TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Inteligentní domácnost – domácí automatizace

Diplomová práce

Studijní program:	N2612 Elektrotechnika a informatika
Studijní obor:	Informační technologie
Autor práce:	**Bc. Jaroslav Vondrák**
Vedoucí práce:	Ing. Tomáš Martinec, Ph.D. Ústav mechatroniky a technické informatiky

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Inteligentní domácnost – domácí automatizace

Jméno a příjmení: **Bc. Jaroslav Vondrák**
Osobní číslo: M19000161
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Seznamte se se systémy domácí automatizace typu IFTTT (If This Then That) a možností implementace takové funkčnosti do stávajícího vlastního systému inteligentní domácnosti.
2. Navrhněte i další možnosti rozšíření tohoto vlastního systému inteligentní domácnosti o další hardwarové prvky (senzory, řídicí moduly) a softwarová rozšíření (např. možnost integrace kamerového systému nebo možnost definice vlastních maker a profilů).
3. Vytvořte prototyp alespoň jednoho nového hardwarového modulu a implementujte navržená softwarová rozšíření.

Rozsah grafických prací: dle potřeby dokumentace
Rozsah pracovní zprávy: 40–50
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] Inteligentní řízení objektů – zabezpečení, provoz, efektivita [online]. [cit. 2019-04-18]. Dostupné z: https://www.stavebnictvi3000.cz/clanky/inteligentnirizeni-objektu-zabezpeceni-provoz-efektivita

[2] 802.11 Wireless Standards [online]. Dostupné z: http://www.pearsonitcertification.com/articles/article.aspx?p=1329709&seqNum=4

[3] NodeMCU a jeho verzie: doska s Wi-Fi čipom ESP8266 [online]. Dostupné z: https://www.root.cz/clanky/nodemcu-a-jeho-verzie-doska-s-wi-fi-cipom-esp8266/

[4] Eleven internet of things iot protocols you need to know about. In: Rs-online [online]. [cit. 2018-03-13]. Dostupné z: https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about

Vedoucí práce: Ing. Tomáš Martinec, Ph.D.
Ústav mechatroniky a technické informatiky

Datum zadání práce: 9. října 2020
Předpokládaný termín odevzdání: 17. května 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 9. října 2020

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

13. května 2021 Bc. Jaroslav Vondrák

Poděkování

Rád bych poděkoval Ing. Tomášovi Martincovi Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování této práce.

Anotace

Práce se zabývá rozšířením IoT-HS systému pro chytré domácnosti o službu IFTTT. Této službě je věnována celá teoretická část, ve které je detailně popsáno, jak tuto službu využívají jiné systémy.

Byly navrženy nové metody rozšiřování systému o nové funkce. Ovladače jsou novým způsobem, jak do systému přidat zařízení od komerčních systémů jako je např. Philips-Hue nebo Fibaro. Tato zařízení lze pak využívat v rámci systému a ovladače zajišťují abstrakci těchto zařízení. Pro otestování funkčnosti IFTTT služby byl navržen jeden zásuvný modul a celkem tři ovladače pro komunikaci s řídicími jednotkami IoT-HS a zařízeními od komerčního systému Philips-Hue. Dále byly navrženy a realizovány nové řídicí jednotky pro IoT-HS systém.

Klíčová slova

Inteligentní domácnost, automatizace, IFTTT, IoT

Annotation

This work is about improvements of IoT-HS system and how to add IFTTT service to this system. How others commercial systems are using this service is described in theoretical part.

New methods of extending IoT-HS system with new functions have been developed. Drivers are a new way to add devices from commercial systems such as Philips-Hue or Fibaro to the IoT-HS system. These devices can then be used within the system and drivers provide the abstraction of these devices. To test the functionality of the IFTTT service, one plug-in module and three drivers were designed for communication with IoT-HS control units and with Philips-Hue devices. Furthermore, new control units for the IoT-HS system were designed and implemented.

Key words

Smart home, automation, IFTTT, IoT

Obsah

Seznam ilustrací

Seznam tabulek

Seznam použitých zkratek a symbolů

IFTTT – If This Then That
ZM – Zásuvný Modul
API – Application Programming Interface, Rozhraní pro programování aplikací
TCP – Transmission Control Protocol, protokol kontroly přenosu
TAPI – TCP API
GUI – Graphical User Interface, Grafické uživatelské rozhraní
JSON – JavaScript Object Notation
IoT – Internet of Things, Internet věcí
Bluetooth LE – Bluetooth Low Energy
IoT-HS – Internet of Things – Home System

Pokud zkratka nemá připsaný český překlad, znamená to, že oficiální český překlad neexistuje nebo je překlad pro danou zkratku nepřesný a nevystihuje tak její přesný účel nebo funkci

1. Úvod

Domácí automatizace je jistě pojem, pod nímž si každý představí něco trochu jiného. Pod tímto pojmem si může někdo jako první představit různá zařízení, která fungují autonomně a vykonávají předem určenou činnost. Můžeme si také představit množinu zařízení, která spolupracují, předávají si data a na základě těchto dat se provádí různé automatické funkce. Mnoho uživatelů se spíše setkává s pojmem inteligentní domácnost. Slovo „inteligentní“ je v dnešní době, z marketingových důvodů, používáno ve velké míře. Ovšem ne každé zařízení má právem ve svém názvu „inteligentní“ [1]. Aby bylo možné použít toto slovo, musí zařízení splňovat určité požadavky. Jeden z takových požadavků je, aby zařízení fungovalo, v jisté míře, samostatně. Slovem samostatně se v tomto projektu myslí to, aby zařízení provádělo automatické funkce např. na základě události nebo v nastavený čas. Existují systémy, které mají centrální řídicí jednotku (tzv. bridge). K této jednotce se připojují ostatní zařízení a pomocí této centrální jednotky může např. uživatel ovládat zařízení a měnit jejich nastavení. Vzhledem k tomu, že všechna zařízení jsou součástí jednoho velkého systému, je možné, aby spolu spolupracovala a uživatel tak může sám vytvářet automatické funkce pomocí různých druhů zařízení (pohybové senzory, chytré žárovky, chytré zásuvky, termostaty, zámky atd.). Jelikož zařízení fungují v takovém systému autonomně, vykonávají různé funkce na základě událostí nebo předdefinovaných profilů a tento systém je součástí domácnosti, můžeme tuto domácnost označit jako „inteligentní domácnost“.

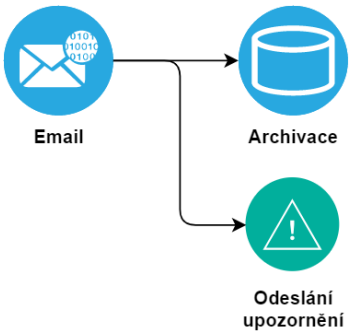
Tato práce míří právě tímto směrem, tedy vytvořit systém a služby, respektive bridge (v této práci nazván jako server), který z běžné domácnosti vytvoří inteligentní domácnost. Pomocí tohoto systému si uživatel může vytvářet své vlastní automatické funkce nebo profily. Lze využívat zařízení od různých výrobců a ta spojit právě s tímto systémem. Pro pokročilé uživatele jsou připraveny knihovny, díky kterým je možné psát své vlastní zásuvné moduly (dále jako ZM) nebo ovladače pro řízení různých zařízení a rozšiřovat tak systém o své vlastní nápady a funkce.

Server využívá službu domácí automatizace typu IFTTT (If This Then That). Právě pomocí této služby je možné, aby si uživatel vytvořil libovolnou automatickou funkci, kde může určit podmínku a co se stane, pokud podmínka platí. Pokud podmínka platí, může uživatel např. změnit nastavení jinému zařízení, spustit alarm, poslat e-mail, rozsvítit světla, odeslat upozornění do telefonu a další pokročilé funkce, které jsou detailně popsány v následujících kapitolách. Aby bylo možné otestovat funkčnost této služby, byly vytvořeny ovladače pro ovládání pohybových senzorů a různého osvětlení. Dále byl vytvořen příklad ZM, řídicí jednotky pro senzory a další rozšíření, díky kterým lze realizovat inteligentní domácnost.

2. Úvod do IFTTT automatizace

Zkratkou IFTTT se označují služby, které uživateli pomáhají definovat automatizované funkce v rámci nějakého systému, domácnosti nebo jiných služeb. Výhodou této služby je to, že uživatel nemusí mít žádné pokročilé zkušenosti s programováním nebo vytvářením složitých automatických funkcí pomocí skriptů nebo jiných metod [2]. Ve většině služeb, které mají implementovanou službu IFTTT, je zmíněno anglické slovo „Applet“. Tímto slovem se většinou označuje podprogram, který je součástí větší aplikace (v minulosti bylo slovo applet často spojováno s programováním v Javě) [3]. V mnoha aplikacích se uživatel může setkat se slovem „plugin“. Ve službě IFTTT je slovo Applet označení pro vytváření automatických funkcí (někdy označováno i jako „recipes“, recepty). Jelikož applet nemá v českém jazyce přesný překlad, bude v této práci toto slovo překládáno jako „scénáře“. Scénář definuje přesný postup, co se má stát, pokud se něco stane.

Existují dvě základní rozdělení, jak lze scénář vytvořit. Prvním způsobem je pomocí různých obrázků. Uživatel má na výběr určitou množinu ikon, které může různě pospojovat a tím vytvořit podmíněné akce. Tuto metodu využívá např. česká služba Integromat.



Služba Integromat nabízí spoustu pokročilých funkcí, díky kterým lze vytvářet jednoduché nebo složité scénáře. Na obrázku 1 je příklad, jak je takový scénář graficky znázorněný. Tímto scénářem byla vytvořena automatická funkce, která má následující úkoly. Pokud se v e-mailové schránce objeví nová zpráva, tato zpráva se archivuje a odešle se upozornění např. do chytrého telefonu.

Obrázek 1 - vytvoření scénáře pomocí obrázků

Spojování různých ikon se může zdát na první pohled jako jednoduché řešení pro vytváření scénářů. Může nastat ale případ, kdy je scénář opravdu rozsáhlý a pokud scénář obsahuje např. více jak 20 ikon, tak orientace v takovém scénáři může být pro běžného uživatele velmi obtížná. Proto služba Tallyfy nabízí jiné řešení, a to vytváření scénářů pomocí textu. Uživatel si jednoduše v nabídnutém seznamu vybere zařízení, u kterého se má hlídat podmínka, v dalším seznamu zvolí podmínku atd. Stejným směrem se dala i služba ifttt.com. Tato služba se stala velmi známou a mnoho zdrojů na ní odkazuje právě se zkratkou IFTTT. Aby nedocházelo k záměně zkratky se službou, bude vždy k této službě připsáno „.com“.

Ifttt.com podporuje velmi velké množství aplikací [4]. Tuto službu lze využít i pro více účelů než jen pro chytrou domácnost. Je možné např. vytvořit scénář, který přidá tu samou fotografii na Instagramový účet, která byla přidána na Facebookový účet. Nebo pokud byla fotografie přidána na Facebookový profil, tak se tato fotografie zálohuje na cloud.

V dalších kapitolách je popsaná služba ifttt.com. Dále jsou popsány další příklady systémů, které poskytují automatizaci, jejich výhody, nevýhody a limity. Nakonec je provedeno srovnání komerčních systémů s IoT-HS.

2.1. Popis služby IFTTT.com

Ifttt.com je webová služba, která umožňuje vytvářet scénáře pomocí jiných webový služeb nebo systémů. Aby bylo možné využít ifttt.com službu, musí být tato služba implementována do daného systému a využívat tedy API této služby. Např. pokud uživatel bude vlastnit zařízení od Philips-Hue nebo Fibaro. Každý systém má své vlastní zařízení a nelze je nijak propojit (např. aby pohybový senzor od Fibaro rozsvítil světlo od Philips-Hue). Pokud tyto systémy mají implementovanou službu ifttt.com, je možné pomocí této služby propojit tyto dva systémy, aby pracovaly jako jeden celek.

Scénáře jsou vytvářeny pomocí aplikace pro chytré telefony (Android a iOS). Do aplikace je možné se přihlásit pomocí Facebook účtu nebo Google účtu. V hlavním menu jsou dostupné aplikace, které podporují tuto službu (např. Amazon Alexa, Dropbox, Facebook, Philips Hue, Fibaro, Nuki atd.). Scénáře může uživatel sám vytvářet nebo jsou k dispozici předpřipravené scénáře, které jsou nejčastěji používané ostatními uživateli. Např. lze zvolit scénář, který každý večer odešle uživateli informaci o počasí na další den jako zprávu na Facebook.

Obrázek 2 - GUI služby ifttt.com

Na obrázku 2 je ukázáno GUI aplikace pro chytrý telefon. Uživatel má možnost vidět své scénáře, upravovat je nebo vytvářet. Stejné funkce pak poskytuje webové rozhraní.

Pokud chceme, aby aplikace podporovala službu ifttt.com, je možné využít REST API. Pro pochopení principu fungovaní této služby je nutné zavést si 7 důležitých bodů, respektive co vše se musí implementovat, aby bylo možné scénář vytvořit. Jak sám název IFTTT napovídá, prvním nastaveným údajem je podmínka (v této službě nazvaná jako **Triggers**, což lze přeložit jako událost, respektive event). Uživatel si tedy může vybrat událost např. detekce pohybu. Poté je vyžadováno tzv. **Trigger Fields**. Což může být textové pole, do kterého uživatel zadá např. název senzoru u kterého má být detekována událost. Jako další musí uživatel zadat **Ingredients**, přísady. Tato možnost se využívá v případě, kdy uživatel zvolí jako událost např. již zmiňované zálohování fotografií. Tedy do přísad se přidá URL odkud se mají fotografie stahovat. Pro zadání dodatkových informací pro událost slouží **Queries**, dotazy. Pomocí dotazů se může událost, respektive podmínka doplnit o dodatečné informace. Formuláře, ve kterých je možné vyplnit tyto informace se nazývají **Query Fields**. Po nastavení podmínky je nutné specifikovat akci. Tento krok se nazývá **Action**. Akce může být např rozsvícení světla, zálohování fotek nebo poslání oznámení do telefonu. Formulář pro zadání akce se nazývá **Action Fields**.

Pokud aplikace, do které chceme implementovat ifttt.com, má již vytvořené své API, je možné zvolit dvě strategie [5], jak tuto aplikaci připojit ke službě ifttt.com. Jedna z možností je vytvořit tzv. prostředníka mezi aplikací a ifttt.com. Tento prostředník se připojí k API navržené aplikace, od které bude přijímat data. Tato data pak převede do požadovaného formátu a odešle službě ifttt.com. Druhá možnost je implementovat tuto službu přímo do aplikace a přizpůsobit API tak, aby bylo možné se službou ifttt.com komunikovat.

Jak probíhá připojení komerční aplikace ke službě ifttt.com bude ukázáno na systému Philips-Hue (rozsvícení světla) a SkylinkNet (senzor pohybu).

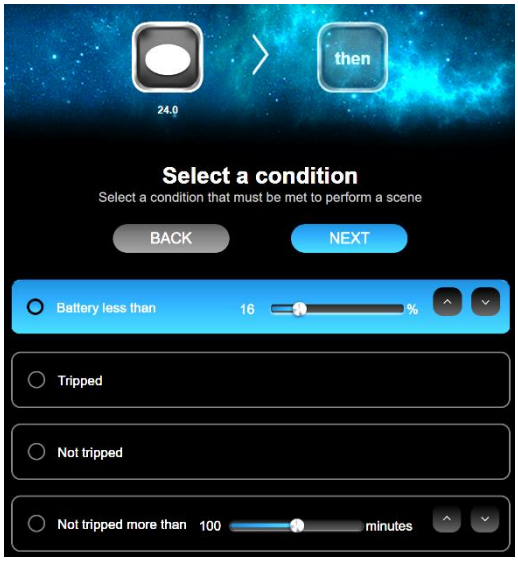
Uživatel si nejdříve vytvoří učet na webových stránkách služby. Na tento účet si pak může přidávat různá rozšíření. Uživatel si musí nejdříve přidat SkylinkNet (webová služba) do svého účtu. To je provedeno tak, že uživatel se přihlásí do této služby pomocí svého SkylinkNet účtu. Tento účet se pak propojí s účtem ifttt.com. Stejný postup je pak proveden s Philips-Hue. Tedy přihlásit se pomocí Philips účtu do Philips-Hue služby, která se propojí s ifttt.com. Nyní jsou všechny tři služby propojené a lze vytvořit scénář.

Službu ifttt.com lze využívat bezplatně s jistým omezením. Pokud je využívána licence Standard, je možné vytvořit 3 své vlastní scénáře a využívat neomezený počet již vytvořených scénářů. Pokud se využívá Pro licence, lze vytvořit neomezený počet svých vlastních scénářů s dotazy (Queries) a další s pokročilejší logikou. V Pro verzi je také zaručeno provedení scénářů v reálném čase (minimální zpoždění).

2.2. Popis systému Fibaro

Velmi silným nástrojem pro vytvoření chytré domácnosti je Fibaro systém. Aby bylo možné využívat Fibaro zařízení, je nutné zakoupit jeden z druhů tzv. Home Center. Ten je dostupný ve dvou variantách, a to Home Center 3 Lite a Home Center 3 (dále nazýván jako server). K tomuto serveru lze připojit zařízení, která komunikují pomocí protokolu Z-Wave Plus, Zigbee, Bluetooth LE a WiFi. Pro vytvoření automatických funkcí, respektive scénářů, není třeba server připojovat k jiné cloudové službě. Veškeré scénáře, přidávání zařízení, přidávání místností nebo jiná konfigurace, se musí provádět přes webové rozhraní. Je také k dispozici aplikace pro chytré telefony, ve které lze pouze sledovat stav zařízení, spouštět scénáře nebo ovládat zařízení. Fibaro spoléhá na svůj vlastní systém automatizace, který nabízí hned ve třech následujících variantách.

Vytvoření Magické scény



Obrázek 3 – Fibaro vytvoření scénáře

Vytvoření tzv. magické scény (v systému nazváno jako Magic scene) je typ vytváření scénářů pomocí obrázků. Jedná se o nejjednodušší metodu vytváření scén v tomto systému. Uživatel si jednoduše vybere podmíněné zařízení, zvolí podmínku a poté vybere akci, která se má stát, pokud je podmínka splněna (např. aktivovat jinou scénu).

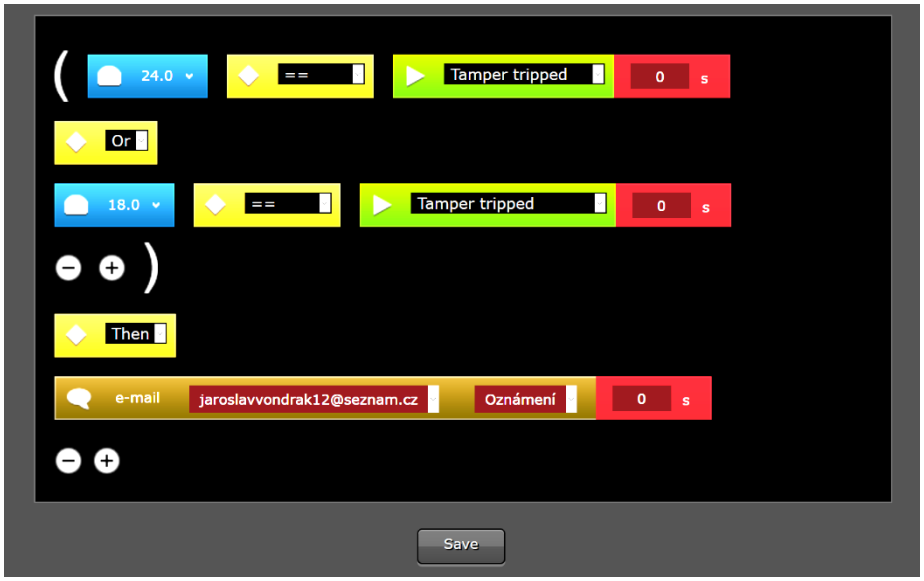
Na následujícím obrázku je příklad vytvoření magické scény. V prvním kroku byl přidán senzor, který detekuje vodu (např. pokud v koupelně dojde k úniku vody, senzor na tuto událost upozorní uživatele). V další kroku se nastaví podmínka (pokud má baterie méně jak 16 %). V posledním kroku se vybere, co se má stát, pokud

podmínka platí. Pro tento příklad se odešle upozornění do chytrého telefonu. To je provedeno tak, že uživatel zvolí ikonu obálky (Notifications) a poté zvolí název telefonu, který byl spárován s tímto systémem.

Vytvoření scény pomocí grafických bloků

Druhou variantou je vytvoření scény pomocí grafických bloků. Jedná se o kombinaci textového nastavení scény a grafického nastavení. Tato varianta vytvoření scény je už pro středně pokročilejší uživatele, kteří chtějí vytvořit trochu komplexnější scénu. Jako podmínku lze vybrat zařízení, počasí, GPS, stav bezpečnostních kamer a mnoho dalšího. Lze určit v jaký čas se mají rozsvítit světla a po jakou dobu mají být rozsvícená nebo definovat čas od kdy do kdy mají svítit. Je na výběr také mnohem více možností nastavení zařízení jako např. termostaty atd.

Oproti magickým scénám se vytváření pomocí grafických bloků liší tím, že je zde možnost přidání více podmínek. Na obrázku 3 je vidět, že podmínku lze zadat pouze jednu, stejně tak i nastavení nějakého zařízení.



Obrázek 4 – Fibaro vytvoření scénáře pomocí blokového diagramu

Na obrázku 4 je ukázán příklad vytvoření scény, která odešle e-mail, pokud alespoň jedna podmínka platí. Modrá barva reprezentuje zařízení vybrané pro danou podmínku, žlutá barva výběr operátoru, zelená barva pro výběr statusu zařízení a červená je časové zpoždění. Pomocí ikony plus může uživatel přidat další podmínky s výběrem „Or“ nebo „And“. Ikona plus, která se nachází ve spodní části, slouží pro přidání dalšího nastavení zařízení nebo jiné činnosti.

Vytvoření scény pomocí programovacího jazyku LUA

Pro velmi pokročilé uživatele je k dispozici vytváření scén pomocí programovacího jazyku LUA. Díky tomuto způsobu lze vytvářet scény jakýmkoliv způsobem a přesně podle představ uživatele. Výměna dat mezi zařízením a serverem probíhá pomocí JSON formátu. LUA nabízí jednoduché řešení, jak získat informaci ohledně jakéhokoliv zařízení, které je přidáno do systému.

Např. příkazem Fibaro:getValue() se dá získat jakákoliv hodnota daného zařízení (např. jas zařízení s id 25.

```
Local value = Fibaro:getValue(25, 'brightness')
```

Hodnotu lze využít v podmínce a popřípadě nastavit zařízení.

```
if (tonumber(value) > 70) then
  fibaro:call(24, 'turnOff')
end
```

V tomto příkladu se vypne zařízení s id 24, pokud je jas větší jak 70 [7].

Fibaro systém nabízí mnoho možností, jak vytvořit scény, zařízení, místnosti atd. Tento systém není třeba napojovat na žádnou cloudovou službu ani nemusí mít přístup k internetu pro svou funkci. Systém také nabízí možnost rozšíření pomocí pluginů od firem třetích stran a připojení zařízení od různých firem (pouze Home Center 3). Home Center 3 lze v dnešní době pořídit kolem 15 000 Kč. Verzi Lite kolem 4 000 Kč.

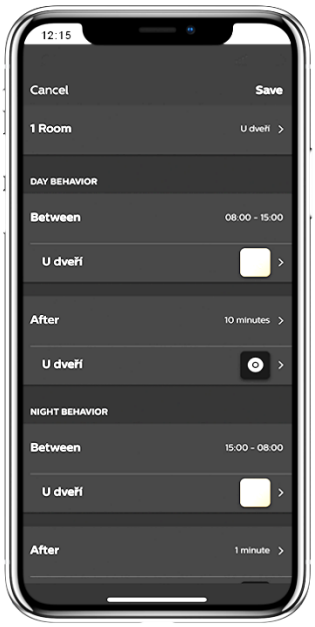
2.3. Popis systému Philips-Hue

Philips-Hue je jedním z dalších systémů pro chytrou domácnost. Tento systém se zaměřuje převážně na osvětlení a nabízí chytré žárovky, lustry, lampy, pohybové senzory, ovladače atd. Veškeré ovládání nebo nastavení systému je možné provádět přes aplikaci, která je dostupná pro iOS nebo Android. Tento systém nenabízí tak silné nástroje jako tomu bylo např. u Fibaro systému. Je ale stále možné nastavit základní podmíněné funkce. Aplikace poskytuje funkce nazvané jako routines, rutiny. Pomocí rutiny může uživatel nastavit funkci „Home & Away“, která sleduje pohyb uživatele a když je uživatel daleko od domu (Away) světla se automaticky zhasnou. Pokud se uživatel vrací domů (Home) jsou světla automaticky zapnuta. Uživatel může také nastavit funkci „Go to sleep“, která ve večerních hodinách zvolí takové osvětlení, které nemá rušivé účinky na organismus a může tak zlepšit lidské vnitřní biorytmy [8]. Stejně tak lze nastavit „Wake up“ funkce, která slouží pro zpříjemnění vstávání.

Pro trochu pokročilejší funkce má Philips-Hue k dispozici tzv. Hue Labs. Jedná se o rozšíření tohoto systému o další chytré funkce. Pomocí Hue Labs už lze vytvářet, i když omezeně, scénáře (v aplikaci pojmenováno jako Formulas, vzorce).

Hue Labs poskytují různé funkce od automatizace po zpříjemnění pobytu tím, že systém sám nastavuje různou intenzitu osvětlení na základě událostí. Uživatel má možnost nainstalovat různé doplňky jako např. „Sensing the weekend“, který dovoluje nastavit různá nastavení pohybových senzorů pro každý den v týdnu jinak. Pokud uživatel bude požadovat pokročilejší funkce automatizace je třeba využít aplikace třetích stran nebo Philips-Hue napojit na ifttt.com službu. V kapitole 2.1. byl popsán jednoduchý postup, jak lze tuto službu

s Phlips-Hue spojit. Jelikož všechna zařízení Philips-Hue komunikují pomocí protokolu ZigBee, je možné je napojit na Fibaro server a využít tak automatizaci tohoto systému.

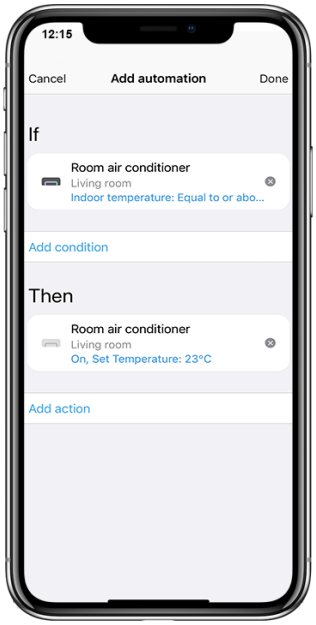


Obrázek 5 - Philips Hue aplikace

Na obrázku 5 je ukázán příklad grafického rozhraní aplikace Philips-Hue. Na uvedeném příkladu je ukázáno, jak nastavit rozsvícení světla u vchodových dveří, když je zaznamenán pohyb. Jak lze vidět, jsou k dispozici tři možnosti nastavení osvětlení v určitý čas. Samozřejmě nechybí možnost nastavení intervalu, po kterém se světlo vypne.

Z příkladu je patrné, že základní aplikace neposkytuje tolik možností vytváření scény a spíše záleží na tom, jaké možnosti podmínky a nastavení určí vývojáři.

2.4. Popis systému Samsung SmartThings



Obrázek 6 - Samsung SmartThings aplikace

Samsung SmartThings posouvá hranici v domácí automatizaci zase o trochu výše. Tento systém nabízí celou řadu dalších zařízení, pro která lze vytvářet scénáře (chytré televize, ledničky, pračky, klimatizace, reproduktory atd.). Samsung kombinuje použití bridge, respektive serveru a cloudu zároveň. Pokud uživatel nemá k dispozici server a všechna zařízení jsou připojená k internetu, lze automatizaci vytvořit pomocí cloudu, ke kterému se tato zařízení automaticky připojí. Uživatel si musí pouze vytvořit Samsung účet ve kterém si přidá daná zařízení a může vytvářet scénáře pomocí aplikace pro Android nebo iOS (v aplikaci jsou scény brány jako skupiny do kterých se přidají zařízení a tyto scény lze volat v automatických funkcích, které si uživatel vytvoří). Na obrázku 6 je ukázán příklad, jak si v aplikaci může uživatel nastavit podmínku a akci. Aplikace už nabízí mnohem více možností, než tomu bylo Philips-Hue (možnosti podmínky a akce jsou závislé na daném zařízení). Podmínky je možné větvit a stejně tak nastavit více akcí, když je podmínka splněna.

Pokud uživatel bude požadovat komplexnější scénáře, je možné SmartThings připojit k systému Fibaro.

2.5. Hlasoví asistenti a jejich využití v chytré domácnosti

Hlasoví asistenti se v poslední době stali oblíbenou metodou [9], jak ovládat chytrou domácnost bez použití např. telefonu nebo různých ovladačů. Tento způsob jistě zasahuje i do problematiky IFTTT automatizace. Např. služba ifttt.com nabízí spojení s Google Assistant, díky kterému lze ovládat různá zařízení tak, že se vytvoří scéna a do podmínky se vloží hlasový příkaz. Hlasový asistent umí např. vypnout nebo zapnout osvětlení, sdělit aktuální počasí nebo vyhledat různé informace na internetu a ty pak předložit uživateli.

Aby bylo možné využít hlasového asistenta, je nutné si pořídit bridge (u hlasových asistentů se většinou označuje jako chytrý reproduktor). Nejpopulárnější jsou hlasoví asistenti od firmy Apple a Google. Apple poskytuje asistenta nazvaného jako Siri. Google poskytuje asistenta Google Home a na trhu je ještě k dispozici např. hlasový asistent od Amazonu, Alexa.

Siri, Google a Alexa poskytují API, pomocí kterého se aplikace může spojit s daným asistentem. Např. Philips-Hue má možnost připojení k Siri a pak pomocí jednoduchého příkazu jako např. „Hej Siri, turn on all lights“ lze rozsvítit všechna světla v domácnosti. Bohužel Siri, Google ani Alexa zatím nemají plnou podporu českého jazyka.

Hlasoví asistenti jistě zpříjemní ovládání chytré domácnosti. Co když je ale uživatel mimo domácnost a chce si stále s domácností „promluvit“. Na řadě jsou textoví asistenti, kteří jsou schopni si s uživatelem psát zprávy ohledně stavu domácnosti nebo stavu určitých zařízení. Uživatel pak jednoduše může napsat zprávu „Jaký je stav?“ a asistent odepíše např. „vše v pořádku“. Tuto možnost nabízí služba ifttt.com.

2.6. Porovnání IoT-HS s ifttt.com, Fibaro a Philips Hue

Jelikož služba ifttt.com je používaná ve velké míře a nabízí spoustu možností, je na místě ji porovnat s IoT-HS systémem. Nejdůležitější rozdíl mezi IoT-HS a ifttt.com je ten, že ifttt.com je cloudová služba. Tedy vyžaduje neustálé připojení k internetu jak např. Philip-Hue bridge tak ostatních IoT zařízení. V kapitole 2.1. byl uveden příklad spojení více služeb, kde bylo potřeba spojit celkem tři cloudové služby, aby bylo možné propojit zařízení mezi sebou a vytvořit scénář. Ve chvíli, kdy jedna služba přestane fungovat, scénář již nelze využívat. V případě, kdy není dostupné připojení k internetu, scénář opět nelze použít. IoT-HS systém je v tomto ohledu více uzavřený. Server lze spustit na lokálním počítači a připojení k internetu není třeba. Vše lze ovládat lokálně např. pomocí ovládacího panelu. Scénáře fungují i v případě, kdy se nelze připojit k internetu. Samozřejmě může být přidán ovladač, který využívá webovou službu. Pokud ale tento ovladač nebude schopný se připojit k internetu, nebude to mít vliv na chod systému a jiné scénáře (které nejsou závislé na tomto ovladači) se mohou spouštět.

Dalším rozdílem jsou ovladače a ZM. Ovladač by se mohl pokládat právě za prostředníka mezi aplikací a ifttt.com. Ovladače v IoT-HS jsou ovšem instalovány přímo na straně serveru a jsou k dispozici knihovny, které usnadňují celkový vývoj ovladače nebo ZM. Dalším rozdílem jsou klientské části ZM nebo ovladače, které poskytují další způsob, jak uživateli

prezentovat data nebo zjednodušit ovládání. Pokročilí uživatelé si mohou IoT-HS systém zcela přizpůsobit ke své představě. Pro méně pokročilé uživatele je tu možnost rozšíření systému o různé ovladače nebo ZM od jiný uživatelů a poskládat si tak systém opět podle vlastních představ.

Systém Fibaro nabízí velmi podobné funkce a služby jako IoT-HS. Nenabízí ovšem možnost vytváření svých vlastních klientských částí, tedy vytvoření nového grafického rozhraní, které by šlo propojit s Fibaro systémem. IoT-HS také nabízí jednodušší vývoj ZM a ovladačů, a především není třeba tento ZM nebo ovladač nikam registrovat, aby ho bylo možné do systému nainstalovat. Uživatel má plnou kontrolu, co se do IoT-HS serveru přidá nebo nainstaluje. Server není nijak hardwarově závislý a lze ho spustit na jakémkoliv počítači, serveru nebo jednodeskovém počítači. U Fibaro ani Philips-Hue toto není možné. Je vždy nutné zakoupit bridge (Home Center nebo Philips-Hue bridge).

Philips-Hue nenabízí žádný pokročilý systém automatizace. Proto právě Philips-Hue byl vybrán jako komerční systém, pro který byl navržen a implementován ovladač pro spojení s IoT-HS systémem. Díky tomuto ovladači je možné využívat Philips-Hue zařízení v rámci celého systému a vytvářet pokročilejší scénáře.

3. Implementace služby IFTTT do IoT-HS systému

Doposud IoT-HS systém nedisponoval žádnou funkcí, díky které by si mohl uživatel nastavit automatizované funkce. Veškerá automatizace a správa zařízení byla prováděna ZM. Na první pohled by se mohlo zdát, že takové řešení usnadní programování nových ZM. Ovšem čím více ZM systém má, tím méně jsou mezi sebou kompatibilní a při vytváření nového ZM se musí brát ohled na již vytvořené nebo přidané ZM do systému. Samozřejmě pokud se vytváří ZM, který spravuje jen určitá zařízení, je toto řešení jistě jednodušší. Většinou ale uživatel chce, aby zařízení mezi sebou spolupracovala. Např. pokud pohybové čidlo zaznamená pohyb, musí se rozsvítit světlo. Pokud takovou akci bude uživatel vyžadovat od starého IoT-HS systému, musí jeden ZM obstarávat komunikaci s pohybovým čidlem a zároveň i se světlem. Pokud by se do systému přidalo jiné světlo (např. od jiného výrobce, a tedy spravovaný jiným ZM), bylo by nemožné dané světlo rozsvítit na základě stavu pohybového čidla z jiného ZM.

Tuto problematiku řeší služba typu IFTTT na globální úrovni. Implementace této služby, do již existujícího systému, vyžaduje jisté úpravy a implementaci nových služeb a funkcí. Službu IFTTT poskytuje server a ten se také stará o vykonávání nadefinovaných automatických funkcí. Proto právě na straně serveru se muselo udělat mnoho úprav za účelem přidání této služby.

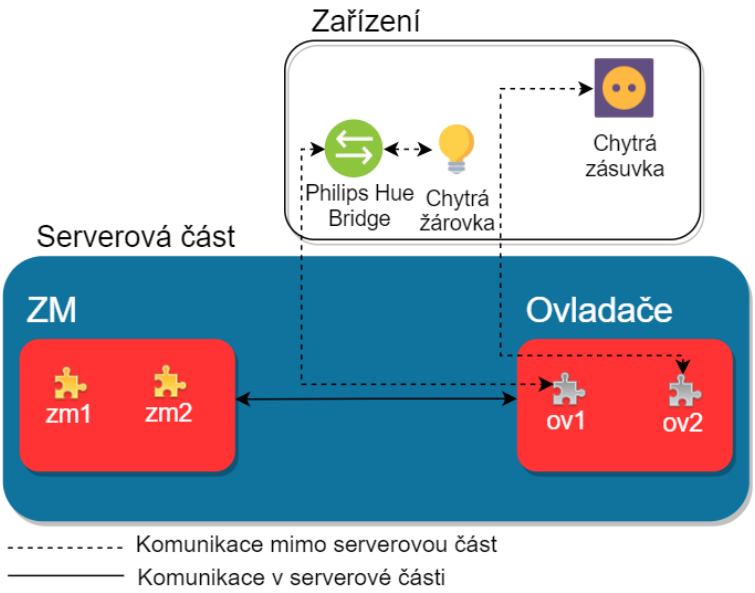
V dalších kapitolách bude upřesněna implementace služeb a funkcí, které vedou k přidání IFTTT služby jak na straně serveru, tak na straně klienta.

3.1. Server (bridge) – Popis a zavedení ovladačů

Hlavním cílem přidání IFTTT služby je využití všech přidaných zařízení v rámci celého systému. Tedy pokud se přidá jakékoliv zařízení do systému, mohou daná zařízení využívat všechny ZM nebo server samotný. Za tímto účelem jsou do systému instalovány ovladače. Ovladače jsou novým způsobem, jak vytvořit komunikaci mezi serverem a různými HW zařízeními. Ovladač se tedy stará o způsob komunikace s daným zařízením.

Příklad využití ovladačů:

Předpokládejme dvě různá zařízení. Jedno zařízení od firmy Philips Hue a druhé od IQTECH. Philips Hue využívá bridge, který pomocí technologie ZigBee komunikuje s různými zařízeními (např. žárovkou). IQTECH má způsob komunikace přes protokol Wi-Fi (např. chytrá zásuvka). Pro každé zařízení se tedy využívá jiný způsob komunikace. Každé zařízení má jinak pojmenované atributy, kterými lze dané zařízení ovládat. Cílem ovladačů je jistá abstrakce těchto zařízení. Dejme tomu, že žárovka má atribut „On“, který nabývá hodnoty „true“ nebo „false“ (tedy rozsvíceno nebo zhasnuto). Chytrá zásuvka může mít atribut např. „state“, který nabývá hodnot „1“ nebo „0“.



Obrázek 7 - základní princip ovladačů

Každý ZM by tedy musel mít informaci o daném zařízení a jeho atributech. Atribut „On“ a atribut „state“ přitom mají v zásadě tu samou funkci. Tedy vypínají nebo zapínají dané zařízení. Standard tedy uvádí, že pokud se jedná o zapínání nebo vypínání zařízení, je atribut pojmenován „state“ a nabývá hodnot „true“ nebo „false“. Ovladač se tedy spojí se zařízením, přepíše daný atribut a pokud systém požádá o stav zařízení, bude ovladač informace distribuovat standardizovaným způsobem. Nyní tedy každý ZM, který se řídí standardem, může ovládat dané zařízení nebo číst jeho nastavení. Standardům, kterými by se měl řídit každý ZM a ovladač, bude věnována samostatná kapitola.

Na uvedeném příkladu bylo ukázáno, jak jsou ovladače užitečné a jaký je jejich hlavní účel. Samozřejmě ovladače nejsou to jediné, co se na straně serveru muselo přidat, aby bylo možné implementovat IFTTT. Je potřeba zmínit, že ovladače mají (stejně jako ZM) serverovou i klientskou část. Serverová část se stará o komunikaci se zařízením a o předávání informací ostatním ZM nebo serveru. Klientská část slouží pro správu zařízení uživatelem. Tedy uživatel může zařízení přidat, odstranit, upravit nastavení (např. IP adresa, název zařízení) nebo kontrolovat stav zařízení (zobrazení uložených hodnot v databázi pomocí JSON formátu). Ovladač ovšem neumožnuje jeho plnou kontrolu (např. u LED žárovek měnit barvy nebo jas). Toto nastavení má na starosti ovládací panel nebo ZM k tomu určený. Ve většině případech, kdy se jedná o standardizované zařízení (v rámci sytému IoT-HS), lze zařízení ovládat rovnou pomocí ovládacího panelu a není potřeba vytvářet navíc ZM. ZM mohou být ovšem užitečné v jiných případech, kdy se jedná o zařízení, které má ojedinělé vlastnosti. Např. nestandardní ovládání, vytváření různých profilů nebo vytvoření přehlednějšího GUI pro daný typ zařízení.

Další důležitou funkcí je služba na sledování událostí (eventů). Tedy služba, která poskytuje registraci sledovaného zařízení. Pokud je tedy požadavek takový, že se mají rozsvítit světla, když senzor zaznamená pohyb, je tuto událost potřeba zaregistrovat do této služby (registrace pohybového senzoru). Ovladač zaznamená, že se změnil relevantní atribut daného zařízení a tuto informaci pošle serveru. Server informaci dále předá službě na sledování událostí, která prověří, zda je zařízení registrováno. Pokud ano, event předá dále

např. ZM nebo službě obstarávající automatizaci. Pokud zařízení není registrováno, event se ignoruje.
Další důležitou službou je automatizace, která se skládá z více služeb. Této problematice bude věnována jedna celá kapitola.

Vzhledem k tomu, že převážná většina IoT systému využívá k předávání dat datový formát JSON, byl i tento projekt zcela předělán tak, aby byl využíván právě formát typu JSON. Pro programování SW byl vybrán programovací jazyk Java s použitím frameworku Spring Boot. Dále jsou připravené knihovny, které usnadňují vývoj ZM a ovladačů. V celé práci je předpokládáno, že je vše programováno právě pomocí frameworku Spring Boot. Tomuto předpokladu budou také přizpůsobeny příklady využití knihoven.

3.2. Server – Popis API

API prošlo výraznou změnou oproti předchozí verzi, a tak je nutné nejprve porozumět komunikaci mezi serverovou částí (bridge) a klientskou částí (ovládací panel), serverovou částí ZM a klientskou částí ZM, serverovou částí ovladače a klientskou částí ovladače atd. Při spuštění serveru je vytvořena služba (ve zdrojovém kódu označováno jako „service“), která má na starosti správu API (třída „TcpApiService“). Po zavedení této služby se zpřístupní API. V tomto projektu bude API označováno pojmem „TCP API“ (dále jako TAPI), jelikož komunikace probíhá pomocí TCP/IP protokolů, nikoliv pomocí HTTP požadavků, jak je tomu u REST API.

TAPI je ve skutečnosti server, který odposlouchává na nastaveném portu. Na tento port se pak mohou připojovat ZM, ovladače a ovládací panely. Služba, která se stará o zavedení TAPI vytvoří celkem dva servery (tedy dva TAPI servery). Jeden TAPI pro ovladače a ZM, druhý TAPI slouží pro komunikaci s ovládacími panely. TAPI pro ovládací panely může být zpřístupněno v rámci lokální sítě. Např. server má IP adresu „192.168.2.20“ a ovládací panel „192.168.2.25“. TAPI pro ZM a ovladače musí být přístupné pouze na adrese „127.0.0.1“, proto aby se k němu mohly připojit pouze ZM nebo ovladače, které jsou instalovány na daném počítači a zamezilo se tak bezpečnostním hrozbám.

Veškerou komunikaci mezi ZM/ovladačem a serverem/klientem zajišťují knihovny. Jedná se o funkce a služby, které zajišťují připojení, obnovení připojení, přístup do databáze, vytváření událostí, volání událostí atd. Tím, jak používat pro vytváření ZM knihovny a jak správně nastavit konfigurační hodnoty, se zabývá následující kapitola.

Jak již bylo výše napsáno, veškeré předávání dat probíhá pomocí formátu typu JSON. TAPI poskytuje celkem 6 základních příkazů.

Všechny níže vypsané příkazy, kromě příkazu „DRIVER COM“, mají jednoduchou strukturu a celkem snadné použití. Příklady požadavků a odpovědí lze nalézt v příloze 1.

Tabulka 1 - TAPI příkazy

Příkaz	Popis
LOG	Zapsání zprávy do logu serveru. (např. error hlášení, warning hlášení atd.)
ECHO	Kontrola spojení s TAPI.
GET DBS	Žádost o přístupová data do databáze.
DRIVER COM	Příkaz určený ovladači/ovladačům
ADD EVENT	Přidání události pro určité zařízení podle ID.
CALL EVENT	Zavolání eventu pro určité zařízení podle ID.

DRIVER COM

Vzhledem k tomu, že ovladače komunikují pouze v rámci lokální sítě (IP adresa „127.0.0.1“), tak se ZM nemůže spojit s ovladačem napřímo nebo využít API daného ovladače. Server tedy musí tvořit prostředníka, jelikož zpřístupňuje TAPI, díky kterému ZM může komunikovat s ovladačem. Tento příkaz lze tedy chápat jako přemostění komunikace mezi ZM a ovladačem.

Pro správnou komunikaci s ovladačem je zapotřebí dodržovat formát odesílaného požadavku.

Příklad:

```
{
  "driver":"LightsDriver",
  "comType":"IoT-HS",
  "deviceType":"Lights",
  "driverCommand":"DRIVER SET DEVICE",
  "command":"DRIVER COM",
  "device":[
      "status":true
  ]
}
```

V příkladu se vyskytují tří klíče, které musí mít ovladač na své straně specifikované (jak správně nastavit konfigurační soubor ovladače lze nalézt v kapitole 3.5.).

Klíč „driver“ obsahuje jedinečný název ovladače. Tento klíč se používá k přímému spojení s daným ovladačem.

Klíč „comType“ (Company Type) specifikuje od jaké firmy, společnosti nebo skupiny je daný ovladač vytvořený. Lze tím také specifikovat, jaká zařízení má ovladač na starosti. Pokud bude hodnota např. „Philips-Hue“, tak ovladač zpřístupňuje zařízení od Philips-Hue (ovladač ale může být vytvořen kýmkoliv).

Posledním klíčem je „deviceType“. Hodnota tohoto klíče specifikuje, jaká zařízení ovladač zpřístupňuje. Pokud je hodnota např. „Lights“, ovladač poskytuje zařízení, která jsou určena

k osvícení místnosti (žárovky, LED pásky, …). Tento klíč by se měl určovat podle stanovené normy, aby byla zajištěna kompatibilita se všemi ZM.

Požadavek by měl obsahovat alespoň jeden z těchto tří klíčů. Může samozřejmě ale obsahovat dva nebo všechny tři tyto klíče. Pokud tedy ZM bude požadovat všechna dostupná světla, stačí přidat do požadavku pouze klíč „deviceType“. Tím je zajištěno, že server kontaktuje všechny ovladače, které poskytují osvětlení a odpověď všech těchto ovladačů odešle zpátky ZM.

Pokud je odeslání dat na server úspěšné, server odešle odpověď s požadovanými daty (např. všechna zařízení, která zajišťují osvětlení). Odpověď obsahuje také klíče „command“ a „answeredTo“, díky kterým ZM rozpozná, že se jedná o odpověď od ovladačů. Klíč „command“ má vždy hodnotu „DRIVER ANS“, tedy označení toho, že se jedná o odpověď od ovladačů. Klíč „answeredTo“ má vždy stejnou hodnotu jako klíč „driverCommand“. Pokud tedy ZM bude požadovat světla od všech ovladačů, „driverCommand“ bude mít hodnotu „DRIVER GET DEVICE“, odpověď pak může vypadat následovně.

Příklad:

```
{
  "command":"DRIVER ANS",
  "answeredTo":" DRIVER GET DEVICE",
  "LightsDriver":[
      "devices": […]
   ],
  "anotherDriver":[
      "devices": […]
   ],
}
```

3.3. Server – Knihovna pro vytváření ZM a ovladačů

Pro vytváření ZM a ovladačů byla vytvořena knihovna „LibraryPluginServerSide“. Knihovna je napsaná v programovacím jazyce Java. Vzhledem k tomu, že ZM a ovladače jsou z hlediska komunikace velmi podobné, lze využít knihovnu v obou případech, pouze s menší úpravou konfiguračních souborů.

Pokud se vytváří nový ovladač nebo ZM, je požadováno vytvoření konfiguračních souborů ve složce „resources“. Jedná se o soubory s předem určeným názvem (analogie z frameworku Spring Boot):

1 - application_IoTHS.properties
2 - application-dev_IoTHS.properties
3 - application-prod_IoTHS.properties

První konfigurační soubor slouží pouze pro upřesnění toho, jaký profil je zvolen. Pokud je ovladač ve vývojové fázi, je profil nastaven následovně.

profile.active=application-dev_IoTHS

Pokud je ovladač nasazený na serveru, profil musí být nastaven na hodnotu „prod“.

profile.active=application-prod_IoTHS

Konfigurační soubory 2 a 3 obsahují následující nastavení (klíče).

Tabulka 2 - přehled klíčů (knihovna)

Klíč	Popis
Pluginconfig.driverMode	Určení, zda se jedná o ovladač nebo ZM. V případě, že je knihovna využívána k vývoji ovladače, je tento klíč nastaven na „true“, jinak „false“.
pluginConfig.driverDeviceType	Určení, jaká zařízení ovladač spravuje (doporučeno používat standardní pojmenování)
pluginConfig.driverDeviceCom	(Company Type) specifikuje od jaké firmy, společnosti nebo skupiny je daný ovladač vytvořen.
pluginConfig.pluginName	Název ZM/ovladače.
pluginConfig.debugMode	Testovací mód.
pluginConfig.ipAddressServer	IP adresa serveru TAPI (vždy by mělo být nastaveno na 127.0.0.1).
Pluginconfig.portServer	Nastavení portu, na kterém odposlouchává server (bridge).
pluginConfig.ipAddressApi	Nastavení IP adresy pro připojení klientské části ZM (**Pouze pro debug mode**).
Pluginconfig.portApi	Nastavení portu, na kterém má serverová část ZM odposlouchávat (**Pouze pro debug mode**).

Předpokládá se, že ke každé serverové části ZM nebo ovladači bude také vytvořena klientská část. Proto knihovna obsahuje služby, které vytvoří TAPI, na které se mohou připojit klientské části. Jak správně nastavit klientskou část ZM nebo ovladače je podrobně vysvětleno v dalších kapitolách.

Připojení k serveru probíhá ve třech krocích. První krok je navázání TCP spojení pomocí socketu (knihovna java.net.Socket). Ve druhém kroku je ověřeno, zda je ZM nebo ovladač správně nastaven pomocí příkazu „CHECK“. Ve třetím kroku se čeká na odpověď od ZM nebo ovladače společně s požadovanými informacemi. Po odeslání příkazu „CHECK“ server očekává odpověď, která obsahuje příkaz „OK“, zda se jedná o ZM nebo ovladače, název ZM nebo ovladače, „deviceType a „comType“ (pokud se jedná o ovladač).

Pro ověření, zda ZM nebo ovladač je správně připojený, a tedy je přístupné jeho TCPI, lze využít příkaz „ECHO“ v příkazové řádce serveru. Např. „ECHO ‘extLED01‘. Odpovědí ZM nebo ovladače je pak řetězec „Hi, this is ExtLED01“.

Příklad:

Server požadavek

```
{
  "command":"ECHO"
}
```

Odpověď serveru

```
{
  "command":"ECHO",
  "message":"Hi, this is ExtLED01"
}
```

Pokud je spojení se serverem navázáno, knihovna požádá o přístupové údaje ke sdílené databázi. Tyto údaje lze získat zavoláním funkcí „getDbsURL“, „getDbsUser“ a „getDbsPass“. Dále veškerá nastavení ZM nebo ovladače (název, ip adresa, atd.) jsou k dispozici v konfigurační třídě „ConfigPropertiesPlugin“. Knihovna samozřejmě disponuje funkcemi, díky kterým lze komunikovat se serverem a s klientskou částí ZM nebo ovladače. Veškerá komunikace probíhá asynchronně. Pro odesílání dat jsou připraveny tři následující funkce.

sendData(JSONObject jsonObject)

Tato funkce slouží pro komunikaci se serverem. Argumentem je JSONObject, který musí obsahovat klíč „command“, dále pak může obsahovat další potřebné informace (příklad viz obrázek 2).

sendData(JSONObject jsonObject, TcpApiPluginClientSideListener listener, String comForClient)

Tato funkce opět slouží k odesílání dat serveru s tím rozdílem, že odpověď serveru je přeposlána klientské části ZM. Prvním argumentem je JSONObject obsahující klíč „command“ a další potřebné informace. Dále musí být předána třída „TcpApiPluginClientSideListener“ (jak tuto třídu získat a k čemu slouží bude vysvětleno později). Posledním argumentem je příkaz, který se danému ZM odesílá.

Pokud klientská část ZM odešle data serverové části ZM, je zavolána předem určená funkce, které se předá JSONObject a třída zajišťující komunikaci s klientskou částí. Funkci, které jsou předány tyto argumenty lze nastavit pomocí příkazu „setReceiveDataPluginClass“.

setReceiveDataPluginClass (Object receiveDataServerClass, String receiveDataServerFunction)

Funkci se v prvním argumentu předá třída, ve které se funkce nachází. Druhým argumentem se definuje název funkce, které se předají data (JSONObject) a třída „TcpApiPluginClientSideListener“. Obdobně se pomocí příkazu „setReceiveDataServerClass“ definuje funkce pro předávání dat od serveru. Před samotnou inicializací je povinné definovat funkci, která se zavolá po připojení k serveru nebo pokud server odpojí daný ZM. Programátor by měl vždy pamatovat na to, že se musí počkat na

připojení ZM k serveru. Pokud je připojení úspěšné, mohou se provádět další operace, spouštět služby nebo volat potřebné funkce.

Příklad:

```
public class Controller {

    public void receiveDataServer(JSONObject data){ . . . }

    public void receiveDataPlugin(JSONObject data,
                          TcpApiPluginClientSideListener listener){
        . . .
    }

    public void connected(){ . . . }

    public void shutDown(){ . . .}

}


public class Main{

    static IoTHS ioTHS;
    static Controller controller;

    public static void main(String[] args) {

       ioTHS = new IoTHS();

       controller = new Controller();

       ioTHS.setConnectedClass(controller, "connected");
       ioTHS.setShutDownClass(controller, "shutDown");
       ioTHS.setReceiveDataServerClass(controller, "receiveDataServer");
       ioTHS.setReceiveDataPluginClass(controller, "receiveDataPlugin");
       ioTHS.init(args);

    }

}
```

Obrázek 8 - příklad zdrojového kódu serverové části

Na výše uvedeném příkladu je ve funkci „receiveDataPlugin“ k dispozici listener, který je třeba předat funkci pro posílání dat klientské části ZM („sendData“). Jak již bylo řečeno, funkce slouží pro odesílání dat serveru. Pokud je předána třída „TcpApiPluginClientSideListener“, odpověď od serveru se přepošle klientské části ZM. Problém nastává v případě, kdy více klientských částí odešle požadavek a čeká na odpověď. Komunikace se serverem je asynchronní. Je tedy potřeba ošetřit odpovědi serveru a přeposlat data určeným klientským částem. Tento problém řeší funkce knihovny.

3.4. Server – Event handler

Novou důležitou funkcí, respektive službou, na straně serveru je zachytávání událostí. Tato služba značně ulehčuje implementaci IFTTT služby. Službu zachytávání událostí mohou využívat ZM i ovladače. ZM tuto službu většinou využívají k zobrazování dat v reálném čase. Tedy pokud ZM bude zobrazovat např. to, zda je rozsvícené světlo, nemusí kontaktovat ovladač ohledně zjištění změny daného zařízení v určitém časovém intervalu. Pomocí služby zachytávání událostí stačí dané zařízení zaregistrovat do této služby (knihovna zajišťuje tuto funkci pomocí příkazu „addEvent“). Pokud ZM bude požadovat zaregistrování zařízení, musí odeslat serveru id daného zařízení, jaký ovladač dané zařízení obsluhuje a za jaké události má server odeslat změnu stavu zařízení. V případě použití knihoven lze zařízení registrovat pomocí jednoduchého příkazu, např.

IoTHS.addEvent(Events.EVENT_DEVICE_UPDATE, "LightsDriver", "id123")

Server poskytuje celkem tři předdefinované druhy událostí.

Tabulka 3 - předdefinované události

UDÁLOST	POPIS
EVENT_DEVICE_UPDATE	Aktualizace zařízení.
EVENT_DEVICE_ADD	Přidání zařízení.
EVENT_DEVICE_REMOVE	Odebrání zařízení.

Server také umožňuje registraci událostí, které nejsou předdefinované. Pokud tedy ovladač i ZM požadují nestandardní pojmenování události, je tato možnost k dispozici.

Druhým argumentem této funkce je název ovladače daného zařízení. V případě, že nastane nečekané ukončení ZM (server ztratí spojení), jsou všechny eventy pro daný ZM odstraněny.

Ovladače za normální situace funkci „addEvent“ vůbec nevyužívají. Pro ovladače je ovšem relevantní funkce „callEvent“. Tuto funkci ovladače volají, pokud nastane nějaká změna (např. zhasnutí světla, zaznamenání pohybu, …).

Příklad použití:

IoTHS.callEvent(Events.EVENT_DEVICE_UPDATE, "id123", JsonDeviceInfo)

Při vytváření ovladače musí programátor pamatovat na to, že funkce „callEvent“ musí být volána pokaždé, pokud dojde k jakékoliv změně daného zařízení. Pokud by tomu tak nebylo, mohlo by dojít k nečekanému chování automatických funkcí. Funkce totiž zajišťuje nejen odesílání událostí ZM, ale také je provedena kontrola, zda existuje pro dané zařízení automatická funkce (tedy nastavení IFTTT). Pokud tomu tak je, je událost předána službě IFTTT, která provede požadované funkce.

3.5. Server – Ovladače

Ovladače jsou velmi důležitou částí celého systému, a tak se musí dodržovat jistá pravidla pro jejich vývoj. Vzhledem k tomu, že ovladač se serverem komunikuje synchronně, je třeba dbát na časovou náročnost různých operací. Pokud tedy server požádá ovladač o data, ovladač by měl v co nejkratší době odpovědět, jelikož na odpověď čekají další služby serveru a špatně navržený ovladač může mít negativní vliv na celkový chod serveru. Pokud se navrhuje ovladač, **musí** splňovat níže uvedená důležitá kritéria.

- Ovladač musí serveru odpovědět maximálně do 2 sekund. Doporučená doba odpovědi je maximálně 300 ms.
- Pokud je zaznamenána jakákoliv změna stavu zařízení, ovladač musí tuto informaci předat správci událostí (zachytávání událostí) pomocí příkazu **„callEvent“**.
- Příkaz **„sendData“** ovladač může využít pouze při odpovídání serveru, jinak tento příkaz nesmí být používán.
- Ovladač musí odpovědět na každý příkaz serveru. Pokud nejsou žádná data k odeslání, server očekává odpověď, která obsahuje klíč **„ans“** a např. hodnotu **„ok“**.
- Pokud driver ukládá nastavení zařízení (JSON formát), která mu přeposlal server, musí se vždy odstranit klíče **„command“**, **„event“** a **„comDriver“**.
- Veškerá komunikace musí probíhat pomocí JSON formátu.
- Každý příkaz musí začínat slovem „DRIVER“. Např. DRIVER GET DEVICES, DRIVER SET DEVICE.

Aby byla zajištěna kompatibilita mezi všemi ovladači a serverem, je třeba dodržet následující standardní pojmenování. V případě, kdy se zvolí jiný název typu zařízení, není možné používat v automatických funkcích zařízení spravována tímto ovladačem.

Tabulka 4 - standardní pojmenování "driverDeviceType"

Standardní pojmenování	Popis
Lights	Zařízení, která se starají o osvětlení domácnosti.
LightsOut	Zařízení, která se starají o osvětlení venkovního prostoru.
LEDstrips	LED pásy
LEDstripsOut	Venkovní LED pásy
Motion	Používá se u ovladačů, které obsluhují pohybové senzory.
MotionOut	Venkovní pohybové senzory.
MotionSecure	Podobné jako Motion s tím rozdílem, že tyto pohybové senzory jsou určené pro zabezpečení domácnosti.
Controllers	Různé ovládací prvky jako např. tlačítka, přepínače, ovladače atd.
Others	Ostatní zařízení

Ovladač dále musí mít implementované 4 základní příkazy

Tabulka 5 - základní příkazy pro ovladač

PŘÍKAZ	POPIS
DRIVER GET DEVICES	Výstupem musí být pole všech zařízení, která jsou ovladače spravována.
DRIVER SET DEVICE STATUS	Příkaz pro změnu nastavení daného zařízení.
DRIVER GET DEVICE	Žádost o data zařízení podle přezdívky.
DRIVER GET DEVICE BY ID	Žádost o data zařízení podle id.

Příklad „DRIVER GET DEVICES“

```
{
  "devices":[
    {
      "id":"id123",
      "brightness":58
    },
    {
      "id":"id456",
      "brightness":120
    }
  ]
}
```

Příklad „DRIVER GET DEVICES“

```
{
    "id":"id123",
    "brightness":58
}
```

Příklad „DRIVER SET DEVICE STATUS“

```
{
    "command":"DRIVER SET
                DEVICE STATUS",
    "id":"id123",
    "brightness":255
}
```

V konfiguračních souborech (dev, prod) musí mít ovladač specifikován svůj unikátní název. Dále nastavení „driverDeviceCom“ a důležité nastavení „driverDeviceType“. U tohoto nastavení je doporučeno používat standardní pojmenování typu zařízení (viz tabulka 4). Dále je doporučeno, aby ovladač neprováděl žádnou úlohu před připojením k TAPI serveru. Podrobné nastavení konfiguračních souborů a příklad použití knihovny je ukázán v kapitole 2.2.3.

Instalace ovladače probíhá zkopírováním do složky „plugins“. Složka, kde se pak nachází ovladač musí být pojmenována stejně jako ovladač. Stejně tak musí být pojmenován spustitelný soubor (.jar, .bat) a textový soubor, kde se specifikuje, jak se má spustit .jar soubor. Vzhledem k tomu, že je server připravený k nasazení na jakémkoliv systému, který podporuje JVM, je někdy nutné specifikovat příkaz na spuštění .jar souboru. Také může být v některých případech potřeba spustit ovladač s externími knihovnami. Tento příkaz se specifikuje v textovém souboru, který má stejný název jako ovladač.

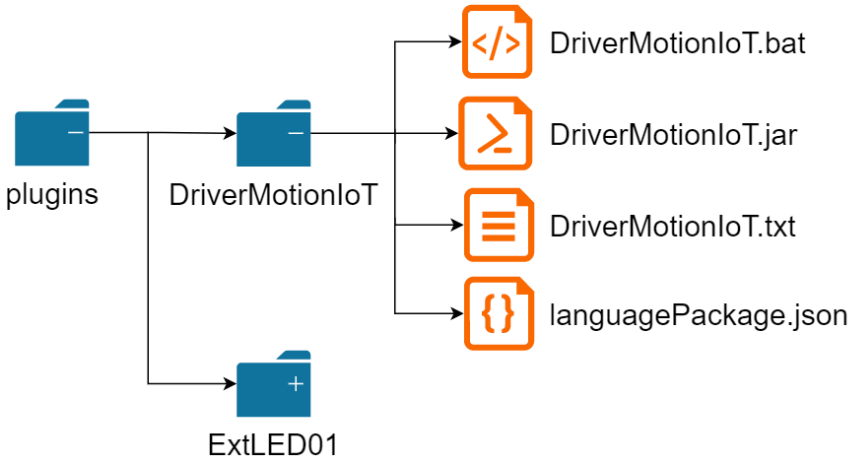
Příklad:

```
java -jar plugins?[$name]?[$name].jar [$arg1] [$arg2] [$arg3] [$arg4] [$arg5]
```

argumenty pro spuštění arg1 až arg5 jsou povinné a musí být vždy vstupními argumenty při spuštění ovladače. Tyto argumenty nahrazuje server např. IP adresou a portem. Tyto argumenty jsou pak předány funkcím knihovny (viz příklad v kapitole 2.2.3.).

Veškeré označení [$name] je serverem nahrazeno za název ovladače. Znak „?“ je nahrazen za znak „/“ nebo „\“. Je doporučeno vždy používat znak „?“ pro určení systémové cesty např. ke knihovnám. Vygenerovaný příkaz na spuštění se pak uloží do .bat souboru, který je následně spuštěn. Pokud bude server spuštěný v Linuxovém prostředí, je třeba nastavit tento soubor jako spustitelný příkazem „chmod +x“.

Souborová struktura pro ovladač „DriverMotionIoT“ a ZM „ExtLED01“ vypadá následovně.



Obrázek 9 - souborová struktura uložení ovladače/ZM

Dále je nutné na straně serveru ovladač přidat pomocí příkazu „ADD PLUGIN“. Přidání je možné buď pomocí příkazové řádky serveru nebo pomocí JSON příkazu.

Formát přidání pomocí příkazové řádky:

```
ADD PLUGIN -N '[název_ovladače] ' -A '[1|0]' -D '[1|0]'
```

-N – název ovladače (Name)
-A – zda je ovladač povolen (Allow)
-D – zda se jedná o ovladač (Driver)

Příklad přidání pomocí příkazové řádky:

```
ADD PLUGIN -N 'DriverMotionIoT ' -A '1' -D '1'
```

Příklad přidání pomocí JSON:

```
{
  "command":"ADD PLUGIN",
  "name":"DriverMotionIoT",
  "allow":true,
  "driver":true
}
```

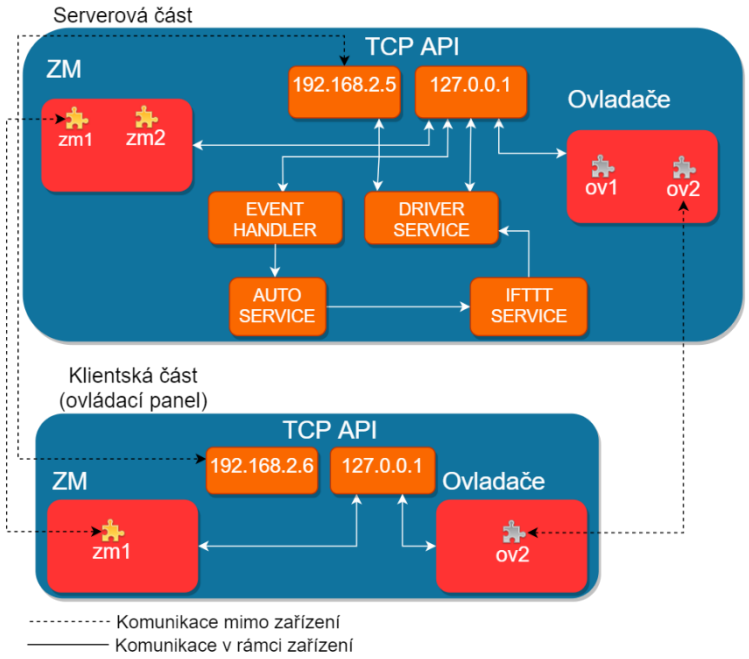
3.6. Server – Zásuvné moduly (ZM)

U ZM není třeba dodržovat tak striktní pravidla jako u ovladačů. Komunikace se serverem probíhá asynchronně. Pokud se tedy ZM zpozdí s odpovědí nebo přestane odpovídat, tak to na celkový výkon serveru nemá vliv.

Opět platí, že ZM musí mít jedinečný název v rámci systému a nastavení „driverMode“ nastaveno na „false“. Instalace ZM probíhá stejně jak je tomu u ovladačů. Stejně tak se musí dodržet souborová struktura (viz obrázek 4).

3.7. IFTTT služba

Nově implementované služby (services) poskytují velmi silný nástroj, díky kterému lze snadno implementovat službu IFTTT. Obrázek 10 poskytuje představu o tom, jak mezi sebou služby komunikují, včetně služby IFTTT.



Obrázek 10 - princip komunikace

Tato kapitola se zabývá především „IFTTT service“, „AUTO service“ a okrajově „Event hander“. Jak je z obrázku patrné, IFTTT službu může volat pouze služba „AUTO service“, které předává informace služba předávání událostí.

Pro základní představu funkčnosti předpokládejme případ, kdy si uživatel nastavil automatickou funkci, která rozsvítí světlo v místnosti, pokud pohybový senzor zaznamená pohyb. Aby bylo možné takovou funkci realizovat, musí být zavedeny ovladače, které zajišťují komunikaci mezi serverem a samotným zařízením (pohybový senzor a světlo). Taková funkce je v tomto systému označována jako „Event“, respektive událost. Tedy čeká se, až ovladač pošle službě zachytávání událostí data, která se pro dané zařízení změnila. V prvním kroku služba zachytávání událostí odešle data o zařízení službě „AUTO service“. Ta prověří, zda pro dané zařízení existuje automatická funkce, která se má spustit. Pokud tedy pohybový senzor zachytí v místnosti pohyb, pošle tuto událost ovladači a ovladač událost předá službě zachytávání událostí, která předá data službě „AUTO service“. Pro pohybový senzor existuje automatická funkce, a tak se předávají data dále IFTTT službě. Tato služba ověří, zda senzor zaznamenal pohyb (událost mohla být vyvolána na základě jiné událost, než je detekce pohybu). Pokud je podmínka splněna, přes službu „DRIVER service“ odešle příkaz ovladači, který zapne osvětlení. Služba zachytávání událostí může pak událost distribuovat dál a to např. serverové části ZM.

V tomto úvodním jednoduchém příkladu bylo ukázáno, jaký je základní princip komunikace mezi službami, ovladači a zařízeními. Služby nabízí mnoho dalších funkcí, které jsou nedílnou součástí automatizace.

Automatické funkce lze přiřadit do skupin. Skupiny slouží pro přehledné řazení automatických funkcí. Pravidla pro automatické funkce a skupiny jsou následující.

- Každá skupina může obsahovat neomezený počet automatických funkcí
- Skupinu lze volat pouze automatickou funkcí
- Automatická funkce musí být přiřazena právě do jedné skupiny
- Automatická funkce může volat právě jednu skupinu nebo funkci
- Rekurzivní volání funkcí není povoleno
- Funkce může být volána jinou funkcí, automaticky volána po X sekundách nebo volána na základě události

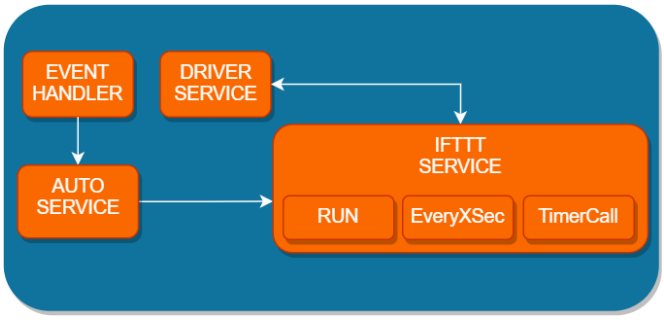
Skupinu lze nastavit jako profil. Pokud je skupina nastavena jako profil, lze ji spouštět ručně pomocí ovládacího panelu. Lze jí také přiřadit do místnosti za účelem rychlého ovládaní zařízení, která jsou do skupiny přiřazené. Skupinu lze vytvořit pomocí příkazu „ADD GROUP“ nebo upravovat příkazem „EDIT GROUP“. Dále se musí specifikovat název skupiny, ikona a zda se jedná o profil. Pokud se posílají data pro editaci skupiny, musí se odeslat ID skupiny, která je editována a její nové nastavení.

Příklad:

```
{
  "command":"EDIT GROUP",
  "groupID":"10-abc",
  "name":"Skupina1",
  "ico":"light.png",
  "isProfile":true,
}
```

Systém při svém spuštění kontroluje, zda existuje pět základních profilů. Jedná se o profily Doma, Pryč, Den, Noc a Alarm. Tyto profily nelze upravovat ani odstraňovat. Profily Noc a Den jsou volány na základě nastavení času, který určuje západ a východ slunce. Toto nastavení lze nastavit staticky nebo nastavit automatické nastavení času z internetu. Profil Alarm je zavolán, pokud je spuštěn globální alarm. Profily Doma a Pryč se mohou volat ručně nebo opět pomocí nadefinovaného času. O volání těchto profilů se stará služba „AUTO service“ (tedy kontroluje čas např. východu a západu slunce). Výjimku má profil globální alarm. O globální alarm se stará nezávislá služba. Pokud je zapnutý globální alarm, služba kontaktuje službu „AUTO service“ a ta poté zavolá příslušný profil.

IFTTT služba obsahuje tři pod-služby. Jak již bylo výše řečeno, automatická funkce se může volat na základě události, pokud tomu tak je, služba „AUTO service“ kontaktuje pod-službu „RUN“ (dále jako služba spuštění). Této službě je předáno nastavení automatické funkce, popřípadě data o zařízení, které událost vyvolalo (pro ověření podmínky). Předání automatické funkce je vykonáno pouze v případě, kdy je funkce nastavená jako aktivní, jinak je ignorována. Právě služba spuštění je zodpovědná za zjištění, zda je nastavená podmínka splněna nebo nesplněna. Pokud ano, vykoná se nastavená činnost (např. zapnutí osvětlení, zavolání jiné funkce, odeslání emailu atd.).

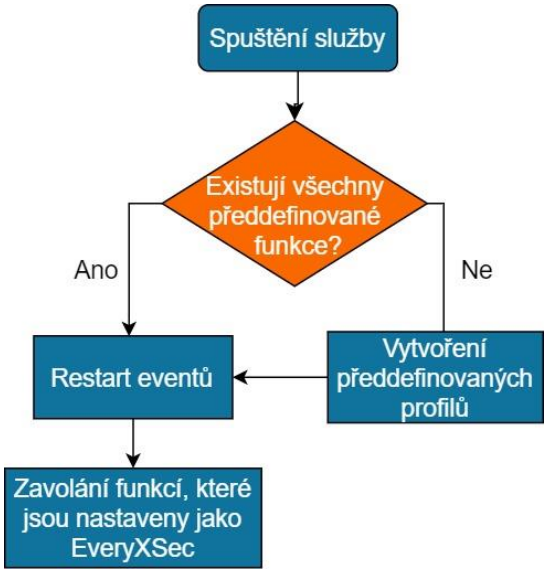


Obrázek 11 - architektura IFTTT

Automatická funkce může být také volána v časovém intervalu. K tomuto účelu slouží pod-služba „EveryXSec“ (dále jako služba časovač). Taková služba se může použít např. k blikání světla nebo periodickému kontrolování stavu zařízení.

Dále je k dispozici speciální pod-služba „TimerCall“ (dále jako služba volání). Díky této službě je možné zavolat (Call) jinou funkci uvnitř funkce nebo skupinu po stanoveném čase (Timer). U automatické funkce je k dispozici okamžité zavolání jiné funkce nebo skupiny. Ovšem to je možné pouze v případě, kdy je nastavená podmínka (např. detekce pohybu) a nelze volání odložit.

Příklad reálného použití: Předpokládejme situaci, kdy je nastavená automatická funkce, která zapne osvětlení v místnosti, pokud je detekován pohyb. Nyní je ale požadavek takový, že po 30 sekundách se má osvětlení vypnout. Tento požadavek lze splnit tak, že se nastaví speciální funkce Timer&Call, ve které se specifikuje zavolání funkce pro vypnutí světla po stanoveném intervalu.



Obrázek 12 - vývojový diagram spuštění služby "AUTO service"

Na obrázku 12 je znázorněn vývojový diagram služby „AUTO service“. V prvním kroku se ověří, zda existují všechny předdefinované skupiny, respektive profily.

Pokud je automatická služba nastavena tak, že je volána na základě události, je nastavena v databázi jako „event“. Pokud je zavolána, přepíše se tento atribut na „InProgress“. Tímto služba „AUTO service“ zjistí, zda je automatická funkce již spuštěna a nemusí jí tedy znovu spouštět.

Pokud dojde k nečekanému ukončení systému, musí se všechny automatické funkce, u kterých zůstal atribut nastavený na hodnotu „InProgress“, vrátit do stavu „event“.

V posledním kroku jsou spuštěny funkce, které mají nastavená periodická spuštění v stanoveném čase (EveryXSec).

Automatickou funkci lze vytvořit pomocí formátu JSON, ten pak odeslat přes API serveru příkazem „ADD AUTO FUNCTION“. Server provede validaci a pokud je automatická funkce správně nastavena a splňuje všechny podmínky, je uložena a dále zpracována.

Automatická funkce se skládá celkem z 9 částí. Některé části je povinné vyplnit, některé části se ve funkci nemusí vůbec nacházet, respektive nemusí být specifikované. Další části jsou podmíněné jinou části. Pokud např. nebude specifikovaná podmínka, nelze nastavit volání funkce nebo odeslání emailu.

Sestavení JSON klíčů a hodnot je následující (praktický příklad lze nalézt v příloze č. 3).

1. Část – skupina a název funkce

Volitelné: Ne
Podmíněné použití: Ne

Pro specifikování názvu funkce je určen klíč „functionName“. Pro specifikaci skupiny (id skupiny) je určen klíč „group“. Oba klíče mají datový typ String.

2. Část – podmínka

Volitelné: Ano
Podmíněné použití: Ne

Tato část slouží k upřesnění, za jaké podmínky se má automatická funkce spustit (spuštěním se myslí např. nastavení zařízení, odeslání e-mailu, zavolání jiné funkce atd.). Pokud je specifikována podmínka, klíč „ifOrSet“ musí být nastaven na hodnotu „if“. Pokud podmínka nastavena není, klíč musí mít hodnotu „set“. V následující tabulce je ukázáno, jaké klíče jsou požadovány a jakých hodnot mohou nabývat.

Tabulka 6 – IFTTT JSON podmínka

Název	JSON klíč	Datový typ/hodnoty	Popis
Ovladač	ifDriver	string	Název ovladače, který spravuje dané zařízení.
Zařízení	ifDevice	string	ID, pod kterým je zařízení uloženo v daném ovladači.
Klíč	ifKey	string	Klíč, který má být u daného zařízení prověřován.
Podmínka	ifCondition	=, !=, <, >, <=, >=	Podmínka.
Hodnota	ifValue	string, long, bool	Hodnota, které se např. musí rovnat klíč.

3. Část – nastavení zařízení

Volitelné: Ano
Podmíněné použití: Ne

Pokud se má po spuštění funkce nebo splnění podmínky nastavit zařízení, klíč „set“ musí být nastaven na hodnotu „true“ (datový typ bool). V opačném případě na „false“.

Tabulka 7 - IFTTT JSON nastavení zařízení

Název	JSON klíč	Datový typ/hodnoty	Popis
Ovladač	setDriver	string	Název ovladače, který spravuje dané zařízení.
Zařízení	setDevice	string	ID, pod kterým je zařízení uloženo v daném ovladači.
Klíč	setKey	string	Klíč, který má být u daného zařízení změněn.
Změna	setChoice	set, invert, add, deduct	Co se má stát s nastaveným klíčem.
Hodnota	setValue	String, long, bool	Hodnota, která se má nastavit.

Stav zařízení lze měnit pomocí klíče „setKey“ (např. změna klíče „state“ na „true“). Je také možné invertovat danou hodnotu. Tato funkce funguje, pokud je klíč datového typu bool nebo long (vynásobení čísla -1). Dále je k dispozici přičtení nebo odečtení hodnoty. Tato možnost je povolena pouze pro číselné hodnoty.

4. Část – akce

Volitelné: Ano
Podmíněné použití: Ano – musí být specifikována podmínka

Po splnění podmínky je možnost spustit globální poplach, aktivovat automatickou funkci nebo deaktivovat funkci (lze aktivovat nebo deaktivovat celou skupinu).

Aktivace nebo deaktivace funkce se může využít opět v příkladu rozsvícení a zhasnutí světla. Předpokládejme případ, kdy se světlo v místnosti rozsvítí na základě pohybu. Uživatel chce světlo manuálně zhasnou a z místnosti odejít. Je nežádoucí, aby se světlo opět na základě jeho pohybu po zhasnutí ihned rozsvítilo. V tomto případě se hodí využít funkci deaktivace automatické funkce pro rozsvícení světla na základě pohybu a po stanovené době opět její aktivace. Uživatel tedy zhasne světlo a zároveň se deaktivuje funkce na rozsvícení např. po dobu 10 sekund. Po deseti sekundách se funkce opět aktivuje.

Pokud je akce specifikována, klíč „action“ musí být nastaven na hodnotu „true“. V opačném případě na „false“.

Tabulka 8 - IFTTT JSON akce

Název	JSON klíč	Datový typ/hodnoty	Popis
Akce	doAction	„global alarm ON“, „activate function“, „deactivate function“	Nastavení akce.
Skupina	actionFunctionCall	string	ID volané skupiny nebo automatické funkce.
Funkce	actionCallGroup	bool	Pokud „true“ jsou volány všechny automatické funkce ve skupině. Pokud „false“, ID patří jedné funkci.
Časovač	actionTimeAD	long	Na kolik sekund se má funkce aktivovat nebo deaktivovat.

Pro stálou aktivaci/deaktivaci funkce musí být časovač nastaven na nulu.

5. Část – odeslání e-mailu

Volitelné: Ano
Podmíněné použití: Ano – musí být specifikována podmínka

Pro odeslání e-mailů musí být klíč „email“ nastaven na hodnotu „true“ v opačném případě na „false“. Systém podporuje pokročilé odesílání e-mailů, které je popsáno v kapitole 2.2.10.

Tabulka 9 - IFTTT JSON e-mail

Název	JSON klíč	Datový typ/hodnoty	Popis
Předmět	subjectEmail	String	Předmět e-mailu.
Text	textEmail	String	Text e-mailu.

6. Část – zavolání funkce

Volitelné: Ano
Podmíněné použití: Ano – musí být specifikována podmínka

Možnost zavolání jedné specifické automatické funkce nebo celé skupiny. Pro zavolání funkce nebo skupiny musí být klíč „callFunction“ nastaven na hodnotu „true“ v opačném případě na „false“.

Zavolání funkce nebo skupiny může být užitečné v případě, kdy se má např. rozsvítit více světel v místnosti na základě pohybu. Pro takovou situaci se vytvoří skupina s název např. „Rozsvítit světla kuchyň“ a tato skupina se na základě pohybu v kuchyni zavolá. V této skupině se pak může nacházet množina funkcí, které postupně zapnout osvětlení v dané místnosti.

Tabulka 10 - IFTTT JSON zavolání funkce

Název	JSON klíč	Datový typ/hodnoty	Popis
Akce	doAction	„global alarm ON“, „activate function“, „deactivate function“	Nastavení akce.
Skupina	actionFunctionCall	string	ID volané skupiny nebo automatické funkce.
Funkce	actionCallGroup	bool	Pokud „true“ jsou volány všechny automatické funkce ve skupině. Pokud „false“, ID patří jedné funkci.
Časovač	actionTimeAD	long	Na kolik sekund se má funkce aktivovat nebo deaktivovat.

7. Část – nastavení času

Volitelné: Ne
Podmíněné použití: Ne

Určení, od kdy do kdy je možné funkci volat. Např. rozsvícení světel pouze od 18:00 večer do 7:30 ráno

Tabulka 11 - IFTTT JSON nastavení času

Název	JSON klíč	Datový typ/hodnoty	Popis
Hodina od	timeFromH	string	Od (hodiny)
Minuta od	timeFromM	string	Od (minuty)
Hodina do	timeToH	string	Do (hodiny)
Minuta do	timeToM	string	Do (minuty)

8. Část – speciální funkce

Volitelné: Ne
Podmíněné použití: Ne

Mezi speciální funkce patří zpoždění provedení automatické funkce a odložení volání (Timer&Call) jiné automatické funkce nebo skupiny funkcí. Odložení volání jiné funkce může být užitečné v případě, kdy se např. má rozsvítit světlo po určitou dobu. Tedy první automatická funkce rozsvítí světlo v místnosti (např. při detekci pohybu) a funkce Timer&Call bude nastavena na zavolání jiné funkce (nebo skupiny), která dané světlo zhasne např. po 30 sekundách.

Pokud je specifikována speciální funkce, je k dispozici funkce „restart“. Restart slouží pro restartování časovače volání. Pokud se bude vycházet z výše uvedeného případu a světlo se rozsvítí na základě pohybu a zároveň bude povolený restart, tak se časovač restartuje na 30 sekund (dokud tedy bude pohyb v místnosti, světlo se nezhasne, jelikož se časovač vždy restartuje). Pokud nebude nastavený, světlo se zhasne po 30 sekundách.

Odložení nebo zpoždění je možné nastavit v sekundách, milisekundách a minutách. Minimální čas odložení volání automatické funkce je 100 ms.

Tabulka 12 - IFTTT JSON speciální funkce

Název	JSON klíč	Datový typ/hodnoty	Popis
Speciální funkce	specialFunction	Delay, TimerCall	Typ speciální funkce.
Zpoždění/ zavolání	specialFunctionTime	long	Zpoždění provedení funkce/zpoždění zavolání další funkce
Jednotky	specialFunctionUnits	ms, s, min.	
Restart	restart	bool	Restart časovače
Skupina	TimerCallGroup	bool	ID volané skupiny nebo automatické funkce.
Funkce	specialFunctionCall	string	Pokud „true“ jsou volány všechny automatické funkce ve skupině. Pokud „false“, ID patří jedné funkci.

9. Část – volba volání automatické funkce

Volitelné: Ne
Podmíněné použití: Částečné

Poslední částí se musí zvolit, zda automatická funkce bude volána na základě nějaké události, periodicky nebo jestli funkce bude volána jinou funkcí. Volání funkce na základě události lze zvolit pouze tehdy, pokud je nastavena podmínka (2. část). Ostatní volby je možné použít bez ohledu na předešlé nastavení.

Tabulka 13 - IFTTT JSON volba volání

Název	JSON klíč	Datový typ/hodnoty	Popis
Zavolání	callEvent	Event, Called, EveryXsec	Při jaké příležitosti se má funkce zavolat.
Perioda	callEventSec	long	Nastavení časovače v sekundách (pouze pro EveryXsec)

Perioda volání automatické funkce nesmí být menší jak 1 sekunda.

3.8. IFTTT služba – Jazykový balíček

Jazykové balíčky slouží pro překlad klíčů z JSON formátu. Příkladem může být standardizovaný klíč pro úpravu jasu. Tento klíč je pojmenován jako „brightness“. Pokud je např. ovládací panel nastaven v českém jazyce, je pro uživatele přívětivější, aby i nastavení různých hodnot zařízení bylo přeloženo do nastaveného jazyku. Klientská část tedy musí mít k dispozici jazykový balíček, ve kterém je jasně specifikované, jak daný klíč přeložit. Pokud má ovladač specifikovaný svůj vlastní balíček, musí se nacházet ve stejné složce, kde je nainstalovaný ovladač (viz obrázek 14)

V jazykových balíčcích lze také specifikovat, jaké hodnoty lze zadat, jaká je maximální nebo minimální hodnota, zda lze klíč použít pouze v podmínce nebo i v nastavení. Vytvoření jazykového balíčku lze opět rozdělit do určitých částí.

1. Část – název ovladače

Název ovladače se určí pomocí klíče „name_[jazyk]”. Např. „name_cz“ nebo „name_de“. Povinný klíč „name_def“ (name default) určuje výchozí název ovladače. Tedy pokud je zvolen jazyk, pro který daný ovladač nemá překlad, zvolí se výchozí název, který by měl mít vždy anglický překlad.

2. Část – název klíče

Název klíče zařízení (např. brightness) lze určit pomocí klíče „title_[jazyk]”. Např. „title_cz“ nebo „title_de“. Vždy musí existovat výchozí název klíče. Ten lze určit pomocí klíče „title_def“ (title default). Výchozí název klíče by měl mít vždy anglický překlad.

3. Část – vstup

V této části lze určit, zda hodnotu pro daný klíč může uživatel nastavit pomocí „textBox“ nebo „comboBox“. V případě „textBox“ se uživateli zobrazí komponenta, do které lze ručně zadat hodnotu. Pokud se bude jednat o „comboBox“, bude uživateli zobrazen seznam, ze kterého může hodnotu vybrat.

Vstup se určuje klíčem „input“ a nabývá hodnot „textBox“ nebo „comboBox“.

V případě, kdy je zvolen „comboBox“ je nutné určit jeho položky. Pro tento účel slouží klíč „items_[jazyk]” a povinný klíč „items_def“. Tyto klíče pak obsahují další JSON objekt, ve kterém se specifikuje název a hodnota. Předpokládejme případ, kdy se nastavuje senzor pohybu, který má klíč „motion“, jenž nabývá hodnot „true“ (pohyb) a „false“ (žádný pohyb). Název klíče je překlad, tedy např. „Detekován pohyb“ a hodnota tohoto klíče pak hodnota, které nabývá klíč „motion“. V části 6 je ukázán příklad, kde jsou specifikované dvě položky pro klíč „motion“.

4. Část – podmínka/nastavení

V případě např. pohybového senzoru nemá smysl, aby detekce pohybu šla nastavit. Detekce pohybu by tedy měla být dostupná pouze jako podmínka. K tomuto účelu slouží klíč „if“ a „set“, které nabývají hodnot „true“ nebo „false“. V případě pohybového senzoru by bylo nastavení takové, že klíč „if“ by byl nastaven na „true“ a „set“ na „false“.

5. Část – určení datového typu

Pokud je „input“ nastaven na „textBox“, musí se určit datový typ zadávané hodnoty pomocí klíče „values“. Tento klíč může nabývat hodnot „N“ (přirozená čísla), „R“ (reálná čísla) a „S“ (řetězec).

6. Část – maximální/minimální hodnota

Určení maximální a minimální hodnoty lze v případě, kdy je datový typ nastaven na „N“ nebo na „R“. Maximální hodnotu lze nastavit pomocí klíče „maxValue“, minimální hodnotu pomocí klíče „minValue“.

```
{
  "motion":{
    "title_cz":"Pohyb",
    "title_def":"Motion",
    "input":"comboBox",
    "if": true,
    "set": false,
    "items_cz":{
      "Detekován pohyb":true,
      "Nedetekován pohyb":false
    },
    "items_def":{
      "Motion":true,
      "No motion":false
    }
}
```

```
{
  "brightness":{
    "title_cz":"Jas",
    "title_def":"Brightness",
    "input":"textBox",
    "if": true,
    "set": true,
    "values":"N",
    "maxValue": 255,
    "minValue": 0
  }
}
```

Veškeré standardizované klíče má server již předpřipravené v JSON souboru „defaultLanguagePackage.json“. Pokud tedy ovladač nevyužívá nestandardní klíče, není třeba vytvářet zvláštní jazykový balíček. V případě, kdy je třeba přepsat např. maximální hodnotu jasu, je nutné vytvořit JSON soubor a celý klíč „brightness“ definovat znovu.

Pokud klient zažádá o jazykový balíček daného ovladače, je poté provedeno sjednocení základního balíčku a vytvořeného balíčku ovladačem. Klíče, které jsou specifikované ovladačem, se nepřepisují.

3.9. IFTTT služba – Proměnné

Pro pokročilé uživatele je k dispozici vytváření vlastních proměnných. Proměnné je výhodné využít např. v případě řetězení automatických funkcí. Předpokládejme případ, kdy se při detekci pohybu v místnosti rozsvítí více jak jedno světlo. V tomto případě se vytvoří automatická funkce, která má jako podmínku detekci pohybu. Pokud je podmínka splněna, nastaví se proměnná na hodnotu „true“. Server vyvolá událost a proměnnou předá službě události, která proměnnou předá příslušným automatickým funkcím. V těchto funkcích je podmínka nastavena tak, že pokud je daná proměnná nastavena na „true“, rozsvítí se určená světla. Je třeba pamatovat na to, že proměnnou je třeba opět změnit na hodnotu „false“, pokud není zaznamenán pohyb.

Jeden z důvodů, proč používat proměnné je snížení výpočetní náročnosti vykonávání automatických funkcí. Pokud je vyvolána událost, musí se předávat různým službám JSON data, která mohou mít až přes 30 klíčů. V těchto klíčích se musí najít jeden konkrétní klíč ke zjištění, zda platí podmínka nebo převod z řetězce do JSON objektu atd. V případě, kdy má server přidaných minimum automatických funkcí, nemusí být toto zrychlení nějak zřetelné. Server je ale určený pro nasazení na jednodeskových počítačích a pokud server budeme muset obstarávat např. přes 50 automatických funkcí, je používání proměnných velkou úsporou výpočetního výkonu.

Dalším důvodem může být úspora času pro uživatele při změně zařízení. Pokud např. pohybový senzor používá více automatických funkcí, bylo by nutné při změně senzoru přepsat všechny tyto funkce. Pokud se využije proměnná, je třeba změnit pouze funkci, která přepisuje danou proměnnou např. na základě detekce pohybu. Proměnné mohou být dále využity např. pro diagnostické nebo informativní účely.

Povolené datové typy pro proměnné jsou long, bool, string. K proměnné lze přičíst nebo odečíst číslo, invertovat nebo nastavit na danou hodnotu.

3.10. IFTTT služba – Pokročilé odesílání e-mailů

Pokročilé odesílání e-mailů je další funkce pro pokročilé uživatele, kterou lze využít při nastavování automatizace. Tato funkce nabízí posílání diagnostických informací o zařízeních nebo je možné odeslat hodnotu proměnné. Tuto funkci, která se nastavuje v 5. části automatické funkce, lze uplatnit v předmětové i textové, respektive obsahové části e-mailu.

Pro získání informací od určitého zařízení je nutné specifikovat ovladač, id zařízení a klíč, který má být zobrazen.

Formát:

<název_ovladače:id_zařízení:klíč>

Názvem ovladače se rozumí název ovladače, který je specifikován pomocí **pluginConfig.pluginName.** Tedy stejný název, jakým je pojmenovaná složka, ve které je ovladač nainstalovaný.

Pro zobrazení proměnné je nutné pouze specifikovat název proměnné

Formát:

<value:název_proměnné>

Tabulka 14 - příklad e-mailu

Část e-mailu	Text
Předmět	Denní hlášení
Text	Stav senzoru: <DriverMotionIoT:id123:online> Čas poslední změny: <DriverMotionIoT:id123:lastUpdate> Dnes detekováno pohybů v místnosti: <value:motion_count>

V případě, že je k dispozici jazykový balíček ovladače, jsou všechny klíče přeloženy podle tohoto balíčku (např. klíč „online“ se přeloží jako dostupné nebo nedostupné). V opačném případě jsou přeloženy podle základního balíčku serveru. Nepřeložené klíče budou zobrazeny ve své původní podobě (klíč „online“ by se zobrazil jako „true“ nebo „false“).

3.11. IFTTT služba – API

Díky IFTTT službě se API rozšířilo o mnoho dalších příkazů, které lze využít např. k získání skupin, automatických funkcí, ovladačů atd. Tyto příkazy lze využít pouze v případě, kdy je klient připojený na klientskou TAPI. Ovladače ani ZM tyto příkazy nemohou používat. Ovladače ani ZM nemají přístup k informacím ohledně jiných ovladačů, ZM, skupin a automatických funkcí.

Pro IFTTT službu jsou důležité příkazy, které jsou uvedeny v příloze 2. V případě, kdy klient (např. ovládací panel) požádá o seznam automatických skupin, je serverem vygenerován seznam, který již obsahuje názvy skupin a daných zařízení. Tento seznam ovšem neobsahuje překlady klíčů. Klient tedy musí požádat server o zaslání jazykového balíčku a pak sám přeložit dané klíče do zvoleného jazyku.

Připojení ke klientské TAPI nevyžaduje žádné pomocné knihovny. Pro navázání spojení se serverem je třeba navázat TCP spojení na nastavené IP adrese a portu. K tomuto účelu lze využít např. knihovny java.net.Socket v Javě nebo System.Net.Socket v C#. Po úspěšném připojení k TAPI je odeslán serverem požadavek na odeslání názvu připojeného klienta

pomocí příkazu „GET NAME“. Klient pak musí odpovědět pomocí příkazu „CLIENT NAME“ a v klíči „name“ uvést svůj název (např. „ovládací panel obyvák“ nebo „web klient“).

3.12. Klient (ovládací panel)

Hlavním cílem ovládacího panelu je poskytnout uživateli co nejjednodušší ovládání zařízení, vytváření automatických funkcí, poskytování informací ohledně systému atd. Za tímto účelem bylo vytvořeno grafické uživatelské rozhraní pomocí frameworku JavaFX a Spring Boot. Stejně jako server, je i klienta možné používat na všech operačních systémech, které podporují JVM. Klient dále podporuje přidávání klientských části ZM a ovladačů. K tomuto účelu byla vytvořena knihovna „LibraryPluginClientSide“, která umožňuje připojení k TAPI klienta, serveru a serverové části ZM nebo ovladače.

3.13. Klient – Knihovna pro vytváření ZM a ovladačů

Použití knihovny pro vytváření klientských částí ZM a ovladačů je velmi podobné jako u serverové části. Platí stejné použití konfiguračních souborů jako u serverové části. V následující tabulce jsou konfigurační klíče, které se používají pro nastavení klientské části.

Tabulka 15 - konfigurace

Klíč	Popis
Pluginconfig.driverMode	Určení, zda se jedná o ovladač nebo ZM. V případě, že je knihovna využívána k vývoji ovladače, je tento klíč nastaven na „true“, jinak „false“.
pluginConfig.pluginName	Název ZM/ovladače.
pluginConfig.debugMode	Testovací mód.
pluginConfig.ipAddressClient	IP adresa klienta TAPI (vždy by mělo být nastaveno na 127.0.0.1).
Pluginconfig.portClient	Nastavení portu, na kterém odposlouchává klientské TAPI.
Pluginconfig.portServer	Nastavení portu, na kterém odposlouchává serverová část ZM nebo ovladače. (**Pouze pro debug mode**)

Knihovna poskytuje funkce, díky kterým lze komunikovat s klientem, respektive ovládacím panelem.

sendDataClient(JSONObject jsonObject)

Dále je k dispozici funkce pro odesílání dat serverové části ZM nebo ovladače

sendDataPlugin(JSONObject jsonObject)

Klientská část ZM nebo pluginu se spouští jako samostatný program. V případě, kdy uživatel vybere zobrazení vybraného ZM nebo ovladače, klient odešle příkaz pro zobrazení grafického rozhraní, respektive scény příkazem „SHOW“. Pokud uživatel ukončí zobrazení daného ZM nebo ovladače, klient odešle příkaz „HIDE“ pro schování dané scény. ZM nebo ovladač tedy nejsou ukončeny, ale pouze je schována grafická scéna a program je stále spuštěn na pozadí. Jelikož k vývoji může být využita libovolná technologie pro vytváření grafického rozhraní, je nutné knihovně předat funkce, které obstarávají zobrazení a skrytí scény.

Příklad:

```
public class Controller {

    public void receiveDataPlugin(JSONObject data){ . . . }
    public void connected(){ . . . }
    public void shutDown(){ . . .}
    public void show(String room){ . . . }
    public void hide(){ . . . }
    public void setSize(int height, int width){ . . . }
    public void setPosition(int x, int y){ . . . }

}


public class Main{

    static IoTHS ioTHS;
    static Controller controller;

    public static void main(String[] args) {

       ioTHS = new IoTHS();

       controller = new Controller();

       ioTHS.setConnectedClass(controller, "connected");
       ioTHS.setShutDownClass(controller, "shutDown");
       ioTHS.setReceiveDataPluginClass(controller, "receiveDataPlugin");
       ioTHS.setShowClass(controller, "show");
       ioTHS.setHideClass(controller, "hide");
       ioTHS.setSizeChangedClass(controller, "setSize");
       ioTHS.setPositionClass(controller, "setPosition");
       ioTHS.init(args);

    }

}
```

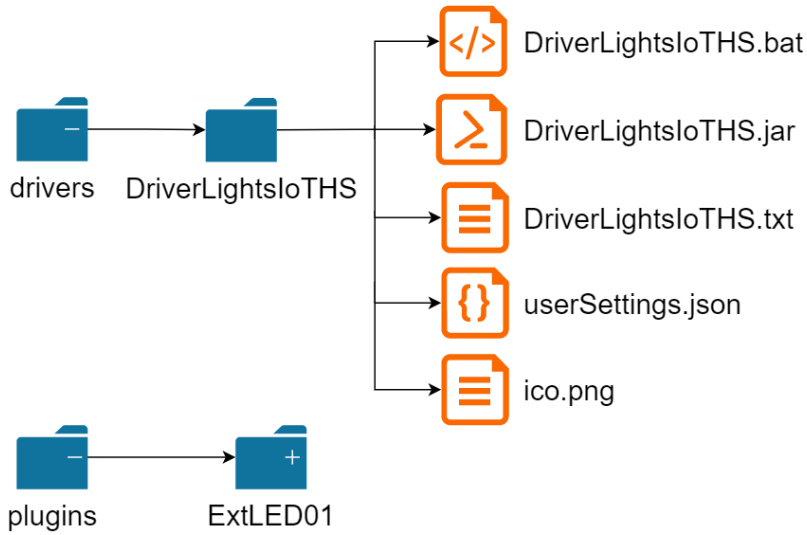
Obrázek 13 - příklad zdrojového kódu klientské části

Na výše uvedeném příkladu se nastavuje funkce, která se volá po úspěšném připojení k TAPI klienta, popřípadě pokud klient odešle příkaz k ukončení ZM nebo ovladače. Dále stejně jako u serverové části se musí specifikovat funkce pro přijetí dat od serverové části ZM nebo ovladače. Jak již bylo výše zmíněno, ZM nebo ovladač je spuštěný jako samostatná aplikace.

Je tedy nutné mu předat informace o tom, kde se má na obrazovce zobrazit a v jaké velikosti. Tyto funkce jsou volané vždy, když se změní pozice nebo velikost hlavní aplikace, tedy klienta. Každý ZM musí být přidělený alespoň do jedné místnosti. To je důvod, proč ve funkci „show“ je vstupním argumentem název místnosti, ve které uživatel daný ZM vybral. ZM tuto informaci může odeslat serverové části a ta poté odešle např. zařízení, která si uživatel přidal právě do této místnosti. V případě, kdy je scéna zobrazena, musí se zobrazené okno nastavit na nejvyšší prioritu, tedy aby jiné okno, respektive program nemohl ZM nebo ovladač překrýt. Např. v JavaFX knihovně lze nejvyšší prioritu přiřadit pomocí funkce „setAlwaysOnTop(true)“.

Instalace ZM a ovladačů na straně klientské části je trochu odlišná, než je tomu u serverové části. ZM je třeba instalovat do složky „plugins“, zatímco klientské části ovladače do složky „drivers“. Nastavení spouštěcích souborů je stejné jako u serverové části. U klientské části je možné vložit jazykový balíček (pod názvem „userSettings“), který obsahuje přeložený název daného ZM nebo ovladače. Pokud tedy je nainstalován ovladač „DriverLightsIoTHS“ a je ve složce nalezen jazykový balíček, je tento název přeložen např. na „Osvětlení“. Pokud balíček nalezen není, je ZM nebo ovladač prezentován svým názvem (tedy např. „DriverLightsIoTHS“).

Dále je vyžadováno ke každému ZM a ovladači přiložit ikonu, pod kterou se má daný ZM nebo ovladač prezentovat. Pokud není přidána žádná ikona, je zobrazen pouze název. Ikonu lze přidat pod názvem „ico.png“. Doporučená velikost ikony je 128 x 128 pixelů.



Obrázek 14 - souborová struktura klientská část

Příklad userSettings.json

```
{
    "title_cz":"Osvětlení",
    "title_def":"Lights"
}
```

3.14. Klient – Funkce a GUI

Klient nabízí mnoho funkcí, které může uživatel využívat a různě nastavovat. V záložce nastavení má uživatel možnost změnit název ovládacího panelu, zapnout/vypnout hlasové upozornění, zapnou hlídání stavu baterie ovládacího panelu, měnit vzhled a mnoho dalšího. Jedna z funkcí ovládacího panelu je funkce QuickControl.

Obrázek 15 - ukázka GUI nastavení

Tato funkce umožňuje volat skupiny, respektive automatické funkce pomocí různých gest na dotykovém displeji. Tuto funkci může uživatel aktivovat v nastavení a zvolit dobu, po kterou se tato funkce zobrazí. Zobrazení funkce se projeví tak, že celá obrazovka zčerná a uprostřed je pomocí ikony zobrazeno místo, kterého se má uživatel dotknout pro zobrazení hlavní nabídky. Pokud je zobrazen QuickControl, je možné celkem šesti gesty zavolat předem nastavené funkce. Na výše uvedeném obrázku je ukázán příklad, jak si může uživatel nastavit volání funkcí. V nastavení je dále možnost např. nastavit čas východu a západu slunce nebo vložit API klíč k OpenWeather API.

V dolní části se nachází ikony pro rychlé přepínání mezi různými scénami. Pomocí ikony domečku se uživatel dostane do seznamu místností. Do těchto místností je možné přidat skupiny označené jako profily a ZM.

Dále pomocí ikony ozubeného kolečka se uživateli zobrazí seznam vytvořených skupin. Na obrázku 16 jsou zobrazeny skupiny, respektive profily. U profilů je zobrazeno tlačítko „spustit“, zatímco skupiny je možné jen volat, proto se u nich žádné tlačítko na spuštění nenachází a nelze je tedy ručně spustit. V případě, kdy uživatel klikne na název skupiny, zobrazí se seznam automatických funkcí. V tomto seznamu může uživatel vidět, v jakém čase je daná funkce aktivní, jaký ovladač a zařízení používá, nastavenou podmínku atd. Automatickou funkci je pak možné editovat opět tím, že se klikne na její název. U každé automatické funkce je možnost její aktivace nebo deaktivace. V pravém horním rohu se

nachází tlačítka pro editaci skupiny nebo její odstranění. V horní části obrazovky je informativní panel, ve kterém systém zapisuje veškerou svou aktivitu, chybová hlášení atd.

Obrázek 16 - ukázka GUI skupiny

Do hlavní nabídky se uživatel může dostat pomocí přejetím prstem od dolní části obrazovky nahoru. V dolní části obrazovky se nachází šipka, kterou se uživatel dostane na předposlední zobrazený panel (stejně tak uživatel může přejet prstem od levé části obrazovky doprava). Dvojím klikem na šipku je zobrazena hlavní scéna. Pokud se uživatel nachází v hlavní nabídce, může vybrat možnost „zařízení“, kde se nachází klientské části ovladačů (tuto možnost lze vybrat pouze v případě, že je uživatel přihlášený na administrátorský účet). Ovladače nelze přiřadit do místností.

V pravé horní části displeje se zobrazují ikony o stavu baterie, stav připojení k síti a serveru, zda je uživatel přihlášený na svůj účet nebo jestli je spuštěný alarm. V levé horní části je zobrazeno aktuální počasí (pomocí OpenWeather API).

V případě, že je připojení se serverem ztraceno, je po 30 sekundách spuštěn alarm, aby byl uživatel na tuto událost upozorněn. Alarm lze vypnout v hlavní scéně. Globální alarm lze vypnout pouze v části zabezpečení pomocí bezpečnostního kódu.

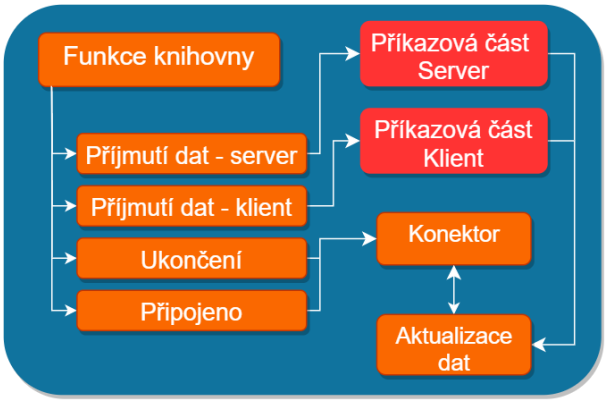
4. Návrh softwarového rozšíření systému

Veškerá softwarová rozšíření se provádí pomocí ZM nebo ovladačů. V dalších kapitolách je popsán návrh jednoho ZM, který poskytuje rozšířené možnosti pro ovládání osvětlení. Dále jsou navrženy celkem tři ovladače. První ovladač slouží ke komunikaci s pohybovými senzory, druhý s LED osvětlením a třetí zpřístupňuje zařízení od komerčního systému Philips Hue.

4.1. Ovladač – DriverMotionIoTHS

Úkolem tohoto ovladače je komunikace s pohybovými senzory navrženými speciálně pro tento projekt. Ovladač se pomocí serverové části připojí k danému senzoru. Jelikož tyto senzory lze použít i jako bezpečnostní, bude ovladač každé dvě sekundy kontrolovat, zda senzor stále odpovídá a nebyl odpojen od sítě. V případě, že je stav senzoru změněn (odpojení, připojení, pohyb atd.) je tato skutečnost poslána správci událostí pomocí příkazu „callEvent“.

Ovladač dodržuje všechna kritéria, která jsou stanovena v kapitole 2.2.5. Navržený ovladač také splňuje všechna doporučení o používání knihovny pro vývoj ovladače. V první části programu se specifikuje funkce, která je zavolána v případě, kdy je navázáno spojení se serverem. Dále se specifikuje funkce, kterou má knihovna zavolat v případě ukončení programu a funkce, které slouží pro přijetí dat od serveru nebo klientské části ovladače.



Obrázek 17 - základní princip ovladače

Na obrázku je předveden jeden z možných způsobů, jak navrhnout ovladač. Právě tímto způsobem je navržen i ovladač pro pohybové senzory. Jak je z obrázku patrné, existují dvě příkazové části. Serverová část obsahuje standardní 4 příkazy (viz kapitola 2.2.5.). Klientská část se už nemusí řídit žádným standardem ani dodržovat žádná předem určená pravidla. Jelikož klientská část ovladače slouží hlavně pro přidání senzoru nebo upravení nastavení, obsahuje příkazy pro přidání zařízení, úpravu a odstranění. U zařízení je možné pomocí klientské části nastavit IP adresu, komunikační port, stav (aktivovat/deaktivovat), přezdívku (v rámci IoT-HS systému) a název zařízení, pod kterým se má prezentovat v lokální počítačové síti.

Pro každé přidané zařízení je vytvořen samostatný tzv. konektor. Tento konektor se stará o spojení se zařízením. Konektory jsou vytvořeny pouze v případě, kdy je zavolána funkce „Připojeno“. Takto funkce je volána knihovnou po úspěšném připojení k serveru. Pokud je serveru odeslán příkaz na ukončení ovladače, je zavolána funkce „Ukončení“. Tato funkce ukončí veškerou komunikaci se zařízeními, uloží poslední známý stav připojených zařízení a ukončí ovladač.

Konektory se také starají o předávání událostí. Tedy pokud je zaznamenán pohyb, konektor tuto událost pomocí knihoven odešle serveru společně s daty o daném zařízení. Je také možné, že server změní stav zařízení (standardizovaný příkaz „DRIVER SET DEVICE STATUS“). Aktualizace dat probíhá tak, že konektor předá data o daném zařízení a funkce „Aktualizace dat“ změní pouze ty klíče, které byly aktualizované. Pozměněná data poté předá zpátky konektoru a ten tuto skutečnost odešle zařízení.

Pro změnu nastavení pomocí klientské části ovladače (příkazová část – klient) existují dva různé příkazy. Příkaz „DRIVER SET DEVICE STATUS“ je použitý v případě, kdy je provedena změna stavu zařízení. Příkaz „DRIVER SAVE SETTINGS DEVICE“ je používán pro změnu interních nastavení (např. změna IP adresy) a nastavení zařízení (např. změna názvu v rámci lokální sítě).

Příkaz „DRIVER SET DEVICE STATUS“ se tedy hodí např. pro vypnutí nebo zapnutí světla, změnu stavu senzoru atd. Restart konektoru není proveden, jelikož data jsou určena pouze pro zařízení. Restart zařízení není také proveden, jelikož se nejedná o změnu nastavení ale změnu stavu zařízení.

Příkazem „DRIVER SAVE SETTINGS DEVICE“ se mění IP adresa, port. Pokud je tedy změna nastavení provedena tímto příkazem, je proveden restart konektoru pro dané zařízení. Pokud je změněno nastavení zařízení, je proveden restart tohoto zařízení.

Princip těchto dvou příkazů je lépe vidět u ovladače „DriverLightsIoTHS“, jelikož obsahuje více stavů.

4.2. Ovladač – DriverLightsIoTHS

Tento ovladač slouží pro obsluhu LED osvětlení, které bylo speciálně navrženo pro tento projekt. Ovladač využívá stejný způsob návrhu jako ovladač pro pohybové senzory. U osvětlení se už posílá více konfiguračních dat, než tomu bylo u pohybových senzorů.

Příkazem „DRIVER SET DEVICE STATUS“ se aktualizují následující stavy zařízení.

Tabulka 16 - DriverLightsIoTHS příkazy

Stav (klíč)	Datový typ	Popis
state	bool	Zapnutí/vypnutí osvětlení.
colorR	long	Nastavení barvy (Red).
colorG	long	Nastavení barvy (Green).
colorB	long	Nastavení barvy (Blue).
brightness	long	Nastavení jasu.

Výše standardizované stavy jsou odeslány danému zařízení a jelikož se jedná pouze o změnu stavu (např. změna barvy), není třeba restartovat zařízení.

Příklad změny stavu

```
{
  "command":"DRIVER SET
            DEVICE STATUS",
  "id":"id123",
  "brightness":255
}
```

Přeposlání změny stavu

```
{
  "command":"SET STATUS",
  "brightness":255
}
```

Příkazem „DRIVER SAVE SETTINGS DEVICE“ se aktualizuje následující nastavení daného zařízení.

Tabulka 17 - DriverLightsIoTHS příkazy nastavení

Nastavení (klíč)	Datový typ	Popis
nickName	string	Název zařízení v rámci IoT-HS systému
IPaddr	string	IP adresa zařízení.
port	long	Port zařízení.
LEDcount	long	Počet led na jeden metr.
LEDspeed	long	Rychlost rozsvícení LED.
profile	long	Volba Profilu 1-3.
name	string	Název zařízení v rámci lokální počítačové sítě.

Příklad změny nastavení

```
{
  "command":"DRIVER SAVE
             SETTINGS DEVICE",
  "id":"id123",
  "profile":1
}
```

Přeposlání změny nastavení

```
{
  "command":"SET SETTING",
  "profile":1
}
```

Příkaz „DRIVER SET DEVICE STATUS“ se vyskytuje také v serverové příkazové části. Ovladače „DriverMotionIoTHS“ a „DriverLightsIoTHS“ tedy umožnují měnit stav zařízení (např. vypnout/zapnout), ale neumožňují měnit jeho nastavení. Nastavení může měnit pouze klientská část ovladače.

Ovladačem je možné nastavit počet LED diod, které má kontrolér ovládat pomocí „LEDcount“ (LED/m). Pomocí „LEDspeed“ lze určit jakou rychlostí se mají rozsvěcet, popřípadě zhasínat. Posledním nastavením je „profile“. Tímto nastavením se určuje, jestli se mají LED diody rozsvěcet pomocí tzv. fade efektu nebo postupně např. zleva doprava.

4.3. Ovladač – Philips Hue

Ovladač Philips Hue poskytuje zařízení, která jsou určena k osvětlení. Např. chytré žárovky, lustry, lampy, LED pásky atd. Philips Hue také poskytuje pohybové senzory nebo různé ovládací prvky. Tato zařízení tento ovladač spravovat nemůže, jelikož je typu „Lights“. Struktura návrhu ovladače je stejná jako v obou předchozích případech. Nastává ovšem problém v aktualizaci stavu zařízení.

IoT-HS systém poskytuje pomocí API registraci do správce události. Pokud je tedy vyvíjen ZM, klient nebo jiný program, je možné využívat API a pomocí správce událostí registrovat daná zařízení do správce událostí a pokud je stav zařízení změněn je o tom např. ZM informován. Philips Hue žádnou takovou funkci nemá, a tedy není jiného řešení než periodicky kontrolovat zařízení, zda nebyl změněn jejich stav (např. zhasnutí, rozsvícení, změna barvy atd.).

Klientská část pro Philips Hue slouží pouze pro vložení API klíče, díky kterému lze REST API využívat. Přidávání, odstraňování a úpravy zařízení je možné provádět v oficiální aplikaci od Philips Hue. Tento ovladač se tedy pouze spojí s Philips Hue bridgem a stáhne si všechny informace o zařízeních, která obstarávají osvětlení. Konektor tedy musí být v tomto případě upraven. Než se začnou vytvářet konektory pro jednotlivá zařízení, je nutné nejdříve určit o jaká zařízení se jedná a pro ty vytvořit jednotlivé konektory, které se budou dotazovat Philips Hue bridge na stav daného zařízení.

4.4. ZM – LightsIoTHS

Jelikož ovladač obstarává pouze komunikaci se zařízením je třeba vytvořit ZM, kterým bude moci uživatel ovládat daná zařízení. Server obsahuje základní funkce pro ovládání zařízení jako např. vypnutí nebo zapnutí, ale již není možné nastavit barvu nebo jas. Toto nastavení je možné změnit pouze pomocí automatických funkcí. Automatická funkce ale nastaví hodnotu a uživatel si tak nemůže vybrat barvu podle svého uvážení např. pomocí barevné palety nebo změnit jas. Pomocí tohoto ZM uživatel může jednoduše měnit nastavení daného zařízení nebo skupin zařízení.

Jelikož ZM nemá přímý přístup k zařízením, je třeba si zařízení vyžádat od ovladačů pomocí příkazu „DRIVER COM“, ve kterém se specifikuje, že se požadují pouze ta zařízení, respektive ovladač, který má "driverDeviceType" nastaveno na hodnotu „Lights“. Tedy server vrátí pouze zařízení od ovladačů, která nesou toto označení. Tato zařízení jsou pak zobrazena uživateli v klientské části ZM. Uživatel si pak může vybrat, jaká zařízení chce do dané místnosti vložit. Vybraná zařízení se pak pošlou serverové části ZM, která uloží ID vybraných řízení do databáze s přidělenou místností.

Uložená zařízení se pak musí pomocí API serveru registrovat do správce událostí funkcí „addEvent“, do které se předá druh události „UPDATE“, název ovladače pro registrované zařízení a jako poslední ID zařízení. Díky tomuto systému jsou změny okamžitě distribuovány klientské části ZM a uživatel může hned pozorovat změny daného zařízení.

U ZM není nijak definováno, jaké příkazy musí mít specifikované nebo pravidla, kterými se musí řídit, jak je tomu u ovladačů. Je možné využívat stejný princip, jak je tomu u ovladačů. Jediný rozdíl je v tom, že není třeba implementovat konektor a je zapotřebí pozměnit příkazové části. Serverová příkazová část obsahuje pouze příkaz „CALL EVENT“. Tímto příkazem server, respektive správce událostí, odešle ZM změnu registrovaných zařízení. Klientská příkazová část obsahuje šest jednoduchých příkazů, díky kterým si klientská část může např. vyžádat uložená zařízení nebo přidat nová zařízení.

Tabulka 18 - příkazy LightsIoTHS

Příkaz	Popis
GET DEVICES	Vyžádání všech zařízení od ovladačů. Pouze ty ovladače, které jsou označeny jako „Lights“.
SET DEVICE STATUS	Nastavení stavu zařízení.
GET SAVED DEVICES	Vyžádání uložených zařízení.
DELETE DEVICE	Odstranění zařízení podle ID.
ADD DEVICE	Přidání zařízení.

První příkaz se využívá v případě, kdy chce uživatel přidat zařízení do místnosti. Uživateli je tedy ukázán seznam všechny dostupných zařízení a z tohoto sezamu si může vybrat zařízení, které pak bude přidáno pomocí příkazu „ADD DEVICE“. Příkazem „GET SAVED DEVICES“ si klientská část vyžádá zařízení, která byla přidána pro specifickou místnost.

Příklad žádosti

```
{
 "command":"GET SAVED DEVICES",
 "room":"Kuchyň"
}
```

```
{
 "command":"ADD DEVICE",
  "device":{
   "id":"id789",
   "brightness":58,
   "room":"Kuchyň"
  }
}
```

Příklad odpovědi

```
{
 "command":"SAVED DEVICES",
  "devices":[
  {
   "id":"id123",
   "brightness":58
  },
  {
   "id":"id456",
   "brightness":120
  }
 ]
}
```

Příkazem „GET SAVED DEVICES“ ZM přijme veškerá zařízení a k nim veškeré informace o stavu nebo nastavení daného zařízení. Stejně tak správce událostí odesílá veškeré informace o zařízení (tedy neposílá pouze to co se změnilo, ale vše včetně změněného atributu, respektive klíče). ZM si tedy ukládá o tomto zařízení všechny informace a ty jsou posílány klientské části ZM. Klientská část poté vybere pouze klíče, které jsou pro ni relevantní (např. barvy, jas, stav zařízení atd.) a ty zobrazí uživateli.

5. Návrh hardwarového rozšíření systému

Pro tento projekt byly navrženy řídicí jednotky pro pohybové senzory a ovládání osvětlení. Aby bylo možné zkonstruovat tyto jednotky, byly navrženy univerzální plošné spoje pro wifi moduly NodeMCU ESP8266, NodeMCU ESP32 a mini NodeMCU ESP8266 (WeMos D1 mini).

Princip fungování řídicích jednotek je takový, že každá jednotka založí server (TAPI), na který se může připojit právě jeden ovladač. Pro komunikaci s řídicími jednotkami mohou ovladače využívat následující příkazy.

Tabulka 19 - příkazy pro komunikaci s řídicími jednotkami

Příkaz	Popis
GET STATUS LIGHT	Jednotka odešle aktuální stav o osvětlení (zapnuto/vypnuto, barvu a jas)
GET STATUS MOTION	Jednotka odešla stav pohybového senzoru.
SET STATUS	Nastavení stavu jednotky.
SET SETTINGS	Změna nastavení jednotky (jednotka se restartuje).

První příkaz se vyskytuje pouze v jednotkách, které obsluhují osvětlení. Druhý příkaz mají k dispozici pouze jednotky, které obsluhují pohybové senzory. Poslední dva příkazy mohou mít oba druhy jednotek („SET STATUS“ v pohybovém senzoru ale nemá opodstatnění).

Příkaz **„SET STATUS“** musí obsahovat 5 povinných klíčů „colorR“, „colorG“, „colorB“, „state“ a „brightness“ (viz tabulka 16). Stejně tak příkaz **„SET SETTINGS“** musí obsahovat klíče „LEDcount“, „name“, „LEDseed“ a „profile“ (viz tabulky 17).

Řídicí jednotku k Wi-Fi síti lze připojit následujícími způsoby.

Pomocí WPS

Pomocí standardu WPS, je možné snadno a rychle připojit zařízení do sítě bez zadání hesla a SSID. WPS lze na řídicích jednotkách zapnout tak, že se zařízení nejprve odpojí od napájení. Při opětovném zapojení se musí držet reset tlačítko alespoň na dvě sekundy. Pokud je jednotka úspěšně připojena k síti, jednou blikne pomocí LED (jednotka pro pohybové senzory) nebo blikne připojeným LED osvětlením (jednotka pro osvětlení, např. LED pásek).

Pomocí webové stránky

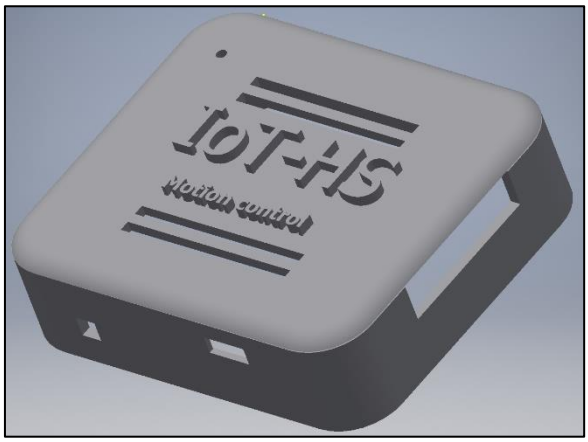
Dalším způsobem, jak jednotku připojit k síti, je pomocí webové stránky, kde uživatel zvolí SSID sítě a heslo. V případě jednotky pro LED osvětlení je možné na webové stránce také nastavit název zařízení a počet LED/m. Při testování se NodeMCU ESP8266 moduly osvědčily jako velmi pomalé, když byly nastaveny v AP módu. Připojení např. chytrého telefonu trvalo déle jak 1 minutu. Byl tedy navržen alternativní postup pro připojení chytrého telefonu a ESP do jedné sítě. Pokud je ESP modul resetován do základního nastavení, automaticky se připojuje k síti s názvem „IoTHSwifi“ a tato síť musí mít nastavené heslo „IoTHSwifi123“. Uživatel může toto AP vytvořit buď pomocí svého např. chytrého telefonu

nebo pomocí routeru. Poté stačí zjistit IP adresu jednotky a připojit se na její webovou stránku na portu 80, kde lze změnit nastavení sítě. I když se tato technika připojování k síti může zdát jako zdlouhavá (hlavně při zjišťování IP adresy jednotky), je nakonec v praxi rychlejší než čekání na připojení např. telefonu k ESP modulu. Hlavně pokud se nastavuje více zařízení najednou, je na každém nastavování zařízení úspora alespoň 1 minuta. V budoucnu je plánovaná aplikace pro chytré telefony, která sama najde jednotku v síti a uživatel tak nemusí zjišťovat IP adresu jednotky (hlavně pro laického uživatele může být tento krok celkem obtížný). Pokud je tedy jednotka připojena k síti, je třeba jednou rychle zmáčknout reset tlačítko, aby se zapnul webový server. Po uložení nových nastavení je jednotka restartována a připojí se k nastavené síti.

Pokud si uživatel bude chtít do místnosti dát pouze jednu jednotku na ovládání osvětlení a zároveň bude chtít toto osvětlení zapnout na základě pohybu, je jistě neúsporné do této místnosti dávat dvě jednotky. Proto byla vytvořena třetí jednotka, která kombinuje vlastnosti obou jednotek. K této jednotce lze tedy připojit osvětlení (např. LED pásky) a zároveň pohybový senzor.

Tato jednotka má k dispozici mód, který je nazván jako „Stand alone“. Pokud je tento mód vybrán, jednotka se nepřipojuje k síti, ale pouze zapíná osvětlení na základě pohybu a po určitém čase osvětlení vypne. Tento mód lze nastavit pouze pomocí webových stránek.

5.1. Návrh krabiček pro řídicí jednotky

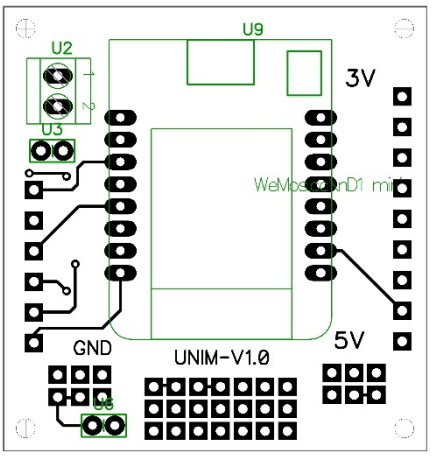


Obrázek 18 - návrh krabičky pro řídicí jednotku

Pro řídicí jednotky byly navrženy tři typy krabiček, které zajistí fyzickou ochranu bezdrátových modulů a dalších hardwarových součástek. Krabička pro pohybový senzor je navržena pro mini NodeMCU ESP8266. Její rozměry jsou 76x65 mm. Krabičku lze vybavit mini USB konektorem pro napájení, reset tlačítkem a svorkovnicí se 6 vstupy, ke které lze připojit až 3 pohybové senzory. Jednotka kontaktuje ovladač v případě, když je zaznamenán pohyb alespoň jedním senzorem. Senzory od sebe nelze rozlišit. Krabička dále obsahuje dostatek místa na upevnění dalších bezpečnostních senzorů nebo jiných komponentů jako např. senzor otřesu, teploty atd. Stejný design má krabička pro řídicí jednotku pro ovládání osvětlení. Krabičku lze opět vybavit mini USB konektorem, reset tlačítkem, svorkovnicí se 2 vstupy pro externí napájení osvětlení a svorkovnicí se 3 vstupy pro ovládání LED pásků nebo jiného osvětlení (GND, VCC, ovládací pin pro 5V LED pásky). Krabička pro jednotku kombinující funkce osvětlení a pohybového senzoru je nazvaná jako „MLED“ (Motion LED). Krabička má celkový rozměr 78x109 mm a je určena pro NodeMCU ESP8266 nebo NodeMCU ESP32. Krabičku lze vybavit svorkovnicí se 2 vstupy (napájení), 3 vstupy (ovládání osvětlení) a další 3 vstupy pro připojení pohybového

senzoru. Dále lze přidat napájení pomocí mini USB a ON/OFF přepínač. Další ukázky krabiček lze nalézt v přílohách 3-6.

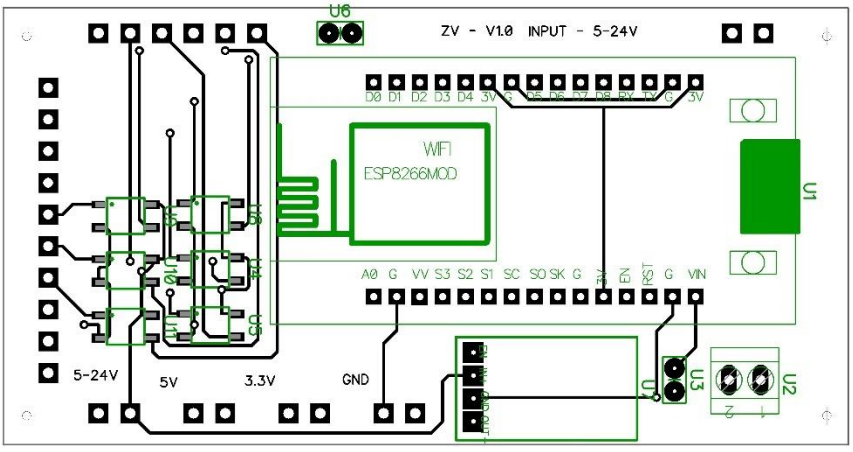
5.2. Návrh plošných spojů pro řídicí jednotky



Obrázek 19 - plošný spoj UNIM-V1.0

Pro tento projekt byly navrženy tři druhy plošných spojů. První druh, pod názvem UNIM-V1.0, je určený pro mini NodeMCU ESP8266 a je určený do krabiček s rozměrem 76x65 mm (plošný spoj má celkovou velikost 51x48,2 mm). Plošný spoj lze vybavit svorkovnicí pro připojení napájení (U2), kondenzátorem (U3). U návrhu plošného spoje bylo počítáno s tím, že se bude připojovat reset tlačítko. Tlačítko musí být připojeno na pin U9:D2 a na pozici U6 plošného spoje musí být připojený odpor. Plošný spoj má dále dostatek místa pro připojení dalších periférii a poskytuje 5V nebo 3,3V výstupy.

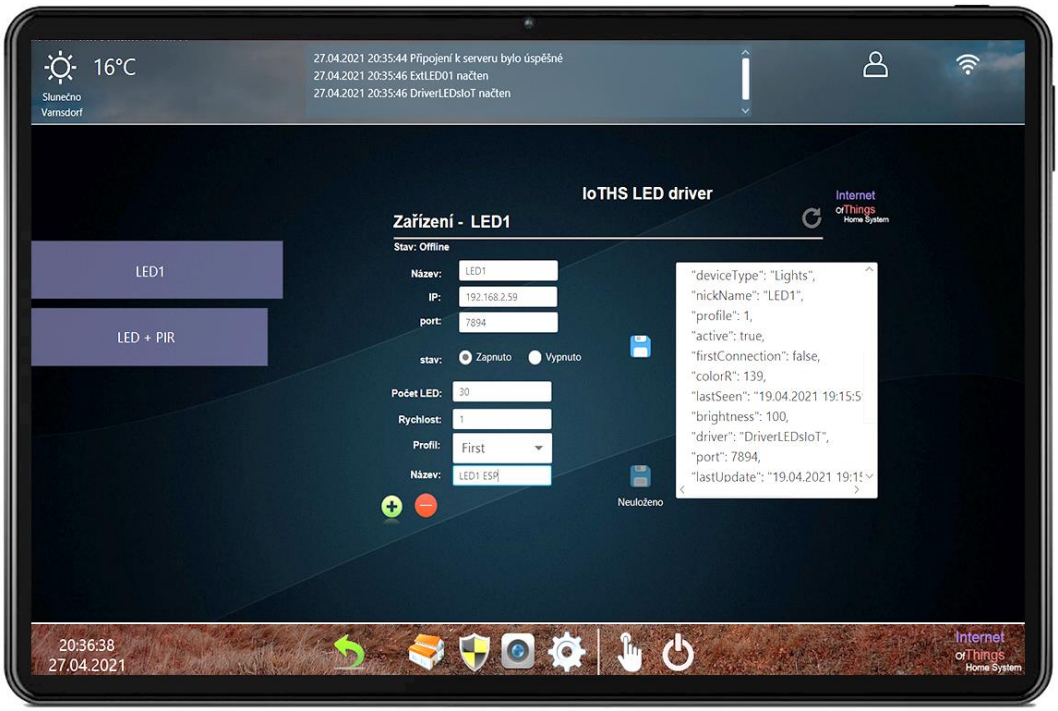
Plošný spoj druhého druhu, pod názvem UNI-V1.2 je určený pro NodeMCU ESP8266/ESP32 a je přizpůsobený do krabiček s rozměrem 78x109 mm (plošný spoj má celkovou velikost 93,3x48,2 mm). Tento plošný spoj obsahuje vše jako UNIM s tím, že je možné připojit tři polovodičová relé. Oba druhy plošných spojů musí být napájeny 5 V, jelikož NodeMCU toto napájení vyžaduje. Relé tedy mohou napájet pouze zařízení, která mají vstupní napájení do 5 V. Pokud je třeba napájet zařízení, které požaduje vyšší vstupní napětí, je zde připraven třetí druh plošného spoje, pod názvem ZV-V1.0. K tomuto plošnému spoji lze připojit pět polovodičových relé. Vstupem může být napětí mezi 5-24 V. K desce je třeba připojit Step Down modul typu „Step down mini buck“, který vstupní napětí sníží na 5 V, aby bylo možné napájet ESP modul. Vstupní napětí je ale vedeno na vstupy polovodičových relé a Step Down modul nemá vliv na výstupní napětí z relé.



Obrázek 20 - plošný spoj ZV-V1.0

6. Implementace softwarového rozšíření systému

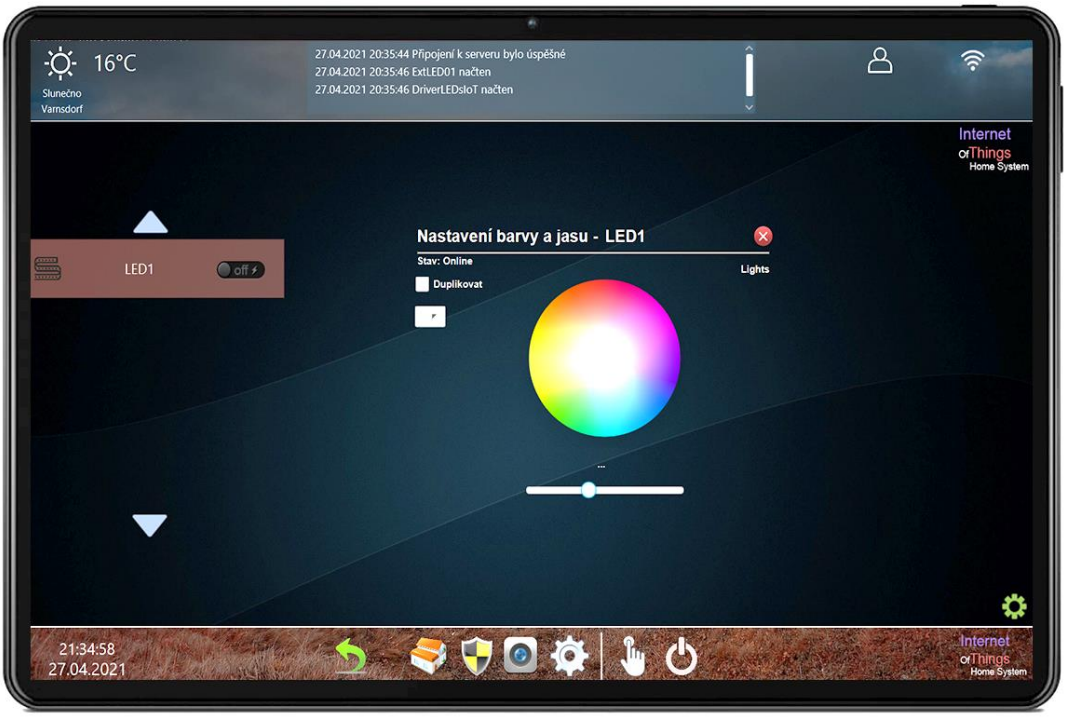
Aby bylo možné otestovat funkčnost IFTTT služby, byly implementovány tři ovladače z kapitoly 2. Ovladač pro pohybové senzory, osvětlení (DriverLightsIoTHS) a ovladač pro ovládání světel Philips-hue. Dále byl vytvořen ZM „LightsIoTHS“ pro otestování funkčnosti knihoven jak na serverové části, tak na klientské části. Na následujícím obrázku je klientská část ovladače „DriverLightsIoTHS“. Uživatel může přidat libovolný počet zařízení a upravit nastavení řídicích jednotek.



Obrázek 21 - ukázka GUI klientské části

Pro pokročilé uživatele se v pravé části nachází výpis všech nastavení daného zařízení v JSON formátu. Uživatel může přidat nové zařízení pomocí plus ikony. V případě, že je přidáno nové zařízení, je ihned zobrazeno v levé části obrazovky a je nastaveno na základní nastavení. Uživatel tedy klikne na zvolené zařízení a může změnit nastavení. Klientská část ovladače neaktualizuje informace o zařízení v reálném čase. Pokud tedy chce uživatel vidět aktuální informace o daném zařízení, je třeba kliknout na ikonu obnovení v pravé horní části obrazovky.

První čtyři nastavení (název, IP, port a stav) se týkají nastavení konektoru v serverové části obrazovky. Restart zařízení není třeba, po změně nastavení se konektor okamžitě připojí k zařízení. Poslední čtyři nastavení (počet led, rychlost, profil a název) se týkají nastavení zařízení, které je popsáno v tabulce 17. Na dalším obrázku je ukázka GUI pro ZM „DriverLightsIoTHS“. Uživatel má možnost změnit barvu pomocí barevné palety nebo pomocí pokročilejší palety, ve které lze zadávat hodnoty RGB. Pokročilejší paletu lze aktivovat pomocí tlačítka, které je umístěné pod nápisem „Duplikovat“. Dále má uživatel možnost změnit jas, vidět stav zařízení a název ovladače daného zařízení.



Obrázek 22 - ukázka GUI klientské části ZM

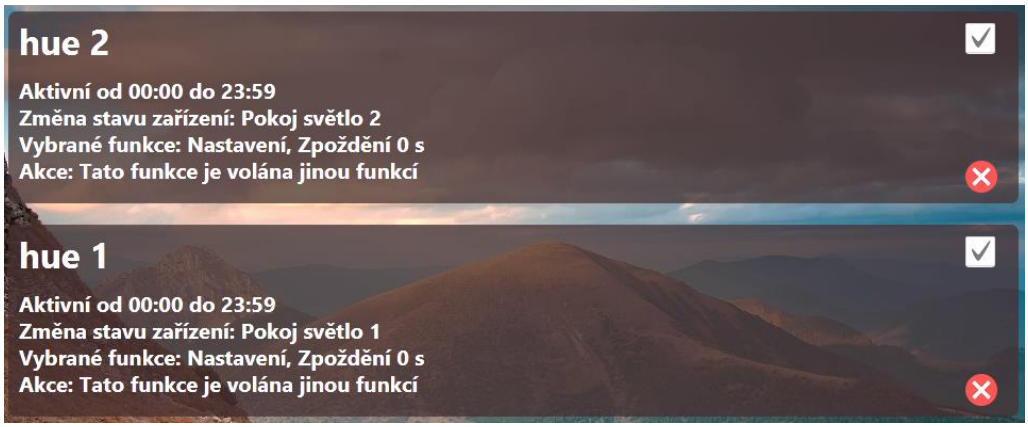
Pokud chce uživatel přidat zařízení do ZM, musí přejít do sekce nastavení. Klientská část ZM kontaktuje serverovou část a ta pošle všechna zařízení, která se starají o osvětlení. Tato zařízení jsou pak zobrazena v seznamu a uživatel má možnost přidat dané zařízení do zvolené místnosti.

7. Testování a realizace hardwarového rozšíření systému

Aby bylo možné realizovat řídicí jednotky, bylo nejdříve zapotřebí vyrobit plošné spoje. Pro výrobu plošných spojů byla vybrána firma Gatema, které byly poslány návrhy plošného spoje UNIM-V1.0 a ZV-V1.0. Pomocí těchto plošných spojů byly vytvořeny řídicí jednotky pro pohybový senzor a LED osvětlení. Krabičky pro řídicí jednoty byly vyrobeny pomocí 3D tiskárny. Fotografie plošných spojů a krabiček lze nalézt v přílohách 7-10.

Pro testování serveru byl vybrán jednodeskový počítač Rasbperry Pi model B s operačním systémem Ubuntu MATE. Klient byl testován na tabletu s operačním systémem Windows 10. Dále bylo vytvořeno několik automatických funkcí pro otestování funkčnosti IFTTT systému a ovladačů. Jako první byl otestován ovladač pro ovládání Philips-Hue zařízení společně s IoT-HS zařízeními.

Nejprve byla vytvořena automatická funkce, která hlídá, zda je zapnuté osvětlení v dané místnosti. Pokud ano, rozsvítí se osvětlení ovládané IoT-HS řídicí jednotkou. Další funkce byla vytvořena pro otestování IoT-HS pohybového senzoru. Pokud je zaznamenán pohyb, rozsvítí se Philips-Hue světlo.



Obrázek 23 - automatické funkce

Na obrázku 23 je ukázán příklad na kterém lze otestovat ovladač pro světla od Philips-Hue. Funkce jsou součástí skupiny „Hue“. Tuto skupinu lze nastavit ve funkci Quick Control. Uživatel pak jednoduše může rozsvítit světla pomocí gest na dotykovém displeji.

Při testování bylo zjištěno, že ovladač pro Philips-Hue má zpoždění při detekci změn stavu zařízení. To je způsobeno tím, že API neposkytuje zachytávání událostí a ovladač každou sekundu musí prověřovat, zda nedošlo u nějakého zařízení ke změně.

U ovladače DriverLightsIoTHS je pomalá reakce u rozsvícení, zhasnutí nebo změny světel při rychlém přepínání stavu. Pokud uživatel změní nastavení, je třeba počkat alespoň 500 ms než se provede další změna. Zpoždění pak není znatelné.

8. Závěr

V rámci práce byl zcela předělán původní systém pro chytrou domácnost IoT-HS. Systém nebyl navržen pro tak zásadní implementaci nové funkce. Musela být předělána celá struktura a logika tohoto systému. Nyní systém disponuje zcela novými funkcemi, díky kterým je možné jednoduše implementovat nové služby. Do systému byla implementována nová služba IFTTT, která nyní tvoří velmi důležitou část v domácí automatizaci. Díky této službě je možné, aby si uživatel nastavil libovolnou automatickou funkci, respektive scénář, ve kterém je možné využívat zařízení od různých výrobců. Aby bylo možné využívat zařízení od komerčních systémů jako např. Philips-Hue nebo Fibaro, byly navrženy knihovny, díky kterým lze vytvářet ovladače. Ovladač pak zpřístupňuje daná zařízení od komerčních systémů a zajišťuje plnou kompatibilitu s IoT-HS systémem pomocí standardizace dat. Pro rozšíření funkčnosti systému lze vytvářet ZM, které většinou slouží pro přehlednější ovládání různých zařízení na straně uživatele. Dále bylo vytvořeno nové grafické rozhraní pro snadné vytváření automatických funkcí nebo ovládání zařízení. Uživatelské rozhraní bylo navrženo s ohledem na to, aby vytváření automatických funkcí bylo snadné a přehledné i pro laického uživatele. Dále byly vytvořeny řídicí jednotky pro hardwarové rozšíření systému. Jedná se o pohybové senzory a jednotku pro osvětlení. Pomocí těchto jednotek pak byly testovány různé ovladače a samotná IFTTT automatizace.

Systém je plně funkční a je možné ho využít v plném provozu. Hlavní přednosti tohoto sytému oproti ostatním je jeho univerzálnost (lze nainstalovat na jakékoliv zařízení s podporou JVM a není třeba kupovat drahý bridge), snadná rozšiřitelnost pomocí ovladačů a ZM, snadné vyvíjení ZM a ovladačů pomocí připravených knihoven a intuitivní grafické rozhraní, které umožňuje vytvářet jednoduché nebo složité automatické funkce.

Služba IFTTT byla testována pomocí IoT-HS řídicích jednotek společně s Philips-Hue zařízeními. Testovalo se převážně to, jestli ovladače dokáží provést takovou abstrakci zařízení, že IoT-HS bude moci zařízení od různých firem, využívat v rámci jednoho systému. I když tyto testy byly úspěšné, je určitě možné tento systém dále rozšiřovat o nové funkce jako např. vylepšení zabezpečení komunikace, přidání dalších ovládacích prvků, které by usnadnily uživateli práci se systémem, nebo přidání hlasového asistenta.

SEZNAM POUŽITÉ LITERATURY

[1] RANDL, Milan. Jaký vybrat systém domácí automatizace? Loxone [online]. 18. 8. 2020 [cit. 2021-4-29]. Dostupné z: https://www.loxone.com/cscz/blog/system-domaci-automatizace/Zdroj2

[2] Co jsou to IFTTT služby? [online]. [cit. 2021-5-5]. Dostupné z: https://www.eon.cz/radce/chytra-domacnost/chytre-domy-a-chytra-domacnost/jak-pouzivat-chytrou-zarovku-2/co-jsou-to-ifttt-sluzby

[3] ROUSE, Margaret. What is applet. TechTarget [online]. 2017 [cit. 2021-5-5]. Dostupné z: https://whatis.techtarget.com/definition/applet

[4] GRIFFITH, Eric. The 25 Best IFTTT Applets [online]. 2019 [cit. 2021-5-5]. Dostupné z: https://www.pcmag.com/news/the-25-best-ifttt-applets

[5] Introduction. IFTTT [online]. [cit. 2021-5-5]. Dostupné z: https://platform.ifttt.com/docs

[6] [HC2/HCL] Creating Block Scenes. Fibaro manuals [online]. [cit. 2021-5-5]. Dostupné z: https://manuals.fibaro.com/knowledge-base-browse/block-scenes/

[7] Úvod do programování v jazyce Lua pro Fibaro HC2. Yatun [online]. 2019 [cit. 2021-5-5]. Dostupné z: https://podpora.yatun.cz/cs/support/solutions/articles/9000065681-uvod-do-programov%C3%A1n%C3%AD-v-jazyce-lua-pro-fibaro-hc2

[8] VRBÍK, Petr. Vliv světla na naše zdraví aneb hygiena osvětlování. Světlo časopis pro světlo a osvětlování [online]. 2015 [cit. 2021-5-5]. Dostupné z: http://www.odbornecasopisy.cz/svetlo/clanek/vliv-svetla-na-nase-zdravi-aneb-hygiena-osvetlovani--1294

[9] How Voice Assisted Technology is Being Used [online]. [cit. 2021-5-7]. Dostupné z: https://www.carbontrack.com.au/blog/voice-assisted-techchnology/

Seznam příloh

Příklady požadavků a odpovědí pro ZM a ovladače

	ODPOVĚĎ

{
"command":"**LOG**",
"type":[ERROR|WARN|INFO]",
"text":"{text}"
}	

{
"command":"**ECHO**"
}
	{
	"command":"**ECHO**",
	"message":"Hi, this is {název_ZM/ovladače}"
	}

{
"command":"**GET DBS**"
}
	{
	"command":"**SET DBS**",
	"url":"{url }",
	"user":"{uživatel}",
	"pass":"{heslo}"
	}

{
"command":"**DRIVER COM**",
"driver":"{název_ovladače}",
"comType":"{}",
"deviceType":"{typ_ovladače}",
"driverCommand":"{příkaz_pro_ovladač}"
}
	{
	"command":"**DRIVER ANS",**
	"answeredTo":"{příkaz_pro_ovladač}",
	"{název_ovladače}":[
	data/odpověď
	]
	}

{
"command":"**ADD EVENT**",
"event":"{typ_eventu}",
"driver":"{název_ovladače}",
"id":"{id_sledovaného_zařízení}"
}	

Příkazy pro komunikaci s API serveru

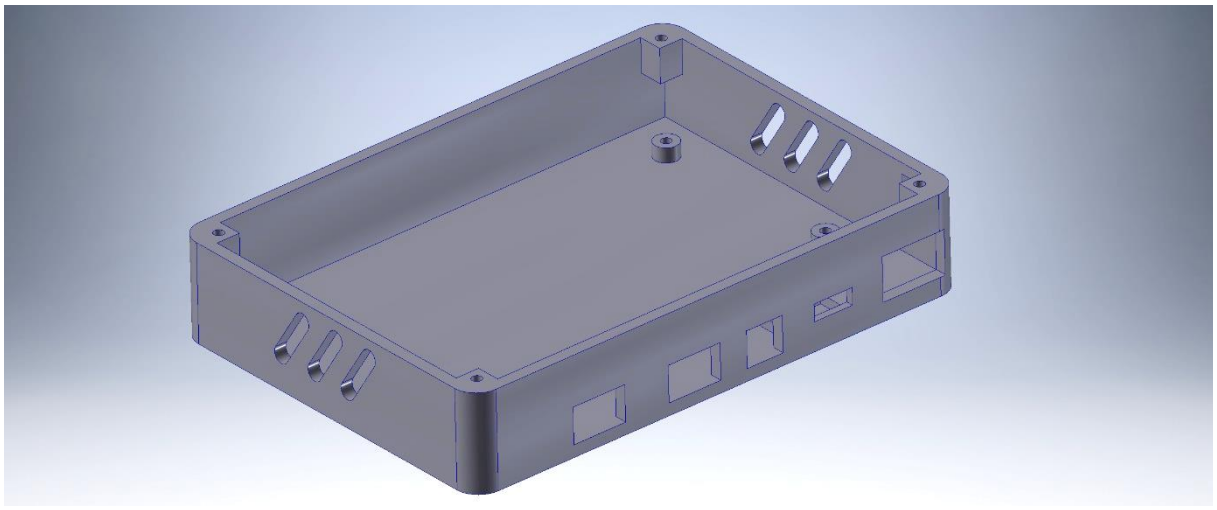
Tyto příkazy nejsou dostupné pro ZM ani ovladače.

Příkaz	KLÍČ:DATOVÝ_TYP
GET PLUG	
GET DRIVERS	
GET GROUPS	
START PROFILE GROUP	id:string
ADD GROUP	name:string, ico:string, isProfile:bool
EDIT GROUP	groupID:string, name:string, ico:string, isProfile:bool
DELETE GROUP	groupID:string
GET AUTO FUNCTION	groupID:string
ADD AUTO FUNCTION	Viz příloha 3
EDIT AUTO FUNCTION	Viz příloha 3
DELETE AUTO FUNCTION	functionID:string
DEACTIVATE AUTO FUNCTION	functionID:string
ACTIVATE AUTO FUNCTION	functionID:string
GET DEVICES FOR AUTO	driverCommand:string, driver:string, deviceType:string, deviceTypeCom:string
ADD ROOM	name:string
EDIT ROOM	roomID:string, name:string
DELETE ROOM	roomID:string
GET ROOMS	
GET LANGUAGE PACKAGE	driver:string

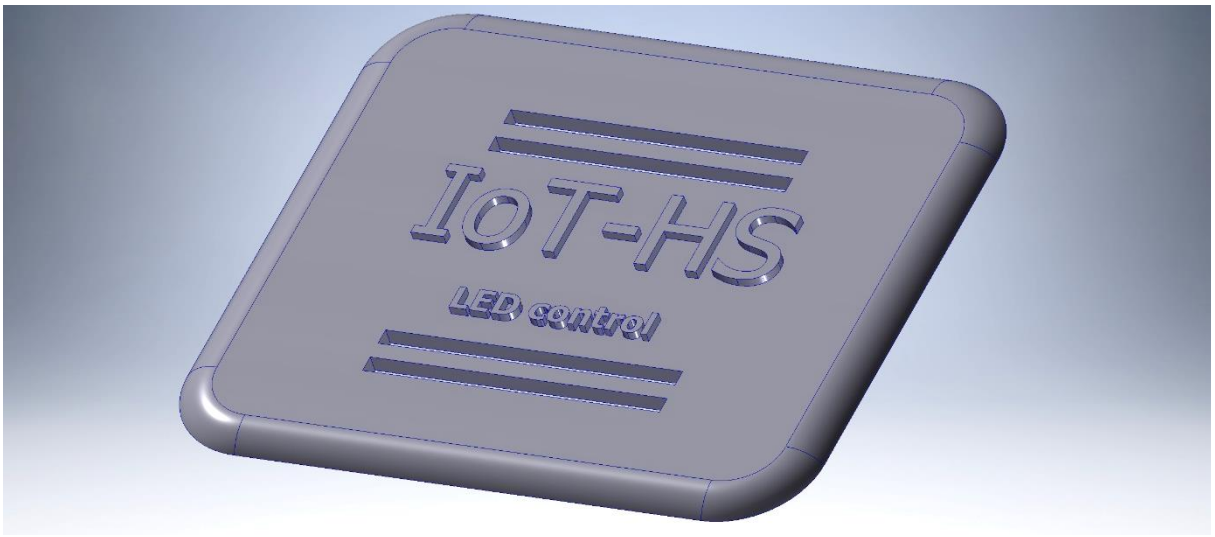
Příklad automatické funkce – JSON

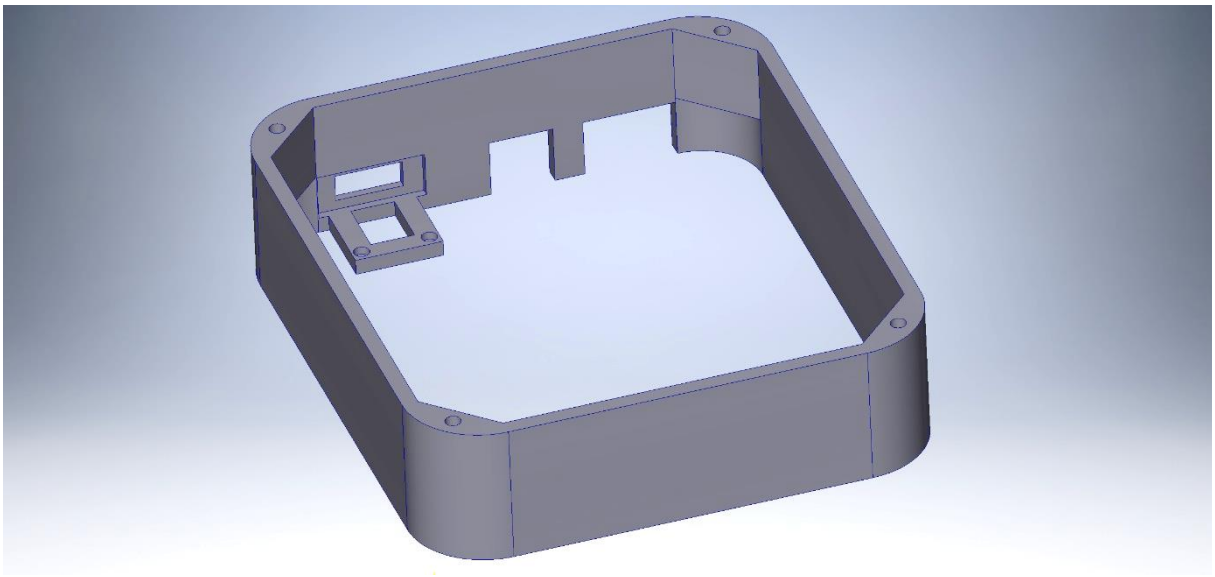

```
{
  "set":true,
  "call":"939b2b42-ca8d ",
  "email":true,
  "group":"5",
  "ifKey":"motion",
  "action":true,
  "setKey":"state",
  "countTo":255,
  "ifOrSet":"if",
  "ifValue":true,
  "restart":true,
  "timeToH":"23",
  "timeToM":"59",
  "doAction":"deactivate function",
  "ifDevice":"5a5bd084-ae18",
  "ifDriver":"DriverMotionIoT",
  "setValue":true,
  "callEvent":"Event",
  "callGroup":false,
  "restartTo":0,
  "setChoice":"set",
  "setDevice":"24",
  "setDriver":"DriverPhilipsHue",
  "textEmail":"Tohle je text e-mailu",
  "timeFromH":"00",
  "timeFromM":"00",
  "functionID":"fc2b1025-1b68
  "ifCondition":"=",
  "actionTimeAD":0,
  "callFunction":true,
  "functionName":"Příklad funkce",
  "subjectEmail":"Tohle je předmět e-mailu",
  "TimerCallGroup":false,
  "actionCallGroup":true,
  "specialFunction":"TimerCall",
  "actionFunctionCall":"1",
  "specialFunctionCall":"e14b5ce7-c9cc“,
  "specialFunctionTime":9,
  "specialFunctionUnits":"s"
}
```

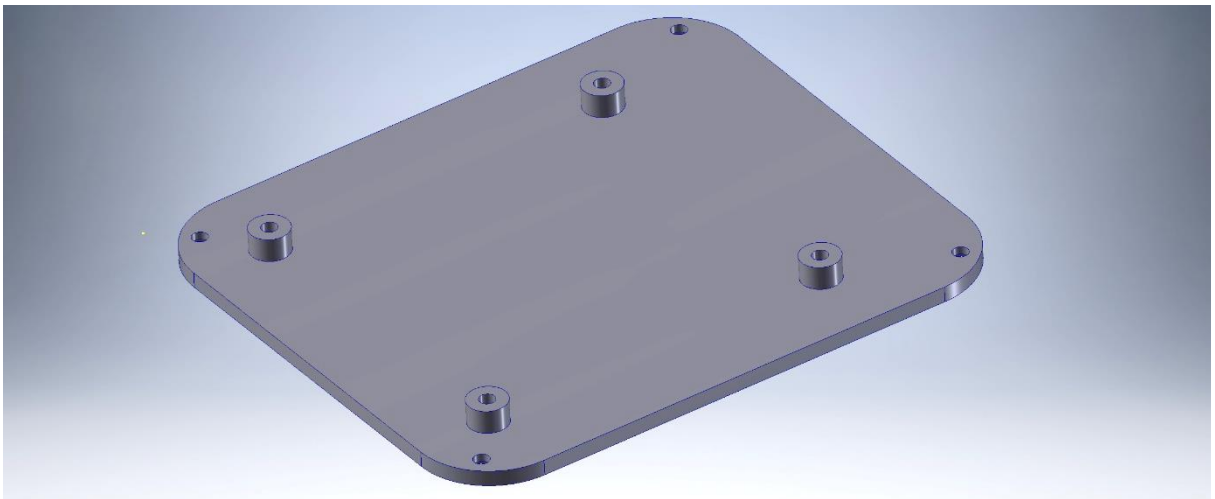
Model krabičky pro řídicí jednotku MLED



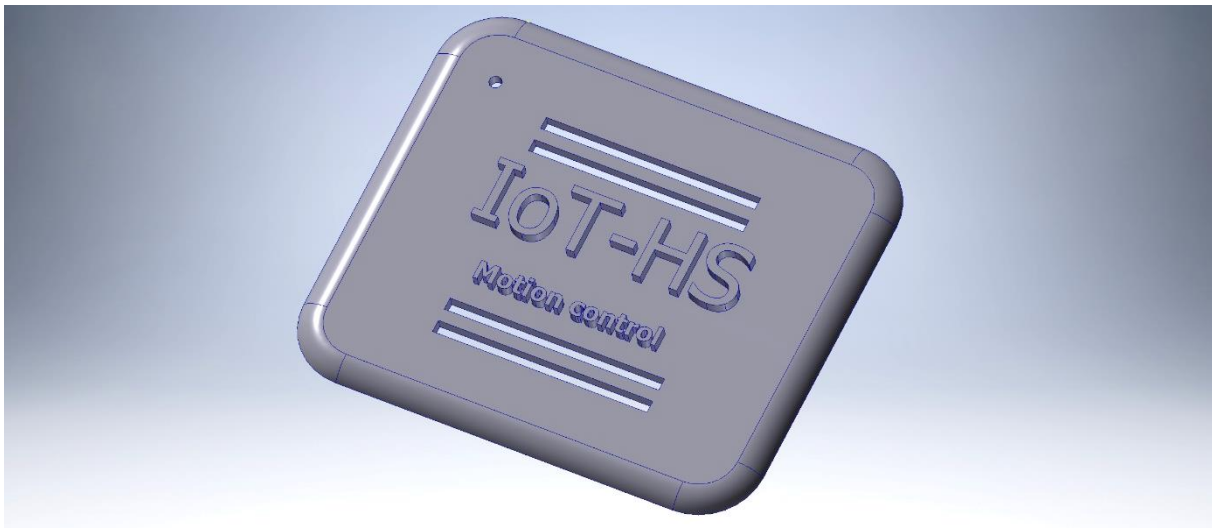
Model krabičky pro řídicí jednotku LED control

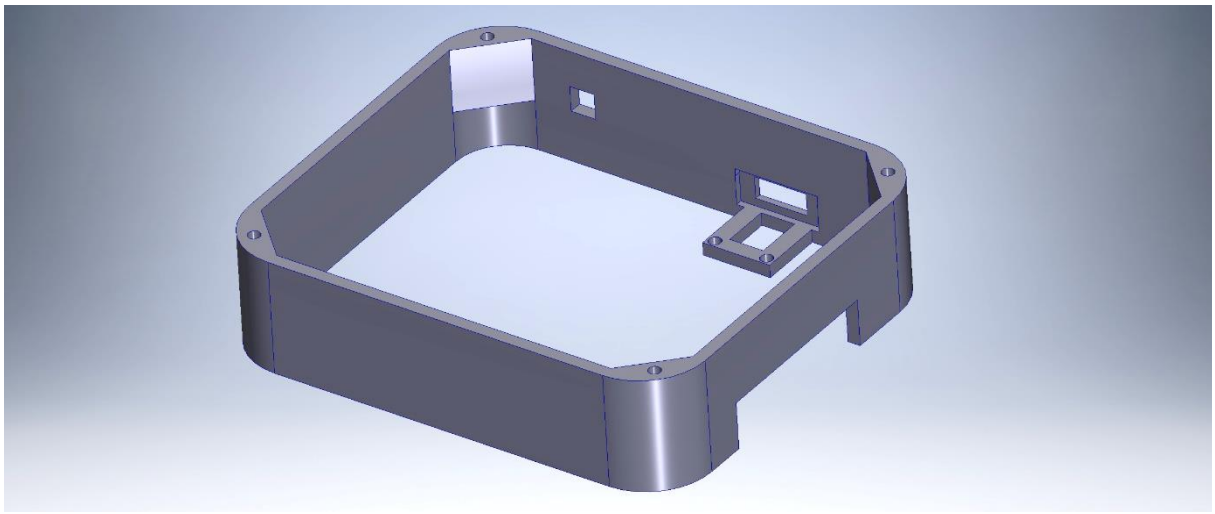


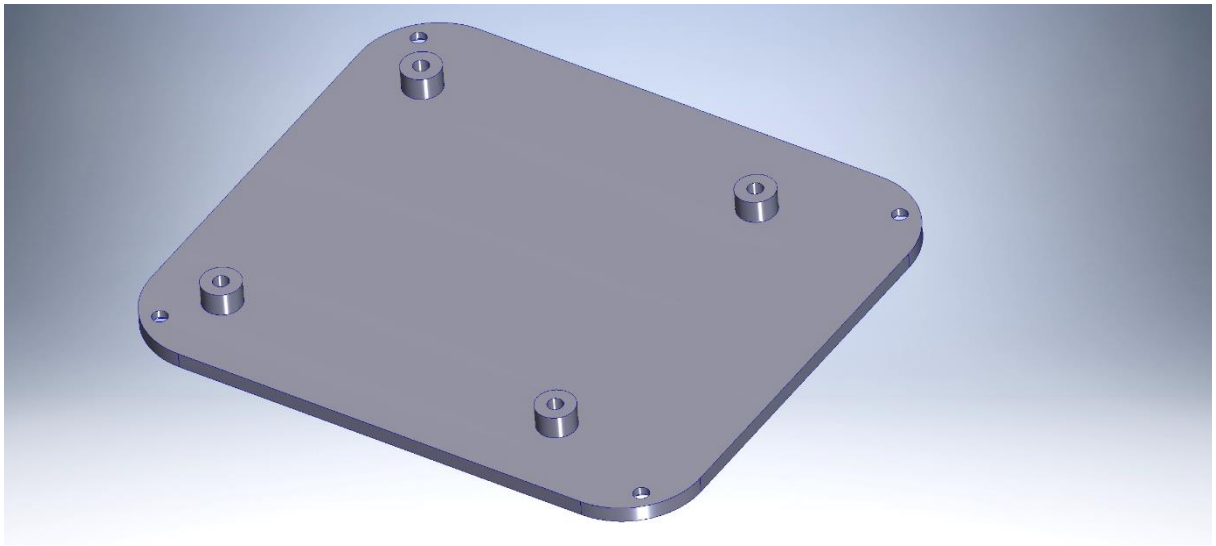




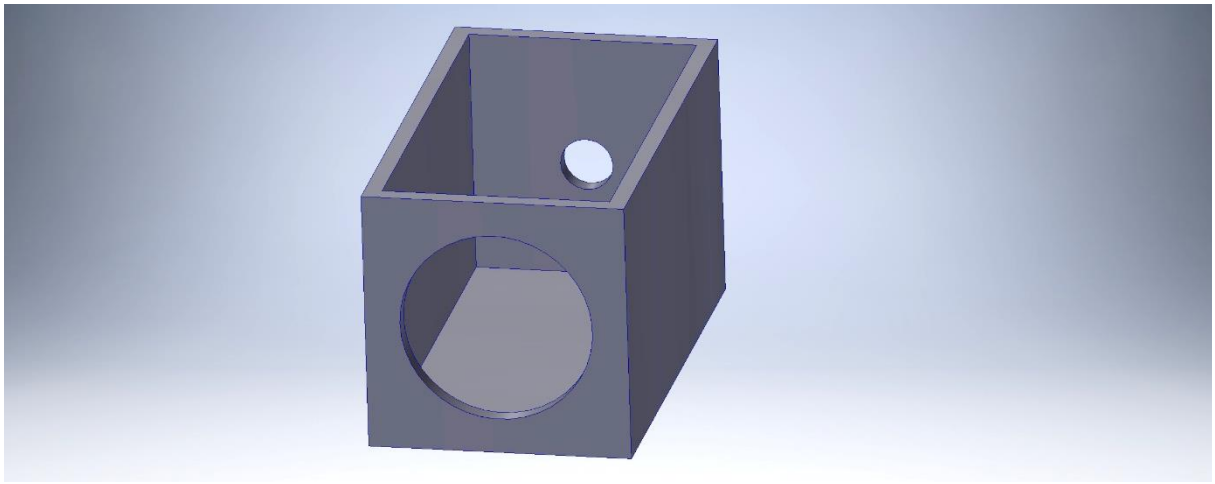
Model krabičky pro řídicí jednotku Motion control

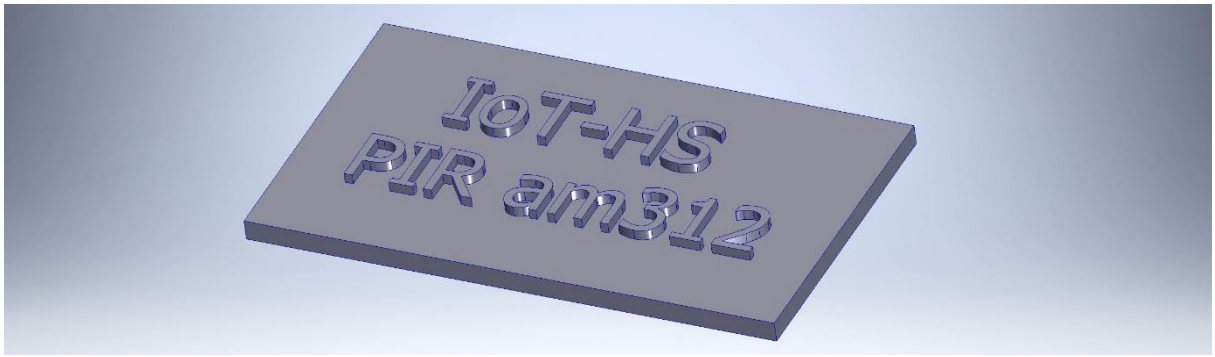




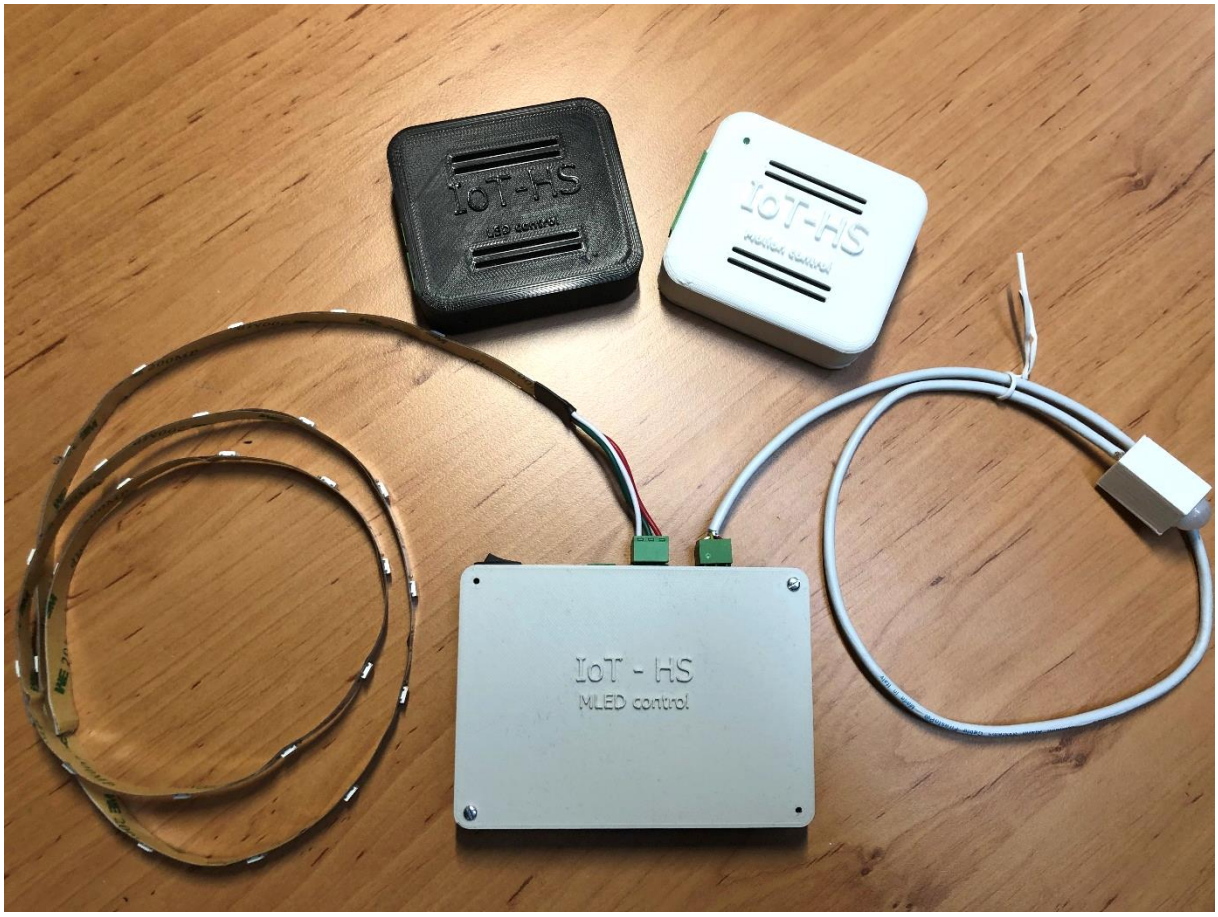


Model krabičky pro PIR senzor

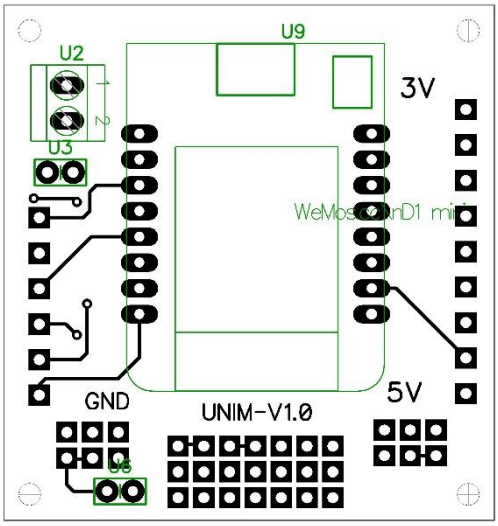




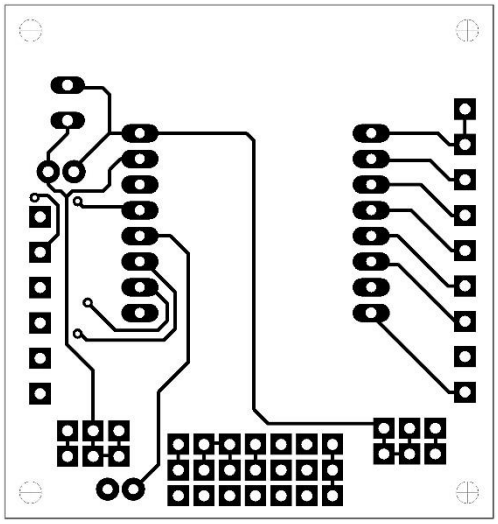
Vytisknuté krabičky pro řídicí jednotky



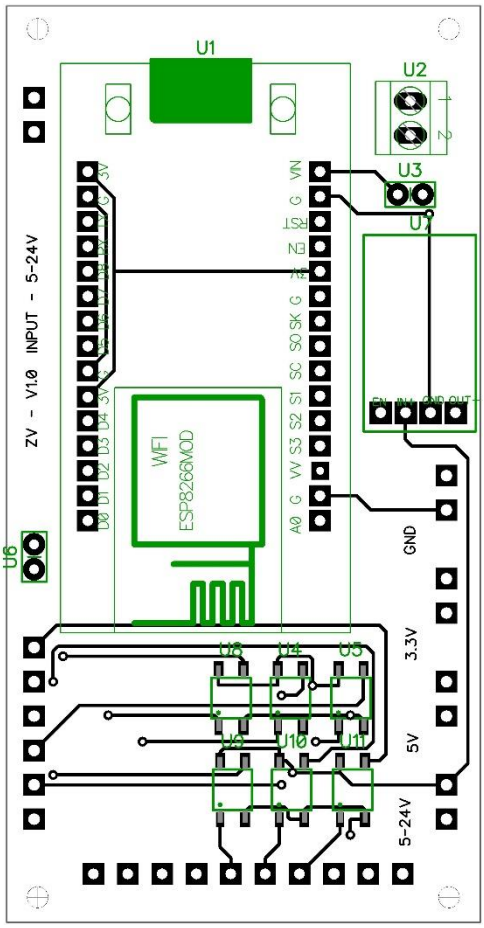
Návrh plošných spojů



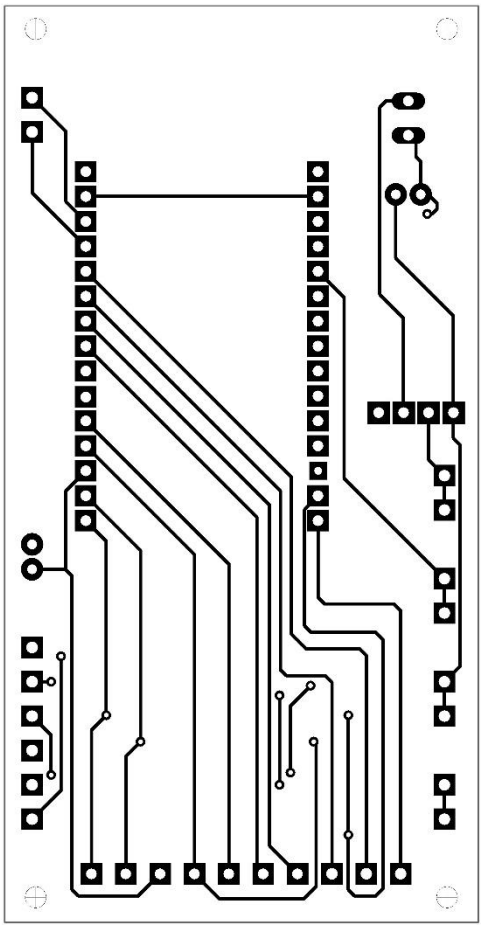
Přední část



Zadní část

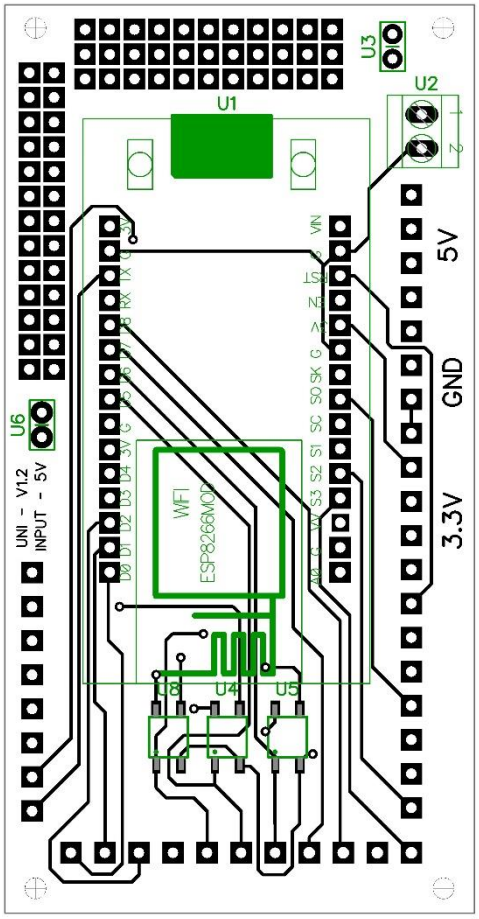


Přední část

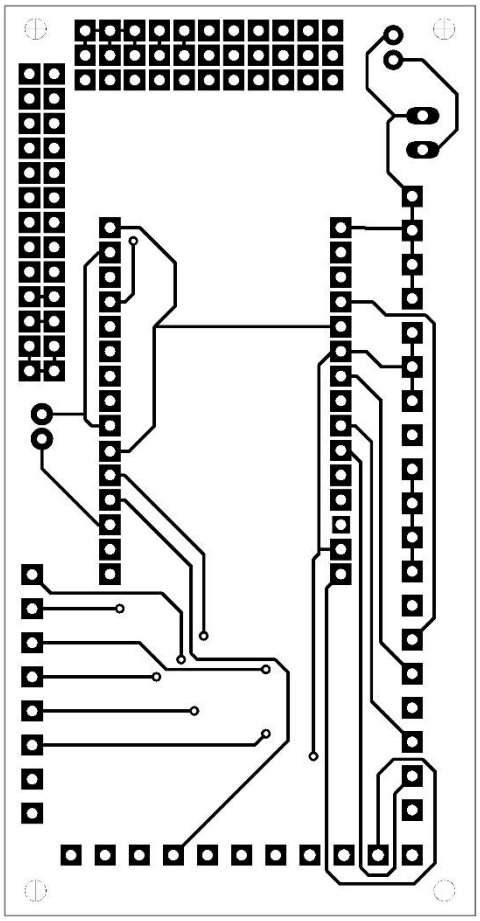


Zadní část

Návrh plošných spojů 2



Přední část



Zadní část

Vyrobené plošné spoje