VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

INTELIGENTNÍ DETEKTOR PREZENCE

INTELLIGENT PRESENCE DETECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **VOJTĚCH KRONIKA**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Kronika Vojtěch**
Program: Informační technologie
Název: **Inteligentní detektor prezence**
Intelligent Presence Detection
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte problematiku IoT a Smart Environments. Zaměřte se na detekci prezence. Seznamte se s existujícími komerčními (Intellithings), tak DIY a opensource řešeními, která využívají bluetooth pro detekci a identifikaci konkrétního uživatele v definované zóně inteligentní budovy.
2. Stanovte požadavky na detekci prezence a na základě inspirace z bodu 1 navrhněte vlastní řešení.
3. Navržený systém realizujte s využitím platforem na bázi SoCs Espressif a Raspberry Pi.
4. Ověřte funkčnost v reálném provozu a pokuste se porovnat dosažené výsledky s jinými známými řešeními z bodu 1.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- První 2 body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Janoušek Vladimír, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

V této práci jsem se zabýval řešením vnitřní lokalizace, jejím návrhem a realizací. Jsme v době, kdy je téma chytré domácnosti denní záležitostí a řešení vnitřní lokalizace zatím nemá jistého vítěze. Cílem práce je vytvořit inteligentní detektor prezence, pomocí technologie Bluetooth, který rozpozná, v které místnosti se zařízení nachází, a odhadne, v které části místnosti se zařízení může nacházet. Pro dosažení cíle používám minimálně tři senzory v každé místnosti a porovnávám síly signálů mezi senzory a lokalizovaným zařízením. Na redukci šumu z datových signálů od senzorů je použit Kalmanův filtr. Výsledkem práce je funkční řešení pro vnitřní lokalizaci. Zjistil jsem, že není jednoduché použít rychle se měnící sílu signálu pro triangulaci a získat tak přesnou polohu zařízení. Další práce by se mohla zabývat tím, co je potřeba udělat, aby triangulace byla možná a funkční. Jednou z možných cest je prozkoumání technologie Bluetooth Direction Finding.

Abstract

In this work I dealt with the solution of indoor localization, its design and implementation. We are at a time when the topic of smart home is a daily affair and the solution for indoor localization does not have a certain winner yet. The goal of this work is to create an intelligent presence detector, using Bluetooth technology, which recognizes in which room the device is located. And it estimates in which part of the room the device can be located. To achieve the goal, I use at least three sensors in each room and compare the signal strengths between the sensors and the localized device. A Kalman filter is used for a noise reduction on data signals from sensors. The result of the work is a functional solution for indoor localization. I have found that it is not easy to use rapidly changing signal strength within triangulation to obtain the exact position of the device. Further work could address what needs to be done to make triangulation possible and functional. One of possible directions is an exploration of Bluetooth Direction Finding technology.

Klíčová slova

Vnitřní lokalizace, Detektor prezence, Bluetooth, Chytrá domácnost, Inteligentní zařízení, ESP32, Home Assistant, ESPresense, RoomMe od Intellithings, Go, React, PlatformIO, Kalmanův filtr

Keywords

Indoor localization, Presence detection, Bluetooth, Smart Home, Smart device, ESP32, Home Assistant, ESPresense, RoomMe by Intellithings, Go, React, PlatformIO, Kalman filter

Citace

KRONIKA, Vojtěch. *Inteligentní detektor prezence*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Vladimír Janoušek, Ph.D.

Inteligentní detektor prezence

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Vladimíra Janouška, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Vojtěch Kronika
6. května 2022

Poděkování

Rád bych poděkoval mému vedoucímu práce doc. Ing. Vladimírovi Janouškovi, Ph.D. za konzultace, rady, nápady a čas věnovaný mé bakalářské práci.

Obsah

1 Úvod **2**

2 Detekce prezence **3**
2.1 Detekce blízkosti 4
2.2 Triangulace 4
2.3 Analýza scény 5

3 Existující řešení **6**
3.1 RoomMe 6
3.2 ESPresence 7

4 Návrh řešení **10**
4.1 Detektor pohybu 11
4.2 MQTT Broker 12
4.3 Webová aplikace 13
4.4 Home Assistant 14

5 Realizace řešení **15**
5.1 Detektor pohybu 15
5.1.1 Použité knihovny, popis a důvod použití 16
5.1.2 Nastavení detektoru 16
5.1.3 Program 17
5.2 Webová aplikace 19
5.2.1 Backend 19
5.2.2 Frontend 22
5.3 Home Assistant 25

6 Ověření funkčnosti v reálném provozu **27**
6.1 Ověření funkčnosti 27
6.2 Testování 28
6.3 Shrnutí výsledku provedených testů 31

7 Závěr **32**

Literatura **34**

A Obsah paměťového média **35**

B Grafické rozhraní webové aplikace **36**

Kapitola 1

Úvod

S novými technologiemi vznikají i nová inteligentní zařízení, která umí monitorovat různé veličiny z reálného světa. Tato zařízení se navzájem propojují do informačních sítí, pro které se používá označení Internet of Things (IoT). V této práci se zabývám úkolem, jak monitorovat prezenci takovéhoto zařízení v místnosti v rámci IoT.

Internet of Things je v informatice označení pro síť zařízení, která dokáží spolu komunikovat. Proč je IoT důležitý? Propojit malá zařízení s limitovaným výkonem, vlastní pamětí a nízkou energetickou spotřebou v sobě skrývá obrovský potenciál. Tato chytrá zařízení nám mohou sama sbírat data a informace. Díky rozmachu odvětví jako jsou Big data a umělá inteligence pak můžeme tato data zpracovat a učit se z nich, v budoucnu tak budeme moci celou řadu systémů a věcí automatizovat a následně jednotlivé procesy efektivně zlepšovat. Jestliže jsou zařízení připojená k internetu, respektive ke speciálním IoT sítím, pak tato zařízení mohou dělat tyto základní činnosti: sbírat a odesílat informace, přijímat informace a následně na ně reagovat.

Jednou ze základních úloh, kterou IoT musí zvládnout, je detekce prezence zařízení. Ve své práci se budu zabývat existujícími řešeními této úlohy, ale zároveň se pokusím i navrhnout své vlastní řešení.

Cílem práce je vytvořit detektor prezence pomocí technologie Bluetooth, který bude schopen identifikovat konkrétní zařízení v definované zóně inteligentní budovy. Nastudoval jsem si problematiku inteligentních zařízení, které umí detekovat prezenci pomocí Bluetooth. Seznámil se s existujícím komerčním řešením od Intellithings, které dále prezentuji v sekci 3.1 a open source řešením ESPresence představeným v sekci 3.2. Obě řešení jsem otestoval a zjistil, jak fungují.

Na základě získaných zkušeností jsem navrhl vlastní systém, který se inspiruje zmíněným komerčním i open source řešením. Ve svém řešení budu využívat tři senzory pro každou kontrolovanou místnost, které budou komunikovat skrze MQTT broker s webovou aplikací. Ta bude data zpracovávat a vyhodnotí, v jaké místnosti a v jaké definované zóně se konkrétní uživatel a chytrá zařízení nachází. Aplikace bude umět data zobrazit ve webovém uživatelském rozhraní jako administrátorský panel nebo panel pro každou místnost zvlášť s její mapou. Pro lepší přehlednost uživatelského prostředí na mapě bude možno vidět rozměry místnosti, definované zóny, pozice senzorů a pohyblivé zařízení podle odhadu jejich pozic. Zpracovaná data se odešlou pomocí MQTT do hubu chytré domácnosti.

Navržený systém jsem realizoval pomocí senzorů ESP32 a webové aplikace, která zpracovává data a zobrazuje uživatelské a administrativní rozhraní. Pro senzory jsem naprogramoval dva vhodné firmwary. Následně jsem prokázal funkčnost celého řešení pří plnění zadaného úkolu detekce zařízení v místnosti.

Kapitola 2

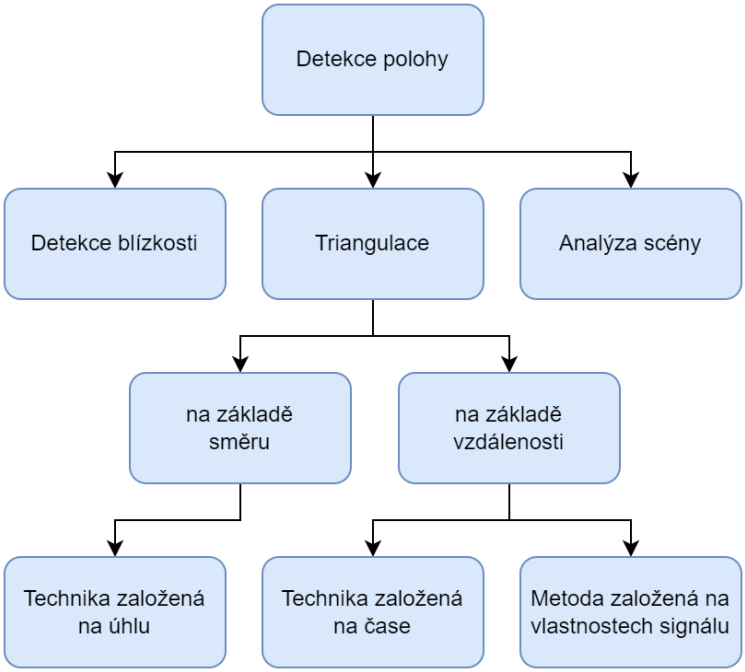
Detekce prezence

Lokalizační služby jsou významnou technologií a stávají se nezbytnou součástí života. V této době, zejména v bezdrátových komunikačních sítích, lokalizační služby existují v širokém rozpětí od komunikace krátkého dosahu až po telekomunikační sítě dlouhého dosahu [8]. Telekomunikační sítě dlouhého rozsahu se používají hlavně ve volném prostoru. Jsou velmi přesné, ale mají i svá omezení. Nefungují například v členitém terénu městské aglomerace, neboť často dochází k silnému stínění signálu.

Ve své práci se zabývám lokalizačními službami krátkého dosahu. Pro detekci prezence uvnitř budov se používají systémy na základě rádiových frekvencí, které nemají problémy s průchodem signálu zdí, ani s překonáváním překážek. Takovými systémy jsou vybavena zařízení používající technologie WiFi, Bluetooth či ZigBee[1]. Z tohoto důvodu detekční systémy na bázi rádiových frekvencí jsou schopné pokrýt větší prostor a vyžadují často méně komplikovaný hardware v porovnání s jinými systémy. Ve své práci se budu zabývat hlavně radiofrekvenční technologií Bluetooth, která je v současné době bezdrátovým standardem pro Wireless Personal Area Network (WPAN[2]). Téměř všechna WiFi mobilní zařízení, jako například mobilní telefony či přenosné počítače, mají zabudovanou Bluetooth technologii. Hlavními výhodami používání Bluetooth pro výměnu informací mezi zařízeními je to, že tato technologie je velmi bezpečná, vyžaduje nízké náklady, vystačí si s malou spotřebou energie a použitá zařízení mívají malé rozměry. Každé Bluetooth zařízení má unikátní identifikační číslo ID, které může být použito k lokalizaci a k identifikaci zařízení. Nevýhodou této Bluetooth technologie je skutečnost, že při každé lokalizace probíhá takzvaná Device Discovery procedura, která výrazně prodlužuje zpoždění lokalizace zařízení o cca 10 až 30 sekund. Zároveň tím dochází i k vyšší spotřebě energie. Z toho důvodu jsou Bluetooth zařízení nevhodná pro aplikace vyžadující rychlé určování přesné polohy v reálném čase.

Zaměřím se na bezdrátové vnitřní lokalizační techniky a systémy pro vnitřní aplikace. Pro lokalizační techniky a algoritmy v bezdrátové lokalizaci se používá několik různých metod. Techniky detekce polohy lze rozdělit do tří obecných kategorií: detekce blízkosti, triangulace a analýza scény, jak je znázorněno na obrázku 2.1 [8].

[1] https://cs.wikipedia.org/wiki/ZigBee
[2] https://en.wikipedia.org/wiki/WPAN



Obrázek 2.1: Způsoby detekce polohy. Převzato z [8]

2.1 Detekce blízkosti

Detekce blízkosti je jednou z nejjednodušších metod určování polohy, kterou lze implementovat. Tato metoda je založena na konektivitě a využívá měření síly signálu. Například u Bluetooth technologie síla signálu zavisí na vzdálenosti mezi zařízeními. Tohoto jevu se využívá v různých aplikacích tak, že se zařízení připojí k vysílači nebo přijímači, který je nejblíže a má tedy nejsilnější signál.

2.2 Triangulace

Triangulace využívá geometrické vlastnosti trojúhelníků k bližšímu určení cílového umístění. Jsou dvě různé techniky jak využít triangulaci. První technika je založena na směru resp. na úhlu přijatého signálu. Druhá technika je založena na čase zpoždění signálu.

Technika založená na úhlu

Metoda vyžaduje vybavení přijímačů směrovými anténami pro přesné určení úhlů přijímaného signálu. Tímto se celé zařízení stává technicky komplikovaným i finančně drahým. Tato metoda může fungovat dobře venku, ale ve vnitřní lokalizaci se nedoporučuje kvůli odrazům signálu od stěny a dalších objektů. Pro získání odhadu 2D pozice tento přístup vyžaduje pouze dva přijímače. Pro zlepšení přesnosti odhadu můžeme použít přijímačů více. Abychom úhel zjistili, přijímač musí umět získat informaci, kterým směrem se zařízení nachází. Tuto informací umí získat zařízení s implementovanou Bluetooth technologií od verze 5.1 díky funkci Bluetooth Direction Finding.

Technika založená na čase

Technika založená na čase využívá měření času šíření signálu od pohybujícího se zařízení k pevně umístěným přijímačům. Aby tato technika dostatečně přesně fungovala, je nutné, aby vysílací zařízení poslalo časovou značku směrem k přijímačům. Jakmile je tato značka přijata, vzdálenost mezi pohybujícím se zařízením a přijímačem je vypočítána z časového zpoždění signálu a z rychlosti signálu. Tato metoda vyžaduje přesnou znalost počátečního času odeslání signálu. Proto je nutné, aby všechny přijímače byly velmi přesně synchronizovány s časovým zdrojem. Časová synchronizace se tak stává hlavní nevýhodou tohoto postupu, neboť vyžaduje použití dalšího synchronizačního zařízení. Další nevýhodou metody je skutečnost, že v případě nehomogenního prostředí, kde se signál šíří nerovnoměrnou rychlostí, dochází k dalšímu zpoždění signálu a následně i k nepřesnosti detekce lokalizace pohybujícího se předmětu.

2.3 Analýza scény

Mezi metody používající analýzu scény pro detekci přítomnosti se řadí Dead Reckoning a Map Matching. Dead Reckoning je proces odhadu současné pozice na základě poslední určené pozice a předpokládaného směru pohybu a rychlosti pohybu. Tento princip je široce používán v celé řadě navigačních systémů. Nevýhodou metody je skutečnost, že s narůstajícím časem chyba způsobená nepřesností systému narůstá. Map Matching je ještě složitější metoda využívající znalosti příslušných elektronických map a pokročilejších technologií jako jsou topologické analýzy či rozpoznávání vzorů.

Kapitola 3

Existující řešení

Prozkoumání existujících řešení mi pomůže v lepší orientaci v dané problematice a k rychlejšímu návrhu vlastního řešení. V této kapitole se budu věnovat již existujícím systémům, které řeší tuto problematiku, ať už se jedná o komerční řešení či o systémy na bázi open source technologií. Zaměřím se základní kritéria hodnocení systémů jako jsou jejich funkčnost, spolehlivost, rychlost, jednoduchost instalace a používání či cena řešení.

3.1 RoomMe

Jedná se o inteligentní zařízení vyšší cenové kategorie od společnosti Intellithings, které dokáže zjistit, zda se uživatel s mobilní aplikací RoomMe nachází v blízkosti jednoho ze senzorů zapojených do systému. Jeho výhodou je, že se chová jako smart hub. Smart hub je zařízení, ke kterému se jiná chytrá zařízení připojí a přes něj komunikují s chytrou domácností, chytrými asistenty a dalšími chytrými zařízeními systémů IoT. Vidím v tom velkou výhodu pro nově budované chytré domácnosti, které ještě nemají v každé místnosti smart hub. RoomMe smart hub nepotřebuje, protože jím sám je. Další výhodou tohoto řešení je například v možnosti propojení s chytrými asistenty, kteří se také často chovají jako smart hub. Vzhledem k tomu, že většinou nemáte ve všech místnostech chytré domácnosti asistenta, tak ho můžete nahradit právě RoomMe senzorem. Velkou nevýhodou je, že zařízení ovládá RoomMe sám na vámi vytvořených automatizacích, většinou tedy pouze při příchodu a odchodu z místností. Nemůžete tedy zařízení ovládat na dálku jako je tomu například v případě chytrého asistenta od společnosti Google pomocí aplikace Google Home.

Detekce prezence v místnosti pomocí tohoto systému funguje na bázi radiofrekvenčního signálu Bluetooth tak, že zařízení nainstalujete na strop místnosti ve vzdálenosti jeden až dva metry od vchodových dveří. Zvolená vzdálenost závisí na výšce stropu. Čím vyšší strop, tím dále senzor nainstalujete. Nepřijde mi moc praktické kvůli senzoru vrtat do stropu a žádná jiná varianta od výrobce není k dispozici. Senzor vysílá Bluetooth signál a má zónu detekce ve tvaru kužele pod senzorem. Když zónou projde zařízení, zařízení zabliká LED indikátorem a vás rozezná podle vašeho mobilu, který musí být se senzorem spárovaný.

RoomMe používá Open API, přes které dokáže pomocí vašeho mobilního zařízení odesílat data o vašem vstupu do místnosti nebo odchodu z místnosti na lokální IP adresu chytré domácnosti. Toto API funguje následujícím způsobem:

> Když osoba vstoupí do místnosti, API odešle událost vstupu do místnosti se jménem vstupujícího uživatele a také s jeho/její prioritou ovládání pro místnost. Když osoba opustí místnost, rozhraní API odešle událost opuštění místnosti

se jménem uživatele, který odešel, a také s uvedením, zda byl tento uživatel poslední osobou v místnosti.[4]

RoomMe lze objednat do České republiky z USA v balení po dvou kusech za cenu 5500 Kč, z této částky činí 1000 Kč poštovné. Tedy cena za jedno zařízení je 2250 Kč.

Zařízení jsem osobně testoval, takže spolehlivost mohu hodnotit jako velmi vysokou. Dokonce i v případě dvoupatrového obytného domu a senzorů umístěných přímo nad sebou na každém patře, senzor fungoval spolehlivě. Z hlediska hodnocení rychlosti bohužel musím uvést, že funguje rychle pouze při vstupu do místnosti. Skutečnost, že jste z místnosti odešli před pěti minutami nemá systém jak zjistit. Alespoň v případě mého testovaní to takto fungovalo. Rychlost detekování vstupu osoby se pohybovala okolo jedné až dvou sekund od průchodu pod senzorem, což je pro Bluetooth zařízení poměrně rychlé. Tyto výsledky odpovídají i údajům publikovaným například v rámci této recenze [2].

Instalace celého systému je jednoduchá i pro málo zkušeného domácího uživatele. Nejsou zapotřebí žádné programátorské schopnosti jako u většiny open source řešení. Na strop místnosti nainstalujete držák pro senzor a senzor na něj dáte obdobně jako stínidlo na světlo. Nainstalujete si aplikaci RoomMe App. Poté se pomocí této aplikace připojíte s využitím Bluetooth signálu k senzoru. Vytvoříte si uživatelský účet a přihlásíte se do aplikace jako administrátor. Poté můžete do aplikace přidat případně další uživatele. V aplikaci můžete nastavit, které chytrá zařízení a chytré systémy máte doma a nastavit scény (úlohy), které se mají udělat v okamžiku, když detekce zjistí příchod do místnosti.

Výhodou systému je nízká energetická náročnost na straně senzoru. Podle údajů výrobce vám na dvě baterie typu D vydrží zařízení funkční i dva roky. Z důvodu omezeného časového prostoru jsem výdrž baterie netestoval. Další pěknou funkcí v aplikaci je nastavení priorit u uživatelů s ohledem na používané automatizace. Správným nastavením například zabráníte, aby vám člen rodiny zhasl ve Vašem pokoji, když se v něm právě nacházíte. Nevýhodou celého systému je, že musíte mít mobil pořád při sobě nebo se Vám budou světla zhasínat kvůli špatně nastavené automatizaci při detekci odchodu.

RoomMe řešení se mi líbí hlavně z hlediska rychlosti a spolehlivosti. Také využití implementovaného Open API může být velmi zajímavé. Na druhou stranu mě plno věcí odrazuje od implementace tohoto systému v mé domácnosti. Cena je podle mě příliš vysoká. Baterie místo kabelu znamená pravidelnou kontrolu stavu baterií, i když zařízení by mělo na dvou bateriích vydržet funkční až dva roky. Celou řadu funkcí, které RoomMe nabízí jako zařízení s implementovaným smart hubem uživatelé s rozsáhlou chytrou domácností nevyužijí, protože již mají ostatní chytrá zařízení implementována v eco systému chytré domácnosti přes jiný smart hub než je RoomMe. Navíc RoomMe smart hub je dost omezující a umožňuje pouze automatizace při příchodu a odchodu, tudíž ho jako smart hub velmi pravděpodobně ani používat nebudete.

3.2 ESPresence

Jako příklad opensource řešení pro detekci prezence jsem testoval software vytvořený pro zařízení ESP32, pro použití s komponentou Home Assistant mqtt_room. Zařízení s tímto softwarem umístíte do všech pokojů, v kterých chcete detekovat prezenci. Zařízení detekuje prezenci pomocí síly signálu, tedy využívá metodu detekce blízkosti popsanou v 2.1. Jednoduchá řídící logika zde funguje následujícím způsobem. Podle toho, u jakého zařízení se nacházíte nejblíže, v té místnosti jste. Zařízení komunikuje s Home Assistantem pomocí MQTT.

Vlastní instalace systému je jednoduchá. Pomocí WiFi se připojíte k zařízení, otevře se vám prohlížeč, ve kterém nastavíte následující parametry: velikost místnosti, název zařízení, název lokální sítě a heslo lokální sítě. Pomocí Home Assistant MQTT Discovery se vaše zařízení jednoduše najde a automaticky nahraje do Home Assistant konfigurace. Software se umí sám aktualizovat za pomoci Github Release. Tato možnost jde samozřejmě i vypnout. Pro sledování ostatních zařízení používá fingerprint metodu místo mac adresy. Díky této technologii je schopen sledovat i zařízení s dynamickou mac adresou jako jsou některé mobilní telefony od společnosti Apple. Dále pro přesnější měření vzdálenosti využívá software implementovaný mediánový prefilter a Kalmanův filtr, jak je uvedeno na obrázku 3.1. Tento filtr snižuje negativní vliv šumu signálu s cílem dosažení větší přesnosti detekce.

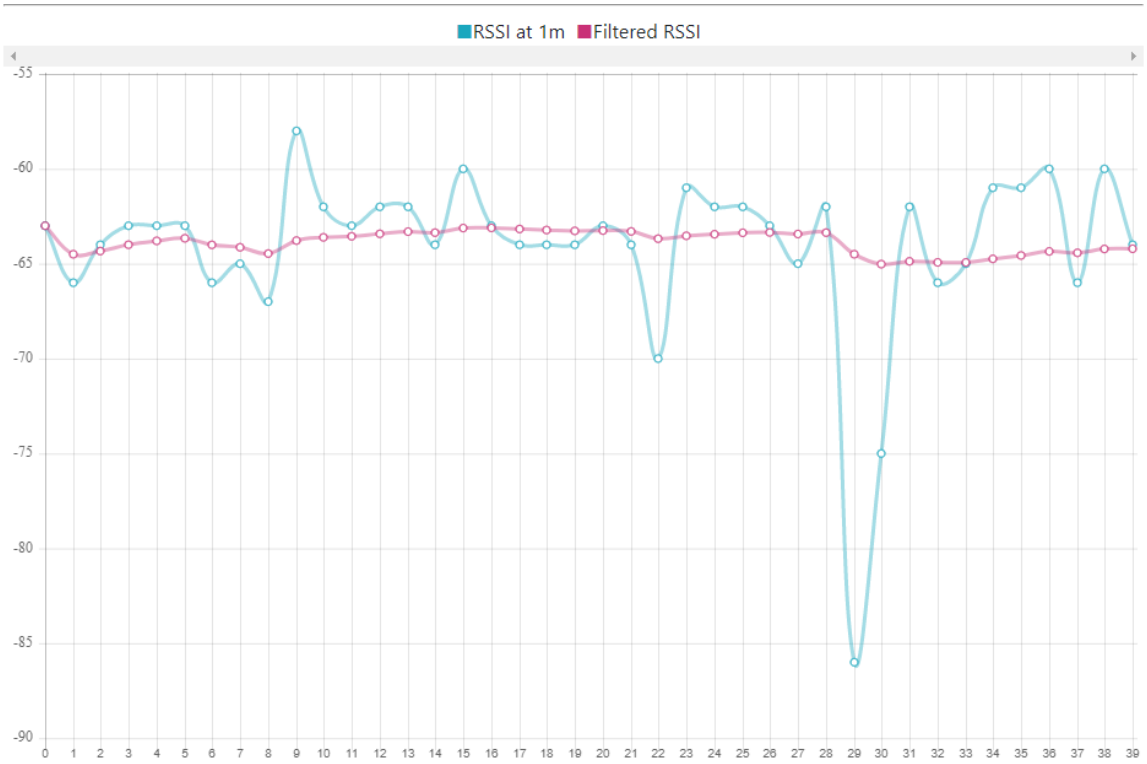
Senzor je doporučeno umístit na takové místo, aby detekoval sledovaná zařízení a zároveň byl umístěn co nejdál od dalších senzorů v chytré domácnosti. Řešení je postaveno na bázi open source. Tedy nulová cena softwaru a cena hardwarového řešení odpovídá ceně jednoho ESP32, která se v České republice pohybuje okolo 200 Kč. Jeden mikročip ESP32 potřebujeme pro každou jednu místnost. Oproti komerčnímu RoomMe máme k dispozici 10 krát levnější produkt, který nám poskytne velmi podobný výsledek. Samozřejmě ESP32 není smart hub, ale pro účely detekce přítomnosti je toto řešení dostačující.

Hlavní výhoda tohoto řešení spočívá v tom, že je velmi levné. Nepotřebujete žádnou aplikaci a stačí vám jakékoliv zařízení komunikující přes Bluetooth technologii jako například mobilní zařízení, chytré hodinky nebo Bluetooth tagy. Nevýhoda celého systému je jednoznačná, musí vám doma běžet MQTT server, který většinou zapnete na nějakém serveru s chytrou domácností. Bez Home Assistantu vám tohle řešení fungovat nebude. Je vytvořené tak, aby spolupracovalo právě s Home Assistantem.

Kalmanův filtr

Kalmanův filter je účinný rekurzivní filtr, který pomáhá najít správnou hodnotu signálu ze zašuměných měření v čase viz obrázek 3.1. Na obrázku je vidět, jak moc je RSSI signál zašuměný a proto je důležité přijímaný signál filtrovat. V našem případě použijeme Kalmanův filtr na filtrování RSSI. Hodnota RSSI udává výkon přijímaného rádiového signálu (měřeno v dB). Čím vyšší je hodnota RSSI, tím vyšší je síla signálu. V ideálním světě je hodnota RSSI závislá pouze na vzdálenosti mezi dvěma zařízeními. Ve skutečnosti jsou však hodnoty RSSI silně ovlivněny prostředím, kterým signál prochází a proto mají vysokou hladinu šumu [1]. Při testování a výpočtu vzdálenosti zařízení od senzoru jsem získával kvůli šumu velké odchylky naměřených hodnot a nebylo možné určit polohu zařízení. Po implementaci Kalmanova filtru bude možné získat přesnější signál RSSI a následně i přesněji odhadnout vzdálenost a tudíž i polohu zařízení. Bohužel pořád nedostatečně přesně na to, aby se poloha zařízení dala určit skutečně přesně.

Jedinou nevýhodou použití filtrace signálu pomocí Kalmanova filtru je skutečnost, že pro filtrování potřebujeme získat předem vzorek signálu v čase. Detekce přítomnosti bude z toho důvodu trošku opožděná. Musím zvolit a nastavit správný poměr mezi zpřesněním výsledku a zpožděním reakce. Pro implementaci Kalmanova filtru jsem musel přidat k naměřeným hodnotám časové razítko s informací, kdy byly hodnoty signálu naměřeny.



Obrázek 3.1: Vliv Kalmanova filtru na nezpracovaná data RSSI odebraná ze statického zařízení. Kalmanův filtr odstraňuje ze signálu velkou část šumu [1].

ESP32

Obrázek 3.2: Vývojová deska ESP32. Obrázek dostupný na [6]

ESP32 je vývojová deska s již integrovanou anténou a RF balunem, výkonovým zesilovačem, nízkošumovými zesilovači, filtry a modulem řízení spotřeby. Tato deska má navíc integrovaný dual-core processor s 2.4 GHz Wi-Fi a Bluetooth čipy s technologií TSMC 40nm s nízkou spotřebou.

Kapitola 4

Návrh řešení

Cílem práce je vytvořit detektor prezence, pomocí technologie Bluetooth, který bude schopen identifikovat konkrétního uživatele v definované zóně inteligentní budovy. Nastudoval jsem problematiku inteligentních zařízení, která umí detekovat prezenci pomocí Bluetooth radiofrekvenčního signálu. Seznámil se s existujícím komerčním řešením od Intellithings v sekci 3.1 a open source řešením ESPresence v sekci 3.2. Obě řešení jsem otestoval jak fungují.

Svým studiem jsem zjistil, že ideálním řešením pro tento problém by bylo využití Bluetooth Direction Finding, který byl hlavní přidanou vlastností v nové Bluetooth 5.1 technologii. Tato technologie dokáže určit, v kterém směru se zařízení nachází. Bohužel kvůli vysoké ceně čipů s touto technologií jsem se rozhodl pro levnější variantu využívající komponenty ESP32 popsané v sekci 3.2, které se používají v systému ESPresence.

Na základě získaných zkušeností jsem navrhl vlastní systém, který se inspiruje oběma zmíněnými systémy, jak komerčním, tak i open source řešením. Abych docílil podobného výsledku jako umožňuje Bluetooth Direction Finding technologie, bude mé řešení používat minimálně tři ESP32 v každé místnosti inteligentní budovy. Tyto tři senzory budou komunikovat skrze MQTT broker s webovou aplikací, která bude data zpracovávat a následně vyhodnotí, v jaké místnosti a v jaké definované zóně se konkrétní uživatel a chytré zařízení nachází. Aplikace bude umět data zobrazit ve webovém uživatelském rozhraní jako administrátorský panel nebo panel pro každou místnost zvlášť s její mapou. Na mapě půjdou vidět rozměry místnosti, definované zóny, pozice senzorů a pohyblivé zařízení podle odhadu jejich pozic. Zpracovaná data se odešlou pomocí MQTT do hubu chytré domácnosti, v mém případě se jedná o aplikaci Home Assistant server. V Home Assistant je možno realizovat téměř jakékoliv automatizace, které vás v chytré domácnosti napadnou.

Navržený systém jsem realizoval pomocí senzorů ESP32 a webové aplikace, která zpracovává data a zobrazuje uživatelské a administrativní rozhraní. Pro senzory jsem naprogramoval dva firmwary, které jsem pro jejich snadné rozlišení označil Gate a Room. Gate senzor, který bude umístěn u dveří, se chová jako vysílač i jako přijímač a odesílá, pomocí MQTT, názvy zařízení, které detekuje. Room senzor, který je ideálně umístěn v nějakém rohu pokoje, se chová pouze jako přijímač. Ukládá si data o síle signálu na zařízení a na Gate senzoru a časové razítko, kdy tato data byla zaznamenána. Zařízení, o kterých si informace ukládá Room senzor, jsou ta zařízení, která Gate senzor již detekoval a uložil do systému. Díky tomu budou mít všechny senzory, které patří k danému Gate senzoru, tedy do stejné místnosti, data o stejných zařízeních.

Systém, který bude detekovat zařízení v inteligentní budově a odhadovat jejich přibližnou polohu se bude skládat ze čtyř částí:

První část budou detektory pohybu neboli senzory, které budou zaznamenávat sílu signálu detekovaného zařízení a určí, zda se pohybuje a zda se nachází v blízkosti detektoru, tudíž je v místnosti využívána již zmíněná metoda detekce blízkosti 2.1.

Druhá část systému je MQTT broker, přes který se budou prvky tohoto systému domlouvat, posílat si důležité data jako je detekce zařízení, uživatele, informace o síle signálu a časová razítka měření hodnoty.

Třetí část systému bude webová aplikace, která bude zpracovávat data z detektorů uvedených v první části. Tato aplikace bude provádět veškerou logiku nad polohovým systémem. Webová aplikace bude také vizualizovat objekty v inteligentních mistnostech. Bude zobrazovat rozměry místnosti, pozice senzorů, sektory, tedy dílčí části místnosti, a pohyblivé zařízení.

A poslední, čtvrtá část systému je Home Assistant, na který bude webová aplikace posílat zpracovaná data a použije Home Assistant jako most (bridge) mezi inteligentními detektory a zbytkem chytré domácnosti. Zároveň v Home Assistantu bude možné se podívat na interaktivní mapy místností z webové aplikace.

4.1 Detektor pohybu

Hardwarová část detektoru pohybu je tvořena pouze vlastním modulem ESP32, protože je svými schopnostmi a funkčností pro naši jednoduchou detekci pohybu dostačující. Toto zařízení má integrovanou anténu s 2.4GHz Wi-Fi a Bluetooth. Zařízení bude statické, to znamená, že se neočekává žádný pohyb s přijímačem. Důležité je napájet detektor na požadovaných 5V napájecího napětí, jinak síla přijatého signálu nebude dostatečně přesná.

Detektory rozdělíme na 2 typy: hlavní a sekundární. Ty hlavní jsem pojmenoval Gate, protože se budou vyskytovat u vstupu do místnosti. Sekundární jsem pojmenoval Room, protože jsou umístěny v místnosti, nejlépe však v rozích. Měly by být cíleně rozmístěny podle toho, na které zóny pokoje se chcete zaměřit a dosáhnout tak u nich nejrychlejší a nejpřesnější odezvu. Poloha detektorů bude hrát velkou roli na vyhodnocení pozice zařízení, proto při testování je důležité, umístit detektory na dobře zvolené místo.

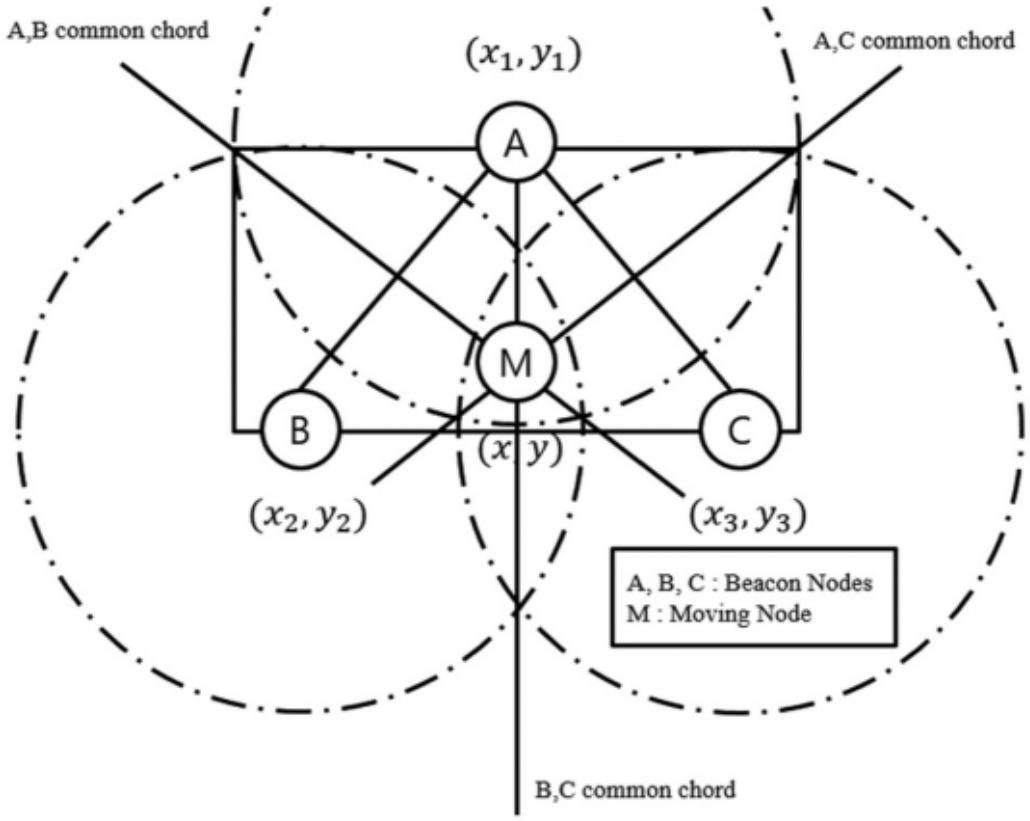
V místnosti potřebujeme rozmístnit více detektorů z toho důvodu, že nechceme jen detekovat, zda je zařízení v místnosti, ale také chceme znát, v jaké části místnosti se nachází. Rozhodl jsem se pro použití minimálně třech senzorů v místnosti z toho důvodu, že budeme detekovat přítomnost a polohu zařízení v 2D prostoru a to by s jedním nebo dvěmi přijímači nebylo možné. Pro detekci zařízení budeme používat jednoduchý matematický přístup, jak je popsáno na obrázku 4.1. Aby byly detektory co nejvíc přesné, chci na ně aplikovat již zmiňovaný Kalmanův filtr [1]. V případě, že by detektory nezvládaly dostatečně rychle zpracovávat přijímaný signál, dá se filtr použít před zpracováním dat ve webové aplikaci.

Oba typy detektorů, které ve svém řešení používán, využívají tabulku o zařízeních, do které si ukládají sílu signálu a tyto data pak posílají přes MQTT. Významný rozdíl mezi hlavním Gate senzorem a sekundárním Room senzorem je v tom, jak s danou tabulkou pracují. Gate senzor zmíněnou tabulku naplňuje názvy nebo-li identifikátory zařízení, které mají všechny detektory detekovat a zapisovat si jejich data o síle signálu. Room senzor jenom z tabulky umístěné v MQTT čte, které zařízení tam Gate senzor zapsal, ale dále do této tabulky nezapisuje nová zařízení.

Vzhledem k tomu, že oba typy senzorů používají jiný způsob práce s tabulkou zařízení, budou používat každý typ senzorů i jiný firmware. Firmware se liší také z důvodu, že Gate senzor se bude používat nejen jako přijímač signálu podobně jako Room senzor, ale na rozdíl od Room senzoru bude Gate senzor signál i vysílat. Toho budu využívat v programu, který

bude analyzovat sílu signálu pro detekci přítomnosti zařízení. Když porovnáme na Room senzoru sílu signálu detekovaného zařízení se silou signálu přicházejícího od Gate senzoru, zjistíme, zda-li je detekované zařízení blíže k Room senzoru než Gate senzor. Pokud to nastane, můžeme situaci vyhodnotit jako stav, kdy detekované zařízení je uvnitř kontrolované místnosti.

V prototypové fázi se parametry protokolu budou nastavovat přímo ve firmwaru. V této fázi nepokládám za vhodné věnovat čas vývoji konfigurace zařízení pomocí nějaké aplikace. Zařízení bude mít tyto parametry: název zařízení, název místnosti, v které se bude nacházet, o kolika detekovaných zařízeních se budou ukládat data a v neposlední řadě název hlavního Gate senzoru v místnosti, aby Room senzory mohly od něj převzít tabulku zařízení.



Obrázek 4.1: Triangulation using points crossed common chords. [5]

4.2 MQTT Broker

Jako instalaci brokeru jsem zvolil implementaci jednoho z již existujících open source řešení. MQTT je již standardem pro posílání zpráv v IoT. V našem případě budeme používat MQTT 3.1.1, protože ho podporují všechny technologie, které jsem si na tento projekt vybral. MQTT nebude šifrované, protože se jedná o prototypové řešení a šifrování by mohlo zpomalit detektory při zpracování signálů, při přijímání a odesílání zpráv.

MQTT funguje na principu topicu a zpráv. Vyberete si topic a ten začne odebírat (subscribe). Následně začnete dostávat všechny zprávy, které jsou odeslány na daný topic (publish). MQTT ještě používá řadu dalších funkcí jako je například QoS (Quality of Ser-

vice). Pomocí QoS lze například nastavit způsob kontroly zpráv. Lze definovat, jaké kontroly se budou provádět - zda se bude kontrolovat, jestli odeslaná zpráva přišla či přišla právě jednou. Lze definovat i režim, kdy vás nezajímá, jestli zpráva přišla a odesíláte takzvaně naslepo.

Pro aplikované řešení jsou podstatné tři základní pojmy: topic, subscribe a publish. Komunikace v našem systému bude využívat několik vytvořených topiců a subtopiců. Základním funkčním principem je vytvoření specifického topicu pro každou místnost. Poté každý senzor v dané místnosti bude mít svůj subtopic, který bude identifikován jménem senzoru. A jako poslední subtopic bude detekované zařízení, v kterém se bude ukládat zpráva o síle signálu. Webová aplikace bude všechny tyto topicy odebírat podle nastavení uživatele v aplikaci a dle získaných dat. Následně bude tato data zpracovávat a vizualizovat. Výsledná data odešle znovu přes MQTT na Home Assistant server, který s nimi dokáže pracovat jako například spouštět různé automatizace, scény.

4.3 Webová aplikace

Spolu s vedoucím bakalářské práce jsme se domluvili, že webová aplikace bude nejlepší řešení pro tento typ aplikace. A chci využívat technologie, které mě baví a chci se v nich zlepšovat. Proto jsem si na aplikační logiku vybral programovací jazyk Go, který je rychlý a má nativně implementované pomocné knihovny, jak pro MQTT, tak i pro matematické a geometrické funkce, které se budou využívat při detekci pozice. Ve webové aplikaci musí být vytvořeno databázové úložiště, kde budeme ukládat veškeré data o uživatelích, pokojích, senzorech a zařízeních. Vybral jsem si platformu MongoDB, protože je to NoSQL jazyk, který je rovněž dobře implementován do jazyka Go. To zajistí dobrou kompatibilitu mezi jazykem Go a MongoDB. Další výhodou jazyka Go je JSON-like formátování, což je užitečné hlavně při práci s JavaScriptem. Zajímavá je i možnost využití hostingu serveru pro malé aplikace zadarmo na MongoDB Cloud.

Poslední a nejdůležitější část naší aplikace bude tvořit JavaScriptový framework React.js, který mi přijde na tuhle aplikaci ideální hlavně kvůli způsobu, jakým chceme data vizualizovat. Aplikace bude zobrazovat interaktivní mapu o rozměru místnosti, s interaktivními prvky jako budou senzory, části místnosti a pohyblivé zařízení.

Webová aplikace bude zpracovávat data, která však musíme nejdříve získat pomocí MQTT. Uživatel této webové aplikace bude moci vytvořit pokoj nebo víc pokojů. Dále bude moci nastavit mu/jim základní parametry jako jsou délky zdí, tvar pokoje, určité zóny (část pokoje), název a MQTT topic. V uživatelském rozhraní pokoje bude možné pokoj upravit, smazat nebo do něj vložit senzory. U senzorů bude možné nastavit jeho pozici, typ (zda se jedná o Gate senzor nebo Room senzor), název a MQTT topic. Senzory i po definování v aplikaci bude možno později upravit či změnit jejich pozici, popřípadě jiné parametry. Díky této struktuře bude aplikace vědět, které MQTT topicy má odebírat.

Topicy se budou vytvářet následujícím způsobem:

mqtt_room/<název_místnosti>/<název_senzoru>/<název_zařízení>

Při odběru takového topicu bude aplikace získavat data o síle signálu mezi zmíněným zařízením a zmíněným senzorem. Poté si aplikace získaná data uloží do objektu senzoru. Při dostatku potřebných dat bude aplikace schopná odhadnout polohu zařízení a tudíž i určit, v jaké oblasti se zařízení nachází.

Získaná data bude možné vizualizovat do interaktivních map místností. Abychom ve webové aplikaci mohli zobrazit, kdo v místnosti je, bude nutno vytvořit uživatele. Přímo v aplikaci vytvoříme uživatele, přidělíme mu jednoznačné jméno a zařízení, které jsme nalezli

pomocí MQTT. Uživatel si dále bude moci vybrat, které zařízení k němu patří a zda ho chce trackovat (to zařízení, které má v ruce). Může si nastavit i režim, kdy nechce být trackován vůbec, aby se zbytečně nespouštěly nastavené automatizace.

Kombinací polohy zařízení a zařízení vybraného uživatelem, můžeme vytvářet výsledná data o tom, v jaké zóně inteligentní budovy se uživatel nachází. Tato data aplikace pomocí MQTT odešle do chytré domácnosti, v našem případě do již zmiňované aplikace Home Assistant.

4.4 Home Assistant

Home Assistant je centrální řídicí systém pro chytrou domácnost, který je bezplatný a open source. Home Assistant podporuje řadu komerčních chytrých zařízení díky jeho široké a pracovité vývojářské komunitě, která je pomáhá integrovat. Je samozřejmostí, že umí komunikovat s MQTT serverem, dokonce můžete MQTT server na něm zprovoznit pomocí doplňku (addonu).

S Home Assistantem mám mnoho osobních praktických zkušeností, proto jsem si ho vybral jako most (bridge) pro spojení webové aplikace s dalšími zařízeními v inteligentní budově.

V mém případě Home Assistant poběží na linuxovém serveru v dockeru a budu ho používat k spouštění automatizací a scén při vstupu do nebo odchodu z různých zón. Dále je možné některá data z Home Assistantu vizualizovat ve webové aplikaci jako například teplotu v místnostech.

Kapitola 5

Realizace řešení

Realizované řešení se od původního návrhu částečně liší ve způsobu lokalizace zařízení v sektoru místnosti. Ke změnám jsem přistoupil na základě praktických zkušeností získaných během testování zařízení, neboť jsem zjistil, že síla signálu Bluetooth RSSI se prudce mění vlivem různých ruchů pocházejících z jiných okolních Bluetooth zařízení. Z toho důvodu jsem na odhad 2D pozice v místnosti nepoužil metodu triangulace, i když jsem ji implementoval. Místo toho jsem se rozhodl pro metodu detekce blízkosti, kdy detekuji postupné přibližování k nejbližšímu senzoru. Při vstupu do místnosti se odhadne vaše pozice uprostřed místnosti a následně se spustí časová iterace, kdy každou vteřinu se detekovaná poloha zařízení mění směrem k senzoru, který přijímá nejsilnější signál.

V prvním kroku jsem naprogramoval programy pro ESP Gate a pro ESP Room, v kterých se získávala a posílala jenom surová data. Vzhledem k tomu, že k použití Kalmanova filtru je také důležitá časová značka identifikující, kdy se tato data získala, programy jsem upravil pro práci s časovou značkou. Poté jsem naprogramoval druhou část systému, tedy webovou aplikaci, která administrátorovi nebo správci chytré domácnosti umožní nastavení parametrů pokojů, zařízení, senzorů a sektorů. Lze tak nastavovat parametry od velikosti a pozice až po barvu a název objektu. Všechny atributy objektů neboli struktur lze nalézt v balíčku models 5.2.1.

Jako poslední část jsem nastavil senzory v konfiguraci Home Assistantu, aby se data správně ukazovala a mohl s nimi uživatel chytré domácnosti bez problému pracovat. Zároveň v Home Assistantu lze využívat interaktivní mapu pro sledování zařízení, přidávání, mazání a upravování senzorů a sektorů jak ukazuje tento obrázek 5.8.

5.1 Detektor pohybu

Detektor pohybu je realizovaný pomocí ESP32 3.2. Tento mikročip je pro naše detektory funkčně i výkonově dostačující. Vzhledem k tomu, že požadavek na přesnost detekce vyžaduje použití více detektorů v jedné místnosti, je i otázka ceny při volbě řešení podstatná. Chceme, aby náš hardware byl levná záležitost.

Naprogramoval jsem program pro detektor pohybu v jazyce C. Ve frameworku Arduino pomocí vývojového prostředí Visual Studio Code a rozšíření PlatformIO. Program je rozdělený na dvě části: Gate senzor a Room senzor. Gate senzor je v místnosti pouze jeden a pro nejlepší funkčnost logiky systému detekce je vhodné jej umístit u dveří. Room senzor je vhodné umístit do rohu pokoje nebo do strategického místa, kde chcete mít jistotu, že

vstup do daného sektoru pokoje bude rychle a spolehlivě oznámen aplikaci.

5.1.1 Použité knihovny, popis a důvod použití

- Arduino framework, který nám umožní nahrát program a komunikovat s ostatními knihovnami. Umožňuje také ovládat LED diody a tlačítka jako například tlačítko na restart zařízení.
- WiFi knihovna, která nám umožní se připojit k WiFi síti a internetu. Potřebujeme ji ke komunikaci s MQTT serverem a NTP serverem. Co je NTP server a k čemu slouží vysvětluji později u popisu NTPClient knihovny.
- AsyncMqttClient knihovna, která umožňuje připojit se k MQTT brokeru, tedy serveru. Umožňuje odesílat a přijímat zprávy pomocí Publish a Subscribe.

  Async v názvu knihovny znamená, že je asynchronní. Znamená to, že je jedno, co právě program dělá, ale v okamžiku kdy přijde zpráva z odebíraného topicu, tak ji začne client zpracovávat. Kdyby program nepracoval v asynchronním režimu, tak se musí vždy čekat na program až se dostane ve smyčce k funkci „přečti si přijaté zprávy“. Pro naší aplikaci detekce v reálném čase potřebujeme asynchronní přístup.
- Ticker třída nám umožní měřit čas od nějaké akce. V mém případě třídu Ticker používám při Reconnectu k Mqtt serveru. Když je ticker 2 sekundy aktivní, tak zavolá funkci connectToMqtt. Samozřejmě se také zaktivuje, když se AsyncMqttClient odpojí od MQTT brokeru.
- NimBLEDevice knihovna se používá pro inicializaci zařízení jako Bluetooth low energy zařízení.
- NimBLEAdvertisedDevice knihovna je určena pro odesílání dat a přijímání respektive skenování okolních zařízení se spuštěnou Bluetooth radiofrekvenční komunikací. Skenuje tedy přítomnost zařízení, které chceme detekovat.
- WiFiUdp and NTPClient jsou knihovny, které potřebujeme na získání NTP z lokální WiFi sítě. NTP je zkratka pro Network Time Protocol a používá ho každý router (směrovač), který je připojený k nějakému NTP serveru. Díky tomuto protokolu všechna spolukomunikující zařízení ví, jaký je společný čas a tedy dokážou udělat synchronizovanou časovou značku akce, kterou provedla. Tato možnost je u našeho detektoru velmi důležitá, neboť jinak by nebylo možno použít Kalmanův filtr 3.2, který nám pomáhá k odstranění šumu a chyb v získaných datech.

  Existující řešení ESPresence 3.2 používá Kalmanův filtr přímo v programu ESP. V mém realizovaném řešení se tato úloha řeší ve webové aplikaci, do které se odesílají data s časovou značkou.

5.1.2 Nastavení detektoru

Na každém zařízení se musí nastavit řada základních parametrů jako konstantní hodnoty, aby zařízení bylo schopné se připojit k WiFi a MQTT brokeru, aby využívalo příslušný topic a správně fungovalo. Tyto údaje by se daly vyplnit automaticky pomocí aplikace při připojení mobilního zařízení na detektor. Z časových důvodů jsem u mého prototypového

řešení tuto funkci v aplikaci neimplementoval, ale u široce používaného řešení by bylo vhodné tuto funkci implementovat.

- WIFI_SSID nastaví název WiFi sítě, na které chcete tato zařízení používat. Většina domácností má pouze jednu WiFi síť.
- WIFI_PASSWORD nastaví heslo k zmiňované WiFi síti.
- MQTT_HOST nastaví IP adresu MQTT brokeru. Na tento broker musí být povoleno připojení z vaší lokální sítě, jinak aplikace nebude fungovat.
- MQTT_PORT nastaví port k danému MQTT brokeru.
- DEVICE_NAME název a tedy i MQTT topic, na kterém bude zařízení posílat zprávy.
- GATE_NAME název a tedy i MQTT topic, na kterém posílá zprávy GATE senzor, tedy hlavní senzor v pokoji. Nachází se u dveří.
- ROOM_NAME název pokoje, v kterém se Room senzor nachází.
- LED_BUILTIN Pin, na který je připojena led dioda. Dioda v mé aplikaci pouze signalizuje, zda je zařízení v provozu.
- NUMBER_OF_DEVICES počet zařízení, která si zařízení uloží do své tabulky. Nastaveno na 10.
- SERVICE_UUID a CHARACTERISTIC_UUID jsou UUID 128-bitové hodnoty, které by měly být pro každé zařízení náhodně vygenerované. Poté můžete tyto Bluetooth služby sledovat pomocí aplikací jako například nRF Connect nebo nRF Toolbox.

5.1.3 Program

Program v Arduino frameworku se vždy skládá ze 2 hlavních částí: setup a loop. První část programu, setup, se vykoná jenom při prvním spuštění zařízení a tedy před prvním loopem. V setupu se nastaví všechny důležité proměnné a funkce tak, aby v loopu fungovaly správně. Loop program se opakuje pořád dokola, dokud loop nepřerušíme například restartem nebo vypnutím zařízení.

V setupu nastavuji pin, na kterém se nachází led dioda na OUTPUT, tedy výstupní pin. Poté se zde definuje rychlost, na které bude mikročip pracovat a jak rychle se na něj bude daný program nahrávat při flashování paměti.

Je nutno nastavit callback funkci, která se provede, když se zařízení připojí k WiFi. Tato aplikace pošle na mqtt server topic pokoje, název zařízení a jeho IP adresu. Následně se nastaví název Gate senzoru podle údaje uvedeného v nastavení GATE_NAME.

Poté se nastaví callback funkce pro asynchronní MQTT klient, která spravuje události zařízení, když se připojíme k MQTT serveru, když se odpojíme od serveru, když odešleme nebo přijmeme zprávu a když začneme nebo přestaneme odebírat topic. Nastavíme MQTT server hodnotami podle nastavení parametrů MQTT_HOST a MQTT_PORT. Poté se MQTT server připojí k WiFi síti.

Dále se v setupu nastaví zařízení pomocí parametrů funkce init z knihovny BLEDevice. Vytvoří se skener, který hledá všechna zařízení, na které se dá připojit pomocí Bluetooth. Nastavíme callback funkci popisující, co se má dít, když skener nalezne nějaké zařízení. Nastavíme sken do režimu active, který si sice odebírá více energie z napájení, ale je rychlejší.

Vytvoříme Bluetooth Low Energy server a spustím jeho službu, kterou používáme pro odesílání skenování Bluetooth zařízení a sběr jejich dat. Tato služba se nazývá Inzerce, anglicky Advertising. Následně spustíme Advertising.

A jako poslední věc v setupu vytvoříme časového klienta, který se připojí na WiFi a získává informace o aktuálním čase. V této části programu tedy používáme NTP časový server poskytující časovou značku.

Loop program začíná tím, že si uloží aktuální systémový čas, to znamená čas od zapnutí zařízení. Každých dvacet loopů se zavolá funkce časového klienta, která aktualizuje čas podle NTP serveru.

V této části programu se smyčka loop liší v Room programu a Gate programu. Gate program má první dvě smyčky prodloužené o 10 sekund z důvodu prvních skenů sloužících k vytvoření tabulky zařízení, které posílá na MQTT broker. Room program žádné prodloužení nemá.

V dalším bloku programu loop začneme hledat zařízení pomocí skener funkce po dobu scanTime. V Room programu je nastavena tato hodnota na tři sekundy. V Gate programu je nutno zajistit provedení ještě dalších kroků (jako například zápis do databáze) a proto je nutné skenovat pouze po kratší dobu. Tato hodnota je u Gate programu nastavena na jednu sekundu.

Vždy, když skener nalezne Bluetooth zařízení, zavolá se callback funkce. Tato funkce začíná kontrolou, jestli se jedná o Gate senzor nebo o zařízení, které má data o výrobci. Jestliže se nejedná ani o jeden typ zařízení, přijatá data se ignorují. Jestliže se jedná o Room senzor, tak se vytváří fingerprint, který slouží jako identifikační značka. Jestliže se jedná o Gate senzor, tak se místo fingerprintu použije název senzoru. Poté se získává údaj o RSSI síle signálu zařízení, který se ukládá do tabulky podle názvu zařízení. Z této tabulky na základě názvu zařízení dostaneme medián, tedy střední hodnotu z posledních tří naměřených sil signálu. Tuto střední hodnotu spolu s časovou značkou odešleme pomocí MQTT na MQTT broker.

Po provedení skenu a odeslání naměřených údajů vymažeme nalezené zařízení a čekáme, dokud se systémový čas nebude rovnat uloženému času z prvního kroku prodloužený o nastavenou periodu. Perioda je nastavená na 3300 milisekund.

Při připojení k MQTT serveru se začne odebírat topic pokoje z nastavení ROOM_NAME a subtopic enter. A odešle se IP adresa zařízení na topic pokoje a subtopicu zařízení. Při odpojení od MQTT serveru se zapne reconnect timer.

Vždy při příchodu zprávy z jednoho z odebíraných topiců víme, že se jedná o název zařízení. Jestliže zařízení s tímto názvem ještě nemáme uložené v tabulce, tak zařízení vytvoříme a uložíme do tabulky.

V případě Gate senzoru název zařízení vždy pošleme na již zmíněný topic pokoje z nastavení ROOM_NAME a subtopic enter.

5.2 Webová aplikace

Původně jsem aplikaci psal jako desktopovou pomocí frameworku Wails. Wails je framework pro Go, který pro tvorbu uživatelského rozhraní UI používá webové technologie jako jsou Vue.js nebo React. Po konzultacích s mým vedoucím práce jsme usoudili, že by větší smysl dávalo udělat webovou aplikaci. Z toho důvodu jsem backend nechal napsaný v Go, přidal jsem MongoDB databázi a přepsal jsem UI desktopové aplikace do Reactu. V této sekci se zaměřím na popis hlavních částí webové aplikace: na backend a jeho členění, frontend a databázi.

5.2.1 Backend

Backend webové aplikace je napsaný v programovacím jazyce Go. Go je kompilovaný multiparadigmatický programovací jazyk vytvořený v Google Inc. v roce 2007. Jeho původní autoři jsou Robert Griesemer, Rob Pike a Ken Thompson [7]. Aplikaci jsem rozdělil na balíčky podle toho, jaké mají funkce a metody. Jednotlivé balíčky vám podrobně popíšu v následující části práce:

- main - hlavní balíček, kde se nachází začátek aplikace. Funkce main načte uloženou konfiguraci v souboru `app.env`. Následně vytvoří MongoDB klienta, který se připojí na vzdálenou databázi. Uloží přístup do databáze do své globální proměnné. Inicializuje MQTT klienta, který začne odebírat topic device. V této části programu se také nastaví websocket pro komunikaci s frontendovou částí aplikace. Vytvoří se zde také asynchronní smyčka, která aktualizuje zařízení každé 2 sekundy.

- util - V tomto balíčku se nachází struktura konfigurace, která se načítá ze souboru `app.env`. Pro danou funkci využívám populární knihovnu viper od uživatele spf13. Konfigurace aplikace se skládá ze zdroje pro připojení k databázi, IP adresa a port MQTT brokeru a názvy Gate senzorů oddělené čárkou.

- db - Balíček obsahující globální proměnné, které umožňují přístup do MongoDB databáze. Jako globální proměnné jsou také uložena všechna zařízení, které aplikace vytvořila z toho důvodu, že atributy zařízení se často mění jako například pozice či v kterém pokoji nebo sektoru se zařízení nachází. Tato data se proto neukládají do MongoDB databáze, ale pouze do globální proměnné v aplikaci. Další bližší informace uvedu v popisu balíčku models.

- mqtt - V tomto balíčku se nachází všechny potřebné funkce a metody pro komunikaci s MQTT brokerem. Ve svém řešení používám Paho mqtt knihovnu od Eclipse. Dále je zde uložena struktura pro mého MQTT klienta obohaceného o metodu pro posílání mqtt zpráv. Je zde také vytvořena struktura payloadu zpráv, protože mé detektory posílají JSON zprávy ve formátu {"rssi":-44","time":"17:31:38"}. V tomto balíčku jsou definovány funkce pro zpracování zprávy, odeslání zprávy, připojení, ztráta připojení, začátek odebrání jednoho nebo více topiců a inicializace MQTT klienta, která se volá ve funkci main.

- models - V tomto balíčku jsou definovány všechny používané struktury a metody pro objekty uložené v MongoDB nebo používané ve frontendu.

  Jako jednu ze základních struktur aplikovaných v mém řešení používám strukturu pro **Room**, která má identifikator, název, MQTT topic a rozměry. Rozměry pokoje jsou určené pozicemi rohů pokoje.

Další struktura **Sector** je vlastně menší pokoj, který používá ještě jednu pozici, která se váže na levý horní roh rozměrů sektoru a má atribut barvy, pro odlišení v uživatelském rozhraní.

Další struktura je **Sensor**, která používá atributy identifikátor, název, MQTT topic, pozici a barvu, ale nemá rozměry. Jedná se vlastně pouze o bod v pokoji. Tyto tři struktury (pokoj, sektor, senzor) se ukládají do MongoDB databáze, protože se jejich atributy často nemění a chceme si tyto údaje uchovat i při restartu aplikace.

Struktura **Signal** obsahuje MQTT topic, který definuje následující parametry signálu: k jakému senzoru patří, RSSI síla signálu, čas a Kalmanův model a Kalmanův filter. Informace o parametrech Topic, síla signálu a čas získáme z MQTT zprávy.

Struktura **Device** má definované dvě metody: AddSignal a UpdateSignalRSSI. V metodě AddSignal se vytvoří nový Kalmanův model podle času, síly signálu a jednoduché konfigurace modelu. Potom se vytvoří Kalmanův filtr na základě tohoto modelu. A jako poslední se vytvoří struktura signálu, do které se uloží parametry jako topic senzoru, síla signálu, čas, Kalmanův model a filtr. Tato struktura se uloží do pole signálu daného zařízení.

Device je nejkomplikovanější struktura, která využívá řadu metod, protože se neukládá v MongoDB databázi. Struktura se skládá z identifikátoru, názvu zařízení, identifikátor pokoje, název pokoje, v kterém se zařízení nachází a název sektoru, v kterém se nachází. V této proměnné je uložen název pokoje, jestliže v žádném sektoru zařízení není. Dále má zařízení uložené již zmíněné pole signálu, pozici, identifikátor nastavení zařízení. Nastavení zařízení vysvětlím v dalším odstavci. Posledními atributy jsou duplicitní s nastavením zařízení a jedná se o atribut uživatele, tedy vlastníka zařízení. Důležitými atributy této struktury jsou Schovat zařízení a Sledovat zařízení. Když je atribut schovat nastavený na pravda, tak zařízení není vidět v uživatelském rozhraní, ani se neposílají zprávy o zařízení na MQTT broker.Když je atribut sledovat nastavený na pravda, tak se na MQTT broker posílají zprávy pod názvem uživatele sledovaného zařízení. Zařízení se ukládají do globální proměnné z balíčku db. Struktura má definované metody pro správu signálu, které umožňují smazání, získání, přidání nebo aktualizace signálu. Lze zde využívat i metody pro aktualizace pokoje a sektoru, v kterem se zařízení nachází a nebo uložení nastavení ze struktury nastavení zařízení.

Device setting je poslední struktura, která se ukládá do MongoDB databáze a používá atributy jako identifikátor, název, identifikátor zařízení, uživatel, schovat zařízení a sledovat zařízení. Většina atributů se nastavuje v uživatelském rozhraní.

- dao - V softwaru je k dispozici objekt pro přístup k datům (Data access object). Jedná se o balíček s funkcemi CRUD pro mé objekty: vytvořit (create), číst (read), editovat (update), smazat (delete). Funkce z tohoto balíčku jsou často využívány HandleMessege metodou z websocket balíčku, o kterém se více zmíním dále.

  Funkce používané z tohoto balíčku komunikují s MongoDB clientem a získávají nebo upravují data v databázi. Jsou zde k dispozici funkce pro všechny typy objektů neboli struktur, které využívám v programu jako jsou pokoj, senzor, sektor a nastavení zařízení.

V programu jsem vytvořil funkci, která nastavuje, v které místnosti a sektoru se nachází dané zařízení. Tato funkce je volána v cyklu bloku programu main a volá další funkce v balíčku models.

Logika fungování programu je následující. Nejdříve pomocí programu zjistím informace o přijatých signálech od detekovaného zařízení na všech senzorech. Podle naměřené síly signálu předpokládám, že se zařízení nachází v pokoji, kde je umístěn senzor s nejsilnějším signálem od daného zařízení. Poté si zjistím, které senzory se v daném pokoji nachází a jestli signály přicházející na tyto senzory jsou nejsilnější ze všech ostatních signálů, které zařízení vysílá. Tímto způsobem je potvrzeno, že se zařízení nachází v daném pokoji. Kvůli rychlosti celé aplikace jsem na základě provedených testů nastavil v aplikaci zjednodušující pravidlo, kde předpokládám, že když je síla signálu u nejsilnějšího signálu vyšší než -60 dB, tak je senzor blíže než jeden metr od zařízení a tudíž je v dané místnosti.

V aplikaci také zjišťuji, jestli zařízení neopustilo danou místnost. V této části aplikace porovnávám síly signálů od Gate senzoru naměřených na Room senzorech vůči síle signálu od zařízení naměřených na Room senzorech. Pokud je přijímaný signál od zařízení slabší než přijatý signál od Gate senzoru, potom zařízení opustilo pokoj. Z důvodu velikosti pokoje se dá síla signálu na hlavní senzor měnit o desítky procent pomocí pokojového atributu sensitivity (citlivost).

- websocket - websocket je základní nástroj, který využívám ke komunikaci mezi frontendem a backendem této aplikace. V balíčku websocket používám knihovnu od Gorilla. Gorilla je webová sada nástrojů pro programovací jazyk Go, která poskytuje užitečné, navzájem propojitelné balíčky pro psaní aplikací založených na HTTP[3]. V balíčku je funkce, která vytvoří websocket. V rámci balíčku používám několik základních struktur. Struktura pool, do které se ukládá každý klient této aplikace, který otevře webovou stránku nebo její novou záložku. Struktura klient, která má nadefinované metody na přijímání, zpracování a odesílání dat skrz websocket. Struktura zprávy, s kterou pracuje klient, se skládá z typu a z těla zprávy. Typ zprávy v mé aplikaci nepoužívám. Tělo zprávy používám strukturované do struktury objekt, která obsahuje identifikátor a typ objektu jako je pokoj, senzor, zařízení, sektor, signál. Ve zprávě je též uvedena operace objektu jako je přidat (insert), získat (get), editovat (update), smazat (delete).

- position - Balíček Position slouží k počítání pozice zařízení. Využívá čtyři základní struktury: pozice, kruh, úsečka a vektor. Tyto struktury a metody jsem použil pro výpočet pozice zařízení. Bohužel z důvodu rychle se měnící síly signálu od zařízení jsem nebyl úspěšný ve snaze vytvořit takovou funkci, která by nevracela příliš mnoho falešných pozic. Proto jsem se po řadě provedených testů rozhodl pro určení pozice na základě odhadu místo čistého výpočtu. Tato řešení není složité a jistě existují další možnosti jeho postupného vylepšení.

  Funkce se jmenuje UpdateDevicePosition a posílám do ní informace o aktuální pozici zařízení a pozice senzorů s přijatými signály od zařízení. Počáteční pozice zařízení v místnosti je vždy definována u nově detekovaného zařízení ve středu místnosti. Funkce si senzory seřadí podle síly přijatého signálu od zařízení od nejsilnějšího signálu po nejslabší. Poté použije metody na získání vzdáleností a vektorů z pozice zařízení a z pozice senzoru, aby mohla určit novou pozici zařízení. Tuto metodu jsem

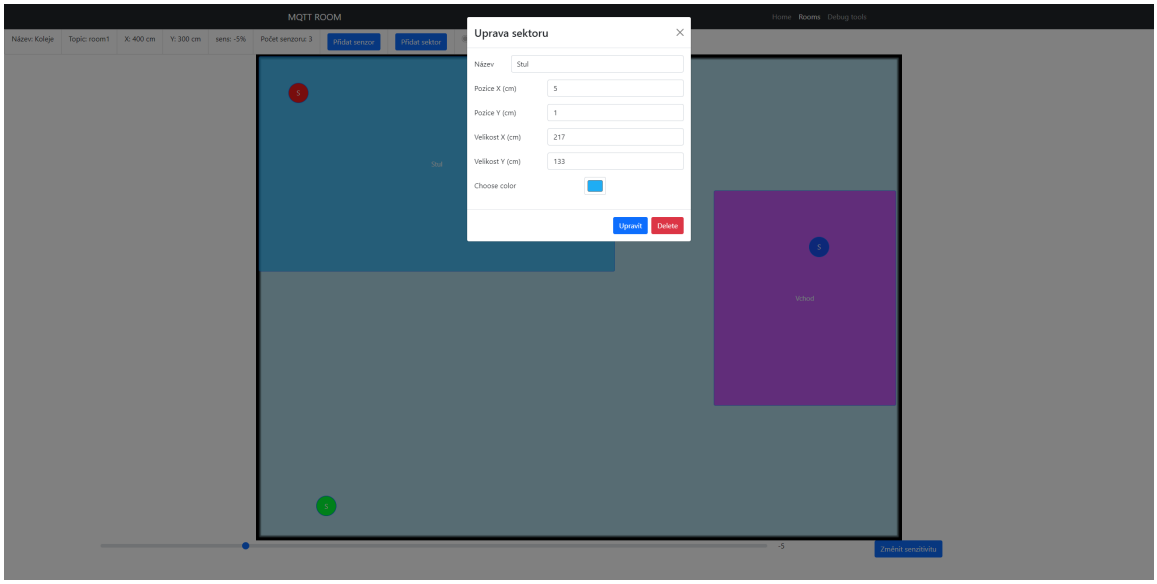
navrhl následujícím způsobem. Každou iteraci neboli zavolání funkce se zařízení posune o polovinu vzdálenosti směrem k senzoru s nejsilnějším signálem a o desetinu vzdálenosti směrem k senzoru s druhým nejsilnějším signálem. Vzhledem k použité iterační metodě je důležité, aby každý sektor, v kterém chci zařízení detekovat, obsahoval alespoň jeden senzor.

Backend webové aplikace získává data primárně z detektorů přes MQTT broker, sekundárně také z frontendu od uživatele. Zpracovaná data odesílá na frontend pomocí websocketu a na MQTT broker, kde si může data zpracovat libovolný připojený systém jako například v mém případě Home Assistant.

5.2.2 Frontend

Uživatelské rozhraní umožnuje uživateli zadávat požadavky na aplikaci a vkládat data potřebná k fungování aplikace. Na tvorbu uživatelského rozhraní jsem použil JavaScriptovou knihovnu React. React efektivně aktualizuje a vykresluje ty komponenty, které je nutno aktualizovat, když dojde ke změně dat. Části webové stránky se píšou do komponent tak, abychom je mohli snadno znovu vygenerovat ve webovém prohlížeči. Toho jsem využil například při zobrazování určeného pokoje pomocí URL.

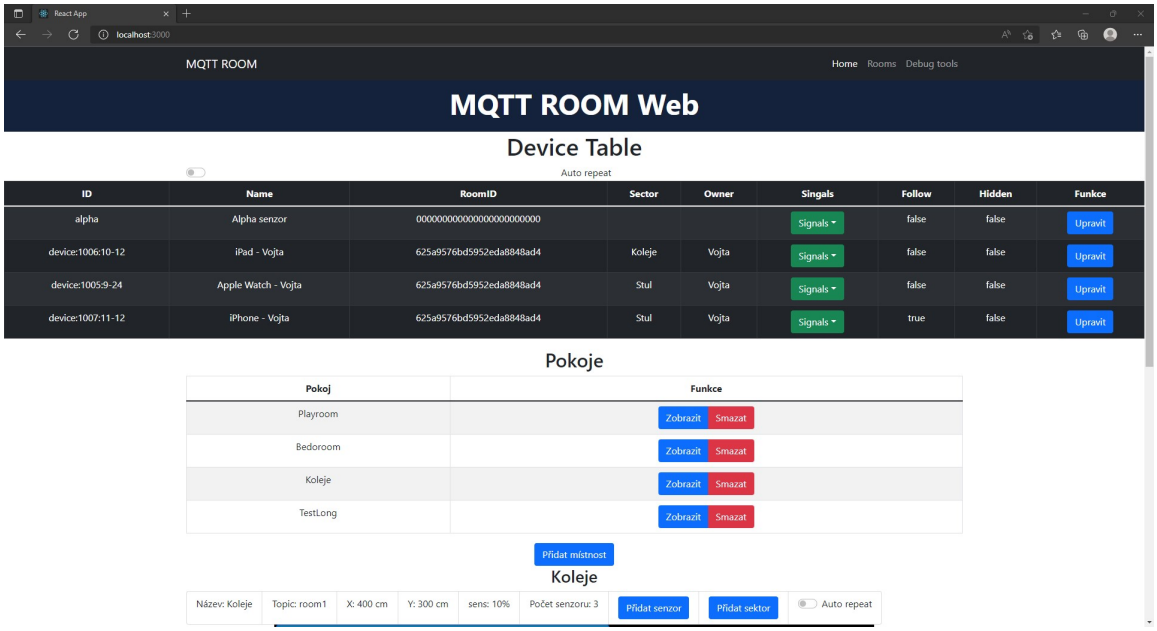
Mým cílem bylo vytvořit uživatelsky přívětivé grafické rozhraní, které bude umožňovat konfiguraci místnosti uživatelem jednoduchým a intuitivním způsobem. Navržené rozhraní lze ovládat pouze myší a klávesnice slouží pouze k zadání základních parametrů jako jsou například rozměry místnosti či velikost jednotlivých sektorů. U všech používaných prvků jako jsou místnost, sektor či senzor lze přes toto grafické rozhraní zvolit barvu, rozměry a umístění prvku v rámci místnosti. Tím se vytváří velmi jednoduchý model místnosti, na kterém lze velmi snadno identifikovat případné chyby zadání jako je například špatná poloha senzoru. Data jsou zadávána pomocí vyskakovacích oken tohoto rozhraní, jak je možno vidět na obrázku 5.1.



Obrázek 5.1: Vyskakovací okno pro zadání parametrů pro sektor stul.

Aplikace je rozdělena na dva hlavní pohledy a jeden vedlejší pohled určený pro účely možnosti snadného ladění programu. První hlavní pohled je administrátorský určený pro

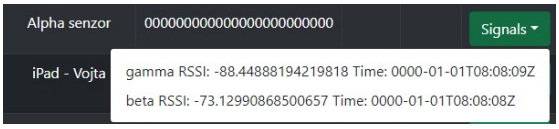
správu administrátora nebo správce chytré domácnosti. Zde jsou vidět všechny definované objekty viz obrázky 5.2 a 5.4. V tomto pohledu je možno objekty vytvořit či odstranit a lze zde také nastavovat různé atributy objektů.



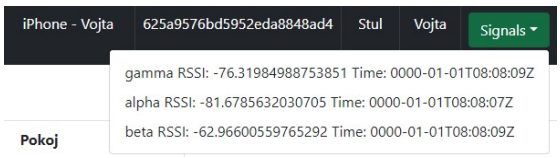
Obrázek 5.2: První část administrátorského pohledu

Administrátorský pohled začíná tabulkou zařízení, kde se zobrazují všechna zařízení, která mají atribut 'schovaný' nastavený na hodnotu nepravda. Zařízení nemá smysl odstraňovat, protože by se znovu vytvořila a museli bychom mít další tabulku smazaných zařízení, což nedává smysl. Další atributy zobrazené na obrázku jsou popsány v odstavci 5.2.1 v popisu balíčku models. Signály jsou vytvořeny jako dropdown menu.

Při rozvinutí dropdown menu Signals aplikace zobrazí informace o naměřených signálech. Z obrázku 5.3 zobrazujícímu ukázky signálů například vidíme, že síla signálu od iPhonu naměřená na Room senzoru gamma i beta je vyšší než naměřený signál alpha od Gate senzoru , tedy iPhone je blíže k těmto senzorům než Gate senzor alpha. Z toho vyplývá, že se zařízení nachází v pokoji.



(a) Ukázka signálů na zařízení alpha (Gate senzor)



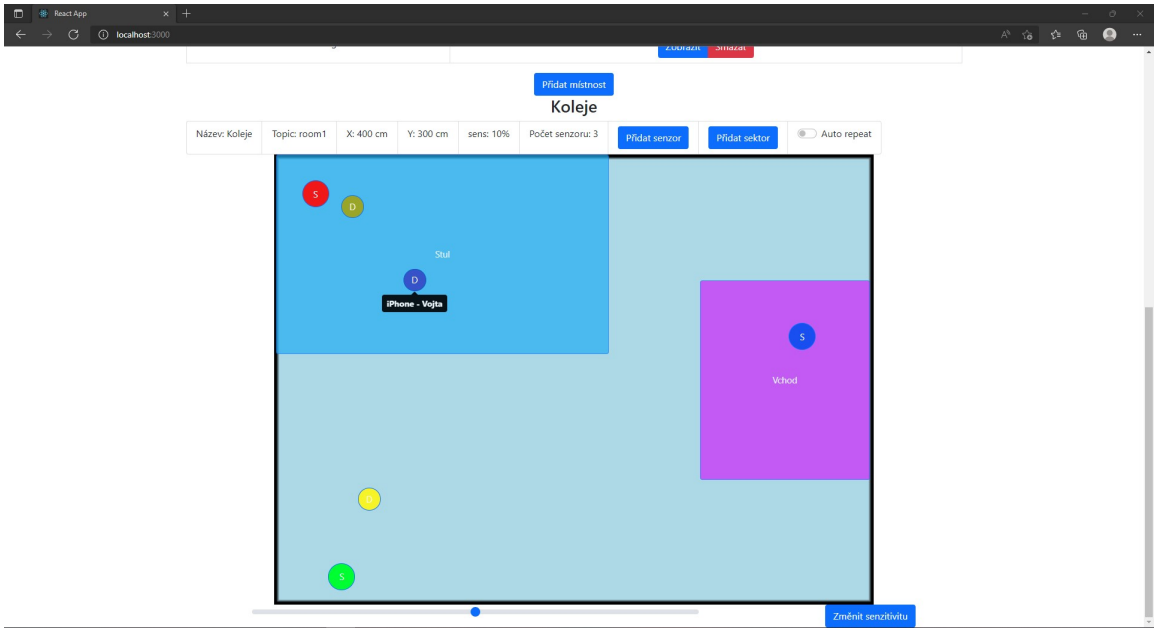
(b) Ukázka signálů na zařízení iPhone - Vojta

Obrázek 5.3: Ukázka signálů

Dále na obrázku 5.2 vidíme tabulku pokojů, kde můžeme pokoj vytvořit, zobrazit nebo smazat. Po kliknutí na tlačítko 'Zobrazit' se nám zaktualizuje poslední část pohledu, jak je vidět na obrázku 5.4.

Druhá část administrátorského pohledu nám zobrazuje mapu pokoje a umožnuje její intuitivní editaci včetně parametrů pokoje v grafickém uživatelském rozhraní. Tato mapa je vykreslena na canvas o rozměrech pokoje. Na canvas jsou namapované pozice senzorů,

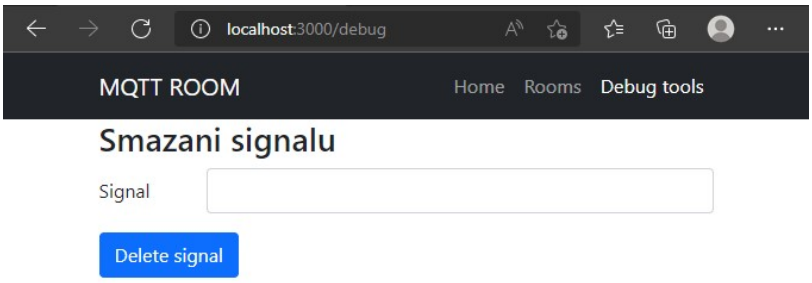
sektorů a zařízení. Na sektory a senzory se dá kliknout jako na tlačítko a následně upravit jejich barvu, pozici, název, topic a u sektoru ještě navíc velikost sektoru, tedy jeho rozměry.



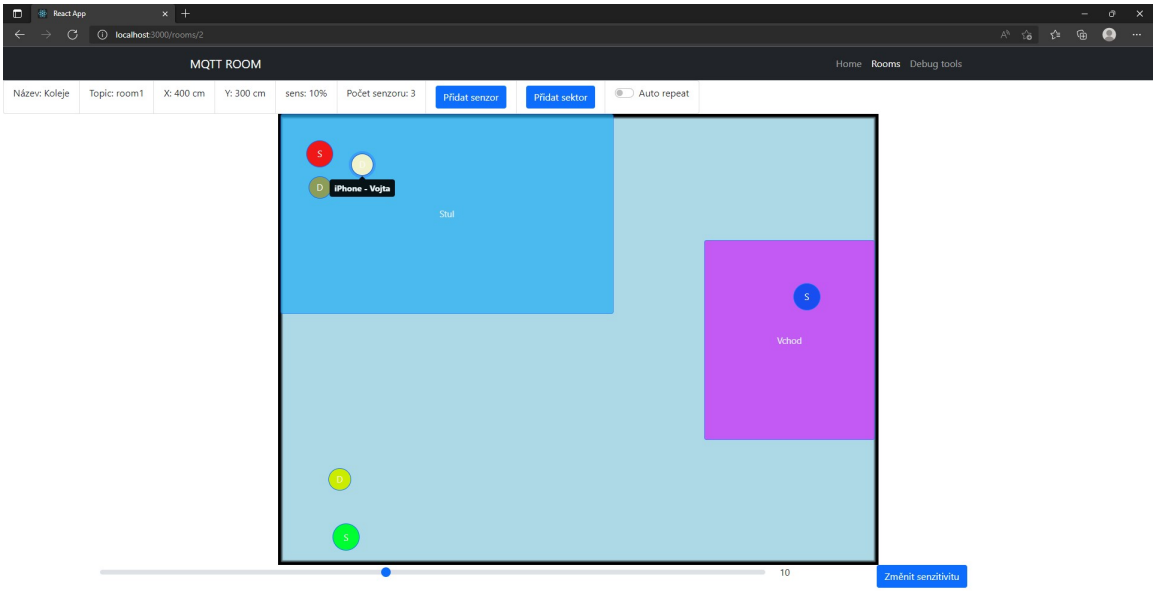
Obrázek 5.4: Druhá část administrátorského pohledu

Druhý hlavní pohled představuje zobrazení pouze mapy pokoje, což je velmi užitečné pro uživatele. Navíc je vše vytvořeno tak, že se dá tato mapa pokoje jednoduše přidat jako iframe panel do Home Assistantu. Na pohled se dostaneme pomocí URL `/rooms/:id`, kde id je pozice v tabulce indexování od 0. Na obrázku 5.6 vidíme například pokoj s názvem 'Koleje', který má id s hodnotou 2.

Poslední pohled je určen pouze pro ladění. Já ho využíval, když jsem potřeboval odstranit signál, který byl dlouho neaktivní viz obrázek 5.5.



Obrázek 5.5: Pohled pro ladění programu



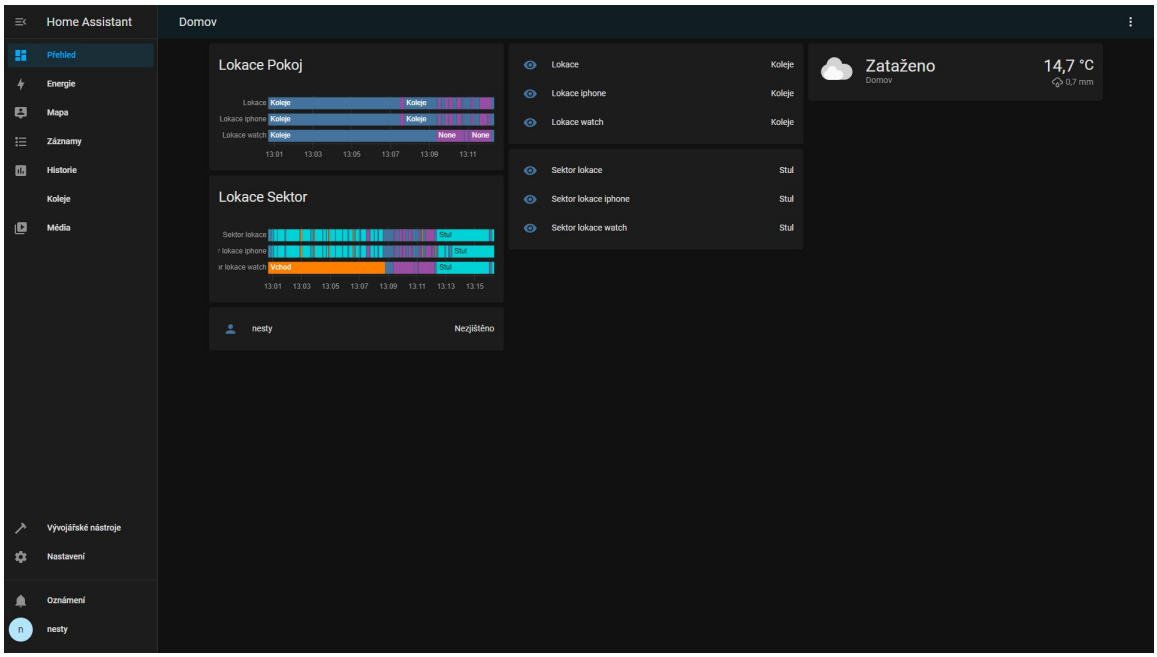
Obrázek 5.6: Uživatelský pohled

5.3 Home Assistant

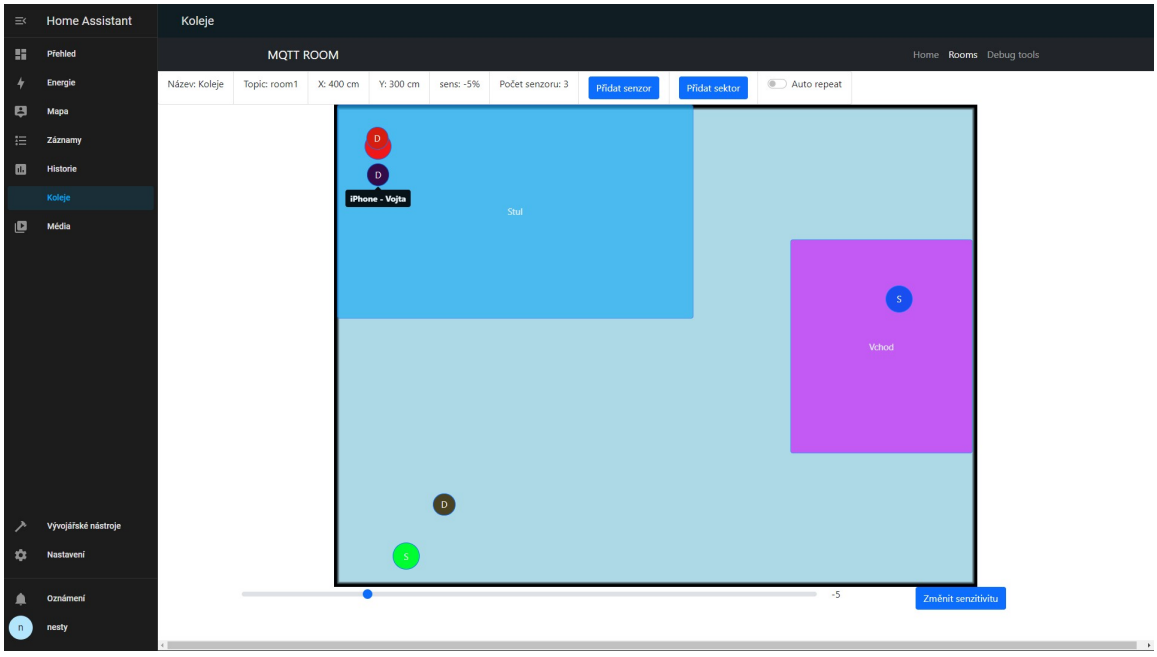
K propojení vytvořené aplikace pro detekci zařízení s chytrou domácností využívám webovou aplikaci Home Assistant. Použil jsem Home Assistant container image, což je samostatná kontejnerová instalace Home Assistant Core, kterou jsem nainstaloval do docker containeru. K zahájení používání aplikace po její instalaci stačilo pouze vytvořit uživatele. Do Home Assistantu jsem přidal integraci MQTT, kterou má Home Assistant standardně bez jakýchkoliv potíží přístupnou.

V konfiguraci se musí nastavit senzory na MQTT topicy, které chceme v aplikaci sledovat jako na obrázku 5.7. Zde vidíte senzory a jejich historii pro iphone, watch a uživatele. Konfigurace je vytvořena přes YAML soubor configuration.yaml v config složce Home Assistantu.

Dále jsem nastavil jako ukázku iframe panel pro místnost 'Koleje', vše je zobrazeno na obrázku 5.8. U Home Assistantu a iframe panelu je důležité, aby bylo vše nastaveno na stejný HTTP protokol - buď je oboje HTTP nebo HTTPS, jinak panel nebude fungovat.



Obrázek 5.7: Home Assistant MQTT senzory



Obrázek 5.8: Home Assistant iframe panel

Kapitola 6

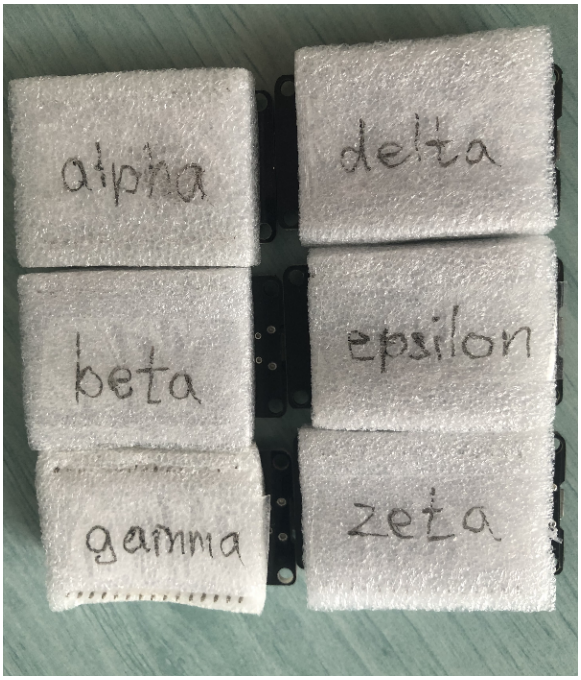
Ověření funkčnosti v reálném provozu

V této kapitole se zabývám testováním a ověřením funkčnosti řešení, které jsem navrhnul a implementoval. Dále otestuju již existující řešení, která jsem blíže popsal v kapitole 3. Výsledky všech testů mezi sebou řádně porovnám a vyhodnotím, které řešení je nejvhodnějším řešením pro dané účely.

6.1 Ověření funkčnosti

K ověření funkčnosti potřebuji mít nainstalovaný MQTT broker Mosquitto[1] od společnosti Eclipse, který si hostuji na lokální síti. Pro ověření funkčnosti a provozu přes MQTT používám program MQTT Explorer[2].

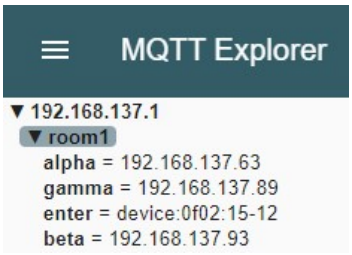
Ověřování funkčnosti jsem začal popisným označením senzorů ESP32 viz obrázek 6.1. Senzory jsem pojmenoval písmeny řecké abecedy. Alfa je Gate senzor v první místnosti a Delta je Gate senzor v druhé místnosti. Beta a Gama jsou Room senzory patřící k Alfě. Epsilon a Zéta jsou Room senzory patřící k Deltě. Na Gate senzory jsem nahrál firmware pro Gate senzor se správným názvem zařízení a názvem místnosti. Na Room senzory jsem nahrál Room senzor firmware se správným názvem řídícího Gate senzoru, názvem zařízení a názvem místnosti. Samozřejmě jsem do každého firmware musel napsat správné informace o WiFi a MQTT brokeru viz 5.1.2. Tyto informace se napříč všemi firmwary nemění.



Obrázek 6.1: Senzory ESP32, které jsem použil při ověření funkčnosti.

[1] https://mosquitto.org/
[2] http://mqtt-explorer.com/

Jestli jsem senzory nastavil správně ověřím tak, že v MQTT Exploreru uvidím topic s názvem pokoje a subtopicy s názvy zařízení se zprávou, která obsahuje IP adresu zařízení viz obrázek 6.2.



Obrázek 6.2:
Ukázka MQTT Exploreru.

Tímto způsobem zjistím, že zařízení jsou připojené k MQTT, k WiFi a používají správný topic pro komunikaci.

Dále nastavím konfiguraci backendu webové aplikace ve složce backend a v souboru `app.env`. V konfiguraci nastavím zdroje pro připojení k databázi, IP adresu a port MQTT brokeru a názvy Gate senzorů oddělené čárkou. Použiji příkaz `go build`, který zkompiluje balíčky spolu s jejich závislostmi. Po vytvoření spustitelného souboru mqtt_room_web mohu soubor spustit.

Když mám spuštěný backend, následně ve složce frontend spustím i frontend aplikaci příkazem `npm start`, který nám spustí webovou aplikaci na portu 3000. Webová aplikace se hned připojí k backendu.

Poslední část postupného spuštění celého systému se týká Home Assistantu. Pro komunikaci s chytrou domácností je potřeba nastavit konfiguraci v Home Assistantu. V konfiguraci se musí nastavit MQTT senzory pro každý topic, který chceme sledovat. Topic se skládá následovně.

`home-assistant/<NAME>/<DATA_OPTION>`

NAME je jméno uživatele nebo zařízení, které chceme sledovat. Parametr DATA_OPTION nám umožňuje definovat, zda-li budeme zařízení sledovat podle sektorů nebo podle místností.

Výsledek

Detekce zařízení v místnosti funguje správně a rychle. Při odchodu zařízení z místnosti se aplikace nastaví tak, že zařízení není v místnosti, což je správné a požadované chování. Webová aplikace neukazuje přesnou pozici zařízení, ale pouze odhad. Proto se zařízení na mapě místnosti hýbe i když je reálně v klidu. Při nastavení dostatečně velkých sektorů okolo senzoru fungují detekce v sektorech správně a v Home Assistantu nebudete mít chybná data. Detektor se dá použít i pro detekci zařízení v místnosti a vytvářet na této akci automatizace jako například zapnutí a vypnutí světel na základě přítomnosti zařízení v místnosti.

6.2 Testování

Prostředí

Všechna tři řešení budu testovat ve dvou stejných místnostech, ale senzory budou umístěny tak, jak doporučují technické příručky nebo dokumenty příslušné danému řešení. Místnosti nejsou vedle sebe, ale jsou oddělené chodbou asi pět metrů dlouhou. Místnosti jsou přibližně stejně velké. U každého z řešení bude obrázek s pozicemi senzorů na plánu místností.

Průběh testu

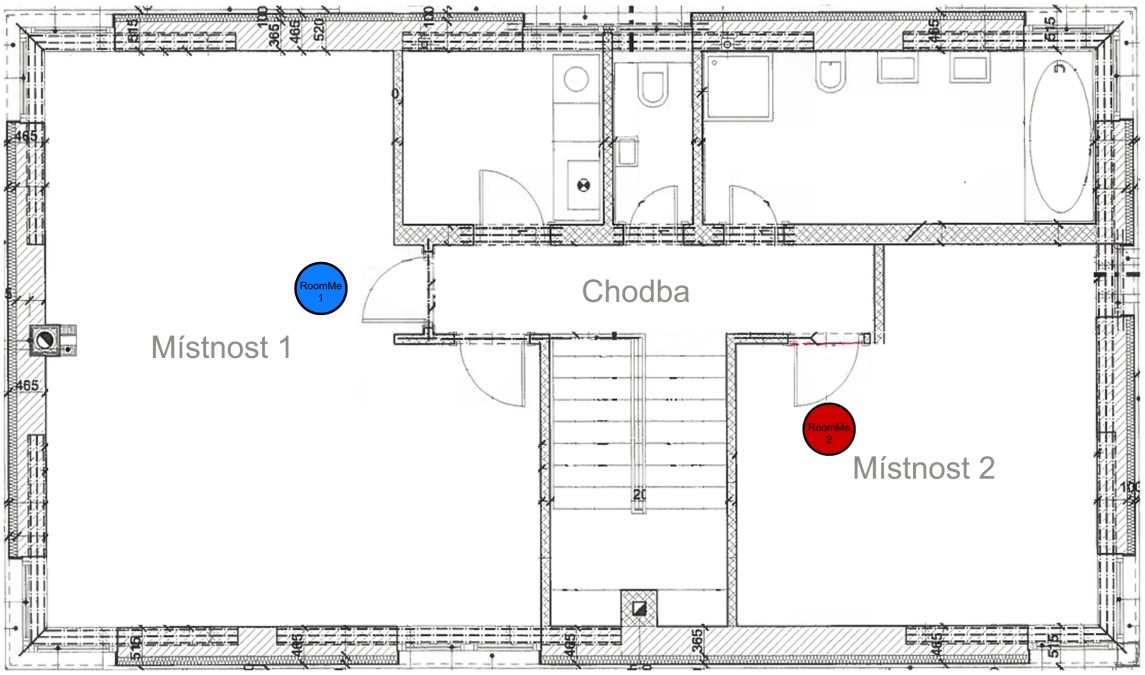
Nastavíme senzory v obou místnostech a půjdeme s Bluetooth zařízením z jedné místnosti do druhé. V druhé místnosti budeme třicet vteřin a vrátíme se zpět do první místnosti. Budeme pozorovat hodnotu senzoru v případě mého řešení nebo ESPresence. V případě RoomMe budeme sledovat, jestli se zapínají světla v místnosti. Tento test provedeme pětkrát.

RoomMe

V testování RoomMe od Intellithings jsem použil dva senzory RoomMe. Každý jsem nainstaloval do jedné místnosti viz. obrázek 6.3 a nastavil je tak, aby poznaly moje mobilní zařízení.

Při odchodu z místnosti se světla nezhasnou. Samotné zjištění, že zařízení se již nenachází v místnosti trvá 20 sekund až 1 minutu. Při příchodu do místnosti, tedy zastavení se pod senzorem, senzor rozsvítí světla během pár sekund. Zároveň je na senzoru led dioda, která zabliká vždy, když se pod ní spárované zařízení objeví. Chování senzoru se při opakovaném testování neměnilo. Nestandardního chování jsem si všiml pouze tehdy, když jsem se se zařízením vracel do místnosti číslo jedna - k zhasnutí světel nedošlo vždy.

Senzor zvládá velmi rychle detekovat zařízení při vchodu do místnosti. Senzoru trvá poměrně dlouho než zjistí, že už zařízení v místnosti není a to i přesto, že se zařízení nachází v jiné místnosti s dalším senzorem RoomMe. Tato logika by se dala doimplementovat s využitím Home Assistantu.



Obrázek 6.3: Plán rozmístění senzorů v případě testování RoomMe

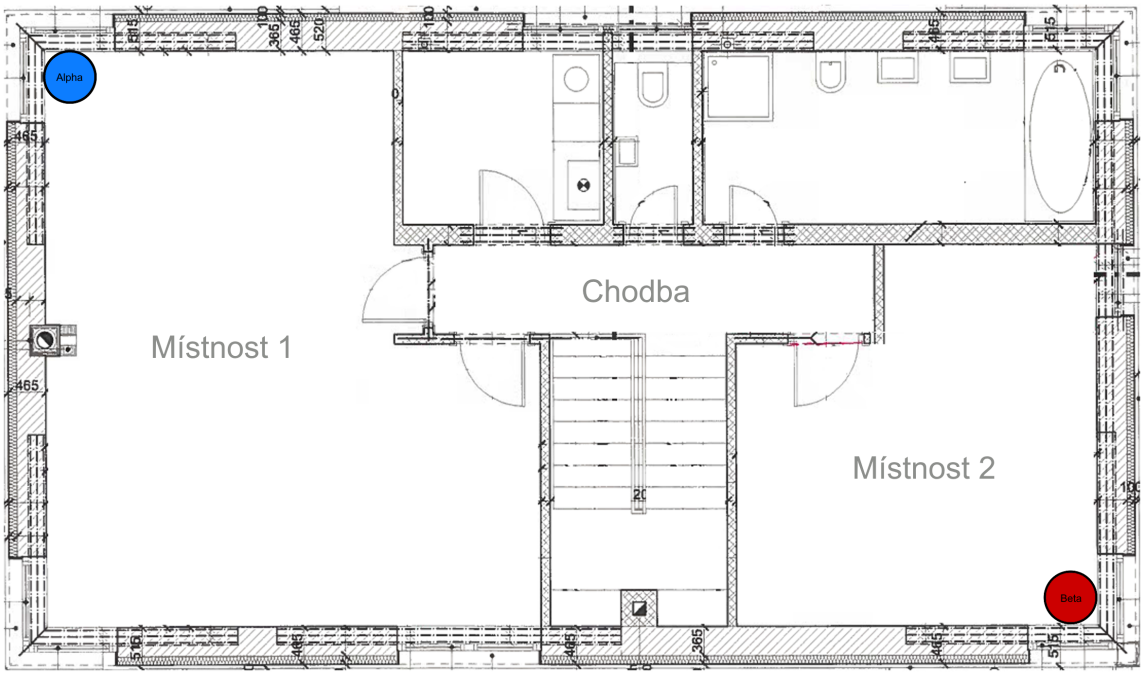
ESPresence

V testování ESPresence jsem použil dva mikročipy ESP32. Na každý jsem nahrál ESPresence firmware. K ESP jsem se připojil jako na hotspot a nastavil název, název místnosti, WiFi, MQTT broker a velikost místnosti. Velikost místnosti se musí nastavit, aby senzor nedělal neplatné vyhodnocení vchodu do místnosti. Senzory jsem nainstaloval do místností viz. obrázek 6.4. Do Home Assistantu jsem musel přidat zařízení ručně. Přidal jsem senzor na MQTT platformu s topicem zařízení.

Při příchodu do místnosti senzor detekuje zařízení poměrně rychle. Čím blíže k senzoru jdete, tím rychleji zareaguje. Při příchodu k druhému senzoru se hodnota, v které místnosti se zařízení nachází, automaticky změní na nejnovější detekci. To znamená, že při vstupu

do nové místnosti by se světla v předchozí místnosti automaticky zhasla. Na testování jsem nemusel mít spuštěnou aplikaci a mohl jsem využít i jíné zařízení než mobil například hodinky s Bluetooth signálem.

Senzor zvládá rychle detekovat zařízení při vstupu do místnosti. Odchod z místnosti detekuje způsobem, že se hodnota pozice přepíše detekcí jiným senzorem umístěným v jiné místnosti. Senzor nepotřebuje mobilní aplikaci a tím šetří baterii snímaného zařízení.



Obrázek 6.4: Plán rozmístění senzorů v případě testování ESPresence

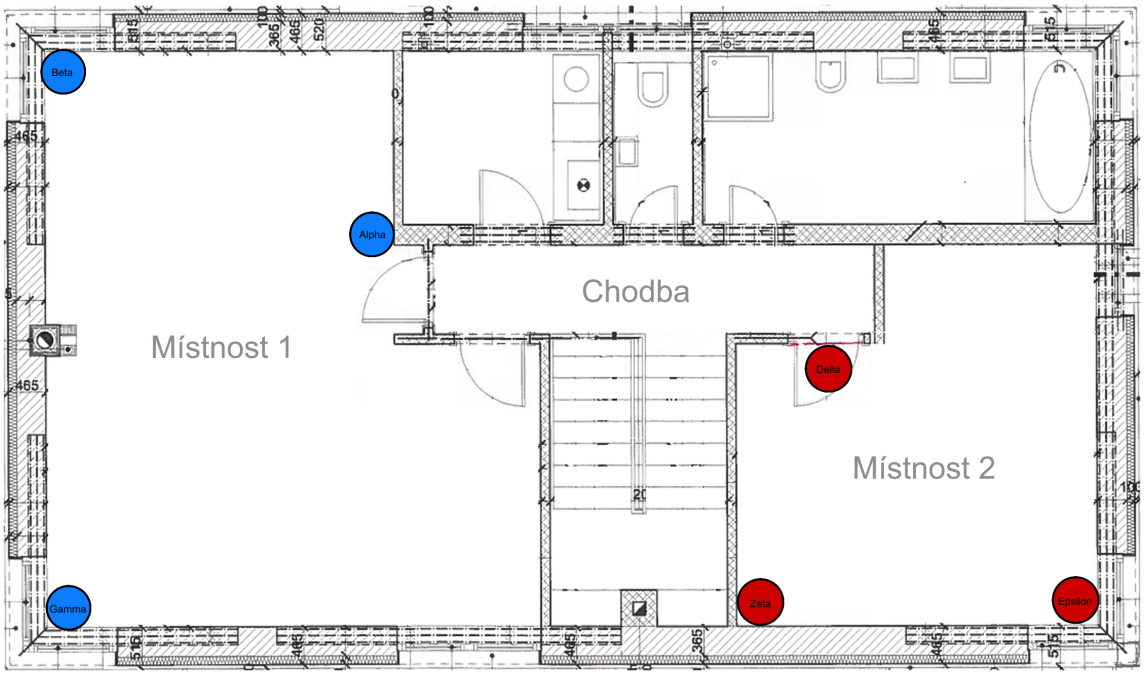
Mé řešení

V testování mého řešení jsem použil šest mikročipů ESP32. Na dva jsem nahrál firmware Gate senzor pro každou z místností, jak je zobrazeno na obrázku 6.5. Jedná se o senzory Alfa a Delta. Na zbylé čtyři mikročipy jsem nahrál firmware Room senzor. Senzory jsem musel nastavit ručně v kódu firmwaru. Nastavil jsem název zařízení, název místnosti, WiFi, MQTT broker a název Gate senzoru, z kterého si příslušné Room senzory budou načítat tabulku zařízení, o kterých si mají ukládat data. Senzory jsem nainstaloval do místností tak, jak je vyobrazeno na obrázku 6.5. Do Home Assistantu jsem musel přidat zařízení ručně. Přidal jsem senzor na MQTT platformu s topicem zařízení a subtopicem sektor nebo pokoj podle toho, kterou informaci jsem chtěl získavat.

Při příchodu do místnosti senzor detekuje zařízení rychle díky poloze Gate senzoru u dveří. Při odchodu z místnosti senzory detekují odchod rychleji než v případě RoomMe řešení. Průměrně pět až deset sekund. Při vstupu do další místnosti se zařízení opět detekuje rychle. Chybná detekce se dá regulovat ve webové aplikaci pomocí posuvníku citlivosti. Senzory dokážou spolehlivě detekovat sektor, pokud se sektor nachází v blízkosti nebo nejlépe okolo jednoho z pokojových senzorů.

Mé řešení plní korektně veškeré očekávané úkoly. Dokáže jednoznačně identifikovat vstup do místnosti, dokáže odhadnout polohu v rámci místnosti pomocí sektorů a dokáže i jed-

noznačné identifikovat odchod z místnosti. Moje řešení má na rozdíl od předchozích dvou testovaných řešení uživatelsky přívětivou aplikaci, která umožňuje jednoduchým způsobem nakonfigurovat všechny parametry zadané úlohy.



Obrázek 6.5: Plán rozmístění senzorů v případě testování mého řešení

6.3 Shrnutí výsledku provedených testů

Významné výsledky testů, které se dají snadno porovnat jsem uvedl do tabulky 6.1. Z tabulky vyplývá, že pro spolehlivou detekci odchodu z místnosti se dá použít pouze mé řešení. Protože se jedná o prototypové řešení, nastavení senzorů je u mého řešení složitější a musí se dělat přímo v programu. Pro běžného uživatele to samozřejmě není ideální přístup.

Kritéria	Mé řešení	RoomMe	ESPresence
Rychlost detekce vstupu	rychlé	nejrychlejší	rychlé
Rychlost detekce výstupu	rychlé	pomalé	pouze při vstupu do jíné místnosti
Spolehlivost detekce	vyhovující	vyhovující	vyhovující
Přesnost detekce	místnost a sektor	pouze místnost	pouze místnost
Složitost instalace pro uživatele	nejsložitější	jednoduchá	složitá
Složitost nastavení pro uživatele	nejsložitější	jednoduché	jednoduché
Pořizovací náklady (Kč / místnost)	600 Kč	2250 Kč	200 Kč

Tabulka 6.1: Porovnávající tabulka výsledků provedených testů

Kapitola 7

Závěr

Cílem práce bylo prostudovat problematiku IoT se zaměřením na detekci prezence. Nastudoval jsem si lokalizační metody, které jsem zpracoval v 2. kapitole. Díky seznámení s těmito metodami jsem získal základní přehled o používaných metodách a jejich vhodnosti pro aplikaci v mém řešení.

Seznámil jsem se s existujícími komerčním řešeními, které jsem nakonfiguroval a otestoval. Výsledky testování si můžete přečíst v 6. kapitole.

Existujícími řešeními jsem se inspiroval při návrhu vlastního řešení. Přijal jsem základní principy, které jsem rozšířil a aplikoval s využitím open source řešení pro dosažení požadovaného výsledku. Můj návrh řešení jsem rozšířil o možnost detailnější detekce přítomnosti zařízení v rámci jedné místnosti. K tomuto účelu jsem vytvořil strukturu sektor, která představuje specifickou část místnosti. Místnost si tak může uživatel ve webové aplikaci rozdělit přímo v uživatelském grafickém rozhraní na části (sektory).

Systém je navržen k používaní komunikace mezi webovou aplikací a senzory v místnosti pomocí služby MQTT. Výsledky detekce jsou posílané přes MQTT do Home Assistantu, kde si uživatel může nastavit automatizace umožňující provádět operace na základě detekce přítomnosti zařízení.

Detekovaná zařízení je nutné jednoznačně identifikovat. Původní záměr předpokládal využití Bluetooth MAC adresy daného zařízení. U řady moderních zařízení jsou však tyto adresy dynamické a v čase se mění. Toto komplikuje jednoznačnou identifikaci detekovaných zařízení. Jako vhodné řešení jsem navrhl využít technologie fingerprintů, které dokážou z dynamické Bluetooth MAC adresy udělat statický identifikátor, který se nemění. Zjistil jsem, že i tato metoda má nedostatky a při větším počtu zařízení se může stát, že více zařízení mají stejný identifikátor. To může zapříčinit chybné výsledky. Pro účely řešení v rámci chytré domácnosti pracující s malým počtem zařízení je metoda fingerprintů dostatečně učinná.

Zařízení jsou detekována pomocí senzorů umístěných v místnosti. Tyto senzory si ukládají informace o síle signálu a časové razítko, které napomáhá k synchronizaci odebíraných signálů od detekovaných zařízení. Detekce pozice zařízení je určována pomocí porovnání síly signálu od zařízení a od senzorů. Ve své práci předpokládám, že nominální výkon vysílaného signálu od Gate senzoru je přibližně stejné úrovně jako výkon signálů vysílaného Bluetooth vysílačem detekovaného zařízení. Tento předpoklad nemusí být vždy pravdivý, ale pro účely této práce předpokládám, že všechna zařízení pracují s přibližně stejně výkonnými Bluetooth vysílači.

Naměřená data senzory odesílají přes MQTT na webovou aplikaci, která data zpracovává a vyhodnocuje lokalizaci. Průmyslová řešení využívají triangulační metody pro výpočet

pozice zařízení. Při použití triangulace v mém řešení postaveném na nízkonákladovém hardwaru na bázi čipu ESP32 jsem zjistil, že i filtrovaná data pomocí Kalmanova filtru mají tak vysoký rozptyl, že se pozice nedá dostatečně přesně určit. Proto jsem ve finálním realizovaném řešení místo výpočtu pozice použil metodu odhadu, kde by zařízení se mohlo nacházet. Tento odhad systém automaticky stanovuje na základě předchozí pozice zařízení a pozice dvou nejsilnějších senzorů. Tato metoda mi umožnila získat spolehlivé informace o detekci zařízení v místnosti.

Uživatel může ve vytvořené webové aplikaci jednoduše spravovat detekované zařízení s použitím nově vytvořeného uživatelského grafického rozhraní. Toto rozhraní umožňuje také jednoduchým způsobem definovat kontrolovaný prostor, to znamená místnosti a její sektory a nadefinovat pozice umístění snímacích senzorů. Webovou aplikaci jsem navrhoval tak, aby byla jednoduchá také její integrace do systému pro správu chytré domácnosti jako je například Home Assistant. Ve své bakalářské práci jsem uvedl krátký návod, který popisuje způsob integrace mého řešení do systému Home Assistant s využitím MQTT brokeru.

Funkčnost v reálném provozu jsem ověřil a aplikaci budu dále vyvíjet a používat v mé chytré domácnosti.

Literatura

[1] Bulten, W., Rossum, A. C. V. a Haselager, W. F. G. Human SLAM, Indoor Localisation of Devices and Users. In: *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. April 2016, s. 211–222. DOI: 10.1109/IoTDI.2015.19.

[2] Close, C. *RoomMe Personal Location Sensor Review: Automation from above* [online]. [cit. 2022-10-01]. Dostupné z: https://www.imore.com/roomme-personal-location-sensor-review.

[3] Gorilla. *Gorilla WebSocket* [online]. [cit. 2022-20-04]. Dostupné z: https://github.com/gorilla/websocket.

[4] Intellithings®, L. *RoomMe Open API* [online]. [cit. 2022-10-01]. Dostupné z: https://www.intellithings.net/roommeapi.

[5] Park, H., Noh, J. a Cho, S. Three-dimensional positioning system using Bluetooth low-energy beacons. *International Journal of Distributed Sensor Networks*. 2016, sv. 12, č. 10, s. 1550147716671720. DOI: 10.1177/1550147716671720. Dostupné z: https://doi.org/10.1177/1550147716671720.

[6] RPishop.cz. *ESP32 Development Board* [online]. [cit. 2022-10-01]. Dostupné z: https://rpishop.cz/prototypovaci-desky/1500-esp32-vyvojova-deska.html.

[7] Wikipedia. *Go (programming language)* [online]. [cit. 2022-20-04]. Dostupné z: https://en.wikipedia.org/wiki/Go_(programming_language).

[8] Zahid, F., Rosdiadee, N. a Mahamod, I. Recent Advances in Wireless Indoor Localization Techniques and System. *Journal of Computer Networks and Communications*. 2013. DOI: 10.1155/2013/185138. Dostupné z: https://doi.org/10.1155/2013/185138.

Příloha A

Obsah paměťového média

V této příloze je uveden seznam jednotlivých souborů uložených na paměťovém médiu.

README.md Soubor, ve kterém je uvedený autor a licence.

app Adresář se zdrojovými soubory aplikace.

gate-senzor Adresář se zdrojovými soubory Gate senzoru.

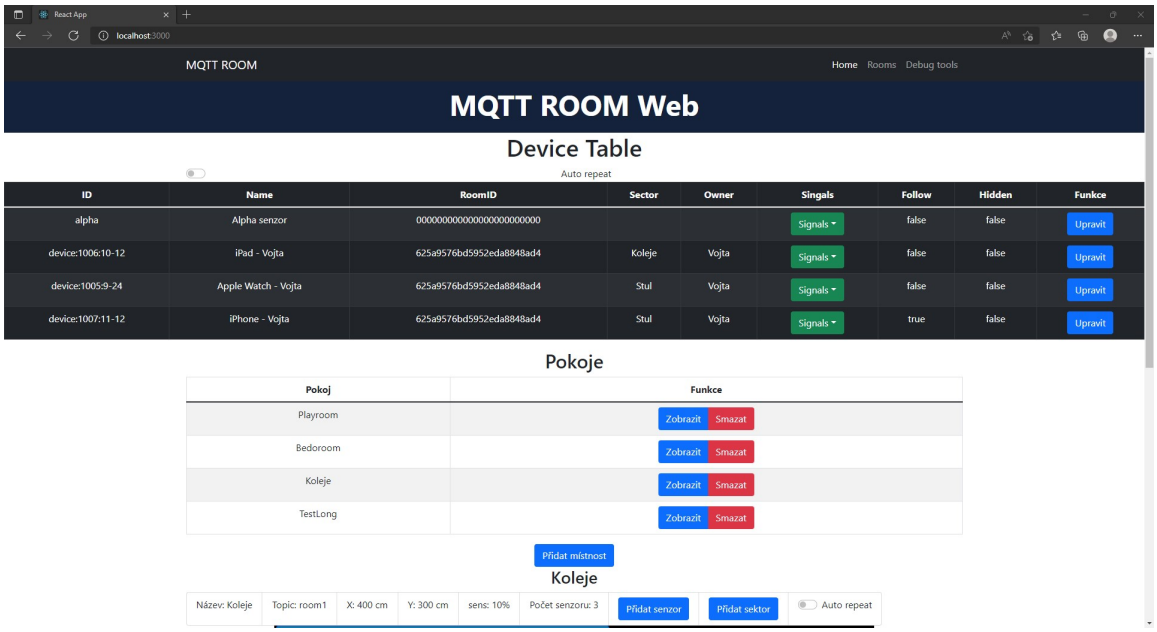
room-senzor Adresář se zdrojovými soubory Room senzoru.

latex Adresář s obsahem bakalářské práce.

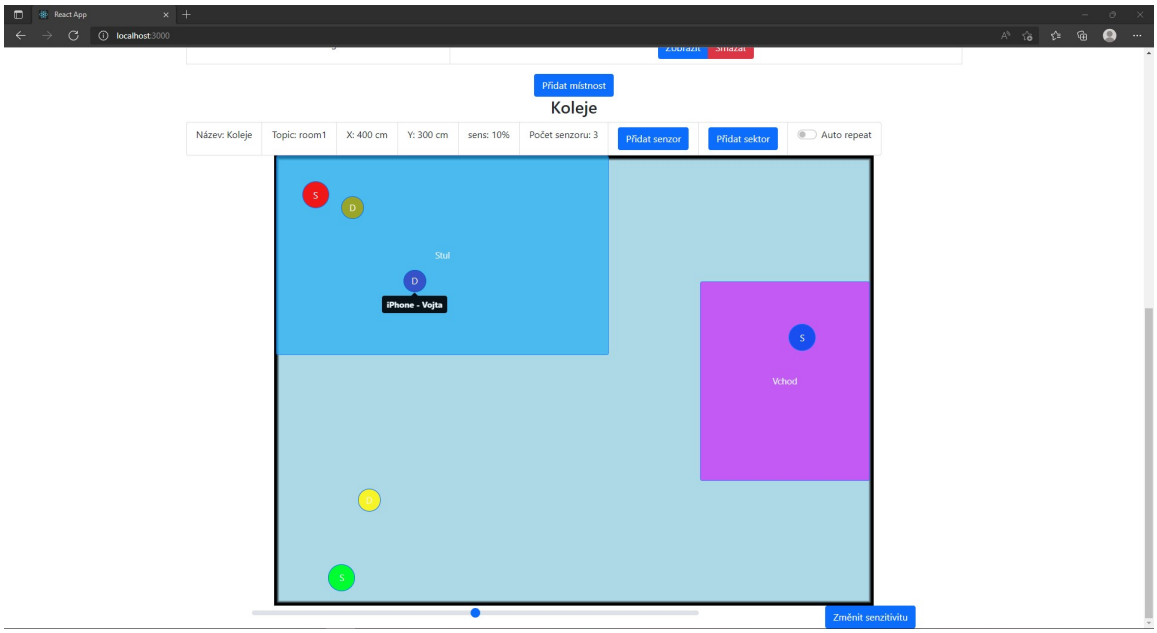
Příloha B

Grafické rozhraní webové aplikace

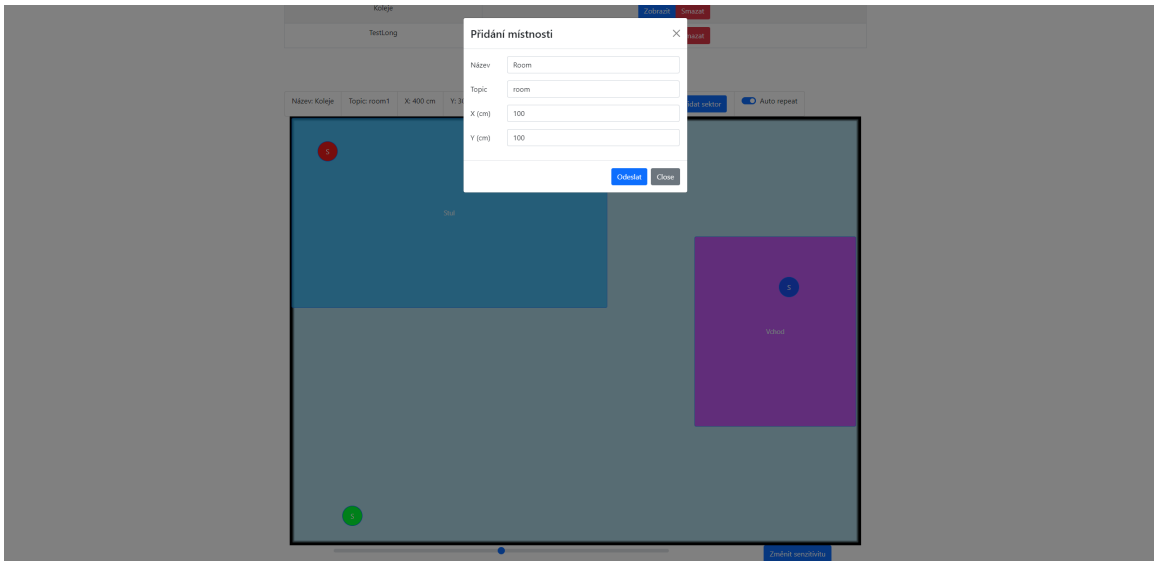
V této příloze jsou zobrazeny snímky obrazovky webové aplikace.



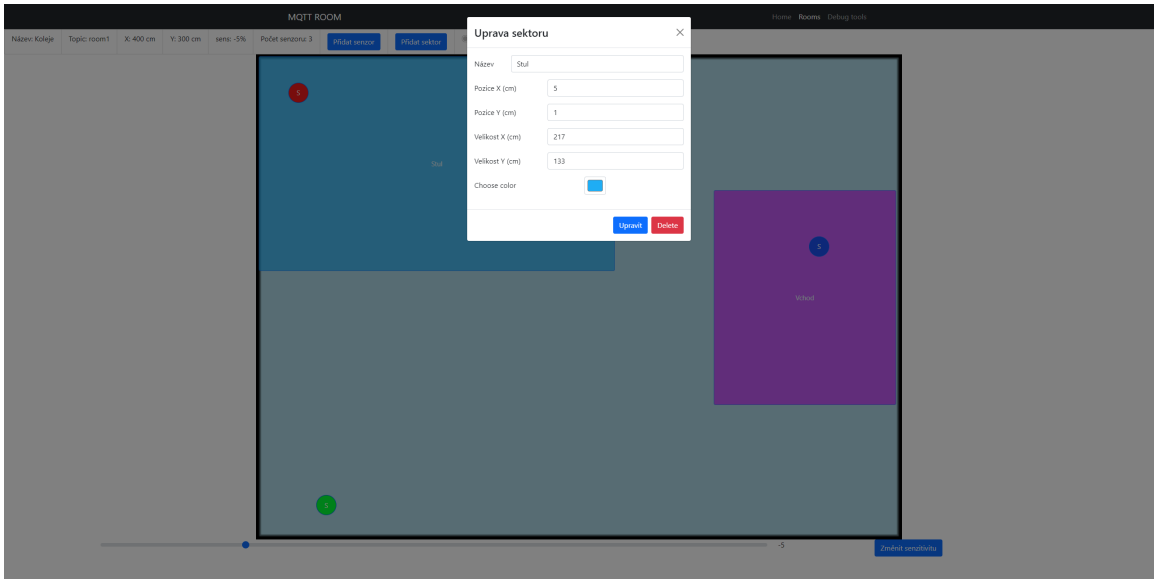
Obrázek B.1: První část administrátorského panelu



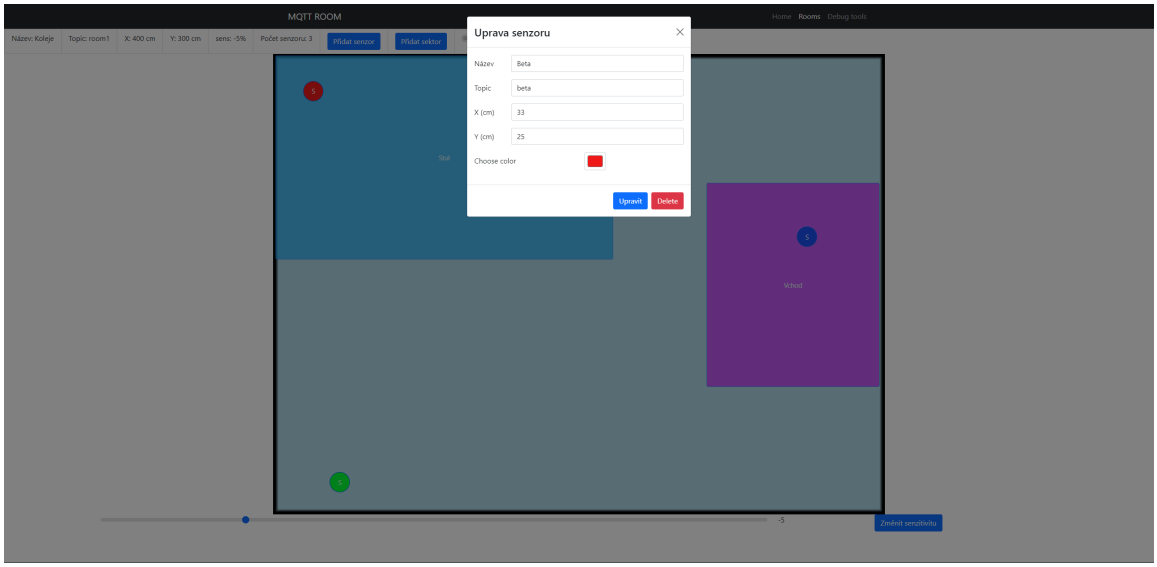
Obrázek B.2: Druhá část administrátorského panelu



Obrázek B.3: Vyskakovací okno pro vytvoření nové místnosti



Obrázek B.4: Vyskakovací okno pro zadání parametrů pro sektor stul.



Obrázek B.5: Vyskakovací okno pro zadání parametrů pro senzor Beta.