VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE PŘÍTOMNOSTI OSOB V MÍSTNOSTI

ROOM PRESENCE DETECTION IN THE SMART HOME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DAVID PRŮDEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN PLUSKAL**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Průdek David**
Program: Informační technologie
Název: **Detekce přítomnosti osob v místnosti**
Room Presence Detection in the Smart Home
Kategorie: Počítačové sítě

Zadání:

1. Proveďte průzkum technik pro zjištění přítomnosti osob v místnosti používaných v systémech pro domácí automatizaci. Zaměřte se ne jenom na detekci pohybu pomocí PIR, ale i určení přítomnosti například pomocí Bluetooth.
2. Navrhněte a implementujte patřičné senzory na vhodně zvolené platformě jednodeskového mikrokontroléru.
3. Navrhněte modul do domácího automatizačního nástroje Home-Assistant, který určí, zda-li se v dané místnosti někdo nachází.
4. Implementujte navržené řešení s využitím technologií dle pokynů vedoucího.
5. Proveďte měření ve vhodně zvoleném prostoru schváleném vedoucím a zhodnoťte přesnost a použitelnost svého řešení.

Literatura:

1. Blum, J. (2013). *Exploring Arduino: tools and techniques for engineering wizardry*. John Wiley & Sons.
2. Gill, K., Yang, S. H., Yao, F., & Lu, X. (2009). A zigbee-based home automation system. IEEE Transactions on consumer Electronics, 55(2).
3. Gomez, C., & Paradells, J. (2010). Wireless home automation networks: A survey of architectures and technologies. IEEE Communications Magazine, 48(6).
4. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Pluskal Jan, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 28. října 2018

Abstrakt

Cílem této práce je navrhnout a implementovat senzor detekce přítomnosti osob v místnosti vhodný pro použití v domácí automatizaci. Zaměřil jsem se na nalezení takového řešení, které pro svou činnost využívá běžnou nositelnou elektroniku. Senzor umístěný v místnosti detekuje tato nositelná zařízení a na základě síly signálu určí jeho pozici. Pro tento případ užití jsem použil technologii Bluetooth LE, která bývá součástí většiny nositelné elektroniky a v poslední době se často využívá k navigaci ve vnitřních prostorech. Použití tohoto senzoru pro automatizaci je zajištěno pomocí systému Home Assistant. Hlavním přínosem této práce je levně a jednoduše rozšířit možnosti běžné domácí automatizace detekci osob v jednotlivých místnostech, nikoli pouze v široké oblasti, kterou nabízejí lokace pomocí GPS nebo připojení k Wi-Fi přístupového bodu.

Abstract

The purpose of this thesis is to design and implement presence detection sensor which works with home automation. I focused on sensor implementation which needs for activity only available wearable electronics. Sensor placed in room can detect other devices and based on signal intensity it determines his location. For this case I chose Bletooth LE which is used to be implemented in the most of wearable electronics and recently new utilization of BLE was found for indoor navigation. Using this sensor for automation is ensured by system Home Assistant. Main benefit of this project is simply and cheap extend home automation capabilities for person detection in rooms, not only detection in large areas which offers GPS and Wi-Fi systems.

Klíčová slova

Detekce osob, Bluetooth, PIR, Vestavěný systém, ESP32, IoT, Platformy domácí automatizace, Home Assistant

Keywords

Person detection, Bluetooth, PIR, Embedded system, ESP32, IoT, Home automation platforms, Home Assistant

Citace

PRŮDEK, David. *Detekce přítomnosti osob v místnosti*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal

Detekce přítomnosti osob v místnosti

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Pluskala. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Průdek
15. května 2019

Poděkování

Děkuji Ing. Janu Pluskalovi za vedení mé bakalářské práce, odbornou pomoc a cenné rady při řešení této práce.

Obsah

Kapitola 1

Úvod

Technologie zaujímají v našich životech stále větší význam. Lidé se odjakživa snaží zlepšovat a hlavně zpříjemňovat svůj život. Stále se s sebou nosíme nejrůznější chytrá zařízené a tyto telefony, hodinky a náramky nám pomáhají s každodenními úkony. Také své domovy obohacujeme o tyto technologie, které již nejsou výsadou pouze movitých lidí, ale stávají se stále dostupnějšími pro širokou veřejnost. Již dnes máme možnost si obohatit domov nejrůznějšími chytrými bytovými doplňky a tradičními domácími spotřebiči, které jsou rozšířené o moderní funkcionalitu zařízení typu Internet of Things (IoT).

Označení IoT poprvé použil v roce 1999 *Kevin Ashton* v souvislosti s řízením dodavatelského řetězce [15]. Pod tímto pojmem si lze například představit pokročilé průmyslové stroje, které umožňují různé způsoby vzdáleného řízení, automatizace a plánování a jsou základem pro tzv. Průmysl 4.0 a chytré továrny. Oblastí, do kterých IoT promlouvá je mnoho a chytré domy jsou pouze jedny z nich. V této kategorii chceme docílit stavu, kdy jednotlivá zařízení mezi sebou komunikují a dokáží navzájem reagovat na vzniklé události. Událost může být například změna teploty, denní doby, světelných podmínek, ale také přítomnost člověka. Poslední zmíněná událost je hlavním předmětem této práce. Chceme, aby systém zaštiťující IoT v domácnosti dokázal na základě polohy člověka tato zařízení řídit. Cílem této práce je nalézt dostatečné levné a dostupné řešení problému zjištění přítomnosti osob v místnosti a tuto informaci použít k řízení IoT zařízení v chytré domácnosti. Systém musí být natolik univerzální, aby senzor bylo možné využít pro automatizaci celé řady IoT zařízení. Z toho důvodu je nutné, aby naše řešení bylo postavené pro známou open source platformu, která bude tato zařízení ovládat.

V práci se nejprve zaměříme na možné způsoby detekce přítomnosti člověka/zařízení. O tomto pojednává kapitola 2. V ní vybereme a rozebereme různé technologie, použitelné pro lokalizaci, zjistíme výhody a nevýhody a ohodnotíme jejich přínos v navrhovaném systému. Největší část zabírá rozbor standardu Bluetooth, který je očekávaným řešením problému detekce osob v místnosti.

Po teoretickém zpracování této problematiky se přesuneme na návrh senzorové části systému v kapitole 4. V této části si vymezíme veškeré požadavky, které musí senzor splňovat. S těmito požadavky dále vybereme vhodný hardware pro naše účely. Po podrobnějším popisu zvolených komponent nakonec zbývá definovat typ přenášených dat také přenosový protokol. Na základě vypracovaného návrhu poté následuje implementační část. Ta obsahuje potřebné informace pro implementování aplikace pro zvolenou platformu od použitého programovacího jazyka, přes použité knihovny, až po nastavení a použití specifických součástí dané platformy.

Kapitola 5 nastíní důvody použití a princip fungování vybrané platformy pro domácí automatizaci. V kapitole analyzujeme také další alternativní systémy a jejich odlišnosti od vybrané automatizační platformy. Poté se přesuneme k návrhu softwarové části na stanici platformy Home Assistant. V této části je popsán obecným způsobem vývoj nových modulů do tohoto systému pro domácí automatizaci. Po vysvětlení architektury automatizační platformy a návrhu rozšiřujících modulů následuje část zabývající se konkrétní implementací rozšíření pro tuto platformu a jeho způsob použití.

Poslední kapitola 6 pojednává o testování a zhodnocení vytvořeného řešení. V první části kapitoly je popsáno v jakém prostředí a s jakými vlastnostmi byl systém pro testovací provoz nasazen. Další část pojednává zejména o konkrétních vlastnostech a odlišnostech mezi nasazenými senzory v testovacím prostředí. Závěr samotné kapitoly je věnován celkovému zhodnocení systému pro detekci osob. Tato část obsahuje také konkrétní naměřené hodnoty, které vysvětlují chování systému. Jsou zde uvedeny a zdůvodněny veškeré nedostatky plynoucí z implementace projektu, metod detekce a technologií použitých k tomuto projektu. Kromě nedostatků jsou zde shrnuty také výhody implementovaného systému a jeho možné využití v reálném životě.

Kapitola 2

Techniky zjišťování přítomnosti osob

V této kapitole se podíváme na vybrané technologie a standardy, které je možné využít k detekci osob. Důležitým předpokladem pro použití vybraného standardu je výskyt použitelného řešení v běžných domácnostech a také nízká pořizovací cena potřebných hardwarových částí pro stavbu senzorů a čidel. Z důvodu nutnosti použít větší počet těchto čidel jsou předchozí kritéria velmi důležitá.

Dalším sledovaným faktorem v návrhu systému je nejvyšší možná přesnost lokalizování osoby (čidla). Jelikož je práce a řešení zaměřeno pro použití v domácnosti a využití pro automatizaci domu, například rozsvícení světel při vstupu do místnosti, je potřeba vybrat správnou metodu detekce, kterou je možné vymezit pouze na jednu místnost. Při použití rádiové techniky hledání zařízení, chceme aby použitá technologie vysílala signál náchylný na pevné překážky a po správném umístění senzoru bylo možné jednoznačně určit polohu zařízení.

2.1 Zařízení Bluetooth

Bluetooth je standard bezdrátové komunikace, vhodný pro přenášení dat mezi dvěma a více zařízeními malé vzdálenosti [4]. Od roku 1994 byl vyvíjen společností Ericsson a v současné době je spravován a zaštiťován skupinou The Bluetooth Special Interest Group (SIG) [3].

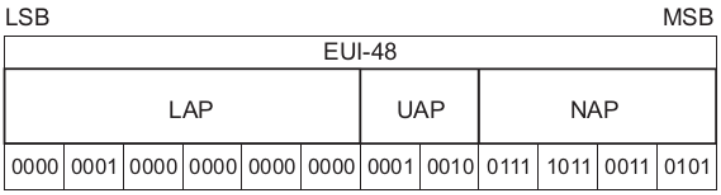
Zařízení Bluetooth komunikují v nelicencovaném 2.4 GHz pásmu Industrial, Scientific and Medical (ISM). Jelikož toto frekvenční pásmo je hojně využívané i jinými standardy, např. Wi-Fi, hrozí rušení signálu. Pro zmírnění následků tohoto rušení systém využívá metodu Frequency Hopping Spread Spectrum (FHSS), jejíž princip spočívá v časté změně kanálu signálu a používání celého frekvenčního pásma [3]. Maximální povolený výstupní výkon pro přenášení signálu je omezen a klasifikován do třech skupin a je stanoven na 100 mW (20 dB m). Verze 4.0 implementuje druhý mód Low Energy (LE), který je zaměřen na nízké energetické nároky, ale také nižší přenosové rychlosti a propustnost. Tato druhá forma nenahrazuje původní režim, nýbrž jej doplňuje a je možné oba režimy zároveň implementovat a provozovat [4].

Pro identifikaci Bluetooth zařízení se využívá čtyřiceti osmi bitová adresa *Bluetooth device address (BD_ADDR)*. Obrázek 2.1 popisuje bitovou strukturu BD_ADDR adresy. Na rozdíl od MAC adresy se liší bitovou strukturou a je rozdělena do 3 částí:

- Non-significant Address Part (NAP) - „Bezvýznamná část adresy“ - Tato část adresy velikosti 2 bajty je přidělovaná výrobcům organizaci IEEE [21].
- Upper Address Part (UAP) - „Vyšší část adresy“ - Podobně jako část NAP, tento bajt přidělují také IEEE, ale slouží jako most mezi NAP a LAP části a může být také odvozena z dat paketu [21].
- Lower Address Part (LAP) - „Nižší část adresy“ - Nejnižší 3 bajty adresy, která je obsažena v hlavičce kteréhokoli zaslaného paketu [21].

Architektura a „jádro“ základní verze Bluetooth (dále označován jako BR/EDR) jsou tvořeny čtyřmi nejnižšími vrstvami a protokoly, které v nich operují a všechna Bluetooth zařízení musí mít tyto základní vrstvy implementována. Nejnižší část je fyzická rádiová. Definuje signálové parametry např. frekvenci, výstupní výkon, FHSS a nebo typ modulace. Nad touto vrstvou leží další hardwarová část nazvaná Baseband layer. Hlavní funkcí Baseband vrstvy je udržování spojení s jiným zařízením a adresování. O navazování spojení a další operace spojené se správou spojení se stará protokol Link Manager protocol (LMP). Obsahuje také řešení autentizace a šifrování. Dalším důležitým protokolem v základní architektuře Bluetooth je protokol Logical link control and adaptation protocol (L2CAP). Přesměrovává data k jednotlivým vyšším aplikačním protokolům. Z nižších vrstev komunikuje buď přímo s LMP nebo prostřednictvím vrstvy Host Controller Intreface (HCI). V základním módu poskytuje nastavitelnou velikost paketů. Režimem Retransmission je rozšířena funkcionalita o zajištění spolehlivosti dat pomocí Cyclic redundancy check (CRC) a režim Flow Control je zaměřen na rychlé doručení dat bez možnosti kontroly, či opravy. Neméně důležitou službou v komunikaci pomocí standardu Bluetooth je protokol Service discovery protocol (SDP). Hledá v okolí jiná Bluetooth zařízení a poskytuje jejich parametry. Kromě těchto povinných protokolů je také důležitý volitelný protokol HCI, který tvoří vrstvu mezi řadičem Bluetooth a hostitelským zařízením [3].

Každá aplikace, která komunikuje s jiným, vyžaduje od komunikačního protokolu jiné vlastnosti. Těchto vlastností a provázanosti aplikace s Bluetooth rozhraním lze docílit pomocí profilů. Bluetooth profily definují požadavky na architekturu a její funkce a vlastnosti. Kromě základního jádra Bluetooth systému přidává další potřebné protokoly. Pokud obě komunikující strany splňují všechny požadavky profilu, aplikace může fungovat [3].

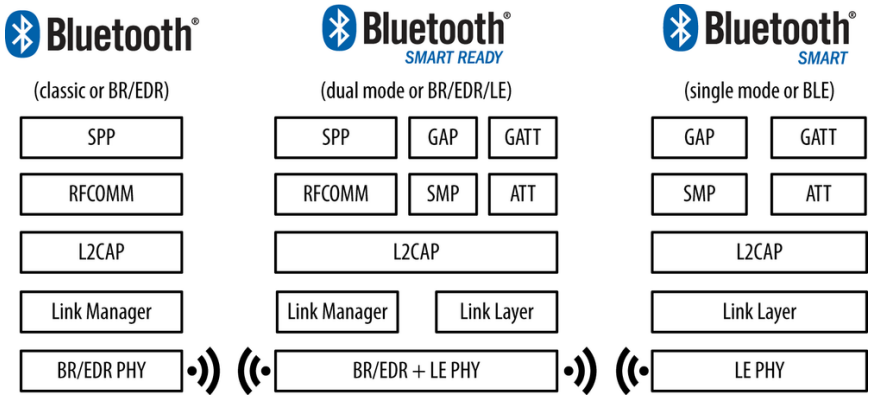


Obrázek 2.1: Bitová struktura BD_ADDR adresy a její rozdělení. Převzato z [4, s. 357].

2.1.1 Bluetooth Low Energy

Od verze 4.0 je součástí Bluetooth specifikace režim Bluetooth Low Energy (BLE), označován také jako Bluetooth Smart. Před převzetím vývoje skupinou SIG, bylo BLE vyvíjeno firmou Nokia pod názevem Wibree. Od základu byl tento standard navrhován s minimálními energetickými nároky. Definuje vlastní způsob vyhledávání zařízení, navazování spojení

a další mechanismy. BLE pracuje stejně jako základní verze v ISM nelicencovaném pásmu s metodou FHSS [23].



Obrázek 2.2: Konfigurace Bluetooth zařízení podle použitého režimu. Převzato z [23].

Podobně jako klasická verze BR/EDR vyžaduje BLE pro svou činnost nejnižší čtyři vrstvy, mezi které patří vrstva PHY layer, Link layer, L2CAP a servisní. Z poslední zmíněné servisní vrstvy jsou požadovány protokoly Security Manager (SM) a Attribute Protocol (ATT). První tři vrstvy BLE standardu mají podobnou funkci jako u základní BR/EDR verze, ale při použití obou režimů je nutné mít obě základní sady implementovány odděleně [23]. Možné kombinace obou standardů popisuje obrázek 2.2.

Typické použití BLE je na základě jeho vlastností vhodné pro komunikaci mezi zařízeními, která nevyžadují vysokou přenosovou rychlost a ani přenos velkých objemů dat, ale zato dovoluje koncovým zařízením udržovat stále bezdrátové spojení s nízkým elektrickým příkonem [23]. To je výhodné pro různá čidla, fitness náramky, hodinky apod.

2.1.2 Typické chování běžných BT zařízení

Díky zavedení technologie Bluetooth do mobilních telefonů se stal tento standard bezdrátové komunikace velmi dostupný. Velkou výhodou je tedy jeho rozšířenost napříč všemi různými druhy zařízení, nejen mobilními. Z toho důvodu je také k dispozici mnoho způsobů použití této technologie, ale ne každý se k účelu detekce přítomnosti osob hodí.

Nejprve je třeba si ujasnit, k čemu takové zařízení chceme použít. Myšlenkou celého projektu je navrhnout řešení, které je možné použít v chytrých domácnostech pro zjištění přítomnosti v místnosti. Jelikož detekovat konkrétní lidské tělo by bylo mnohem náročnější, chceme využít nějaké podpůrné zařízení, které už není tak obtížně hledat. Bluetooth, konkrétně BLE režim, je k tomuto účelu ideální, ale ne všechna zařízení je možná nastavit tak, aby byla stále zjistitelná.

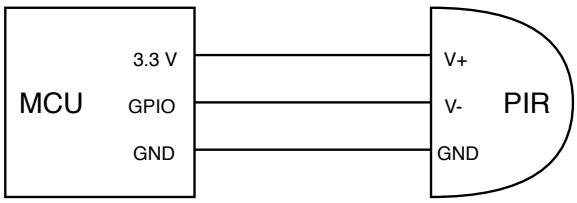
Například mobilní telefony mají možnost v základním nastavení Bluetooth adaptéru zapnout viditelnost, ale není možné mít adaptér z bezpečnostních důvodů dlouhodobě nastavený. Tento problém lze vyřešit softwarovou cestou u chytrých telefonů aplikací, která dokáže simulovat funkci Bluetooth beacon. Beacon je fyzické zařízení postavené na technologii BLE. Jeho funkcí je periodicky distribuovat malý obsah dat všem zařízením v jeho dosahu. Toto zařízení používá například projekt společnosti Google nazvaný Nearby. My můžeme takové zařízení simulovat i v mobilních telefonech s operačním systémem Android a iOS. Vhodnou alternativou k tomuto řešení je koupě hardwarového zařízení typu bea-

con. Toto řešení lze pořídit v čínských internetových obchodech od částky $10 a na běžnou knoflíkovou baterii dokáže fungovat i 1.5 roku.

Další možností je použití chytrého náramku, či hodinek. Jelikož je trh na nositelnou elektronikou velice rozmanitý a mnoho výrobců využívá svá softwarová řešení, nelze jednoznačně tvrdit, zdá je možné všechna zařízení této skupina nastavit podle našich potřeb. Na základě osobních zkušeností lze například chytré náramky řady MiBand od společnosti Xiaomi zapnout ve viditelném režimu, který zajistí, že zařízení je viditelné nepřetržitě. Rozhraní Bluetooth se využívá také pro připojení nejrůznějších počítačových periférií, jako klávesnice, myš a sluchátka, avšak tato zařízení spojuje společný princip párování, který spočívá v tom, že se aparát dočasně přepne do speciálního režimu a pouze v této krátké době je přístroj pro ostatní viditelný.

2.2 Detektor pohybu PIR

Passive Infrared sensor (PIR) je elektromagnetický senzor, který slouží pro měření infračerveného záření vyzařující se z objektů s teplotou nad absolutní nulou. Čidlo obsahuje pyroelektrický materiál, který je dále součástí integrovaného obvodu. Tato čidla jsou dále připojena k diferenciálnímu zesilovači, aby se vyfiltrovala teplotní složka statického okolí. PIR čidlo je tedy pasivní, protože sám o sobě nevyzařuje energii a nedokáže měřit teplotu, ale pouze detekovat lokální teplotní změnu. Při změně v celém zorném poli snímače (rozsvícení světel), se čidlo nespustí. Snímač je běžně osazen plastovým oknem, které propouští pouze infračervené světlo. Velikost zorného pole je nastavováno pomocí čoček [24]. Běžné PIR čidlo má tři vývody. První je napájecí pin, druhý slouží pro výstupní napětí, když je detekován pohyb a třetí je uzemnění, viz. obrázek 2.3. Existují i pokročilejší varianty senzoru, které mají navíc další piny pro ovládání citlivosti a doby aktivace.



Obrázek 2.3: Připojení obecného PIR senzoru k mikrokontroléru. Vstupní napětí na pinu V+ se může lišit podle vybraného modelu senzoru.

Například PIR senzor *AM312* je model se třemi vývody, tudíž má pouze pevně danou dobu aktivace stanovenou na dvě sekundy. Jeho provozní napětí je mezi 2.7 V a 12 V. Největší možná vzdálenost detekce je 3 metry v kuželovitém úhlu do 100°.

Samotné PIR čidlo není možné použít pro detekci osob v místnosti, jelikož nemůže rozpoznat nepochybující se osoby. Avšak přesná informace o zaznamenaném pohybu je důležitá a může být nápomocná ve spolupráci s jinou technologií pro identifikaci osob.

2.3 Síťová zařízení připojená pomocí Wi-Fi

Wi-Fi technologie popsaná rodinou standardů IEEE 802.11 je nejběžnější způsob bezdrátového připojení k LAN síti. Wi-Fi využívá stejně jako Bluetooth bezlicenční 2.4 GHz pásmo, dále pak méně rušené pásmo 5 GHz a existuje i varianta Wi-Fi pracující na frekvenci 60 GHz.

Možnost připojení svého mobilního telefonu nebo notebooku je pro nás samozřejmostí. Z podstaty věci signál Wi-Fi vysíláme s mnohém větším výstupním výkonem než Bluetooth a snažíme se mít naši bezdrátovou sítí pokryté veškeré bytové místnosti. V případě neúspěchu rozšíření sítě pomocí jednoho přístupového bodu připojíme další [1].

Z tohoto důvodu nemůžeme jednoznačně určit, kde se jaké zařízení nachází, i když můžeme zjistit, ke kterému Access Point (AP) je připojené. Avšak velkou výhodou tohoto řešení je existence mnoha nástrojů pro skenování zařízení připojených k lokální síti. Mezi nejznámější je konzolový program nmap. Příkazem `nmap -sP 192.168.1.0/24` se spustí hledání aktivních zařízení v síti `192.168.1.0` [7]. V automatizační platformě Home Asistant je podpora pro tuto aplikaci[1]. Po správné konfiguraci dané komponenty a výběru MAC adres hledaných zařízení, systém automaticky detekuje přítomnost těchto stanic v síti.

2.4 Globální navigační systém

Družicové systémy pro lokalizování aktuální polohy jsou s námi již řadu let. První systémy vznikly pro armádní účely v USA (GPS) a tehdejším SSSR (GLONASS)[2]. Tyto vojenské projekty se nakonec postupem času dostaly mezi běžné lidi a dnes je využíváme hlavně pro automobilovou navigaci na cestách [26].

Základním principem obecného Global Navigation Satellite System (GNSS) spočívá v získávání dat z co největšího počtu satelitních družic. Každá družice vysílá informace o svém označení, poloze a čase, ze kterých navigační zařízení zjistí svou relativní kulovou polohu. Z informací z dalších družic vznikají průsečíky těchto kružnic a tím se zpřesňuje informace o poloze. Obecná struktura globálních navigačních systému se skládá že tří částí: kosmická, řídící a uživatelská. Důležitá je přesnost posílaných časových značek. Z tohoto důvodu družice obsahují přesné atomové hodiny, které jsou několikrát do roka spravovány [26].

Stávající u nás nejpoužívanější navigační systém Global Positioning System (GPS) pro civilní volné účely neposkytuje dostačující absolutní přesnost. Ta se udává někde mezi 5 až 10 m [26]. Kromě přesnosti je také problém u GPS nefunkčnost v budovách, jelikož běžná zařízení nedokážou spolehlivě přijímat signály z družic. Z těchto důvodů globální navigační systémy nejsou vhodné pro určování pozice člověka v budově, ale mohou poskytnout informaci o tom, zda se daný člověk v budově nachází.

2.5 Ostatní zařízení

Další způsob rozpoznávání člověka je pomocí kamer. Ty se používají například v automobilech pro rozpoznávání chodců, cyklistů, či zvěře. Samotná kamera v tomto řešení nestačí a je nutný dostatečný výpočetní výkon a algoritmus, který v reálném čase dokáže rozpoznávat v obraze známé prvky a správně vyhodnocovat objekty. Obrazy z termokamer tomuto velice napomáhají. Jednotlivé snímky obsahují kromě obrazových dat také informace o teplotě jednotlivých objektů, díky vizuálnímu vyobrazení infračerveného záření. Další možností je využití ultrazvukového čidla. To na základě ultrazvukových vln dokáže určit vzdálenost od protějšího objektu a při vstupu do oblasti působení čidla se změní daná vzdálenost a čidlo se aktivuje. Nevýhodou tohoto řešení je nízký detekční rozsah a úzké zorné pole [19].

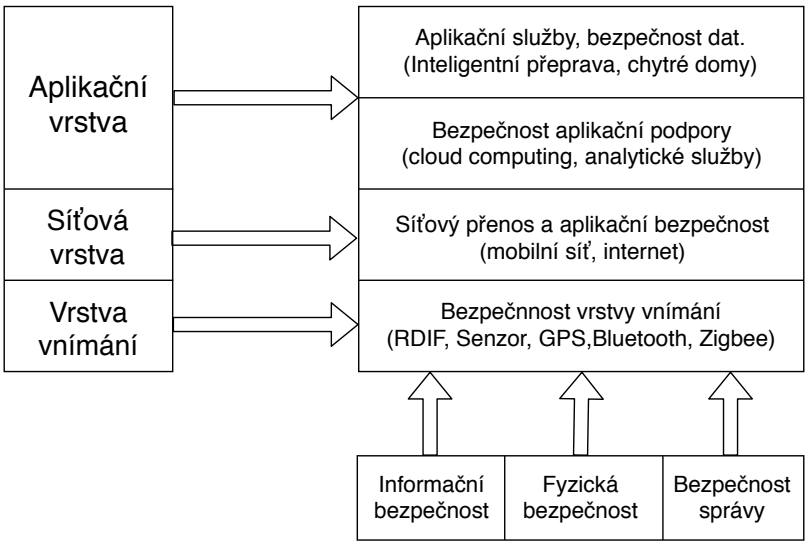
[1] Viz: https://www.home-assistant.io/components/nmap_tracker/

[2] GPS ani GLONASS nebyly první navržené systémy tohoto účelu, ale jsou aktuálně využívány.

Kapitola 3

Komunikační protokoly v IoT

Obecnou strukturu IoT systému lze rozdělit do tří vrstev: vrstva vnímání, síťová a aplikační. Ukázka struktury IoT systému je znázorněna na obrázku 3.1. Jednotlivé části mají odlišnou funkci a také zaujímají například odlišnou bezpečnostní politiku. Vrstva vnímání reprezentuje senzory, které by měly být schopné zajistit komplexnost vnímání tím, že dokážou nabídnout informace kdykoli a kdekoli. Zařízení spadající do této vrstvy jsou velmi různorodá, ale obecně jsou navržena jako velmi jednoduchá zařízení s málo výkonnými procesory, které nejsou schopny dostatečné komplexní ochrany. Síťová vrstva se stará o zajištění důvěrného, bezpečného a spolehlivého přenosu dat do datacenter. Aplikační vrstva poté data sbírá, zpracovává a analyzuje je. Tato část musí zajišťovat zejména autorizaci uživatelů, přístup k datům, ochranu a obnovu dat, a proto je nutné zajistit ochranu spojenou s těmito požadavky [25].



Obrázek 3.1: Architektura IoT systému. Prvky sítě jsou rozděleny do tří kategorií podle jejich účelu. Převzato z [25].

Bezdrátové vestavěné systémy nabízejí výrazný uživatelský komfort a efektivní spravování a řízení celé sítě. Rádiové standardy používané v IoT se dělí na základě velikosti rozlohy, kterou síť dokáže pokrýt. Mezi zástupce řešení pro pokrytí rozlehlých geografic-

kých oblasti jsou například celulární sítě. Středně rozlehlé sítě nabídnou standardy Wi-Fi, Zigbee a Z-Wave a mezi malé sítě patří například Bluetooth [8].

Pro účely této práce jsou vhodné protokoly pokrývající středně rozlehlé oblasti. Tyto protokoly dokážou spolehlivě zajistit potřebné rádiové pokrytí pro komunikaci mezi senzory v budovách.

3.1 Wi-Fi

Základní charakteristika tohoto standardu rodiny IEEE 802.11, zejména z pohledu využití této technologie pro detekci osob, je uvedena v rámci kapitoly 2. Tato technologie byla navržena zejména pro připojení plnohodnotných osobních zařízení a zajišťuje vysoké přenosové rychlosti. Není proto přímo stavěna pro Machine-to-Machine (M2M) komunikaci, avšak připojení k této síti je všude dostupné a nabízí použití specifických aplikačních protokolů vhodných pro IoT, například MQTT.

3.1.1 MQTT

Message Queue Telemetry Transport (MQTT) je M2M protokol, který byl původně navržený společností IBM a postupem času byl přesunut pod open source komunitu, se později stal velmi oblíbený hlavně v oblasti IoT. Jedná se o jednoduchý binární protokol, umožnující dvě základní operace: publikování a odběr. Protokol je vhodný zejména pro zařízení s omezenými zdroji a vestavěné systémy. Například oproti protokolu HTTP, který pracuje na principu žádost-odpověď, je mnohem jednodušší a minimalizuje přenášená data pomocí komprimovaných hlaviček. Všechna zařízení komunikují prostřednictvím prostředníka (Broker) a ten přijaté zprávy PUBLISH rozesílá všem odběratelům [20]. MQTT nabízí tři úrovně zajišťování a řízení datových toků Quality of Service (QoS). Metody jsou rozlišeny podle úrovně zajištění doručení dat číslicí od 0 po 2 [22].

- Typ 0 „At most once“ - Odesílatel i broker pošlou zprávu PUBLISH nejvýše jednou a poté ji zahodí. [22].
- Typ 1 „At least once“ - Odesílatel si zprávu PUBLISH po odeslání uchová a odběratel po přijetí zprávy zašle zpět brokeru potvrzení PUBACK a toto potvrzení pak broker předá původnímu odesílateli. Ten poté může zprávu zahodit [22].
- Typ 2 „Exactly once“ - Odesílatel odešle zprávu PUBLISH. Broker po přijetí zprávu rozešle odběratelům. Poté broker potvrdí odesílateli úspěšné přijetí zprávou PUBREC. Odesílatel potvrzení přijímá a vrací zprávu PUBREL. Broker uloženou zprávu zahazuje a nakonec vše potvrdí metodou PUBCOMP. Odesílatel zahodí i svou kopii zprávy [22].

Všechny publikované zprávy patří pod určitý topic a odběratel, který chce zprávu přijmout, musí tento topic znát a zažádat o jeho odběr u MQTT brokeru. MQTT využívá TCP transportní protokol a pro zabezpečení a šifrování je možné využít variantu s TLS/SSL. Hlavička protokolu má pro malé zprávy do velikosti 256 MB fixní dálku 2 B [20].

3.1.2 CoAP

Constrained Application Protocol (CoAP) je dalším zástupcem M2M protokolů a je vyvíjený skupinou IETF CoRE (Constrained RESTful Environments) Working Group. CoAP

podporuje dva způsoby komunikace: request/response (požadavek/odpověď) a resource/observe (podobný typu publish/subscribe v MQTT). CoAP je navržený tak, aby spolupracoval s HTTP protokolem [20].

Pro identifikaci zdrojů využívá CoAP Universal Resource Identifier (URI), na rozdíl od topicu používaném v MQTT protokolu. Data jsou publikována pro konkrétní URI a odběratel přijímá data od zdrojů určených jejich identifikátorem URI [20].

Hlavička protokolu má fixní délku o velikosti 4 B. Maximální velikost zpráv je definovaná pouze implementací serveru. Protokol je postavený na UDP transportním protokolu a zajištění integrity a důvěrnosti zpráv je řešeno pomocí DTLS protokolu, který je založený na TLS a určený pro datagramové protokoly [20].

Komunikace mezi klienty a servery není spojovaná z důvodu použitého transportního protokolu, avšak nabízí 2 režimy QoS: confirmable (potvrzovací) a non-confirmable (nepotvrzovací). Oproti protokolu MQTT nabízí více funkcí, například umožňuje vyjednávání obsahu s cílem dojednat preferovanou reprezentaci dat [20].

3.1.3 AMQP

Advanced Message Queuing Protocol (AMQP) je dalším příkladem „lehkého“ M2M protokolu. Byl vyvinut Johnem O'Hara ve společnosti JPMorgan Chase v roce 2003. Jedná se o protokol, který je navržen pro komerční účely s ohledem na bezpečnost, spolehlivost a interoperabilitu. Stejně jako CoAP využívá dva typy architektur: request/response a publish/subscribe. Nabízí široký rámec funkcí spojených se zasíláním zpráv, které zajišťují například transakce, flexibilní směrování a publikování/odebírání zpráv označených topicem [20].

Systém pro komunikaci protokolem AMQP vyžaduje, aby nejen vydavatel, ale i odběratel vytvořil „výměnu“ identifikovanou jménem, které se poté vyšle ostatním komunikačním uzlům. Vydavatelé a konzumenti pak využijí jméno výměny pro vzájemné seznámení. Konzument vytvoří frontu, kterou přiřadí k vytvořené výměně. Zprávy přijaté z výměny se přiřadí do fronty. Zasílání zpráv bude probíhat různými způsoby podle typu výměny. Mezi možné způsoby přenosu zpráv je: přímý přenos zprávy a odběr topicu [20].

AMQP je binární protokol s fixní délkou hlavičky 8 B a jeho maximální délka zpráv je určená danou implementací serveru nebo brokera. Výchozím transportním protokolem je TCP a s kombinací protokolu TLS/SSL a autentizační metody SASL je zajištěna bezpečnost přenosu zpráv. Nabízí také dvě úrovně QoS: spolehlivé a nespolehlivé doručování zpráv [20].

3.1.4 HTTP

Hyper Text Transport Protocol (HTTP) je textový protokol převážně využívaný pro web. Protokol vytvořil Tim Berners-Lee, později jeho vývoj převzaly organizace IETF a W3C. První specifikace standardu byla publikována v roce 1997. Pro identifikaci HTTP zdrojů se využívá URI, na jehož základě HTTP server zasílá patřičná data [20].

HTTP podporuje koncepci RESTful webové architektury s request/response způsobem komunikace. Hlavička protokolu i maximální délka dat nemají fixně definovanou délku. Výchozím transportním protokolem je TCP a zabezpečení přenosu dat je pomocí TLS/SSL. HTTP protokol nemá explicitně definované QoS [20].

3.1.5 Srovnání protokolů pro zasílání zpráv

Ačkoli všechny výše uvedené protokoly plní stejnou funkci, lze u nich nalézt některé odlišnosti. Prvním důležitým porovnávacím kritériem je velikost zpráv a množství režie potřebné k přenesení zprávy. Na tuto vlastnost má největší vliv velikost hlaviček paketů a typ použitého transportního protokolu. Na základě těchto vlastností lze určit, že nejméně přenášených dat a nejmenší režii poskytuje protokol CoAP, protože pro svou činnost používá UDP transportní protokol a nemusí vytvářet stabilní spojení mezi klienty. Zbylé protokoly jsou již postavené na TCP transportním protokolu, takže jejich rozdíl v kategorii velikosti zpráv určuje velikost jejich hlaviček. Z tohoto důvodu je druhým nejúspornějším protokolem MQTT, poté AMQP a nakonec protokol s neomezenou délkou hlaviček HTTP [20].

Výše uvedené aplikační protokoly nejsou svázány s konkrétním typem fyzické vrstvy, která má největší vliv na energetickou spotřebu přenosu. Avšak s velikostí zpráv a režií pro zasílání dat také přímo souvisí spotřeba energie a množství zdrojů potřebných pro přenesení zprávy. Z tohoto důvodu je pořadí protokolů podle spotřeby energie stejné, jako u velikosti zpráv [20].

Pořadí v oblasti spolehlivosti přenosu zpráv reflektuje zejména to, do jaké míry má protokol implementované režimy QoS. V této oblasti je nejlepší řešení využité protokolem MQTT, který disponuje třemi různými režimy QoS. Protokoly CoAP a AMQP nabízí dva režimy, které nastavují chování protokolu pro zajištění kvality služeb. Rozdíl mezi těmito dvěma protokoly tvoří fakt, že AMQP využívá navíc TCP transportní protokol, oproti UDP u CoAP, takže i „nespolehlivý“ režim zajistí TCP stabilní spojení. HTTP nenabízí ve standardu žádné režimy QoS [20].

3.2 Zigbee

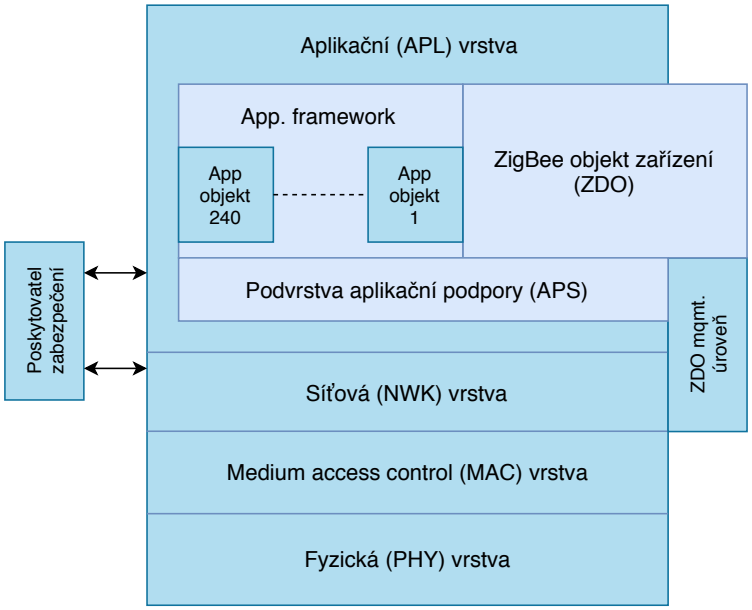
Zigbee je bezdrátová síť vyvinutá aliancí Zigbee Aliance pro aplikaci datově nenáročných senzorů v malém rozsahu. Struktura Zigbee protokolu je rozložena do čtyř vrstev: fyzická vrstva (PHY), vrstva medium access layer (MAC), síťová (NWK) vrstva a aplikační (APL) vrstva. Zigbee poskytuje bezpečnostní funkcionalitu napříč všemi vrstvami. Spodní dvě vrstvy protokolu jsou definovány standardem IEEE 802.15.4. Zbylé vrstvy určuje ZigBee specifikace. Na základě IEEE 802.15.4 je definovaná pracovní frekvence v pásmu 868 MHz, 915 MHz a 2.4 GHz [14].

ZigBee přistupuje k fyzické vrstvě pomocí dvou metod: beacon-enabled a beaconless. V režimu beacon-enabled se zařízení chová jako koordinátor Personal Area Network (PAN) sítě a vysílá synchronizační signály. V režimu beaconless umožňuje IEEE 802.15.4 potvrzovací rámce pro unicastové přenosy. Zigbee zařízení může být využito ve třech rolích. První je role Zigbee koordinátora, druhá je role routeru a třetí je koncové zařízení [14].

Síťová vrstva Zigbee protokolu podporuje adresování a směrování pro stromovou a mesh topologii. Stromová topologie se rozpíná od kořene, který je Zigbee koordinátorem sítě. Toto schéma zahrnuje také mechanismy pro přiřazování adres a usnadňuje multihop data delivery. V mesh topologiích jsou trasy vytvářeny na základě požadavků a jsou spravovány pomocí ad hoc on-demand distance vector (AODV) směrovacího protokolu [14].

3.3 Z-Wave

Z-Wave je architektura bezdrátového protokolu vyvíjená společnostmi ZenSys a později Sigma Design. Je podporována a komerčně podněcována aliancí Z-Wave Aliance. Hlavním



Obrázek 3.2: Architektura Zigbee systému. Převzato z [14].

účelem protokolu Z-Wave je zajistit spolehlivý přenos malých zpráv. Architektura systému je rozložena do pěti hlavních vrstev: PHY, MAC, transfer, směrovací a aplikační vrstva [14].

Z-Wave nejčastěji pracuje v pásmu 900 MHz (868 MHz v Evropě, 908 MHz ve Spojených státech) a v poslední době s čipy Z-Wave 400 series také podporuje volné ISM pásmo 2.4GHz [14].

Vrstva MAC definuje mechanismy pro zabránění kolizí a umožňuje přenos datových rámců přes volné kanály. V opačném případě se přenos zpozdí o náhodnou časovou periodu. Transfer vrstva spravuje komunikaci mezi dvěma uzly a tato komunikace je řízená pomocí potvrzovacího mechanismu [14].

Z-Wave definuje dva typy zařízení: controllers a slaves. Zařízení typu Controller rozesílá příkazy skupině Slaves, kteří tyto požadavky vykonávají [14].

Směrovací vrstva zajišťuje směrování na základě možností zdrojového zařízení. Z-Wave controller přenáší paket, který obsahuje informace o cestě, kterou má být směrován. Paket může být přenesen až do čtyř kroků. Zařízení typu Controller si uchovává tabulku s informacemi o celé topologii sítě. Přenosný controller se nejprve snaží přenést paket přímo k příjemci, ale pokud to není možné, pak controller navrhne nejlepší trasu k cíli. Zařízení Slave také může figurovat jako směrovač, pokud si uchovává statické trasy a může přeposílat zprávy dalším uzlům [14].

Kapitola 4

Platforma senzoru

První část systému je senzor detekující primárně Bluetooth beacon. Komunikuje s Home Assistant instancí a periodicky ji předává informace o svém stavu. Senzor může být také rozšířen o další možnosti detekce. Návrh počítá také s použitím PIR čidla, ať už jako samostatný senzor a nebo v kombinaci s Bluetooth detektorem. Při návrhu senzoru bude nutné si uvědomit, kde a jak bude zařízení používáno a stanovit si veškeré požadavky, které musí senzor splňovat. Poté můžeme vybrat správné komponenty použité pro náš účel.

Dále se podíváme na implementační detaily použití při vytváření řídícího programu senzoru. V rámci implementace je nutné vyřešit několik problémů. Prvním z nich je zajistit nejnižší možnou spotřebu energie a vhodnou optimalizaci s použitím Ultra Low Power (ULP) koprocesoru. Dále je potřeba zajistit automatické zotavení se z nečekaných chyb, například přerušení Wi-Fi spojení. V neposlední řadě je vhodné, aby senzor zaznamenával logy o své činnosti a exportoval je do stanovené stanice.

4.1 Požadavky na systém

Jelikož senzor chceme používat v mnoha místnostech v domě a nejlépe v celém bytě, klademe na něj jisté požadavky. Přestože senzor neobsahuje žádnou složitou logiku, potřebujeme primárně zajistit komunikaci s Home Assistant stanicí. Z toho důvodu je potřeba použít System on Chip (SoC) mikrokontrolér.

Aby instalace senzorů v domě byla dostatečně pohodlná a nebyla příliš destruktivní k našim domovům, chceme aby komunikace probíhala bezdrátově. V oblasti IoT se pro tyto účely využívají různé standardy rádiové komunikace, mezi které patří například Wi-Fi, Z-Wave a nebo Zigbee. Standard Z-Wave, který má také podporu v platformě Home Assistant, je nízkoenergetický bezdrátový komunikační protokol pracující ve frekvenčním pásmu 800 až 900 MHz. Z-Wave zařízení tvoří Mesh netvork a komunikace mezi nimi prochází přes různé prostředníky [14]. Zigbee je další rádiový komunikační standard založený na IEEE 802.15.4. Zigbee síť tvoří koordinátor, přes který všechna zařízení navzájem komunikují [13]. Avšak na základě předpokladu, že v chytré domácnosti můžeme počítat s téměř plným pokrytím signálu Wi-Fi a při návrhu našeho senzoru se snažíme docílit jednoduchosti a nízké ceny, je nejvhodnější způsob připojení senzorů klasická lokální počítačová síť. Z toho vyplývá další požadavek, že vybraný mikrokontrolér musí podporovat toto bezdrátové připojení.

Primární způsob detekce osob v navrhovaném systému je pomocí hledání Bluetooth zařízení. Proto je důležité, aby vybraný mikrokontrolér (MCU) měl v sobě toto rozhraní a hlavně verzi BLE již integrovanou. Pro dodatečné senzory, například PIR čidlo, požada-

vek Bluetooth není potřebný, avšak v tomto případě potřebujeme mít k dispozici dostatek General-purpose input/output (GPIO) pinů. Další důležitou vlastnostií je spotřeba elektrické energie počítače. Ze stejného důvodu jako u použití bezdrátové komunikace je vhodné, aby bylo možné senzor dlouhodobě a spolehlivě využívat pouze na baterii. Jelikož senzor ve své činnosti pouze odesílá jednou za určitou periodu informace o svém stavu, není nutné, aby po ukončení této činnosti musel stále běžet a aktivně čekat. Proto vybraný mikrokontrolér by se měl umět na určitý čas „uspat“ a poté automaticky opakoval hledání a zaslání dat. U varianty senzoru s PIR čidlem mohou existovat specifické požadavky určené umístěním, například doba aktivace, dosah detekce, úhel detekce a další.

4.2 Výběr komponent

Na základě požadavků sepsaných v sekci 4.1 musíme vybrat vhodný mikrokontrolér pro naše použití. Typ a výkon použitého procesoru není velmi omezen, jelikož funkce, pro kterou bude využíván, není náročná a největší zpožďovací prvek bude ve všech případech hledání Bluetooth zařízení, které vyžaduje svůj čas.

Například otevřená prototypovací platfoma Arduino nabízí intuitivní vývojové prostředí a velkou komunitu, ale pro tento účel není vhodná, protože dostupné vývojové desky neobsahují potřebné bezdrátové technologie [5]. Ty lze rozšířit pomocí přídavných modulů, avšak například nejrozšířenější Wi-Fi karta pro Arduino desky je ESP8266, která sama obsahuje výkonnější a vybavenější SoC. Oproti 8-bit procesorů ATmega použitých v produktech Arduino, disponuje ESP8266 32-bit mikrokontrolérem se základním taktem 80 MHz, ale postrádá také Bluetooth. Jako hlavní BLE senzor je také nevhodný, avšak bohatě postačí pro ovládání PIR čidla a díky jeho nízké pořizovací ceně je ideální pro použití ve velkém množství. Přímý nástupce tohoto MCU od firmy Espressif Systems je ESP32.

ESP32 je mikrokontrolér pro veřejnost vydaný v roce 2016 a na rozdíl od svého předchůdce, který byl vytvořen primárně jako Wi-Fi čip a jeho potenciál pro IoT objevila následně až komunita, je díky nízké ceně kolem $5 a integrovaným Bluetooth 4.2 BLE a BR/EDR velmi vhodným základem pro primární senzor. Zbývající specifikace ESP32 jsou popsány v sekci 4.3. Mezi další možnosti lze zvážit jednodeskové počítače s ARM mikroprocesory. Ty disponují často nejen vysokým výkonem, velkým množstvím periferií a komunikačních rozhraní, ale také mohou být provozovány plnohodnotnou linuxovou distribucí. Avšak jejich pořizovací hodnota, výkon a spotřeba zdaleka předimenzovává potřeby provozu jednoúčelového senzoru.

4.3 Mikrokontrolér ESP32

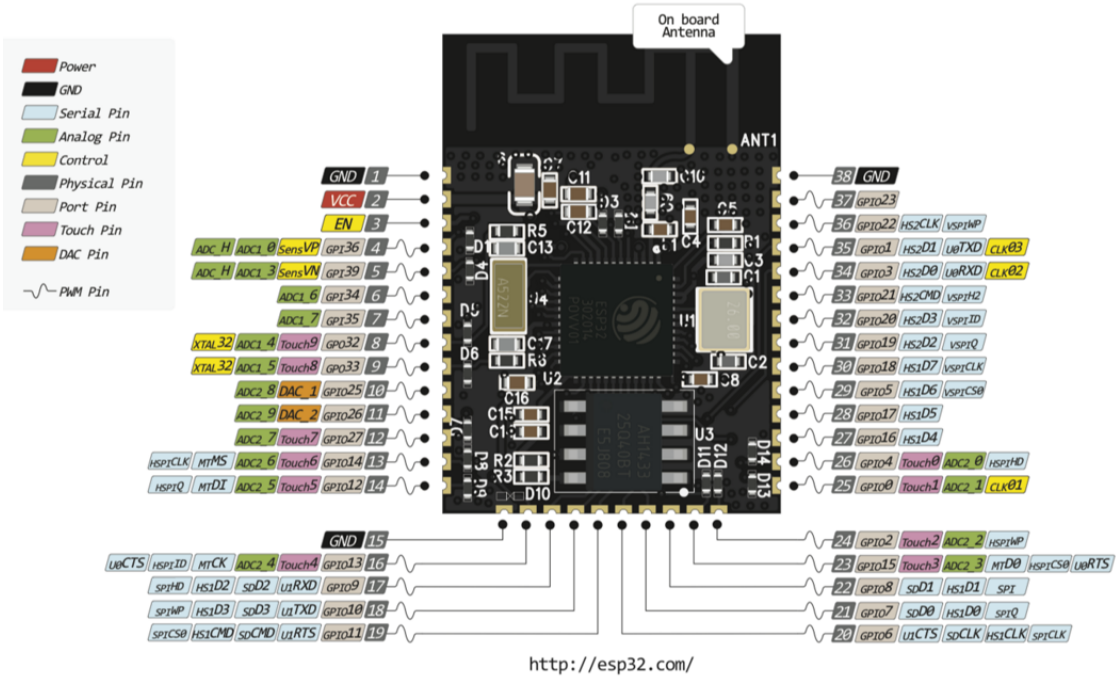
Jak bylo v části 4.2 zmíněno, mikrokontrolér ESP32 je vhodným kandidátem pro použití hledání Bluetooth zařízení. Jeho základní parametry jsou sepsány v tabulce 4.1. Čínská společnost Espressif Systems jej vyrábí od roku 2016 v několika provedeních. Varianty se liší zejména ve velikosti osazené paměti a existuje i ESP32 čip s jedním jádrem procesoru Xtensa LX6. Espressif Systems ale i další výrobci produkují na základě těchto MCU vývojové desky a kity v různých konfiguracích, například s baterií, LED displejem a nebo tlačítky [18].

Přestože ESP32 obsahuje celou řadu komunikačních rozhraní, viz tabulka 4.1, nejdůležitějším z nich je asynchronní sériové rozhraní Universal Asynchronous Receiver/Transmitter (UART). To lze jednoduše převést na Universal Serial Bus (USB), které je vhodné pro připojení PC k tomuto MCU. Dále je to primární rozhraní pro přepisování Flash paměti

Specifikace	ESP32
CPU	2 x Tensilica Xtensa LX6 microprocessor @ 160 or 240 MHz
Paměť	448 KiB ROM, 520 KiB SRAM, až 16 MiB Flash
Wi-Fi	802.11 b/g/n with WFA, WPA/WPA2 and WAPI
Bluetooth	v4.2 BR/EDR, BLE
I/O periferie	ADC, DAC, I^2C, UART, CAN 2.0, SPI, I^2S, RMII, PWM
Šifrování	AES, SHA-2, RSA, ECC
Jiné	RNG, ULP koprocesor, 34 x GPIO

Tabulka 4.1: Specifikace mikrokontroléru ESP32. Převzato z [18].

mikrokontroléru a může sloužit také jako tunel pro komunikaci se vzdáleným terminálem. ESP32 má pro UART vymezené piny *TX* a *RX*. Při bootování se MCU pokouší automaticky odhadnout přenosovou rychlost rozhraní na druhé straně, při výchozím nastavení je tato hodnota 115200 Bd [18].



Obrázek 4.1: Pouzdro ESP32-WROOM-32 modulu. Převzato z [18, s. 64].

Procesor Tensilica Xtensa LX6, který je obsažen v MCU ESP32, může být poháněn hodinovými signály z několika možných zdrojů. Samotné ESP32 má vnitřní oscilátor o frekvenci 8 MHz, avšak výchozím zdrojem hodinového signálu je externí krystal. Ten je připojený k PLL obvodu, aby generoval typický hodinový signál o frekvenci 160 MHz. Samotný procesor si dokáže přímo řídit zdroj signálu a také na základě aplikace tento signál podle potřeby dělit [12].

Procesor má také k dispozici pět zdrojů Real-time clock (RTC) reálného hodinového signálu:

- externí nízkorychlostní krystal — 32 kHz
- externí krystal dělený čtyřmi — 32 / 4 kHz
- interní RC oscilátor (nestabilní okolo 150 kHz)
- interní 8 MHz
- interní 31.25 kHz = 8 / 256 MHz

Externí zdroje signálu jsou přesnější, avšak také energeticky náročnější. Procesor v normálním režimu si může vybrat kterékoli z externích řešení. V režimu nízké spotřeby může využít některý z interních zdrojů [12].

Externí flash paměť může být osazena pomocí několika SPI linek. ESP32 obsahuje čtyři SPI kontroléry, z nichž tři jsou připojeny ke společným dvěma DMA kanálům. Poslední z SPI kontroléru (SPI0) je určen pro připojení externí paměti a je spojen s rychlou cache pamětí. Každý SPI kontrolér obsahuje čtyři full-duplex a tři half-duplex linky. Procesor mikrokontroléru ESP32 však dokáže zamapovat maximálně 16 MB flash paměti. Pokud namapována externí flash paměť je do velikost 4 MB, pak je podporováno čtení v režimu 8 bit, 16 bit a 32bit. Dále pak pokud je namapována paměť vetší než 3 MB, pak se snižuje výkonnost cache paměti, kvůli spekulativnímu čtení procesoru. ESP32 podporuje také hardwarové šifrování programové i datové části v flash paměti pomocí symetrické šifry AES pro ochranu dat [11].

ESP32 má zabudované čtyři 64 bitové časovače, určené pro obecné použití. Všechny časovače mají k dispozici 16 bitový dělič frekvence (prescaler) a podporují up/down režim. Dalšími funkcemi časovačů jsou generátory přerušení na základě hrany nebo úrovně, automatické a programově řízené okamžité znovunačtení [12].

ESP32 má k dipozici tři Watchdog časovače, které resetují systém při chybě nebo zacyklení. Watchdog časovač pracuje ve čtyřech fázích a každá z těchto fází může spustit jednu ze tří nebo čtyř akcí po expiraci naprogramovaní časové periody. Mezi možné akce patří: přerušení, reset procesoru, reset jádra a reset systému. Časový limit lze pro každou fázi nastavit zvlášť [12].

Wi-Fi je implementována na základě TCP/IP architektury a standardu 802.11 b/g/n. Dokáže pracovat v režimu klientské stanice (STA) a nebo vytvořit AP v režimu SoftAP. Správa napájení je definována s ohledem na nízkou spotřebu energie a je toho docíleno pomocí minimální možné interakce a maximální možné periody aktivní práce. Wi-Fi v ESP32 podporuje také použití více antén a poté jejich přepínání na základě rádiového přepínače. Tento RF přepínač může být řízen pomocí jednoho nebo více GPIO portů, aby se vybrala vždy nejlepší anténa s nejmenšími ztrátami [12].

ESP32 má integrovány Bluetooth se základními hardwarovými vrstvami a protokoly, které zajišťují modulaci, zpracování paketů, FHSS a směrování. Rádiová vrstva Bluetooth podporuje všechny tři režimy výstupního přenosového výkonu: Class-1, Class-2 a Class-3. Podporuje modulaci typu: $\pi/4$ DQPSK a 8 DPSK. Poskytuje logiku pro opravu chyb, kontrolu hlaviček, CRC, šifrování bitového streamu. Nabízí správu napájení a podporuje SBC audio kodek [12].

Další velkou výhodou ESP32 je kromě rozsáhlé konektivity jeho spotřeba energie. Ta sice závisí zejména na aplikaci a při plném nasazení je při provozním napětí 3.3 V potřebný elektrický proud 260 mA, avšak pomocí ULP koprocesoru v hlubokém spánku potřebuje ESP32 pouze 20 µA [18].

Příkladem ESP32 modulu, který se často vyskytuje na vývojových deskách, je například *ESP-WROOM-32*, jehož vyvedení pinů je ilustrováno na obrázku 4.1. Tento modul vyráběný firmou Espressif Systems integruje ve svém těle 4 MiB flash paměť. Z těla obalu, které má velikost 18 mm x 25.5 mm, je vyvedeno 38 pinů, ze kterých je 6 určeno pro ovládání flash paměti a není možné je využít k jiným účelům [18].

S vydáním ESP32 firma Espressif Systems představila také referenční vývojovou desku *ESP32-DevKitC*. Na desce lze nalézt micro USB port připojený k USB UART převodníku, který slouží pro napájení, přepisování paměti a komunikaci přes UART rozhraní. Dále jsou na desce umístěna dvě tlačítka nazvaná *EN* a *BOOT* [18].

4.4 Návrh komunikačního protokolu

Na základě požadavků chceme, aby MCU komunikoval s Home Assistant stanicí prostřednictvím LAN sítě připojené pomocí Wi-Fi. Výběr použité technologie není jedinou volbou v oblasti komunikačních protokolů. Je třeba si stanovit aplikační protokol, pomocí kterého se budou přenášet potřebné informace.

Nejprve si řekněme, jaká data bude senzor centrální stanici zasílat. Dvojhodnotová informace o tom, jestli senzor má v dosahu nějaké Bluetooth zařízení, není dostačující. Může se totiž stát, že v dané místnosti jsou nějaká neosobní Bluetooth zařízení, která se sice v místnosti nachází a mohou vysílat Bluetooth advertisment, ale svou přítomností neindikují i přítomnost člověka. Tato zařízení je potřeba odfiltrovat od vyhodnocení situace v místnosti, a proto je nezbytné je jednoznačně identifikovat. K identifikaci postačí zaslat BD_ADDR daného Bluetooth rozhraní. Přestože hledáme primárně BLE beacon a ten nevysílá velmi vysokým výkonem, takže signál těžko pronikne přes zdi domu, bude potřeba zajistit jejich filtrování od určité hladiny síly signálu RSSI. Nakonec chceme posílat řetězec, který bude obsahovat pro každé nalezené zařízení informaci o adrese rozhraní a sílu signálu. Tato data je nutné zapouzdřit vhodnou standardní notaci. V tomto případě se zcela nabízí použití JSON notace, protože je jednoduše implementovatelná a nepřidává do řetězce nadbytečná data (typ kódování, hlavičky). Výsledný řetězec se skládá ze struktur a polí a hodnoty lze vyjádřit jako string (řetězec ohraničený dvojitými uvozovkami), number (číslo tvořené číslicemi), boolean („true“, „false“) a prázdná hodnota „null“. Příklad 4.1.

```
{
    "00:00:00:00:00:00" : -56,
    "11:11:11:11:11:11" : -77,
    "22:22:22:22:22:22" : -91
}
```

Výpis 4.1: Příklad přenášených dat ve formátu JSON.

Pro přenos zpráv ze senzoru do centrální stanice je k dispozici mnoho možností. Některé vhodné standardy pro vytváření senzorových sítí jsou popsány v kapitole 3. Všechny uvedené možnosti bývají pro podobné účely využívány a všechny dokážou splnit požadovanou funkcionalitu. Na poli rozdílů mezi jednotlivými rádiovými standardy lze nalézt několik zásadních prvků, které jsou rozhodující ve výběru technologie a návrhu senzorové sítě. Technologie typu Z-Wave nebo Zigbee nabízí oproti Wi-Fi například výrazně nižší spotřebu a topologie typu mesh používaná v těchto standardech je vhodná pro senzorové sítě, avšak

na základě požadavků, kterými byly nízká cena a dostupnost, byl vybrán mikrokontrolér ESP32 a ten již v základu podporuje Wi-Fi. Navíc není nutné pořizovat Gateway zařízení, která tvoří bránu mezi různými druhy sítí.

Kapitola 3 se také zabývala srovnáním internetových komunikačních protokolů, vhodných ke komunikaci typu M2M. Z tohoto srovnání vyšlo najevo, že nejvhodnějším protokolem je MQTT, protože nabízí dostatečnou spolehlivost a malou velikost přenášených zpráv.

4.5 Implementační jazyk a vývojový framework

Výběr použitého programovacího jazyka a způsobu zavádění programu do MCU je omezen typem použitého mikrokontroléru. ESP32, zvolená vývojová platforma pro tento projekt, nabízí široké možnosti vývoje. Základní a nativní možností je použití oficiálního *ESP-IDF* frameworku. Ten nabízí všechny základní operace, jako překlad C a C++ programů, nahrávání přeložené aplikace do flash paměti, úpravu tabulky diskových oddílů a další. Navíc *ESP-IDF* využívá pro sou činnost operační systém typu FreeRTOS, který dokáže efektivně využít relativně výkonný dvoujádrový procesor v čipu ESP32 a obsluhovat velké množství periferií [9].

Přímou alternativou k ESP-IDF je programování pomocí Arduino frameowrku. Oproti nativnímu ESP-IDF nabízí odlišnou koncepci. Výhodou Arduino frameworku je zejména jednoduchost jeho použití a možnost napsat funkční program, který je možné nasadit i na jiné platformy, než je ESP32. Dále Arduino framework nabízí velké množství vytvořených knihoven, které mohou výrazně ulehčit vývoj aplikace využívající například specifický hardware [9].

Předchozí nástroje slouží pro programování vývojových desek v jazyce C/C++. U programování embedded systému jsou tyto jazyky výhodné, protože jsou dostatečně nízkoúrovňové a umožňují programátorovi provést různé paměťové optimalizace. Existují ale i další možnosti. Nástroj MicroPython zpřístupňuje Python pro ESP32 a umožňuje spouštět Python programy přímo v mikrokontroléru. Totéž nabízí například Espuino pro JavaScript a nebo Mruby pro Ruby. Na ESP32 lze provozovat také různé RTOS, které se starají například o lepší správu paměti [9].

4.6 Vývojové prostředí

Mezi oblíbené vývojové prostředí pro programování vestavěných systémů patří Arduino IDE[1]. To nabízí jednoduché prostředí a možnost programování ESP32 desek pomocí dodatečného přídavného externího modulu od Espressif Systems. Tuto funkcionalitu lze jednoduše přidat pomocí manažéru desek v Arduino IDE. V grafickém prostředí aplikace je možné vybrat správnou vývojovou desku, nastavit rychlost nahrávaní programu do kitu, vybrat správný port pro nahrání a zobrazit různé informace o desce. V prostředí Arduino IDE je také integrován sériový monitor pro výpis dat zapsaných na sériový port MCU. Arduino IDE také obsahuje grafického správce knihoven. Příliš jednoduchý editor zdrojových souborů je hlavním nedostatkem vývojového prostředí a není vhodný pro komfortní správu větších projektů.

[1] Viz: https://www.arduino.cc/en/Main/Software

Nejznámější alternativou k vývoji v Arduino IDE je použití PlatformIO[2] IDE. Toto open source multiplatformní prostředí, napsané v jazyce Python, neexistuje ve formě explicitní aplikace, avšak lze jej používat prostřednictvím oficiálního rozšíření do externích textových editorů Atom a Visual Studio Code. Z hlediska lepší integrace a lepší funkcionality je výhodnější použít editor VSCode. Systém PlatformIO nabízí kompletní podporu pro vývoj vestavěných systémů. Samozřejmostí jsou nástroje pro kompilaci zdrojového kódu a nahrávání na vývojovou desku. PlatformIO nabízí monitor sériového portu, podobně jako Arduino IDE. Veškeré nastavení celého projektu na nachází v konfiguračním souboru *platformio.ini*, který je umístěn v kořenovém adresáři projektu. Editor zdrojového programu nabízí možnosti domovské aplikace rozšíření PlatformIO. Obě oficiální aplikace Atom a VSCode nabízí přehledné prostředí a mnoho dalších rozšíření editoru vylepšující vývojový zážitek. Platform IO lze také použít s jinými vývojovými prostředími, například Eclipse IDE.

4.7 Potřebné knihovny

Seznam knihoven potřebných pro vývoj aplikace závisí zejména na vybraném vývojovém frameworku. Pokud bychom použili Arduino framework, máme k dispozici jejich celou řadu. Protože pro vývoj používáme prostředí PlatformIO, pro nalezení potřebných knihoven můžeme použít jeho databázi a funkce pro stahování a instalování potřebných závislostí zajistí všechny soubory před samotnou kompilací programu.

Z této databáze je potřeba použít zejména základní Arduino knihovnu, která zpřístupňuje třídy a funkce pro práci s jednotlivými částmi MCU. Jelikož senzor předává své informace prostřednictvím počítačové sítě, tudíž je třeba také použít dodatečnou Arduino knihovnu *Wi-Fi*, která obsahuje potřebné funkce pro navazování a správu spojení. Dále senzor má za úkol hledat ve svém okolí BLE zařízení. Funkce potřebné pro tuto činnost poskytuje knihovna *ESP32 BLE Arduino* od Neil Kolban a Dariusz Krempa. Tato knihovna je řešena objektově a z její sady tříd je potřeba použít třídy *BLEDevice*, *BLEUtils*, *BLEScan* a *BLEAdvertisedDevice*. Funkci MQTT klienta v Arduino frameworku plní například knihovna PubSubClient. Ta sice není součástí základní sady Arduino knihoven, ale je open source distribuována pod licencí MIT.

Pokud použijeme vývojový framework ESP-IDF, počet potřených knihoven pro funkci programu je minimální. Jelikož je ESP-IDF referenční sada nástrojů pro vývoj programu na ESP32, musí sám obsahovat a podporovat všechny funkce mikrokontroléru. Tudíž v tomto případě nejsou nutné žádné knihovny. Avšak je vhodné si při vývoji pomoct například s implementací aplikačního protokolu MQTT. V databázi PlatformIO je možné najít například knihovnu *MQTT library for ESP32 (ESP-IDF)* od Andrey Mitrokhin, která komunikaci pomocí tohoto protokolu umožňuje.

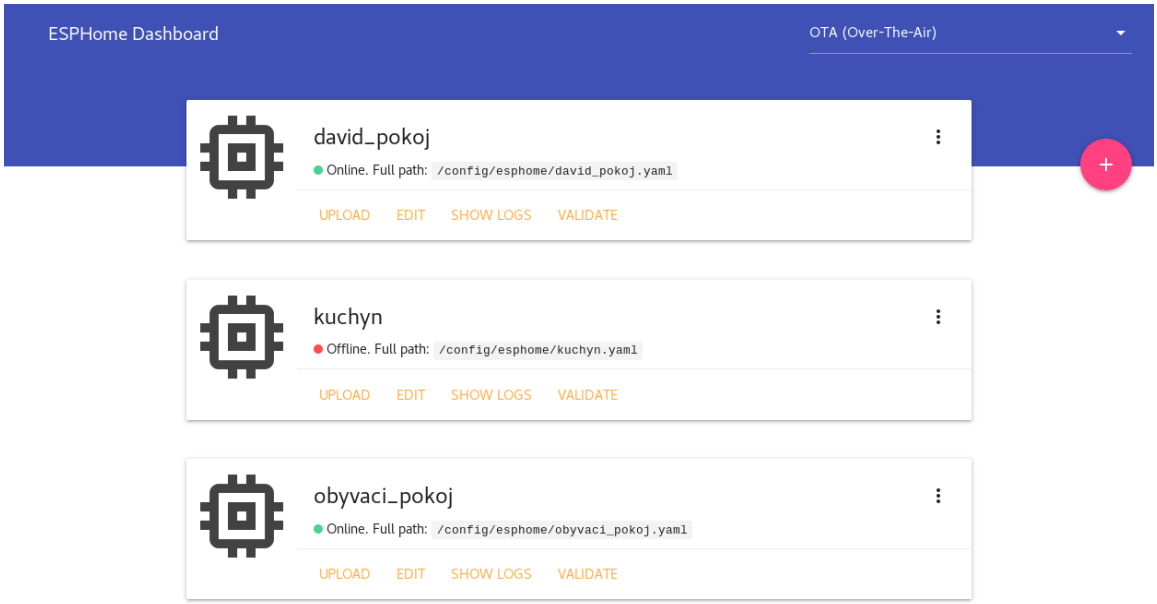
4.8 ESPHome framework

ESPHome je open source systém pro vytváření, řízení a spravování zařízení postavených na platformách ESP32 a ESP8266. Jeho cílem je nabídnout možnost jednoduše stavět vlastní vestavěné systémy, které se dokáží automaticky integrovat do domácí automatizační platformy Home Assistant. Značnou výhodnou prostředí PlatformIO je velké množství hardwarových platforem a také podporovaných vývojových frameworků [10].

[2]Viz: https://platformio.org/

ESPHome nabízí webový dashboard, který je možné použít pro vytváření popisů senzorů a generování, kompilování a nahrávání na ESP mikrokontrolér. Nabízí také textový výstup pro výpis logu. Připojení k ESP je možné pomocí USB portu na zařízení a nebo Over The Air (OTA) [10]. Ukázka rozhraní je vykreslena na obrázku 4.2.

Výchozí způsob, jakým lze funkci vytvářeného zařízení popsat, je pomocí YAML souboru. YAML je serializační formát, mezi jehož hlavní přednosti pro člověka patří snadná čitelnost. Soubor, který je možné použít pro vygenerování patřičného firmware, použitelného pro vybrané MCU, musí obsahovat objekt s názvem *esphome*. Tento objekt definuje název (name) zařízení, podle kterého lze zařízení v Home Assistantu identifikovat, dále pak informaci o platformě (platform) a nakonec o vybraném vývojovém kitu (board) [10]. Příklad takové konfigurace je vypsán ve výpisu 4.2.



Obrázek 4.2: Webové rozhraní ESPHome. Zobrazuje seznam a stav nasazených zařízení.

Objekt *esphome* nedefinuje převážně funkce, ale jeho nastavení ovlivňuje kompilaci zdrojového kódu. Je možné například určit verzi *Arduino* frameworku používaného tímto projektem, dále pak verzi esphome-core knihovny obsahující funkce pro ESP zařízení a adresář uložení přeložených binárních dat. ESPHome využívá pro sestavování programu systém PlatformIO, jehož chování je definované konfiguračním souborem *platformio.ini*. Avšak v rámci ESPHome tento soubor nevytváříme, nýbrž si ho framework generuje sám. Z toho důvodu se nachází v objektu *esphome* také volitelné možnosti *platformio_options*, *includes* a *libraries*. Atribut *libraries* obsahuje seznam potřebných PlatformIO knihoven, které si systém při kompilaci automaticky stáhne [10].

```
esphome:
    name: kuchyn
    platform: ESP32
    board: esp32doit-devkit-v1
```

Výpis 4.2: Příklad minimální nutné konfigurace jádra ESPHome programu

Výsledný firmware lze sestavit přidáním dalších YAML objektů. Důležitou *core* komponentou je Wi-Fi. Jak už název vypovídá, jedná se o součást systému, která se stará o Wi-Fi připojení. Základní nastavovací atributy jsou *ssid*, pro zadání jména AP a *password*, heslo k jeho připojení. Wi-Fi komponenta v ESPHome navíc umožňuje najednou nastavit více sítí a poté se dokáže MCU připojit k nejlepší z nich. Samozřejmostí je možnost manuálně nastavit IP adresu. Toho lze dosáhnout pomocí atributu *manual_ip*, jehož hodnotou je struktura obsahující *static_ip*, *gateway* a *subnet*. Wi-Fi komponenta dokáže spustit Wi-Fi i v režimu AP, aby se na něj mohla jiná zařízení připojit. Tento režim se konfiguruje pomocí struktury uložené v atributu AP. Zajímavou funkcí této komponenty je možnost provozovat Wi-Fi připojení ve třech různých energeticky úsporných režimech. Výchozím módem je režim *NONE*, který nabízí nejmenší úsporu energie, ale zato lepší stabilitu spojení. Další dva *LIGHT* a *HIGH* se liší výslednou spotřebou a frekvencí odpojování se od AP [10].

```
wifi:
  ssid: SSID
  password: PASSWORD

  manual_ip:
    static_ip: 192.168.1.100
    gateway: 192.168.1.1
    subnet: 255.255.255.0
```

Výpis 4.3: YAML konfigurace Wi-Fi připojení.

Další důležitou *core* komponentou je MQTT. Ta, jak již z názvu vyplývá, řídí komunikaci pomocí protokolu MQTT. Konfigurační YAML objekt MQTT obsahuje atributy *broker* pro nastavení adresy MQTT brokera, dále pak atributy pro nastavení přístupových údajů k brokeru. Dále je možné nastavit různé automatizace pro události *on_message* a *on_json_message*. Komponenty, které umožňují zasílání dat, například binary senzor, po správné konfiguraci komponenty MQTT automaticky zasílají data také přes tento protokol. Specifické chování MQTT protokolu u jednotlivých částí lze dále upřesnit v rámci daného objektu. Těmito specifickými možnosti je například explicitně definovaný topic, na který MQTT data zasílá, dále pak je možné povolit či zakázat MQTT discovery funkci pro objevování nových zařízení v platformě Home Assistant. Implementace MQTT protokolu v ESPHome frameworku umožňuje provozovat komunikaci pomocí tohoto protokolu zabezpečeného SSL šifrování. SSL funkce jsou avšak omezeny několika problémy. První omezení je použití pouze na mikrokontroléru ESP8266 a nikoli ESP32. Avšak funkce pro toto MCU by měla být v budoucnu přidána. Další limitací SSL protokolu v ESPHome je ověřování integrity certifikátů pouze pomocí SHA-1 hash hodnoty. Bohužel je známo, že hashovací funkce SHA-1 není již bezpečná a je možné vytvořit falešný otisk. Nastavení SSL certifikátů je možné pomocí atributu *ssl_fingerprints*, jehož hodnotou je seznam jejich otisků [10].

Mikrokontroléry ESP32 a ESP8266 umožňují aktualizovat firmware pomocí OTA aktualizace. Framework ESPHome tuto funkci u obou MCU dokáže využít. OTA komponenta se nastavuje pomocí *ota* objektu v YAML souboru, který obsahuje atributy: *save_mode*, *password*, *port* a *id*. Atribut *save_mode* je volitelný atribut typu boolean a určuje, zda aktualizace pracují v bezpečném režimu. Ve výchozím nastavení je tento režim zapnutý. Další položky nastavení OTA aktualizací jsou také nepovinné a nastavují například přístupové heslo *passsword* a port používaný pro přenos aktualizace. Výchozí porty používané pro aktualizace jsou 3232 pro ESP32 a 8266 pro ESP8266 [10].

Důležitou komponentou, kterou MCU ESP32 disponuje, je Bluetooth a BLE. Na rozdíl od BLE nemá ESPHome pro základní verzi Bluetooth definované konkrétní použití. Pro již zmíněnou variantu LE poskytuje ESPHome *ESP32 Bluetooth Low Energy Tracker Hub*, který dokáže trackovat BLE zařízení pomocí ESP32. Konfigurace samotné komponenty v YAML souboru umožňuje definovat pouze časový interval, během kterého hledá nová zařízení. Výchozí hodnota tohoto intervalu je stanovena na 300 s [10]. Příklad konfigurace je ukázán ve výpisu 4.4.

```
esp32_ble_tracker:
  scan_interval: 300s

binary_sensor:
  - platform: esp32_ble_tracker
    mac_address: 00:11:22:33:44:55
    name: "BLE Tracker"

sensor:
  - platform: ble_rssi
    mac_address: 00:11:22:33:44:55
    name: "BLE RSSI"
```

Výpis 4.4: YAML konfigurace BLE trackeru.

Jelikož jsou knihovny poskytující funkce Bluetooth adaptéru velmi obsáhlé, je nutné při prvním nasazení na ESP32 nahrát firmware pouze pomocí USB, protože se musí nasadit systém s novým rozložením úložiště. Funkcí této komponenty je pouze periodicky vyhledávat zařízení, avšak na základě této činnosti jsou postaveny další součásti ESPHome, které poskytují informace například Home Assitant stanici [10].

První BLE součástí je binární senzor platformy *esp32_ble_tracker*. Ten umožňuje pro každou zadanou MAC adresu BLE zařízení definovat stav v podobě binární hodnoty. Ta představuje přítomnost daného zařízení v jedné periodě skenovacího intervalu komponenty *esp32_ble_tracker*. Definovaný senzor poté zasílá informaci pomocí protokolu MQTT nebo pomocí ESPHome API. Pokud je v systému Home Assistant povoleno hledání nových zařízení, měl by se tento binární senzor také automaticky integrovat [10].

Dalším druhem senzoru využívající BLE je *BLE RSSI Senzor*. Ten poskytuje informaci o RSSI síle signálu z BLE zařízení. Konfigurace takového senzoru je podobná jako u předchozího typu senzoru. Pro každé zařízení se nastavuje pouze jeho MAC adresa a jméno, které se použije při integraci v Home Assistentu [10].

ESPHome podporuje pomocí komponenty BLE trackeru typu senzor, které pracují s některými proprietárními zařízeními od společnosti Xiaomi. Tyto komponenty získávají data z přijatých Bluetooth advertisement paketů, které obsahují informace například o teplotě nebo vlhkosti a přeposílají je Home Assistant stanici pomocí MQTT protokolu [10].

Jako pomocná komponenta pro vývoj a sledování stavu zařízení slouží *logger*. Ten nabízí několik logovacích úrovní odlišných mírou detailu zpráv, jejich množstvím a také barvou výpisu. Mezi tyto úrovně patří: NONE (žádné zprávy), ERROR (chybové zprávy), WARN (varovné zprávy), INFO (informační), DEBUG (ladící výpisy), VERBOSE („ukecaný“ režim), VERY_VERBOSE (mnoho interních zpráv, včetně dat na sběrnicích). Výchozí úroveň zobrazovaných výpisů je nastavena na DEBUG, takže se vypíšou všechny zprávy z této úrovně a vyšších. Kromě základní úrovně logu nastavitelného pomocí atributu *level* lze dále nastavit, který UART má být použit pro výpis a s jakou rychlostí má komunikovat

(argument *baud_rate*). Globální úroveň logu lze navíc manuálně upravovat pro jednotlivé komponenty. Atribut *logs* obsahuje seznam komponent, kde každá komponenta je definovaná pomocí názvu a její hodnotou je úroveň logu [10]. Příklad konfigurace ve výpisu 4.5

```
logger:
  level: VERBOSE
  logs:
    mqtt.component: DEBUG
    mqtt.client: ERROR
```

Výpis 4.5: YAML konfigurace logger komponenty.

ESPHome podporuje připojení celé řady různých senzorů a přepínačů a umožňuje také vytvářet nebo použít vlastní komponenty. Vlastní řešení je možné stavět například na základě tříd senzoru, binárního senzoru a nebo textového senzoru. Ty poté řeší obecnou funkcionalitu daného typu zařízení a automaticky poskytují zasílání zpráv pomocí všech dostupných nakonfigurovaných možností přenosu dat [10].

4.9 Implementace senzoru

Každý řídící program nějakého vestavěného systému je v základu tvořen dvěma funkcemi *loop* a *setup*. Funkce *setup* obsahuje příkazy pro počáteční nastavení a podmínky před spuštěním nekonečné smyčky funkce *loop*, která poté definuje chování celého systému. Na základě tohoto principu fungují všechny vývojové frameworky pro vestavěné systémy.

V průběhu vývoje senzoru jsem vytvořil hned tři jeho varianty a vyzkoušel několik způsobů stavby řídícího programu. Varianty mých tří vytvořených prototypů senzorů byly postavené na frameworku Arduino, ESP-IDF a výsledné řešení je nakonec vytvořené pro ESPHome, který využívá upravené Arduino knihovny. Správně fungující senzor s požadovanou funkcionalitou je možné vytvořit všemi způsoby, avšak každý z nich přináší jiná úskalí. Pro výsledný výběr použití ESPHome prostředí nakonec rozhodl fakt, že rozšíření stávajícího řešení o potřebnou funkcionalitu je ve všech ohledech výhodnější, než znovu vytvářet celý program od základu. Navíc ESPHome nabízí skvělý uživatelský komfort a modularitu programu, kterou by ostatní verze nebyly schopny dosáhnout.

4.9.1 Řešení Arduino

Hlavní definice programu, který využívá Arduino framework, je popsána pomocí funkcí *loop* a *setup*. Počáteční nastavení, definované příkazy ve funkci *setup* se skládají z volání inicializačních funkcí pro Wi-Fi, MQTT klienta a BLE adaptéru. Jednotlivé části jsou reprezentovány pomocí tříd, které obsahují všechny potřebné metody pro zajištění požadované funkcionality.

Třída *MQTTClient* reprezentuje klienta pro zasílání MQTT zpráv a její konstruktor a inicializační metoda *init* přijímá IP adresu a číslo portu MQTT brookeru a řetězec *topic*, pod kterým klient publikuje zprávy a druhé zařízení je odebírá na základě hodnoty *topic*. Konstruktor nastaví instanční proměnné a inicializuje *PubSubClient*. Další veřejnou metodou třídy *MQTTClient* je metoda *send*. Zasílá pomocí inicializované instance *PubSubClient* zprávu předanou pomocí parametru *payload*.

Část systému, která definuje funkci pro hledání BLE zařízení, formuje třída *BLETracker*. Ta zapouzdřuje další třídy, obsažené v knihovně *ESP32 BLE Arduino*. Knihovní třída *BLEDeviceBuffer* reprezentuje část senzoru, která sbírá informace o nalezených BLE zařízeních

a vrací data v JSON formátu. Reprezentuje data pouze z jednoho hledání. Konstruktor přijímá hodnotu typu *BLEScanResults* a z ní vybere adresy a hodnoty RSSI a uloží je do proměnné *deviceMap* typu *map*. Metoda *getPayload* dále zpracuje data z *deviceMap* a konvertuje do JSON podoby a nakonec vrací řetězec s připravenými daty.

Pomocí třídy *WIFIConnection* využívající Wi-Fi knihovnu ESP32 dále pak vytváří a udržuje spojení s AP. To je řešeno pomocí metody *loop*, která je volána při každé iteraci nekonečné smyčky programu, v níž kontroluje spojení a v případě chyby jej obnoví.

4.9.2 Řešení ESP-IDF

Framework ESP-IDF využívá ke svému řízení programu operační systém FreeRTOS. Ten umožňuje pokročilou správu procesů a dokáže lépe využít dostupný výkon ESP32. Navíc ESP-IDF je výchozí a nativní řešení explicitně vytvořené pro mikrokontroléry řady ESP32. Ve svých knihovnách nabízí veškeré funkce, které hardware ESP32 může nabídnout. Program využívající ESPIDF a FreeRTOS se skládá z hlavní funkce *app_main*, která je podobná funkci *setup* z Arduino frameworku a obsahuje volání inicializačních metod dalších programových součástí. FreeRTOS vykonává funkci pomocí úloh (tasks) a tudíž je nutné inicializovat jejich volání. Při vytváření úlohy je nutné definovat její název, prioritu, parametry a velikost přiděleného zásobníku.

Program je rozdělen do dvou hlavních úloh: *task_ble* a *task_pir*. První zmíněná úloha implementuje nekonečnou smyčku pro popis chování Bluetooth LE skeneru. Druhá úloha je určena pro řízení PIR senzoru. Detekce změny stavu na PIR senzoru je řešena pomocí přerušení, avšak funkce pro zaslaní informace o pohybu pomocí MQTT jsou využity v definici této úlohy.

Chování Wi-Fi adaptéru, Bluetooth LE a MQTT je definováno pomocí řídících funkcí typu *event handler*. Každá komponenta má definovaný svůj vlastní *event handler*, který přijímá v parametru funkce *event*, pro který je ve switch výrazu obsaženém ve funkci handleru definovaný případ, který se následně vykoná.

Loggování běhu programu je řešeno pomocí vestavěných funkcí základní knihovny z hlavičkového souboru *esp_log.h* a je použito v celém ESP-IDF frameworku. Systém logování rozlišuje pět tradičních úrovní záznamů: ERROR, WARNING, INFO, DEBUG, VERBOSE. Pro každou z těchto úrovní logu jsou pro vytváření nových záznamů k dispozici pomocná makra. Ve výchozím nastavení se veškeré logy vypisují na sériový UART výstup. Ovšem toto chování lze specifikovat. Výpis logu je definován pomocí funkce typu *vprintf* a tato funkce lze nahradit jinou definicí, pomocí procedury *esp_log_set_vprintf()* přijímající referenci na novou *vprintf* funkci.

4.9.3 Řešení ESPHome

ESPHome nabízí kompletní řešení pro stavbu firmwaru pro ESP mikrokontroléry. Na základě informace ze sekce 4.8 je známo, že ESPHome řeší mnoho dílčích úloh, které jsou implementovány pomocí komponent. Z části, které jsou potřeba pro projekt, je vhodné použít Wi-Fi komponentu starající se o připojení a udržování Wi-Fi spojení. Komunikace pomocí MQTT protokolu, vzdálená správa a loggování je také řešeno pomocí jednotlivých komponent.

V případě detekce BLE zařízení ESPHome také obsahuje komponentu s touto funkcionalitou. Avšak její úlohou je pouze zařízení vyhledávat. O zpracování nalezených dat se starají další varianty senzoru. Komponenta senzoru, který by zasílal celý seznam nalezených zařízení s hodnotami RSSI, není zatím k dispozici a o tuto část je nutné ESPHome rozšířit.

Rozšíření o požadovanou funkcionalitu je primárně určeno pro aktuální vydanou verzi ESPHome, která má označení 1.12.x. Tato verze je implementována a rozdělena ve dvou projektech: esphome a esphome-core. Projekt *esphome* je z velké části napsán v jazyce Python 2.7 a implementuje webový dashboard, funkce pro zpracovávání a generování firmware v jazyce C/C++ a nahrávání programů do vývojových kitů a jejich následné monitorování. Projekt *esphome-core* obsahuje programové části určené pro ESP mikrokontroléry. Tento stav se výrazně mění, jelikož v aktuální vývojové větvi systému probíhá spojení obou částí a toto spojení by mělo následně vyústit ve verzi 1.13. Z toho důvodu jsem se rozhodl nejprve pro vytvoření rozšíření v podobě funkční uživatelské komponenty pro aktuální stabilní verzi a poté, až se fúze ESPHome projektů dokončí, bych chtěl vyvíjenou funkcionalitu začlenit přímo do projektu.

Současné řešení, určené pro verzi 1.12, je vytvořeno jako uživatelská komponenta typu textový senzor. Součástí textové zprávy senzoru jsou data serializovaná do JSON formátu a jejich podoba je znázorněna ve výpisu 4.1.

Funkcionalita BLE trackeru je popsaná v *esphome-core* projektu v části hlavičkového souboru *esp32_ble_tracker.h*. ESPHome je postaveno na Arduino frameworku a pro funkcionalitu Bluetooth adaptéru na ESP32 používá upravenou verzi knihovny ESP32 BLE Arduino od Neilu Kolbana, ořezanou o nepotřebnou funkcionalitu, aby se snížily paměťové nároky na její použití. Přestože je použitá verze knihovny odlehčená, vyžaduje okolo 500 kB paměti.

Všechny komponenty v ESPHome dědí z třídy *Component* a jsou definované pomocí dvou hlavních metod: setup a loop. Plní identickou funkci jako v předešlých metodách. Setup je volán pouze jednou a nastaví vše potřebné v inicializační fázi programu. Až poté se v nekonečné smyčce vykonává loop. Hlavní třídou reprezentující skenování a trackování BLE zařízení je *ESP32BLETracker*. Její metoda setup obsahuje inicializaci potřebných semaforů a BLE adaptéru. Po úspěšném nastavení zařízení nastavuje metoda setup referenci na svůj objekt do globální proměnné.

Součástí metody loop v třídě *ESP32BLETracker* je kontrolování záznamů o přijatých advertisement paketech. Pro každý záznam metoda vyfiltruje duplicitní záznamy z jedné iterace skenování a pro nové záznamy zavolá parsovací funkce určené pro konkrétní ESPHome komponentu. Metody pro publikování výsledků se volají podle typu komponenty buď na konci skenování, nebo ihned během zaznamenání a zpracování informace o BLE zařízení.

Pro rozšíření ESPHome o požadovanou funkcionalitu je tedy nutné upravit konkrétně třídu *ESP32BLETracker*. Ve výsledném řešení je tato úprava součástí nové třídy *BLETextTracker*, která dědí funkce z *ESP32BLETracker* a *TextSensor*. Metoda setup zůstává zachována a volá svého předka BLE trackeru. Loop rozšiřuje funkci o parsování nových zařízení pro text sensor. Parsování je voláno pro list zařízení, který již neobsahuje duplicitní záznamy. Parsování je definováno metodou *parse_found_device_sensors_()* a informace o MAC adrese a síle signálu RSSI ukládá do objektu *map*. Odesílání datového rámce je voláno v *ESP32BLETracker* metodě *start_scan()*. Před odesláním je volaná metoda *get_payload()*, která vrací JSON data z naplněného objektu *map*.

4.10 Nastavení cílové platformy

Většina ESP32 vývojových desek obsahuje flash paměť o velikost 4 MiB. Tato paměť je ve výchozím nastavení rozdělena do několika oddílů. Obsahuje oddíl pro program o velikosti pouhých 1.3 MiB, dále druhý oddíl o stejné velikosti, který slouží pro zálohu programu při aktualizaci pomocí OTA. A kromě několika malých pomocných oddílů je zde i spiffs část pro

statická data. Jelikož používáme knihovny, které velmi rychle zaplní původní 1.3 MiB oddíl, je nutné toto rozložení změnit. Nejjednodušším způsobem, jak rozšířit paměťový prostor, je zrušit druhý OTA oddíl. Tímto krokem je možné zdvojnásobit původní velikost oddílu, ale odstraní se také možnost vzdálené aktualizace programu. Preferovanou možností, jak zajistit vetší prostor pro program a OTA aktualizace, je zmenšení spiffs oddílu. Toto rozložení je jedno z předdefinovaných a nazývá se *min_spiffs*. Toto rozložení nastavuje oba programové oddíly na velikost 1 966 080 B, která již postačí pro použití požadovaných knihoven. Pro oddíl spiffs zbývá 61 440 B.

```
# Name, Type, SubType, Offset, Size, Flags
nvs, data, nvs, 0x9000, 0x5000,
otadata, data, ota, 0xe000, 0x2000,
app0, app, ota_0, 0x10000, 0x1E0000,
app1, app, ota_1, 0x1F0000,0x1E0000,
eeprom, data, 0x99, 0x3F0000,0x1000,
spiffs, data, spiffs, 0x3F1000,0xF000,
```

Výpis 4.6: Příklad nového rozdělení diskových oddílů

4.11 Použití

Pro použití a nahrání programu do vývojového kitu s ESP32 nabízí ESPHome své vlastní nástroje. Jak už bylo popsáno v sekci 4.8, víme, že ESPHome se při generování zdrojového kódu řídí YAML souborem. Pokud chceme použít externí část zdrojového kódu, je nutné doplnit konfiguraci o některé dodatečné příznaky a parametry. Ty se vkládají do YAML objektu *esphome*.

Prvním z nich je parametr *use_custom_code*, který je potřeba nastavit na hodnotu *true*. Pokud tento parametr není nastaven, pak ESPHome není schopný program úspěšně zkompilovat z důvodu nemožnosti zahrnout do programu všechny potřebné programové závislosti. Dále je třeba specifikovat cestu k hlavičkovým souborům nových uživatelských částí. To je možné definovat parametrem *include*.

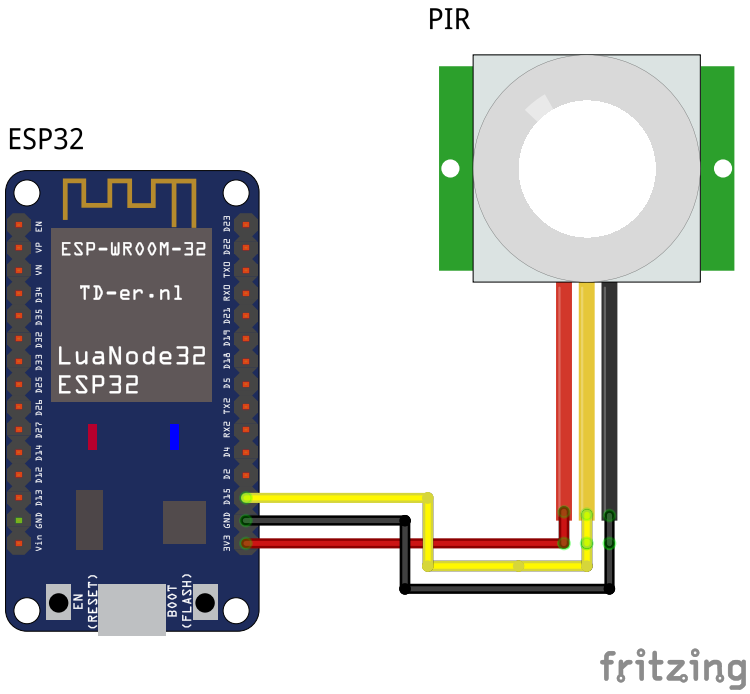
Vytvořené řešení používá Bluetooth knihovnu, která zabírá mnoho místa v paměti ESP32 a v kombinaci s přidanými úpravami pak nelze přeložený program nahrát do přípravku. Z toho důvodu je nutné použít vlastní rozložení paměťových oddílů, ukázané ve výpisu 4.6. Toto rozložení uložené v souboru *min_spiffs.csv* se aplikuje pomocí PlatformIO příznaku *board_build.partitions*. Nastavení specifických PlatformIO parametrů se v rámci ESPHome YAML souboru nastavuje položkou *platformio_options*. Ta obsahuje seznam těchto PlatformIO nastavení.

Pro přidání upraveného BLE skeneru je dále potřeba vložit do YAML souboru zařízení typu *text_sensor* platformy *custom*, viz výpis 4.7. Jelikož uživatelsky vytvořená komponenta nemá definovaný generátor C zdrojového kódu, musí se do konfiguračního souboru přidat také lambda výraz, který tento nedostatek řeší. Výraz obsahuje vytváření instance nové třídy, registraci nové instance u řídícího objektu *App* a následné vracení tohoto objektu z lambda výrazu. Také se před registrací definuje časový interval BLE skenování. Kromě lambda výrazu je součástí *text_sensor* YAML objektu název daného senzoru.

```
text_sensor:
- platform: custom
  lambda: |-
    auto ble_text_sensor = new BLETextTracker();
    ble_text_sensor->set_scan_interval(45);
    App.register_component(ble_text_sensor);
    return {ble_text_sensor};
  text_sensors:
    name: "Obyvaci pokoj BLE"
```

Výpis 4.7: Příklad definice BLE senzoru pomocí YAML souboru

Pro PIR senzor není potřeba přidávat speciálních součástí. Tento senzor je reprezentován binárním senzorem platformy *gpio*. Nejdůležitějším parametrem je výběr GPIO portu ESP32 připojeného na datový vodič PIR senzoru. Mezi další nastavení je název senzoru a jeho typ, kterým je senzor reprezentován v Home Assistantu. Připojení PIR senzoru k GPIO pinu 15 desky Node32 je ukázáno na obrázku 4.3.



Obrázek 4.3: Připojení PIR senzoru k GPIO pinu 15 desky Node32.

Kapitola 5

Modul pro systém Home Assistant

Při budování chytré domácnosti v dnešní době není problém najít komponentu, která by splňovala uživatelovy požadavky. Na trhu lze nalézt různé typy zařízení, od chytrých žárovek a rolety až po ledničky a pračky od mnoha různých výrobců. Každý z nich si však buduje vlastní ekosystém těchto zařízení a využívá individuální rozhraní jejich ovládání. Tyto desítky různých rozhraní částečně spojují univerzální platformy pro domácí automatizaci. Ty nabízí své uživatelské rozhraní a umožňují spojovat a vytvářet různé skriptované akce. Jednou z nich je například Home Assistant, který si podrobněji představíme.

V této části si také vytvoříme návrh rozšíření pro tento systém. Na základě počtu a pojmenování senzorů se pro každý vytvoří entita, která bude přijímat zprávy ze senzoru přes MQTT. Dále pak systém vyhodnotí podle obsahu zpráv a známých BD_ADDR adres, zda se v místnosti nachází člověk a nastaví stav pro příslušnou entitu.

5.1 Domácí automatizační platformy

Popularita chytrých domácností přinesla kromě nových zařízení, také mnoho dalších alternativních automatizačních platforem, které řídí, monitorují a automatizují chytrá zařízení v domácnosti. Část z nich jsou proprietární řešení velkých společností, například Google, Amazon, SAP, Microsoft a další. Tyto firmy nabízejí produkty, které spravují IoT zařízení určené zejména pro firemní sektor, ale také v případě společností Google a Amazon jsou tyto prvky řízení IoT použity pro komerční účely v hlasových asistentech.

Z open source systémů pro domácí automatizaci vhodných pro domácí DIY projekty je kromě Home Assistant, také OpenHUB a nebo Mozilla IoT. OpenHUB je IoT systém pro domácí automatizaci, který může být provozován v mnoha různých prostředích. Podporuje všechny majoritní operační systémy, Docker a také je dostupný obraz operačního systému pro Raspberry PI. Nabízí webové rozhraní a mobilní aplikace pro Android i iOS. Umožňuje také různě modifikovat svůj dashboard. Podobně jako Home Assistant podporuje velké množství doplňků pro rozšíření podpory o zařízení různých výrobců [2]. Společnost Mozilla vytváří v rámci projektu *Things* IoT platformu, ale také se snaží o vytvoření *Web Things API*, které by standardizovalo rozhraní pro komunikaci s IoT zařízeními.

5.2 Home Assistant

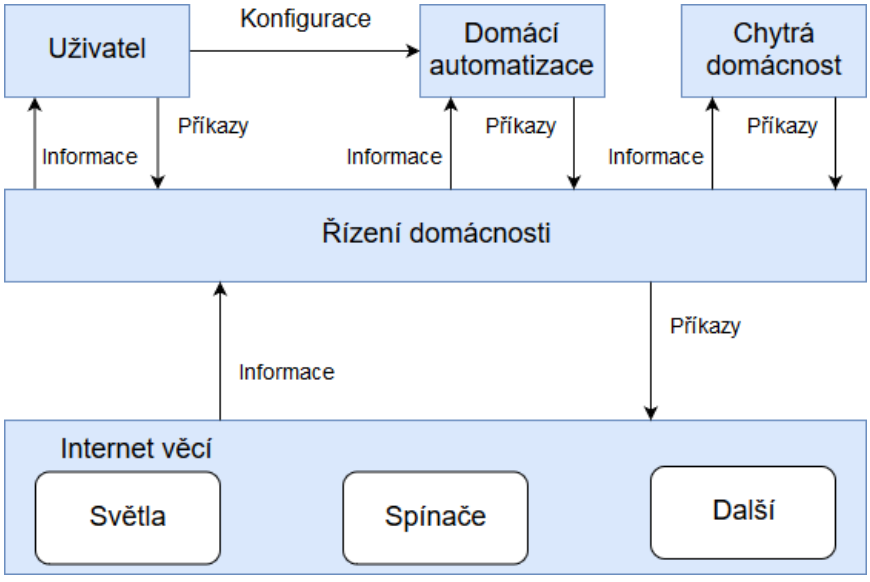
Home Assistant je open source systém automatizace chytrých domů, jehož prioritami je soukromí a lokální řízení. Je navržen jako modulární a jednoduše rozšiřitelný. Podporo-

vaná zařízení jsou popsáná jako platformy a komponenty. O podporu nových inteligentních zařízení se stará celosvětová komunita. Systém je napsán v jazyce Python 3, tudíž je multiplatformní. Kromě podpory všech nejpoužívanějších operačních systémů, je vytvořen také speciální linuxový operační systém založený na kontejnerech Docker s názvem HassOS, který je základem pro *Hass.io*, obraz systému určený pro jednodeskový počítač Raspberry PI 3. Pro tento počítač je také vytvořen upravený obraz linuxové distribuce Raspbian s předinstalovaným Home Assistant s názvem Hassbian [17].

5.2.1 Architektura systému

Na obrázku 5.1 můžete vidět základní architekturu a části Home Assistant systému. Entity, které se systémem pracují, reprezentuje blok *Uživatel*. Část *Internet věcí* značí skupinu všech IoT zařízení. Uživatel má práva konfigurovat chování systému a spouštět příkazy. Systém uživateli zpětně poskytuje informace o stavu všech komponent. Vrstva IoT přijímá a vykonává dílčí příkazy a vrací informace o svém stavu [17]. Zbylé části na obrázku reprezentují systém Home Assistant a jejich funkce je popsána takto:

- *Řízení domácnosti* je část, která se stará o sběr dat a jejich reprezentaci. Dále komunikuje a ovládá zařízení v IoT vrstvě [17].
- *Domácí automatizace* reprezentuje uživatelsky konfigurovatelné spouštěče příkazů IoT zařízení [17].
- *Chytrá domácnost* definuje spouštěče příkazů založené na předchozích událostech [17].



Obrázek 5.1: Architektura systému Home Assistant. Převzato z [17].

Home Assistant je rozšiřitelný pomocí komponent. Jsou psány v jazyce Python, tvoří služby a jejich funkcí je naslouchat spouštěčům událostí a nastavovat stav zařízení. Komponenty se dělí na dva typy. První typ interaguje s doménou IoT zařízení a druhá skupina komponent reaguje na události nezávislé na stavu Home Assistant. Komponenta definuje a komunikuje se zařízením nazvaným *Entity Component*. Tato entita je zodpovědná za

distribuování konfigurace všem platformám dané entity. Předává nové konfigurační vstupy a znalosti, shromažďuje entity pro servisní volání a volitelně spravuje skupiny entit. Platformy jsou součástí komponent. Definují funkci konkrétní skupiny zařízení typu daného entitou komponenty. Sama vytváří *Entity Platform*, která spravuje všechny entity, reprezentující koncová zařízení, zaznamenané v registru entit *Entity Registry*. V případě, že chceme vytvořit platformu podporující například nový typ světel a komponentu definující světla již máme, stačí vytvořit platformu, která bude rozšiřovat entitu světel o novou funkcionalitu [17].

5.2.2 Uživatelské rozhraní

Home Assistant nabízí uživateli monitorovat a řídit svou domácnost, vytvářet automatizace a integrace pomocí progresivní webové aplikace. Design je zaměřen zejména na moderní mobilní použití, podobné jiným mobilním aplikacím. Uživatelské rozhraní je postaveno na webových technologiích a umožňuje aplikaci různých šablon a témat [17]. Architektura webového rozhraní může být rozdělena do 4 částí:

- *Bootstrap* - Jedná se o jednoduchý skript (*core.js*) a první věc, která je načtena na stránce. Tento skript je zodpovědný za autentizaci uživatele a nastavení spojení s backend části systému.
- *App shell* - Tato část je implementována ve skriptu *app.js* a reprezentuje funkci bočního panelu a směrování obsahu okna.
- *Panels* - Panely reprezentují všechny stránky zobrazované v rozhraní Home Assistant. Komponenty mohou registrovat nové panely, které lze zobrazit uživateli.
- *More info dialog* - Jedná se o dialogy, které uživateli zobrazují dodatečné informace o stavech entit a jejich řízení.

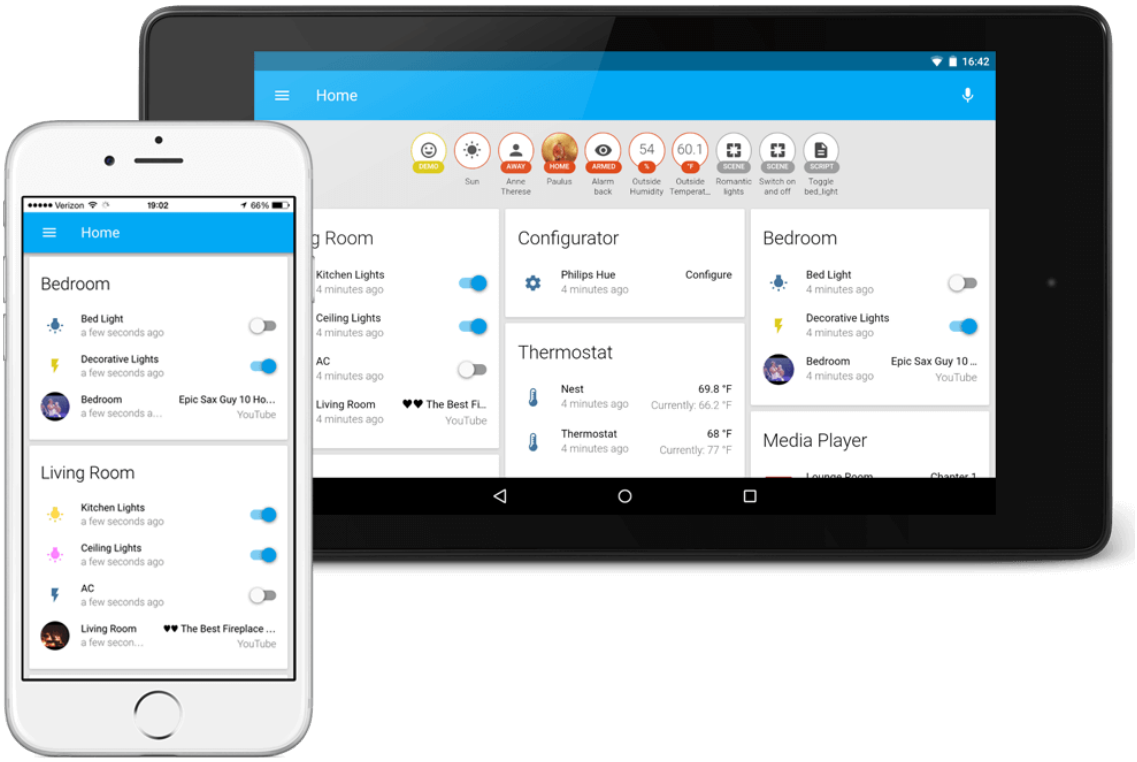
Pro komunikaci mezi GUI a Home Assistantem se využívá Websocket APi a Rest API. Websocket API je rozhraní, které se používá pro streamování informací z Home Assistanetu klientovi. WebSocket API běží na portu 8123. Pro připojení k tomuto rozhraní je potřebný validní přístupový token. REST API poskytuje rozhraní ve výchozím stavu na stejném portu jako WebSocket, tedy na portu 8123. Rest API Home Assistentu přijímá a vrací data pouze v JSON formátu [17].

Home Assistant umí své uživatelské prostředí různě upravovat a rozšiřovat. Entity je možné slučovat a seřazovat do oblastí. Stavové karty lze rozšířit o další informace, například zobrazit obraz z kamer. Pro Home Assistant je možné vytvořit náhled patra s aktivními prvky, které dokáží zobrazovat aktuální stav a nebo ovládat přepínače [17].

5.2.3 Hass.io

Hass.io je obraz operačního systému s integrovaným Home Assistantem. Původně byl postavený na operačním systému ResinOS, ale aktuálně běží na vlastní distribuci HassOS. Oba systémy jsou založené a určené pro spouštění Docker kontejnerů. Systém je optimalizovaný pro běh na zařízeních typu Raspberry Pi. Doporučené zařízení je Raspberry Pi 3B [17].

Oproti běžnému linuxovému systému s nainstalovaným Home Assistantem nabízí jednodušší instalaci, stabilnější běh a aktualizace systému a nebo vytváření záloh a obnovovacích snapshotů. Hassio kromě provozování Home Assistantu umožňuje běh dalších aplikací. Všechny doplňky a rozšíření jsou ve skutečnosti Docker obrazy, které jsou publikovány



Obrázek 5.2: Home Assistant uživatelské rozhraní. Převzato z [16].

také prostřednictvím Docker Hub. Některé užitečné nástroje vhodné k použití u Home Assistantu jsou oficiálně k dispozici přes obchod Hass.io addon store, mezi nimiž je například SSH server, Mosquitto broker pro MQTT a Samba server nebo FTP pro přenos souborů. V základní instalaci je k dispozici také komunitní repozitář, který obsahuje některé experimentální doplňky. Je zde mezi nimi také Portainer pro správu Docker kontejnerů [17].

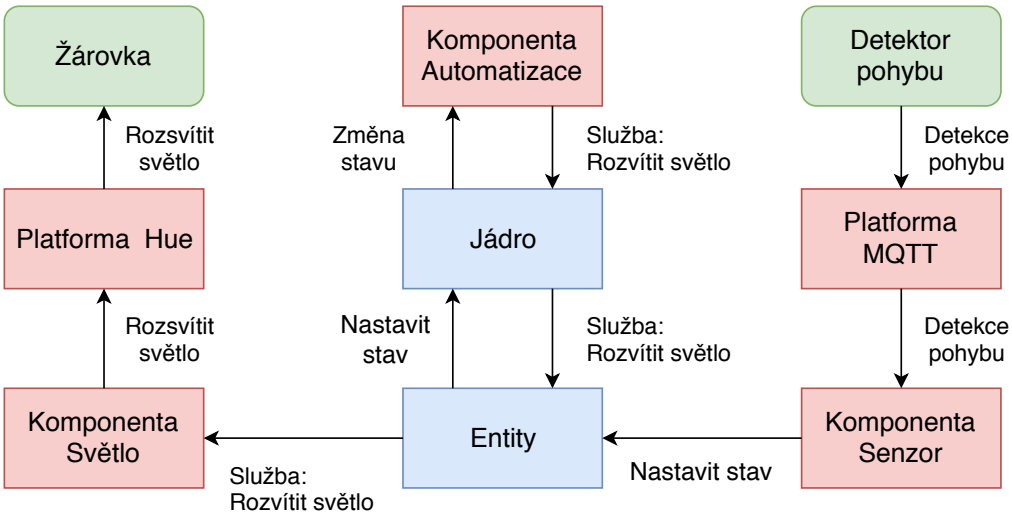
Důležitým doplňkem pro Hass.io je ESPHome. Ten se sice nenachází v oficiálním a ani v komunitním repozitáři, avšak na základě instrukcí na stránkách ESPHome je možné přidat repozitář třetí strany s tímto doplňkem [17].

5.3 Možnosti běhového prostředí

Home Assistant je napsán v Pythonu a dokáže běžet na mnoha různých platformách. Může být provozován na jakémkoli počítači s operačním systémem Windows, Linux a nebo Mac OS. Pro správnou funkčnost systému je nutné, aby Home-Assistant pracoval nepřetržitě. Z toho důvodů je výhodné jej provozovat na různých jednodeskových ARM počítačích, zejména Raspberry PI. Další možností je provoz v prostředí Vagrant a nebo Docker. Právě díky podpoře kontejnerů Docker funguje Home-Assistant na některých NAS serverech společnosti Synology. Nakonec je zde také podpora pro FreeNAS otevřený operační systém pro NAS servery [17].

5.4 Koncepce modulu

Na základě architektury systému Home Assistant víme, že lze rozšiřovat pomocí definic nových komponent a platforem. Ukázka složení Home Assistant systému využívající platformy a komponenty je vyobrazena na obrázku 5.3. Již v základním stavu HA obsahuje velké množství těchto rozšíření. Senzor přítomnosti udává pouze dvě hodnoty. Buď se v místnosti někdo nachází nebo nenachází. Pro takový senzor se nejlépe hodí entita *binary sensor*. Tato entita má pouze dva parametry. První povinný parametr je *boolean* hodnota *is_on*, která udává, zda je senzor ve stavu *true* nebo *false*. Druhý volitelný parametr *device_class* je datového typu *string*, který uvádí třídu senzoru. Mezi předdefinovanými třídami je také *presence* pro definování přítomnosti.



Obrázek 5.3: Diagram HA systému využívající dvě různé platformy a typy komponent. Převzato z [17].

Platforma senzoru přítomnosti bude záviset zejména na definované a správně nakonfigurované komponentě pro MQTT. Pomocí ní systém přijímá data ze senzorů. Pro fungování MQTT protokolu je nutné provozovat také MQTT broker. K tomuto účelu je možné využít otevřenou externí aplikaci *Mosquitto*, avšak tuto funkci dokáže zajistit i samotný Home Assistant, jelikož obsahuje MQTT broker *HBMQTT*. Mezi další závislosti komponenty binárního senzoru může být zařazena také platforma *person*. Ta definuje osoby, které se systémem Home Assistant pracují a může k nim přiřadit jednotlivá zařízení typu *device_tracker*.

Kromě binární informace ohledně přítomnosti člověka v místnosti, může být užitečné také vědět, kde se člověk nachází, respektive jeho BLE zařízení. Jelikož definice platformy není omezena pouze jedním typem entit, můžeme pro platformu definovat další, které bude zprostředkovávat data z jiného pohledu a nabídnou tak uživateli více možností automatizace.

Pokud chceme získat tedy informaci ohledně konkrétní lokace daného zařízení, nabízí se vytvořit vlastní komponentu typu *device_tracker*. Ta může brát informace přímo z binárního senzoru a využít je pro definování nového stavu.

5.5 Implementace platformy

Uživatelské komponenty a platformy jsou implementovány jako celý Home Assistant v jazyce Python 3. První částí vyvíjeného rozšíření je binární senzor reprezentující stav přítomnosti osob/zařízení v místnosti. Toto rozšíření je tvořeno třídou, která dědí z komponenty pro tento typ senzoru.

Při spuštění Home Assistantu se inicializují všechna zařízení uvedena v konfiguračním souboru. Pro tuto činnost je potřeba implementovat funkci *setup_platform()*. Ta vytváří nové instance dané třídy a registruje je pomocí funkce *add_device()*. Konkrétní nastavení pro dané zařízení inicializační metoda získá z parametrů uvedených v konfiguraci. Kontrola nastavovacích parametrů v YAML souboru je prováděna pomocí validační knihovny *voluptuous*. Příklad použití knihovny pro kontrolu vstupu je ukázán ve výpisu 5.1. Knihovna u každého argumentu kontroluje správnost názvu, nutnost jeho použití a datový typ hodnoty.

```
import voluptuous as vol

PLATFORM_SCHEMA = PLATFORM_SCHEMA.extend({
  vol.Required(CONF_NAME): cv.string,
  vol.Required(CONF_TOPIC): cv.string,
  vol.Optional(CONF_MODE,default=DEFAULT_MODE): PRESENCE_MODE_SCHEMA,
  vol.Optional(CONF_RSSI,default=DEFAULT_RSSI_LIMIT_VALUE):negative_int,
  vol.Optional(CONF_PIR,default=DEFAULT_PIR): cv.boolean
})
```

Výpis 5.1: Použití voluptuous pro kontrolu konfigurace

Třída *BinarySensorDevice*, která je základem pro binární senzor, definuje tři své vlastní property proměnné: is_on, state a device_class. Poslední zmíněná proměnná definuje, o jaké zařízení se jedná. Stav senzoru bude vždy pouze hodnota True nebo False, avšak tato informace se dá uživateli různě reprezentovat a tuto reprezentaci určuje právě device_class. Ze seznamu možností se nejvíce blíží účelu typ presence, přestože je tato hodnota v uživatelském rozhraní definována jako Doma/Pryč, což není nejvhodnější reprezentace.

V konstruktoru binárního senzoru třídy nazvané *PresenceSensor* je kromě inicializace proměnných také registrace k MQTT odběru příslušných topiců a callback funkcí obsluhující přijetí MQTT zprávy. Tato funkce pak dále volá metodu *fetch_state()*, která vrací novou hodnotu stavu na základě přijatých dat. Metoda *fetch_state* obsahuje přepínač, který na základě nastaveného atributu *mode* volá odlišné funkce. Senzor v režimu *all* pouze filtruje vstupní data a nechává záznamy o zařízeních s větší sílou signálu, než je hodnota proměnné *rssi*. Režim *person_only* navíc poté vybere seznam nakonfigurovaných osob a jejich zařízení a najde shody s daty ze senzoru. Kromě stavu True/False třída nastavuje do entitního atributu *found_devices* seznam zařízení a atribut *present_people* naplní seznamem jmen osob potencionálně přítomných v místnosti.

Další část platformy pro detekci přítomnosti BLE zařízení implementuje sledování zařízení typu *device_tracker*. To uživateli říká, zda se zařízení nachází doma, avšak v tomto případě bude nejdůležitější informace o tom, kde se nachází. Oproti platformy binárního senzoru se inicializace sledovaného zařízení trochu liší. Rozdílem je, že se volá funkce *setup_scanner()*, která kromě typických parametrů ve formě hlavního objektu *hass* a konfigurační mapy *config* předává také referenci na metodu *see*. Ta během volání říká, že zařízení

definované v parametrech funkce bylo viděno a má být po určitou dobu označeno jako přítomné. Výchozí časový limit pro přítomnost zařízení je tři minuty.

Součástí inicializační metody je vytvoření pole instancí třídy *Device* s parametry z YAML konfiguračního souboru a callback funkci pro aktualizaci stavů všech sledovaných zařízení. Tato callback funkce je volaná periodicky po uplynutí určitého času.

Třída *Device* reprezentuje jedno zařízení typu *device_tracker*. Obsahuje metodu *update*, která přijímá *see* funkci. Třída *Device* obsahuje další pomocné metody, které slouží pro rozhodnutí, zda je zařízení v budově viditelné a ve které místnosti se aktuálně nachází. Používají se k tomuto účelu informace z binárního senzoru této platformy, konkrétně z atributu *found_devices*. Pokud se sledovaný prvek nachází v některé z nakonfigurovaných místností s detektorem BLE zařízení, pak se zavolá funkce *see* s parametry zařízení a atributem *room* obsahující název místnosti.

5.6 Použití

Pro použití a spuštění nově vytvořených částí, je nutné je nejprve umístit do adresáře *custom_components* v konfiguračním adresáři Home Assistantu. Správně umístěný program pak dokáže Home Assistant detekovat a spustit.

Adresářová struktura rozšiřujících komponent a platforem má přesně definovaný formát. Tato struktura se začala hromadně uplatňovat pro všechny komponenty v Home Assistantu od verze 0.87. Pravidla pro organizaci souborů spočívají v tom, že každá platforma musí být umístěna ve vlastním adresáři s inicializačním souborem *___init___.py*. Všechna další rozšíření platformy jsou umístěna v tomto adresáři a nazvaná podle typu rozšíření. Pro příklad: pokud máme platformu, která rozšiřuje třídu *BinarySensosrDevice* umístěnou v platformě *binary_sensor*, pak tato děděná třída by měla být uložena v souboru *binary_sensor.py*. To samé platí i pro *device_tracker*.

Aby byla komponenta použita, musí se správně nakonfigurovat ve správném YAML souboru. Základní konfigurační soubor Home Assistantu se nazývá *configuration.yaml*. Na základě závislostí je pro správnou funkčnost potřeba mít funkční MQTT broker a Home Assistant nakonfigurovaný pro komunikaci s ním.

Jelikož si binární senzor a tracker zařízení předává některé informace prostřednictvím datové proměnné objektu hass a inicializace tohoto paměťového místa probíhá v komponentě platformy, musí se tato komponenta použít vložením *ble_presence:* do konfigurace.

```
ble_presence:
binary_sensor:
  - platform: ble_presence
    name: Obyvaci pokoj
    topic: obyvaci_pokoj
    pir: true
    mode: person_only
device_tracker:
  - platform: ble_presence
    devices:
      miband: 00:00:00:00:00:00
      amazfit: 11:11:11:11:11:22
```

Výpis 5.2: Konfigurace platformy senzoru pro Home Assistant.

Dále pro vytvoření binárního senzoru pro reprezentaci přítomnosti osob/zařízení v místnosti domu je potřeba vložit záznam typu *binary_sensor* s atributem *platform* nastaveným na hodnotu *ble_presence*. Tento senzor má také několik dalších parametrů. Mezi povinné parametry patří *name*, který určuje jméno entity a parametr *topic*, která definuje základní MQTT topic pro daný senzor. Topic musí korespondovat s ESPHome názvem senzoru a z něj se dále určí konkrétní řetězec pro přijímání zpráv ohledně BLE zařízení a pohybu z PIR senzoru.

Prvním volitelným atributem je položka s názvem *mode*. Ten určuje v jaké režimu má senzor pracovat a jakým způsobem se detekuje přítomnost člověka. V současné době jsou k dispozici dva režimy. První *all* určí přítomnost ze všech nalezených zařízení a druhý *person_only* se zabývá pouze zařízeními *device_tracker* spárovanými s konkrétními osobami. Výchozí hodnota je *all*. Parametr *rssi* určuje mezní sílu signálu. Pokud parametr není zadán, hraniční hodnota RSSI je -90 dBm. Volitelný parametr *pir* určuje přítomnost PIR senzoru.

Platforma obsahuje také druhý typ entit s názvem *device_tracker*. Entity tohoto typu nejsou navrženy, aby fungovaly samostatně a vyžadují ke své činnosti alespoň jeden binární senzor této platformy. Vytvoření tohoto zařízení je prováděno vložením YAML objektu *device_tracker:* s platformou *ble_presence*. Dalším parametrem je atribut *devices*, který obsahuje seznam zařízení ve formě <název>:<MAC adresa>. Příklad kompletní konfigurace je ukázán ve výpisu 5.2.

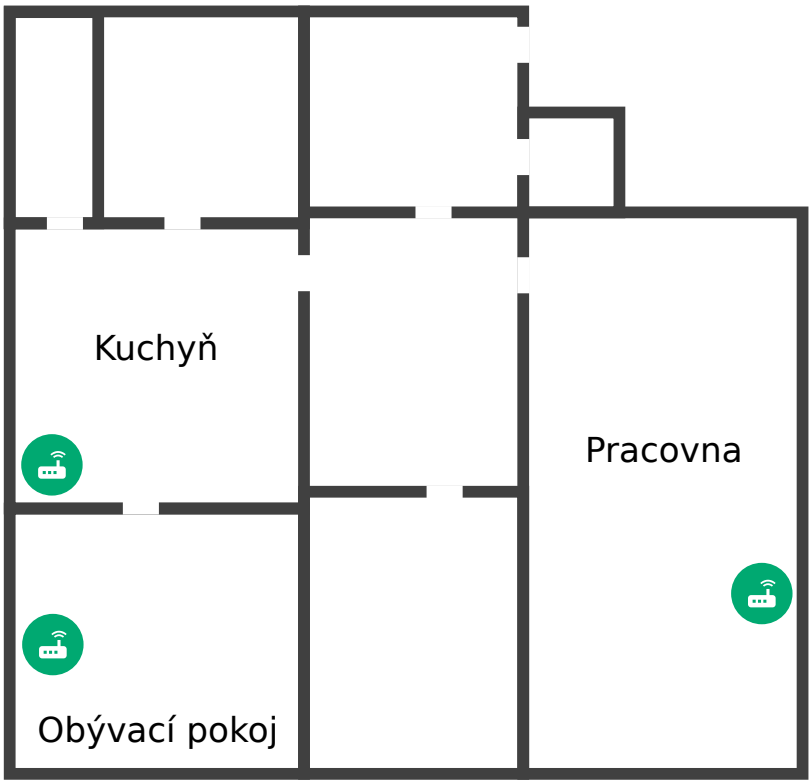
Kapitola 6

Měření a testování

Jelikož je projekt zaměřen na vytvoření řešení pro chytré domácnosti, bude i reálné testování probíhat v domácnosti. Nejlepší možné fungování výsledného produktu je založeno zejména na konfiguraci známých zařízení, se kterými systém může pracovat. Tudíž pro použití ve veřejných prostorech, kde se vyskytují neznání lidé, toto řešení není vhodné.

6.1 Testovací prostředí

Primární testovací prostředí bylo zasazeno do přízemí staršího rodinného domu vyobrazeného půdorysem tohoto patra na obrázku 6.1. Tato budova nabízí pro systém detekce



Obrázek 6.1: Půdorys domu použitého pro testování. Zelené značky označují senzory.

přítomnosti osob na základě BLE zařízení ideální prostředí. Nabízí totiž silné zdi, které napomáhají izolaci signálů v dané místnosti.

Pro testování byly umístěny po domě tři senzory. Každý z nich se nacházel v jedné místnosti. Jednalo se o místnosti, které jsou nejčastěji dlouhodobě obývané a patří mezi ně: kuchyň, obývací pokoj a velká pracovna.

Konkrétní umístění senzoru bylo v každé místnosti trochu specifické. Například senzor v obývacím pokoji se nacházel v těsné blízkosti s kuchyňským senzorem, což mohlo způsobit nežádoucí ovlivňování detekce. Dále pak tento senzor se během testování nacházel nízko od země a byl obklopen různou elektronikou a rovněž Wi-Fi routerem, který také způsobuje další rušení.

Druhý kuchyňský senzor sdílel problém malé vzdálenosti se sousedním senzorem. To může způsobit špatnou reprezentaci informací přijatých na Home Assistantu. V jiných ohledech se senzor liší. První odlišností byla skutečnost, že se senzor nacházel v částečně průchozí místnosti a takové prostředí není příliš vhodné pro detekce. Další důležitým atributem mohl být fakt, že se kuchyňský senzor se během testování nacházel velmi vysoko u stropu bez žádných výrazných překážek.

Poslední senzor byl umístěn v nejrozlehlejší místnosti domu, středně vysoko od země a kolem něj se nacházely menší překážky. Senzor nebyl přímo viditelný ze všech koutů místnosti. Toto prostředí mělo zjistit, jak spolehlivě řešení funguje ve větších místnostech s nepřímou viditelností.

6.2 Testování

Testování probíhalo v prostorech, které jsou popsány v sekci 6.1. Účel testování spočíval zejména ve sledování chování systému v běžném prostředí a reakce na přechody lidí mezi místnostmi. Dalším důležitým poznatkem mělo být zjištění, jakým způsobem se chovají různá BLE zařízení a které z těchto přístrojů jsou vhodná pro detekci. Senzory použité v testovacím provozu byly sestaveny z vývojových desek *Doit ESP32 Devkit v1* a PIR senzorů typu *AM312*.

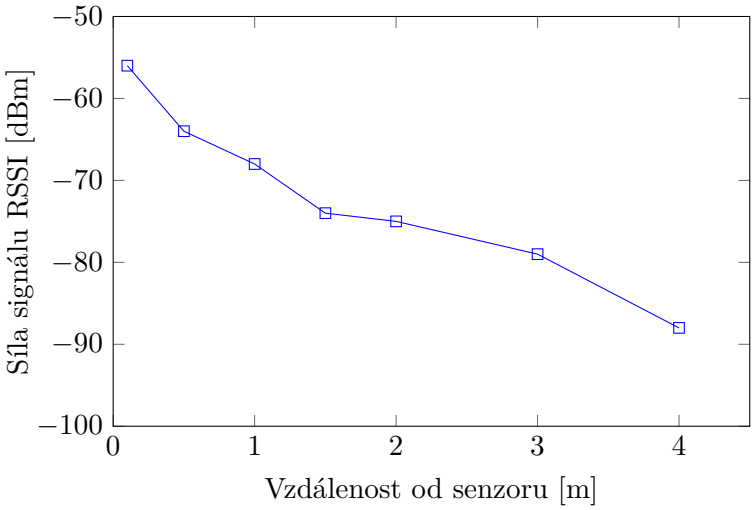
V rámci testování systému figurovali v systému tři lidé se svými BLE zařízeními z kategorie nositelné elektroniky. Byly použity chytré náramky Xiaomi Mi Band 2 a Mi Band 3 a chytré hodinky také od společnosti Xiaomi, konkrétně Amazfit Bip. Všechna tato zařízení umožňují zapnout se ve viditelném režimu a je možné tato zařízení sledovat pomocí ESP32 BLE skeneru. Testovací provoz senzoru a jeho platformy pro Home Assistant probíhal několik dní, během kterých jsem postupně ladil funkcionalitu, aby bylo dosaženo nejlepšího možného uživatelského zážitku.

6.3 Výsledky a hodnocení

Testování přineslo několik zajímavých poznatků a některé z nich pomohly systém vylepšit. Prokázalo se také několik chyb a nedostatků, které však souvisí s celkovou koncepcí a metodou detekce. Na základě této zkoušky bylo možné zhodnotit reálnou použitelnost vytvořeného systému.

První poznatky vyplývající z testování se týkají reálného chování BLE na ESP32 a vlastností přijímaných signálů. V reálném použití se hodnota síly signálu změřeného na vývojovém kitu pohybuje od -50 dBm a méně. Hodnota RSSI kolem -50 dBm znamená, že zdrojové zařízení je v těsné blízkosti senzoru. Pokud se nacházíme v běžně velké bytové místnosti

několik jednotek metrů od senzoru bez významné překážky, pak tato hodnota se pohybuje okolo -70 dBm, viz obrázek 6.2. Pokud mezi senzor a zařízení postavíme zeď, síla signálu se dále sníží o cca 10 - 20 dB. Přesnou hranici mezi přítomností a nepřítomností člověka v místnosti není možné přesně definovat. Ta se pohybuje od -90 až do -80 dBm a velmi závisí na bytové struktuře, síle stěn, ale také i na umístění senzoru.



Obrázek 6.2: Graf závislosti vzdálenosti na síle signálu RSSI.

Během reálného zkoušení se projevil určitý rádiový nedostatek BLE při použití této technologie pro určování vzdálenosti na základě síly signálu. Problém spočívá v tom, že technologie jako Wi-Fi nebo Bluetooth využívají signál v úzkém pásmu s dlouhými vlnami a pro takový signál je složité určit hodnotu amplitudy v určitý čas. To je navíc ještě ovlivněno překážkami v cestě, které signál ruší, zpožďují a degradují. Proto je měření síly signálu velice nestabilní a definovat vhodné mezní hodnoty je obtížné [6].

Rychlost skenování a přechodu mezi stavy je také další problém. Senzor PIR sice zajišťuje okamžitou aktivaci při zaznamenání pohybu, avšak v původní verzi senzoru se seznam nalezených zařízení posílal vždy až na konci skenování a posílal se vždy pouze první záznam od každého zařízení. Ten však již nemusel být aktuální a senzor mohl během této doby již získat další aktualizaci. To bylo podnětem pro změnu způsobu hledání a zasílání nalezených zařízení. Aktuálně senzor zašle potřebné informace ihned po přijetí advertisement zprávy a modul běžící v Home Assistantu přijatý záznam obohatí o časovou známku a uloží si ho. Záznamy starší než určitý časový interval poté automaticky odstraňuje. Tato změna pomohla k výrazně rychlejším aktualizacím záznamů.

Při celkovém zhodnocení použitelnosti vytvořeného řešení je potřeba vzít v úvahu všechny výše zmíněné nedostatky, ale také předpokládané způsoby použití. Díky použité metodě a vybraných technologií je řešení funkční a levné, avšak také občas nepřesné. Pro nejlepší možnou funkčnost je potřeba vhodně nastavit několik časových intervalů, například interval skenování senzorů v ESP32 a nebo interval mazání starých záznamů v HA. Například: čím je čas pro mazání záznamů delší, tím se stav entity více ustálí na jedné hodnotě, avšak na druhou stranu systém pomaleji reaguje na nepřítomnost člověka. Prostřednictvím těchto intervalů je potřeba nalézt vhodný kompromis. Určitou korekci lze dosáhnout také při vytváření automatizačních pravidel podmíněných nejen stavem přítomnosti, ale také dalšími podmínkami, například časovým limitem.

Kapitola 7

Závěr

Tato práce se zabývala vytvořením platformy senzoru pro detekci osob v místnosti určenou pro automatizaci v chytrých domácnostech. Hlavní myšlenkou projektu bylo nalézt jednoduché řešení tohoto problému využívající ke své činnosti běžná a finančně dostupná zařízení.

Průzkum různých metod detekce běžných nositelných zařízení ukázal, že při použití v obytných prostorech domu dokáže nabídnout požadovanou přesnost pouze detekce blízkých BLE zařízení. Při analýze možných metod detekce osob se zjistilo, že přesnost detekce je možné zvýšit pomocí PIR senzoru, přestože tento senzor dokáže detekovat pouze pohyb a nikoli přítomnost člověka. Na základě provedené analýzy a definovaných požadavků pro senzor, bylo možné vybrat vhodnou platformu pro základ senzoru. Důležité parametry ve výběru tvořily: podpora potřebných standardů, nízká spotřeba energie a dostupná pořizovací cena. Po srovnání různých mikrokontrolérů a vývojových platforem jsem se rozhodl pro použití ESP32. Navržený senzor, který periodicky skenuje okolí a hledá blízká zařízení a posílá zprávy pomocí Wi-Fi MQTT protokolem, jsem implementoval v jazyce C++.

Podpora domácí automatizace je zajištěna pomocí rozšíření v domácím automatizačním systému Home Assistant. Na základě návrhu se podařily implementovat všechny doplňující komponenty do automatizačního systému Home Assistant. Systém dokáže pracovat s informacemi o známých lidech a jejich zařízeních vhodných k detekci. Reálné použití senzoru v domácnosti je velmi otevřené a závisí pouze na fantazii uživatele. Může například rozsvěcovat světla, měnit teplotu, zapínat různá zařízení a mnoho dalších činností.

Zadání a stanovené cíle bakalářské práce se podařilo splnit. Vytvořené řešení funguje podle předpokladu a je možné ho v praxi využít, avšak s některými výhradami. Senzor byl testován zejména v místnostech se silnými zdmi, které znatelně izolují Bluetooth signál. V jiných prostředích se senzor může chovat jinak, avšak systém umožňuje nastavovat různé mezní hodnoty pro kterékoli čidlo a do jisté míry přizpůsobit své chování danému místu.

Cenu výsledného produktu se podařilo uchovat nízko. Automatizační platforma je otevřený nástroj, který může běžet na kterémkoli počítači v domácnosti, ale nejvhodnější variantou je použití jednodeskového počítače Raspberry PI. Hlavní část senzoru mikrokontrolér ESP32 je možné pořídit za cenu cca $5 a cena PIR senzoru se pohybuje okolo $1.

Budoucí rozšíření systému pro detekci osob v místnosti vidím zejména ve zlepšení a zdokonalení uživatelského rozhraní pro jednodušší nastavování a obsluhu senzorů. Bylo by vhodné rozšířit podporu senzoru i pro jiné domácí automatizační platformy.

Slovník

Android mobilní operační systém vyvíjený společnosti Google. 7, 30

ARM architektura procesoru typu RISC. 16

baseband základní pásmo. 6

beacon maják. 7, 15, 19

BR/EDR Basic Rate/Enhanced Data Rate, označení základního typu Bluetooth. 6, 7, 16

Docker otevřený software pro virtualizaci kontejnerů pro běh aplikací. 30

embedded vestavěný. 20

Flow Control řízení toku. 6

iOS konkurenční mobilní OS společnosti Apple. 7, 30

Link layer linková vrstva. 7

MAC fyzická adresa síťového zařízení. 5, 9, 24, 27

nmap Network MAPer, program pro skenování lokální sítě. 9

PHY layer fyzická vrstva. 7

pyroelektrický vlastnost materiálu, který přeměňuje teplo, kterému je vystaven, na elektrickou energii. 8

Python interpretovaný programovací jazyk. 20, 30, 31, 34

retransmission opakování přenosu. 6

Zkratky

ADC Analog-to-Digital Converter. 16

AES Advanced Encryption Standard. 16

AMQP Advanced Message Queuing Protocol. 12, 13

AP Access Point. 9, 18, 22, 26

ATT Attribute Protocol. 7

BLE Bluetooth Low Energy. 6, 7, 15, 16, 19, 21, 23, 24, 25, 26, 27, 28, 34, 35, 36, 38, 39, 40, 41

CAN Controller Area Network. 16

CoAP Constrained Application Protocol. 11, 12, 13

CRC Cyclic redundancy check. 6

DAC Digital-to-Analog Converter. 16

ECC Elliptic curve cryptography. 16

FHSS Frequency Hopping Spread Spectrum. 5, 6, 18

GNSS Global Navigation Satellite System. 9

GPIO General-purpose input/output. 15, 16, 18

GPS Global Positioning System. 9

HCI Host Controller Intreface. 6

HTTP Hyper Text Transport Protocol. 12, 13

I²C Inter-Integrated Circuit. 16

I²S Integrated Inter-IC Sound. 16

IoT Internet of Things. 3, 10, 11, 15, 16, 30, 31

ISM Industrial, Scientific and Medical. 5, 6

L2CAP Logical link control and adaptation protocol. 6, 7

LAP Lower Address Part. 6

LE Low Energy. 5, 23

LMP Link Manager protocol. 6

M2M Machine-to-Machine. 11, 12, 20

MCU mikrokontrolér. 15, 16, 17, 19, 20, 21, 22, 23

MQTT Message Queue Telemetry Transport. 11, 12, 13, 20, 21, 23, 24, 25, 26, 30, 34, 35, 36, 41

NAP Non-significant Address Part. 5

OTA Over The Air. 21, 23

PIR Passive Infrared sensor. 8, 15, 16, 26, 29, 36, 37, 39, 40, 41

PWM Pulse Width Modulation. 16

QoS Quality of Service. 11, 12, 13

RMII Reduced Media-Independent Interface. 16

RNG Random number generator. 16

RSA Rivest, Shamir, Adleman. 16

RTC Real-time clock. 17

SDP Service discovery protocol. 6

SHA-2 Secure Hash Algorithm 2. 16

SIG The Bluetooth Special Interest Group. 5, 6

SM Security Manager. 7

SoC System on Chip. 15, 16

SPI Serial Peripheral Interface. 16

UAP Upper Address Part. 6

UART Universal Asynchronous Receiver/Transmitter. 16, 19

ULP Ultra Low Power. 15, 16, 18

URI Universal Resource Identifier. 12

USB Universal Serial Bus. 16, 19

Literatura

[1] Afaqui, M. S.; Garcia-Villegas, E.; Lopez-Aguilera, E.: *IEEE 802.11ax: Challenges and Requirements for Future High Efficiency WiFi. IEEE Wireless Communications*, ročník 24, č. 3, June 2017: s. 130–137, ISSN 1536-1284, doi:10.1109/MWC.2016.1600089WC, [Online; navštíveno 11.1.2019].
URL https://ieeexplore.ieee.org/document/7792393

[2] Baker, J.: *6 open source home automation tools*. 12 2017, [Online; navštíveno 28.1.2019].
URL https://opensource.com/tools/home-automation

[3] Bisdikian, C.: *An overview of the Bluetooth wireless technology. IEEE Communications Magazine*, ročník 39, č. 12, Dec 2001: s. 86–94, ISSN 0163-6804, doi:10.1109/35.968817, [Online; navštíveno 16.1.2019].

[4] Bluetooth Special Interest Group: *Bluetooth Specification Version 5.0*. 12 2016, [Online; navštíveno 27.11.2018].
URL
https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=421043

[5] Blum, J.: *Exploring Arduino: Tools and Techniques for Engineering Wizardry*. USA: John Wiley & Sons, Inc., první vydání, 2013, ISBN 1118549368, 9781118549360.

[6] Ciaran Connell: *What's The Difference Between Measuring Location By UWB, Wi-Fi, and Bluetooth?* 2 2015, [Online; navštíveno 10.5.2019].
URL https://www.electronicdesign.com/communications/what-s-difference-between-measuring-location-uwb-wi-fi-and-bluetooth

[7] Dočekal, M.: *Správa linuxového serveru: Úvod do skenování sítí pomocí Nmap*. 3 2012, [Online; navštíveno 11.1.2019].
URL https://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-uvod-do-skenovani-siti-pomoci-nmap

[8] Eller, J.: *IoT Wireless Technology: Enabling Radio Frequencies*. [Online; navštíveno 20.4.2019].
URL https://bridgera.com/wireless-technology-for-iot/

[9] ESP32net: *Development*. [Online; navštíveno 16.12.2018].
URL http://esp32.net/

[10] ESPHome: *ESPHome*. [Online; navštíveno 12.5.2019].
URL https://esphome.io/

[11] Espressif Systems: *ESP32 Series Technical Reference Manual*. 12 2018, [Online; navštíveno 15.4.2019].
URL https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[12] Espressif Systems: *ESP32 Series Datasheet*. 4 2019, [Online; navštíveno 15.4.2019].
URL https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[13] Gill, K.; Yang, S.; Yao, F.; aj.: *A zigbee-based home automation system. IEEE Transactions on Consumer Electronics*, ročník 55, č. 2, May 2009: s. 422–430, ISSN 0098-3063, doi:10.1109/TCE.2009.5174403.

[14] Gomez, C.; Paradells, J.: *Wireless home automation networks: A survey of architectures and technologies. IEEE Communications Magazine*, ročník 48, č. 6, June 2010: s. 92–101, ISSN 0163-6804, doi:10.1109/MCOM.2010.5473869.

[15] Gubbi, J.; Buyya, R.; Marusic, S.; aj.: *Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems*, ročník 29, č. 7, 2013: s. 1645 – 1660, ISSN 0167-739X, doi:https://doi.org/10.1016/j.future.2013.01.010.
URL http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[16] Home-Assistant: *Frontend of Home Assistant*. [Online; navštíveno 1.5.2019].
URL https://www.home-assistant.io/docs/frontend/

[17] Home-Assistant: *Home Assistant Developer documentation*. [Online; navštíveno 11.1.2019].
URL https://developers.home-assistant.io/

[18] Kolban, N.: *Kolban's book on ESP32*. Leanpub, 2018.
URL https://leanpub.com/kolban-ESP32

[19] Mustapha, B.; Zayegh, A.; Begg, R. K.: *Ultrasonic and Infrared Sensors Performance in a Wireless Obstacle Detection System*. In *2013 1st International Conference on Artificial Intelligence, Modelling and Simulation*, Dec 2013, s. 487–492, doi:10.1109/AIMS.2013.89.

[20] Naik, N.: *Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP*. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, Oct 2017, s. 1–7, doi:10.1109/SysEng.2017.8088251.

[21] Nccgroup: *Bluetooth Defining NAP + UAP + LAP*. 2012, [Online; navštíveno 28.11.2018].
URL https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2012/february/bluetooth-defining-nap-uap-lap/

[22] Oasis: *MQTT Version 3.1.1*. [Online; navštíveno 11.1.2019].
URL http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[23] Townsend, K. and Davidson, R. and Cufí, C.: *Getting Started with Bluetooth Low Energy*. O'Reilly, 2014, ISBN 9781491949511, [Online; navštíveno 16.1.2019].

URL https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch01.html

[24] Tsai, C. F.; Young, M. S.: *Pyroelectric infrared sensor-based thermometer for monitoring indoor objects*. *Review of Scientific Instruments*, ročník 74, č. 12, 2003: s. 5267–5273, doi:10.1063/1.1626005, https://doi.org/10.1063/1.1626005.
URL https://doi.org/10.1063/1.1626005

[25] Zhao, K.; Ge, L.: *A Survey on the Internet of Things Security*. In *2013 Ninth International Conference on Computational Intelligence and Security*, Dec 2013, s. 663–667, doi:10.1109/CIS.2013.145.

[26] Štroner, M.: *Globální navigační satelitní systémy (GNSS)*. ČVUT – Fakulta stavební, Praha, [Online; navštíveno 16.12.2018].
URL http://k154.fsv.cvut.cz/vyuka/geodezie_geoinformatika/vy1/OBS/GNSS_obs.pdf

Příloha A

Obsah přiloženého paměťového média

- text/ — Bakalářská práce.
  - src/ — LaTeX zdrojové kódy tohoto dokumentu.
  - xprude02_DetekcePritomnostiOsob.pdf — PDF soubor s bakalářskou prací.
- esp32/ — Implementační část ESP32.
  - src/ — Zdrojové soubory
  - example/ — Příklad konfigurace
- homeassistant/ — Implementační část Home Assistant.
  - src/ — Zdrojové soubory
  - example/ — Příklad konfigurace
- README — Soubor s popisem obsahu CD