependency injection (vkládání závislostí), routování a další definiční informace. Pokud chceme určitý namespace, anotaci nebo interface implementovat do všech Razor souborů, je to možné udělat uvedením požadovaného namespace, anotace nebo interface do „_Imports“. V druhé časti je obsaženo vykreslování HTML kódu a ostatních pomocných funkcí pro vykreslování. A v třetí části je obsluha životního cyklu Razor souboru a pomocné funkce, dalo by se říci kontrolér z MVC pohledu.

```
@page "/Code/ShowAllCodes/{deviceId:int}"

@inject NavBarLeftInjectableMenu menu
@inject IJSRuntime JsRuntime;
@using IrController.Model;

<Table>
    <TableHead>
        <TableHeadCell>Code name</TableHeadCell>
        <TableHeadCell></TableHeadCell>
    </TableHead>
    <TableBody>
        @{ foreach (CodeModel item in codes)
            {
                <TableRow>
                    <TableCell>@item.Name</TableCell>
                    <TableCell>
                        <a  href="/Code/EditCode/@item.Id">Edit</a> |
                        <a  @onclick="@(e => DeleteCode(item.Id))">Delete</a>
                    </TableCell>
                </TableRow>
            }
        }
    </TableBody>
</Table>

@code {
    [Parameter] public int deviceId { get; set; }
    List<CodeModel> codes = new List<CodeModel>();
    protected override async Task OnInitializedAsync()
    {
        if(deviceId < 1)
            codes = api.Code.GetAllCodes().ToList();
        else
            codes = api.Code.GetCoddesByDevice(deviceId).ToList();
    }
```

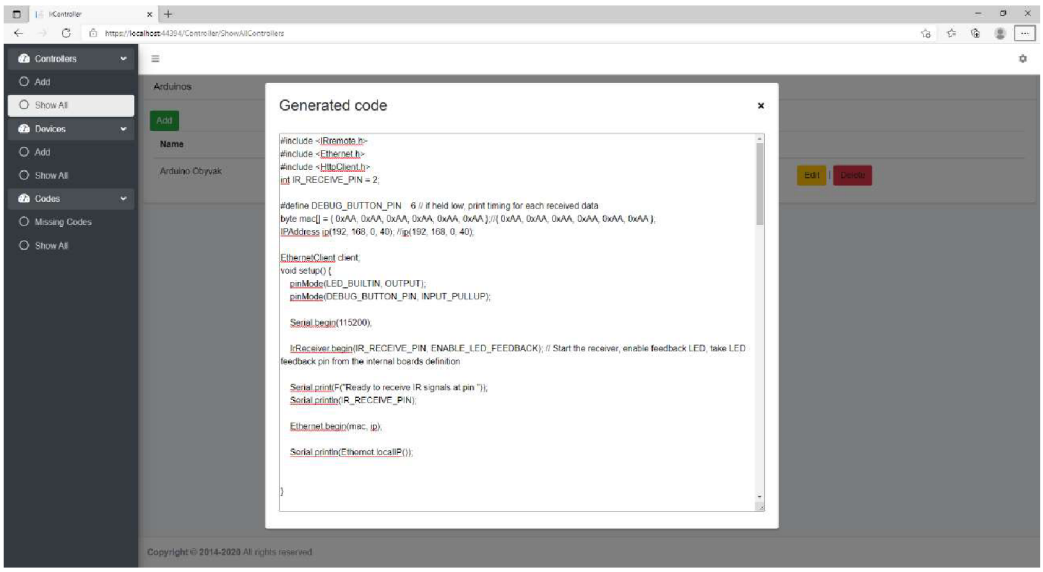
**Obrázek 30: Ukázka kódu v Razor (vlastní zpracování)**

### 6.2.3.2 Napojení na API

Připojení na API je realizováno vygenerovanou knihovnou IrController.ApiProvider, tato knihovna je vložena jako singleton při inicializaci stránek, aby bylo možné API použít ve všech Razor souborech. Tak aby bylo možné API inicializovat je v něm nastaven defaultní token, který dovoluje přistoupit pouze k API pro přihlášení, jinak by nebylo možné API použít. Po přihlášení uživatele je tento token nahrazen novým tokenem, jenž umožní přistupovat k celému API. Dále je důležité v Startup.cs nastavit URL adresu, na kterém je poskytováno API.

### 6.2.3.3 Webová aplikace

Webová aplikace slouží k administraci Arduino prvků a přiřazováním jim načtených kódů. V první řadě je nutné přidání nového Arduino zařízení, u kterého je nutné uvést MAC adresu a IP adresu. Poté je možné z administračního webu vygenerovat kód pro přijímač nebo vysílač. Následně je kód možné vložit do Arduino IDE a nahrát na Arduino. Generování kódu (ukázka na Obrázku 31) do administračního webu bylo uvedeno ve snaze zjednodušil ovládání Arduino prvků.

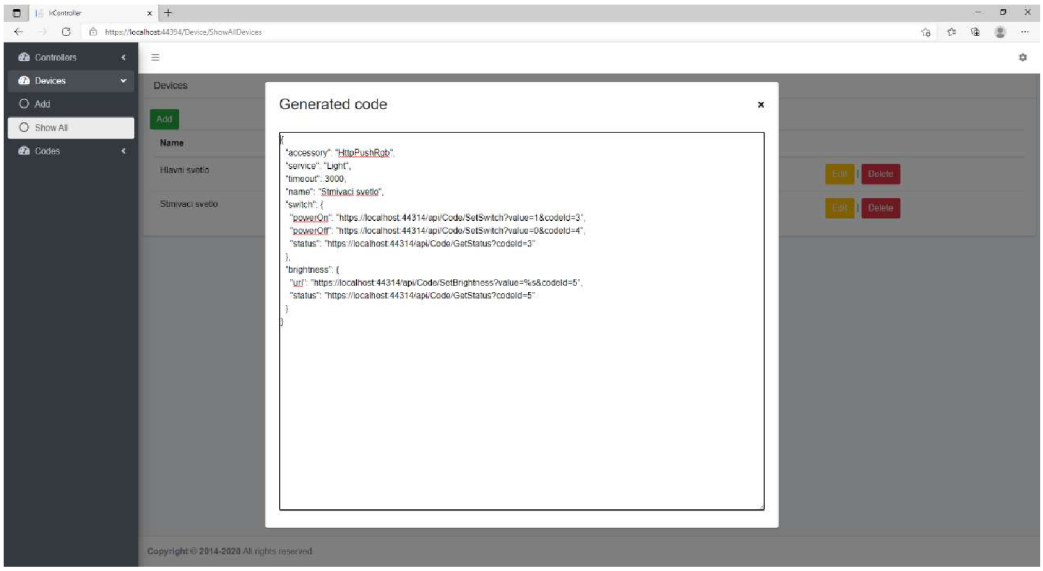



**Obrázek 31: Ukázka vygenerovaného kódu pro Arduino přijímač (vlastní zpracování)**

Po přidání Arduino prvku uživatel může vytvářet a přidávat jednotlivé zařízení k tomuto Arduino kontroléru. Jednotlivá zařízení lze vybrat ze 3 kategorií a to: „Simple Light“, které obsahuje pouze vypínání a zapínání. Další kategorií je „Brightness Light“, které obsahuje zapínání, vypínání, a navíc je přidáno ovládání jasu. Poslední kategorie je „Color Light“. Ta obsahuje vše z „Brightness Light“ a navíc je rozšířena o barvu světla. K Arduino kontroléru lze samozřejmě přiřadit více zařízení.

K jednotlivým zařízením lze přiřazovat naskenované kódy jednoduchou editací naskenovaného kódu. Po editaci stačí zadat popis naskenovaného kódu a vybrat zařízení ke kterému ho požadujete zadat. Po uložení se nepřiřazený naskenovaný kód přesune do přiřazených kódů.

Po přidání všech kódů k zařízení lze na záložce zařízeních vygeneroval JSON (ukázka na Obrázku 32) kód pro přidání zařízení do Homebridge. Zde bylo zvažováno, zda přímo neukládat všechna zařízení přímo do konfigurace Homebridge a dále Homebridge restartovat, aby byli provedeny změny. Problém by ale nastal při ruční editaci JSON v Homebridge. Z tohoto důvodu bylo rozhodnuto každé zařízení generovat zvlášť a nechat uživatele ovlivnit




**Obrázek 32: Ukázka vygenerovaného kódu pro zařízení pro Homebridge (vlastní zpracování)**

vygenerovaný JSON.

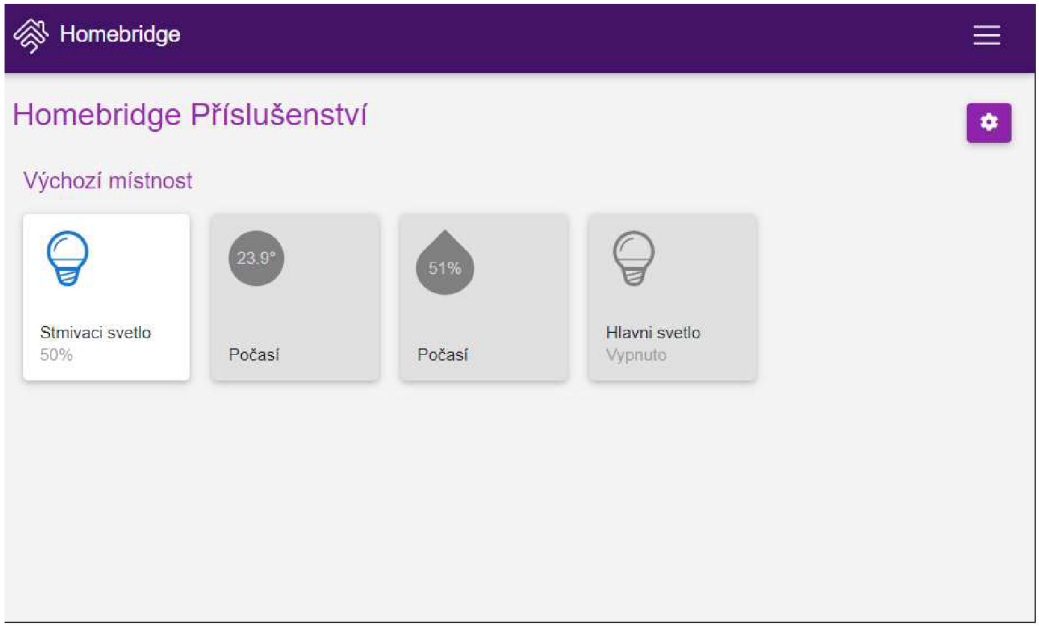
### 6.2.4 Homebridge

Homebridge po instalaci neobsahuje žádnou databázi zařízení obsažených v HAP specifikaci. Homebridge tedy implementuje pouze HAP rozhraní. Pro přidání zařízení do Homebridge se používají pluginy, které lze nainstalovat po přihlášení do administrace. V pluginech lze najít, jak implementace pro zařízení certifikovaných MFi výrobců chytrých zařízení jako je třeba Philips Hue, tak i necertifikovaných MFi výrobců jako je Xiaomi. Po nainstalování těchto pluginů lze ovládat přes aplikaci Home i zařízení, které je podporováno pouze v chytré domácnosti pro konkurenční společnosti. Za zajímavost ještě stojí zmínění některých pluginů:

- Homebridge Weather – zdarma aktuální počasí
- Homebridge Alexa – umožňuje ovládat zařízení certifikované pro Amazon Alexa
- Homebridge Mi Aqara – umožňuje ovládat zařízení Xiaomi Mi Aqara

Pro implementaci napojení na API byla potřeba pluginu, který umožňuje odesílání HTTP požadavků. Jelikož není možné v IR posílat příkazy s parametrem, jako je tomu u chytrých zařízení. Například pokud chceme u chytrého zařízení změnit jas světla, odesílá se na koncové zařízení parametr, který obsahuje jas v procentech (například /Brightness/50 pro 50 % jasu). To ale u IR není možné, jelikož by se muselo definovat 101 příkazů IR signálu. Bohužel při hledání nebyla nalezena žádná obecná knihovna, která by obsahovala více zařízení. K dispozici byly vyhledány pouze jednotlivé knihovny pro závěsy, vypínače, osvětlení, otevírání dveří, různé televize, atd. A proto bylo rozhodnuto o vytvoření ovládání koncového zařízení z aplikace Home pouze pro osvětlení, jelikož ovládání pro všechny zařízení by byla práce pro celý tým lidí. Nejvhodnější pro implementaci poskytoval plugin jménem Homebridge Http Rgb Push [51]. Tento plugin je založen na pluginu homebridge-better-http-rgb [52] a rozšiřuje ho o podporu homebridge-http-notification-server [53], který umožňuje odeslání o změně stavu místo vyžádaného

dotazu. Dále homebridge-better-http-rgb upravuje časovým limitem, aby bylo možné zabránit uvíznutí Homebridge na nereagujícím zařízení.




**Obrázek 33: Ukázka Homebridge zařízení (vlastní zpracování)**

Přidání Homebridge probíhá buď manuálně nebo automaticky naskenováním QR kódu na úvodní straně Homebridge. Manuální přidávání se provádí v aplikaci Home po kliknutí na přidat zařízení. Po kliknutí je možné načíst již zmíněný QR kód, nebo kliknutím na „Nemám kód nebo nelze naskenovat“, kde by se měl Homebridge objevit. Pokud se neobjeví, není nejspíše připojeno do stejné sítě nebo nemá otevřený port pro párování.

# 7 Testování

Testování bylo potřeba provádět jak na jednotlivých prvcích, tak mezi jednotlivými implementacemi (Arduino přijímač – API, Arduino vysílač – API, API – Homebridge, Homebridge – HomeKit. Až po otestování jednotlivých prvků a komunikací mezi nimi bylo prováděno celkové testování na koncových zařízeních. Testování bylo provedeno uživatelskými testy a nebylo použito automatizovaných testů.

Pro testování byl použit jako server NAS server Synology DiskStation DS918+ (Obrázek 34), který má procesor Intel Celeron J3455 (2,3 GHz), 4096 MB DDR3L a byl osazen dvěma disky Western Digital Red 3TB a SSD M.2 diskem (Samsung SSD 860 EVO 250 GB) pro vyrovnávací paměť. Dále na serveru byl nainstalován Docker pro kontejnerizaci Homebridge, API, Webové aplikace a MS SQL 2019 Express. Jako přijímač a vysílač bylo dále použito Arduino UNO R3 s Arduino Ethernet Shield. Pro spouštění testů byl použit dále stolní počítač s procesorem Intel Core i5-8600K (6 jader se základní frekvencí 3,6 GHz/ turbo 4,3 GHz), pamětí 16 GB DDR4 a operačním systémem Windows 10 Enterprise.

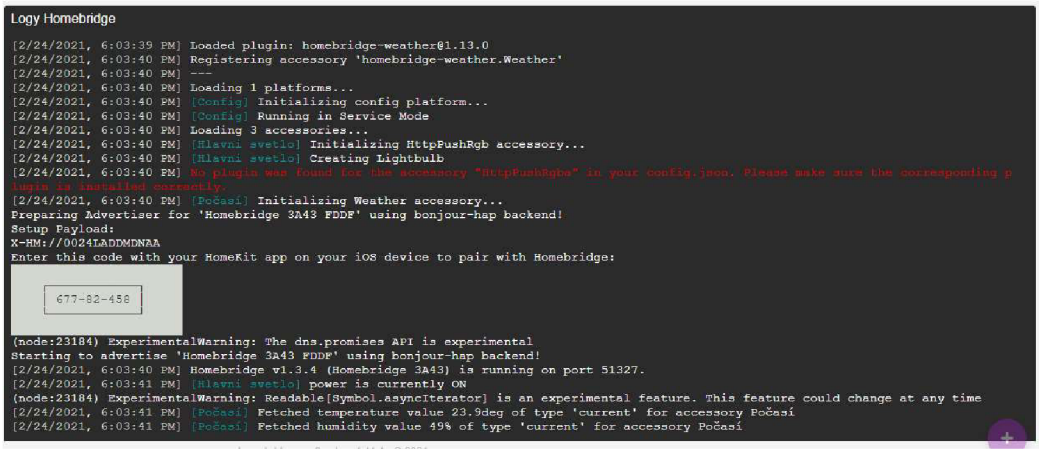

**Obrázek 34: Synology DS918+**
**(vlastní zpracování)**

## 7.1 Testování API

Při testování rychlosti API byly všechny odpovědi zpracovány v rámci desítek milisekund a nebyla objevena žádná zajímavost v rychlosti odpovědi.

## 7.2 Testování Homebridge

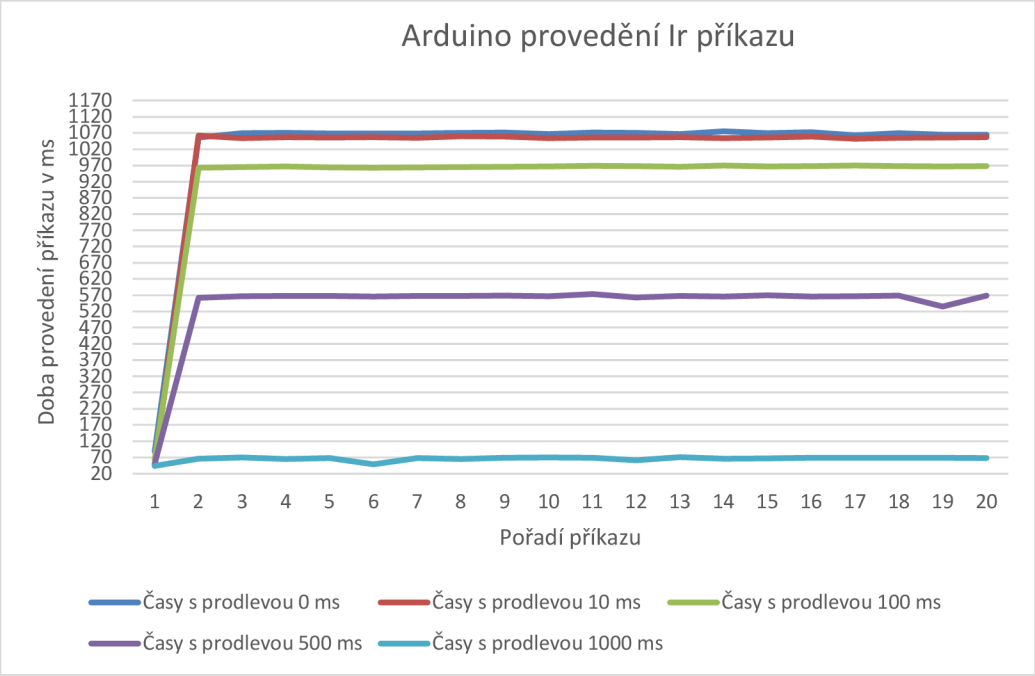
Na Homebridge bylo testováno ukládání JSON konfigurace. JSON konfigurace Homebridge brání uživateli zadat nevalidní JSON, ale dovolí uživateli v konfiguraci zadat například špatný název pluginu. Při každé úpravě je potřeba restartovat Homebridge, aby si znovu načetl konfigurační JSON. Po načtení Homebridge vypíše všechny chyby v JSON konfiguraci (ukázka výpisu chyby na Obrázeku 35). Pokud je chyba u jednoho zařízení, toto zařízení je vyjmuto z ovládání (není zobrazeno ani v aplikaci Home), ale ostatní validní zařízení nejsou ovlivněna.




**Obrázek 35: Ukázka chyby v konfiguračním JSON pro Homebridge (vlastní zpracování)**

## 7.3 Testování Arduino vysílače

U Arduino vysílače byla testována rychlost provést určitý IR příkaz. Testování bylo prováděno aplikací Postman – Collection Runner [54], kde byl vyvolán příkaz na API ze stolního počítače a byl měřen čas odpovědi Arduino vysílače, že byl IR příkaz proveden. Při odeslání požadavku a ovládání koncového zařízení byl první požadavek zpracován v rámci desítek milisekund, ale další požadavky byly zpracovány okolo jedné sekundy (Graf 36 a Tabulka 6). Pomalost dalších požadavků je zapříčiněna odesíláním IR signálu z Arduino vysílače na koncové zařízení.




**Obrázek 36: Graf měření provedení příkazu v ms (vlastní zpracování)**

| Prodleva mezi příkazy | 0 | 10 | 100 | 500 | 1000 |
|---|---|---|---|---|---|
| Průměrné časy | 1019.15 | 1006.15 | 921.4 | 540.05 | 65.55 |
| Průměrné časy s prodlevou | 1019.15 | 1016.15 | 1021.4 | 1040.05 | 1065.55 |

**Tabulka 6: Průměrné časy provedení IR příkazu (vlastní zpracování)**

Problém u odesílání IR signálu z Arduino vysílače je, že koncový prvek neodpoví zpět, že byl IR příkaz proveden. Pokud tedy není IR dioda Arduino vysílače v dosahu koncového zařízení, neprovede se Ir příkaz, ale v aplikaci Home se status změní, jako by se provedl. Další problém nastává, pokud je proveden příkaz fyzickým IR

ovladačem. Tento příkaz není zaznamenán do stavu aplikace Home, jelikož nebyla zjištěna změna stavu.

Další problém byl objeven, pokud máme například dvě stejné koncové zařízením nebo koncové zařízení, které používají stejný IR kód. Takováto zařízení jsou totiž nechtěně ovládána. Řešením je tedy vytvořením více Arduino vysílačů a namířením IR diody tak, aby nechtěně neovlivňovala ostatní zařízení, která používají stejný IR kód. Naopak, pokud chceme ovládat více zařízení, která nejsou v dosahu IR diody, je zde možnost do Arduino vysílače přidat na další digitální PIN (až 14) IR diodu.

## 7.4 Testování Arduino přijímače

Testování Arduino přijímače bylo provedeno na čtyřech IR ovladačích (Obrázek 37). Na všech čtyřech ovladačích byly úspěšně dekódovány a zavedeny kódy do databáze. Bohužel tyto ovladače používají pouze protokol NEC [23] a Pulse Distance modulace [55]. Problém nastal při testování rychlého mačkání různých tlačítek. Při rychlém mačkání různých tlačítek se správně všechny nezaznamenaly. Pokud totiž dochází k dekódování tlačítka, Arduino nezpracovává další signály IR. Je tedy potřeba po stisknutí tlačítka zhruba 2–3 sekundy počkat. Vzdálenost načítání byla testována až do 6 m a při přímém namíření (odchylka byla 30 stupňů) se nevyskytl žádný problém.

Další problém nastal při snaze ovládat všechna IR tlačítka z ovladačů. Tento úkol nebyl vůbec jednoduchý, jelikož každý IR ovladač obsahuje jiná tlačítka. Například ovladač vlevo z obrázku číslo 30 obsahuje tlačítka Mode 1-8, které přepínají rychlost blikání LED osvětlení a není je možné ovládat z aplikace Home.

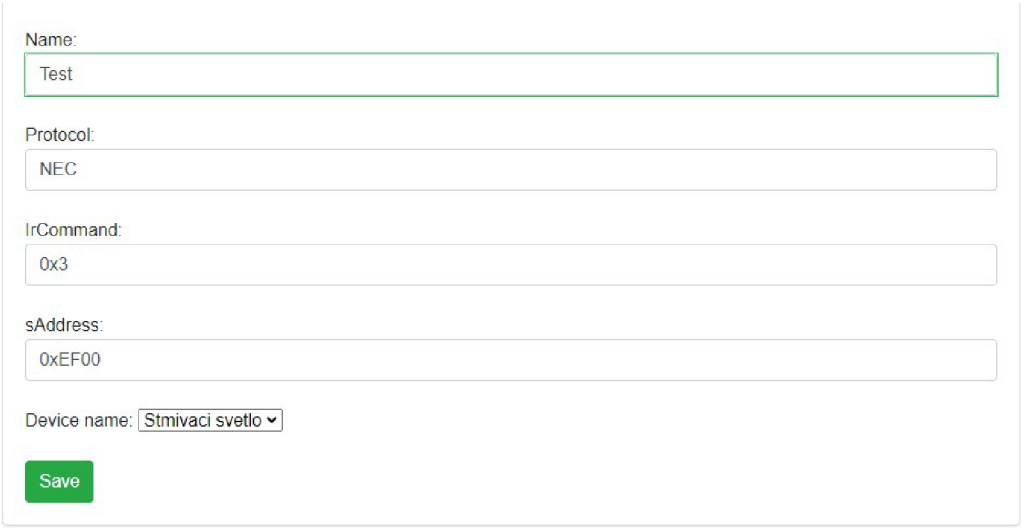

**Obrázek 37: Použité ovladače pro testování (vlastní zpracování)**

## 7.5 Testování klientského rozhraní

Aby bylo možné korektně vlastní zařízení správně nastavit, bylo nutné udělat uživatelsky intuitivní a jednoduchou administraci zařízení. Nyní zde nastíním problém s přidáváním a spravováním zařízení. Předvedu zde (viz. níže) průběh přidání jednoho zařízení od naskenování IR kódu po přiřazení.

Po naskenování kódu z Arduino přijímače se přijatý kód uloží do nepřiřazených kódů. Pokud už kód existuje v nepřiřazených kódech, tak se pouze aktualizuje datum naskenování na aktuální datum. Tak v nepřiřazených kódech nejsou stejné kódy

vícekrát. Zde bylo testováno vkládání více stejných kódu a zda se správně všechny informace ukládají. Dále bylo testováno přiřazení kódu k zařízení a jeho odmazání z nepřiřazených kódů.




**Obrázek 38: Ukázka přiřazení kódu k zařízení (vlastní zpracování)**

Po přiřazení je kód přesunut do kódu k zařízení, tím se nastaví jeho funkcionalita. Zde se mohou u kódu upravovat všechny informace (jméno, protokol, příkaz, adresa, zařízení a funkce). Testována byla úprava jednotlivých změn a poté celková úprava. Dále bylo testování mazání jednotlivých přiřazených kódů a generování kódů pro Homebridge a Arduino prvky.

Generování kódu pro Homebridge a Arduino prvky bylo testováno na nově přidaných zařízení a aplikováním vygenerovaného kódu. Při nepřiřazení všech kódů pro zvolené zařízení se neprovede příkaz a záznam je přidán v logu Homebridge. Pokud by byl z nějakého důvodu JSON nevalidní (pokud by uživatel upravil záznam přidáním do jména zařízení například znak “, Homebridge tuto chybu pozná a nedovolí konfigurační soubor uložit. Při špatném vygenerování kódu pro Arduino se do vygenerovaného kódu přidává IP adresa a MAC adresa, kterou uživatel zadal. Pokud už v síti existuje zařízení ze shodnou MAC adresou, způsobí to nefunkčnost jednoho ze zařízení se stejnou MAC adresou. To samé se stane při přiřazení 2 stejných IP adres.

# 8 Diskuze výsledků

U vytváření vlastního řešení bylo nutné prozkoumat mnoho zdrojů a pochopit princip fungování ekosystému HomeKit. Jelikož oblast chytré domácnosti se rychle mění, bylo v průběhu psaní této práce nutné neustále ověřovat zdroje a zda nebylo vyvinuto nějaké nové zařízení.

Při tvorbě vlastního zařízení bylo zapotřebí se vypořádat s mnoha problémy jako třeba volba zařízení vhodných pro realizaci. Většinu zařízení jsem zvolil, protože byly již zakoupené nebo pro mě byly ze subjektivního pohledu nejlepším řešením. Například Arduino UNO R3 s Arduino Ethernet Shield není pro všechny uživatele vhodné. Toto zařízení jsem zvolil, protože domácnost, ve které je vlastní zařízení používáno, se nachází v silně obydlené oblasti s velkým frekvenčním šumem, bylo nově zrekonstruováno a jelikož se při realizaci rekonstrukce počítalo s chytrou domácností, tak je na mnoha místech rozveden UTP kabel. Pro většinu uživatelů se silným pokrytím WiFi bude daleko vhodnější variantou ESP8266 nebo jiné zařízení podporující WiFi. Dalším zařízením, které je vhodné pro většinu uživatelů je použití Raspberry Pi jako server, pokud doma server nemáte.

Při odesílání z Arduino přijímače došlo k neočekávané události, jelikož pro Arduino Ethernet Shield není podpora TLS, to způsobuje nemožnost posílat data na HTTPS. To bylo způsobeno malým výpočetním výkonem zařízení pro šifrování komunikace.

Problém u IR ovládání je, že není možné zjistit stav v jakém se koncové zařízení nachází (jestli je vypnuto, jaká barva je rozsvícena, atd.). Pokud bude uživatel tedy ovládat zařízení, jak dálkovým ovladačem, tak přes aplikaci Home, tak v aplikaci Home stav zařízení nebude odpovídat reálnému stavu.

Kdyby chtěl uživatel použít IR Arduino vysílač pro více koncových zařízení používajících stejný IR kód, bude docházet k nechtěnému ovládání všech těchto zařízení se stejným IR kódem. Jediným možným řešením tedy je namířit IR diodu jen na jedno koncové zařízení a vytvořit druhý Arduino vysílač, kde bude IR dioda namířena na další koncové zařízení se stejným IR kódem.

Podle mého názoru nelze dosáhnout obecného řešení pro všechna IR koncová zařízení, protože by se jednalo o uživatelsky skoro nemožné nastavování. Pro

rozšíření na více zařízení by tedy bylo vhodné, aby byla podpora realizována pomocí balíčků koncových zařízení. V praxi by to znamenalo, že by si uživatel našel balíček k příslušnému ovladači a doinstaloval jej. Takto by byla zaručena maximální kompatibilita. Vlastní zařízení je navrženo tak, aby se jednoduchým vložením do databáze takto dalo rozšiřovat stávající řešení.

Velký důraz byl kladen na zabezpečení vlastního řešení. Zabezpečení obsahuje dvě vrstvy. První vrstva spoléhá na to, že do domácí sítě nebude přidána nekorektní osoba, jelikož ve většině případů je domácí síť zabezpečena, což způsobuje nemožnost připojení do domácí sítě. Druhá vrstva je v rámci sítě, když už se útočník dostal přes první vrstvu. Zde je bezpečnost IR ovládání ošetřena, neboť proto veškerá komunikace s API (kromě přihlášení) požaduje token, jinak vrátí chybový HTTP kód 401 (uživatel neautorizován).

Firma BroadLink jako jediná na českém trhu nabízí univerzální ovladač, který je možné propojit s aplikací Homebridge (HAP server). Homebridge lze následně propojit s Apple Home. BroadLink nabízí dva produkty a to RM4C MINI a PRO. Hlavní rozdíl je, že PRO dokáže komunikovat i v bezlicenčním frekvenčním pásmu 433MHz. BroadLink podporuje více než 50 000 zařízení v kategoriích vypínač, zásuvka, ventilátor, osvětlení, garážové dveře, zámek dveří, klimatizace a okenní rolety.

Výsledkem diplomové práce je řešení, které umožňuje nasnímat a uložit jakýkoliv typ IR ovladače (např. televize, rádio). Diplomová práce také umožňuje rozšíření programového kódu a tím možnost přizpůsobit funkcionalitu ovládacímu prvku. Například, když originální ovladač nebo univerzální ovladač odesílá kód pro zvýšení jasu. HAP ale podporuje pouze číselné příkazy pro nastavení jasu a neuchovává stávající hodnotu, proto není možné přesně určit jaká hodnota má být nastavena. Další výhodou vlastního řešení oproti řešení od BroadLink je v automatickém přidávání zařízení do Homebridge pomocí autorem vytvořeném generátoru kompletních JSON souborů (metadata i IR kódy), oproti řešení od společnosti BroadLink. Zde je nutné ručně vytvořit JSON i s IR kódy. Přehledné porovnání vlastního řešení s BroadLink lze nalézt v Tabulce 7.

| | BROADLINK RM4C MINI | VLASTNÍ ŘEŠENÍ |
|---|---|---|
| Přidání nového IR kódu | NE | ANO |
| Přidání nového zařízení pro Apple Home | NE | ANO |
| Připojení přes Wifi | ANO | ANO s WiFi modulem |
| Připojení přes LAN | NE | ANO s LAN modulem |
| Snadné generování JSON pro HAP | NE | ANO |
| Přizpůsobení ovládání | NE | ANO |

**Tabulka 7: Porovnání vlastního řešení s konkurencí (vlastní zpracování)**

# 9 Závěry

Cílem této diplomové práce bylo aplikování IR zařízení do chytré domácnosti od společnosti Apple s názvem HomeKit. Z analýzy trhu jsem našel pouze 2 firmy, které se zabývají tvorbou IR univerzálních ovladačů, které jsou rozšiřitelné po instalaci HAP (Homebridge) a pluginu o implementaci do aplikace Home. Bohužel ani jedna z firem neposkytuje univerzální řešení pro všechny IR zařízení. Ani vlastní řešení není schopno vyřešit tento problém bez nutnosti programátorského upravování kódu. To je způsobeno rozdílností ovladačů, která je nutná řešit přizpůsobením funkcí v API. Vlastní zařízení je schopné pracovat s koncovými zařízeními typu světlo (podpora ovládání jasu, barvy a vypínač) a vypínač. Pokud si uživatel chce přidat jiné zařízení a ovládat ho, například klimatizaci, je nutné tento typ přidat do databáze a nadefinovat jeho funkce. Bohužel nelze přidat univerzální řešení, jelikož každý ovladač poskytuje jiné funkce.

Vlastní řešení pro odposlechnutí IR kódů a replikaci je jednoduché na instalaci ale je nutná alespoň základní znalost počítačových sítí a pokročilá znalost ovládání počítače pro instalaci serverové části, kde je potřeba nastavovat porty a statické IP adresy.

Na základě poznatků z načerpaných znalostí při tvorbě diplomové práce není možné udělat univerzální IR zařízení, které by integrovalo všechny IR zařízení do aplikace Home. To je z toho důvodu, že Ir ovládání jsou velice rozmanité a jejich funkční tlačítka neodpovídají standardům v HAP Specification. Tento problém by bylo možné dočasně odstranit přizpůsobení IR ovladače v programovém kódu. Problém ale nastává v tom, že by úpravu bylo nutné shromáždit všechny IR vyráběné ovladače a zavést jejich funkcionalitu.

# 10 Seznam použité literatury


[1] Internet of Things (IoT) [online]. *TechTarget*, 2020 [cit. 06.08.2021]. Dostupné z: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[2] Zigbee [online]. *Zigbee Alliance*, 2021 [cit. 06.08.2021]. Dostupné z: https://zigbeealliance.org/solution/ZigBee/

[3] Z Wave Vs ZigBee: Which Is Better For Your Smart Home? [online]. *thesmartcave.com*, 2017 [cit. 06.08.2021]. Dostupné z: https://thesmartcave.com/z-wave-vs-zigbee-home-automation/

[4] ZigBee [online] *Zigbee Alliance*, 2021 [cit. 23.07.2021]. Dostupné z: https://ZigBeealliance.org/wp-content/uploads/2019/12/ZigBee-Alliance-Trademark-and-Logo-Usage-Guidelines-and-Terms.pdf

[5] Z-Wave Alliance [online]. *Z-Wave Alliance*, 2021 [cit. 06.08.2021]. Dostupné z: https://www.Z-Wave.com/

[6] MFi Program [online]. *Apple*, 2021 [cit. 06.08.2021]. Dostupné z: https://mfi.apple.com/en/home.html

[7] MFi Program [online]. *Apple*, 2021 [cit. 06.08.2021]. Dostupné z: https://mfi.apple.com/en/how-it-works.html

[8] LIEW, S. C. Electromagnetic Waves. Centre for Remote Imaging, Sensing and Processing. [online]. *Centre for Remote Imaging, Sensing and Processing*., 2015 [cit. 06.08.2021]. Dostupné z: https://crisp.nus.edu.sg/~research/tutorial/em.htm

[9] Meliconi CONTROL 2.1 [online]. *Meliconi Italia*, 2021 [cit. 09.08.2021]. Dostupné z: https://www.meliconi.com/download/

[10] Bond Home [online]. Olibra, 2021 [cit. 06.08.2021]. Dostupné z: https://bondhome.io/

[11] Homebridge Broadlink RM [online]. *Homebridge*, 2021 [cit06.08.2021]. Dostupné z: https://lprhodes.github.io/slate/

[12] Dvorak, Jan & Krejcar, Ondrej & Cheng, Lim. (2019). *Application of universal remote control of non-smart home appliances for smart home concepts. International Journal of Digital Enterprise Technology*. 1. 276. 10.1504/IJDET.2019.097847.

[13] FEILER, Jesse. Learn Apple HomeKit on iOS: a home automation guide for developers, designers, and homeowners. [New York]: *Apress*, [2016]. ISBN 978-1484215289.

[14] HomeKit [online]. *Apple*, 2021 [cit. 06.08.2021]. Dostupné z: https://developer.apple.com/HomeKit/

[15] Homebridge [online]. *Homebridge, 2021 [cit.* 06.08.2021*]. Dostupné z: https://github.com/homebridge/homebridge*

[16] Arduino [online]. *Arduino, 2021 [cit.* 06.08.2021*]. Dostupné z: https://store.arduino.cc/digital/create*

[17] Arduino - Software [online]. *Arduino, 2021 [cit.* 06.08.2021*]. Dostupné z: https://www.arduino.cc/en/software*

[18] IRMP [online]. *GitHub*, 2021 [cit. 06.08.2021]. Dostupné z: https://github.com/ukw100/IRMP

[19] IRLremote [online]. *GitHub*, 2021 [cit. 06.08.2021]. Dostupné z: https://github.com/NicoHood/IRLremote

[20] IRLib2 [online]. *GitHub*, 2021 [cit. 06.08.2021]. Dostupné z: https://github.com/cyborg5/IRLib2

[21] IRremote Arduino Library [online]. *GitHub*, 2021 [cit. 06.08.2021]. Dostupné z: https://github.com/Arduino-IRremote/Arduino-IRremote

[22] Minimal NEC [online]. *GitHub*, 2021 [cit. 06.08.2021]. Dostupné z: https://github.com/Arduino-IRremote/Arduino-IRremote/tree/master/examples/MinimalReceiver

[23] NEC Infrared Transmission Protocol [online]. Texas Instruments, 2021 [cit. 29.07.2021]. Dostupné z: https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol

[24] Ethernet library [online]. *Arduino*, 2021 [cit. 06.08.2021]. Dostupné z: https://www.arduino.cc/en/Reference/Ethernet

[25] SPI library [online]. *Arduino*, 2021 [cit. 06.08.2021]. Dostupné z: https://www.arduino.cc/en/Reference/SPI

[26] KELLYN GORMAN, ALLAN HIRT, DAVE NODERER, JAMES ROWLAND-JONES, ARUN SIRPAL, DUSTIN RYAN, AND BUCK WOODY *Introducing Microsoft SQL Server 2019*. 1st ed. 2019. ISBN 978-1-83882-621-5.

[27] LACHEWICZ, Katarzyna. Performance analysis of selected database systems: MySQL, MS SQL, PostgerSQL in the context of web applications. *Journal of Computer Sciences Institute* [online]. 2020, **14** [cit. 2021-7-16]. ISSN 25440764. Dostupné z: doi:10.35784/jcsi.1583

[28] LACKO, Ľuboslav. Microsoft SQL Server 2008 - praktický sprievodca novinkami. 2 aktualizované. *Microsoft*, 2008.

[29] What is SQL Server Management Studio (SSMS)? [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15

[30] Visual Studio 2019 [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://visualstudio.microsoft.com/cs/vs/

[31] GRPC [online]. *CNCF*, 2021 [cit. 06.08.2021]. Dostupné z: https://grpc.io/

[32] Jim Webber, Savas Parastatidis a Ian Robinson. REST in Practice. First. 1005 Gravenstein Highway North, Sebastopol, California*: O'Reilly Media*, říj. 2010.

[33] OpenAPI Specification [online]. *SmartBear Software*, 2021 [cit. 06.08.2021]. Dostupné z: https://swagger.io/specification/

[34] Maximum URL length [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://support.microsoft.com/en-us/topic/maximum-url-length-is-2-083-characters-in-internet-explorer-174e7c8a-6666-f4e0-6fd6-908b53c12246

[35] IIS [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://www.iis.net/

[36] The OAuth 2.0 Authorization Framework [online]. *datatracker.ietf.org*, 2012 [cit. 06.08.2021]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc6749

[37] OAuth 2 Simplified [online]. *Aaron Parecki*, 2021 [cit. 06.08.2021]. Dostupné z: https://aaronparecki.com/oauth-2-simplified/

[38] Zabezpečení webového rozhraní API pomocí individuálních účtů a místního přihlášení v ASP.NET Web API 2,2 [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/web-api/overview/security/individual-accounts-in-web-api

[39] ASP.NET Core Middleware [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/fundamentals/middleware/?view=aspnetcore-5.0

[40] HTTP Over TLS [online]. *The Internet Society, 2000 [cit.* 06.08.2021*]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc2818*

[41] FIELDING, R.; GETTYS, J.; MOGUL, J.; aj.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), *datatracker.ietf.org,* 1999, [citováno dne 18-11-2013]. URL http://www.ietf.org/rfc/rfc2616.txt

[42] Callegati, Franco & Cerroni, Walter & Ramilli, Marco. (2009). Man-in-the-middle attack to the HTTPS protocol. Security & Privacy, IEEE. 7. 78 - 81. 10.1109/MSP.2009.12.

[43] Postman [online]. *Postman*, 2021 [cit. 06.08.2021]. Dostupné z: https://www.postman.com/

[44] Open Source [online]. *DigitalOcean*, 2021 [cit. 06.08.2021]. Dostupné z: https://opensource.org/

[45] Úvod do Razor stránek v ASP.NET Core [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/razor-pages/?view=aspnetcore-5.0&tabs=visual-studio

[46] ASP.NET MVC Pattern. [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://dotnet.microsoft.com/apps/aspnet/mvc

[47] Přehled ASP.NET Core MVC [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/mvc/overview?view=aspnetcore-5.0

[48] JavaScript [online]. *Pluralsight*, 2021 [cit. 06.08.2021]. Dostupné z: https://www.javascript.com/

[49] WebAssembly [online]. *W3C Community Group*, 2021 [cit. 06.08.2021]. Dostupné z: https://webassembly.org/

[50] Úvod do ASP.NET Core Blazor. [online]. *Microsoft*, 2021 [cit. 06.08.2021]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/blazor/?view=aspnetcore-5.0

[51] Homebridge plugin to control a web/http-based RGB device.. GitHub: Where the world builds software [online]. *GitHub*, 2021 [cit. 06.08.2021]. Dostupné z: https://github.com/QuickSander/homebridge-http-rgb-push

[52] Homebridge plugin to control a web-based RGB device [online]. *GitHub*, 2021 [cit. 06.08.2021]. Dostupné z: https://github.com/jnovack/homebridge-better-http-rgb

[53] An http/https server inside Homebridge to receive notifications from external programs. [online]. *GitHub*, 2021 [cit. 06.08.2021]. Dostupné z: https://github.com/Supereg/homebridge-http-notification-server

[54] Using the Collection Runner [online]. *Postman*, 2021 [cit. 06.08.2021]. Dostupné z: https://learning.postman.com/docs/running-collections/intro-to-collection-runs/

[55] Analog | Embedded processing [online]. *Texas Instruments*, 2015 [cit. 06.08.2021]. Dostupné z: https://www.ti.com/lit/an/slaa644b/slaa644b.pdf

# 11 Přílohy

## *11.1 Seznam použitých zkratek*

ADO - Microsoft ActiveX Data Objects

API – Application Programming Interface

ASP - Active Server Pages

AWS - Amazon Web Services

BI – Business intelligence

CF - Code First

DB - Database

DF - Database First

DHPC – Dynamic Host Configuration Protocol

EF - Entity Framework

EFC - Entity Framework Core

gRPC - high performance Remote Procedure Call

HAP – HomeKit Accessory Protocol

HTML -Hypertext Markup Language

HTTP - Hypertext Transfer Protocol

HTTPS - Hypertext Transfer Protocol Secure

IDE - Integrated Development Environment

IFTTT – If This Then That

IIS - Internet Information Services

IoT – Internet of Things

IP – Internet Protocol

Ir – Infrared

JSON - JavaScript Object Notation

LTS - Long-term support

MAC - Media Access Control

MCR - Microsoft Container Registry

MFi – Made For iPhone

MS - Microsoft

MSSQL- Microsoft SQL

MVC - Model-view-controller
NAS - Network Attached Storage
OAS -OpenAPI Specification
ORM - Objektově relační mapování
RF - Radio Frequency
PHP - Hypertext Preprocessor
REST - Representational state transfer
RPC - Remote Procedure Call
SOAP - Simple Object Access Protocol
SPI - Serial Peripheral Interface
SQL - Structured Query Language
SSMS - SQL Server Management Studio
TCP - Transmission Control Protocol
TCP – Transmission Control Protocol
TLS - Transport Layer Security
UID - Unique Identification
URL - Uniform Resource Locator
VS - Visual Studio
WCF -Windows Communication Foundation
WiFi - Wireless High-Fidelity
XML - Extensible Markup Language

## *11.2 Obsah adresáře*

V adresáři se nachází tyto přílohy:

- Zdrojové kódy pro jednotlivé implementace
- Schéma pro Arduino přijímač a vysílač
- Ukázky webového rozhraní

Univerzita Hradec Králové
Fakulta informatiky a managementu

# Zadání diplomové práce

**Autor:** **Bc. Václav Divíšek**

Studium: I1800315

Studijní program: N1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

**Název diplomové práce:** **Analýza možností systému iOS pro implementace Smart Home řešení**

Název diplomové práce AJ: Analysis of Possibilities for Smart Home Solution on iOS Platform

**Cíl, metody, literatura, předpoklady:**

Cíl: Analyzovat možnosti vývoje softwarových řešení s využitím platformy iOS pro potřeby regulace a řízení Smart Home řešení. Navrhnout softwarové řešení s využitím moderních HW prvků včetně implementace a otestování v reálném prostředí.

Osnova práce:

1. Úvod a cíle práce
2. Definice problému a přehled existujících výzkumných prací
3. iOS platforma a specifika vývoje tématicky blízkých řešení
4. Návrh řešení ovládacího software
5. Implementace navrženého řešení
6. Testování, experimenty a jejich vyhodnocení
7. Diskuze dosažených výsledků a možné další rozšíření práce
8. Závěr

Dokumentace -

1. Apple Inc. : HomeKit Accessory Protocol Specification
2. Apple Inc. : https://developer.apple.com/ (Dokumentace k vývoji na Apple zařízení)
3. Arduino : https://www.arduino.cc/en/main/docs

Knihy

1. FEILER, Jesse. Learn Apple HomeKit on iOS: a home automation guide for developers, designers, and homeowners. New York: Apress, [2016]. ISBN 978-1484215289.
2. Gerard O'Driscoll. Definitive Guide to Apple's HomeKit Smart Home Automation System: Discover How to Use the Home App in iOS 10 To Build Your Own Smart Home Using Apple? s ... Home Automation Essential Guides Book 7), [2017]. ASIN B01NBSOXYN
3. JOHN, Nussey. *Arduino for dummies, 2nd edition . 2nd edi. Indianapolis, IN: John Wiley, 2018. ISBN 978-1119489542.*

Garantující pracoviště: Katedra informačních technologií, Fakulta informatiky a managementu

Vedoucí práce: prof. Ing. Ondřej Krejcar, Ph.D.

Datum zadání závěrečné práce: 15.9.2020