Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Senzory domácí automatizace

Plzeň 2021 Jiří Mraček

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jiří MRAČEK**
Osobní číslo:	**A16B0086P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Senzory domácí automatizace**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s platformami pro domácí automatizaci, jejich omezeními a možnostmi připojení senzorů. Zvolte jednu platformu a výběr zdůvodněte.
2. Prozkoumejte vhodné moduly a realizujte alespoň dva spolupracující, typově odlišné, sensory a připojte je k vybrané platformě. Důraz je kladen na možnou rozšiřitelnost navržené realizace.
3. Realizujte možnost ovládání a sledování přes mobilní telefon.
4. Zdokumentujte systém a jeho rozhraní. Funkčnost systému ověřte reprezentativní množinou testů. Definujte případná omezení.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Tomáš Mainzer, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **5. října 2020**
Termín odevzdání bakalářské práce: **6. května 2021**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 26. října 2020

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 4. května 2021

Jiří Mraček

Abstract

The purpose of this Bachelor Thesis is to examine the possibilities of home automation and to design sensors. At first, we review known solutions for home automation, especially automation platforms available under `Open Source license (OpenHAB, Home Assistant and Domoticz)`. This is followed by a choice of one of the platforms. As a next step, we study the possibilities of the design and realization of sensors based on `ESP32-CAM and ESP8266`. To enable future extensibility, we also provide a library for Wi-Fi sensor, which is already used in our implementation of some sensors, e.g. thermometer or relay module. We have paid attention to allow users to control and monitor the household via mobile phone or voice assistant.

Abstrakt

Tato bakalářská práce se zabývá možnostmi domácí automatizace a návrhem senzorů. Nejdříve jsou prozkoumána dostupná řešení domácí automatizace, především automatizační platformy šířené pod licencí `Open Source (OpenHAB, Home Assistant a Domoticz)`. Následuje výběr jedné automatizační platformy.

Dále jsou prostudovány možnosti návrhu a realizace senzorů založených na `ESP32-CAM a ESP8266`. Z důvodu možné rozšiřitelnosti je vytvořena knihovna pro Wi-Fi senzor, která je následně použita při implementaci konkrétních senzorů např. senzoru teploty či relé modulu.

Je zde kladen důraz na ovládání a sledování stavu domácnosti z mobilního telefonu, případně pomocí hlasového asistenta.

Obsah

1 Úvod

Svět, ve kterém dnes žijeme, nás obklopuje mnoha moderními technologiemi, které nám dokáží usnadnit každodenní život. Co se týče plnění každodenních rutinních činností jako je větrání, nastavení teploty vytápění v závislosti na venkovní teplotě či spínání venkovního osvětlení můžeme využít možnosti domácí automatizace. Použití automatizačních systémů již nachází velké uplatnění v průmyslu.

V poslední době zažívá obrovský nárůst popularity internet věcí (Internet Of Things - IOT), jedná se o propojení několika zařízení, které spolu komunikují a následně vyhodnocují přijatá data, na jejichž základě jsou spínány různé prvky domácnosti jako je např. tepelné čerpadlo nebo venkovní osvětlení. Díky rostoucí dostupnosti vývojových modulů a čidel je jejich integrace do domácnosti stále více dostupnější.

V této práci se budu zabývat výběrem vhodné platformy pro domácí automatizaci. Zaměřím se na platformy, které jsou volně šiřitelné pod licencí Open Source, následně vyberu jednu platformu, kterou detailně prostuduji a otestuji. Dále se budu zabývat návrhem senzorů, které následně připojím k vybrané platformě. Při výběru platformy se zaměřím na možnosti ovládání, sledování stavu jednotlivých senzorů a zařízení pomocí mobilní aplikace, protože většina populace má mobilní telefon neustále ve své blízkosti, ovládání chytré domácnosti pouze pomocí počítače by bylo značně nepraktické.

Výsledkem práce bude návrh a realizace vstupního systému u zahradní branky, kde je kladen důraz na možnou rozšiřitelnost. Z důvodu snazší implementace jednotlivých senzorů (domovní zvonek, senzor teploty, kamerový modul, relé modul, displej modul a RFID modul) bude navržena knihovna pro obecný senzor, která bude následně použita při implementaci všech výše zmíněných modulů.

Celý tento systém bude možné ovládat prostřednictvím mobilního telefonu případně pomocí počítače.

2 Automatizační platformy

Automatizační platforma představuje centrální prvek celé chytré domácnosti, komunikuje se všemi senzory a zařízeními, přijatá data vyhodnocuje a na jejich základě vykonává automatizační úkony. Komerční řešení mohou být finančně nákladná, proto se především zaměřím na automatizační platformy šířené pod licencí Open Source.

Požadavky při výběru automatizační platformy

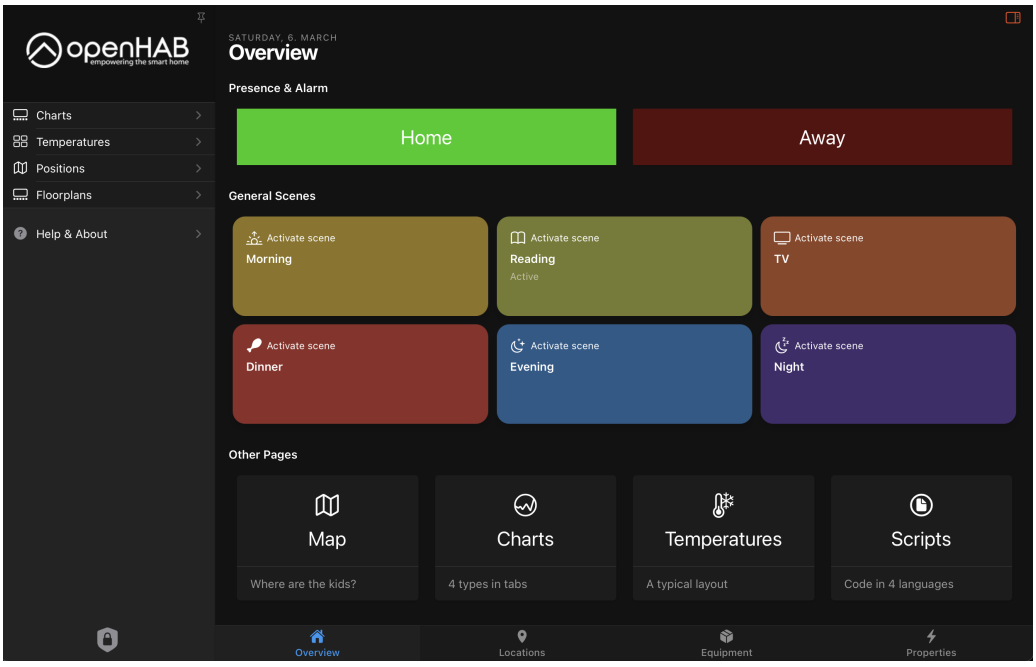
- Možnost provozu na Rasbperry Pi
- Integrace co největšího množství rozšíření a služeb
- Integrace komunikačního protokolu MQTT
- Přívětivé uživatelské prostředí a přehlednost konfiguračních souborů
- Přehledná dokumentace a komunitní podpora
- Možnost ovládání z mobilní aplikace
- Bezpečnost a stabilita

2.1 Open Source řešení

2.1.1 OpenHAB

OpenHAB[57] je jedna z nejpoužívanějších Open Source platforem domácí automatizace, která se opírá o obrovskou komunitní podporu. Na diskuzních fórech openHABu diskutují tisíce uživatelů, kteří jsou ochotni pomoci s řešením problémů. Jedná se o Open Source platformu, kterou komunita uživatelů neustále aktualizuje a vylepšuje. Tato platforma umožňuje integraci mnoha zařízení od různých výrobců nebo vlastních zařízeních, které komunikují pomocí standardních protokolů jako je např. MQTT protokol. Platformu je možné provozovat na široké škále zařízeních, od jednodeskových počítačů až po výkonné servery, toto zařízení nemusí být připojeno k internetu, což umožní vytvořit zcela izolovaný lokální systém.

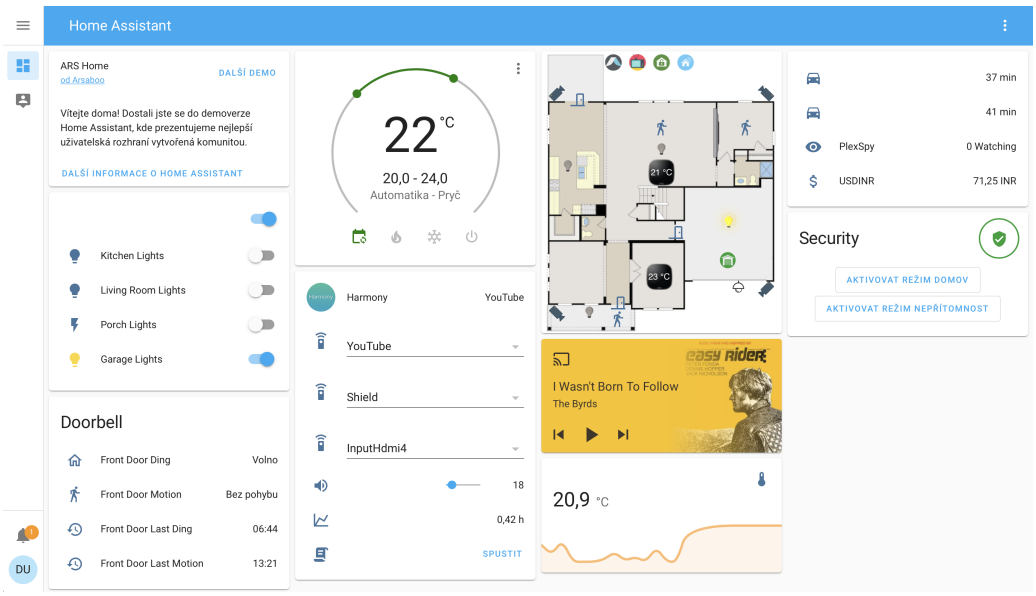
Platforma OpenHAB využívá modulů[54], které umožňují např. komunikaci s jednotlivými senzory, je vytvořena v programovacím jazyku JAVA a vyžívá frameworku OSQi, který umožňuje přidávání jednotlivých modulů bez nutnosti přerušení běhu programu. Příklad přehledové obrazovky platformy `OpenHAB` je na obrázku 2.1.



Obrázek 2.1: Platforma OpenHAB - Demo[51]

2.1.2 Home Assistant

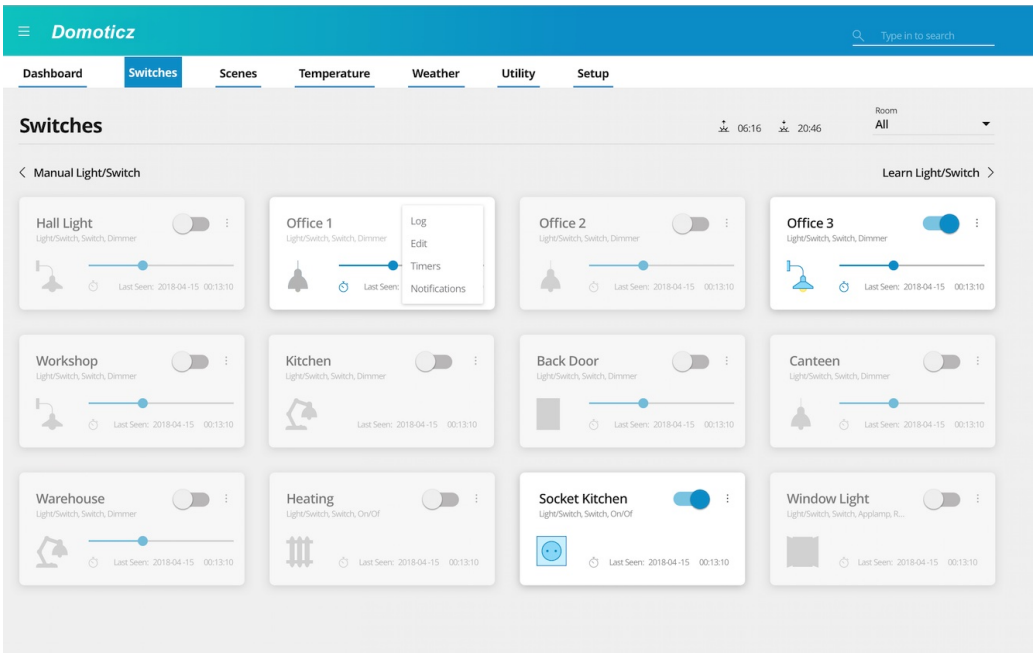
Jedná se opět o platformu[11] šířenou pod licencí Open Source, kde je kladen velký důraz na lokální ovládání a bezpečnost. K provozu není nutné využívat žádný cloud, veškeré výpočty jsou prováděny na zařízení, které je umístěno v domácnosti. Obdobně jako je tomu u předchozí platformy má i tato obrovskou komunitní podporu na diskuzních fórech, protože zde diskutuje velké množství uživatelů z celého světa. Platforma Home Assistant je implementována pomocí jazyka Python a je možné ji provozovat na několika typech zařízení, jako je např. Raspberry Pi nebo stolní počítač. Příklad přehledové obrazovky platformy `Home Assistant` je na obrázku 2.2.



Obrázek 2.2: Platforma Home Assistant - Demo[12]

2.1.3 Domoticz

Dalším silný hráčem na poli Open Source platforem pro domácí automatizace je platforma Domoticz[23], jejíž první vydání proběhlo v prosicni 2012. Jak už tomu bývá standardem je provoz této platformy možný na široké škále zařízení od Raspberry Pi přes Linux až po Mac OS. Systém této platformy je napsán v C++. Hlavní síla platformy Domoticz spočívá v její jednoduchosti, její nasezení je vhodné spíše pro nenáročná řešení. Grafické prostředí lze přizpůsobit řadou dostupných doplňku jako je např. `Machinon_Theme`[22], které je zobrazeno na obrázku 2.3.



Obrázek 2.3: Platforma Domoticz - Demo[22]

2.2 Komerční řešení

2.2.1 Amazon Alexa, Home Connect

Amazon Alexa[4] je virtuální hlasový asistent, pocházející od společnosti Amazon. Díky značné rozšířenosti je možná integrace velkého množství zařízení jako jsou např. světla, temostaty nebo kamery. Ačkoliv se jedná o cloudovou službu, je její použití možné i lokálně. Nějčastěji je tato služba využívána jako hlasový asistent. Amazon Alexa je možné využít v kombinaci se službou Home Connect[18], pomocí které můžeme prostřednictvím hlasového asistenta ovládat řadu chytrých domácích spotřebičů jako je např. pračka nebo kávovar.

2.2.2 Google Assistant, Google Smart Home

Jedná se o jednoho z nejrozšířenějších hlasových asistentů[32], který je již integrován v mnoha zařízeních - jako je např. mobilní telefon s operačním systémem Android. Jak už tomu bývá u většiny komerčních platforem, i zde probíhá zpracování dat na vzdáleném serveru společnosti Google. Hlasového asistenta od společnosti Google můžeme využít s platformou Google Smart Home[33] k ovládání zařízení v domácnosti jako jsou např. reproduktory nebo televize.

2.2.3 Siri, Apple HomeKit

Siri[35] je virtuální hlasový asistent vytvořený společností Apple. Je integrován ve všech zařízeních od této společnosti, jedná se opět o cloudové řešení, zpracování příkazů probíhá na serveru společnosti Apple.

S využitím Apple HomeKit[34] můžeme ovládat chytrá zařízení prostřednictvím hlasového asistenta Siri.

2.3 Srovnání automatizačních platforem

2.3.1 Komerční vs Open Source řešení

- **Platformy šířené pod licencí Open Source**
  Do této kategorie patří automatizační platformy `OpenHAB, Domoticz a Home Assistant`. Hlavní výhoda této platformy, která je šířena pod touto licencí, spočívá v možnosti bezplatného použití a dostupnosti zdrojového kódu. Další výhodou je velmi obsáhlá skupina uživatelů, kteří jsou otevřeni pomoci s řešením problémů a chyb. Další výhodou použití otevřené platformy je možnost téměř libovolné konfigurace chytré domácnosti. Tato skutečnost přináší i jistá úskalí, neboť je vyžadováno již zkušeného uživatele, který ovládá programovací techniky. Většinou jsou data zpracovávána lokálně.

- **Komerční řešení domácí automatizace**
  Mezi komerčně dodávané produkty pro damácí automatizace můžeme zařadit hlasové asistenty, kteří s využitím doplňků (`Amazon Alexa - Home Connect, Google Assistant - Google Smart Home, Siri - Apple HomeKit`) lze využít k řešení domácí automatizace. Nespornou výhodou nasazení komerčního řešení je často snadné zprovoznění, které zvládne i méně zkušený uživatel. Zpracování dat často probíhá na výpočetních serverech společnosti.

2.3.2 Srovnání Open Source platforem

Ke srovnání automatizačních platforem můžeme přistoupit jak z hlediska uživatelského tak i implementačního. Návrhem a implementací chytré domácnosti se většinou bude zabývat pouze jeden člen rodiny a ostatní budou pouze běžnými uživateli.

Výše zmíněné platformy šířené pod licencí Open Source (`OpenHAB, Home Assistant a Domoticz`) jsou implementovány pomocí rozdílných technolo-

gií. OpenHAB je vytvořen pomocí programovacího jazyka Java, Home Assistant byl implementován pomocí jazyka Python a Domoticz je napsán v C++.

Platforma Domoticz je vhodná především pro nasazení do menších projektů a vyniká svou jednoduchostí, zatímco platformy OpenHAB a Home Assistant díky své široké škále použití a možnosti integrace velkého množství doplňků jsou vhodné i pro projekty s obsáhlejším záběrem působnosti.

Definice automatizačních pravidel u všech těchto Open Source automatizačních platforem je možná pomocí grafického prostředí, případně u platformy Home Assistant může uživatel využít programovacího jazyka YAML a u platformy OpenHAB je možné definovat pravidla pomocí programovacího jazyka Xtend.

Dalším klíčovým faktorem při výběru vhodné platformy pro následující použití je dostupnost a kvalita dokumentace. Platforma OpenHAB nabízí kvalitně a uživatelsky velmi přehlednou dokumentaci, kde nechybí ani řada ukázkových příkladů, podobně je tomu i v případě platformy Home Assistant. Zatímco platforma Domoticz nabízí sice obsáhlou, ale uživatelsky méně přívětivou dokumentaci. Platformy OpenHAB a Home Assistant disponují velkou skupinou uživatelů, kteří horlivě diskutují na diskuzních fórech, kde je možné nalézt řešení případných vzniklých problémů.

Ovládání chytré domácnosti z mobilního telefonu je pro zvýšení komfortu použitelnosti významný faktor a tyto Open Source platformy jím disponují. Z mobilního telefonu je možné ovládání pomocí mobilní aplikace případně pomocí webového rozhranní.

Instalace systému na Raspberry Pi je v případě platforem OpenHAB nebo Home Assistant rychlá a přímočará z důvodu možnosti stažení předpřipraveného obrazu, který dostačuje pouze nahrát na microSD paměťovou kartu. Dalším krokem je nastavení přístupových údajů k WiFi v konfiguračním souboru, který se nachází na paměťové kartě a vložit kartu do Raspberry Pi. Následně počkat několik minut, než bude systém připravem. V případě platformy Domoticz je nejdříve nutné mít na Rasbperry Pi instalovaný operační systém (např. Raspbian) a následně můžeme přistoupit k instalaci systému Domoticz.

Mezi platformou OpenHAB a Home Assistant nejsou až na několik drobností velké rozdíly. Z hlediska použití nabízí obě platformy uživatelsky přívětivé prostředí, možnost ovládání z mobilního zařízení a širokou škálu možných doplňků pro rozšíření funkčnosti systému. Platforma Home Assistant je velmi rychlým tempem aktualizována, zatímco OpenHAB vydává aktualizace méně často.

Pro následující použití jsem se rozhodoval mezi automatizační platformou OpenHAB a Home Assistant, volím OpenHAB verzi 3.0.1. Následující

volbu jsem učinil z důvodu velmi přehledné a uživatelsky přívětivé dokumentace, široké skupiny diskutujících uživatelů na diskuzních fórech, moderně působícího grafického prostředí a možnosti definice automatizačních pravidel v programovacím jazyku `Xtend`.

3 Možnosti připojení senzorů

K připojení senzorů a zařízení k automatizační platformě je možné přistupovat dvěmi způsoby:

- **Fyzickým propojením**
  V tomto případě budou nejčastěji využity metalické vodiče. Použití vodičů přináší jednu zásadní nevýhodu, kterou je nutnost budování propojení mezi senzorem a řídící jednotkou. Jsou tu i jisté benefity jako je např. vyšší bezpečnost a stabilita připojení v porovnání s bezdrátovou technologií. A pokud se bude útočník snažit proniknout do vnitřní sítě, bude se muset napojit na některý z vodičů.

- **Bezdrátově**
  Připojení senzoru k platformě bezdrátovým způsobem je ve většině případů jednodušší, než využití metalických vodičů. K vytvoření spojení můžeme použít několik dostupných technologií. Při využití této technologie je klíčovým faktorem zabezpečení případně šifrování komunikace. Hlavní výhodou tohoto způsobu řešení je, že není nutné budovat fyzické propojení mezi jednotlivými senzory.

3.1 Připojení senzoru fyzickým propojením

Při volbě této možnosti[62] se nabízí možnost použití komunikace prostřednictvím `RS485`, pomocí níž můžeme připojit např. teplotní senzor nebo LCD displej. Dále je možné použít `CAN sběrnici`, přes kterou může probíhat komunikace např. s RFID čtečkou. Připojení spínače nebo relé je možné realizovat s využitím libovolného vstupního/výstupního pinu.

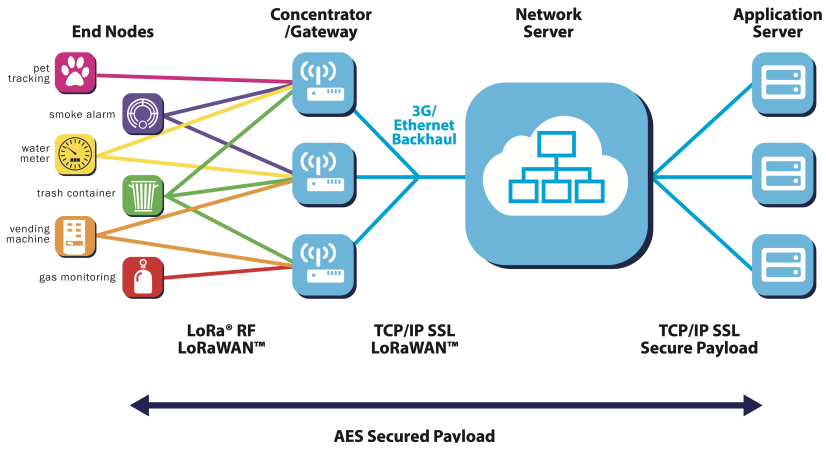
3.2 Bezdrátové komunikační technologie

3.2.1 Wi-Fi

Jedná se o velmi rychle rostoucí bezdrátovou síť[3], ve které je již připojeno přibližně 16 bilionů zařízení. Takto hojně využívaná síť vyžaduje jasně definovaná pravidla, která jsou reprezentována skupinou standardů nesoucích označení `IEEE 802.11`. Technologie je neustále vyvíjena a zlepšována, aby bylo dosaženo co možná nejvyšších přenosových rychlostí a kvality přenosu. Nejnovější verze `Wi-Fi 6`, která je definována standardy `IEEE 802.11ax`, pracuje na frekvenci `6GHz`, ostatní běžně používané standardy pracují na frekvenci `2.4 a 5 GHz`.

3.2.2 LoRaWAN®

LoRaWAN[2] (Long Range Wide Area Network) je síť, která je určená především pro komunikaci na dlouhou vzdálenost. Dosah je samozřejmě ovlivněn prostředím, ale většinou se pohybuje v řádu několika kilometrů. Použití této sítě je především vhodné u zařízeních napájených z baterie, z důvodu nízké spotřeby elektrické energie. Komunikace probíhá v bezlicenčním pásmu, v Evropě na frekvenci `433 nebo 868 MHz` a v Americe na kmitočtu `915 MHz`. Komunikace je realizována sítí bran (Gateway), které komunikují s koncovými prvky. Přenos dat je realizován pomocí `TCP/IP protokolu` k aplikačním serverům. Na obrázku 3.1 je zobrazeno schéma komunikace zařízeních v síti.



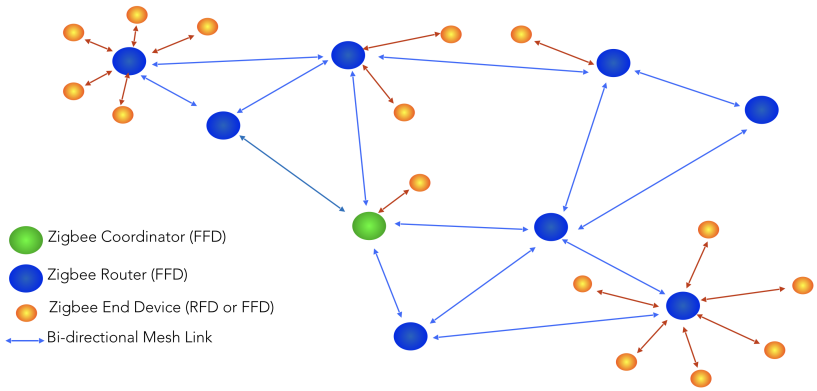
Obrázek 3.1: Schéma komunikace - LoRaWAN[2]

3.2.3 Bluetooth

Technoloie Bluetooth[64] byla navržena firmou Ericsson v roce 1994. Jedná se o osobní síť (PAN - Personal Area Network), která se řídí standardem IEEE 802.15.1, veškerá komunikace probíhá v bezlicenčním pásmu na frekvenci 2.402 - 2.480 GHz. Nejčastěji je používána verze Bluetooth® Classic, která pro svoji komunikaci využívá až 79 kanálů a přenosové rychlosti se pohybují v rozmezí od 1 Mb/s do 3 Mb/s. Dále je dostupná verze Bluetooth® Low Energy (LE). Tato verze využívá 40 kanálů a dosahuje velmi nízké spotřeby elektřiny. Přenosová rychlost se pohybuje od 125 Kb/s do 2 Mb/s. V dnešní době je tento bezdrátový způsob komunikace hojně využíván např. u bezdrátových sluchátek, či jiné přenosné elektroniky. Přenos dat pomocí této bezdrátové technologie je možné realizovat v ideálních podmínkách až do vzdálenosti 100 metrů.

3.2.4 Zigbee

Zigbee[77] představuje kompletní komunikační řešení. Komunikace probíhá v Evropě na frekvenci 868.3 MHz a 2.4 GHz, zatímco v Americe zařízení komunikují na frekvenci v rozmezí 902 - 928 MHz a 2.4GHz. Hlavní výhoda tohoto řešení spočívá v nízké spotřebě elektrické energie a eliminace možného rušení. Zigbee se řídí standardy podle protokolu IEEE 802.15.4. Síť je možné realizovat v topologii typu hvězda - Star Network (obsažen centrální prvek), strom - Tree Network nebo síť - Mesh Network. V síťové topologii jsou obsažena redundatní spojení, která poslouží v případě nedostupnosti některé z komunikačních cest. Síť slouží především na komunikaci na krátkou vzdálenost, ovšem díky síťové topologii (Mesh Network), je možné dosáhnout velkého dosahu. Na obrázku 3.2 je zobrazeno schéma komunikace v síti.



Obrázek 3.2: Schéma komunikace - Zigbee[77]

3.3 Shrnutí kapitoly

Většina domácností je pokryta Wi-Fi sítí. Z tohoto důvodu je vhodné realizovat komunikaci mezi jednotlivými senzory s centrálním ovládacím prvkem prostřednictvím Wi-Fi sítě. Případně se nabízí využití technologie Zigbee nebo kombinace obou výše zmíněných technologií.

V případě kamerových modulů je pro přenos dat třeba využít technologii, která disponuje dostatečnou rychlostí přenosu. Mezi takové sítě bezpochyby patří Wi-Fi, případně Bluetooth. Naopak pro komunikaci mezi světelným spínačem a řídící jednotkou je možné využití technologie Zigbee, v tomto případě je objem přenášených dat velmi malý.

Pro komunikaci senzorů umístěných v domácnosti s centrální jednotkou naopak není vhodné použití technologie LoRaWAN®, která je především určena pro komunikaci na velkou vzdálenost. Její nasazení je možné v případě, že uživatel bude chtít do své chytré domácnosti integrovat senzory umístěné např. na chatě nebo v garáži. Ovšem toto místo musí disponovat pokrytím sítě LoRaWAN®.

V případě potřeby je možné využití přijímače a vysílačů pracujících na frekvenci např. 433MHz nebo 2.4GHz a realizovat přenos dat pomocí vlastního přenosového protokolu.

Pro následující použití volím pro přenos dat mezi senzory a centrální jednotkou Wi-Fi síť, z důvodu dostupnosti ve většině domácností a nepotřebě použití externích komunikačních modulů (centrální jednotka i řídící jednotka senzorů obsahuje integrovanou Wi-Fi). Hlavní nevýhoda použití Wi-Fi sítě spočívá v možné vyšší spotřebě elektrické energie u jednotlivých senzorů.

4 OpenHAB

4.1 OpenHAB úvod

OpenHAB[47] je otevřená platforma, která slouží k ovládání chytré domácnosti, pomocí níž vytvoříme centrální prvek, který bude komunikovat s jednotlivými zařízeními a senzory. Mezi jednu z hlavních výhod[56] patří možnost snadné definice pravidel, pomocí kterých bude domácnost ovládána. Platforma dokáže komunikovat s velkým množstvím systémů, tudíž není nutné při návrhu chytré domácnosti spoléhat na senzory a výstupní zařízení od jediného výrobce.

Platforma OpenHAB využívá modulů, které umožňují např. komunikaci s jednotlivými senzory. Je vytvořena v programovacím jazyku JAVA a využívá frameworku OSQi, který umožňuje přidávání jednotlivých modulů bez nutnosti přerušení běhu programu.

4.2 OpenHAB Items

Položky (Items)[53] jsou jednotlivé dílčí prvky, pomocí kterých uživatel interaguje s platformou OpenHAB. Definice položek je možná dvěma způsoby (od verze 2.x).

- Pomocí uživatelského grafického prostředí, ke kterému získáme přístup skrze webový prohlížeč. Tato varianta je vhodná především pro méně zkušené uživatele, kteří nemají žádné zkušenosti s programováním.
- Pro pokročilejší uživatele se nabízí možnost editace konfiguračních souborů, které jsou umístěny ve složce $OPENHAB_CONF/items, je možné vytvořit jeden nebo více souborů s koncovkou .items, ale je důležité zachovat jedinečnost názvu položky ve všech dokumentech.

Pomocí obou výše uvedených způsobů můžeme dosáhnout stejných výsledků, ale liší se způsob dosažení cíle. Důležitým faktorem pro vytvoření přehledného kódu je vhodné pojmenování jednotlivých položek. Jméno každé položky musí být jedinečné a vytvořené pouze z písmen a číslic (název položky nesmí začínat číslicí a v celém názvu nesmí být obsaženy speciální znaky). Vhodný název položky může např. vycházet z umístění fyzického

objektu, ke kterému se tato položka vztahuje např. světlo umístěné v koupelně v prvním patře - položku nazveme `PrvniPatro_Koupelna_svetlo1`. V uživatelském prostředí, které bude sloužit pro ovládání chytré domácnosti, není vhodné mít zobrazený název položky ve tvaru `Patro_Místnost_NázevPoložky`, ale přizpůsobit zobrazený název do určité míry lidskému oku. K tomuto účelu složí `Štítek (Label)`, který určuje pod jakým názvem se bude položka zobrazovat v uživatelském prostředí. Pomocí štítku můžeme ovlivnit i formátování výstupu např. zobrazení teploty na dvě desetinná místa.

Jednotlivé položky je možné přidávat do tzv. `skupin (Group)`. Skupiny mohou být rozděleny podle využití daného zařízení např. osvětlení nebo lokalizace v domě např. zásuvky v kuchyni.

Položky mohou být několika datových typů např.:

- přepínač, kontakt, posuvník
- barva, obrázek, lokace
- číslo, textový řetězec

4.3 OpenHAB Things

Openhab Things[55] mohou reprezentovat od fyzických objektů až po objekty představující nějakou službu. Většina entit typu `Things` poskytuje několik služeb.

Každý objekt typu `Things` se může nacházet v několika stavech např. `UNINITIALIZED`, které nám umožní snadněji identifikovat případné problémy.

`OpenHAB` má implementovanou službu `Thing Discovery[52]`, která slouží k automatickému objevování objektů (typu `Things`) v síti. K nalezeným objektům můžeme přistupovat pomocí webového prohlížeče. Následující postup je pouze na uživateli. Schválením nalezeného objektu se stane tento objekt dostupný v celém systému a je připraven pro další použití. Je zde možnost i automatického schvalování nalezených objektů, která v defaultním nastavení není aktivována.

Příkladem objektu typu `Thing` může být např. `MQTT Bridge`, který slouží k provázání komunikace probíhající prostřednictvím `MQTT protokolu` a platformy `OpenHAB`.

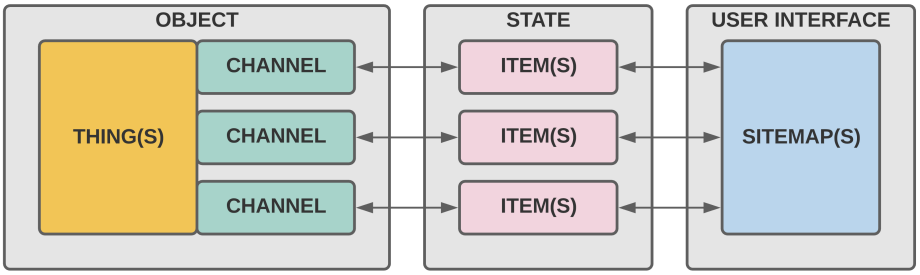
4.4 Channels

`Kanály (Channels)[49]` slouží k propojení mezi `fyzickou a virtuální vrstvou`. Kanály jsou reprezentovány jako odkazy. Přiřazením konkrétního odkazu určité položce bude mít následně za efekt, že daná položka bude reagovat na zprávy určené tomuto kanálu. Odkaz pro jeden kanál musí být jedinečný v celém systému.

Kanál může představovat např. odkaz na poslední doručenou zprávu prostřednictvím `Telegramu`.

4.5 Sitemap

Openhab `Sitemap` slouží ke snadné definici uživatelského rozhraní. V souboru `*.sitemap` je možné definovat např. přepínač, který bude sloužit k ovládání stavu světla. Na obrázku 4.1 je zobrazena návaznost mezi jednotlivými konfiguračními soubory.



Obrázek 4.1: OpenHAB - konfigurační soubory

4.6 Doplňky

Platforma OpenHab ve verzi 3 umožňuje přidání několika doplňků[48], které umožní snadnější integraci systému OpenHab do chytré domácnosti. Pomocí doplňků může platforma OpenHab spolupracovat s mnoha zařízeními od různých výrobců.

Jednotlivé doplňky můžeme rozdělit do několika základních kategorií.

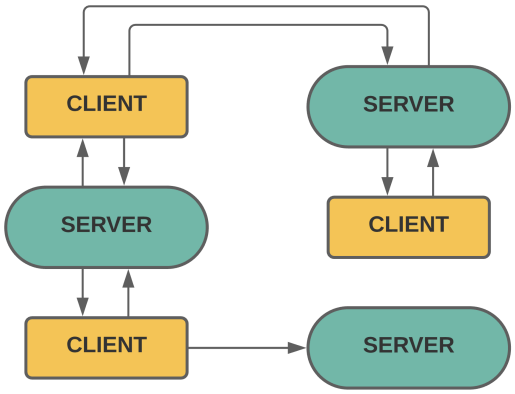
- Bindings - vazby mezi platformou OpenHab a hardwarem od ostatních výrobců jako jsou např. Xiaomi nebo MQTT brokerem Eclipse Mosquitto.
- System Integrations - integrace systémů ostatních výrobců jako jsou např. Google nebo Apple.
- Actions - slouží k vykonání předdefinovaných metod. Je možné odesílat zprávy na aplikaci Telegram nebo s využitím doplňku Twitter Actions přidat příspěvek na Twitter.
- Data Persistence - slouží k vytváření statistických dat. Existuje zde možnost připojení do MySQL databáze.
- Data Transformation - pomocí tohoto doplňku můžeme vykonávat Exec příkazy nebo využít rozšíření JavaScript Transformation Service k přeměně vstupu na výstup pomocí JavaScript.
- Voice - slouží k převodu mluveného slova na text nebo naopak.

5 Komunikační protokol

Komunikační protokol bude využit při předávání všech zpráv mezi jednotlivými senzory a centrálním prvkem. Centrální prvek bude např. komunikovat s dveřním zámkem. Z tohoto důvodu je nutné, aby přenos dat byl zabezpečen. Další klíčovým parametrem je spolehlivost přenosu. V některých případech je možné akceptovat ztrátu některých dat např. přenos dat z teplotního senzoru, ale vetšinou je tato ztráta dat neakceptovatelná. Přenos zpráv mezi jednotlivými senzory a centrálním prvkem bude realizován po síti, takže volíme nějaký z dostupných síťových komunikačních protokolů. Nabízí se možnost použití `HTTP protokolu[28]`, který ovšem není nejvhodnější z důvodu, že je určen primárně pro přenos dokumentů. Mezi senzory a centrálním prvkem bude typicky přenášen malý objem dat. Další možností je využití komunikačního protokolu `CoAP[73]` nebo `MQTT[40]`. Oba tyto protokoly jsou vhodné pro nasazení do internetu věcí.

5.1 Constrained Application Protocol

Constrained Application Protocol zkráceně `CoAP[74]` je moderní komunikační protokol splňující normu `RFC 7252`, který je především určen pro komunikaci typu `M2M (machine-to-machine)`. Byl navržen pro přenos dat v internetu věcí. Jeho silnou stránkou je možnost přenosu dat v nestabilních sítích. Pomocí tohoto komunikačního protokolu můžeme přenášet data v různém datovém formátu. Protokol je bezpečný a jeho nasazení je možné na mikrokontrolérech přibližně s `10 KiB RAM a 100 KiB ROM`.

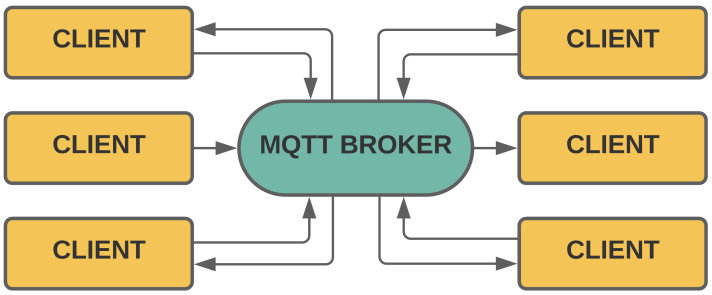


Obrázek 5.1: Schéma komunikace - CoAP[73] protokol

5.2 Message Query Telemetry Transport

Message Query Telemetry Transport zkráceně MQTT protokol[40] vznikl v roce 1999. Na jeho vytvoření se podíleli Dr. Andy Stanford-Clark (IBM) a Arlen Nipper (Arcon). Hlavním cílem bylo vytvoření jednoduchého komunikačního protokolu. Již od roku 1999 se rozšířilo použití MQTT protokolu do několika průmyslových odvětvích např. automobilový či ropný průmysl.

Díky jednoduchosti protokolu je možný jeho provoz na zařízeních s nízkým výpočetním výkonem, což přináší možnost využití komunikace pomocí MQTT protokolu v nepřeberném množství senzorů a čidel, které mohou být napájena ze sítě nebo z baterie. Přenos dat pomocí tohoto protokolu je možné realizovat i prostřednictvím sítí s nízkou spolehlivostí či vysokou latencí (velký čas odezvy). Zároveň díky implementaci QoS je dosaženo požadovaného stupně doručení přenášených dat. Protokol je založen na publikování a odběru dat. Na obrázku 5.2 je zobrazeno komunikační schéma, ze kterého je patrné, že ne každý klient musí současně provádět operace publikování i odběru např. teplotní čidlo bude pouze publikovat a relé modul pouze odebírat data.



Obrázek 5.2: Schéma komunikace - MQTT[40] protokol

5.2.1 MQTT Quality of Service

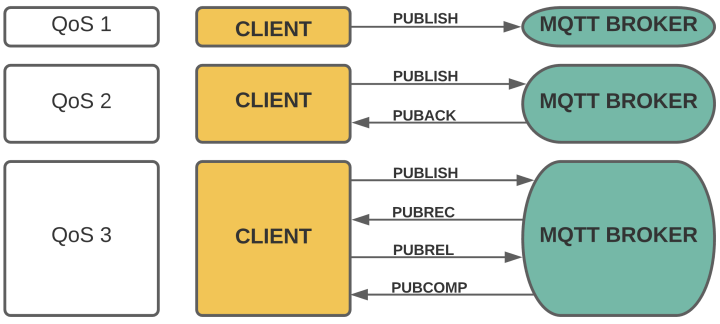
- `Stupeň 0` - Doručení zprávy maximálně jednou
- `Stupeň 1` - Doručení zprávy alespoň jednou
- `Stupeň 2` - Doručení zprávy přesně jednou

Volba stupně `QoS`[30] má vždy své opodstatnění. `Stupeň nula` volíme v případě, že nedoručení některých dat nezpůsobí žádný problém např. data z teplotního senzoru. Při použití této konfigurace je značnou výhodou malé množství přenášených dat, nedochází zde k žádnému potvrzení o doručení či nedoručení datového paketu. Tento způsob přenosu dat můžeme přirovnat k `protokolu UDP`.

`Stupeň jedna` volíme v případě, že je nutné alespoň jednou přijmout daný datový paket, ale zároveň systém dokáže správně reagovat na duplicitně přijaté datové pakety. Volba této možnosti je jakýsi kompromis mezi rychlostí a kvalitou doručení dat. Přenos dat zabere delší časový úsek oproti stupni kvality doručení nula `(QoS 0)`, ale je zde garantováno doručení. Tahle volba je vhodná např. u přístupového systému, kdy je důležité alespoň jednou doručit data, že určitá osoba vstoupila do budovy a zároveň s duplicitními daty není problém se vypořádat.

Nejvyšší volbou kvality služeb je volba `stupně dva`, zvolením této možnosti zajistíme doručení zprávy přesně jednou. Ze všech dříve zmíněných možností je zde největší datový tok a doručení zprávy zabere největší časový úsek.

Na obrázku 5.3 je zobrazena komunikace při jednotlivých úrovních QoS.



Obrázek 5.3: Schéma komunikace QoS

5.2.2 Bezpečnost

Jelikož pomocí MQTT protokolu komunikují klíčové prvky chytré domácnosti např. zámek na vstupních dveřích, je zde velmi důležitým faktorem bezpečnost[31]. Zabezpečení provedeme na síťové, transportní a aplikační vrstvě ISO/OSI modelu.

- Síťová vrstva - Zde je klíčové zabezpečení sítě po které zařízení (klienti a server) komunikují. Můžeme využít VPN síť nebo např. na routeru využít zabezpečení pomocí filtrování MAC adres zařízení.
- Transportní vrstva - Použití šifrovacích algoritmů TLS / SSL. Algoritmy zajistí integritu a zabezpečení přenášených dat.
- Aplikační vrstva - V aplikační úrovni můžeme využít ověření zařízení nebo uživatele pomocí přístupového jména a hesla, případně přenášet již zašifrovaná data (celkový přenos dat není šifrován).

5.2.3 Shrnutí kapitoly

Výše zmíněné komunikační protokoly jsou nejpoužívanější protokoly v internetu věcí. Protokol CoAP[73] je vhodný především pro komunikaci typu klient-server, zatímco MQTT protokol[40] je vhodný pro komunikaci typu M:N díky publikování a odběru zpráv. Pro následující použití volím protokol MQTT, z důvodu vhodnosti využití odběru a publikování zpráv.

QoS je vhodné u většiny senzorů nastavit na stupeň 1 (doručení zprávy alespoň jednou). Návrh automatizačních pravidel a logiky bude navržen způsobem, aby doručení duplicitních dat nezpůsobilo žádné problémy. Při volbě stupně 0 by např. nemusela být doručena zpráva o stisku vypínače, což by způsobilo nerozsvícení světla. Případně při volbě stupně 2 (doručení zprávy přesně jednou) by byly kladeny vyšší nároky na síťový provoz. U senzorů, ze kterých nepřicházejí příliš důležitá data, je interval mezi jednotlivými měřeními malý a tudíž je možné nastavit QoS na stupeň 0 (doručení zprávy maximálně jednou).

6 Použitý Hardware

6.1 Výběr vhodného hardware pro senzory

Vhodné je, aby senzory mohly komunikovat bezdrátovým způsobem. Je zde více možností komunikace např. prostřednictvím `Wi-Fi`, případně použití komunikačních modulů pracujících na frekvenci `433 nebo 868 Mhz`. Výhoda použití sítě `Wi-Fi` spočívá v tom, že vetšinou touto sítí je pokryta celá domácnost, naopak použití externích komunikačních modulů přináší výhodu spočívající v možnosti komunikace mimo dosah `Wi-Fi`, u běžných vysílačů až na `150m` v otevřeném prostoru.

V případě použití senzorů v domácnosti ve většině případů není nutný provoz na baterii, ale spotřeba elektrické energie senzoru je důležitým faktorem, zvláště při neustále se zvyšujících cen energií. Dalším kritériem při výběru modulu, který bude použit v senzoru, je pořizovací cena.

Při výběru vhodného kamerového modulu se nabízelo více možností. První možností je použití vývojové desky `ESP32-CAM`, která vyniká především nízkou pořizovací cenou a malou spotřebou elektrické energie. Další finančně náročnější možností je použití modulu `Raspberry Pi`, případně využití některé z průmyslově vyráběných `IP kamer`.

Po zvážení všech možností se jeví jako nejvhodnější použití vývojové desky `ESP32-CAM`.

Dále je nutné zvolit hardware pro obecný typ senzoru jako je např. teplotní čidlo nebo pohybový senzor. Na výběr je zde z několika možností. První možnost je použití modulu `ESP8266`, který nabízí možnost komunikace pomocí `Wi-Fi` sítě. Další možnou alternativou je použití vývojové desky `Arduino NANO`, která ovšem nenabízí možnost komunikace bez použití externího komunikačního modulu.

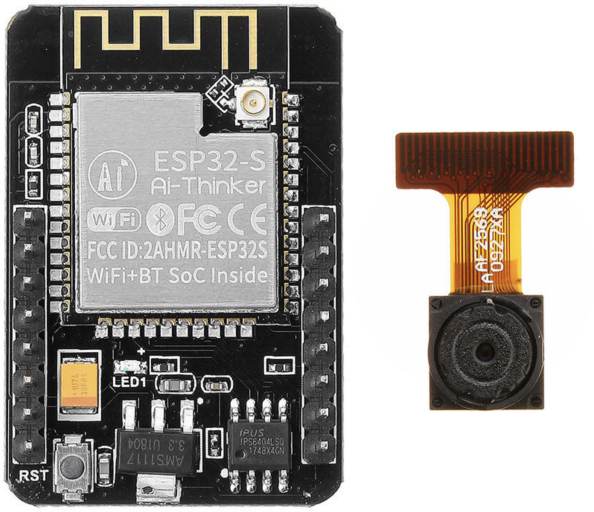
Pro obecný senzor jsem zvolil modul `ESP8266` konkrétně typ `ESP12-F`. Jedná se o modul, který má integrovanou `Wi-Fi` anténu, dále obsahuje několik `GPIO pinů` a jeho pořizovací cena je u některých českých obchodníků nižší než 100 Kč.

6.1.1 ESP32 - CAM

ESP32-CAM[24] je vývojová deska, která nabízí velmi všestranné využití. Deska je osazena kamerovým modulem OV2640, slotem pro paměťovou microSD kartu a několika GPIO piny, pomocí kterých můžeme připojit různé senzory. S okolním světem nabízí možnost komunikace pomocí Wi-Fi nebo Bluetooth.

Srdcem celé vývojové desky je kombinovaný modul ESP32-S[17], který je založen na chipsetu ESPRESIFF. Modul je osazen dvoujádrovým procesorem Xtensa® 32-bit LX6, který dosahuje výpočetního výkonu až 600 DMIPS. Díky nízké spotřebě energie je modul ESP32-S za určitých podmínek vhodný pro nasazení do mobilních zařízení napájených z baterie.

Vývojová deska ESP32-CAM z důvodu integrace RAM paměti a dalších pomocných obvodů již nedosahuje tak nízké spotřeby elektrické energie jako samostatný modul ESP32-S. Spotřeba elektrické energie [72] celého modulu dosahuje při zhasnuté led diodě cca 180 mA a při rozsvícené až 310 mA. Led dioda slouží pro osvětlení např. fotografované scény. V režimu spánku se pohybuje spotřeba modulu od 6 mA do 20 mA, takto široký rozptyl spotřeby je způsoben několika možnostmi režimu spánku.



Obrázek 6.1: Vývojová deska ESP32 s kamerovým modulem OV2640 [24]

6.1.2 ESP12F

Jedná se o jednojádrový procesor[16] vyvinutý společností `Ai-Thinker - Technology`.

Tento `32 bitový` procesor pracuje na `frekvenci 80 MHz a 160 MHz`. Modul je možné provozovat jako samostatně běžící a díky integraci `Wi-Fi` a kompletního `TCP/IP zásobníku` je modul vhodný pro nasazení do IoT. Modul obsahuje několik `GPIO pinů` a komunikace je možná pomocí `SPI/SDIO rozhraní` nebo využitím `I2C/UART sběrnice`. Modul má velmi malé rozměry `16mm x 24mm`, pracuje s napětím `3.3V`, tudíž pro připojení zařízení pracujících s větším napětím je nutné např. využití převodníku napěťových úrovní.

Obrázek 6.2: Modul ESP12-F [10]

6.2 Výběr vhodného hardware pro centrální prvek

Výběr vhodného hardware je jedním z klíčových rozhodnutí při realizaci chytré domácnosti.

Velký důraz je kladen na možnou **rozšiřitelnost**, tzn. přidání nového senzoru musí být snadné a **bez změn síťové infrastruktury nebo změn konfiguračních souborů**. Aby bylo toto možné použijeme **centralizovaný systém**. Na centrální prvek budou kladeny několikrát větší výpočetní nároky, než tomu je u jednotlivých modulů v senzorech. Dále toto zařízení musí být vhodné pro nepřetržitý provoz a zároveň musí dosahovat vysoké spolehlivosti.

Jako centrální prvek můžeme použít např. **stolní počítač, notebook, server** nebo některý z dostupných **jednodeskových počítačů** jako je např. `Raspberry Pi`. Při použití výkonného serveru případně počítače získáme

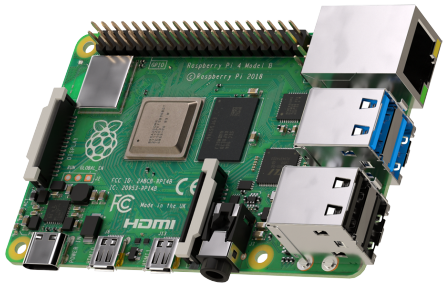
výpočetní stroj s rychlou reakcí na přicházející podněty, což oceníme např. při zpracování obrazu. Podstatnou nevýhodou je spotřeba elektrické energie takových zařízení. V případě výpadku elektrické energie jejich dlouhodobější napájení z baterie není snadno realizovatelné.

Volím jednodeskový počítač `Raspberry Pi 4 Model B s 8GB RAM`. Výhodou je dostatečný prostor pro možnosti rozšiřování navržené realizace. Další podstatnou výhodou je možnost napájení z baterie a díky malé spotřebě je možné napájení realizovat po dostatečnou dobu, než dojde například k obnovení dodávky elektrické energie.

6.2.1 Raspberry Pi 4 Model B - 8GB RAM

`Raspberry Pi4[37]` je jednodeskový počítač o velikosti platební karty. Tento model obsahuje `64 bitový procesor` se `čtyřmi jádry` a `8GM RAM` paměti. Nechybí zde integrace `Wi-Fi (802.11 b/g/n/ac)` a `Bluetooth 5.0`. Pomocí dvou micro-HDMI portů můžeme připojit až dva 4K monitory. Dále `Raspberry Pi4` disponuje čtyřmi USB porty, gigabit Ethernet portem, konektorem pro připojení displeje a kamery. Nechybí zde ani `28 GPIO` pinů a slot pro paměťovou micro SD kartu.

Na tomto jednodeskovém počítači můžeme provozovat několik operačních systémů např. `Raspbian` nebo některé distribuce `Linux (Ubuntu Core)`. Zařízení pro správný chod vyžaduje stabilní `5V 3A` napájecí zdroj. Napájení je realizováno pomocí USB C konektoru. Doporučená provozní teplota se pohybuje od 0 do 50 stupňů Celsia. Systém řízení při malé zátěži procesoru snižuje napětí a výkon procesoru, čímž je zabráněno nepotřebnému zahřívání zařízení. Teplota procesoru nikdy nepřesáhne 85 stupňů Celsia. Externí chlazení není vyžadováno, ale při provozu výkonově náročných aplikací lze přidat externí chladič.



Obrázek 6.3: Raspberry Pi 4 Model B - 8GB RAM

6.3 Výběr vhodných senzorů

6.3.1 Senzor teploty

Bezpochyby mezi nejzákladnější senzory patří teplotní senzor. Při výběru tohoto senzoru se nabízí několik možností. Jednotlivé senzory dosahují různých přesností měření. Většinou zde platí přímá úměra mezi pořizovací cenou a přesností měření. Další odlišnost mezi jednotlivými teplotními senzory spočívá ve způsobu připojení k mikrokontroléru. Některé komunikují pomocí `I2C sběrnice(bme 280[29])`, jiné využívají pouze jeden digitální pin(`DHT11[27], Dalas DS18B20`). Na trhu jsou dostupné senzory k měření teploty kapalin tzn. jsou obaleny vodotěsným pouzdrem. V tomto provedení je dostupný např. teplotní senzor `Dalas DS18B20[76]`.

6.3.2 Relé modul

Jedná se o výstupní prvek[67], který lze využít ke spínání libovolných elektrických zařízení jako je např. stropní ventilátor nebo lampička na nočním stolku. Většina relé modulů, které jsou dostupné pro Arduino, je napájeno napětím `5V`. Vstupní pin, který slouží ke spínání relé je opticky oddělen, což snižuje pravděpodobnost poškození ovládacího mikrokontroléru. Ve většině případů je možné spínat napětí až `AC250V 10A nebo DC30V 10A`.

Modul, který obsahuje elektromagnetické relé, při sepnutí nebo rozpojení vydává zvuk. Další možností je použití `SSR Relé`, kde je ke spínání použit `triak`, který zvuk nevydává. Využití `SSR Relé` má několik výhod např. bezhlučný provoz a nízkou spotřebu elektrického proudu při spínání. Delší životnost je způsobena absencí mechanických částí. Je zde i několik nevýhod jako je např. zahřívání při vysokém protékajícím proudu (nutno chladit).

6.3.3 433MHz přijímač

Využití toho senzoru (přijímače) přináší možnosti použití bezdrátových ovladačů (pracujících na frekvenci 433 MHz) a příjem libovolných dat, která jsou na této frekvenci vysílána. Pro následující použití jsem zvolil přijímač `433MHz SRX887[25]`, který dosahuje vysoké citlivosti a nízké spotřeby elektrického proudu v pohotovostním režimu.

6.3.4 Kamerový modul

Vývojová deska ESP32-CAM[24] obsahuje kamerový modul OV2640 Color CMOS UXGA[43].

OV2640 je kamerový modul, který je schopen pracovat při nízkém napětí a pořizovat snímky s rozlišením až 1600 x 1200 pixelů (2.0 MegaPixel). Na výstupu jsou 8/10 bitové obrazy. Modul je plně ovládán pomocí Serial Camera Control Bus (SCCB). Při plném rozlišení je schopný pracovat s 15 fps (15 snímků za vteřinu). Uživatel má plnou kontrolu nad všemi klíčovými parametry, které ovlivní finální podobu fotografie, jako je např. expozice, sytost barev, potlačení šumu nebo vyvážení bílé barvy.

Nechybí zde ani implementace několika automatizačních funkcí jako je např. AEC-(automatické řízení expozice), AEW-(automatické vyvážení bílé barvy) nebo možnost potlačení šumu.

6.3.5 Display modul

K předání informací uživateli můžeme přistoupit dvěma základními způsoby. Využitím **audio příslušenství v kombinaci s hlasovým asistentem**, nebo některou z dostupných **zobrazovacích jednotek**. Zobrazovací jednotky můžeme taktéž rozdělit na dvě skupiny. Mezi nejrozšířenější zcela jistě patří využití **displeje (digitální zobrazovače)**, méně častou variantou je využití **analogových modulů** např. k zobrazení teploty.

Pro využití v domácí automatizaci k zobrazení jednoduchých údajů můžeme využít LCD displej[66]. Tento displej neslouží k zobrazování obrázků nebo diagramů, ale pouze textových a číselných hodnot, což je pro zobrazení např. teploty či vlhkosti dostačující. Dále tento displej je možno použít k zobrazení dalších informativních účelů, jako je např. zobrazení krátkého textového pokynu. LCD displeje jsou nejčastěji dostupné s možností zobrazení 16 znaků na 2 řádcích nebo 20 znaků na 4 řádcích.

6.3.6 RFID modul

Pro identifikaci osob, např. při vstupu do budovy, můžeme použít několik způsobů identifikace. Mezi nejčastější patří použití **klíče**, což přináší výhody např. dostupnost a pořizovací cena. Podstatnou nevýhodou je nemožnost definice přístupových práv bez změny klíče případně zámku. Další alternativou je použití **čtečky otisků prstů** nebo jiných **biometrických údajů**, což přináší vysoké zabezpečení. Nasazení těchto technologií je finančně nákladnější a ne vždy vhodné, např. v zimních měsících, kdy lidé nosí rukavice nelze použít čtečku otisku prstů. Mezi další možnosti patří použití identifikačních

karet. Identifikační karty jsou levné, některé obtížně kopírovatelné.

Nejběžnější a nejdostupnější technologií je použití RFID čtečky karet. Pro použití se nejlépe hodí modul RC522[41], pro který jsou dostupné i knihovny, tudíž jeho nasazení je finančně dostupné a snadno implementovatelné. Tento modul je možné používat s většinou karet, které pracují na frekvenci 13.56 MHz.

6.3.7 Pohybový Senzor

Jedná o senzor určený k detekci pohybu. Nejčastěji jsou používány pasivní infračervená čidla PIR Senzor[39], která detekují pohyb podle vyzařování infračerveného záření z objektů. Vzhledem k tomu, že v plánovaném systému bude obsažen kamerový modul, je možné použít způsob detekce pohybu založený na metodě porovnávání snímků pořízených z kamerového modulu. Tento způsob jsem zvolil z důvodu širokých možností konfigurace detekce pohybu a případné možnosti další rozšiřitelnosti.

7 Vývojová prostředí pro ESP32-CAM a ESP12F

7.1 Microsoft Visual Studio

Microsoft Visual Studio[38] je všestranné vývojové prostředí pocházející od firmy Microsoft. Verze Microsoft Visual Studio Community je volně ke stažení pro nekomerční účely. Toto prostředí na rozdíl od Arduino IDE nemá tak úzké pole působnosti, ale můžeme do něj přidat různá rozšíření a použít ho pro vývoj různých konzolových nebo grafických aplikací. Než začneme s vývojem programu pro Arduino je nejprve nutné doinstalovat plugin VisualMicro, který nám umožní vytvořit projekt kompatibilní s Arduino. Díky pluginu VisualMicro můžeme také využít sériového monitoru ke sledování komunikace, která probíhá mezi PC a ESP32.

Nejprve je potřebné stáhnout instalační soubor, který je dostupný na následující adrese:
`https://code.visualstudio.com`. Pro naše potřeby postačuje Visual Studio Code, které je na hranici textového editoru a IDE. Po instalaci Visual Studio Code je nutné přidat rozšíření PlatformIO IDE for VSCode, díky kterému můžeme pracovat s mnoha vývojovými deskami včetně ESP32-Cam Ai-Thinker nebo ESP8266.

7.1.1 Instalace - verze: 1.39.2

1. Otevřeme Visual Studio Code
2. Přejdeme na záložku Extensions: MARKETPLACE
3. Vyhledáme rozšíření PlatformIO IDE a nainstalujeme.
4. Restartujeme VSCode, rozšíření spustíme kliknutím na domeček ve spodní liště.

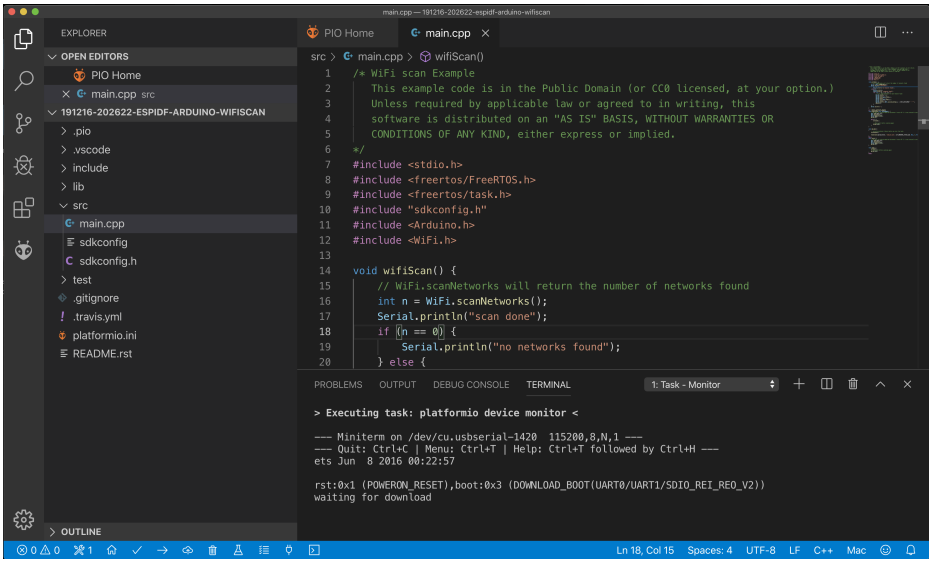
7.1.2 Spuštění prvního programu ESP32-CAM

Spustíme Visual Studio Code, automaticky dojde k aktivaci rozšíření PlatformIO IDE. Dále pokračujeme kliknutím na tlačítko `Project Examples` a zvolíme některý z ukázkových programů od `Espressif 32` v našem případě `espidf-arduino-wifiscan` a potvrdíme stisknutím tlačítka import.

Po importu projektu přejdeme do třídy main.cpp, kde je hlavní zdrojový kód. V případě macOS použijeme klávesovou zkratku `command+shift+p`, kterou vyvoláme příkazový řádek, pomocí kterého spustíme jednotlivé příkazy PlatformIO IDE.

- PlatformIO: Serial Monitor
- PlatformIO: Build
- PlatformIO: Upload

Na obrázku 7.1 můžeme vidět vývojové prostředí Visual Studio Code s rozšířením PlatformIO IDE. Ve spodní části vývojového prostředí se nachází terminál (sériový monitor).



Obrázek 7.1: Visual Studio Code + PlatrormIO

7.2 Arduino IDE

Arduino IDE[6] je open source vývojové prostředí, které umožňuje od samotného vytvoření programu až po přenos binární podoby kódu do ESP modulu. Při prvním spuštění vývojového prostředí je nutné doinstalovat modul pro práci s vývojovou deskou ESP32 a ESP8266. Toto prostředí je velmi vhodné především pro méně zkušené uživatele, protože obsahuje několik ukázkových zdrojových kódů, které stačí pouze nahrát na desku a můžeme přistoupit k testování. Dále je ve vývojovém prostředí integrován tzv. sériový monitor, který slouží ke sledování komunikace mezi PC a vývojovou deskou.

Samozřejmě Arduino IDE je sice nejpoužívanější vývojové prostředí pro tvorbu programů pro ESP32/ESP12F, ale není jediným vývojovým prostředím, které můžeme použít. Jeho nesporná výhoda spočívá ve velké rozšířenosti, tudíž lze dohledat spoustu návodů, jak ho správně používat, případně jak řešit vzniklé problémy.

Na obrázku 7.2 je zobrazeno vývojové prostředí, ve spodní části se nachází terminál, dále je zde zobrazen sériový monitor. Vývojové prostředí je zachyceno v okamžiku po sestavení a následném zavádění programu na vývojovou desku.

7.2.1 Instalace - verze: 1.8.10

1. Nejprve je nutné stáhnout instalační soubor z `https://www.arduino.cc/en/Main/Software`. Je zde [60] několik variant jak pro macOS, Windows i Linux - v našem případě pro macOS

2. po instalaci spustíme aplikaci **Arduino IDE** a v horním panelu zvolíme `Arduino > Preferences` vyhledáme `Additional Boards Manager URLs:` a vložíme:
   `https://dl.espressif.com/dl/package_esp32_index.json http://arduino.esp8266.com/stable/package_esp8266com_index.json`

3. Otevřeme Tools > Board > Boards Manager a do vyhledávacího pole zadáme "esp32"a zvolíme `ESP32 by Espressif Systems` verze 1.0.2, stejným způsobem přidáme potřebné soubory pro ESP8266 `esp8266 by ESP8266 Community`, tímto je instalace dokončena.

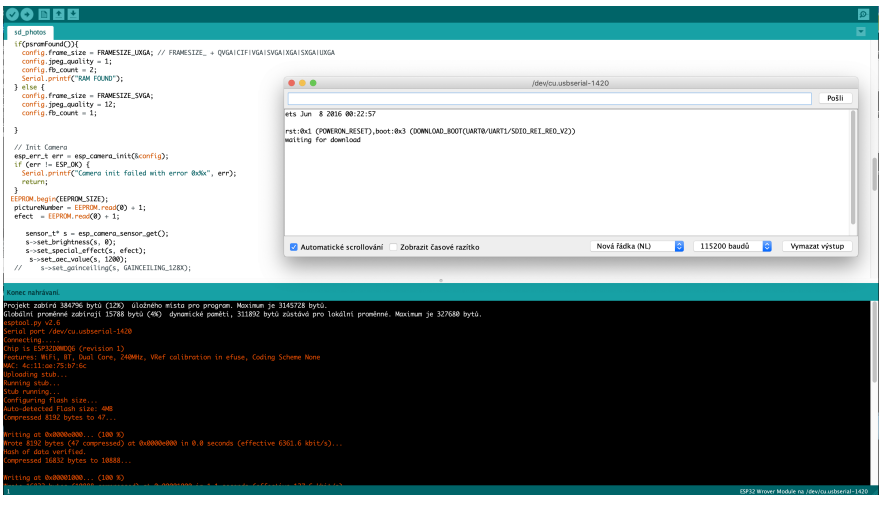
7.2.2 Spuštění prvního programu ESP32-CAM

Spustíme aplikaci Arduino IDE[6] a nastavíme správnou vývojovou desku `Tools -> Board -> DOIT ESP32 DEVKIT V1`. Pokračujeme volbou portu na kterém je zařízení připojeno `Tools -> Port -> XXXX` (XXXX = číslo portu), nastavíme přenosovou rychlost na 115200 baud a rychlost čtení z paměti na 80 MHz a Flash Mode na `QIO` [1].

Dále zvolíme Partion Scheme jako `Huge APP (3MB No OTA)`. `No OTA` znamená, že aplikace není vhodná pro zavedení programu na vývojovou desku pomocí WiFi. Pokud chceme vyzkoušet funkčnost na některém z přiložených vzorových programů, tak volíme `File -> Examples -> WiFi (ESP32) -> WiFi Scan`, na tomto příkladu navíc otestujeme funkčnost WiFi.

V tomto okamžiku máme otevřený ukázkový projekt a můžeme pokračovat uploadem programu na vývojovou desku, který provedeme stisknutím tlačítka `Upload`. Tlačítko se nachází v horní části vývojového prostředí. Před zavedením programu je vhodné si otevřít sériový monitor a zkontrolovat, zda komunikace mezi PC a vývojovou deskou probíhá správně.

Nejdříve dojde k sestavení samotného programu. Následuje zavedení programu na vývojovou desku, kde ve spodní části aplikace Arduino IDE je možné si zobrazit terminál, na kterém můžeme sledovat aktuální stav sestavení i zavádění programu. V případě, že nastanou nějaké komplikace, ať už s přiřazením některé z knihoven při sestavování programu nebo chyba v komunikaci při zavádění, tak díky terminálu jsme schopni identifikovat vzniklou chybu.



Obrázek 7.2: Vývojové prostředí Arduino IDE

[1] Flash Mode určuje kolik pinů je použito pro přístup k FLASH paměti. Režim `QIO` je nejrychlejší možný režim práce s pamětí, který vývojová deska podporuje. [59]

7.3 ESP-IDF (Espressif IoT Development Framework)

Testováno na MacBook PRO 2019
ESP-IDF je vývojové prostředí od společnosti Expresif, které slouží k zavedení programu do vývojové desky (v našem případe ESP32 - Ai-Thinker nebo ESP8266).

7.3.1 Instalace - verze 3.3.1

Nejprve je potřeba instalovat Python verzi 2.7 (řádek 1), dále pokračujeme instalací modulu pro přístup k sériovému portu (řádek 2). Následuje instalace souborového manageru, jako je např. **Homebrew**, který je dostupný z `https://brew.sh`. Jeho stažení a instalaci provedeme příkazem, který je na řádku 3. Dále je potřebné nainstalovat **cmake Ninja** (řádek 4). Jedná se o alternativu klasického `make`, ale s rozdílem rychlosti kompilace.

Pokračujeme vytvořením složky (řádek 5) **esp-idf** v domovském adresáři, do které následně stáhneme z gitu soubory potřebné jak pro běh, tak instalaci. Po stažení souborů z gitu následuje přechod do dané složky (řádek 6) a spuštění instalačního scriptu (řádek 7). Poslední nezbytný úkon před samotným testováním je přidání cesty `PATH` (řádek 8).

```
1 sudo easy_install pip
2 pip install --user pyserial
3 /usr/bin/ruby -e "$(curl -fsSL
    https://raw.githubusercontent.com/Homebrew/install/master/install)"
4 brew install cmake ninja
5 mkdir esp
6 git clone --recursive https://github.com/espressif/esp-idf.git
7 cd ~/esp/esp-idf
8 ./install.sh
9 . $HOME/esp/esp-idf/export.sh
```

7.3.2 Spuštění prvního programu ESP32-CAM

Velkou výhodou je, že stažený balík již obsahuje ukázkové programy. Pro spuštění některého z ukázkových programů nejprve musíme přejít do složky `esp` (řádek 1). Následuje překopírování námi zvoleného ukázkového programu ze složky `examples` do složky `esp` (řádek 2). Dále přejdeme do složky, která má název námi zvoleného příkladu (v našem případě `hello_word`) (řádek 3). Pokračujeme spuštěním `menuconfig` (řádek 4), kde je nutné nastavit `flash frequency` na 80Mhz (výchozí hodnota je nastavena na 40 MHz). Dále pokračujeme sestavením (řádek 5) a zavedením programu (řádek 6) na vývojovou desku.

Již můžeme pokračovat spuštěním programu. Nejprve je potřeba zjistit na jakém portu je zařízení připojené. V případě macOS název sériového portu začíná `/dev/cu.` Sériový monitor spustíme příkazem na řádku 7 a za "XXXX"doplníme číslo sériového portu, následně již můžeme monitorovat datový tok na sériovém rozhraní. Pokud si přejeme spustit sériový monitor již při samotném zavádění programu, učiníme tak příkazem `idf.py -p /dev/cu.usbserial-XXXX flash monitor` .

Ukončení sériového monitoru provedeme klávesovou zkratkou **control+)**.

```
1    cd ~/esp
2    cp -r $IDF_PATH/examples/get-started/hello_world .
3    cd ~/esp/hello_world
4    idf.py menuconfig
5    idf.py build
6    idf.py
7    idf.py -p /dev/cu.usbserial-XXXX monitor
```

7.4 Možnosti zavedení programu

K zavedení programu (jeho binární podoby) do paměti vývojové desky můžeme přistupovat dvěma základními principy.

- **Bezdrátově** (Over The Air - OTA[36]) s využitím WiFi - Tento způsob nahrání nového programu se zdá být atraktivnější. Velkou výhodou je možnost automatického nahrání do více modulů, ke kterým nemusíme mít fyzický přístup (stačí pouze připojení ke stejné WiFi síti). Pokud se rozhodneme využít programování přes Wi-Fi, máme více možností, jak postupovat. Můžeme využít vývojové prostředí Arduino IDE nebo webový prohlížeč. Tento způsob programování přináší i několik nevýhod oproti použití USB kabelu. Pokud při přenosu dojde k nějakému problému může nastat situace, že se k desce již nedokážeme pomocí WiFi připojit a bude nutné nahrát fireware pomocí kabelu. Další téměř zanedbatelnou nevýhodou je, že každý program musí obsahovat část kódu, která se postará o následující programování desky s využitím OTA. Dále jsme limitováni maximální velikostí přenášeného programu, která je 2Mb.

- **Drátově** - ESP32S a ESP12F nepodporují možnost přímého programování pomocí USB. Pro zavedení programu musíme využít UART [65]. Jedná se o zařízení, které slouží k sériové komunikaci mezi PC a vývojovou deskou. V našem případě využijeme FTDI převodník [71] (USB -> UART). Hlavní výhoda použití této metody zavedení programu do paměti spočívá v možnosti monitorování přenosu dat (komunikace je obousměrná), tudíž pokud nastane nějaký problém, jsme schopni ho jednoznačně a okamžitě identifikovat.

8 Instalace OpenHAB v3.1.0

Platformu OpenHAB můžeme provozovat na většině operačních systémů jako je např. Windows, Linux (Raspbian) nebo macOS. Pro běh aplikace je možné využít od jednodeskového počítače s Linuxem (např. Raspberry Pi) až po MacBook PRO.

Při instalaci platformy OpenHAB na Raspberry Pi můžeme zvolit dva odlišné postupy:

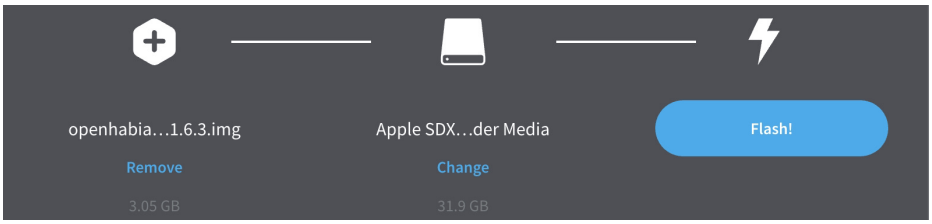
- Stažení již předpřipraveného operačního systému s OpenHAB
- Manuální instalace - instalace Raspbianu + instalace OpenHAB

8.1 Instalace OpenHAB v1.6.3

V následujících několika bodech bude popsána instalace automatizační platformy OpenHAB v1.6.3 na Raspberry PI v.4 8GB a zprovoznění všech komponent, které budou použity k interakci mezi jednotlivými zařízeními a cetrálním prvkem.

Jak již bylo výše zmíněno, k usnadnění instalace je možno využit předpřipraveného operačního systému (openHABian v1.6.3), který je nejprve nutné stáhnout z: https://github.com/openhab/openhabian/releases/tag/v1.6.3.

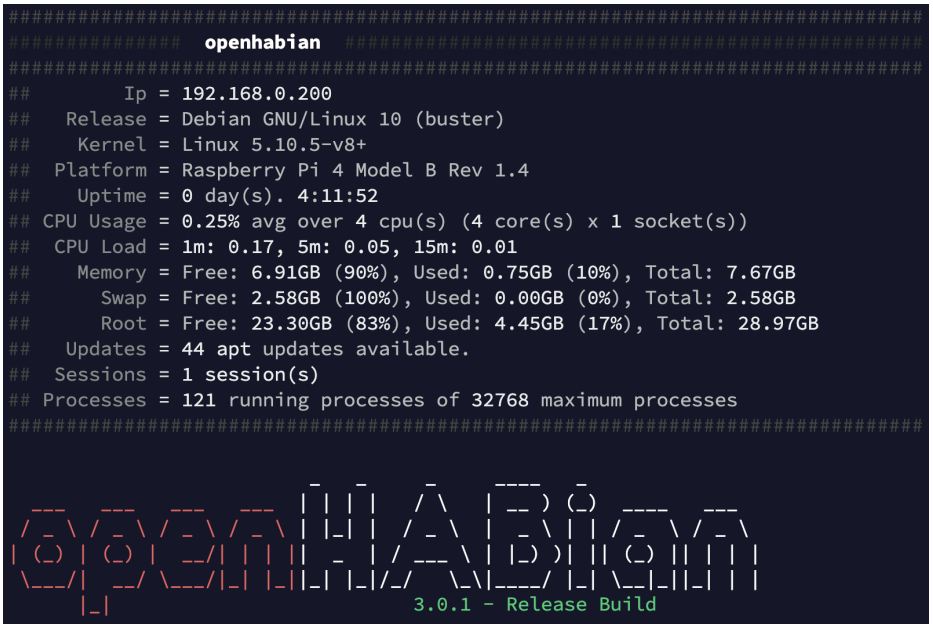
Pro samotný operační systém je vhodné zvolit paměťovou kartu o minimální velikosti 16GB. K zápisu obrazu na paměťovou kartu můžeme využít např. program balenaEtcher[14]. Na obrázku 8.1 je zobrazeno uživatelské prostředí tohoto programu.



Obrázek 8.1: Program balenaEtcher

V případě úspěšného zavedení obrazu na paměťovou kartu je vhodná editace konfiguračního souboru `openhabian.conf`, kde je možné nastavit např. přístupové údaje k `Wi-Fi`. Tento soubor se nachází v kořenové složce paměťové karty, na kterou byl zaveden obraz. Editaci je nutné provést před prvním startem systému.

Posledním krokem je vložení paměťové karty do `Raspberry Pi`. Pro první start systému je žádoucí použít kvalitní napájecí zdroj `5V 3A`. Pokud použijeme zdroj s nedostatečným výkonem, může dojít k poškození dat uložených na paměťové kartě. Průběh instalace je možné sledovat prostřednictvím webového prohlížeče (zadáním IP adresy zařízení), případně připojením `Raspberry PI` pomocí `HDMI kabelu` k monitoru. První spuštění trvá několik minut (cca 15 - 20 min).



Obrázek 8.2: Uvítací výpis OpenHAB

Nyní se můžeme připojit k automatizační platformě pomocí `ssh protokolu`. Při prvním připojení jsou přihlašovací údaje (jméno i heslo) nastaveny na `openhabian`. Pokud vše proběhlo v pořádku, zobrazí se uvítací výpis, který je zobrazen na obrázku 8.2. Případně je možné k nastavení a ovládání automatizační platformy využít grafické uživatelské prostředí, které je dostupné prostřednictvím webového prohlížeče na adrese `http://openhabian:8080`, případně `http://IP_ADDRESS:8080`.

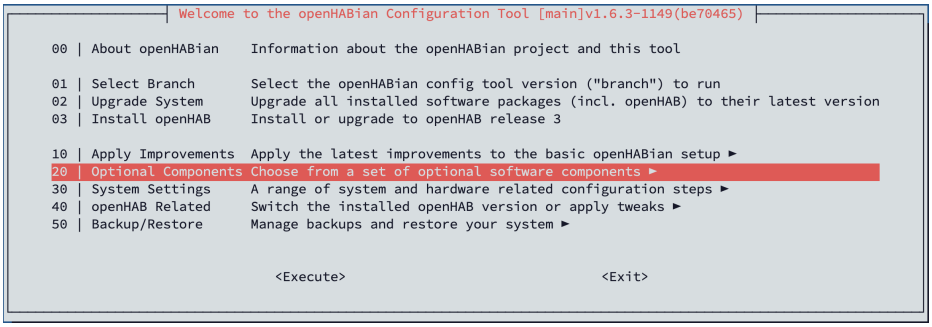
8.2 Eclipse Mosquitto

`Eclipse Mosquitto[26]` je `MQTT Broker`, který je volně šiřitelný pod licencí `open source`. Výhoda tohoto software spočívá v možnosti jeho provozování na široké paletě zařízení, od jednodeskových počítačů jako je např. `Raspberry PI` až po výkonné servery.

Tento software implementuje `MQTT protokol`, který je využit pro předávání zpráv mezi jednotlivými zařízeními chytré domácnosti.

8.2.1 Instalace Eclipse Mosquitto

Jeden z možných postupů instalace `MQTT Brokeru` je využití grafického uživatelského prostředí, ale v tomto případě je instalace s využitím příkazového řádku rychlejší a intuitivnější. Nejprve je nutné se pomocí `SSH protokolu` připojit k platformě `OpenHAB`. Následuje zobrazení konfiguračního dialogového okna, které provedeme zadáním příkazu `sudo openhabian-config`, heslo pro `root uživatele` je `openhabian` (v případě, že nebylo změněno), na obrázku 8.3 je výše zmíněné konfigurační okno zobrazeno.



Obrázek 8.3: OpenHABian konfugirační okno

Nejprve zvolíme položku `Optional Components`, následuje výběr přidávaných komponentů. V našem případě `Mosquitto` a potvrzení instalace (výběr možnosti `Continue`). Po úspěšné instalaci jsme vyzváni k zadání hesla, které slouží pro připojení k `MQTT brokeru`. Uživatelské jméno je defaultně nastaveno na `openhabian`. Posledním nezbytným krokem, než bude možné `Mosquitto` používat, je restart zařízení příkazem `sudo reboot`.

8.2.2 Testování Eclipse Mosquitto

Správná funkčnost `MQTT Brokeru` je klíčová pro komunikaci jednotlivých senzorů s centrální jednotkou. Pro ověření funkčnosti se připojíme pomocí

SSH Protokolu k centrální jednotce Raspberry Pi a s využítím příkazu mosquitto_sub otestujeme odběr všech nebo jen určitého tématu.

Na následujícím kódovém bloku je zobrazeno možné použití příkazu (řádek 1 - vykonávaný příkaz, řádky 2-4 - přijaté zprávy pomocí MQTT protokolu). Při použití parametru -v je na prvním místě vypsán štítek, pod kterým byla zpráva doručena. Nastavení parametru -t na hodnotu # znamená, že budou vypsány všechny zprávy přijaté pod libovolným štítkem. Parametr -u slouží k nastavení uživatele a volbou parametru -pw se nastavuje heslo uživatele.

```
1 openhabian@openhabian:~ $ mosquitto_sub -u openhabian --pw
    openhabian -v -t "#"
2 test_topic_1 test_message_1
3 test_topic_2 test_message_2
4 test_topic_3 test_message_3
```

K publikování zpráv je možné využití jiného terminálového okna (sysetému OpenHAB), případně se nabízí možnost použít jiné zařízení k publikaci zpráv. Na následujícím kódovém bloku je použit příkaz mosquitto_pub, který slouží k publikaci zpráv pomocí MQTT protokolu s využitím Mosquitto brokeru.

V příkazu je použit parametr -p, který určuje port, na kterém je broker provozován, parametr -h nastavuje ip adresu brokeru, parametr -t značí štítek, pod kterým je zpráva přenášena a parametr -m slouží k nastavení samotné přenášené zprávy.

```
1 mosquitto_pub -u openhabian --pw openhabian -p 1883 -h
    192.168.0.200 -t "test_topic_1" -m test_message_1
```

8.3 MQTT Binding

Dalším zásadním krokem je nastavení interakce mezi automatizační platformou OpenHAB a komunikací probíhající pomocí MQTT protokolu.

K dosažení výše zmíněné součinnosti slouží doplněk MQTT Binding[46], jehož přidání do systému OpenHAB provedeme následujícím způsobem:

1. Pomocí webového prohlížeče přejdeme na adresu http://OPENHAB_IP:8080, kde OPENHAB_IP nahradíme IP adresou zařízení, na kterém je automatizační platforma provozována.

2. V záložce Nastavení zvolíme položku Things a klikneme v pravém dolním rohu na tlačítko +.

3. Potvrdíme Install Bindings a v rozbalovacím seznamu vyhledáme a nainstalujeme MQTT Binding.

Testování doplňku Mqtt binding provedeme následujícím způsobem:

1. Připojíme se pomocí SSH protokolu k automatizační platformě

2. Přejdeme do složky /etc/openhab/things ve které vytvoříme soubor s příponou .things (soubor je možné libovolně pojemenovat, při zachování pravidel pojmenovávání souborů), vytvoříme např. soubor default.things.

3. Ve vytvořeném souboru realizujeme jakýsi most (bridge), který slouží k propojení komunikace pomocí MQTT protokolu a automatizační platfomy OpenHAB. Na následujícím kódovém bloku je ukázána definice takového mostu. První řádek představuje vytvoření MQTT brokeru Mosquitto, 2 řádek IP adresu zařízení na kterém je MQTT broker provozován, na 3 řádku je definován port a na řádku 4-5 jsou definovány přístupové údaje k MQTT brokeru.

```
1    Bridge mqtt:broker:mosquitto [
2    host="localhost",
3    port=1883,
4    username="openhabian",
5    password="openhabian" ]{
6
7  }
```

Ověření, zda byl most správně vytvořen, můžeme provést pomocí webového prohlížeče. Na hlavní stránce (http://OPENHAB_IP:8080) zvolíme položku Nastavení, kde vybereme Things. Nyní můžeme provést kontrolu, že mqtt:broker:mosquitto se nachází ve stavu ONLINE 8.4. Pokud je MQTT Broker ve stavu OFFLINE, je nutné přistoupit k hledání příčiny problému. Jako možná příčina se jeví zadání chybné IP adresy zařízení. V tomto případě je vypsána chybová hláška COMMUNICATION_ERROR nebo chybně zadané přístupové údaje.

Obrázek 8.4: MQTT Broker - online stav

8.3.1 Generic MQTT Things

Jedná se o obejkt typu `Thing`, který umožnuje přidání několika obecných objektů různých datových typů, jako je např. `number` nebo `switch`. Pomocí kanálů `(channels)` je možné realizovat integraci jednotlivých položek do systému, např. propojení určité položky s konkrétním štítkem označujícím zprávu přenášenou pomocí `MQTT` protokolu. U každé položky může být definován libovolný počet kanálů.

Na následujícím kódovém bloku je ukázka definice `MQTT Brokeru` (řádky 1-5), uvnitř kterého je definována položka typu `Generic MQTT Thing` (řádek 6), která je označena popiskem `Entry system` a její umístění je v místě `Entrance gate`. V položce typu `Generic MQTT Thing` s názvem `entry_system` je definován konkrétní objekt (řádek 7). Objekt je typu `number`, označen je identifikátorem `temperature` a je mu nastavený parametr `stateTopic` na hodnotu `/gate/temp`. Tento parametr označuje štítek pro přiřazení zpráv, které jsou zpracovávané pomocí `MQTT protokolu` - tzn. položce `temperature` jsou přižazeny zprávy se štítkem `/gate/temp`.

Veškerá konfigurace probíhá v souboru s příponou `.thigns`, v našem případě se jedná o soubor `default.things`.

```
1  Bridge mqtt:broker:mosquitto [
2  host="localhost",
3  port=1883,
4  username="openhabian",
5  password="openhabian" ]{
6    Thing topic entry_system "Entry system" @ "Entrance gate" {
7      Type number : temperature [stateTopic="/gate/temp"]
8    }
9  }
```

V tomto okamžiku je možné z kódu získat kanály `(channels)` jednotlivých položek definovaných v objektu `entry_system`. Kanál vychází z posloupnosti názvů jednotlivých nadřazených objektů.

V případě položky `temperature` bude přiřazen následující kanál:

```
1  mqtt:topic:mosquitto:entry_system:temperature
```

8.4 Telegram Binding

Toto rozšíření slouží k odesílání a přijímání zpráv pomocí aplikace `Telegram`. Přidání doplňku je realizováno stejným způsobem jako tomu bylo u `MQTT Binding`, pouze volíme doplněk `Telegram Binding`.

Po instalaci tohoto doplňku již další veškerá konfigurace probíhá prostřednictvím konfiguračního souboru (v našem případě `default.things`). V následujícím kódovém bloku je zobrazeno použití tohoto doplňku - vytvoření objektu typu `Thing`. Údaje `ID a TOKEN` získáme dle níže uvedeného postupu.

```
1 Thing telegram:telegramBot:Telegram_Bot [ chatIds="ID",
    botToken="TOKEN" ]
```

Aby bylo možné domácnost ovládat pomocí aplikace `Telegram`, je nutné vytvoření komunikačního automatu tzv. `Telegram Bot`.

Nejprve je nutné stažení aplikace `Telegram`. Tato aplikace je dostupná pro širokou škálu zařízení od mobilního telefonu s operačním systémem `Android` nebo `iOS` až po stolní počítač s `Widows`. Po stažení pokračujeme s registrací a vytvořením uživatelského účtu.

Vytvoření komunikačního automatu `Telegram bot`:

1. Otevřeme záložku `kontakty`, vyhledáme účet `BotFather`
2. Klikneme na tlačítko `Start`, případně odešleme příkaz `\start`
3. Zadáme příkaz `\newbot`
4. Vložíme název a uživatelské jméno. Pokud se vše podařilo, přijde odpověď, ve které je uveden `HTTP API TOKEN`, který slouží k přístupu k automatu.
5. Otevřeme záložku `kontakty`, vyhledáme účet `IDBot`
6. Zadáme příkaz `\getid`, získáme identifikátor (`ID`) mobilního zařízení, z tohoto zařízení mohou být odesílány příkazy.

Kontrolu, zda se přidání a integrace doplňku `Telegram Binding` podařila, provedeme pomocí webového prohlížeče. V sekci `Nastavení` volíme položku `Things` a následně bychom měli vidět, že se `telegram:telegramBot:Telegram_Bot` nachízí ve stavu `ONLINE` 8.5.

Obrázek 8.5: Telegram - online stav

Na následujícím příkladu je zobrazena jedna z možností použití tohoto doplňku. V případě, že položka `motion_sensor` změní svůj stav na `ON` dojde k odeslání zprávy pomocí `Telegramu`.

```
rule "Telegram"
when
  Item motion_sensor changed to ON
then
  val telegramAction =
      getActions("telegram","telegram:telegramBot:Telegram_Bot")
  telegramAction.sendTelegram("Motion detect")
end
```

8.5 Exec Binding

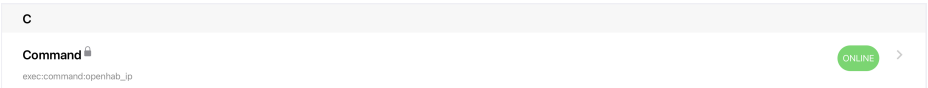
Další velmi užitečný doplněk je `Exec Binding`, který slouží k vykonávání `shell příkazů`. Jeho instalaci provedeme stejným způsobem jako u dvou výše zmíněných doplňků. Při instalaci volíme doplněk `Exec Binding`.

Tento doplněk opět použijeme v souboru `default.things` uvnitř mostu `mqtt:broker:mosquitto`. Použití je realizováno následujícím způsobem. Je zde definován příkaz `command`, interval po kterém bude opakováno volání příkazu `interval` a parametr `timeout` značí čas na vykonání příkazu.

```
Thing exec:command:openhab_ip [command="hostname -I",
    interval=60000, timeout=5]
```

Pro vykonání příkazu je nezbytné jeho vložení do souboru `exec.whitelist`, který se nachází ve složce `/etc/openhab/misc`.

Pokud vše proběhlo správně, tak vidíme v seznamu aktivních objektů typu `Things exec:command:openhab_ip`, že se nachází ve stavu `ONLINE` 8.6.



Obrázek 8.6: Exec binding - online stav

8.6 HTTP Binding

Jedná se o rozšíření, které umožnuje zaslání HTTP requests. Instalace tohoto rozšíření je realizována stejným způsobem jako tomu bylo u výše zmíněných doplňků např. (MQTT Binding), pouze s rozdílem, že instalujeme rozšíření HTTP Binding.

Na následujícím kódovém bloku je ukázána možnost využití tohoto doplňku ke spuštění PhP skriptu.

```
1 rule "PHP action"
2 when
3   Item add_user received update
4 then
5   var String ip = openhab_ip.state.toString.replace(' ', '')
6   var String add_user_url = "http://"+ip+"/add_user.php
7   sendHttpGetRequest(add_user_url)
8 end
```

8.7 OpenHAB Cloud Connector

Jedná se o nástroj[50], který slouží k bezpečnému vzdálenému přístupu k OpenHAB serveru bez potřeby veřejné IP adresy nebo použití VPN.

Před samotným používáním OpenHAB Cloudu je zásadním krokem přidaní doplňku openHAB Cloud Connector, které provedeme pomocí webového rozhraní. Nejprve se příhlásíme jako administrátor systému a přejdeme do záložky Settings, zvolíme položku MISC a následně klikneme v pravém dolním rohu na tlačítko +, nakonec vyhledáme a potvrdíme instalaci doplňku openHAB Cloud Connector

Před samotnou registrací na https://myopenhab.org/login potřebujeme znát jedinečný identifikátor UUID (obdoba přihlašovacího jména) a tajný kód Secret. V následující tabulce se nachází umístění přístupových údajů na OpenHAB serveru.

File	Regular Installation
UUID	userdata/uuid
Secret	userdata/openhabcloud/secret

File	APT Installation
UUID	/var/lib/openhab/uuid
Secret	/var/lib/openhab/openhabcloud/secret

Po dokončení registrace restartujeme zařízení, na kterém je provozován OpenHAB server a přejdeme na webovou stránku OpenHAB Cloudu https://myopenhab.org, kde v případě úspěšné registrace a správného zadání přístupových údajů (UUID a Secret) uvidíme, že se náš server nachází ve stavu Online. OpenHAB server nyní můžeme ovládat i přes Cloud na webové stránce https://home.myopenhab.org/ nebo pomocí mobilní aplikace - OpenHAB.

8.8 HomeKit

Doplněk HomeKit slouží k ovládání platformy OpenHAB pomocí HomeKit protokolu. Díky integraci tohoto doplňku je možné sledování a ovládání chytré domácnosti s využitím zažízeních od firmy Apple, samozřejmností je využití hlasového asistenta Siri.

Instalace doplňku je shodná, jako v případě OpenHAB Cloud Connector, pouze s rozdílem, že pro instalaci volíme doplněk HomeKit Integration.

8.9 Persistence

Tato vlastnost slouží k ukládání dat při běhu aplikace. Pomocí těchto dat je možné obnovit stav systému po restartu zařízení. V systému OpenHABian je již přidán doplněk RRD4j Persistence. Jediný nutný krok před jeho použitím je nastavit tento typ databáze jako defaulní, což provedeme v záložce Nastavení, kde rozevřeme položku Persistence (v bloku System Services) a následně nastavíme službu RRD4j jako defaultní a uložíme tlačítkem Save.

Veškerou další konfiguraci je možné realizovat pomocí konfiguračního souboru rrd4j.persist, který vytvoříme ve složce /etc/openhab/persistence. Na následujícím kódovém bloku je zobrazen příklad, kdy bude každá změna u všech položek ukládána a po restartu zařízení bude stav položky obnoven.

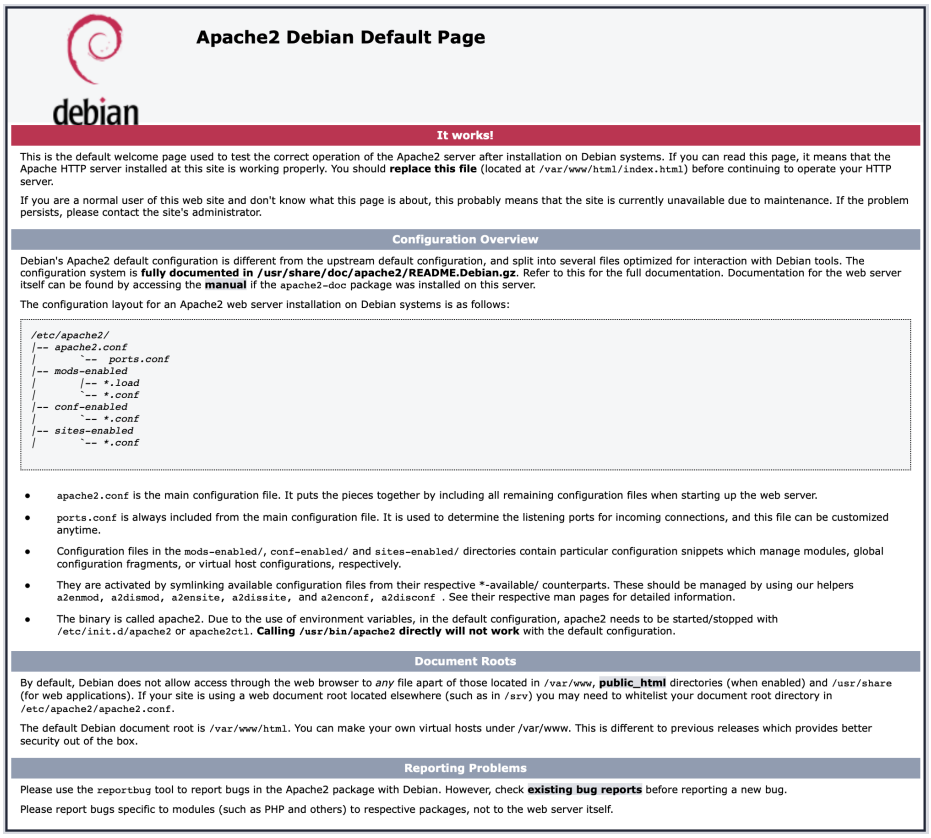
```
1 Strategies {
2   default = everyUpdate
3 }
4 Items {
5    * : strategy = everyChange, restoreOnStartup
6 }
```

8.10 Apache

Jedná se o jeden z nejrozšířenějších webových serverů, který slouží pro zobrazení webových stránek definovaných pomocí `HTML`, případně zpracování `PHP skriptů`. Instalaci provedeme z příkazového řádku zadáním následujícího příkazu:

```
1 sudo apt install apache2 -y
```

Po dokončení instalační procedury můžeme otestovat funkčnost `Apache serveru` zadáním `IP adresy` zařízení, na kterém byl server instalován. V případě úspěsné instalace se zobrazí uvítací stránka 8.7. Pokuď není známa `IP adresa` zařízení, lze ji zjistit zadáním příkazu `hostname -I` do příkazového řádku.



Obrázek 8.7: Uvítací stránka Apache

Dokončení konfigurace provedeme změnou složky, ve které následně budou uloženy `HTML a PHP sobory`. Ve složce `/etc/apache2/sites-available` v souboru `000-default.conf` nastavíme hodnotu `DocumentRoot` na `/etc/openhab/html`. Dále ve složce `/etc/apache2` přidáme do souboru `apache2.conf`

následující kód, který zpřístupní složku `/etc/openhab/html` pro webový server `Apache 2`.

```
<Directory /etc/openhab/html>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
```

Server `Apache` sám o sobě nedokáže zpracovávat `PHP sobory`, pro jejich zpracování je nutné přidání modulu `PHP apache2`, který nainstalujeme příkazem:

```
sudo apt install php libapache2-mod-php -y
```

Pro ověření funkčnosti výše zmíněných nastavení si můžeme vytvořit jednoduchý `PHP` soubor, který uložíme do složky `/etc/openhab/html` a pojmenujeme ho např. `test.php`. Na následujícím kódovém bloku je zobrazen obsah tohoto souboru. Následně pomocí webového prohlížeče zobrazíme stránku na adrese `http://IP_ADDRESS/text.php` a pokud veškeré předchozí nastavení proběhlo úspěšně, uvidíme na obrazovce text `PHP TEST`.

```
<?php
  echo '<H1>PHP TEST</H1>';
?>
```

Pro správnou funkci je zásadním krokem nastavení patřičných přístupových práv jednotlivým souborům. Nastavení provedeme pomocí příkazu `chmod`.

8.11 MaryTTS

Toto rozšíření slouží k převodu textu na řeč. Instalaci provedeme v záložce `Nastavení` v záložce `Voice` a následnou instalací `Mary Text-to-Speech`.

8.12 MPG321

Jedná se o přehrávač zvukových souborů, který je následně použit např. při přehrání melodie při stistku zvonkového tlačítka. Instalaci tohoto přehrávače provedeme příkazem:

```
sudo apt-get -y install mpg321
```

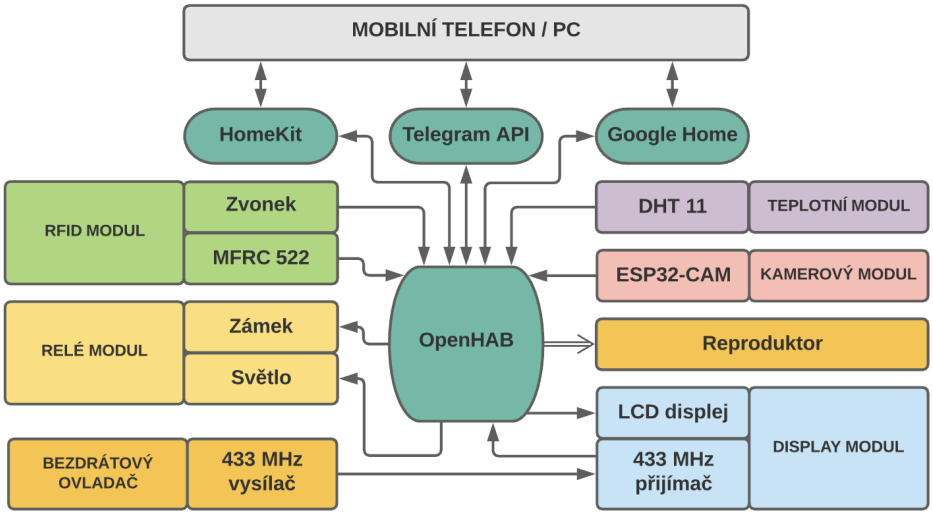
9 Implementace systému domácí automatizace

V tomto příkladu je demonstrována funkčnost vstupního systému u zahradní branky.

Použité senzory:

- RFID čtečka - `MCRF-522`
- Zvonek - `mikrospínač`
- Pohybový senzor - `ESP32-CAM Ai-Thinker`
- Kamerový modul - `ESP32-CAM Ai-Thinker`
- Spínač otevírání branky - `Relé modul`
- LCD display
- 433 MHz přijímač
- Senzor teploty a vlhkosti - `DHT11`

Na obrázku 9.1 je zobrazeno blokové schéma vstupního systému. Zprávy mezi jednotlivými senzory a centrálním prvkem jsou předávány pomocí `MQTT protokolu`.



Obrázek 9.1: Schéma komunikace - vstupní systém

9.1 Základní popis funkčnosti

Vstupní systém je složen z několika senzorů a výstupních zařízení, které reagují na příchozí podněty. Ovládání systému je možné jak z mobilního telefonu, tak z počítače.

V systému je integrován `kamerový modul`, který slouží jak k pořízení fotografie k vyhodnocení situace před brankou, tak zároveň i jako `pohybový senzor`. Při zaznamenání pohybu je možné odeslat snímek, který zachycuje situaci před brankou uživatelům domácnosti na mobilní telefon např. prostřednictvím aplikace `Telegram`, případně přehrát nějaký zvuk na reproduktoru.

Nechybí zde ani klasické zvonkové tlačítko. Při stisku zvonkového tlačítka dojde k přehrání melodie a následnému odeslání fotografie aktuální situace před brankou.

Hlavním výstupním prvkem systému je `dveřní zámek`, který je ovládán z centrální jednotky. Stav zámku je možné sledovat a ovládat hned několika způsoby:

- Přiložení RFID karty.
- Pomocí mobilní aplikace nebo hlasového asistenta.
- Pomocí vysílače pracujícího na frekvenci 433 MHz.

Jako další výstupní prvek je integrován `LCD displej`, který slouží k zobrazení libovolných textových informací. Pro sledování meteorologické situace je zde umístěn teplotní a vlhkostní senzor.

10 Knihovna pro Wi-Fi senzor

V této kapitole je hlavním cílem návrh a implementace knihovny pro Wi-Fi senzor, která následně usnadní implementační část jednotlivých senzorů, jako je např. `senzor teploty` nebo `kamerový modul`. Mezi hlavní požadavky na knihovnu patří:

- Použitelnost na `ESP32-CAM` a zároveň na `ESP12F`.
- Navázání připojení k `Wi-Fi`. Při neúspěšném připojení, nebo nedostupnosti dat vytvoření přístupového bodu, pomocí kterého bude možné nastavení přihlašovacích údajů k `Wi-Fi`.
- Kompletní obsluha webového rozhraní.
- Navázání a následná obsluha přenosu zpráv pomocí `MQTT protokolu`.
- Nastavení komunikace (např. `IP adresa MQTT Brokeru`) s centrální jednotkou pomocí webového rozhraní (kapitola 10.3).
- Zavední programu prostřednictvím `Wi-Fi (OTA)` (kapitola 10.4).

10.1 Funkce knihovny

Hlavní třída této knihovny je `Sensor.cpp`. Při vytváření nové instance této třídy je nejprve volán konstruktor, který se postará o vytvoření ukazatele na `AsyncWebServer`.

K aktivaci knihovny slouží metoda `begin()`, která nejprve načte nastavení z virutuální `EEPROM` paměti voláním metody `EEPROM.begin(SIZE)`, kde vstupním parametrem je velikost paměti, o kterou program žádá. Následuje získání dat uložených v této virtuální `EEPROM paměti`, která jsou přečtena pomocí metody `read_eeprom()`. Tato metoda načtená data uloží do struktury `boot_data`, která je zobrazena na následujícím kódovém bloku.

```
1 struct BOOT_DATA{
2   char wifi_ssid[WIFI_SSID_ARRAY_SIZE];
3   char wifi_passwd[WIFI_PASSWORD_ARRAY_SIZE];
4   char broker_ip_add[BROKER_IP_ADD_ARRAY_SIZE];
5   int  broker_port;
6   char mqtt_nick[MQTT_NICK_ARRAY_SIZE];
7   char mqtt_name[MQTT_NAME_ARRAY_SIZE];
8   char mqtt_passwd[MQTT_PASSWORD_ARRAY_SIZE];
9 };
```

Dalším krokem je pokus o navázání připojení k `Wi-Fi`, o které se postará metoda `wifi_connection_init()`, která nejprve ověří, zda jsou dostupné přihlašovací údaje ve struktuře `boot_data`. Pokud jsou data dostupná, dojde k volání metody `wifi_connect`, která se z dostupných dat pokusí navázat připojení k `Wi-Fi`. V případě neúspěšného připojení (ve struktuře je např. nesprávné heslo k `Wi-Fi`), nebo v případě, že struktura `boot_data` neobsahovala žádné přihlašovací údaje, dojde k opětovnému volání metody `wifi_connect` s rozdílem, že budou použity defaulní přístupové údaje definované v souboru `boot_info.h`. V případě selhání obou výše zmíněných pokusů dojde k vytvoření přístupového bodu, pomocí něhož je možné se k senzoru připojit na nastavit přístupové údaje. Následně dojde k inicalizaci webového serveru, o kterou se postará metoda `web_server_init()`.

V aktuálním okamžiku je buď dostupné připojení k `Wi-Fi`, nebo je vytvořený přístupový bod. Pokud je dostupné `Wi-Fi` připojení, je volána metoda `mqtt_init()`, která se postará o navázání připojení k `MQTT Brokeru`.

V opačném případě program čeká, než uživatel změní přístupové údaje k `Wi-Fi`. Pokud ke změně nedošlo do předem definovaného časového okamžiku a k přístupovému bodu není připojen žádný klient, dojde k opětovnému pokusu o připojení k `Wi-Fi` z dostupných údajů - v okamžiku prvních

pokusů o připojení mohlo dojít k výpadku sítě, ale údaje byly správné.

V metodě `loop()`, která je cyklicky vykonávána, je volána metoda `connection_check()`, která kontroluje stav připojení k `Wi-Fi` a k `MQTT Brokeru`. V případě nedostupnosti `Wi-Fi` připojení je volána metoda `wifi_connection_init()`. Pokud je `Wi-Fi` připojení aktivní, ale `MQTT Broker` je nedostupný, je volána metoda `mqtt_init()`, která se pokusí opětovně navázat připojení.

Dále jsou v této třídě definované metody na obsluhu a nastavení `MQTT spojení`.

Metoda `mqtt_callback(char* topic, byte* payload, unsigned int length)` slouží ke zpracování příchozích zpráv pomocí `MQTT protokolu`. Po přijetí a zpracování zprávy je volána `virtuální metoda process_message(char* topic, char* message)`, která je již definována v závislosti na konkrétním senzoru.

O přídání štítků pro odběr zpráv přenášených s využitím `MQTT protokolu` se postará metoda `add_sub_topic(char* topic)`, která štítek uloží do pole (z důvodu dostupnosti štítků po výpadku připojení k `MQTT Brokeru`) a následně metoda `set_mqtt_topic()` nastaví štítky pro odběr u instance třídy `PubSubClient`. Pro zrušení odběru všech štítků slouží metoda `mqtt_unsubscribe()`.

Pro publikaci dat s využitím `MQTT protokolu` je implementována metoda `mqtt_publish(char* topic, char* message)`, do které vstupuje štítek a samotná přenášená zpráva.

K usnadnění nastavení konfiguračních dat pro připojení k `Wi-Fi` nebo `MQTT Brokeru` byla implementována možnost nahrání konfiguračního souboru ve formátu `Json`, o jehož zpracování se postará metoda `read_file`. Je zde možnost načtení defaultních dat, která jsou předdefinována v souboru `boot_info.h`. K tomuto účelu slouží metoda `read_default_data()`.

V případě, že byla zadána data do formuláře ve webovém rozhranní a následně uložena stisknutím tlačítka `SAVE`, je volána metoda `save_form_data`, která získá data z formuláře a předá je na zpracování metodě `check_data`. V případě, že došlo ke změně libovolných dat, je volána metoda `save_struct()`, která se postará o uložení nových dat do virtuální paměti `EEPROM`. Pokud změna souvisí s daty významnými pro `Wi-Fi` připojení, je spuštěna procedura navázání nového připojení s aktuálními daty. V případě změny dat významných pro připojení k `MQTT Brokeru` dojde k vytvoření požadavku o volání metody `mqtt_init()`.

Pro práci s virtuální pamětí `EEPROM` slouží metody `save_struct()`, která se postará o uložení struktury `boot_data` do paměti a metoda `read_eeprom()` slouží k načtení dat uložených v paměti `EEPROM` do struktury `boot_data`.

10.2 Použité externí knihovny

Pro usnadnění implementace a zvýšení přehlednosti kódu byly použity následující knihovny:

- `<EEPROM.h>`[8] Knihovna pro práci s virtuální pamětí `EEPROM`.
- `<PubSubClient.h>`[42] Knihona pro práci s `MQTT protokolem`.
- `<ArduinoJson.h>`[15] Knihovna pro práci s formátem `Json`.
- `<ESPAsyncWebServer.h>`[21] Knihovna pro obsluhu `webového serveru`.
- `<Arduino.h>`[7] Knihovna pro práci s `Arduinem`.
- `<WiFi.h>`[70] Knihovna pro práci s `Wi-Fi (ESP32)`.
- `<ESP8266WiFi.h>`[9] Knihovna pro práci s `Wi-Fi (ESP8266)`.
- `"soc/soc.h"`[68] Knihovna pro práci s `brownout detector`.
- `"soc/rtc_cntl_reg.h"`[69] Knihovna pro práci s `brownout detector`.
- `<AsyncTCP.h>`[19] Knihovna pro práci s `TCP protokolem (ESP32)`.
- `<ESPAsyncTCP.h>`[20] Knihovna pro práci s `TCP protokolem (ESP8266)`.
- `<AsyncElegantOTA.h>`[63] Knihovna pro zavedení programu přes `Wi-Fi`.

10.3 Webové rozhraní

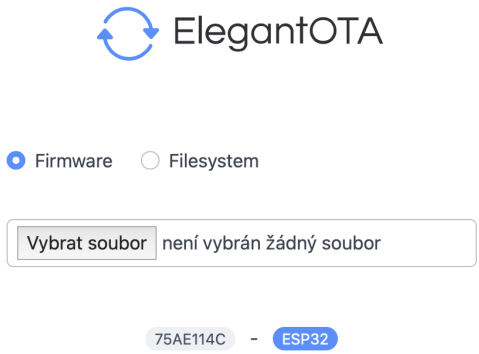
Natavení základních konfiguračních parametrů je možné pomocí nahrání konfiguračního souboru ve formátu `Json`. Takový soubor je zobrazen na následujícím kódovém bloku, případně je možná konfigurace s využitím webového rozhraní, které je zobrazeno na obrázku 10.1. U formuláře není zakryto heslo z důvodu kontroly nahrání dat konfiguračního souboru, po jehož nahrání dojde k vyplnění nastavených dat do formuláře.

```
1 "WIFI_SSID":"host",
2 "WIFI_PASSWORD":"navsteva",
3 "BROKER_IP_ADD":"broker.emqx.io",
4 "MQTT_NICK":"RFID_MODULE",
5 "MQTT_NAME":"openhabian",
6 "MQTT_PASSWORD":"openhabian",
7 "BROKER_PORT":1883
```

Obrázek 10.1: Konfigurační prostředí senzoru - kamerový modul

10.4 Vzdálené zavedení programu

Z důvodu, že senzor nemusí být vždy umísteň na uživatelsky přístupném místě, jsem se rozhodl použít externí knihovnu `AsyncElegantOTA`, která slouží pro vzdálené zavedení programu prostřednictvím `Wi-Fi`. Na obrázku 10.2 je zobrazeno grafické prostředí, které slouží pro zavedení programu. Tato funkce je dostupná jak na senzoru založeném na `ESP12-F`, tak i na `ESP32-CAM`.



Obrázek 10.2: AsyncElegantOTA

11 Implementace senzorů

V této kapitole bude popsána implementační část jednotlivých senzorů. Při implementaci je využita knihovna pro univerzální senzor, jejíž funkce je popsána v předchozí kapitole. Tato knihovna obsahuje několik `virtuálních metod`, které jsou ve všech níže zmíněných příkladech využity. Jedná se o medoty:

- `virtual void server_config()` - Tato metoda slouží k nastavení akce, která má být vykonána v případě, že bude zobrazena určitá webová stránka prostřednictvím webového rozhraní serveru např. při otevření adresy `http://SENSOR_IP/cam` bude volána metoda na vytvoření fotografie viz níže.

```
1  void Camera::server_config(){
2    server->on("/cam", HTTP_GET, [](AsyncWebServerRequest
         *request){
3        create_photo(request);
4    });
5  }
```

- `virtual void form_worker(AsyncWebServerRequest *request)` - Metoda slouží pro zpracování získaných dat z formuláře u konkrétního senzoru např. štítku, pod kterým budou publikována data o stisku zvonkového tlačítka. Vstupním parametrem je `request` pocházející z `AsyncWebServer`.
- `virtual void process_message(char* topic, char* message)` - Metoda slouží pro zpracování příchozích zpráv prostřednictvím `MQTT protokolu`. Na následujícím příkladu je zobrazeno publikování zpráv s využitím `MQTT protokolu`.

```
1    sensor->mqtt_publish(config_data.temperature_label,temp);
2    sensor->mqtt_publish(config_data.humidity_label,hum);
```

- `virtual void sensor_config(SensorJson json_file)` - Metoda slouží ke zpracování konfiguračních dat konkrétního senzoru, konfigurační data pochází ze souboru ve formátu typu `Json`, který uživatel nahrál prostřednictvím webového rozhraní.
- `virtual String generate_html()` - Metoda používaná pro gene-

rování HTML kódu konkrétního senzoru. Pro usnadnění generování HTML kódu vznikla třída Generator. Tato třída obsahuje metodu generate_label, která slouží ke generování HTML kódu pro formulář. Na následujícím kódovém bloku je ukázka způsobu generování HTML kódu pro konkrétní senzor. Do metody generate_label vstupují tři parametry:

1. parametr představuje identifikátor daného řádku formuláře.
2. parametr určuje popisek pro vstupní pole formuláře.
3. parametr udává hodnotu, která se má předvyplnit do vstupního pole.

```
1  String Temp_module::generate_html() {
2    String html_code = "<h1>SENSOR CONFIG</h1>";
3
4    html_code+= generator.generate_label(TEMPERATURE,
         TEMPERATURE_LABEL,
         String(config_data.temperature_label));
5    html_code+= generator.generate_label(HUMIDITY,
         HUMIDITY_LABEL, String(config_data.humidity_label));
6    html_code+= generator.generate_label(TEMPERATURE_PERIOD,
         TEMPERATURE_PERIOD_LABEL, String(config_data.period));
7
8    return html_code;
9  }
```

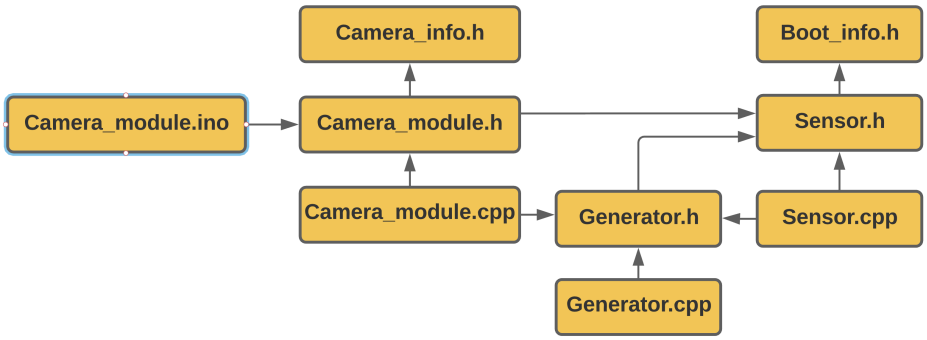
Na obrázku 11.1 je zobrazen výsledek generování HTML kódu, který je zmíněn výše.

Obrázek 11.1: Výsledek generování HTML kódu

Kód jednotlivých senzorů byl vytvořen pomocí programovacího jazyka C++. Všechny senzory komunikují prostřednictvím sériového portu, podávají informace o stavu připojení k Wi-Fi a k MQTT Brokeru.

11.1 Diagram závislostí tříd

Na obrázku 11.2 je zobrazen ukázkový příklad závislosti tříd u kamerového modulu.



Obrázek 11.2: Diagram závislosti tříd

11.2 Kamerový modul

Jako `Kamerový modul` slouží vývojová deska `ESP32-CAM`. Kamerový modul slouží jak k pořízení smímku, tak zároveň i jako `pohybový senzor`.

11.2.1 Popis funkce programu

`Kamerový modul` představuje výstupní prvek. Nejdříve dojde k načtení nastavení z paměti, následuje připojení k `Wi-Fi` a k `MQTT Brokeru`. Detekce pohybu je založena na porovnání dvou po sobě následujících snímků. Snímek je rozdělen na čtvercové bloky, ve kterých je vytvořen součet dat (snímky jsou pořizovány ve formátu `GRAYSCALE`, tudíž hodnota jednoho pixelu se pohybuje v rozmezí `0-255`). Detekce pohybu je realizována porovnáním vzájemně si odpovídajících bloků dvou po sobě následujících snímků. Bloky jsou považovány za shodné s určitou procentuální tolerancí, kterou je možné nastavit prostřednictvím wevobého rozhraní. Vyhodnocení, zda došlo k zaznamenání pohybu, je opět realizováno s nastavitelnou procentuální tolerancí tzn. počet bloků, který se může lišit, aby byl snímek považován za shodný.

Vždy je uchován poslední pořízený snímek při zaznamenání detekce pohybu, který je možno zobrazit prostřednictvím webového prohlížeče.

11.2.2 Implementace

Program je implementován v souboru `Camera_module.cpp`. Metoda `setup()` slouží k prvotnímu nastavení jako je např. zahájení funkce knihovny obecného senzoru a inicializace kamerového modulu.

Metoda `loop()`, která je cyklicky vykonávána nejprve testuje, zda je povelena činnost senzoru tzn. nacházíme se v režimu detekce pohybu. V režimu detekce je nejprve testováno, zda již uplynula perioda pro pořízení nového snímku a následnou detekci pohybu.

Detekce pohybu je realizována nejprve pořízením snímku a jeho následou separací do jednotlivých bloků, k čemuž slouží metoda `motion_capture()`. Velikost bloku a rozlišení snímku je definováno v souboru `camera_info.h`. V tomto případě je velikost snímku nastavena na rozlišení `320x240 pixelů` a velikost bloku na `10x10 pixelů`.

K vyhodnocení, zda došlo k pohybu slouží metoda `motion_detect()`, která v případě zaznamenání pohybu vrací hodnotu `true`. Po zaznamenání pohybu dojde k publikaci dat o zaznamenání pohybu.

Na následujícím kódovém bloku je zobrazena struktura pro uložení konfiguračních dat. Na `(řádku 2)` je štítek, pod kterým bude publikována zpráva `(řádek 3)` o zaznamenání pohybu. `(Řádek 4)` označuje maximální možnou procentuální odlišnot dvou bloků, do níž jsou bloky považovány za shodné. Dva smímky lze považovat za shodné (není zaznamenán pohyb) v případě, že je procentuální odchylka menší než hodnota definovaná na `(řádku 5)`. Na `(řádku 6)` je perioda během níž dochází k testování detekce pohybu.

```
1 struct CONFIG_DATA {
2         char motion_sensor_label[MQTT_LABEL_ARRAY_SIZE];
3         char motion_sensor_message[MQTT_LABEL_ARRAY_SIZE];
4         int motion_sensor_delta;
5         int motion_sensor_block;
6         int period_motion_check;
7 };
```

11.3 Teplotní senzor DTH11

Jedná se o digitální senzor teploty a vlhkosti, který komunikuje pomocí jednoho digitálního pinu. O chod a řízení komunikace senzoru se postará integrovaný 8bitový mikrokontrolér. Nasazení senzoru `DHT11` je vhodné do velmi široké škály zařízení od teplotního senzoru v místnosti až po senzor vlhkosti v klimatizaci.

Senzor vyžaduje napájecí napětí DC 3.3-5.5V, což dovoluje bezproblémovou komunikaci s mikrokontrolérem ESP12-F. Není nutné využití převodníku napěťových úrovní. Průměrná spotřeba elektrického proudu dosahuje 0.2 mA. Přesnost měření teploty je +- 2 °C (při 25 °C), při měření vlhkosti dosahuje přesnost měření -+5%RH (při 25 °C). Hodnoty o teplotě a o vlhkosti pochází vždy **z předchozího měření**. Pro získání dat v reálném čase je potřeba dvou měření.

11.3.1 Popis funkce programu

Teplotní modul je vstupní prvek. Prvním krokem je načtení dat z paměti, následuje připojení k Wi-Fi a MQTT Brokeru. Data o teplotě a vlhkosti jsou publikována pomocí MQTT protokolu v určitém časovém rozmezí. Pomocí webového prohlížeče je možné nastavit štítky pro publikaci dat teploty i vlhkosti. Dále je možné nastavit periodu (časové rozmezí, po kterém budou publikována data).

11.3.2 Implementace

Program pro tento modul je implementován v souboru Temp_module.cpp. Je zde obsažena metoda setup(), která slouží pro prvotní nastavení. Dále je zde obsažena metoda loop(), která je cyklicky vykonávána. V této metodě se dle periody měření (vždy za určitý čas) (řádek 5) volá metoda sensor_data(), která získá data o teplotě a vlhkosti ze senzoru. Následně jsou data publikována pomocí MQTT protokolu. Tato data jsou označena štítky, které jsou definované ve struktuře CONFIG_DATA, pro teplotu na řádku 3 a pro vlhkost na řádku 2.

```
1 struct CONFIG_DATA{
2         char humidity_label[MQTT_LABEL_ARRAY_SIZE];
3         char temperature_label[MQTT_LABEL_ARRAY_SIZE];
4         int period;
5 };
```

11.4 Relé modul

Ne každý spotřebič v domácnosti můžeme označit za „chytrý“. Většina základních spotřebičů nedovoluje možnost dálkového ovládání pomocí Wi-Fi. Pro integraci těchto spotřebičů do chytré domácnosti můžeme využít např. relé modulu, který připojíme do domácí sítě pomocí ESP-12F.

11.4.1 Popis funkce programu

V případě relé modulu se jedná o výstupní prvek. Nejprve je potřeba načtení nastavení z paměti. Dalším krokem je připojení k Wi-Fi síti a k MQTT Brokeru. V okamžiku úspěšného připojení k MQTT Brokeru dojde k nastavení štítků pro odběr dat.

Načtené nastavení se zobrazí v konfiguračním okně wevobého prohlížeče. Změna nastavení je možná jak způsobem editace, tak nahráním konfiguračního souboru ve formátu Json.

Po dokončení inicializační části programu následuje čekání na příchozí zprávy, u kterých je testováno, k jakému relé náleží (relé modul obsahuje více relé). Na základě obsahu přijaté zprávy je nastaven stav relé (aktivace nebo deaktivace).

11.4.2 Implementace

Hlavní třída tohoto modulu nese označení Relay_module.cpp. Je zde definována metoda setup(), která slouží k prvnotnímu nastavení voláním inicializační funkce relay_init(). Touto funkcí dojde k nastavení režimu pinů u ESP8266 jako výstupních. Na pinech jsou připojena jednotlivá relé.

Následuje volání funkce begin(), která je definována v knihovně obecného senzoru. Jelikož se jedná o výstupní senzor, je nutné nastavení šťítků pro odběr zpráv pomocí MQTT protokolu, k čemuž slouží metoda set_topic.

V případě příchozí zprávy pomocí MQTT protokolu, která je označena jedním z odebíraných šťítků, je volána metoda process_message, která dle šťítku vyhodnotí, kterému relé daná zpráva náleží. Následně dojde k volání metody relay_action(int pin, char* data), do které vstupuje pin, na kterém je dané relé připojeno a doručená zpráva - dle níž následně dojde k aktivaci nebo deaktivaci relé.

Na následujícím kódovém bloku je zobrazena struktura CONFIG_DATA, která slouží pro uložení štítků pomocí nichž je následně identifikováno, jakému relé zpráva náleží (řádky 2-5). Dále je zde definován text, který slouží k aktivaci relé (řádek 6) a na řádku 7 je definována zpráva, která slouží pro

deaktivaci relé. Veškerá konfigurační data v této struktuře je možné editovat s využitím webového rozhraní.

```
struct CONFIG_DATA{
        char relay_1_label[MQTT_LABEL_ARRAY_SIZE];
        char relay_2_label[MQTT_LABEL_ARRAY_SIZE];
        char relay_3_label[MQTT_LABEL_ARRAY_SIZE];
        char relay_4_label[MQTT_LABEL_ARRAY_SIZE];
        char relay_active_message[MQTT_LABEL_ARRAY_SIZE];
        char relay_deactive_message[MQTT_LABEL_ARRAY_SIZE];
};
```

11.5 Display modul

V tomto případě se jedná o kombinovaný modul realizovaný s využitím `ESP12-F`. Je zde obsažen přijímač pracující na frekvenci `433MHz`, tudíž tento modul umožnuje zobrazovat jak data přijatá na frekvenci `433MHz`, tak i pomocí `MQTT protokolu`.

Při implemetaci byla použita knihovna[61] pro práci s `I2C displejem` a knihovna[58] na obsluhu přijímače pracujícího na frekveci `433MHz`.

11.5.1 Popis funkce programu

V případě `display modulu` se jedná o `vstupně / výstupní` prvek. Prvním krokem je načtení nastavení z paměti. Následuje připojení k `Wi-Fi` a k `MQTT Brokeru`. Konfigurace načtených dat je možná jak prostřednictvím editačního formuláře, tak nahráním konfiguračního souboru ve formátu `Json`. Po úspěšném připojení k `MQTT Brokeru` následuje nastavení štítku pro odběr dat, která náleží `LCD Displeji`. Obsah přijaté zprávy, která je označena určitým štítkem se zobrazí na `LCD Displeji`.

Jako vstupní prvek je zde přijímač pracující na frekvenci `433MHz`, který přijatá data publikuje pomocí `MQTT Protokolu`.

11.5.2 Implementace

Implementace modulu je provedena v souboru `Lcd_module.cpp`. Prvotní nastavení je zajištěno voláním metody `setup()`, která zahájí funkci knihovny `obecného senzoru`, `rc přijímače` a `lcd displeje`.

Následuje volání metody `set_topic`, která nastaví štítek pro odběr zpráv pomocí `MQTT protokolu`. V tomto případě jsou odebírány zprávy pouze se

štítkem, který náleží pro zobrazení zpráv na displeji. Tento štítek je definován ve struktuře `config_data (řádek 3)`.

Je zde definována metoda `loop()`, která je cyklicky vykonávána. Jejím úkolem je volání metody `loop()` u knihovny obecného senzoru a kontrola, zda nebyla přijata nějaká data na frekveci `433MHz` pomocí metody `rc_check()`. V případě přijmutí dat jsou bezprostředně publikována pomocí `MQTT protokolu`. Tato data jsou označena štítkem, který je definovaný ve struktuře `config_data (řádek 2)`.

```
1 struct CONFIG_DATA{
2       char receiver_label[MQTT_LABEL_ARRAY_SIZE];
3       char display_label[MQTT_LABEL_ARRAY_SIZE];
4 };
```

11.6 RFID modul

K vytvoření modulu RFID čtečky použijeme mikročip `ESP12-F`, k němuž připojíme pomocí `SPI sběrnice` RFID čtečku `RFID-RC522`, která je založena na mikročipu `mcrf-522`. Jak čtečka RFID karet, tak `ESP12-F` pracují se stejnosměrným napětím o velikosti `3.3V`. Tudíž při jejich vzájemném propojení není nutno využít převodníku napěťových úrovní. Modul s centrálním prvkem komunikuje prostřednictvím `Wi-Fi` sítě s použitím `MQTT protokolu`.

Jedná se opět o kombinovaný modul, k mikročipu `ESP12-F` je také připojeno zvonkové tlačítko.

11.6.1 Popis funkce programu

V případě `RFID modulu` se jedná o `vstupní prvek`. Nejprve dojde k načtení nastavení z paměti. Následuje připojení k `Wi-Fi` a `MQTT Brokeru`. Z důvodu, že se jedná pouze o vstupní prvek a není potřeba odběru žádných dat, nedochází k nastavení odběru štítků. Program cyklicky kontroluje, zda nebyla přiložena `RFID karta` nebo stistknuto `zvonkové tlačítko`.

V případě přiložení `RFID Karty` následuje publikace `UUID Karty` pomocí `MQTT protokolu`. Data budou označena štítkem, který je možné nastavit pomocí webového rozhraní.

Při stisku `zvonkového tlačítka` dojde k odeslání předem definované zprávy pomocí `MQTT protokolu`. Odeslanou zprávu i štítek je možné nastavit prostřednictvím webového rozhraní.

11.6.2 Implemenace

Implementace RFID modulu je realizována v souboru Rfid_module.cpp. Při implementaci byla využita knihovna MFRC522.h[13], která je určena pro práci se RFID čtecím zařízením.

Prvotní nastavení je realizováno prostřednictvím metody setup(), která voláním metody mfrc_init() provede inicializaci RFID čtečky. Následuje nastavení pinu, na kterém je připojeno zvonkové tlačítko jako vstup a posledním krokem metody setup je zahájení činnosti knihovny Wi-Fi senzoru voláním metody begin().

Metoda loop() je cyklicky volána. Je zde kontrolováno, zda není stisknuto zvonkové tlačítko (metoda doorbell_check()) a oveření, zda nedošlo k přiložení RFID karty ke čtecímu zařízení (metoda rfid_check()). V neposlední řadě je volána metoda loop() u knihovny Wi-Fi senzoru.

Na následujícím kódovém bloku je zobrazena struktura CONFIG_DATA. Zpráva (řádek 2) o stisku zvonkového tlačítka je odesílána pomocí MQTT protokolu s určitou periodou (řádek 5). Tato zpráva bude označena štítkem definovaným na (řádku 4). Zprávy, které budou obsahovat informace o přiložené kartě, budou označeny štítkem definovaným na (řádku 3).

```
1 struct CONFIG_DATA{
2         char doorbell_active_message[MQTT_MESSAGE_ARRAY_SIZE];
3         char rfid_reader_label[MQTT_LABEL_ARRAY_SIZE];
4         char doorbell_label[MQTT_LABEL_ARRAY_SIZE];
5         int doorbell_period;
6 };
```

12 Popis funkce systému a pravidel

Příchodem libovolné osoby k brance dojde k zaznamenání pohybu. O této vzniklé situaci je odeslána zpráva pomocí `MQTT protokolu`, která dorazí do centrálního prvku, který následně aktivuje pravidlo `motion_detect.rules`, které odešle fotografii vzniklé situace všem obyvatelům dané domácnosti na aplikaci `Telegram` a zároveň ji uloží do paměti.

Libovolný člen domácnosti má možnost pomocí aplikace `Telegram` (odpovědí na přijatou fotografii) nebo aplikace `OpenHAB` odemknout vstupní branku. Odemčení proběhne tak, že centrální prvek zpracuje zprávu doručenou z aplikace `Telegram` (případně `OpenHAB aplikace`) a odešle pomocí `MQTT protokolu` příkaz k odemčení branky (sepnutí relé, které následně odblokuje magnetický zámek). Pomocí `LCD displeje`, je možnost zobrazení zprávy vstupující osobě.

Dalším možným, v praxi nejčastějším, případem je, že vstupní branku bude otevírat člen domácnosti.

Otevření branky je možné několika způsoby:

- Přiložení RFID karty - Čtečka RFID karet načte jedinečný identifikátor dané karty, který odešle ke zpracování centrálnímu prvku pomocí `protokolu MQTT`. Centrální prvek porovná identifikátor karty a následně rozhodne o povolení či zamítnutí vstupu. V případě známé karty dojde k vyslání zprávy k negaci stavu zámku a zobrazení uvítací zprávy na `LCD displeji`, v opačném případě je vstup zamítnut.
- Pomocí mobilní aplikace.
- Pomocí vysílače pracujícího na frekvenci 433 MHz - Přijímač přijme vysílaný kód a následuje stejný sled událostí jako je tomu u RFID karty. Pouze se nepracuje s identifikátorem karty, ale s přijatým kódem.

Je zde umístěn tlačítkový spínač (klasický zvonek), který reaguje na stisk. Při stisku tlačítka následuje stejný postup jako je tomu při zaznamenání pohybu.

12.1 Automatizační pravidla

Veškerá automatizační pravidla jsou definována na straně platformy `OpenHAB`. Nachází se ve složce `$OPENHAB_CONF/rules` a jsou reprezentována pomocí textových souborů, které mohou mít libovolný název, ale podmínkou je použití koncovky `.rules`.

12.1.1 Stisk zvonkového tlačítka

Při stisku zvonkového tlačítka, dojde k odeslání zprávy uživatelům domácnosti na aplikaci `Telegram`. Bude zde obsažen snímek situace u branky a dvě tlačítka. S pomocí tlačítek je možné `odemčení` případně `zamčení` vstupní branky. Uživatel bude zároveň informován o aktuálním stavu zámkového mechanismu branky. Zároveň bude přehrána melodie na reproduktoru, který je připojen k centrální jednotce (`Raspberry Pi`). Výše zmíněný postup je implementován v souboru `doorbell.rules`.

Možnosti aktivace/deaktivace zámku u vstupní branky

- Pomocí aplikace `Domácnost`
- Pomocí aplikace `Google Home`
- Pomocí hlasového asistenta `Siri` nebo `Google Assistant`
- Pomocí aplikace `OpenHAB` nebo webového rozhraní `OpenHab`
- Pomocí aplikace `Telegram`
- Přiložením přístupové karty k RFID čtečce u branky
- Pomocí dálkového ovladače
- Pomocí aplikace `Shortcuts`, např. definice uživatelské zkratky při přiložení přístupové karty k mobilnímu telefonu.

V tomto případě se nabízí široké možnosti konfigurace, jako je např. řízení přístupu v závislosti na denní době. Při stisku zvonkového tlačítka může být branka automaticky odemčena v určitém čase, případně se automaticky může odemknout vždy, když je někdo z členů domácnosti přítomen.

12.1.2 Přiložení RFID karty

Přiložením RFID karty ke čtecímu zařízení dojde k načtení jejího UUID, které já následně přeneseno k centrální jednotce, kde dojde k vyhodnocení dat.

Nejprve je testováno, zda se nacházáme v režimu učení nebo řízení přístupu.

- Režim učení slouží k nastavení přístupové karty konkrétnímu uživateli. V Main UI zvolíme položku Manage users, následně se přepneme do režimu učení (tlačítko LEARNING). Je zde možnost vytvoření nového uživatele nebo výběr již existujícího, po provedení výběru, případně vytvoření nového uživatele (a jeho následném výběru) stačí přiložit jeho vstupní kartu ke čtečce. Tímto dojde k přiřazení UUID karty uživateli.
- Režim řízení přístupu slouží k vyhodnocení přiložené karty ke čtecímu zařízení. Aktivace režimu se provede stiskem tlačítka ACTIVE. V případě, že se jedná o známou kartu (tzn. je přiřazena některému uživateli), dojde k negaci aktuálního stavu zámku dveří. V opačném případě (přiložení neznámé karty) je branka uzamčena a následuje stejný postup jako při stisku zvonkového tlačítka.

Implementováno v souboru rfid_door.rules.

12.1.3 Příjem dat na frekvenci 433MHz

V tomto případě je postup velmi podobný, jako je tomu u zpracování dat při přiložení RFID karty. Nastavení konkrétního kódu danému uživateli, který když bude přijatý, dojde k negaci stavu zámku, je totožné jako při nastavení UUID přístupové karty danému uživateli. Jediná změna je, že nastovování probíhá v záložce 433MHz.

Změna nastává při vyhodnocení přijatých dat. Pokud je přijat kód, který je přiřazen některému z uživatelů, dojde k negaci stavu zámku. V případě neznámého kódu není vykonána žádná činnost, protože přijatá data mohou pocházet např. z bezdrátového teplotního senzoru.Zpracování těchto dat by vytvářelo falešné podněty.

Tento příklad pouze demonstruje použití bezdrátového ovladače. Přenos dat není žádným způsobem kódován a může být odposlechnut a zneužit. Pro reálné nasazení by přenášená data musela být zabezpečena proti odposlechu a zneužití.

Implementováno v souboru r433_door.rules.

12.2 Správa uživatelů

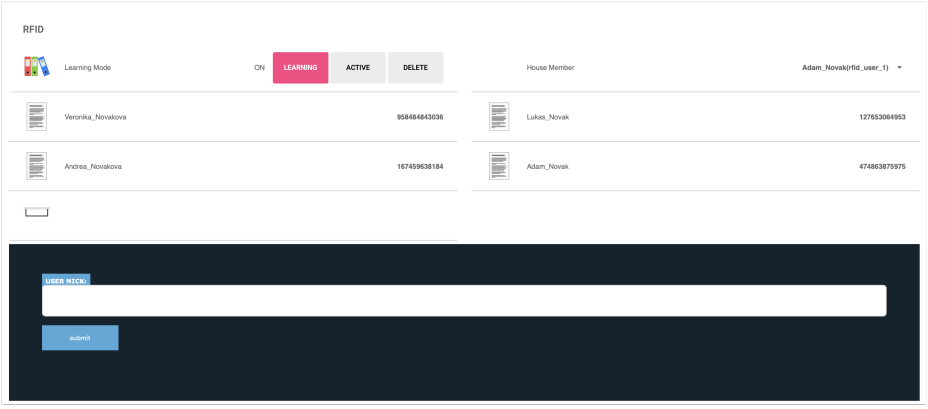
Data všech uživatelů, kteří mohou ovládat stav zámku, ať už pomocí `přístupové karty` nebo s využitím `dálkového ovladače`, jsou uložena celkem ve třech konfiguračních souborech:

- `users.items`
- `default.things`
- `users.sitemap`

V případě souboru `default.things` je nutné uložení dat zde, aby došlo k propojení s `mostem přístupového systému` tzn. položka uživatele dokáže reagovat na příchozí podněty.

Přidání uživatele je realizováno pomocí `HTML formuláře 12.1`, do kterého uživatel zadá jméno nebo přezdívku přidávaného uživatele (budou odstraněny mezery a diakritika). Následně jsou data zpracována pomocí `PHP skriptu`, který vygeneruje a přidá potřebná data do všech konfiguračních souborů. Každému uživateli je přiděleno jedinečné `ID`.

Odstranění uživatele je realizováno výběrem daného uživatele z rozbalovacího seznamu a následně je vyvolán `PHP skript`, který odstraní z konfiguračních souborů data o daném uživateli.



Obrázek 12.1: Správa uživatelů - vstupní systém

13 Mobilní zařízení

Ovládání chytré domácnosti z mobilního zařízení patří mezi klíčové vlastnosti, jejichž integrace by neměla chybět. Platforma OpenHAB nabízí široké možnosti spolupráce s mobilními zařízeními.

13.1 Apple Domácnost

Jedná se o aplikaci dostupnou pouze pro zařízení od firmy Apple, jako je např. (Apple TV, iPhone, iPad a další). Tato aplikace[34] slouží k monitorování a ovládání chytré domácnosti. Je zde možné sledovat stav teploměrů, vlhkoměrů a dalších čidel. Dále je možné pomocí aplikace ovládat osvětlení, dveřní zámek a další zařízení.

Před samotným přidáním zařízení do aplikace Domácnost, je potřeba na straně platformy OpenHAB provést přidání doplňku HomeKit Add-on[44] a konfiguraci v souboru OPENHAB_CONF/services/homekit.cfg.

Přidání zařízení do HomeKitu je možné jak z grafického prostředí platformy OpenHAB, tak i pomocí konfiguračního souboru .items.

V případě přidávání z grafického prostředí vybereme v mainUI (hlavní stránka) záložku items, následuje výběr konkrétní položky, u které pokračujeme volbou Add Metadata. Dojde k otevření seznamu, kde klikneme na Apple HomeKit, následně volíme HomeKit Accessory/Characteristic, kde vybereme, o jaký typ zařízení se jedná (např. MotionSensor) a potvzení provedeme stiskem tlačítka save.

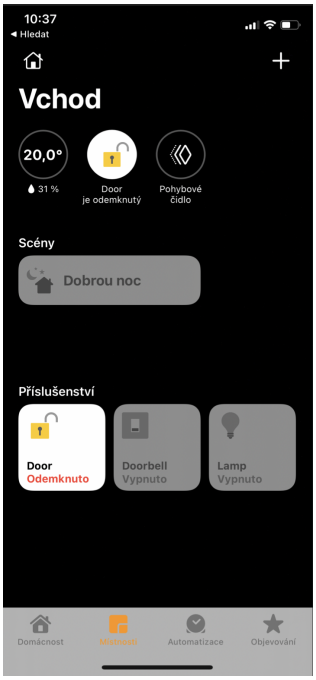
Při rozhodnutí postupovat integrací zařízení do HomeKitu pomocí konfiguračního souboru se veškerá konfigurace provádí v souboru .items, který může mít libovolné, ale jedinečné pojmenování (důležitá je koncovka .items). Na následujícím příkladu je zobrazena stejná akce, jako je již výše popsaná v grafickém prostředí. Jedná se o položku typu spínač s názvem motion_sensor a popiskem Motion Sensor, který je integrován do HomeKitu jako pohybový senzor.

```
1 Switch motion_sensor   "Motion Sensor"     {homekit="MotionSensor"}
```

V aplikaci `Domácnost` je možné definovat scény případně automatizační pravidla. Definice automatizačních pravidel v této aplikaci není příliš vhodná z důvodu, že pravidla jsou již definována na straně platformy `OpenHAB`, ovšem definice scén je možná s výhodou, že každý uživatel má svůj mobilní telefon, kde může mít definované své oblíbené scény.

Ovládání položek integrovaných do `HomeKitu` je možné i pomocí hlasového asistenta `Siri`.

Na obrázku 13.1 je zobrazeno grafické prostředí aplikace na iphonu.



Obrázek 13.1: Aplikace domácnost - iPhone

Nechybí zde ani možnost ovládání pomocí chytrých hodinek `Apple Watch`. Na obrázku 13.2 jsou zobrazeny možnosti ovládání zámku a sledování stavu teploty a vlhkosti.

Obrázek 13.2: Aplikace domácnost - iWatch

13.2 Mobilní aplikace Shortcuts

Jedná se o aplikaci[5] dostupnou pro zařízení `iPhone a iPad`, která nabízí velmi široké možnosti použití v oblasti domácí automatizace pomocí mobilního telefonu.

Automatizační úkony je možné provádět spoluprací aplikace `Shortcuts` a `Apple Domácnost` např. při zahájení nabíjení mobilního telefonu aktivovat scénu vytvořenou v aplikaci `Apple Domácnost`.

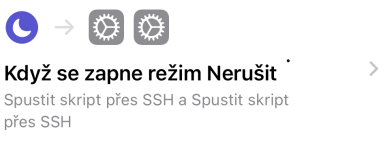
Aplikace `Shortcuts` umožnuje pomocí `SSH` navázat spojení se serverem a na něm vykonávat příkazy případně spustit skript. Pomocí této vlastnosti je možné se připojit k `OpenHAB serveru` a ovládat stav jednotlivých položek.

Aplikace dále umožňuje použití `NFC Tagů` a reakce na události z `Apple Watch`.

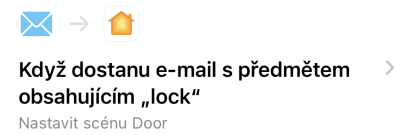
Možné automatizační úkony

- Při připojení mobilního telefonu k nabíječce aktivovat scénu `"Dobrou noc"`, která zamkne dveřní zámek a rozsvítí lampu (obrázek: 13.3a).
- Při aktivaci režimu `Nerušit` aktivovat domovní zámek (obrázek: 13.3b).
- Když přijde email z určité adresy a v předmětu je obsaženo slovo `lock`, dojde k aktivaci dveřního zámku (obrázek: 13.3c).
- Při přiložení `NFC Karty` aktivovat scénu `"Dobré ráno"` (obrázek: 13.3d).

(a) Zkratka napájení



(b) Zkratka nerušit



(c) Zkratka příchozí email

(d) Zkratka přiložení NFC karty

Obrázek 13.3: Mobilní aplikace Shortcuts - možné zkratky

13.3 Mobilní aplikace OpenHAB

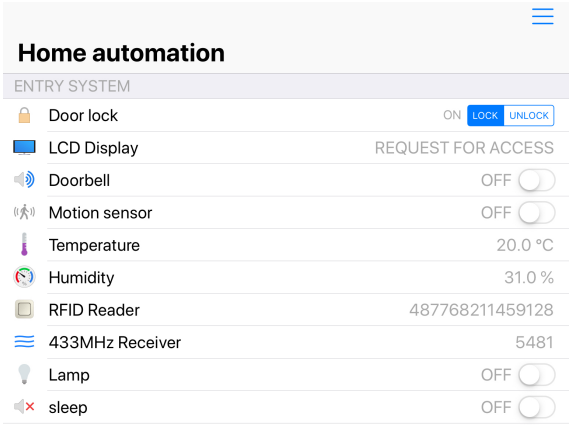
Platforma `OpenHAB` nabízí aplikaci pro mobilní telefony a tablety s operačním systémem `Android`[45] i `iOS`[1]. Aplikace přináší možnost ovládání automatizační platformy pomocí zobrazeného grafického prostředí, které definuje uživatel v souboru `.sitemap`.

V případě provozování aplikace na zařízeních s operačním systémem `Android` je možné vytvářet a použítvat `RFID Tagy`, které následně pomocí `NFC` v mobilním telefonu načteme a můžeme změnit stav některého ze zařízeních.

V případě použití doplňku `openHAB Cloud Connector`[50] je možné ovládání a sledování stavu domácnosti pomocí aplikace i mimo lokální síť. Použití tohoto doplňku přináší i další výhodu v možnosti zasílání notifikací na mobilní zařízení. Na následujícím příkladu je zobrazeno pravidlo (definované v souboru `.rules`), které v případě změny stavu dveří odešle uživateli, který je registrován pod emailovou adresou `mracji99@seznam.cz` notifikaci o změně stavu dveří. Případně funkce `sendBroadcastNotification(msg)` odešle notifikaci všem uživatelům systému.

```
1 rule "Door Notification"
2 when
3   Item Door changed to OPEN
4 then
5   sendNotification("mracji99@seznam.cz", "Door was opened!")
6 end
```

Vytváření uživatelů a definice role uživatele je možná prostřednictvím webového rozhraní na adrese `https://myopenhab.org/users`.



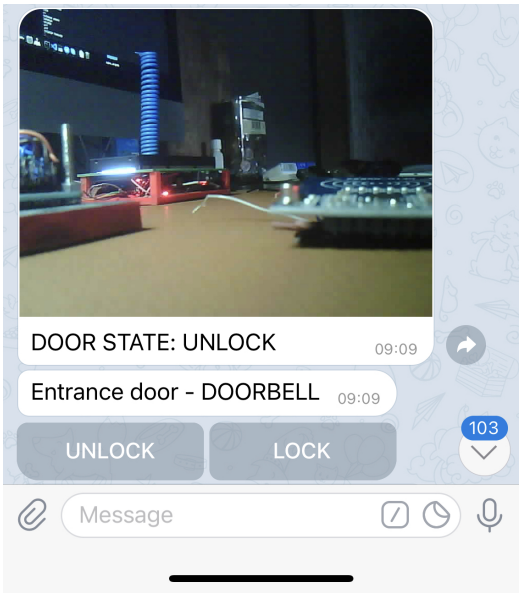
Obrázek 13.4: Aplikace OpenHAB

13.4 Mobilní aplikace Telegram

Telegram[75] je aplikace, která slouží pro výměnu zpráv mezi dvěma a více uživateli. Zprávy mohou být různých datových formátů např. textové, hlasové, fotografie nebo video záznam. U aplikace je kladen důraz na bezpečnost a rychlost provozu. Aplikace je použitelná jak na zařízeních s operačním systémem `Android`, tak i `iOS`. Použití `Telegramu` v chytré domácnosti je možné v několika odvětvích. První možností použití je příjem zpráv, kde je možné uživateli zasílat libovolné informace. Dále může uživatel zaslat zprávu, která bude centrální jednotkou zpracována např. `"odemkni dveře"`.

Nechybí zde ani možnost zasílání fotografií, čehož lze využít v případě kamerového modulu např. při zaznamenání pohybu je pořízen snímek, který je pomocí aplikace Telegram odeslán uživateli. V případě zaslání fotografie při zaznamenání pohybu u dveří je možné zaslat s fotografií i tlačítka (obrázek: 13.5), jejichž stiskem uživatel odešle předdefinovaný text, který bude následně zpracován (v tomto případě odemčení nebo zamčení dveří).

Použití aplikace `Telegram` nabízí možnost ovládání chytré domácnosti z prostředí mimo lokální síť i bez použití doplňku `openHAB Cloud Connector`.



Obrázek 13.5: Aplikace Telegram

13.5 Google Home

Uživatelé s mobilním telefonem či tabletem s operačením systémem Android mohou využít hlasového asistenta Google Assistant spolu s aplikací Google Home[33] ke sledování a ovládání stavu zařízeních v domácnosti.

Pro zprovoznění této aplikace je nejprve nutné přidání doplňku openHAB Cloud Connector[50] na platformě OpenHAB. Následně je možné přikročit k samotné instalaci aplikace Google Home. Po instalaci je potřeba integrace platformy OpenHAB.

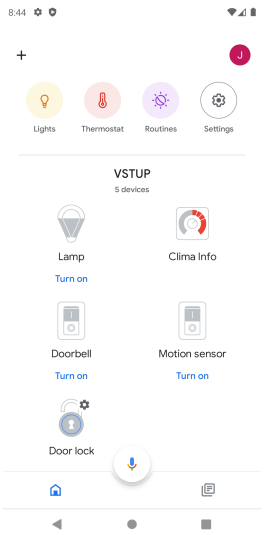
Nastavení položek, které budou následně přidány do Google Home můžeme obdobně, jako tomu bylo u platformy HomeKit, přistoupit editací konfiguračního souboru .items. Na následujícím příkladu je zobrazeno přidání položky lamp, které se zobrazuje v grafickém prostředí pod názvem Lamp a je označena pro Google Assistant (ga) jako Light(osvětlení).

```
1 Switch lamp          "Lamp"          {ga="Light"}
```

Výše zmíněný postup můžeme realizovat i pomocí grafického prostředí platformy openHAB, kde postupujeme následujícím způsobem. Z grafického prostředí vybereme v mainUI (hlavní stránka) záložku items. Následuje výběr konkrétní položky, u které pokračujeme volbou Add Metadata. Dojde k otevření seznamu, kde klikneme na Google Assistant a následně volíme Google Assistant Class, kde vybereme o jaký typ zařízení se jedná (např. Switch) a potvzení provedeme stiskem tlačítka save.

Nejprve přejdeme v aplikaci do Nastavení, kde volíme Assistant a následně položku Home control. Přidáme zařízení OpenHAB a pomocí přihlašovacích údajů k openHAB Cloud Connector se přihlásíme a přidáme položky do Google Home.

V tomto okamžiku můžeme již ovládat domácnost pomocí hlasového Google Assistant. Je možné zadávat povely jako např. "OK Google! Unlock the door.". Část "OK Google!" slouží k aktivaci hlasového asistenta.



Obrázek 13.6: Aplikace Google Home

13.6 Shrnutí kapitoly

Výběr a následné použití aplikace je závislý na operačním systému. V případě operačního systému `Android` je nejlepší volba použití aplikace `OpenHAB`, která dovoluje širší použití v případě provozu na mobilním zařízení s `Androidem`, oproti použití na zařízení od firmy `Apple`. Hlavní odlišnost spočívá v použití v možnosti `NFC Tagů` a zasílání hlasových příkazů přímo z tého aplikace.

Případnou možnou alternativou pro mobilní zařízení s `Androidem` je použití aplikace `Google Home`, kterou je možné ovládat pomocí hlasového asistenta `Google Assistant`. Provoz aplikace `Google Home` je možný i na `iPhone`.

Pro zařízení od firmy `Apple` je vhodné použití aplikace `Domácnost`, kterou je možné ovládat pomocí hlasového asistenta `Siri`. Dále je možné využití aplikace `Shortcuts`, která umožnuje reagovat na vzniklé situace na zařízeních pocházejících od firmy `Apple`, jako je např. otevření aplikace nebo připojení telefonu k nabíječce, případně vytvoření vlastních zkratek. Samožřejmostí je možnost používání aplikace `OpenHAB` i na zařízeních s `iPhone či iPad`.

Použití aplikace `Telegram` je bezproblémové jak na zařízeních s operačním systémem `Android`, tak `iOS`.

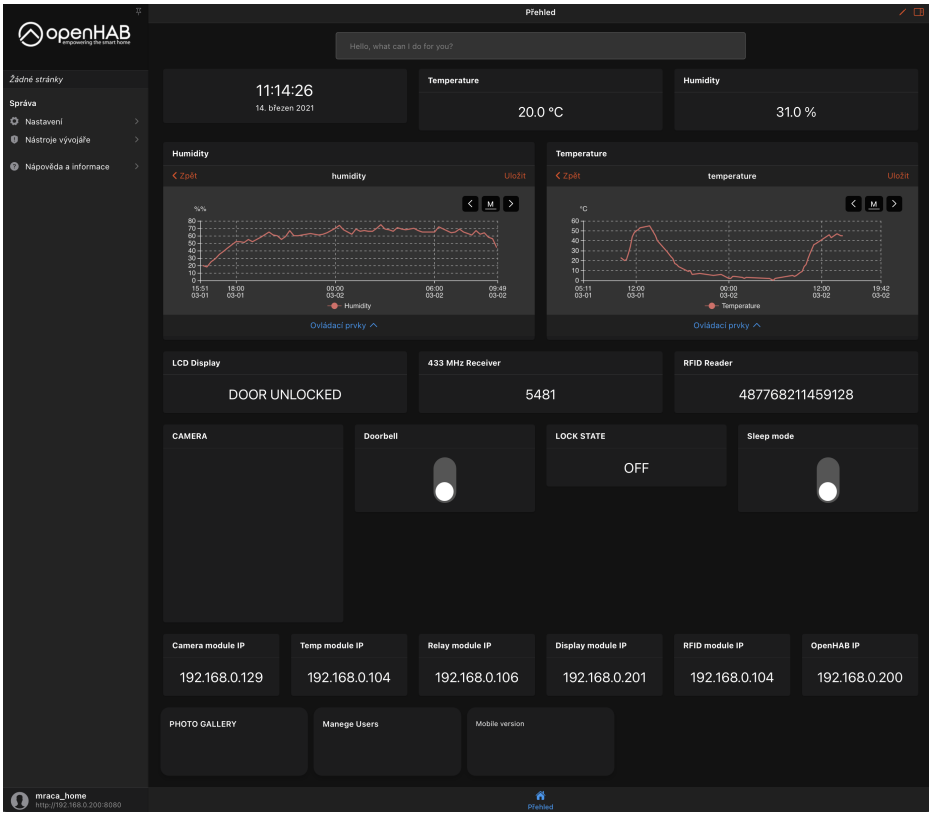
Možnost ovládání `OpenHABu` je možná i prostřednictvím webového prohlížeče dostupného v chytrém mobilním telefonu.

14 Uživatelské rozhraní

Ovládání chytré domácnosti je umožněno pomocí grafického uživatelského prostředí, které musí být přehledné a intuitivní. Dalším klíčovým parametrem je možnost ovládání z mobilího telefonu -> uživatelské prostředí musí být pro tuto možnost uzpůsobeno, ať už pomocí responsibilního zobrazení webového rozhraní nebo dostupností mobilní aplikace.

14.1 Main User Interfaces

Jedná se o hlavní uživatelské rozhraní platformy domácí automatizace, jehož definice je možná pomocí konfiguračního souboru s použitím `YAML kódování`, případně jednotlivé prvky je možné definovat přímo s využitím grafického webového prostředí. Na obrázku 14.1 je zobrazeno hlavní grafické prostředí.



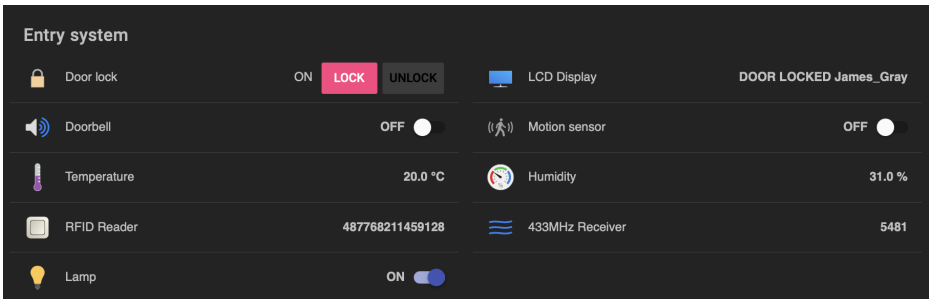
Obrázek 14.1: OpenHAB Main UI - PC verze

14.2 Basic User Interfaces

Basic User Interfaces zkráceně (Basic UI) je grafické uživatelské prostředí platformy `OpenHAB`, které se definuje pomocí konfiguračních souborů ve složce `$OPENHAB_CONFsitemaps`. Soubory mohou být libovolně pojmenovány, ale důležitá je koncovka `.sitemap`. V případě pojmenování konfiguračního souboru `default.sitemap`, bude tento soubor zobrazen bez nutnosti výběru souboru, který se má zobrazit. Na následujícím příkladu je zobrazen příklad definice uživatelského rozhraní, ve kterém bude zobrazen jeden přepínač. Tento přepínač je označen ikonou osvětlení a je umístěň v rámci s názvem `Lighting`.

```
1 sitemap default label="Automation" {
2     Frame label="Lighting" {
3       Switch item=Lamp icon="light"
4     }
5 }
```

Na obrázku 14.2 je zobrazeno grafické uživatelské prostředí, které slouží k ovládání vstupního systému. Definice tohoto prostředí je realizována prostřednictvím souboru `default.sitemap`.



Obrázek 14.2: OpenHAB Basic UI - PC verze

15 Závěr

Při realizaci této bakalářské práce jsem prozkoumal dostupné automatizační platformy, především jsem se zaměřil na automatizační platformy šířené pod licencí `Open Source`. Následně jsem se rozhodl využít automatizační platformu `OpenHAB`. Při realizaci práce byla vydána nová verze této platformy. Z důvodu dalšího praktického využití navrženého systému jsem přešel na nejnovější verzi.

Výběr vhodného zařízení pro jednotlivé senzory byl proveden opět s důrazem na možnou rozšiřitelnost, proto jsem zvolil pro většinu senzorů (teplotní, rfid nebo relé modul) zařízení `ESP12-F`, které díky dostatečnému množství vstupně výstupních pinů disponuje širokou možností rozšiřitelnosti daného senzoru. U kamerového modulu jsem zvolil zařízení `ESP32-CAM`, které opět nabízí široké možnosti rozšiřitelnosti a velkou výhodou je integrace kamerového modulu. Veškeré kryty senzorů jsem vyráběl svépomocí s využitím 3D tisku.

Jelikož jedním z hlavních cílů byla možnost snadné rozšiřitelnosti, tak jsem navrh knihovnu pro univerzální senzor. Použitím této knihovny je implementace nového senzoru značně zjednodušena a výsledný zdrojový kód je zpřehledněn.

Fyzické umístění senzoru nemusí být vždy zcela uživatelsky dostupné, z toho důvodu jsem použit externí knihovnu, která umožňuje vzdálené zavedení programu do vývojové desky `ESP32-CAM` nebo `ESP12-F`.

Pro centrální prvek jsem zvolil `Raspberry PI 4 - 8GB RAM`. Pro běžný provoz platformy `OpenHAB` je toto zařízení zcela dostačující, možná do jistého smyslu až předimenzované.

Realizoval jsem několik možností ovládání z mobilního telefonu, zaměřil jsem se na především na operační systém `Apple`, ale nezapoměl jsem testovat zařízení i na `Androidu`. Největší dojem ve mě zanechala aplikace `Shortcuts`, která je určena pro zařízení od firmy `Apple`. Tato aplikace umožňuje široké možnosti domácí automatizace pomocí mobilního telefonu, např. při zapojení mobilního telefonu na nabíječku aktivovat dveřní zámek, nebo při přiložení `NFC Tagu` k mobilu aktivovat venkovní osvětlení.

Uživatelské rozhraní automatizační platformy `OpenHAB` působí uživatelsky velmi přívětivým dojmem. Možnosti nastavení domácí automatizace jsou velmi široké a taktéž komunitní podpora je více než rozsáhlá. Volbu této automatizační platformy považuji po otestování její funkčnosti a použitelnosti za velmi dobrou.

Vytvořený systém byl testován po dobu realizace této práce. Jedná se o funkční a stabilní návrh.

Seznam použitých zkratek

- [**CMOS**] Complementary Metal–Oxide–Semiconductor
- [**COAP**] Constrained Application Protocol
- [**FPS**] Frames Per Second
- [**GPIO**] General Purpose Input Output
- [**HDMI**] High-Definition Multimedia Interface
- [**I2C**] Inter-IC bus
- [**IEEE**] Institute of Electrical and Electronics Engineers
- [**IoT**] Internet of Things
- [**JSON**] JavaScript Object Notation
- [**LCD**] Liquid Crystal Display
- [**LoRaWAN**] Long Range Wide Area Network
- [**MQTT**] MQ Telemetry Transport
- [**OSI**] Open Systems Interconnection
- [**OTA**] Over the Air
- [**PC**] Personal Computer
- [**PIR**] Passive infrared sensor
- [**QoS**] Quality of Service
- [**RAM**] Random Access Memory
- [**RFID**] Radio-frequency identification
- [**SPI**] Serial Peripheral Interface
- [**TCP/IP**] Transmission Control Protocol/Internet Protocol
- [**UART**] Universal Asynchronous Reciever - Transmitter
- [**USB**] Universal Serial Bus
- [**Wi-Fi**] Wireless fidelity

Obsah přiloženého CD

- Složka **openHAB** - významné konfigurační soubory domácí automatizace.
- Složka **externi_knihovny** - použité externí knihovny.
- Složka **senzor_knihovna** - zdrojové kódy pro Wi-Fi senzor.
- Složka **senzory_json** - konfiguracni soubory pro jednotlive senzory.
- Složka **vybrana_literatura** - významná literatura.
- Složka **JIRI_MRACEK_BPINI.pdf** - bakalářská práce.
- Složka **vstupni_system.mp4** - video demonstrující funkčnost systému.

Literatura

[1] [online]. [cit. 202/0/0]. Dostupné z: https://www.openhab.org/docs/apps/ios.html.

[2] Alliance, L. *What is it?* [online]. LoRa® Alliance, . [cit. 2020.9.23]. Dostupné z: https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf.

[3] Alliance, W.-F. *Discover Wi-Fi* [online]. Wi-Fi Alliance, . [cit. 2020.9.23]. Dostupné z: https://www.wi-fi.org/discover-wi-fi.

[4] Amazon. *What Is Alexa?* [online]. Amazon. [cit. 2020.10.4]. Dostupné z: https://developer.amazon.com/en-US/alexa.

[5] Apple. *Shortcuts* [online]. Apple. [cit. 2020.11.19]. Dostupné z: https://apps.apple.com/app/shortcuts/id915249334.

[6] Arduino. *Arduino IDE* [online]. Arduino, . [cit. 2020.08.09]. Dostupné z: https://www.arduino.cc/en/main/software.

[7] Arduino. *ArduinoCore-avr* [online]. Arduino, . [cit. 2020.10.05]. Dostupné z: https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/Arduino.h.

[8] Arduino. *EEPROM Library* [online]. Arduino, . [cit. 2020.12.18]. Dostupné z: https://www.arduino.cc/en/Reference/EEPROM.

[9] Arduino. *Arduino core for ESP8266 WiFi chip* [online]. Arduino, . [cit. 2021.02.19]. Dostupné z: https://github.com/esp8266/Arduino/tree/master/doc/esp8266wifi.

[10] askarduino.cz. *ESP-12F ESP8266 WIFI modul* [online]. askarduino.cz. [cit. 2020.07.13]. Dostupné z: https://www.laskarduino.cz/esp-12f-esp8266-wifi-modul--tcp-ip/.

[11] Assistant, H. *Awaken your home* [online]. Home Assistant, . [cit. 2020.12.17]. Dostupné z: https://www.home-assistant.io.

[12] Assistant, H. *Home Assistant* [online]. Home Assistant, . [cit. 2020.11.05]. Dostupné z: https://demo.home-assistant.io/.

[13] Balboa, A. *MFRC522* [online]. André Balboa. [cit. 2020.12.07]. Dostupné z: https://github.com/miguelbalboa/rfid.

[14] Balena. *Why balenaEtcher?* [online]. Balena. [cit. 2020.08.10]. Dostupné z: https://www.balena.io/etcher/.

[15] Blanchon, B. *ArduinoJson is a JSON library for embedded C++* [online]. Benoît Blanchon. [cit. 2020.12.18]. Dostupné z: https://arduinojson.org.

[16] Co, A.-T. T. *ESP12F-datasheet* [online]. Ai-Thinker Technology Co. [cit. 2020.05.13]. Dostupné z: https://docs.ai-thinker.com/_media/esp8266/docs/esp-12f_product_specification_en.pdf.

[17] CO, S. A. T. *ESP-32S Datasheet* [online]. Shenzhen Anxinke Technology CO. [cit. 2020.7.13]. Dostupné z: https://www.es.co.th/Schemetic/PDF/ESP32.PDF.

[18] Connect, H. *Welcome to Home Connect* [online]. Home Connect. [cit. 2020.10.4]. Dostupné z: https://www.home-connect.com/global.

[19] Dev, M. N. *AsyncTCP* [online]. Me No Dev, . [cit. 2020.10.03]. Dostupné z: https://github.com/me-no-dev/AsyncTCP.

[20] Dev, M. N. *ESPAsyncTCP* [online]. Me No Dev, . [cit. 2020.10.03]. Dostupné z: https://github.com/me-no-dev/ESPAsyncTCP.

[21] Dev, M. N. *ESPAsyncWebServer* [online]. Me No Dev, . [cit. 2020.10.05]. Dostupné z: https://github.com/me-no-dev/ESPAsyncWebServer.

[22] Domoticz. *About Domoticz* [online]. GitHub, Inc, . [cit. 2020.11.05]. Dostupné z: https://github.com/domoticz/Machinon.

[23] Domoticz. *About Domoticz* [online]. Domoticz, . [cit. 2020.11.05]. Dostupné z: https://www.domoticz.com/wiki/About_Domoticz.

[24] Dratek.cz. *ESP32-CAM vývojová deska WiFi + Bluetooth s kamerovým modulem OV2640* [online]. Dratek.cz, . [cit. 2021.02.11]. Dostupné z: https://dratek.cz/arduino/7587-esp32-cam-vyvojova-deska-wifi-bluetooth-s-kamerovym-modulem-ov2640.html.

[25] Dratek.cz. *NiceRF 433MHz SRX882 přijímač - modul ASK* [online]. Dratek.cz, . [cit. 2020.12.28]. Dostupné z: https://dratek.cz/arduino/2111-nicerf-433mhz-srx882-prijimac-modul-ask.html.

[26] Eclipse Foundation, I. *Eclipse Mosquitto* [online]. Eclipse Foundation, Inc. [cit. 2020.08.10]. Dostupné z: https://mosquitto.org.

[27] Electronics, M. *DHT11 Humidity and Temperature Sensor* [online]. Mouser Electronics. [cit. 2021.01.12]. Dostupné z: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf.

[28] Fielding, e. a. *Hypertext Transfer Protocol* [online]. Fielding, et al. [cit. 2020.9.16]. Dostupné z: https://tools.ietf.org/html/rfc2616#page-176.

[29] GmbH, B. S. *BME280* [online]. Bosch Sensortec GmbH, . [cit. 2021.01.10]. Dostupné z: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf.

[30] GmbH, H. *Quality of Service 0,1 & 2 - MQTT Essentials: Part 6* [online]. HiveMQ GmbH, . [cit. 2020.9.17]. Dostupné z: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/.

[31] GmbH, H. *MQTT Security Fundamentals* [online]. HiveMQ GmbH, 2021. [cit. 2020.9.20]. Dostupné z: https://www.hivemq.com/mqtt-security-fundamentals/.

[32] Google. *Discover what Google Assistant is* [online]. Google, . [cit. 2020.10.4]. Dostupné z: https://assistant.google.com.

[33] Google. *Smart Home* [online]. Google, . [cit. 2020.10.4]. Dostupné z: https://developers.google.com/assistant/smarthome/overview.

[34] Inc, A. *Your home at your command* [online]. Apple Inc, . [cit. 2020.10.4]. Dostupné z: https://www.apple.com/ios/home/.

[35] Inc, A. *Siri does more than ever. Even before you ask* [online]. Apple Inc, . [cit. 2020.10.4]. Dostupné z: https://www.apple.com/siri/.

[36] LastMinuteEngineers.com. *ESP32 Basic Over The Air (OTA) Programming In Arduino IDE* [online]. LastMinuteEngineers.com. [cit. 2020.08.09]. Dostupné z: https://lastminuteengineers.com/esp32-ota-updates-arduino-ide/.

[37] Ltd, R. P. T. *Raspberry Pi 4 Model B* [online]. Raspberry Pi (Trading) Ltd. [cit. 2021.01.16]. Dostupné z: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi_DATA_2711_1p0_preliminary.pdf.

[38] Microsoft. *Visual Studio 2019* [online]. Microsoft. [cit. 2020.08.10]. Dostupné z: https://visualstudio.microsoft.com/cs/vs/.

[39] MPJA.com. *HC-SR501 PIR MOTION DETECTOR* [online]. MPJA.com. [cit. 2021.03.07]. Dostupné z: https://www.mpja.com/download/31227sc.pdf.

[40] *MQTT: The Standard for IoT Messaging* [online]. [cit. 2020.9.16]. Dostupné z: https://mqtt.org.

[41] NXPSemiconductorsN.V. *MFRC522* [online]. NXPSemiconductorsN.V. [cit. 2021.01.23]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf.

[42] O'Leary, N. *Arduino Client for MQTT* [online]. Nick O'Leary. [cit. 2020.12.18]. Dostupné z: https://github.com/knolleary/pubsubclient.

[43] OmniVision. *uctronics.com* [online]. uctronics.com. [cit. 2020.05.13]. Dostupné z: https://www.uctronics.com/download/cam_module/OV2640DS.pdf.

[44] Community – Foundation. *HomeKit Add-on* [online]. openHAB Community and the openHAB Foundation, . [cit. 2020/11/09]. Dostupné z: https://www.openhab.org/addons/integrations/homekit/.

[45] Community – Foundation. *Android App* [online]. openHAB Community and the openHAB Foundation, . [cit. 2020/11/09]. Dostupné z: https://www.openhab.org/docs/apps/android.html.

[46] Community – Foundation. [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.07]. Dostupné z: https://www.openhab.org/addons/bindings/mqtt/.

[47] Community – Foundation. *openHAB* [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.04]. Dostupné z: https://www.openhab.org/.

[48] Community – Foundation. *Add-on Reference* [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.04]. Dostupné z: https://www.openhab.org/addons/.

[49] Community – Foundation. *Binding Definitions* [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.04].

[50] Community – Foundation. *openHAB Cloud Connector* [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.04]. Dostupné z: https://www.openhab.org/addons/integrations/openhabcloud/#openhab-cloud-connector.

[51] Community – Foundation. [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.07]. Dostupné z: https://demo.openhab.org.

[52] Community – Foundation. *Thing Discovery* [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.07]. Dostupné z: https://www.openhab.org/docs/concepts/discovery.html.

[53] Community – Foundation. *Items* [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.04]. Dostupné z: https://www.openhab.org/docs/configuration/items.html.

[54] Community – Foundation. *A Deeper Dive: openHAB Structure for Advanced Users* [online]. openHAB, 2020. [cit. 2021.02.01]. Dostupné z: https://www.openhab.org/docs/.

[55] Community – Foundation. *Things* [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.04]. Dostupné z: https://www.openhab.org/docs/configuration/things.html.

[56] Community – Foundation. *Welcome!* [online]. openHAB Community and the openHAB Foundation e.V., 2021. [cit. 2021.02.04]. Dostupné z: https://www.openhab.org/docs/.

[57] Community – Foundation. *Who We Are* [online]. openHAB, 2020. [cit. 2021.02.09]. Dostupné z: https://www.openhab.org/about/who-we-are.html.

[58] Ozgur, S. *rc-switch* [online]. Suat Ozgur. [cit. 2020.12.07]. Dostupné z: https://github.com/sui77/rc-switch/.

[59] OÜ, S. *Diagnosing ESP32 FLASH memory programming issues* [online]. Sysprogs OÜ. [cit. 2020.08.10]. Dostupné z: https://visualgdb.com/tutorials/esp32/flashdiag/.

[60] Randomnerdtutorials. *Installing the ESP32 Board in Arduino IDE (Mac OS X and Linux instructions)* [online]. Randomnerdtutorials. [cit. 2020.08.10]. Dostupné z: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-mac-and-linux-instructions/.

[61] Rickman, J. *LiquidCrystal I2C* [online]. John Rickman. [cit. 2020.12.07]. Dostupné z: https://github.com/johnrickman/LiquidCrystal_I2C.

[62] Robert Gee, C. P. G. M. I. E. B. M. *CAN vs. RS-485: Why CAN Is on the Move* [online]. maximintegrated. [cit. 2020.9.23]. Dostupné z: https://www.maximintegrated.com/content/dam/files/design/technical-documents/white-papers/can-wp.pdf.

[63] Sharma, A. *ElegantOTA* [online]. Ayush Sharma. [cit. 2021.01.27]. Dostupné z: https://github.com/ayushsharma82/ElegantOTA.

[64] SIG, B. *Bluetooth Radio Versions* [online]. Bluetooth SIG. [cit. 2020.9.23]. Dostupné z: https://www.bluetooth.com/learn-about-bluetooth/radio-versions/.

[65] SLINTÁK, V. *Arduino a sériová komunikace* [online]. uart.cz. [cit. 2020.08.09]. Dostupné z: https://uart.cz/139/arduino-a-seriova-komunikace/.

[66] s.r.o., E. *Display modrý 20x4 znaků HD44780* [online]. ECLIPSERA s.r.o., . [cit. 2021.03.07]. Dostupné z: https://dratek.cz/arduino/986-display-modry-20x4-znaku-hd44780.html?gclid=EAIaIQobChMIufGcirmb8AIVAdd3Ch0aZgDWEAQYBCABEgJO2_D_BwE.

[67] s.r.o., E. *Relé 4 kanály 5V s optočlenem* [online]. ECLIPSERA s.r.o., . [cit. 2021.03.07]. Dostupné z: https://dratek.cz/arduino/2190-rele-4-kanaly-5v-s-optoclenem.html?gclid=EAIaIQobChMI96qzp7ib8AIVw-R3Ch2kkQaQEAQYAyABEgIokvD_BwE.

[68] Systems, E. [online]. Espressif Systems, . [cit. 2021.02.19]. Dostupné z: https://github.com/espressif/arduino-esp32/blob/371f382db7dd36c470bb2669b222adf0a497600d/tools/sdk/esp32s2/include/soc/esp32s2/include/soc/soc.h.

[69] Systems, E. [online]. Espressif Systems, . [cit. 2021.02.19]. Dostupné z: https://github.com/espressif/arduino-esp32/blob/371f382db7dd36c470bb2669b222adf0a497600d/tools/sdk/esp32s2/include/soc/esp32s2/include/soc/rtc_cntl_reg.h.

[70] Systems, E. *arduino-esp32* [online]. Espressif Systems, . [cit. 2020.10.05]. Dostupné z: https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi.

[71] tech, A. *Arduino projekty* [online]. Arduino tech. [cit. 2020.08.09]. Dostupné z: https://www.arduinotech.cz/produkt/usb-uart-ftdi-prevodnik/.

[72] Technology, A. T. [online]. Ai Thinker Technology, . [cit. 2020.05.13]. Dostupné z: https://loboris.eu/ESP32/ESP32-CAM/20Product/20Specification.pdf.

[73] Technology, C. *RFC 7252 Constrained Application Protocol* [online]. Coap Technology, . [cit. 2020.9.16]. Dostupné z: https://coap.technology.

[74] Technology, C. *RFC 7252 Constrained Application Protocol* [online]. Coap Technology, . [cit. 2020.9.16]. Dostupné z: https://coap.technology/spec.html.

[75] Telegram. *Telegram* [online]. Telegram. [cit. 2020.11.19]. Dostupné z: https://telegram.org.

[76] Gavin Electronic Technology Co., L. [online]. Xi 'an Gavin Electronic Technology Co., Ltd. [cit. 2021.01.10]. Dostupné z: https://www.gaimc.com/Uploads/Download/Temperature/GTS200.pdf.

[77] *Zigbee Technical Presentation* [online]. Zigbee Alliance. [cit. 2020.9.23]. Dostupné z: https://zigbeealliance.org/wp-content/uploads/2019/12/Zigbee-Technical-2019.pptx.