VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Libor Frolich

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

VYUŽITÍ PLATFORMY ANDROID WEAR PRO HLASOVÉ OVLÁDÁNÍ ZAŘÍZENÍ V CHYTRÉ DOMÁCNOSTI

UTILIZATION OF ANDROID WEAR PLATFORM FOR VOICE-CONTROL SERVICES IN SMART HOME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Libor Frolich

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. Jiří Hošek, Ph.D.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Libor Frolich ***ID:*** 174201
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Využití platformy Android Wear pro hlasové ovládání zařízení v chytré domácnosti

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce bude seznámení se s možnostmi komunikace nositelné elektroniky v rámci chytré domácnosti. Teoretická část práce bude obsahovat popis platformy Android Wear a vybraných chytrých hodinek (Huawei Watch Sport 2). Dále bude také obsahovat popis Google Assistant a s ním spojeného Google SDK. V rámci praktické části bude proveden vývoj aplikace pro ovládání chytrých zařízení (chytrá žárovka, zásuvky, apod.) v domácnosti pomocí hlasového ovládání.

DOPORUČENÁ LITERATURA:

[1] RUIZ, David Cuartielles. 2014. Professional android wearables. Indianapolis, IN: John Wiley and Sons. ISBN 11-189-8685-7.

[2] YE, Roger. 2016. Embedded programming with Android: bringing up an Android system from scratch. New York: Addison Wesley. ISBN 01-340-3000-1.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 29.5.2018

Vedoucí práce: doc. Ing. Jiří Hošek, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Bakalářská práce se zabývá problematikou spojení chytrých domácností a hlasem ovládaných asistentů. Jsou zde popsány použité zařízení: hodinky Huawei Watch 2, počítač Raspberry Pi 3B+, chytrá žárovka TP Link LB130 a bezdrátové zásuvky Intertechno ITR-1500. Propojení bylo dosaženo díky vývojové platformě Actions SDK, spojené s chytrým asistentem vyvíjeným společností Google. Realizace problému probíhala zapojením simulované domácnosti a návrhu serverové aplikace. Server byl spuštěn na počítači Raspberry Pi pomocí JavaScriptové aplikace Node.js. V závěru práce byl řešen problém se zpracování hlasových příkazů a aktualizací operačního systému chytrých hodinek.

KLÍČOVÁ SLOVA

Actions on Google, Google Assistant, hlasové příkazy, Huawei Watch 2, chytrá domácnost, internet věcí, Node.js, Raspberry Pi

ABSTRACT

The bachelor thesis deals with the connection of smart homes and voice-controlled assistants. There are described used devices: Huawei Watch 2, Raspberry Pi 3B+, TP Link LB130 smart lightbulb and Intertechno ITR-1500 wireless sockets. Linking has been achieved through the Actions SDK, combined with a smart Google assistant. The problem was implemented by connecting a simulated home and designing a server application. The server was running on the Raspberry Pi computer using the Node.js. At the end of the thesis was solved the problem of processing voice commands and updating of the operating system of smart watches.

KEYWORDS

Actions on Google, Google Assistant, Huawei Watch 2, internet of things, Node.js, smart home, Raspberry Pi, voice commands

FROLICH, Libor. *Využití platformy Android Wear pro hlasové ovládání zařízení v chytré domácnosti*. Brno, 2018, 56 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Jiří Hošek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Využití platformy Android Wear pro hlasové ovládání zařízení v chytré domácnosti“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Jířímu Hoškovi, Ph.D. a konzultantovi panu Ing. Kryštofovi Zemanovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno . . . . . .

. . . . . .

podpis autora

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Během několika předchozích let se čím dál tím více firem a následně i zákazníků zajímá o problematiku chytrých domácností [1]. Automatizace zařízeních v chytrých domácnostech je v dnešní době úsporných řešení velice vítaná a s těmito možnostmi se již počítá, jak při zřizování nových staveb, tak pro modernizaci již postavených objektů. Některé domácnosti, takzvané „pasivní domy“, využívají inteligentních prvků společně s vlastním zdrojem energie. Pasivní dům je takový, který na své vytápění ideálně nespotřebovává žádnou energii dodanou městskou sítí. Ve skutečnosti se jedná o malé množství spotřebované energie (do $15\,\mathrm{kWh}/m^2$ ročně) [3]. Hlavní částí projektu pasivního domu je zajisté zateplení, ale dalším důležitým prvkem je samotné ovládání vytápění. S použitím zařízení chytré domácnosti, která využívá automatizace zapojených prvků Internetu věcí, je možné zefektivnit průběh vytápění objektu.

Prvky chytrých domácností však nyní již zasahují do více a více oblastí, například automatické ovládání světel, spínání spotřebičů bezdrátově řízenými zásuvkami, ovládání žaluzií, ale i kontrola zabezpečení domu IP kamerami. Dříve byla komunikace s domácností zajištěna přes webové rozhraní nebo přes aplikaci v počítači či v mobilním telefonu. Nyní vše směřuje k hlasovému ovládání [2]. K tomu slouží chytří asistenti, kteří jsou integrovaní téměř v každém zařízení. Například asistent společnosti Google se nachází v hodinkách, mobilních telefonech, chytrých reproduktorech, počítačích, televizích, chytrých displejích, ale i v osobních automobilech. Mezi tři nejčastěji v dnešní době používané asistenty spadají: výše zmíněný *Google Assistant* [4], od firmy Apple - *Siri* [5] a ze společnosti Amazon - *Alexa* [6].

Náplní této práce byla realizace hlasového ovládání simulované chytré domácnosti s použitím chytrých hodinek s operačním systémem Android. S přiřazenými hodinkami bylo zapotřebí směrovat styl práce na služby společnosti Google. Toho bylo dosaženo použitím vývojového balíčku Actions on Google. Tato práce reaguje na směr a technologickou poptávku dnešní doby.

1 CHYTRÁ DOMÁCNOST A RASPBERRY PI

Princip chytré domácnosti vychází z technologie Internetu věcí, která představuje koncept vzájemně propojených zařízení schopných získávat, odesílat a nebo přijímat data v rámci jedné sítě. V ní jsou zařízení velmi často připojena bezdrátově, ale mohou být fyzicky pomocí kabelu. Velice všeobecně by se dala tato zařízení rozdělit na senzory, akční prvky, řídící jednotky a vstupně výstupní zařízení použitá pro komunikaci s uživatelem. Principem sítě je zajistit automatizovaný dohled/řízení nad určitým prostředím bez přímé účasti člověka. Převážná část komunikace tedy probíhá na úrovni komunikace mezi stroji (*Machine to Machine*), uživatel je pak informován o stavu sledovaného prostředí.

V případě této práce byl pro systém domácnosti vybrán mikropočítač Raspberry Pi 3. K němu byl připojen senzor vzdálenosti pomocí 4 GPIO (*General-purpose input/output*) pinů (dvou programovatelných, napájení a společná zem). Raspberry bylo do lokální sítě připojeno za pomocí ethernetového kabelu, kvůli kvalitnějšímu připojení a eliminaci rušení, protože WiFi směrovač se nacházel v podkroví přibližně 10 m od počítače. Na tomto směrovači byla pro Raspberry přiřazena statická IP adresa, kvůli potřebnému přesměrování portů. To bylo nastaveno pro vzdálený přístup pomocí VNC a pro vystavení lokálního webového serveru spuštěném na Raspberry. Server se spouští na portu 55080, pro přístup na VNC byl přidělen port 5900. V lokální síti se dále nacházely dvě bezdrátové zásuvky a jedna žárovka.

1.1 Chytrá domácnost

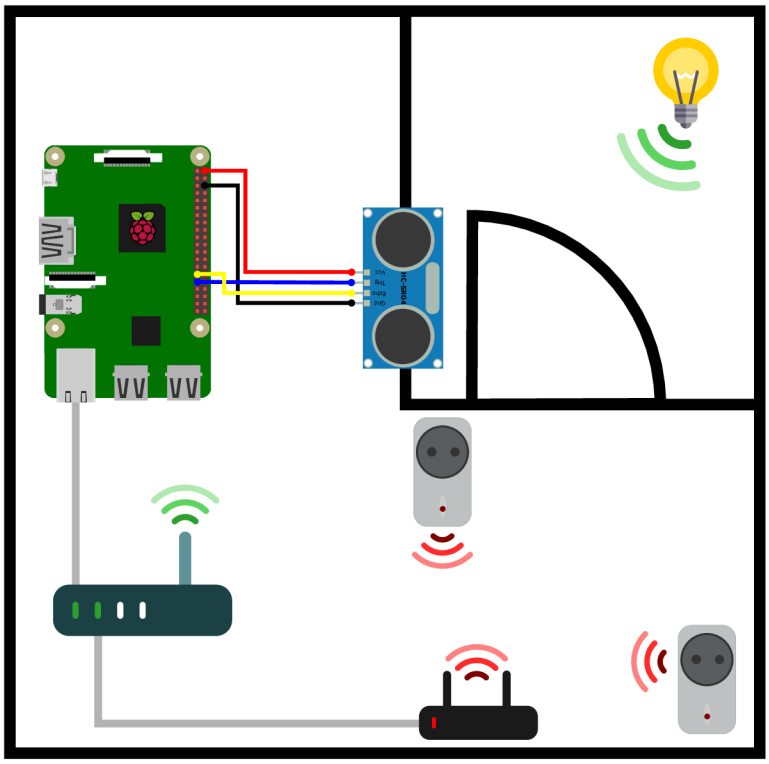
Pod pojmem „chytrá domácnost“ se dá představit libovolný byt či dům, který je určitým způsobem automatizován. V minulosti šlo především o automatizování každodenních činností, které měli opakující se charakter (například vytáhnutí žaluzií nebo vytápění místností podle času). V dnešní době se na chytrou domácnost klade větší důraz a stále přibývají nové prvky a zařízení, která lze do domácnosti integrovat. Domácnosti často obsahují hned několik chytrých zařízení, která jsou ovládána uživatelem (pomocí aplikace nebo hlasu) a nebo přímo chytrým asistentem. Tato zařízení jsou propojena v rámci společné sítě (často řešeno pomocí WiFi sítě), kde spolu komunikují a vzájemně se ovlivňují v závislosti na naprogramované činnosti nebo na podnět uživatele. Základem takovéto domácnosti bývá určitý centrální prvek, který slouží jako hlavní bod pro přístup a komunikaci mezi ostatními zařízeními. Jedním z momentálně čím dál tím více se rozšiřujícím zařízením je chytrý reproduktor, který je zapojen do domácí sítě a zařízení domácnosti jsou k němu přiřazena uživatelem z příslušné doprovodné aplikace.

Díky rozšiřující se nabídce zařízení a rostoucímu počtu společností, které se problematikou zabývají, postupně klesá pořizovací cena zařízení. Často jsou tak nově vznikající stavby domů již plánovány jako chytré domácnosti. I tak jsou ale ceny propastné, například žárovka TP Link LB130 použitá v této práci, se prodává zhruba za 1 500 Kč. Pokud se omezí funkčnost na použití pouze jako světlo, bez možnosti měnit barvy, vyjde normální LED žárovka s ekvivalentem 90 W výkonu na 50 Kč. S podobnými cenami se setkáme i u ostatních chytrých produktů.

Výhodou prvků chytré domácnosti je možnost fungování samostatně, bez zásahu uživatele, kterého na případnou změnu v systému může následně upozornit upomínka v aplikaci na mobilním telefonu. Tyto aplikace jsou často realizovány s možností vytvořit si virtuální dům s pokoji a do nich postupně přidávat zařízení. Takto je možné dále plánovat s časovými intervaly a ovládat větší skupiny různých zařízení pomocí jediné akce.

1.1.1 Simulovaná domácnost

Pro simulaci chytré domácnosti bylo zvoleno prostředí dvou místností propojených společnými dveřmi. Jednoduchý půdorys objektu spolu se zapojením hlavních prvků domácnosti je vidět na obrázku 1.1. V rámci tohoto prostoru byla použita tato zařízení: (i) ultrazvukový senzor vzdálenosti pro kontrolu stavu dveří, (ii) dvě bezdrátově řízené zásuvky, do kterých jsou zapojeny světla (obě se nachází v první místnosti), (iii) chytrá RGB žárovka umístěná v druhé místnosti ve stropní světelné objímce a (iv) počítač Raspberry Pi 3B+, který se kvůli nutnosti drátového zapojení nachází v těsné blízkosti senzoru vzdálenosti. Není potřeba, aby k Raspberry měl uživatel přístup a tak může být ukryté. Pro zajištění celkové funkčnosti těchto prvků, bylo zapotřebí zajistit jejich vzájemné propojení. Raspberry Pi je spojen ethernetovým kabelem s WiFi směrovačem a síťovou bránou umožňující ovládání zásuvek. Brána se zásuvkami komunikuje ve vlastním bezdrátovém pásmu, chytrá žárovka pak využívá připojení WiFi na směrovači. Hlavním cílem bylo demonstrovat ovládání domácnosti z chytrého zařízení, především pomocí hodinek. Ty jsou do lokální sítě taktéž zapojeny pomocí WiFi, spolu se spárovaným mobilním telefonem. Vytvořená aplikace je spojena s chytrým asistentem, který je provázán s prostředím fulfillmentu (část aplikace serveru, která vykonává akce z hlasových příkazů) [7] umístěném na Rasberry, nacházející se na veřejné síti. Díky tomu je pak možné domácnost ovládat z externí sítě, například z práce.



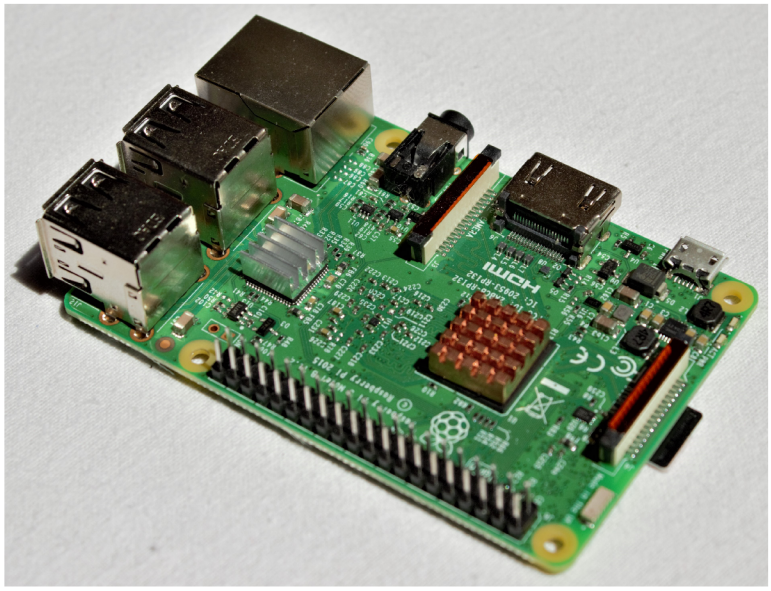
Obr. 1.1: Půdorys a schéma zapojení simulované domácnosti

1.2 Raspberry Pi 3 model B+

Raspberry Pi 3B+ je třetí generace jednočipového mikropočítače od britské firmy Raspberry Pi Foundation [8], která se zabývá distribucí cenově dostupných počítačů do světa, aby podpořili zájem lidí o nové technologie a programování. Věnují se vyučování a kurzům, jak správně a jednoduše Raspberry ovládat a používat. V současnosti již existuje nespočet projektů, které pochází jak z dílny Raspberry Foundation, tak přímo od zákazníků a uživatelů [9, 10].

Pro tuto bakalářskou práci byl vybrán mikropočítač Raspberry Pi 3 model B+, který je dostupný od začátku roku 2018 a jedná se o nejvýkonnější momentálně dostupnou verzi. O dostatečný výkon stroje se stará především nový 4-jádrový 64bitový procesor s frekvencí 1,4 GHz na jádro. Na procesor se dá dokoupit pasivní chladič, který zvyšuje účinnost odvodu tepla při dlouhodobém využití nebo při případném přetaktování. Podporu procesoru obstarává 1 GB paměti RAM. Ta je dostatečně velká pro zajištění běžného provozu. S operační pamětí je sdílená také grafická paměť, kterou má uživatel možnost přerozdělovat (v rozmezí od 16 do 896 MB). Maximální příkon je 12,5 W, jedná se tedy o ideální systém, který může být neustále spuštěný. Zařízení má k dispozici vlastní vestavěný modul Bluetooth 4.2, IEEE 802.11b/g/n/ac (pásma 2,4 a 5 GHz) a gigabitový Ethernet konektor (jde o přepo-

jení USB 2.0 konektoru, který omezuje maximální rychlost přenosu na 300 Mb/s) [11]. Výhodou této verze je již dostatečný počet vstupních a výstupních konektorů. Jsou zde například čtyři USB2.0 konektory, plnohodnotný HDMI konektor a rozšířené pole 40 GPIO pinů. V případě této práce byly využity 4 piny pro zapojení ultrazvukového senzoru vzdálenosti HC-SR04. Jeden pro společnou zem (pin 6) a jeho napájení 5V (pin 2). Další je vstupní pro buzení pulzu (pin 31) a poslední (pin 29) slouží k vyčítání příchozí odražené vlny.



Obr. 1.2: Mikropočítač Raspberry Pi 3 Model B+

1.3 Zařízení pro hlasovou interakci

Pro zajištění správné funkčnosti hlasové interakce s aplikací, byly pro práci použity 3 prvky: hodinky (jako hlavní ovládací prvek), mobilní telefon a simulátor (pro průběžné testování). Simulátor, umožňující i výpis chyb a problémů, je popsán níže v sekci 2.2.1. V době vývoje a prvotního zkoušení funkcí, byly hodinky vráceny škole pro použití na jiných projektech. Proto byl jako testovací prvek použit soukromý mobil Lenovo K3 Note s OS Android verze 6.0 (Marshmallow), který obsahuje Google asistenta.

Po otestování funkčnosti aplikace na mobilním telefonu a následném převzetí hodinek zpět, bylo nutné jim aktualizovat operační systém, aby mohli s aplikací správně komunikovat viz. 3.3.

1.3.1 Lenovo K3 Note

Dnes již starší mobilní telefon se systémem dualních SIM karet, vydaný začátkem roku 2015 [12]. V době vydání obsahoval verzi Androidu 5.0.2 (Lollipop), nyní je ale aktualizovaný na verzi 6.0 (Marshmallow). Další aktualizace pro tento telefon již nebyly vydány a ani nejsou plánovány. Mezi požadavky pro správnou funkcionalitu Google asistenta je zejména Android 5.0, tím pádem bylo možné využít tento mobil pro testování vyvíjené aplikace.

K3 Note má displej o uhlopříčce 5.5" v provedení LCD s fullHD rozlišením. Ochrana skla je dostatečně odolná proti pádům i poškrábání. Výkon stroje obstarává 8 jádrový procesor Cortex-A53 od firmy Mediatek s frekvencí 1,7 GHz na jádro. Oporou je operační paměť 2 GB a interní úložiště 16 GB. To je možné rozšířit za pomocí microSD (*Secure Digital*) karty s maximální kapacitou 32 GB (umístěna ve slotu místo micro SIM (*Subscriber Identity Module*) karty). V horní části mobilního telefonu jsou k dispozici dva konektory: audio 3,5 mm jack a micro USB (*Universal Serial Bus*) pro nabíjení a datový přenos. K základní konektivitě patří IEEE 802.11 b/g/n a Bluetooth 4.1. Kapacita baterie je 3000 mAh, která je dostatečně velká na to, aby při aktivním používáním spolu s hodinkami vydržela 2 až 3 dny. S nízkou aktivitou a se zapnutou WiFi a GPS se vydrž pohybuje okolo 7 až 10 dnů.

1.3.2 Huawei Watch 2 Sport

Hodinky Huawei pochází z druhé generace, která nabízí systém wearOS (dříve Android Wear 2.0). Hlavním prvkem hodinek je dotykový 1.2" AMOLED displej, který je při srovnání s konkurencí v cenové kategorii poměrně malý a způsobuje potíže při práci s textem, například při psaní zprávy na klávesnici. Dále jsou po obvodu displeje umístěna dvě tlačítka: jedno pro vyvolání hlavního menu a druhé programovatelné pro spuštění uživatelem zvolené aplikace. Díky robustní konstrukci těla hodinek z hliníku a vystouplých okrajů displeje, jsou hodinky dobře chráněny proti náhodnému poškození. Přispívá tomu i vodotěsnost s hodnocením IP68, dovolující ponor do hloubky 1,5 m po dobu 30 minut.

O výpočetní výkon se stará 4 jádrový procesor Snapdragon Wear od firmy Qualcomm, pracující na frekvenci 1,1 GHz. Velikost operační paměti je 768 MB a interní úložiště má 4 GB. Hodinky disponují vlastním mikrofonem, reproduktorem a vibračním motorkem. Konektivitu zajišťuje primárně WiFi ve standardu IEEE 802.11 b/g/n a Bluetooth 4.1. Sekundárně se dá použít NFC, pomocí nabíjecího portu připojit k počítači nebo využít slotu pro nano SIM kartu podporující síť LTE.

Pro sledování uživatelovy aktivity je v hodinkách senzor srdečního tepu, spolu s akcelerometrem a gyroskopem pro měření počtu kroků. Pro orientaci a mapování

pohybu je dostupný kompas a integrovaná GPS. Té je možné zvýšit přesnost, pokud uživatel má s sebou jak hodinky, tak mobilní telefon a má na obou zařízeních GPS aktivní. S novější aktualizací systému wearOS je nyní možné na hodinkách snímat spánkovou aktivitu. Dle aktivity využití klesá i celková výdrž baterie. Dalo by se říct, že pokud uživatel využívá hodinky se zvýšenou denní aktivitou, s měřením sportovní činnosti, je nutné hodinky nabíjet každou noc (výdrž asi 18 h). Pokud je aktivita nízká a využití je spíše pro určení času, kontrolu kalendáře, vyřizování zpráv nebo emailů, mohou hodinky vydržet zhruba týden.

Obr. 1.3: Hodinky Huawei Watch 2 Sport

1.3.3 Systém chytrých zásuvek

Zásuvky Intertechno není možné zapojit do elektrické sítě a ovládat na přímo povely z aplikace, protože jejich přenos informací probíhá ve frekvenčním pásmu 433 MHz. Proto byla zapotřebí do lokální počítačové sítě zapojit přístupová brána Mediola, která přijímá příkazy formou http požadavků a po zpracování odesílá informace do zvolené zásuvky.

Mediola AiO IP-Gateway V3

Třetí verze internetové brány Mediola je již poměrně starší model, který má nyní minimální podporu. Brána je umístěna na stejné lokální síti společně s ostatními prvky z projektu. Za pomocí konfigurace směrovače jí byla nastavena statická IP adresa a za pomocí programu Gateway ConfigTool je této bráně napevno přiřazena dříve nakonfigurovaná IP adresa. Kvůli možnosti připojení různých zařízení k této bráně je jejich ovládání podrobněji popsáno v stažitelné dokumentaci, která je dostupná pouze v německém jazyce. S touto variantou brány a zásuvek není možné

zjistit dotazem jejich momentální stav. Jediné možné nastavení je zapnout nebo vypnout. Tyto příkazy jsou vysílány pomocí GET požadavku s určenými parametry v URL odkazu. Díky tomu lze zásuvky jednoduše ovládat z libovolného webového prohlížeče, s podmínkou připojení ke stejné síti.

Zásuvky Intertechno ITR-1500

K dispozici jsou dvě chytré zásuvky ITR-1500 od rakouské společnosti Intertechno. Komunikace je zajištěna za pomocí přístupové brány připojené do lokální sítě ethernetovým kabelem. Uživatel pak komunikuje se zásuvkami skrze tuto bránu. Ta používá své vlastní bezdrátové pásmo na frekvenci 433,92 MHz.

Obr. 1.4: Zásuvka Intertechno ITR-1500

Zásuvky jsou schopné spínat zařízení do výkonu 1500 W a pracují v dosahu asi 30 m od řídící brány. Při pokusu o zapojení do domácí sítě se vyskytl problém se zapojením ovládané zásuvky, protože v chytré zásuvce není zdířka pro ochranný zemnící kolík a tak ji do běžné zásuvky nelze zasunout. Tento nedostatek byl vyřešen za pomocí adaptéru na italskou síť. Vzhledem k tomuto řešení, by nebyl připojený spotřebič v síti jištěný zemnícím kabelem a proto jsou v zásuvkách zapojeny pouze nízko-výkonové spotřebiče.

1.3.4 Žárovka TP Link LB130

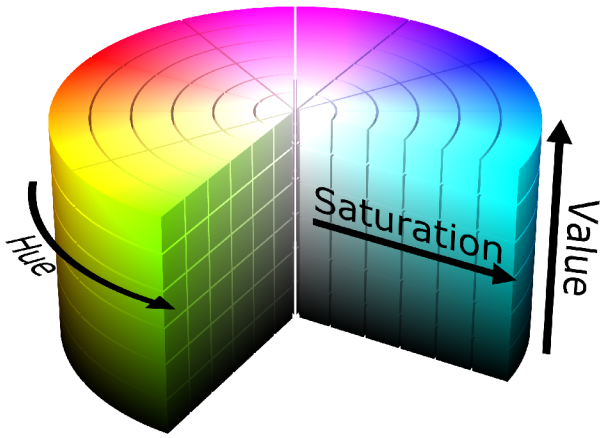
V průběhu práce byla tato žárovka umístěna ve stropním světle v pokoji. Dle výrobce je uvedeno, že zastává stejnou svítivost jako normální „neúsporná“ 60 W žárovka. Díky provedení RGB je možné za pomocí stažitelné aplikace Kasa nastavit

několik předdefinovaných barev, a nebo si vybrat vlastní z dostupné palety. Základním a nejpodstatnějším nastavením této aplikace je pokojová barva světla, kterou lze nastavit v rozsahu 2500 K až 9000 K. Kromě nastavení odstínu barvy a teploty, lze měnit i celkovou svítivost žárovky. V aplikaci Kasa pak lze sledovat momentální a odhadovanou spotřebu žárovky v průběhu roku.

Obr. 1.5: RGB žárovka TP Link LB130

Pro zpracování vlastních příkazů reagujících na asistenta byl použit balíček „tplink-lightbulb“ [13] pro Node.js. Tento balíček umožňuje přímou interakci se žárovkou za pomocí její IP adresy v síti a nahrazuje tak nutnost využívání aplikace Kasa. V práci byly využity dvě funkce: první pro změnu parametrů žárovky a druhá pro vyčítání jejího momentálního stavu.

Pomocí první funkce je možné ovlivňovat až 7 parametrů: stav žárovky (zapnutá/vypnutá), časová prodleva zapnutí (v milisekundách), mód žárovky (zjištěn pouze mód „normal“), odstín (barvy se určují z kruhu odstínů v rozsahu 0 - 360°), sytost (jinými slovy intenzita barvy, nabývá hodnot 0 - 100%), teplotu barvy (2500 K - 9000 K) a jas (množství bílého světla, v rozsahu 0 - 100%). V aktuálním stavu práce parametry módu a teplota barvy nejsou používány a pro všechny použité volby byla časová prodleva zapnutí konstantně nastavena na 10 ms. Z těchto parametrů plyne, že nastavení barvy odpovídá systému HSV/B (*Hue Saturation Value/Brightness* v překladu *Odstín Sytost Hodnota/Jas*). Pro lepší představu nastavení barev systému HSV, je přiložen následující obrázek 1.6.



Obr. 1.6: Zobrazení barevného modelu HSV [14]

Veškerá komunikace se žárovkou probíhá za pomocí asynchronního „slibu“ (Promise). Ten je vyslán na určenou IP adresu žárovky ve formě JSON syntaxe s vyplněnými parametry. Pokud je poslán požadavek z funkce na zjištění stavu žárovky, je očekávána zpětná odpověď, opět formou JSON. Pro změnu stavu žárovky je za formu odpovědi považována vykonaná změna, ze slibu se žádná odpověď neočekává.

1.3.5 Senzory přítomnosti osob v objektu

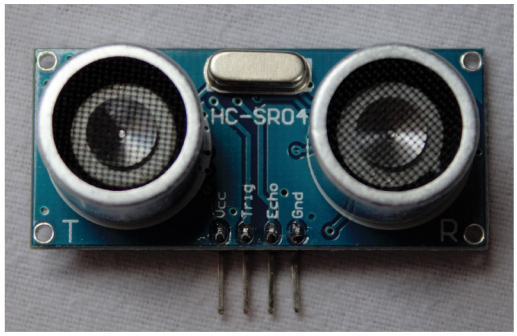
V simulovaném prostoru měli být původně k dispozici dva detektory pohybu. Detektor využívající dopplerův jev (HB100), který by byl umístěn v jedné z místností a monitoroval tak aktuální pohyb v pokoji. Za dne by fungoval jako kontrola úspory energie, kdy by po určité prodlevě bez zaznamenaného pohybu automaticky zhasínal světla v domě. A v noci jako prvek zabezpečovacího systému proti vniknutí neoprávněných osob.

Ultrazvukový detektor vzdálenosti měl být umístěn bezprostředně za vchodovými dveřmi, kde by reagoval na jejich případné otevření. Jakmile by se vzdálenost snížila pod nastavenou hodnotu, signalizovalo by se otevření dveří. V rámci předchozího stavu by se dalo vyhodnotit, zda-li uživatel do objektu vstoupil nebo jej opustil. Pokud by z objektu odcházel, došlo by k dvojitému ověření nepřítomnosti osob v objektu s dopplerovým radarem a po prodlevě bez pohybu by se veškerá propojená zařízení, které zapomněl uživatel aktivní, automaticky vypnula.

Bohužel v rámci zpracování příkazů pomocí Actions nebylo možné doručit zprávu/upozornění uživateli ze strany asistenta. Senzory tak nemohou otevřít vlastní kanál pro přímou komunikaci s asistentem. Řešení by mohla poskytnout mobilní aplikace s vlastním webovým serverem, která by běžela na pozadí a odchytávala POST

metody. Tyto zprávy by vysílaly aktivně spuštěné skripty senzorů, v případě splnění specifické podmínky (jako je otevření dveří nebo pohyb uživatele v místnosti). Vzhledem k tomu, že tento problém nebyl součástí zadání práce, nebyl do systému zapracován. S tímto omezením byl z práce dopplerův radar odebrán. Aby mohl pracovat efektivně, uživatel by se musel neustále periodicky dotazovat na jeho stav. Ultrazvukový senzor také ztratil hlavní význam, když nemůže aktivně sledovat stav dveří. Ale v práci byl ponechán pro účel jednorázového dotazu od uživatele na momentální stav dveří, kde by mohl být později použit pro sledování například garážových dveří, které by mohl majitel následně příkazem zavřít.

Detektor vzdálenosti HC-SR04



Obr. 1.7: Ultrazvukový detektor vzdálenosti HC-SR04

Ultrazvukový senzor je napájen 5 V a je připojený k GPIO pinům Raspberry. Pro porovnání rozdílů zisku dat z dostupných senzorů, je zde popsán i princip radaru HB100, který měří pohyb pomocí změny frekvence vyslaného signálu, proti signálu odraženého a zpětně přijatého. Z frekvenčního posunu známé hodnoty frekvence vyslaného signálu se následně zjišťuje rychlost pohybujícího se objektu. Ultrazvukový senzor HC-SR04 je používán primárně pro měření vzdálenosti snímaného předmětu. Pokud se objekt pohybuje, mění se čas mezi vyslanou a přijatou vlnou, ze které je následně vypočítána vzdálenost. Možná potíž by mohla nastat při pohybu po kružnici okolo detektoru, pak by jeho vzdálenost byla konstantní. Proto byl senzor umístěn bezprostředně za chodbovými dveřmi, kde snímal neměnnou vzdálenost od zdi ke zdi (s určitou tolerancí, v závislosti na teplotě, vlhkosti vzduchu a kvalitě odrazné plochy. Během práce došlo při měření k zjištění asi 10 cm nepřesnosti na vzdálenosti 250 cm). Pokud bude vzdálenost menší nebo větší, kvůli ztrátě odražené vlny, došlo k narušení prostoru mezi detektorem a zdí, tedy někdo otevřel dveře. Systém pracuje s lokální proměnnou, která si udržuje stav uživatele v bytě, následně při detekci pohybu dveří tento stav mění. Proměnná je částečně závislá na detektoru pohybu v obývací místnosti, kvůli dvojitému ověření, zda-li uživatel doopravdy opustil byt.

2 SLUŽBY GOOGLE A SERVER NODE.JS

Ze začátku byla práce řešena službou IFTTT (*IF This, Then That*), která zpracovávala uživatelovy příkazy z asistenta pomocí WebHooků zasílaných metodami GET a POST na server Raspberry. Služba IFTTT nespadá pod společnost Google, ale umožňuje propojení aplikací třetích stran s hlasovými příkazy pocházející z asistenta. Kvůli omezeným možnostem konfigurace, byla služba IFTTT nahrazena vývojovým softwarovým kitem Google Actions SDK, tak aby byla dosažena co nejlepší vzájemná kompatibilita použitých komponent. Ostatní prvky byly v práci ponechány. Jen hlavní aplikaci serveru Node.js bylo nutné celkově přepsat a rozšířit o několik nových funkcí a knihoven. Toto téma je nyní rozšířeno o zpracování primárně pomocí služeb společnosti Google. Jako hlavním hlasový či textový vstup byl použit Google Assistant v mobilních zařízeních se systém Android, ale především s použitím chytrých hodinek. Hlavní náplní této práce byla realizace aplikace, pomocí které je možné ovládat zařízení v chytré domácnosti. Jednalo se o (i) chytrou WiFi žárovku s možností měnit nastavení jasu a barvy a (ii) dvou chytrých zásuvek, které zvládnou příkon až 1,5 kW. Ty mají ale omezené možnosti ovládání, pouze na změnu stavu.

Následující sekce se zabývají všemi použitými funkcemi a nástroji společnosti Google, které bylo během práce potřeba použít a zprovoznit. Jedná se o asistenta, který slouží jako vstupně výstupní bod komunikace a systém funkcí Actions, který dovoluje definici vlastních akcí a propojení serverů Google se zařízením Raspberry.

2.1 Google Assistant

I v předešlé verzi práce byl použit vývojový balík Google Assistant SDK, ale byl na něj kladen větší důraz, protože se používal jako hlavní vstupně/výstupní prvek pro hlasovou komunikaci. Začátkem roku 2018 byla vydána významnější aktualizace, která zásadně ovlivnila základní nastavení knihoven asistenta. S ohledem na rozsah provedených změn, byl na Raspberry přeinstalován operační systém a aktualizované knihovny asistenta znovu staženy. Díky zachování dat projektu na účtu Google, bylo možné obnovit projekt v konzoli Google Cloud a znovu přiřadit Raspberry již vygenerované bezpečnostní a autorizační klíče použité v průběhu minulého semestru.

V nové aktualizaci přibylo několik nových podporovaných jazyků, ale čeština mezi nimi zatím není. Dle dostupných informací ze setkání zpravodajů serveru Technet.cz a vedoucího týmu vývoje konverzačních schopností Google asistenta, Behshada Behzadiho [15] vyplývá, že by měla být čeština do 5 let dostupná. Další neoficiální informace, ze zhruba druhé poloviny minulého roku [16], uvádějí že má asistent částečnou podporu českého a slovenského jazyka. Podmínkou je nastavení těchto jazyků

jako výchozí pro používané mobilní zařízení a použití chatovací aplikace Allo [17], která má integrovaného asistenta, samotný asistent v těchto jazycích nefunguje. Pokud se pak uživatel zeptá na nějakou informaci, asistent dokáže dotaz přeložit a zobrazit správnou odpověď, ale pouze v anglickém jazyce. Proto je třeba mít celý projekt včetně nastavení mobilního telefonu, hodinek a účtu společnosti Google nastavený ve shodném jazyce, například „English (US)“, pro zajištění co nejlepších výsledků při vyhledávání.

2.1.1 Konzole Google Cloud

Tato konzole sloužila k založení a k základnímu nastavení hlavního vývojového projektu. I když je primárně určená pro práci s cloudovými službami účtu, spravuje se v ní většina nastavení, jako je zapnutí Google Assistant API (*Application Programming Interface*) pro zařízení Raspberry Pi. Po zprovoznění API bylo možné vytvoiřit OAuth klíč, ten se používá pro ověření identity uživatele a zařízení zaregistrovaného v projektu. OAuth klíč je nezbytný pro zabezpečenou komunikaci napříč službami Google. Všechny tyto kroky jsou popsány v dokumentaci [1]. Principiálně ale zůstávají velmi podobné. Nejčastěji docházelo k přesunu dokumentací na server jiných služeb a společně s tím i umístění nastavení, kterých se tyto dokumentace týkaly.

2.1.2 Autorizační klíč OAuth2.0

V dalším kroku bylo třeba zařízení zaregistrovat pomocí autorizačního klíče. Tento klíč se generuje v konzoli Google Cloud [18]. Na úvodní stránce se zvolí aktuální projekt, v rozevíracím menu v levé části obrazovky se pak vybere položka *APIs & Services*. V levém sloupcovém menu jsou 3 záložky, je třeba zvolit třetí se symbolem klíče *Credentials*. Zde se zvolí tlačítko *Create credentials* a vybere se položka *OAuth client ID*. Jako typ aplikace se zvolí *Other* a objeví se pole pro vyplnění jména klíče. Může být libovolné, nebude jej třeba nikde používat. Následně se vygeneruje soubor obsahující specifický klíč, kliknutím na šipku pro stažení JSON souboru na konci řádku v přehledu, u právě vytvořeného klíče. Tento klíč slouží k zabezpečení přenosu dat mezi službami Google a autorizaci uživatele vztahující se k danému projektu a Raspberry. Soubor s klíčem je zapotřebí umístit libovolně na Raspberry Pi (například do domácího adresáře /home/pi) a dále použít cestu k tomuto souboru (obsahující i název souboru) pro zpětné ověření s pomocí jednoho ze dvou registračních programů. První *googlesamples-assistant-devicetool* [19] je umístěn přímo v knihovně s asistentem, druhý program *REST API* [20] je dostupný ke stažení.

[1] Setting up OAuth 2.0: `https://support.google.com/cloud/answer/6158849?hl=en`, cit. 21. 5. 2018

2.1.3 Google Assistant SDK

Posledním krokem pro zprovoznění asistenta na zařízení Raspberry, bylo stažení knihoven do virtuálního prostředí Python. Pro správnou funkčnost bylo zapotřebí všechny kroky provádět z domácího adresáře. Knihovny asistenta obsahují jak samotný kód pro spuštění, tak vzorky pro ověření správnosti nastavení a vyzkoušení funkcí. Po dokončení instalace bylo nutné vytvořit další autorizační klíč, nyní však pro ověření totožnosti ve vztahu k asistentovi. Vygenerování a ověření klíče se provedlo pomocí autorizačního nástroje google-oauthlib-tool [21]. V prvním kroku se klíč vygeneruje a stáhne na zařízení, poté je uživatel odkázán na URL odkaz, kde je nutné se přihlásit účtem Google používaný pro projekt a následně zkopírovat výsledný kód a vložit jej zpět do terminálu a potvrdit.

2.2 Actions on Google SDK

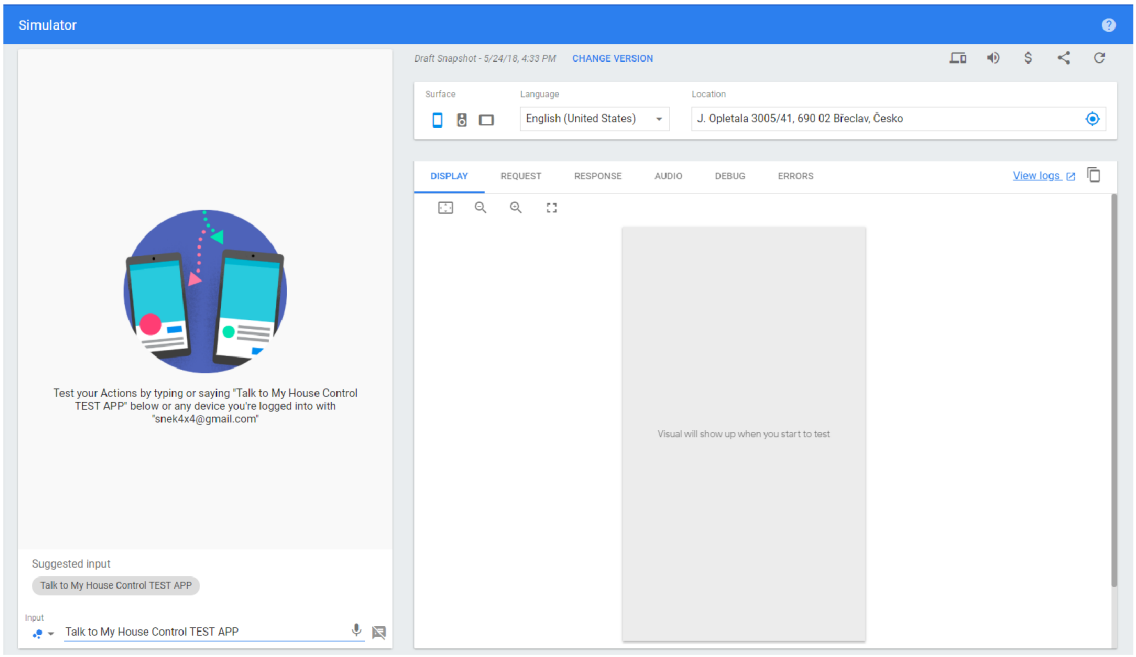
V nabídce vývojové konzole Google je pro vytvořený projekt možnost sestavit aplikaci pomocí čtyř možností: Converse.AI, Dialogflow, Actions SDK a Smart Home. Converse.AI a Dialogflow jsou prostředí spíše pro tvorbu aplikací soustředící se na tvorbu konverzačních aplikací, které například komunikují s externími aplikacemi pro zjištění informací, jako je aktuální počasí nebo dopravní situace v okolí uživatele. Další dvě možnosti jsou již zaměřeny i na kontrolu a ovládání prvků domácnosti. Volba Smart Home slouží pro komunikaci se zařízeními v chytré domácnosti a pro uživatele přináší jistou formu zjednodušení v použití vzoru aplikace, do které uživatel vkládá již známé prvky domácnosti. Tato varianta se z počátku projektu zdála být pro cíl práce ideální, protože dle dokumentace vychází přímo z Actions SDK. Jednalo se tedy o určitou formu zjednodušení Actions specificky navrženou pouze pro řízení chytré domácnosti. Po prostudování přiložené dokumentace se objevil problém s podporou knihovny pro server Node.js. Proto bylo vybráno řešení za pomocí Acitons SDK.

Actions SDK je ze 4 předchozích řešení nejuniverzálnější a mělo by tak pokrýt všechny potřeby projektu. Princip funkčnosti spočívá v sestavení vlastních objektů tzv. Intents, které definuje uživatel v souboru *actions.json*. Tyto objekty mají definované jméno, popis a spouštěcí příkazy, kterými by měl uživatel tuto akci vyvolat. Spolu s Intents je v souboru údaj o odkazu na fulfillment, tedy o prostředí v internetu kde se definované akce mají vykonat. Podmínkou souboru *action.json* je existující URL adresa se zabezpečením HTTPS. Tato podmínka se vztahuje na všechny komponenty projektu využívající služeb Google. Fulfillment byl zprovozněný formou webového serveru na lokálním adrese Raspberry a za pomocí programu ngrok je localhost veřejně vystaven do internetu. Toto vystavení je možné jak za pomocí HTTP

tak i právě HTTPS. Program ngrok v základní verzi má dvě hlavní omezení: adresy jsou náhodně generované a každá má dobu trvání 8 hodin. V placené verzi je možné si stanovit svou vlastní URL adresu, která je přiřazena zařízení na trvalo. Adresu získanou z programu ngrok je třeba po vypršení platnosti znovu vygenerovat a opět vložit do vytvořeného souboru actions.json, který je za pomocí programu *gactions CLI* [22] potřeba nahrát a aktualizovat do prostředí Google Assistant. Po jeho nahrání a přepnutí do fáze testování je možné aplikaci používat z libovolného zařízení, které má integrovaného asistenta.

2.2.1 Simulátor

Slouží k jednoduché simulaci příkazů pro asistenta. Simulátor je určen především pro interní testování nasazených aplikací a nelze na něm využívat služeb jiných funkcí nebo celkově možností asistenta. Aplikace nasazená za pomocí příkazu z gactions se přesune pro testování v simulátoru. Nejedná se tedy o plně nasazenou verzi. Příkaz ze simulátoru lze vkládat jak formou textu, tak hlasově.



Obr. 2.1: Prostředí Actions on Google Simulátoru

Plocha simulátoru je rozdělena na 3 prvky (Obr. 2.1): (i) obrazovka virtuálního zařízení s konverzací mezi uživatelem a asistentem, (ii) základní nastavení s možností vypnutí audio odpovědí a sdílení aplikace na všech zařízeních uživatele, či sdílení projektu mezi kolegy a testery, (iii) okno s podrobným výpisem přenesených dat ve formátu JSON, jak pro odchozí, tak i příchozí směr komunikace. V tomto okně se

lze přepínat mezi pěti záložkami: požadavkem pro asistenta, odpovědí přicházející ze serveru, možnou přidanou audio odpovědí, výpisem chyb a jejich debugem.

2.3 Node.js

Node.js je Javascriptové běhové prostředí, které dovoluje spouštění skriptů ze serverové strany. Je založeno na Chrome V8 enginu, který využívá událostmi řízený model zajišťující jednoduchý a efektivní prostředek pro vytváření serveru. Node.js využívá manažer balíčků npm, který dovoluje efektivně rozšířit schopnosti hlavního programu o funkce, které jsou momentálně potřeba.

Server není hardwarově příliš náročný, na oficiálních stránkách nejsou dostupné žádné minimální požadavky, ale v diskuzi na fórech lze zjistit, že k běhu stačí pouze jedno jádrový procesor a 128 MB paměti RAM [24]. S nízkou spotřebou Raspberry a nenáročným serverem bylo možné nechat zařízení běžet neustále. Jedinou překážkou byla nutnost restartovat ngrok každých 8 hodin. Pomocí balíčku express [23] bylo vytvořeno základní tělo serveru, které přijímá a reaguje na požadavky POST. Ty jsou dále směřovány do další funkce, která vychází z balíčku actions-on-google [25] a umožňuje směrování přijatých POST požadavků ze strany Google Assistant.

Pro zahájení komunikace s navrženou aplikací, je potřeba vyvolat asistenta pomocí vstupní fráze definované v nastavení projektu (jelikož se aplikace nachází ve stavu testování, je třeba ji vyvolávat definovanou frází. V případě úspěšného nasazení aplikace do provozu, je možné funkce aplikace volat z asistenta na přímo). Tato fráze je následně odchycena s pomocí „Main“ Intentu, který se nachází na serverové straně a uživateli je zpětně odeslána odpověď s předdefinovanou uvítací větou. Dále je již sestavena komunikace s aplikací. Nyní jsou veškeré uživatelem zadané příkazy přeposílány s pomocí asistenta přímo do této aplikace a následně zpracovány na straně Raspberry Pi pomocí textového Intentu.

2.3.1 Node.js Package Manager

Jedná se o největší systém balíčků pro platformu Node.js. Obsahuje více jak 650 tisíc volně stažitelných knihoven a řešení pro různé projekty. Manažer se jednoduše nainstaluje pomocí terminálu do počítače a následně se pomocí něj instalují balíčky, podle jedinečných jmen.

Na serveru Node.js na Raspberry Pi je použito několik balíčků stažených z npm:

express Pomáhá vytvářet jednoduché, ale účinné HTTP servery pro jednostránkové weby, programy či internetové aplikace. Pomocí expresu je možné spustit hlavní server pro aplikaci a nadefinovat jej na určitý port. V administraci

lokálního směrovače bylo nastaveno přesměrování veškerého provozu určený pro Raspberry na port 55080. [23]

actions-on-google Knihovna umožňující tvorbu aplikací využívajících Google Assistant, která podporuje jak Actions SDK, tak konverzační Dialogflow. [25]

body-parser Slouží k analýze vstupních dat, často ve formě textu. V aplikaci byl použit primárně pro funkci zpracování JSON požadavků z asistenta přicházejících na express server. Pomocí funkce *bodyParser.json(type: 'application/json')* zajišťuje analýzu dat, které nesou v hlavičce daný typ. [26]

xmlhttprequest Vytváří jednoduché HTTP metody, jako je GET, POST, PUT, DELETE a to jak v synchronním, tak asynchronním režimu. Balíček byl použit pro odesílání GET požadavků na bránu pro ovládání stavu zásuvek. [27]

tplink-lightbulb Knihovna obsahující funkce pro ovládání chytrých zařízení firmy TP Link. Jde především o různé typy bezdrátových zásuvek a žárovek. Před tímto balíkem byl použit podobný systém, který ovládal světlo pomocí synchronizace s cloudovým účtem aplikace Kasa od TP Linku. Při použití ale občas docházelo k nedokončení změny stavu, kvůli dlouhé odezvě. Tento balíček zajišťuje ovládání pomocí předem určené IP adrese žárovky. Která se nachází v lokální síti a odezva provedení požadavku je tak minimální. [13]

pigpio Umožňuje ovládání a vyčítání stavu GPIO pinů na Raspberry Pi, práci s hranami signálu, PWM modulaci a ovládání servomotorů. Knihovna podporuje všechna zařízení Raspberry: Zero, 1, 2 a 3 ve všech modelech. V práci je využita pro měření vzdálenosti pomocí ultrazvukového senzoru HC-SR04. [28]

2.4 Zpracování akcí

V závěrečném stavu práce se nepovedlo zprovoznit spouštění akcí hlavní metodou. Tou je rozlišení vstupních frází pomocí vlastních funkcí Intents, které jsou integrované ve službě Actions. Po několika pokusech (dle oficiálního návodu, později i zkopírováním ukázkových příkladů) se spuštění a odchycení Intents na straně serveru, stále nedařilo. Bylo tedy třeba vytvořit alternativní cestu a to pomocí hlavního textového vstupního řetězce předávaným v celé aplikaci. Tento řetězec se do serverové části posílá nehledě na to, zda-li funguje rozlišení Intents. Pomocí funkce *getRawInput();* byl uložen z příchozího požadavku do globální proměnné aplikace. Získaný text pak lze porovnávat pomocí podmínek, to vyřešilo problém s rozlišením akcí ovládajících prvky domácnosti.

2.4.1 Pomocí Intents

Správnou a očekávanou funkcí mělo být využití definovaných Intents v souboru *action.json*, které by měly určené spouštěcí fráze. Dle příkladů uvedených v dokumentaci [29], by měli stačit přibližně tři a mohou obsahovat i parametry upřesňující požadavky uživatele (například barva, lokace, teplota). Kvůli využití *Google Compute API* je možné odchytávat i obsahem podobné uživatelem řečené věty. Toto API porovnává definované fráze s příchozím příkazem a porovnává slova ve větě. Poté co by se fráze vyhodnotila, že připadá pod definovaný Intent, byla by asistentem vyslána na server fulfillmentu. Pomocí funkce na získání Intentu *getIntent()* by došlo k rozřazení zvolené akce aplikací a akce by byla provedena. Protože při pokusu o rozřazení akcí pomocí Intents došlo k neúspěchu, byl pro aplikaci použit postup v následující sekci 2.4.2.

2.4.2 Pomocí vstupního slova

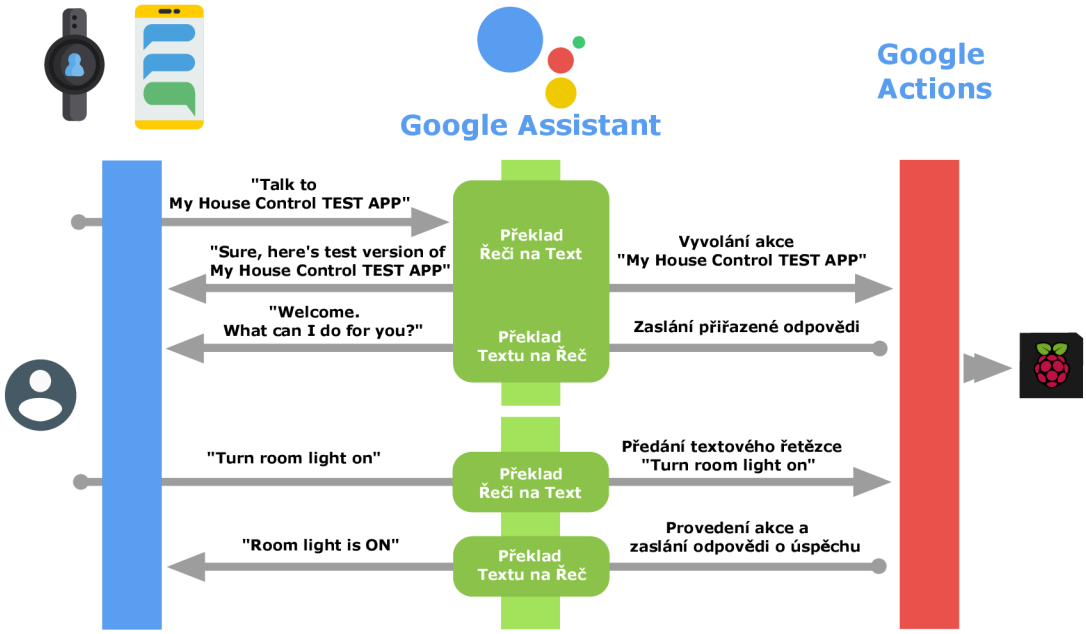
Aktuálně se na server ovšem dostávají pouze informace ze dvou předdefinovaných Intentů - *actions.intent.MAIN* a *actions.intent.TEXT*, které je zpracovány jako *Raw Input*. Tato příchozí vstupní data jsou převedena na malá písmena a porovnány pomocí podmínek ve funkci switch, které při jejich splnění vykonávají požadované akce. I když se systém zpracování dat liší, především stranou na které se data zpracovávají, samotný uživatel aplikace rozdíl ve funkčnosti nepostřehne. Nebo alespoň ne v aplikaci s takto malým rozsahem akcí. Nevýhodu oproti výpočetnímu středisku Google, zpracovávajícímu vstupní povely, představuje nutnost definice přesného znění vstupní fráze. Uživatelský vstup je totiž porovnáván operátorem (===), který hlídá přesnou posloupnost znaků v textovém řetězci a typ dat. Rozdíl v jednom slově či písmenu je tak považován za špatný příkaz a je vyvolána chybová hláška (zajištěna výchozí podmínkou funkce switch a nedojde tak k ukončení aplikace). Pro minimalizaci tohoto problému je u každé podmínky definováno několik obsahově podobných podmínek, kterými je možné určitou akci spustit a minimalizují se tak možnosti chyby při přeřeknutí.

3 KOMUNIKACE V PROJEKTU

V rámci vytvořeného projektu probíhá několik různých stylů komunikace. První (externí) styl závisí na nasazené aplikaci pro testování a jedná se o předávání informací mezi uživatelem, asistentem a aplikací (Actions a Node.js server). Asistent zde tvoří most pro zprostředkování dat mezi uživatelem a chytrou domácností. Druhý styl je z pohledu aplikace interní a propojuje použitá zařízení. Jde především o komunikaci s prvky domácnosti za pomocí GET nebo POST metod a hardwarový zisk hodnot z GPIO pinů.

3.1 Invokace aplikace

Základním prvkem pro komunikaci s aplikací je její navázání. Pro tento stav se počítá se všemi splněnými podmínkami: (i) na Raspberry běží serverová aplikace, (ii) adresa z ngroku je aktuální a je spuštěna na správném portu, (iii) pomocí nástroje gactions je posunut pro testování soubor *action.json* obsahující výše zmíněnou adresu generovanou ngrokem.



Obr. 3.1: Komunikace uživatele s asistentem [30, 31, 33, 32]

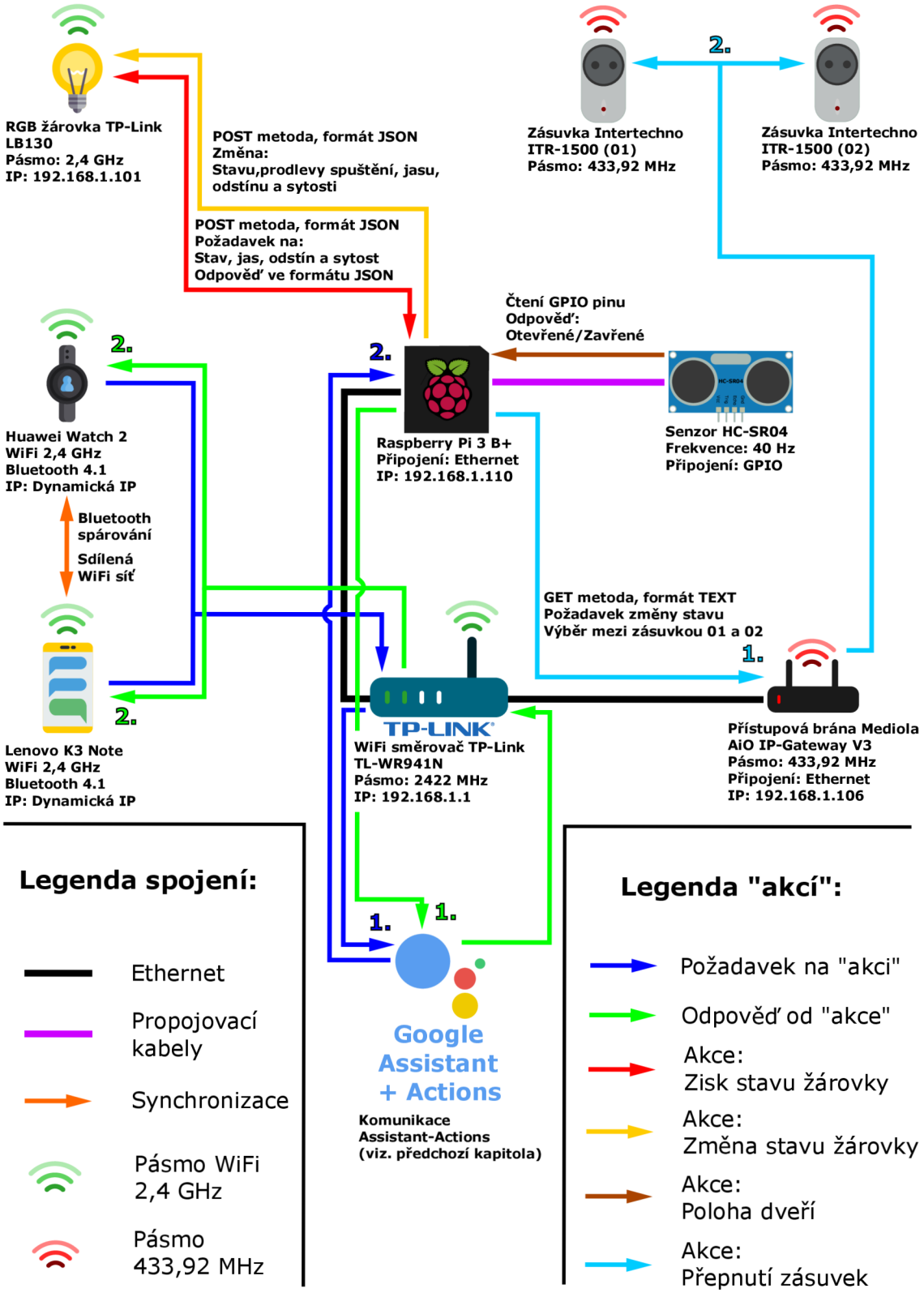
Komunikace se zahájí vyslovením spouštěcí fráze do Google asistenta na libovolném zařízení. V případě této práce se jedná o větu „Talk to my house control test app“. Poslední dvě slova je nutno uvádět, jelikož se aplikace nachází pouze v testovacím módu. Stylem jakým byla aplikace napsána, by nedokázala projít schvalovacím řízením (aplikace byla navržena na určené síti a s konkrétními zařízeními.

Pokud by byla možnost stažení této verze, uživatelé by byli schopni ovládat pouze zařízení v simulované domácnosti. Server ani nabídka zařízení nejsou dynamické). Vstupní fráze je po vyslovení převedena na text a je poslána do prostředí Actions, kde dojde k jejímu rozpoznání a spuštění aplikace. Tímto je komunikace s aplikací navázána. Dokud uživatel příkazem komunikaci neukončí, veškeré dotazy směřovány na asistenta jsou posílány do této spuštěné aplikace. Aplikaci je možné opustit slovem „cancel“, následující příkazy již budou zpracovány standardně asistentem. Pro návrat k ovládání domácnosti je zapotřebí aplikaci znovu vyvolat. V předchozím obrázku 3.1 je uvedena ukázka vyvolání aplikace a komunikace mezi uživatelem a asistentem pro rozsvícení pokojového světla.

3.2 Zpracování vstupních příkazů a odpovědí

Veškeré uživatelem zadané vstupní příkazy, které jsou přeložené z hlasu na text, jsou zasílány na server spuštěný na počítači Raspberry Pi. Veškerá data od asistenta jsou na prostředí serveru zasílany ve formě JSON zpráv. V nich jsou uvedeny informace o probíhající relaci: ID požadavku, čas doručení a uživatelské ID. Dále jsou přenášeny data o použitém Intentu a vstupu (zda-li se jednalo o hlas nebo text). Kvůli potřebě jednoznačně porovnat příkazy, je celý přijatý text zpracovaný pomocí funkce *getRawInput* z knihovny actions-on-google. Ta umožní z přijatého celku JSON vybrat právě hodnotu z parametru „query“, která obsahuje vloženou frázi.

Asistent ne vždy uživateli rozumí přesně. Může to být způsobenou rušným okolím, přízvukem, nedostatečnou vzdáleností od mikrofonu nebo přeřeknutím se. V případě, že v mluveném projevu dojde k delší odmlce mezi slovy, může asistent vyhodnotit začátek nové věty a slovu uprostřed věty přiřadit velké začáteční písmeno. Pokud by se do programu posílal takovýto text, mohlo by docházet k nevyhovění podmínky u zvolené akce. Proto bylo potřeba tuto situaci ošetřit a to tím, že přijímaný text je převeden na malá písmena, s pomocí funkce *toLowerCase*. Takto upravený řetězec je možné poslat dále pro zpracování aplikací.



Obr. 3.2: Komunikační schéma zapojených zařízení [30, 31, 34, 32, 35]

Obrázek 3.2 znázorňuje princip komunikačních cest v simulovaném zapojení. Šipky **tmavě modré** (vstupní příkazy) a **zelené** (odpovědi serveru) znázorňují všechny zprávy, které předávají data od uživatele po Raspberry Pi a nazpět. V podstatě se jedná o komunikaci z obrázku 3.1, kde zde jsou asistent s Actions znázorněny jako jeden celek. **Červená** šipka znázorňuje získání aktuálního stavu WiFi žárovky. Změna stavu, barvy nebo jasu žárovky je docílena pomocí **žluté** cesty. Z hlediska uzavřeného systému lokální sítě, je tato komunikace jednosměrná, protože po zaslání příkazu se od žárovky neočekává odpověď. Z pohledu uživatele jsou ale všechny akce považovány za obousměrné, protože po každé provedené funkci je uživateli zaslána odpověď. Na stejném principu funguje i ovládání zásuvek, zobrazeno **světle modrou** linkou. V prvním kroku je vyslána specifická URL adresa na IP adresu přístupové brány zásuvek. V ní je definováno o jakou zásuvku se jedná a zda-li se má zapnout či vypnout. Brána pak bezdrátově provede akci na zvolené zásuvce. **Hnědá** šipka je zisk stavu dveří získaného z ultrazvukového senzoru vzdálenosti. Ten je zapojen přes GPIO piny, ze kterých se tento stav vyčítá. Poslední spojení představuje základní komunikace mezi spárovanými hodinkami a mobilem, ta je vyjádřena **oranžově**.

3.3 Preview verze wearOS

V závěru práce došlo ke zjištění problému s použitím hodinek jako ovládacího prvku domácnosti. Mohla za to chybějící podpora služeb Actions v operačním systému wearOS na chytrých hodinkách Huawei Watch 2 Sport. Problém nastával při volání testovací aplikace pomocí asistenta, kdy po vložení spouštěcí fráze se nevyvolala aplikace, ale došlo k vyhledání této věty na internetu. Výsledkem pak byla stránka s 11 nejlepšími aplikacemi pro chytrou domácnost. Dne 9. května však vyšla aktualizace vývojové verze systému wearOS [36], která přinesla právě možnost komunikace s uživatelskými akcemi. V momentálním stavu není preview verze volně dostupná klientům pro standardní denní používání, ale pouze jako forma testovacího prostředí pro vývojáře. Proto bylo zapotřebí tento systém manuálně stáhnout a do hodinek nainstalovat z příkazové konzole v počítači.

3.3.1 Instalace vývojové verze

Postup instalace byl proveden podle návodu z uvedené citace [37]. Jelikož v průběhu instalace nového systému je zapotřebí kompletně vymazat stávající operační systém hodinek, je zapotřebí kroky z návodu provádět pečlivě a přesně. Správný výsledek instalace není možné 100% zajistit a je zde určitá pravděpodobnost, že instalace může zařízení odstavit, proto je během instalace uživatel upozorněn, že se tímto dopouští k porušení záruky výrobku.

V první fázi bylo zapotřebí zapnout v hodinkách vývojářské nastavení. K tomuto nastavení se uživatel dostane stisknutím horního tlačítka hodinek, které zobrazí *Hlavní menu*, v něm je třeba se navigovat skrze *Nastavení*, do položky *Systém* a následně do *O systému*. Nyní bylo zapotřebí 7krát zmáčknout položku „Číslo sestavení“. Následně se vrátit o jeden krok zpět, do nastavení systému, kde se nově pod položkou „O systému“ objevila nabídka „Vývojářské nastavení“. Tam byla zapnuta možnost ladění pomocí ADB.

Další krok byl proveden z počítače. Z odkazu byl stažen obraz aktuální preview verze systému. Zároveň bylo nainstalováno Android Studio (v té době, ve verzi 3.1.2), které bylo možné nainstalovat buď plnohodnotně (větší objem dat ke stažení, stahují se již předvybrané balíky) nebo v archivované formě, ve které bylo možné si po prvotním spuštění vybrat jaké balíky rozšíření se mají instalovat. Z instalovaných balíků jsou zásadní dva: Android SDK Tools a Android SDK Platform-Tools. V nástroji pro tyto aktualizace (SDK Manager) je k dispozici adresa, kde jsou tyto balíčky nainstalovány. Do této složky se extrahoval obraz preview verze z dříve staženého archivu. Důležité je, aby se ve stejném adresáři nacházeli (i) instalované soubory: *adb.exe* a *fastboot.exe* a (ii) extrahované soubory: *flash-all.bat* (nebo *flash-all.sh* v případě instalace z Linuxu nebo MacOS) a obraz systému *bootloader-sawfish-XYZ123.img*.

Pomocí dokovacího USB kabelu se hodinky připojily k počítači a v příkazovém řádku se přepnuli do předem zmíněného adresáře. Zde se postupně zadaly čtyři příkazy:

```
adb devices
```

sloužící k ověření, zda-li počítač má přístup k hodinkám.

```
adb reboot bootloader
```

je pro spuštění zaváděcího prostředí, ze kterého se provádí potvrzení instalace systému z hodinek.

```
fastboot oem unlock
```

odemyká možnosti zavádění instalace nové verze systému. Pokud se zařízení neodemkne, nemusí dojít ke správnému vymazání systému a nebo následného zápisu nového softwaru. Tento krok je nutné opět potvrdit z hodinek, pomocí dlouhého stisknutí startovacího tlačítka.

```
flash-all.bat
```

poslední příkaz nainstaloval systém ze staženého obrazu systému do hodinek. Instalace zabrala přibližně dvě minuty. Poté stačilo v bootovacím menu na hodinkách zvolit „reboot“ a počkat na spuštění systému. Během prvního spouštění se hodinky

asi 4krát restartovaly. Dále bylo třeba spárovat hodinky s mobilem. Nejdříve se v nastavení aplikace wearOS smazal záznam o spárování hodinek před flashem, protože došlo k vymazání všech dat, bylo třeba hodinky s mobilem párovat nanovo.

Po úspěšném navázání spojení s mobilem a aktualizacích systémových aplikací, byla možnost ověřit funkci Actions zavoláním testovací aplikace v asistentovi. Instalace byla úspěšná a aplikaci bylo možné vyvolat hned na poprvé. Nyní je možné domácnost ovládat za pomocí hodinek.

4 APLIKACE FULFILLMENTU

Jedná se o serverovou aplikaci, která slouží k vykonávání uživatelských příkazů. Hlavní podmínkou je spuštění serveru na veřejné síti. Aplikace se skládá ze dvou souborů, které se nacházejí v kořenovém adresáři na Raspberry Pi: hlavní soubor serveru *app.js* a soubor *action.json* s definovanými akcemi a odkazující na veřejnou adresu serveru. Serverový program je napsán pomocí JavaScriptu, soubor akcí JavaScriptovou strukturou JSON.

Vstupní příkazy odeslané z asistenta se dostanou na server a jsou následně zpracovány odchycením POST požadavků. Ty jsou předány funkci z knihovny Actions on Google a podle výpisu roztříděny na dva předdefinované Intenty: hlavní pro vyvolání aplikace a textový pro příjem textových řetězců získaných z hlasových příkazů uživatele. Tento text je následně použit pro porovnání s podmínkami v následující funkci switch, který tvoří hlavní část aplikace a řeší se pomocí ní všechny akce. Funkce switch byla zvolena na základě jednoduché možnosti definování více podmínek k jedné výsledné funkci. Ty jsou zapisovány pod sebou a u každé funkce jsou minimálně 2 (u některých až 20), aby se zamezilo chybovým stavům při přeřeknutí uživatele. V každé vykonávané akci musí být definována maximálně jedna možná odpověď asistentovi (maximálně jedna pro každou možnou variantu, pokud je v akci podmínka if/else musí být odpověď v každém z bloků). Odpověď uzavírá dotaz položený uživatelem, pokud není žádná odpověď odeslána, po uplynutí určité čekací době dojde k vyvolání chybového stavu „Aplikace neodpovídá“. Odpověď se skládá z proměnné *app*, která je definována pomocí knihovny *actions-on-google*, a z funkce *ask();*, ve které se nachází definovaná odpověď:

```
app.ask("Odpověď asistentovi");
```

Pokud by byly tyto odpovědi dvě za sebou, první by byla úspěšně odeslána a zobrazena uživateli, druhá by vyvolala chybu, jelikož by se aplikace snažila odpovídat na otázku, která již neexistuje nebo není žádná momentálně aktivně otevřená. Funkce switch končí položkou *default*, ve které se nachází odpověď asistentovi, pokud není splněna ani jedna ze vstupních podmínek.

Celková funkce by se dala popsat následujícím příkladem (*světlo je rozsvíceno červenou barvou*): Uživatel zavolá v asistentovi testovací aplikaci „Talk to my house control test app“, následně je sestaveno spojení s aplikací a zobrazena uvítací zpráva. Uživatel nyní chce získat barvu žárovky pomocí příkazu „Get color“, ten se přes asistenta dostane na vstup aplikace. Funkce switch jej přiřadí správné akci, která pošle na IP adresu žárovky požadavek na informace o jejím stavu. Z odpovědi se zjistí, že je světlo rozsvíceno a podle parametrů modelu HSB je odvozena červená barva. Následně je uživateli doručena odpověď „Color of room light is RED“.

5 ZÁVĚR

Práce na začátku uvádí do problematiky chytrých domácností, vycházejících ze zařízení Internetu věcí. Dále jsou popsány prvky simulované domácnosti pro tento konkrétní projekt. V první kapitole se nachází půdorys simulovaného objektu se zobrazenými zařízeními na jejich pozicích. Pro účel práce byl senzor vzdálenosti ponechán, i přesto že ztratil svůj hlavní potenciál. Zůstal na pozici za průchozími dveřmi, aby bylo možné se dotázat na jejich aktuální stav. V reálném prostředí by mohlo jít o zpětnou kontrolu například garážových dveří nebo střešních oken. Prvky v tomto půdorysu jsou vyobrazen pouze orientačně, pro představu zapojení v lokální síti. Podrobnější zapojení s rozepsanými funkcemi přenosu dat bylo popsáno dále u asistenta.

V teoretické části práce byly řešeny služby a použité komponenty Google, společně s aktualizací knihoven asistenta. Je zde také popsán webový server spuštěný na mikropočítači Raspberry Pi a na něm použité balíčky pro zajištění správnosti všech funkcí a zařízení. V průběhu práce došlo i ke dvěma větším problémům. První se týkal nemožnosti aktivování vlastních funkcí Intents v prostředí Actions on Google. Problém byl následně vyřešen tak, že se přebral na vstup aplikace hlavní vstupní textový řetězec. Následné rozlišení požadovaných akcí se řešilo až na straně serveru. Druhý problém nastal s podporou Actions SDK na zvolených chytrých hodinkách Huawei Watch 2. Pokud uživatel chtěl vyvolat pomocí vstupní fráze vyvíjenou aplikaci, hodinky tuto frázi nedokázali s pomocí asistenta vyhledat v definovaných akcích a místo toho ji jen vyhledali na internetu. Naštěstí v první polovině měsíce května, došlo k vydání vývojářské verze systému pro hodinky, která opravovala právě problém nepodpory Actions na hodinkách Huawei. Pro tento krok bylo zapotřebí originální systém hodinek odinstalovat a přehrát jej vydanou preview verzí. Přeinstalace naštěstí skončila úspěšně a po následném vyzkoušení vyvolání aplikace, byl problém vyřešen. Opravou tohoto problému bylo splněno zadání práce.

LITERATURA

[1] IoT Trends: Big interest in smart homes (but less in 'smart assistants'). In: *Netimperative: Digital intelligence for business* [online]. 1999, 9. 3. 2018 [cit. 2018-05-29]. Dostupné z: http://www.netimperative.com/2018/03/iot-trends-big-interest-in-smart-homes-but-less-in-smart-assistants/

[2] MCGRATH, Jenny. Say what you will, voice control is the future of home automation. In: *Digital Trends: Technology News and Product Reviews* [online]. Oregon (Lake Oswego), 2006, 3. 6. 2016 [cit. 2018-05-29]. Dostupné z: https://www.digitaltrends.com/home/voice-control-smart-home/

[3] CO JE PASIVNÍ DŮM?. *Centrum pasivního domu: passivnidomy.cz* [online]. Brno, 2005 [cit. 2018-05-29]. Dostupné z: http://www.pasivnidomy.cz/co-je-pasivni-dum/t2

[4] *Google Assistant: Just say "Hey Google"and Make Google Do It* [online]. California (Mountain View), 2016 [cit. 2018-05-29]. Dostupné z: https://assistant.google.com/

[5] *iOS - Siri - Apple* [online]. California (Cupertino), 2011 [cit. 2018-05-29]. Dostupné z: https://www.apple.com/ios/siri/

[6] *Amazon Alexa* [online]. Washington (Seattle), 2014 [cit. 2018-05-29]. Dostupné z: https://developer.amazon.com/

[7] Build Fulfillment. *Actions on Google* [online]. California (Mountain View), 2016, 17. 4. 2018 [cit. 2018-05-29]. Dostupné z: https://developers.google.com/actions/sdk/fulfillment

[8] *Rasberry Pi: Teach, Learn, and Make with Raspberry Pi* [online]. Cambridge, 2009 [cit. 2017-12-05]. Dostupné z: https://www.raspberrypi.org

[9] Forums. *Raspberry Pi: Teach, Learn, and Make with Raspberry Pi* [online]. Velká Británie (Cambridge), 2009 [cit. 2018-05-29]. Dostupné z: https://www.raspberrypi.org/forums/

[10] Community. *Raspberry Pi: Teach, Learn, and Make with Raspberry Pi* [online]. Velká Británie (Cambridge), 2009 [cit. 2018-05-29]. Dostupné z: https://www.raspberrypi.org/community/

[11] RASPBERRY PI 3 MODEL B+. *Raspberry Pi* [online]. Cambridge, 2018, 2018 [cit. 2018-05-22]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[12] Lenovo K3 Note Smartphone. *Lenovo: Official Bangladesh Site* [online]. Bangladesh, 1984, 2015 [cit. 2018-05-29]. Dostupné z: https://www3.lenovo.com/bd/en/smartphones/smartphone-k-series/K3-Note/p/PPIPPIK50A4

[13] KONSUMER, David. Tplink-lightbulb. In: *Npm* [online]. California (Oakland), 2009, 8. 1. 2017 [cit. 2018-05-22]. Dostupné z: https://www.npmjs.com/package/tplink-lightbulb

[14] Grafické zobrazení HSV. In: *Wikipedie* [online]. San Francisco, 2003 [cit. 2018-05-22]. Dostupné z: https://cs.wikipedia.org/wiki/HSV#/media/File:HSV_color_solid_cylinder.png

[15] KASÍK, Pavel. Dřív, než jsme čekali. Google už testuje umělou inteligenci v češtině. In: Technet.cz: *Technet.cz: Technika kolem nás* [online]. Krakov, 2017, 7. 9. 2017 [cit. 2018-05-21]. Dostupné z: https://technet.idnes.cz/google-asistent-konverzace-umela-inteligence-google-cesky-jazyky-pet-let-1r1-/sw_internet.aspx?c=A170906_214522_sw_internet_pka

[16] VACULÍK, Přemysl. Google Assistant se pomalinku učí česky a slovensky. In: *Dotekománie.cz: Vše o mobilech a tabletech* [online]. 2017, 10. 7. 2017 [cit. 2018-05-21]. Dostupné z: https://dotekomanie.cz/2017/07/google-assistant-se-pomalinku-uci-cesky/

[17] *Google Allo: A smart messaging app* [online]. USA (California), 2016 [cit. 2018-05-21]. Dostupné z: https://allo.google.com/

[18] *Google Cloud Platform* [online]. California (Mountain View), 2011 [cit. 2018-05-27]. Dostupné z: https://console.cloud.google.com/home/

[19] Registration Tool Help. *Google Assistant SDK for Devices* [online]. California (Mountain View), 2016, 6. 1. 2018 [cit. 2018-05-27]. Dostupné z: https://developers.google.com/assistant/sdk/reference/device-registration/device-tool

[20] Manually Register a Device with the REST API. *Google Assistant SDK for Devices* [online]. California (Mountain View), 2016, 9. 3. 2018 [cit. 2018-05-27]. Dostupné z: https://developers.google.com/assistant/sdk/reference/device-registration/register-device-manual

[21] Install the SDK and Sample Code: Generate credentials. *Google Assistant SDK for Devices* [online]. California (Mountain View), 2016, 7. 5, 2018 [cit. 2018-05-27]. Dostupné z: https://developers.google.com/assistant/sdk/guides/library/python/embed/install-sample

[22] Gactions CLI. *Actions on Google* [online]. California (Mountain View), 2016, 15. 12. 2017 [cit. 2018-05-27]. Dostupné z: https://developers.google.com/actions/tools/gactions-cli

[23] WILSON, Douglas a James SNELL. Express: Fast, unopinionated, minimalist web framework for node. In: *Npm* [online]. California (Oakland), 2009, 24. 10. 2012 [cit. 2018-05-25]. Dostupné z: https://www.npmjs.com/package/express

[24] Server requirements to run Node.js [closed]. In: *Stack Overflow: Learn, Share, Build* [online]. New York, 2008, 26. 7. 2015 [cit. 2018-05-29]. Dostupné z: https://stackoverflow.com/questions/31638324/server-requirements-to-run-node-js

[25] Actions-on-google: Actions on Google Client Library. In: *Npm* [online]. California (Oakland), 2009, 29. 11. 2016 [cit. 2018-05-25]. Dostupné z: https://www.npmjs.com/package/actions-on-google

[26] WILSON, Douglas. Body-parser. In: *Npm* [online]. California (Oakland), 2009, 6. 1. 2014 [cit. 2018-05-25]. Dostupné z: https://www.npmjs.com/package/body-parser

[27] DEFELIPPI, Dan. Xmlhttprequest: node-XMLHttpRequest. In: *Npm* [online]. California (Oakland), 2009, 4. 3, 2011 [cit. 2018-05-25]. Dostupné z: https://www.npmjs.com/package/xmlhttprequest

[28] COOKE, Brian. Pigpio. In: *Npm* [online]. California (Oakland), 2009, 14. 10. 2015 [cit. 2018-05-25]. Dostupné z: https://www.npmjs.com/package/pigpio

[29] Define Actions. *Actions on Google* [online]. California (Mountain View), 2016, 21. 3. 2018 [cit. 2018-05-27]. Dostupné z: https://developers.google.com/actions/sdk/define-actions

[30] Smartphone free icon. In: *Flaticon* [online]. Španělsko, 2013 [cit. 2018-05-25]. Dostupné z: https://www.flaticon.com/free-icon/smartphone_149007

[31] Smartwatch free icon. In: *Flaticon* [online]. Španělsko, 2013 [cit. 2018-05-25]. Dostupné z: https://www.flaticon.com/free-icon/smartwatch_911407#term=smartwatch&page=1&position=11

[32] Raspberry Pi: vector logo. In: WorldVectorLogo [online]. [cit. 2018-05-25]. Dostupné z: https://worldvectorlogo.com/logo/raspberry-pi

[33] RITTMEYER, Wolfram. Using the Actions SDK for Google Assistant Development: Action SDK workflow — picture Google, Inc. In: *DZone* [online]. USA (New York), 1997, 17. 10. 2017 [cit. 2018-05-25]. Dostupné z: https://www.grokkingandroid.com/wordpress/wp-content/uploads/2017/10/action_sdk_workflow.png

[34] Wifi signal free icon. In: *Flaticon* [online]. Španělsko, 2013 [cit. 2018-05-25]. Dostupné z: https://www.flaticon.com/free-icon/wifi-signal_254077

[35] TP Link: vector logo. In: *WorldVectorLogo* [online]. [cit. 2018-05-25]. Dostupné z: https://worldvectorlogo.com/logo/tp-link

[36] LAM, Hoi. Wear OS by Google: AoG support and new enhanced battery saver mode. In: *Android Developers Blog: The latest Android and Google Play news for app and game developers.* [online]. California (Mountain View), 2007, 9. 5. 2018 [cit. 2018-05-24]. Dostupné z: https://android-developers.googleblog.com/2018/05/wear-os-by-google-aog-support.html

[37] Wear OS preview downloads. In: *Android Developers* [online]. California (Mountain View), 2007, 8. 5. 2018 [cit. 2018-05-24]. Dostupné z: https://developer.android.com/wear/releases/wear-preview-downloads

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API	Rozhraní pro programování aplikací
Fulfillment	Prostředí ve kterém jsou vykonávány akce ze služby Actions
GET	Metoda pro získání specifických dat pomocí HTTP protokolu
GPIO	Univerzální vstupně-výstupní rozhraní
HDMI	Multimediální rozhraní pro přenos ve vysoké kvalitě
HTTP	Hypertextový přenosový protokol
HTTPS	Zabezpečený hypertextový přenosový protokol
IFTTT	Internetová služba propojující komunikaci jiných služeb, za pomocí webhooků
IP	Internetový protokol
POST	Metoda pro zaslání/aktualizaci dat na serveru
RAM	Paměť s libovolným přístupem
RGB	Barevný model červená-zelená-modrá
SD	Secure Digital
SIM	Subscriber Identity Module
URL	Jednotná adresa zdroje
USB	Univerzální sériová sběrnice
VNC	Program pro vzdálené připojení ke grafickému rozhraní uživatele, pomocí internetové sítě
WebHook	Odesílá HTTP požadavek na definovanou URL, pokud nastane žádaná akce
WiFi	Wireless Fidelity - „bezdrátová věrnost“

SEZNAM PŘÍLOH

A NODE.JS

A.1 Hlavní aplikace *app.js*

```
1  'use strict';
2  // Importované balíčky a globální proměnné
3  // Google Actions SDK
4  var ActionsSdkApp = require('actions-on-google').
       ActionsSdkApp;
5  // Server
6  var express = require('express');
7  var expressApp = express();
8  var port = process.env.PORT || 55080;
9  var app;
10 var bodyParser = require('body-parser');
11 // Ovládání zásuvek
12 var XMLHttpRequest = require("xmlhttprequest").
       XMLHttpRequest;
13 // Ovládání žárovky
14 var TPLSmartDevice = require('tplink-lightbulb');
15 // Ovládání GPIO pinů na Raspberry
16 var Gpio = require('pigpio').Gpio;
17 var trigger = new Gpio(6, {mode: Gpio.OUTPUT});
18 var echo = new Gpio(5, {mode: Gpio.INPUT, alert: true});
19
20 // Zjištění pozice dveří (Otevřené / Zavřené)
21 function stateOfDoor() {
22   var MICROSECONDS_PER_CM, startTick, endTick, diff,
         dist;
23   const constDist = 200;
24   trigger.digitalWrite(0);
25   (function () {
26     trigger.trigger(10, 1);
27     echo.on('alert', function (level, tick) {
28       if (level == 1) {
29         startTick = tick;
30       } else {
31         MICROSECONDS_PER_CM = 1e6/34321;
32         endTick = tick;
```

```
33          diff = (endTick >> 0) - (startTick >> 0);
34          dist = Math.round((diff / 2 / MICROSECONDS_PER_CM
               ) * 100) / 100;
35          if (dist < constDist) {
36            app.ask("Door is opened.");
37          } else {
38            app.ask("Door is closed.");
39          }
40        }
41      });
42    }());
43  }
44  // Nastavení žárovky na určitou úroveň jasu
45  function brightnessToSpecific(level) {
46    const specLightBrightness = new TPLSmartDevice('
         192.168.1.101');
47    specLightBrightness.info().then(function(info) {
48      if(info.light_state.on_off === 1) {
49        var loadedHue = info.light_state.hue;
50        var loadedSaturation = info.light_state.saturation;
51        roomLight(true, 10, loadedHue, loadedSaturation,
             level);
52        app.ask("Brightness is set to " + level +"%");
53      } else if(info.light_state.on_off === 0) {
54        app.ask("The room light is currently OFF");
55      } else {
56        app.ask("There is an error with reading the value
             of light state.");
57      }
58    }).catch(e => console.error(e));
59  }
60
61  // Zapnutí zásuvek
62  function httpGetOn(theUrl) {
63    var xmlHttp = new XMLHttpRequest();
64    xmlHttp.open( "GET", theUrl, false ); // false pro
         synchronní požadavek
65    xmlHttp.send( null );
66    return xmlHttp.responseText;
```

```
67 }
68 // Vypnutí zásuvek
69 function httpGetOff(theUrl) {
70   var xmlHttp = new XMLHttpRequest();
71   xmlHttp.open( "GET", theUrl, false ); // false pro
       synchronní požadavek
72   xmlHttp.send( null );
73   return xmlHttp.responseText;
74 }
75 // Hlavní nastvení žárovky
76 function roomLight(state, transition, hue, saturation,
     brightness) {
77   const lighting = new TPLSmartDevice('192.168.1.101');
78   lighting.power(state, transition, {"hue": hue, "
       saturation": saturation, "brightness": brightness});
79 }
80 // Získání informací o stavu žárovky
81 function getLightInfo(value) {
82   const currentLight = new TPLSmartDevice('192.168.1.101'
       );
83   switch(value) {
84     case "state":
85       currentLight.info().then(function(info) {
86         if(info.light_state.on_off === 1) {
87           app.ask("The room light is ON");
88         } else if(info.light_state.on_off === 0) {
89           app.ask("The room light is OFF");
90         } else {
91           app.ask("There is an error with reading the
                value of light state.");
92         }
93       }).catch(e => console.error(e));
94       break;
95     case "brightness":
96       currentLight.info().then(function(info) {
97         if(info.light_state.on_off === 1) {
98           app.ask("Actual level of brightness is " + info
                .light_state.brightness + "%");
99         } else if(info.light_state.on_off === 0) {
```

```
100           app.ask("The room light is currently OFF");
101         } else {
102           app.ask("There is an error with reading the
                 value of light state.");
103         }
104       }).catch(e => console.error(e));
105       break;
106     case "color":
107       currentLight.info().then(function(info) {
108         if(info.light_state.on_off === 1) {
109           if(info.light_state.saturation > 0) {
110             if((info.light_state.hue >= 0 && info.
                   light_state.hue <= 25) || (info.
                   light_state.hue > 340 && info.light_state.
                   hue <= 360)) {
111               app.ask("Color of room light is RED");
112             } else if(info.light_state.hue > 25 && info.
                   light_state.hue <= 50) {
113               app.ask("Color of room light is ORANGE");
114             } else if(info.light_state.hue > 50 && info.
                   light_state.hue <= 75) {
115               app.ask("Color of room light is YELLOW");
116             } else if(info.light_state.hue > 75 && info.
                   light_state.hue <= 170) {
117               app.ask("Color of room light is GREEN");
118             } else if(info.light_state.hue > 170 && info.
                   light_state.hue <= 260) {
119               app.ask("Color of room light is BLUE");
120             } else if(info.light_state.hue > 260 && info.
                   light_state.hue <= 290) {
121               app.ask("Color of room light is PURPLE");
122             } else if(info.light_state.hue > 290 && info.
                   light_state.hue <= 340) {
123               app.ask("Color of room light is PINK");
124             } else {
125               app.ask("There is an error with reading the
                     value of light color.");
126             }
127           } else if(info.light_state.saturation === 0) {
```

```
128             app.ask("Color of room light is WHITE");
129           }
130         } else if(info.light_state.on_off === 0) {
131           app.ask("The light is currently OFF");
132         } else {
133           app.ask("There is an error with reading the
                value of light state.")
134         }
135       }).catch(e => console.error(e));
136       break;
137   }
138 }
139 // Hlavní funkce pro zpracování POST požadavků
140 function handlePost(request, response) {
141  // Zpracování požadavků od Google Asistenta
142  app = new ActionsSdkApp( {request: request, response:
       response} );
143  // Výpis Intents pro rozpoznání v požadavcích
144  const actionMap = new Map();
145  actionMap.set(app.StandardIntents.MAIN, mainIntent);
146  actionMap.set(app.StandardIntents.TEXT, textIntent);
147  app.handleRequest(actionMap);
148  // Hlavní funkce s úvodním pozdravem
149  function mainIntent(app) {
150     app.ask("Welcome to My House Control Application.");
151  }
152  // Zpracování textových zpráv zadaných do GA
153  function textIntent(app) {
154   var text = app.getRawInput().toLowerCase();
155   switch(text) {
156 /******** OVLÁDÁNÍ ZÁSUVKY 1 - STOLNÍ LAMPIČKA ********/
157     case "switch light outlet on":
158     case "switch lamp on":
159     case "switch lamp outlet on":
160     case "turn light outlet on":
161     case "turn lamp on":
162     case "turn lamp outlet on":
163     case "switch the light outlet on":
164     case "switch the lamp on":
```

```
165     case "switch the lamp outlet on":
166     case "turn the light outlet on":
167     case "turn the lamp on":
168     case "turn the lamp outlet on":
169       httpGetOn("http://192.168.1.106/command?XC_FNC=
            SendSC&type=IT&data=01E");
170       app.ask("I'm turning on the outlet with the smiley
            face.");
171       break;
172     case "switch light outlet off":
173     case "switch lamp off":
174     case "switch lamp outlet off":
175     case "turn light outlet off":
176     case "turn lamp off":
177     case "turn lamp outlet off":
178     case "switch the light outlet off":
179     case "switch the lamp off":
180     case "switch the lamp outlet off":
181     case "turn the light outlet off":
182     case "turn the lamp off":
183     case "turn the lamp outlet off":
184       httpGetOff("http://192.168.1.106/command?XC_FNC=
            SendSC&type=IT&data=016");
185       app.ask("I'm turning off the outlet with the smiley
             face.");
186       break;
187 /******* OVLÁDÁNÍ ZÁSUVKY 2 - SVĚTLO DO ZÁSUVKY *******/
188     case "switch smiley on":
189     case "switch smile face on":
190     case "switch outlet on":
191     case "turn smiley on":
192     case "turn smile face on":
193     case "turn outlet on":
194     case "switch the smiley on":
195     case "switch the smile face on":
196     case "switch the outlet on":
197     case "turn the smiley on":
198     case "turn the smile face on":
199     case "turn the outlet on":
```

```
200       case "turn outlet two on":
201       case "turn the outlet two on":
202       case "turn outlet 2 on":
203       case "turn the outlet 2 on":
204       case "switch outlet two on":
205       case "switch the outlet two on":
206       case "switch outlet 2 on":
207       case "switch the outlet 2 on":
208         httpGetOn("http://192.168.1.106/command?XC_FNC=
              SendSC&type=IT&data=02E");
209         app.ask("I'm turning on the outlet with the smiley
              face.");
210         break;
211       case "switch smiley off":
212       case "switch smile face off":
213       case "switch outlet off":
214       case "turn smiley off":
215       case "turn smile face off":
216       case "turn outlet off":
217       case "switch the smiley off":
218       case "switch the smile face off":
219       case "switch the outlet off":
220       case "turn the smiley off":
221       case "turn the smile face off":
222       case "turn the outlet off":
223       case "turn outlet two off":
224       case "turn the outlet two off":
225       case "turn outlet 2 off":
226       case "turn the outlet 2 off":
227       case "switch outlet two off":
228       case "switch the outlet two off":
229       case "switch outlet 2 off":
230       case "switch the outlet 2 off":
231         httpGetOff("http://192.168.1.106/command?XC_FNC=
              SendSC&type=IT&data=026");
232         app.ask("I'm turning off the outlet with the smiley
               face.");
233         break;
234   /***************** OVLÁDÁNÍ ŽÁROVKY *****************/
```

```
235     case "room light on":
236     case "set room light on":
237     case "turn room light on":
238     case "the room light on":
239     case "set the room light on":
240     case "turn the room light on":
241       roomLight(true, 10, 0, 0, 100);
242       app.ask("Room light is ON");
243       break;
244     case "room light off":
245     case "set room light off":
246     case "turn room light off":
247     case "the room light off":
248     case "set the room light off":
249     case "turn the room light off":
250       roomLight(true, 1, 0, 0, 0);
251       roomLight(false, 1, 0, 0, 0);
252       app.ask("Room light is OFF");
253       break;
254     case "change color to red":
255     case "set color to red":
256     case "switch color to red":
257       roomLight(true, 10, 0, 100, 100);
258       app.ask("Light is set to red");
259       break;
260     case "change color to blue":
261     case "set color to blue":
262     case "switch color to blue":
263       roomLight(true, 10, 250, 100, 100);
264       app.ask("Light is set to blue");
265       break;
266     case "change color to green":
267     case "set color to green":
268     case "switch color to green":
269       roomLight(true, 10, 115, 100, 100);
270       app.ask("Light is set to green");
271       break;
272     case "change color to white":
273     case "set color to white":
```

```
274     case "switch color to white":
275       roomLight(true, 10, 0, 0, 100);
276       app.ask("Light is set to white");
277     case "get state":
278     case "get the state":
279       getLightInfo("state");
280       break;
281     case "get brightness":
282     case "get the brightness":
283       getLightInfo("brightness");
284       break;
285     case "get color":
286     case "get the color":
287       getLightInfo("color");
288       break;
289 /*************** OVLÁDÁNÍ HLADINY JASU **************/
290     // Snížení jasu o 10%
291     case "lower brightness little":
292     case "lower brightness a little":
293     case "lower brightness little bit":
294     case "lower brightness a little bit":
295     case "lower brightness level little":
296     case "lower brightness level a little":
297     case "lower brightness level little bit":
298     case "lower brightness level a little bit":
299       const brightnessMinusTen = new TPLSmartDevice('
            192.168.1.101');
300       brightnessMinusTen.info().then(function(info) {
301         if(info.light_state.on_off === 1) {
302           var loadedHue = info.light_state.hue;
303           var loadedSaturation = info.light_state.
              saturation;
304           var loadedBrightness = info.light_state.
              brightness;
305           if(loadedBrightness >= 10 && loadedBrightness
              <= 100) {
306             roomLight(true, 10, loadedHue,
                loadedSaturation, (loadedBrightness - 10))
                ;
```

```
307             app.ask("Actual level of brightness is " + (
                    loadedBrightness - 10) + "%");
308           } else {
309             app.ask("Brightness can't be lowered.");
310           }
311         } else if(info.light_state.on_off === 0) {
312           app.ask("The room light is currently OFF");
313         } else {
314           app.ask("There is an error with reading the
                  value of light state.");
315         }
316       }).catch(e => console.error(e));
317       break;
318     // Snížení jasu o 25%
319     case "lower brightness more":
320     case "lower brightness a more":
321     case "lower brightness":
322     case "lower brightness level more":
323     case "lower brightness level a more":
324     case "lower brightness level":
325       const brightnessMinusTwentyFive = new
             TPLSmartDevice('192.168.1.101');
326       brightnessMinusTwentyFive.info().then(function(info
             ) {
327         if(info.light_state.on_off === 1) {
328           var loadedHue = info.light_state.hue;
329           var loadedSaturation = info.light_state.
                  saturation;
330           var loadedBrightness = info.light_state.
                  brightness;
331           if(loadedBrightness >= 25 && loadedBrightness
                  <= 100) {
332             roomLight(true, 10, loadedHue,
                    loadedSaturation, (loadedBrightness - 25))
                    ;
333             app.ask("Actual level of brightness is " + (
                    loadedBrightness - 25) + "%");
334           } else {
335             app.ask("Brightness can't be lowered.");
```

```
336           }
337         } else if(info.light_state.on_off === 0) {
338           app.ask("The room light is currently OFF");
339         } else {
340           app.ask("There is an error with reading the
                value of light state.");
341         }
342       }).catch(e => console.error(e));
343       break;
344     // Zvýšení jasu o 10%
345     case "higher brightness little":
346     case "higher brightness a little":
347     case "higher brightness little bit":
348     case "higher brightness a little bit":
349     case "higher brightness level little":
350     case "higher brightness level a little":
351     case "higher brightness level little bit":
352     case "higher brightness level a little bit":
353       const brightnessPlusTen = new TPLSmartDevice('
            192.168.1.101');
354       brightnessPlusTen.info().then(function(info) {
355         if(info.light_state.on_off === 1) {
356           var loadedHue = info.light_state.hue;
357           var loadedSaturation = info.light_state.
                saturation;
358           var loadedBrightness = info.light_state.
                brightness;
359           if(loadedBrightness >= 0 && loadedBrightness <=
                90) {
360             roomLight(true, 10, loadedHue,
                  loadedSaturation, (loadedBrightness + 10))
                  ;
361             app.ask("Actual level of brightness is " + (
                  loadedBrightness + 10) + "%");
362           } else {
363             app.ask("Brightness can't be lowered.");
364           }
365         } else if(info.light_state.on_off === 0) {
366           app.ask("The room light is currently OFF");
```

```
367         } else {
368           app.ask("There is an error with reading the
                value of light state.");
369         }
370       }).catch(e => console.error(e));
371       break;
372     // Zvýšení jasu o 25%
373     case "higher brightness more":
374     case "higher brightness a more":
375     case "higher brightness":
376     case "higher brightness level more":
377     case "higher brightness level a more":
378     case "higher brightness level":
379       const brightnessPlusTwentyFive = new TPLSmartDevice
            ('192.168.1.101');
380       brightnessPlusTwentyFive.info().then(function(info)
            {
381         if(info.light_state.on_off === 1) {
382           var loadedHue = info.light_state.hue;
383           var loadedSaturation = info.light_state.
                saturation;
384           var loadedBrightness = info.light_state.
                brightness;
385           if(loadedBrightness >= 0 && loadedBrightness <=
                75) {
386             roomLight(true, 10, loadedHue,
                  loadedSaturation, (loadedBrightness + 25))
                  ;
387             app.ask("Actual level of brightness is " + (
                  loadedBrightness + 25) + "%");
388           } else {
389             app.ask("Brightness can't be lowered.");
390           }
391         } else if(info.light_state.on_off === 0) {
392           app.ask("The room light is currently OFF");
393         } else {
394           app.ask("There is an error with reading the
                value of light state.");
395         }
```

```
396         }).catch(e => console.error(e));
397         break;
398       // Nastavení jasu na určitou úroveň
399       case "set brightness to 10":
400       case "set brightness to 10%":
401         brightnessToSpecific(10);
402         break;
403       case "set brightness to 20":
404       case "set brightness to 20%":
405         brightnessToSpecific(20);
406         break;
407       case "set brightness to 25":
408       case "set brightness to 25%":
409         brightnessToSpecific(25);
410         break;
411       case "set brightness to 30":
412       case "set brightness to 30%":
413         brightnessToSpecific(30);
414         break;
415       case "set brightness to 40":
416       case "set brightness to 40%":
417         brightnessToSpecific(40);
418         break;
419       case "set brightness to 50":
420       case "set brightness to 50%":
421         brightnessToSpecific(50);
422         break;
423       case "set brightness to 60":
424       case "set brightness to 60%":
425         brightnessToSpecific(60);
426         break;
427       case "set brightness to 70":
428       case "set brightness to 70%":
429         brightnessToSpecific(70);
430         break;
431       case "set brightness to 75":
432       case "set brightness to 75%":
433         brightnessToSpecific(75);
434         break;
```

```
435       case "set brightness to 80":
436       case "set brightness to 80%":
437         brightnessToSpecific(80);
438         break;
439       case "set brightness to 90":
440       case "set brightness to 90%":
441         brightnessToSpecific(90);
442         break;
443       case "set brightness to 100":
444       case "set brightness to 100%":
445         brightnessToSpecific(100);
446         break;
447       case "get state of door":
448       case "state of door":
449       case "get state of doors":
450       case "state of doors":
451       case "is door opened":
452       case "are doors opened":
453       case "what's the state of door":
454       case "what is the state of door":
455         stateOfDoor();
456         break;
457 // Výchozí odpověď
458       default:
459         app.ask("Sorry, this is not possible right now.");
460     }
461   }
462 }
463 // Spuštění lokálního serveru na portu 55080
464 expressApp.set('port', port);
465 expressApp.use( bodyParser.json( {type: 'application/json
      '} ) );
466 expressApp.post('/', handlePost);
467 expressApp.listen(port);
468 console.log('Home Control Assistant listening on port %s'
      , port);
469 module.exports = expressApp;
```

B ACTIONS ON GOOGLE

B.1 Soubor *action.json*

```
{
"actions": [{
  "description": "Default Welcome Intent",
    "name": "MAIN",
    "fulfillment": {
      "conversationName": "myHouseControl"
    },"intent": {
      "name": "actions.intent.MAIN",
        "trigger": {
          "queryPatterns": [
          "OK, Google talk to My House Control"
          ]
        }
    }
}],
"conversations": {
  "myHouseControl": {
    "name": "myHouseControl",
    "url": "https://51dd7dab.ngrok.io", // URL ngroku má
       trvání 8 hodin. URL zde uvedená již není platná
    "fulfillmentApiVersion" : 2
  }
}
}
```

C OBSAH CD

- / ..... Kořenový adresář přiloženého CD
  - Adresář Raspberry Pi
    - BProjekt ..... Složka se souborem action.json
      - action.json
      - creds.data
      - gactions
      - ngrok
      - npm
    - env ..... Virtuální prostředí pro Google Assistant
      - bin ..... Soubory Google Assistant SDK
      - include
      - lib
        - python3.5 ..... Knihovny Pythonu 3.5
      - share
      - pyvenv.cfg
    - node_modules ..... Knihovny prostředí NPM
    - app.js ..... Soubor hlavní aplikace
    - client_secret_795286533353 ..... Autorizační klíč
    - package.json
    - package-lock.json
  - Bakalářská práce
    - Libor-Frolich-174201.pdf ..... Bakalářská práce
    - loga
    - obrazky
    - pdf
    - text
    - beamercolorthemeVUT.sty
    - beamerthemeVUT.sty
    - sablona-prace.tex
    - sablona-prace.tps
    - sablony-nastav_udaju.tex
    - thesis.sty