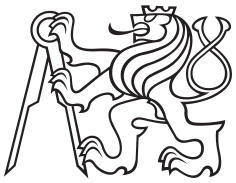
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Smart systém řízení pro žaluzie

Tomáš Bedrník

Školitel: Ing. Vladimír Janíček, Ph.D.
Studijní program: Softwarové inženýrství a technologie
Květen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bedrník** Jméno: **Tomáš** Osobní číslo: **474666**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Smart systém řízení žaluzií

Název bakalářské práce anglicky:

Smart blinds control

Pokyny pro vypracování:

[1] Seznamte se s problematikou smart zařízení pro motorizaci žaluzií, principy komunikace a přenosu dat, principy návrhu systému kompatibilním s Google Home a s problematikou solárního napájení v IoT.
[2] Proveďte analýzu trhu s motorizovanými žaluziemi.
[3] Navrhněte systém pro řízení horizontálních žaluzií ve formě přídavného modulu ke standardním žaluziím. Navrhněte rozhraní pro prvotní nastavení a propojení s Google Home.
[4] Sestrojte model demonstrující zařízení v provozu na reálných žaluziích a otestujte na něm funkčnost řízení vertikální polohy a náklonu. Porovnejte dosažené parametry s minimálně třemi komerčními výrobky.

Seznam doporučené literatury:

[1] Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32, ISBN 978-1789956870
[2] https://www.espressif.com/en/products/hardware/esp32/overview

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Vladimír Janíček, Ph.D., katedra mikroelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.02.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce: **30.09.2021**

Ing. Vladimír Janíček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

Děkuji vedoucímu práce Ing. Vladimíru Janíčkovi, Ph.D. za cenné rady a věcné připomínky při vypracování této práce. Také děkuji tátovi za poskytnutí rad a nástrojů k sestavení modelu řešení.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 22. května 2020

Abstrakt

Cílem této práce bylo navrhnout chytrý systém ovládání žaluzií za využití servo motoru a platformy ESP8266. Pomocí WiFi a MQTT je zařízení spojeno s vlastním NodeJS serverem, který umožňuje ovládání žaluzií z webové aplikace napsané v Reactu, či hlasem z Google Home.

Klíčová slova: chytrá domácnost, žaluzie, Google Home, servo pohon, ESP8266, MQTT, NodeJS, React

Školitel: Ing. Vladimír Janíček, Ph.D. K13134 - Katedra mikroelektroniky Elektrotechnická fakulta, České Vysoké Učení Technické, Technická 2, 16627 Praha 6

Abstract

Goal of this work was to develop a smart controller for blinds using a servo motor and ESP8266 platform. These were connected using MQTT with custom NodeJS server, that let us control blinds from React based web app or by voice from Google Home.

Keywords: smart home, blinds, Google Home, servo motor, ESP8266, MQTT, NodeJS, React

Title translation: Smart blinds control

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Takzvaná „smart“ zařízení jsou zcela jistě aktuálním trendem. Pro svou dostupnost a klesající cenu se stávají součástí všech domácností. Skrze automatizaci se snaží přinést zjednodušení každodenního života. Mezi nejčastější produkty patří televize, světla, reproduktory nebo třeba i kuchyňské spotřebiče. Tato zařízení umožňují jejich ovládání pomocí mobilních aplikací nebo jiných chytrých ovladačů.

Atraktivnější produkty nabízí možnost propojení s ekosystémem chytré domácnosti. Tím může být pro běžného zákazníka Apple Homekit, Amazon Alexa nebo Google Home. Zkušenější uživatel využije systémů jako Home Assistant nebo OpenHAB, kvůli jejich variabilitě a možnostem vlastního nastavení.

Chytrá zařízení nemusí být vždy celý nový produkt, ale mohou být pouze rozšíření, které vdechne staršímu/hloupému spotřebiči nový život. Takovým zařízení je například chytrá zásuvka, díky které můžete telefonem zapínat a vypínat vaši lampu po prababičce. Do kategorie rozšíření spadají i systémy řízení žaluzií.

Tato práce se bude věnovat analýze a návrhu chytrého rozšíření pro již instalované žaluzie. Uvažovány budou běžné dostupné komponenty a technologie. Cílem bude navrhnout řešení kompatibilní s ekosystémem Google Home nabízející parametry srovnatelné s podobnými produkty. Tyto parametry budou otestovány na demonstračním modelu.

Kapitola 2
Analýza

Kapitola bude zpočátku věnována analýze jednotlivých součástí smart systému řízení žaluzií a principů komunikace takového systému. Následně bude zjištěno, co je třeba k propojení zařízení s Google Home. Poté se nastuduje solární napájení a způsoby jeho aplikace v internetu věcí. Nakonec bude provedena analýza trhu existujících řešení.

2.1 Smart zařízení pro motorizaci žaluzií

Tato část se bude zabývat samotnými komponentami, které dohromady utvoří chytře ovladatelné žaluzie.

2.1.1 Konstrukce žaluzií

Běžné v Česku rozšířené okenní žaluzie sestávají z několika částí. Horní lišta zakrývá osu, na kterou jsou v dvou bodech napojeny navijáky a v jednom kraji ovládací mechanismus. Tím je veden kuličkový řetízek. Lamely jsou upevněny na dva provázky a k zabránění výkyvu jim pomáhají postranní vodící lanka. Na nejspodnější lamelu je upevněno břemeno, které drží žaluzie napnuté. Lamelami je vedena látková stuha, která se navíjí na již zmíněné navijáky. Ovládání polohy i náklonu je řízeno řetízkem. Při zatažení v jednom směru se nejprve provede náklon a poté pokračuje výsuv nahoru respektive dolů.

2.1.2 Motorizace

K motorizaci se běžně využívá řetízek a to hlavně kvůli jednoduchosti instalace a také kompatibility s různými typy žaluzií a zatemňovacích rolet. Do spodní smyčky se umístí motor, který žaluzii otáčí za uživatele. Problém může nastat při různých distancích kuliček na řetízku. Lze ale také využít přímo osu. Bohužel v Česku běžné žaluzie mají lištu ve tvaru J a umisťují se na horní okraj okenního rámu. Využitelného prostoru tedy není moc (přibližně 2x2cm krát šířka žaluzií - uprostřed je osa).

Krokové motory (stepper)

Krokový motor se skládá ze statoru tvořeného sadou cívek a rotoru s permanentními magnety. Princip motoru je postupné zapojování jednotlivých cívek, čímž dochází k vytvoření rotujícího magnetického pole, které otáčí rotor. Rozlišujeme jednofázové a dvoufázové řízení. Při jednofázovém je v okamžiku zapnuta pouze jedna cívka (nebo protilehlá dvojice při bipolárním řízení). Při dvoufázovém jsou to dvě sousední cívky, jež poskytují vyšší kroutící moment, ovšem vyžadují vyšší spotřebu. Dále dělíme na řízení s plným a polovičním krokem. Při polovičním kroku dosahují dvojnásobné přesnosti. Výhodou krokových motorů je přesnost, například 200 krokový motor má krok o velikosti 1.8 stupně. Nevýhodami jsou hmotnost, spotřeba pro udržení polohy a v neposlední řadě i cena. Zdroj: [19]

Servo motory

Servo motor je stejnosměrným motorem a snímačem polohy - regulátorem, kterým může být například enkodér nebo potenciometr a převodovka. Používají se jak ve velkých rozměrech, například jako uzávěry plynu, tak i v malých, třeba v RC modelech. Tyto modelářské servo motory většinou nabízejí rotaci v rozsahu 0 až 180 stupňů, který je snímán potenciometrem. Tento typ motoru je řízen pulzně šířkovou modulací. 0 stupňů odpovídá pulzu délky 0.5ms, 180 stupňů délce 2.5ms. Střed, tedy 90 stupňů dosáhne při pulzu 1.5 ms. Pulzy se posílají jednou za 20ms. Odebráním potenciometru lze získat kontinuální rotaci. V tom případě odpovídají pulzy rychlosti rotace ve směru a středová hodnota servo zastaví. Výhodami těchto motorů jsou rozměry, jednoduchost a cena. Nevýhodou může být hluk. Zdroj: [21]

Obrázek 2.1: Servo motor MG996R [16]

2.1.3 Mikrokontroléry

Mikrokontroléry, nebo také jednočipové počítače, jsou integrované obvody určené k zajištění běhu nejčastěji jednoúčelových aplikací. Základní struktura mikrokontroléru je procesor, operační paměť, paměť programu, oscilátor a vstupně výstupní rozhraní.

Platforma ESP8266

Jedná se o velice rozšířený systém na čipu. Ve velice malých rozměrech se nachází piny s podporou analogového a digitálního zápisu a čtení, SPI, I2C sběrnice, pulzně šířkové modulace a hlavně WiFi na frekvenci 2.4GHz. Má možnost pracovat jako stanice, access-point nebo jako obojí zároveň, což lze využít v mesh instalacích. Velice rozšířen je v IoT, kde může v některých instalacích přijít k užitku nízká spotřeba energie a hlavně možnost hlubokého spánku, kdy zařízení nic nevykonává, jen čeká na probuzení a během toho téměř neodebírá elektřinu. Operuje na 3.3V, ale některé mikrokontroléry obsahují regulátor umožňující napájet 5V nebo třeba až 24V, pokud by to bylo třeba. Zdroj: [17]

Obrázek 2.2: Chip ESP8266 (ESP-12F) [22]

Platforma ESP32

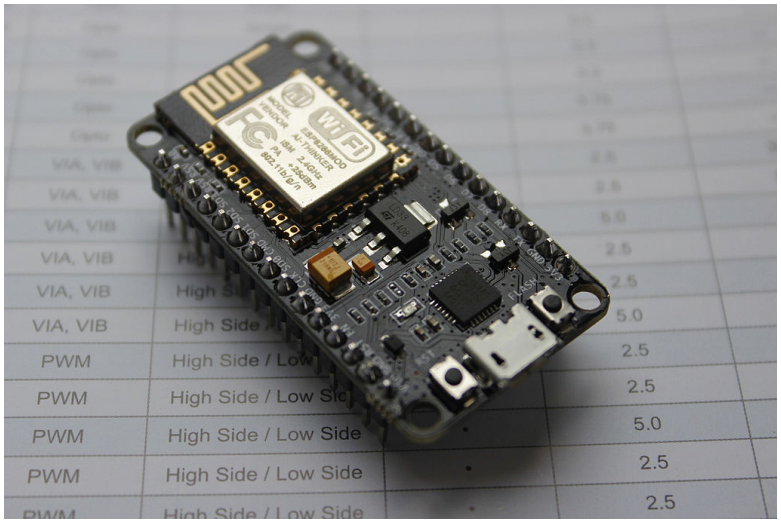
Rozšíření předchozího čipu - ESP32 - disponuje techologií Bluetooth Low Energy, která může v aplikacích nahrazujících WiFi výrazně snížit spotřebu až na desetinu. Dále poskytuje dvoujádrový procesor a integrované flash úložiště o velikosti 4MB. Právě kvůli podpoře BLE a tím nižší spotřebě se více hodí pro bateriově napájené instalace. Oproti ESP8266 má větší rozměry a při zapnuté WiFi spotřebovává více energie. Zdroj: [18]

Vývojová deska NodeMCU v1

Na desce ESP8266 NodeMCU v1.0 ESP-12E nalezneme 11 digitálních I/O pinů a 1 analogový vstupní. Pro komunikaci se servo motorem nám stačí jeden digitální pin. Deska disponuje CP2102 USB převodníkem, který nám umožňuje nahrávat programy z PC pomocí microUSB kabelu. Napětí desky je

Obrázek 2.3: Chip ESP32 (WROOM-12) [23]

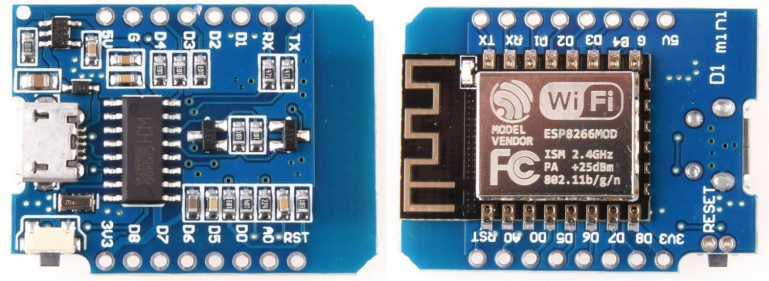
3.3V, nicméně napájení probíhá buď pomocí microUSB portu, kde je napětí 5V, nebo připojením externího zdroje na VIN pin, kde deska požaduje 7-12V. Paměť desky má velikost 4MB, to je pro naše účely dostačující. Hlavní komponentou je WiFi standardu IEEE 802.11 b/g/n. Původně byla deska navržena pro programování v jazyce Lua. Jedná se o skriptovací jazyk, který je svou jednoduchostí a syntaxí podobný Pythonu nebo Javascriptu. Kvůli rychlosti a velikosti kódu se ovšem častěji používá jazyk C/C++ a Arduino IDE. Zdroj: [13]



Obrázek 2.4: Vývojová deska NodeMCU v1.0 [15]

Vývojová deska Wemos D1 mini

Tato deska se vlastnostmi podobá desce výše. Rozdílem jsou o něco nižší rozměry a USB převodník CH340G. Zdroj: [14]



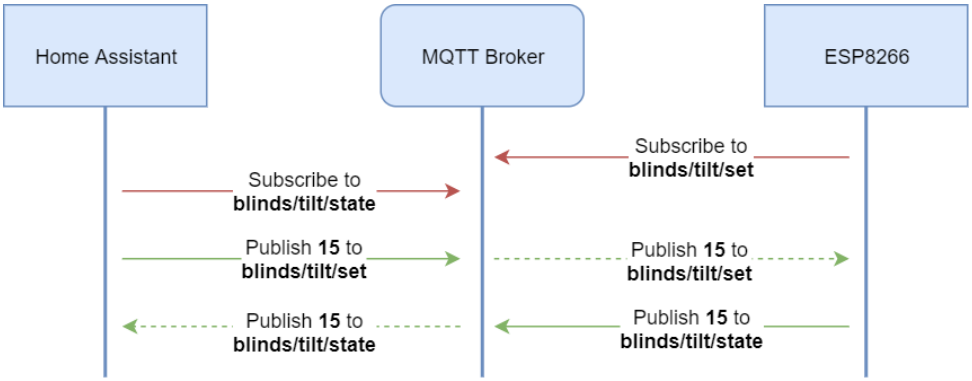
Obrázek 2.5: Vývojová deska Wemos D1 mini [14]

2.2 Principy komunikace a přenosu dat

Většinou podle místa instalace se v IoT využívají různé druhy bezdrátové komunikace, z nichž v chytrých domácnostech jsou rozšířené hlavně ZigBee, Z-Wave, Bluetooth Low Energy (BLE), Near Field Communication (NFC) a WiFi. První dvě technologie vyžadují pro komunikaci s internetem mezičlánek nazývaný Hub. Ten má za úkol předávat informace mezi zařízeními komunikujícími na protokolu ZigBee nebo Z-Wave a sítí/internetem, ke kterému je připojen prostřednictvím lokální domácí sítě po WiFi nebo UTP kabelem. To stejné platí i pro BLE zařízení, které lze ale většinou ovládat také pomocí mobilního telefonu, který Bluetooth také podporuje. NFC zařízení jsou většinou chytré zámky nebo vypínače, kde je vyžadována blízká přítomnost čipu nebo opět mobilního telefonu či chytrých hodinek podporujících NFC. Chytrá zařízení s WiFi mají tu výhodu, že je lze přímo připojit do domácí sítě bez nutnosti Hubu. Nevýhodou oproti technologiím využívajích Hub může být zarušenost prostoru nebo v případě bateriového napájení spotřeba.

2.2.1 MQTT

Jedná se o jednoduchý zprávový obousměrný protokol speciálně navržený pro použití v Internetu věcí. Poradí si v nespolehlivých sítích s nízkou šířkou pásma a vysokou odezvou. Hlavním principem je publisher-subscriber princip a client-broker architektura. Komunikace probíhá v tématech (topics), která, ačkoli to není nutné, většinou popisují cestu k zařízení a jeho funkci, například dům/druhé-patro/ložnice/detektor-kouře/stav. Do témat se buď posílají zprávy (publish) nebo se z nich čtou (subscribe).



Obrázek 2.6: Diagram MQTT

Standardně zprostředkovatel (broker) poskytuje připojení na portu 1883 pro normální komunikaci nebo 8883 pro zabezpečenou komunikaci pomocí SSL. Inicializace spojení probíhá odesláním paketu *CONNECT*, který obsahuje informace potřebné k uzavření spojení:

- Identifikátor klienta
- Kvalita služby (QoS)
  1. zpráva je doručena maximálně jednou
  2. zpráva je doručena minimálně jednou
  3. zpráva je doručena právě jednou
- Uživatelské jméno (nepovinně)
- Heslo (nepovinně)

Zprávy nemají přesně definovaný obsah. Lze použít obyčejný text nebo například data ve formátu JSON. Pro udržení spojení si klient a zprostředkovatel vyměňují zprávy PING v předem definovaném časovém intervalu. Pokud tedy klient ukončí spojení bez oznámení, server se o tom dozví po uplynutí tohoto intervalu. Zdroj: [12]

2.3 Google Home

Google Home je ekosystém a aplikace pro řízení chytré domácnosti. Výhodou je propojení s hlasovým asistentem Google Assistant, díky němuž lze povely pro ovládání chytrých zařízení provádět hlasem. Zařízení propojená s Google Home lze ovládat pomocí mobilní aplikace dostupné na systémech Android a iOS nebo pomocí chytrých reproduktorů implementujících Google asistenta, například Google Home Mini, Google Home Hub nebo Google Home Max.

Pro implementaci vlastního zařízení do infrastruktury Google Home lze využít službu třetí strany, například Home Assistant - open-source chytrá domácnost umožňující připojit vlastní i existující zařízení pomocí jednoduché konfigurace. Uživatel je ale nucen mít jej někde nainstalovaný.

Pro vlastní implementaci propojení je zapotřebí server, který se stará o předávání informací mezi zařízeními a Actions on Google - vývojovou platformou pro Google asistenta. Informace jsou čerpány z oficiální dokumentace [24].

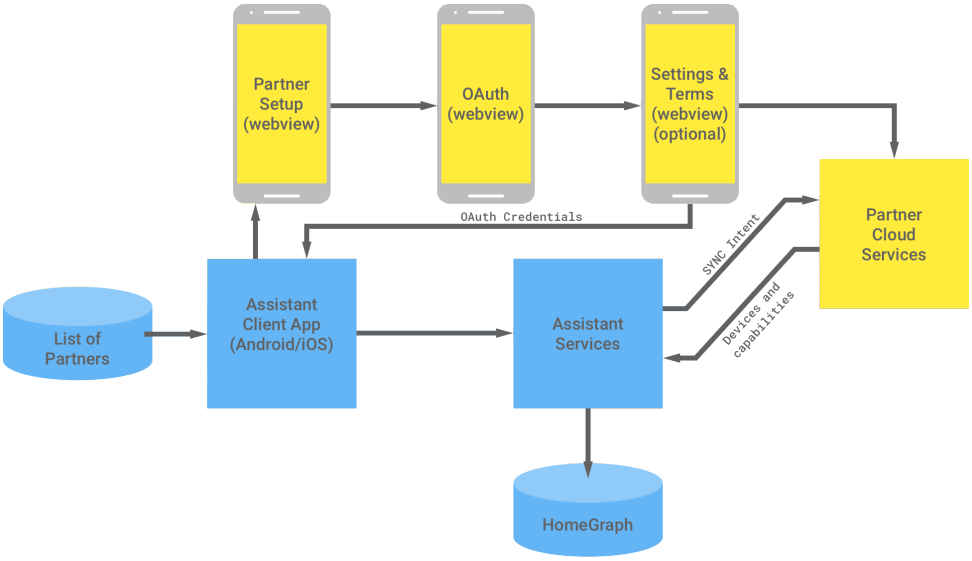
2.3.1 Reprezentace zařízení

Každé zařízení v Google Home má předdefinovaný typ (Type) a vlastnosti (Traits). Pro žaluzie je to typ `action.devices.types.BLINDS` a vlastnosti jsou `action.devices.traits.OpenClose`, u které nastavujeme procentuální hodnotu otevření, a `action.devices.traits.Rotation`, kde dáváme najevo, že zařízení se nějakým způsobem otáčí a podporuje příkazy v procentech nebo stupních. Tyto informace jsou uloženy v databázi Google HomeGraph.

Všechny typy zařízení lze ovládat hlasovými pokyny. Podmnožina těchto zařízení má v Google Home aplikaci a na Google Home Hub displeji grafické ovládání. Žaluzie v této podmnožině doposud nejsou.

2.3.2 Propojení účtu

Uživatel může do svého Google Home ekosystému přidat funkcionalitu třetí strany pomocí propojení účtu. V mobilní aplikaci vybere ze seznamu dostupných doplňků a provede propojení, které je specifikováno na obrázku 2.7. Bližší podrobnosti k jednotlivým částem budou popsány dále v textu.



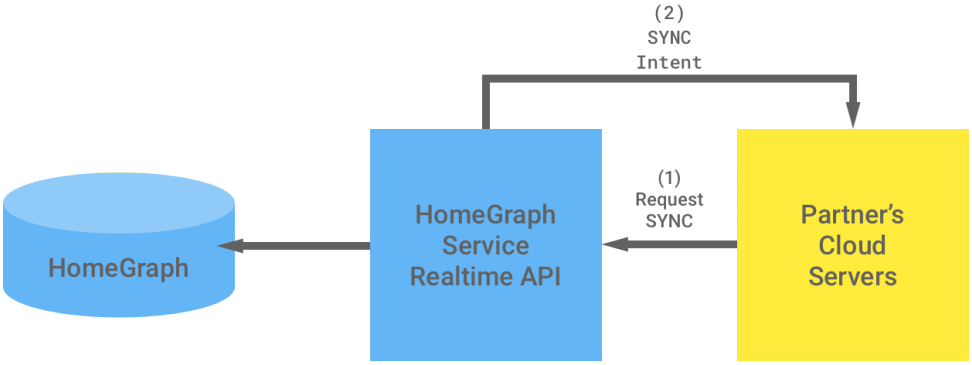
Obrázek 2.7: Diagram propojení účtu s třetí stranou [25]

2.3.3 Komunikace

Google Home komunikuje se serverem pomocí HTTP požadavků typu POST, které obsahují tzv. záměry (Smart Home Intents). Pro opačnou komunikaci, tedy když server potřebuje něco sdělit Googlu, se využívá REST API Homegraph. V té Google ukládá všechny informace o uživatelových zařízeních.

Záměr synchronizace

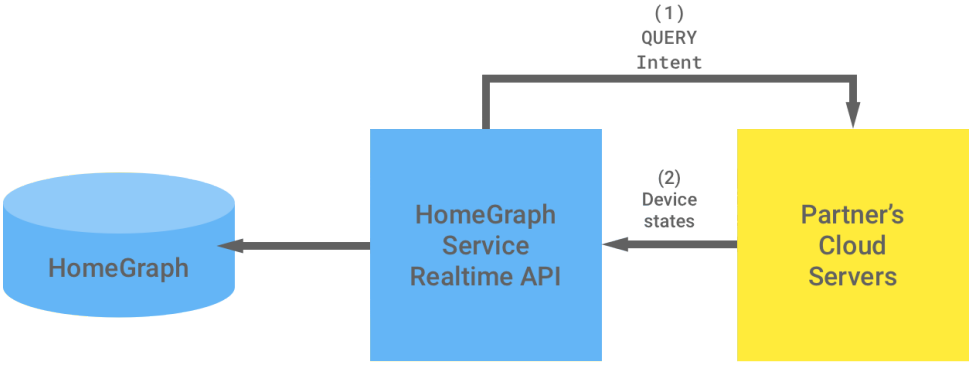
Prvním záměrem je `actions.devices.SYNC`, na který má server odpovědět polem zařízení, které má uživatel k dispozici. Tento záměr je volán při prvotní synchronizace a také při aktualizaci seznamu zařízení. Tu může uživatel vyvolat v aplikaci nebo hlasovým povelem *"Sync my devices"*. O synchronizaci může zažádat sám server, například když uživatel přidá nové zařízení.



Obrázek 2.8: Diagram - Záměr SYNC [25]

Záměr dotazu na stav

Dalším záměrem je `actions.devices.QUERY`, kdy se Google ptá na stav uživatelových zařízení a to hned po provedení synchronizace, nebo když se uživatel zeptá *"Are the blinds open?"*. Odpověď obsahuje seznam zařízení a stavové hodnoty podle vlastností definovaných v synchronizaci.



Obrázek 2.9: Diagram - Záměr QUERY [25]

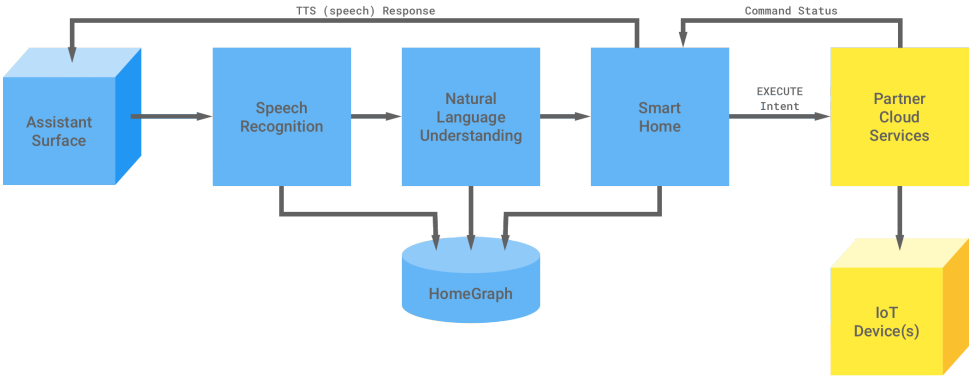
Záměr vykonání příkazu

V záměru `actions.devices.EXECUTE` se vykonávají příkazy. Požadavek obsahuje seznam zařízení a příkazy, které na nich vykonat. Při povelu *"Open kitchen blinds to 50%"* budou v příkazu všechna zařízení, která souvisí s kuchyní svým názvem nebo umístěním a `action.devices.commads.OpenPercent`

příkaz s hodnotou *50*. Na tento záměr server odpovídá seznamem zařízení seskupených podle statusu operace.

- SUCCESS - zařízení úspěšně provedlo příkaz
- PENDING - zařízení přijalo příkaz, ale ještě není vyplněn
- OFFLINE - informuje o nedostupnosti zařízení
- EXCEPTIONS - upozorňuje na problém se zařízením
- ERROR - upozorňuje na chybu při výkonu

Volitelně lze do odpovědi připojit stav těchto zařízení po výkonu příkazů.



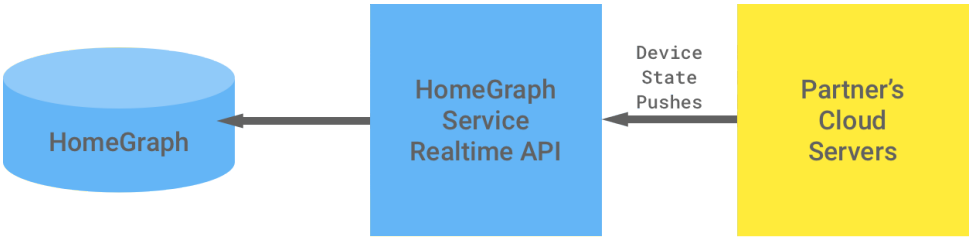
Obrázek 2.10: Diagram - Záměr EXECUTE [25]

Záměr odpojení

Poslední záměr je `action.devices.DISCONNECT`, který serveru oznamuje, že uživatel odpojil svůj účet od Google Home a server by tedy neměl dále posílat aktualizace o stavech zařízení.

Oznámení stavu

Když se změní stav zařízení, server o tom dává vědět požadavkem na Homegraph API typu *ReportState*. Ten obsahuje množinu zařízení s jejich změněnými stavy. Tělo požadavku je shodné s tělem odpovědi na záměr *QUERY*.



Obrázek 2.11: Diagram - Oznámení stavu [25]

Požadavek na synchronizaci

Když uživatel přidá, odebere nebo změní nějaké zařízení, server posílá na Homegraph API požadavek o synchronizaci *RequestSync*. Homegraph po přijmu tohoto požadavku vykoná záměr *SYNC*.

2.3.4 Zabezpečení

Google používá pro ověření identity standardizovaný autorizační framework *OAuth 2.0*. Ten definuje jakým způsobem se žádá o přístup, a jak se ověřuje. Jedná se o tzv. typy grantů (Grant types), z nichž jsou v případě Google Home důležité *Authorization Code* a *Refresh token*. Pomocí OAuth Google žádá uživatele o přístup k aplikaci pod jeho identitou. To je nezbytné k tomu, aby mohl vykonávat požadavky týkající se uživatelových zařízení.

Získání autorizačního kódu

Grant *Authorization code* popisuje získání autorizačního kódu. Ve chvíli, kdy uživatel inicializuje propojení aplikace s Google Home, pošlou Google Actions požadavek na dříve specifikovaný endpoint, typicky *https://server.com/auth*. V parametrech URL se objevuje:

- `client_id` - Identifikátor klienta (Googlu), který se zadává při konfiguraci
- `redirect_uri` - Adresa, na kterou směrovat vyřešení požadavek
- `state` - Informace, kterou pouze předáváme na adresu výše
- `response_type` - Typ odpovědi, v tomto případě vždy *code*

Tento požadavek probíhá s grafickým rozhraním, kde se uživatel přihlásí a následně akceptuje žádost o používání identity. V tu chvíli server vygeneruje autorizační kód, který předává na adresu specifikovanou v požadavku. Server si kód uloží spolu s informací, kterému klientu patří, a ke kterému uživateli se váže.

Výměna autorizačního kódu za přístupový token

Ve chvíli, kdy klient získá autorizační kód, přichází na řadu jeho výměna na přístupový token. Opět se tak děje požadavkem na dříve specifikovaný endpoint, typicky *https://server.com/token*. Tentokrát vše probíhá na pozadí bez uživatelova vědomí. Požadavek je v tomto případě typu POST a v těle obsahuje následující:

- `client_id` - Identifikátor klienta
- `client_secret` - Tajná část indetifikátoru klienta
- `grant_type` - Typ grantu, v tomto případě *autorization_code*

- `code` - Autorizační kód

Pokud jsou veškeré údaje v pořádku, dojde k vygenerování přístupového tokenu. Ten je ve formátu *JWT (Json Web Token)* a obsahuje tajným klíčem zašifrovanou informaci o klientu, uživateli a své platnosti. Dále je vygenerován tzv. *refresh_token*, který slouží k získání nového přístupového tokenu po skončení jeho platnosti. Odpověď ve formátu JSON obsahuje položky:

- `token_type` - Typ tokenu, v tomto případě *Bearer*
- `access_token` - Vygenerovaný přístupový token
- `refresh_token` - Vygenerovaný obnovovací token
- `expires_in` - Za jak dlouho končí platnost ve vteřinách

Výměna obnovovacího tokenu za přístupový token

Tento požadavek probíhá vcelku stejně jako ten předchozí. Rozdíl je pouze v některých položkách těla požadavku, které vypadá takto:

- `client_id` - Identifikátor klienta
- `client_secret` - Tajná část indetifikátoru klienta
- `grant_type` - Typ grantu, v tomto případě *refresh_token*
- `refresh_token` - Samotný obnovovací token

2.4 Solární napájení v IoT

V internetu věcí se lze často setkat s venkovními aplikacemi, například meteo-stanice nebo závlahové systémy. Vzhledem k jejich umístění a nízké energetické náročnosti se nabízí možnost využít pro jejich napájení čistou energii ze slunce. Výhodou je nulová cena provozu a samostatnost systému bez potřeby přívodního kabelu. Je třeba ovšem počítat s tím, že slunce nesvítí vždy, a to nejen ve smyslu den a noc, ale také problém počasí a jeho změn po celý rok.

Sluneční energii získanou ve dne je třeba uložit na noc. Musí se také počítat s napájením po dobu zatažené oblohy, v extrémních případech i 14 dní. Dle zeměpisné polohy zařízení a energetické náročnosti se tedy určuje velikost solárního panelu a kapacita baterie.

2.4.1 Snižování energetické náročnosti

U vývojových desek se při solárním napájení používá funkce hlubokého spánku (*deep sleep*). Jedná se o stav, ve kterém deska spotřebovává minimum energie, protože nevykonává žádnou práci. Do stavu spánku ji lze dostat příkazem v kódu. Z něj ji lze probudit buď přivedením napětí na některý pin, mnohem

častěji ale časem. Příkladem může být chytrý teploměr, který se jednou za 30 minut probudí ze spánku, změří teplotu, připojí se na WiFi, odešle naměřenou hodnotu nějakému serveru a zase se na 30 minut uvede do spánku. Probuzený je tedy třeba jen 5 sekund z 30 minut.

Tuto funkcionalitu ovšem nelze využít v aplikacích, kde zařízení udržuje neustálé spojení se serverem, protože od něj v reálném čase přijímá pokyny. Je zde ale možnost využít spánku alespoň částečně, když zařízení nevyužíváme, například v noci.

2.5 Existující řešení

Sweethome SH-RB230

Výrobek dostupný na asijském trhu umožňuje napojení na kuličkový řetízek. Ovládání probíhá pomocí mobilní aplikace. Zařízení komunikuje pomocí Bluetooth a zároveň může být spárované pouze s jedním mobilem či tabletem. V aplikaci je možné nastavit časovače. V balení nalezneme malý solární panel s přísavkami, kterými se panel přichytí na vnitřek okna. Tento panel poté dobíjí vložený akumulátor. Případně je možné zařízení připojit do zásuvky dodávaným zdrojem. Prodává se na asijském trhu za cenu okolo 30 amerických dolarů.[1]

Ningbo Yihao Blinds NYHBC002

Jedná se o motor pohánějící kuličkový řetízek. Montuje se na konec smyčky řetízku. Přímo na zařízení se nachází tři tlačítka ovládání – nahoru, stop a dolů – každé s vlastní indikační LED diodou. Dálkové ovládání komunikuje na frekvenci 433MHz. Existují různé ovladače pro 1, 2 nebo 15 těchto výrobků. Zařízení obsahuje baterii s kapacitou 1200mAh a napětím 12V. Údajnou výdrž 30-80 dní závisí na rozměru žaluzií a frekvenci používání. K nabíjení slouží dodávaný adaptér.[2] Prodává se na asijském trhu za cenu okolo 69 amerických dolarů.[3]

Ningbo Yihao Blinds YH002

Tento výrobek je velice podobný zařízení výše od stejného výrobce. Liší se v metodě dálkového ovládání, tato verze totiž kromě komunikace na frekvenci 433MHz podporuje i Wi-Fi. Pokyny tedy můžete zasílat mobilní aplikací. Nechybí zde ani možnost propojení s Google Home nebo Amazon Alexa.[4] Prodává se za cenu okolo 65 amerických dolarů.[5] Tento výrobek se nejvíce blíží zadání této práce.

Obrázek 2.12: Řídící jednotka Sweethome SH-RB230 [1]

Obrázek 2.13: Obsah balení Ningbo Yihao Blinds NYHBC002/YH002 [2]

Tilt MySmartBlinds Automation Kit

Tento produkt se na rozdíl od výše zmiňovaných výrobků instaluje přímo do profilu žaluzie. Rozměr profilu musí být 2 až 2.5 palce. Produkt je dostupný v USA, kde jsou žaluzie těchto rozměrů běžné. Pro běžné české žaluzie je tedy nepoužitelný. Zařízení je napájeno dodávanou baterií, která se také instaluje dovnitř profilu. Součástí balení jsou malé solární panely, které se umisťují pod profil na vnitřní stranu okna. Výrobek komunikuje pomocí Bluetooth s vlastní aplikací na Android a iOS.[6] Produkt stojí 149 dolarů, ke kterým lze připočítat dalších 89 dolarů za bridge, který umožňuje propojení tohoto výrobku s Amazon Alexa.[7]

Obrázek 2.14: Obsah balení Tilt MySmartBlinds Automation Kit [8]

Kapitola 3

Návrh

V této kapitole budou popsány jednotlivé části, které společně vytvoří chytré ovládání žaluzií kompatibilní s ekosystémem Google Home. První částí bude modul řízení žaluzií, který na požádání vykoná změnu pozice či náklonu žaluzie. Druhá část bude věnována serveru, který bude mít na starost zpracování požadavků od Google Home a uživatele a komunikaci s žaluziemi. Poslední třetí část bude věnována uživatelskému rozhraní, ve kterém bude uživatel provádět akce spojené s žaluziemi.

3.1 Modul řízení žaluzií

3.1.1 Způsob napojení na žaluzie

K motorizovanému pohonu žaluzií lze využít řetízek nebo přímo osu žaluzie. Všechna existující kompatibilní řešení využívají řetízek. Zařízení se tak mnohdy nachází ve výšce očí a může působit nevkusně. Pokud se na zařízení mají nacházet ovládací prvky, je to ovšem chytré umístění. Pro dosažení elegantnějšího výsledku bude i přes náročnější instalaci využita osa žaluzie. Otáčena bude dvojicí ozubených kol. Jedno bude umístěno na ose žaluzie a druhé bude připevněno na motor. Všechny komponenty se pak mohou umístit na lištu žaluzie a pro dosažení čistého vzhledu lze vše zakrýt.

3.1.2 Pohon žaluzií

Z analýzy motorů lze soudit, že krokový motor je ideálnějším kandidátem, a to z důvodů možnosti přesné rotace. Servo motor ovšem nabízí nižší rozměry i spotřebu. Přesnost lze řídit optimisticky, a to pomocí sepnutí v časovém úseku. I pro jednodušší zapojení a nižší cenu bude zvolen právě servo motor, konkrétně MG996 upravený na kontinuální rotaci.

3.1.3 Mikrokontrolér

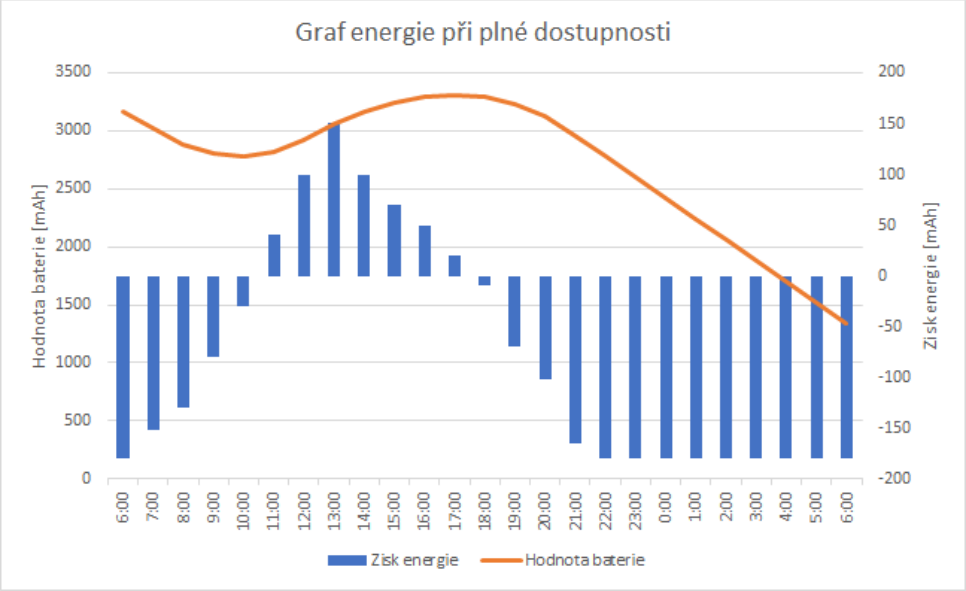
Volba desky závisí na typu zvolené sítě. Pokud pro komunikaci využijeme Bluetooth Low Energy, bude možné využít k ovládání mobilní telefon s Bluetooth, ale přímé spojení s Google Home možné nebude. Byla by možnost

vyvinout můstek, který by byl ve spojený s žaluziemi přes Bluetooth (nebo i Zigbee či Z-Wave), a zároveň by byl připojen do uživatelovi LAN sítě přes WiFi nebo kabelem. Toto řešení by mělo výhodu v nižší spotřebě desky, ale nevýhodou je, že celé řešení je v tu chvíli ekonomicky náročnější. Zvolena tedy bude WiFi a tím pádem bude i dostačující deska s chipem ESP8266. Program bude kompatibilní jak s deskou Wemos D1 mini, tak i s NodeMCU v1.0.

3.1.4 Napájení

Vývojová deska bude muset přijímat požadavky v reálném čase, a proto ji nebude možné uvézt do hlubokého spánku a bude neustále připojena na WiFi. Podle [17] se její odběr při plném využití WiFi pohybuje až okolo 170mA. K této hodnotě se musí přičíst odběr servo motoru. Tady je důležité, jak často a jak dlouho bude uživatel žaluziemi pohybovat. Pokud se chystá ráno změnit náklon do vodorovné polohy a při setmění zpět do polohy uzavřené, tak bude servo v pohybu pouze jednotky vteřin denně. Rozdílné hodnoty dosáhne, když bude žaluziemi několikrát manipulovat nahoru a dolu. To v závislosti na výšce žaluzií mohou být i jednotky minut denně. V normálním případě se počítá velikost solárního panelu potřebného k soběstačnosti systému. V tomto případě jsou ale rozměry limitovány - chceme aby panel zakrýval co nejmenší část okna. Uvažován tedy bude systém umožňující variabilní změnu počtu baterií a velikosti solárních panelů. Systém přijímá požadavky v reálném čase, tzn. mikrokontrolér je 24 hodin denně připojen na WiFi a jeho celková denní spotřeba činní $24h * 170mA * 3.3V \approx 13.5Wh$. Pokud by mělo být servo v provozu celkem 5 minut denně, bude jeho denní spotřeba činit $5/60h * 400mA * 6V \approx 0.2Wh$. Celková denní spotřeba systému bude kvůli ztrátám zaokrouhlena na 15Wh. K pokrytí této hodnoty budou zapotřebí 1~2 baterie 18560. Při uvažovaných 3 hodinách plného slunečního svitu denně potřebujeme solární panel s výkonem 5W. Podle [28] je v Praze při solárním panelu umístěném vertikálně na jih nejnižší sluneční výkon v prosinci s hodnotou $0.70kWh/m^2/den$. Jednoduchou trojčlenkou získáme potřebnou velikost panelu pro napájení po jeden den $200cm^2$ při 100% účinnosti.

Pro zobrazení nevyužitelnosti čistě solárního napájení je uvažována jedna baterie 18560 o kapacitě 3350mAh a solární panel o velikosti $150cm^2$ s uváděnými parametry 6V napětí, 2W výkonu ve špičce a tedy přibližně 330mA. Při stanovení optimálních hodnot slunečního svitu zjistíme, že systém nebude soběstačný, viz 3.1. Vychází se z plně nabité baterie.

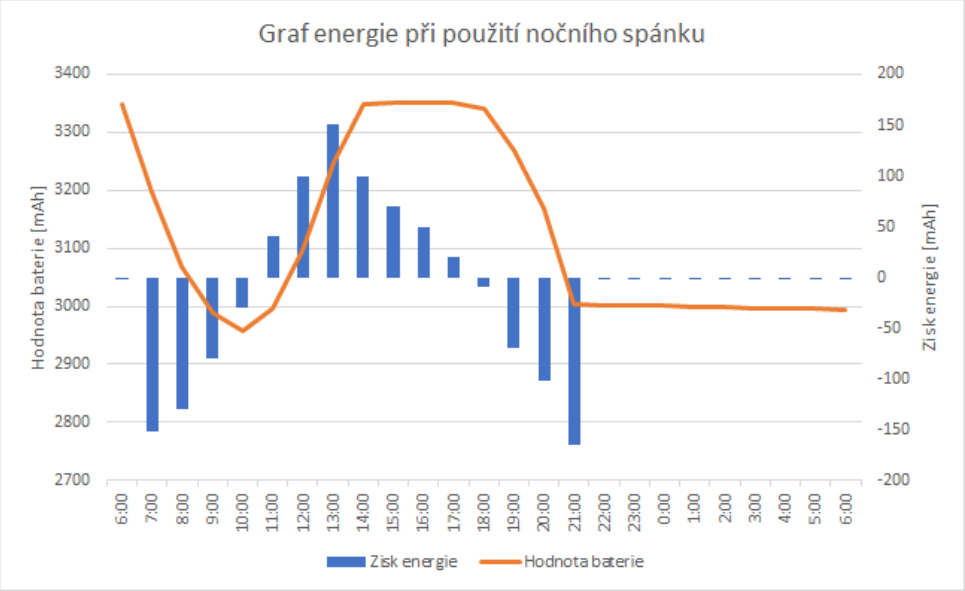


Obrázek 3.1: Graf vývoje hodnot bez optimalizace odběru

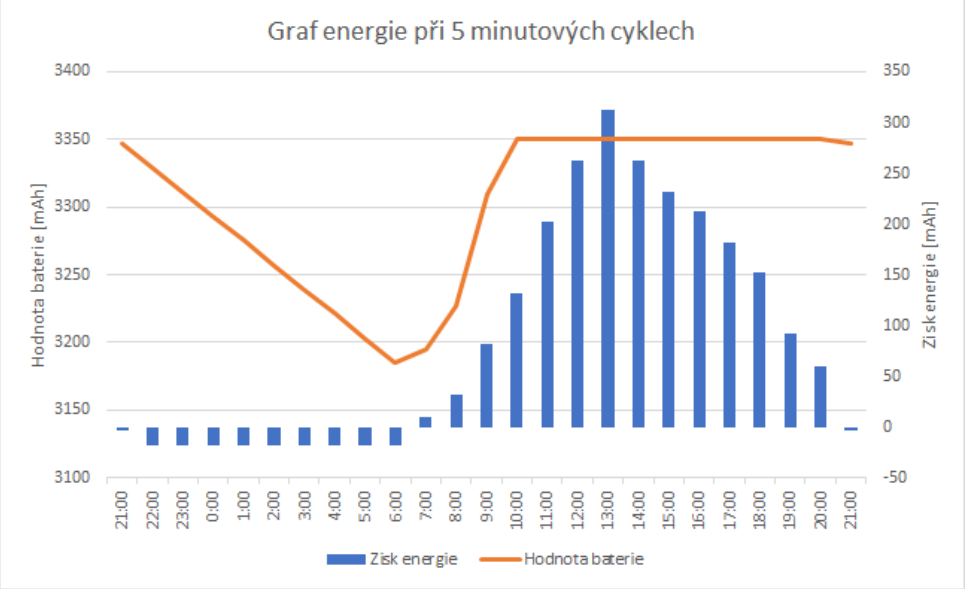
Snížení energetické náročnosti

Zde budou uvedeny možnosti snížení energetické náročnosti mikrokontrolérů. Jak již bylo zmíněno, problém tkví v potřebě práce v reálném čase. V případě využití v domácnosti lze počítat s částmi dne, kdy se zařízení vůbec využívat nebude. (Například, když jsou obyvatelé v práci, nebo když spí.) Znamenalo by to, že zařízení bude aktivní například jen 8 hodin denně namísto 24. Už toto je veliká úspora, která navíc natolik neovlivňuje interakci s uživatelem. Vývoj hodnot je v tomto případě vidět na grafu 3.2

Dále můžeme zmírnit požadavek na okamžitou odezvu. Místo, aby byly serverem odesílány požadavky, žaluzie by se v nějakých časových intervalech dotazovaly serveru na požadované hodnoty. Při tomto intervalu stanoveném na 5 minut pak vychází úspora oproti původnímu řešení 90%. V tomto případě je již možné dosáhnout soběstačnosti systému. Vývoj hodnot pro tuto variantu je viditelný na grafu 3.3



Obrázek 3.2: Graf vývoje hodnot při použití nočního spánku



Obrázek 3.3: Graf vývoje hodnot při použití 5 minutových cyklů

3.1.5 Programové požadavky

Základní myšlenkou programu je změna polohy a náklonu žaluzií. Požadavek na změnu bude přijat v MQTT zprávě. Uživatel bude mít možnost měnit pozici a náklon žaluzií a sledovat jejich stav. Než mu to bude umožněno, musí být zařízení nakonfigurováno. V některých částech bude využit identifikátor zařízení definovaný jako *esp–blinds–acb123*, kde *abc123* je identifikátor ESP chipu, neboli druhá polovina jeho MAC adresy. Dále se setkáme s termínem

spojovací kód (connection token), čímž je myšlen kód, který spojuje zařízení s uživatelem. Jak se tento kód získá bude ukázáno v dalších částech návrhu.

Konfigurace WiFi a MQTT. Než bude umožněno přijímat požadavky, je zapotřebí, aby se zařízení připojilo do místní WiFi sítě a následně inicializovalo spojení s MQTT zprostředkovatelem. K nastavení parametrů těchto spojení bude vytvořen konfigurační portál. Zpřístupnění portálu bude záviset na platnosti parametrů uložených v konfiguračním souboru. Pokud jsou neplatné, začne ESP vysílat svůj WiFi přístupový bod. Ve chvíli kdy se k němu uživatel připojí, přejde na konfigurační adresu, kde mu bude umožněno vyplnit: SSID, heslo, server a port MQTT zprostředkovatele, spojovací kód.

Připojení k WiFi a MQTT. Pokud existuje konfigurace, dojde k pokusu o připojení k WiFi síti. Po připojení k WiFi dojde k pokusu o připojení k MQTT serveru, kde se jako ID klienta a uživatelské jméno využije ID zařízení. Jako heslo bude použit spojovací kód. Při úspěšném spojení se zařízení přihlásí k odběru příkazů v tématu definovaném jako *devices/esp–blinds–abc123/command*. Při problému s připojením jak k WiFi síti tak k MQTT brokeru bude konfigurace vyresetována a opět se zpřístupní konfigurační portál.

Příjem a výkon příkazů. Každá zpráva bude ve formátu JSON a bude obsahovat akci předem definovaného typu, na jehož základě se rozhodne o vykonání akce. Některé typy zpráv odesílá server a některé zařízení. Využití tohoto formátu zpráv je přehledné a dá se jednoduše rozšířit o další typy. Definována je minimální sada typů:

1. Začni kalibraci
   - Zařízení vykoná pohyb servem po definovaný čas v jednom směru
2. Dokonči kalibraci
   - Zařízení využije hodnoty ve zprávě k definování své aktuální pozice a celkovému času potřebnému k překonání polohy shora dolů
   - Následně informuje o dokončení kalibrace (Akce 5)
3. Proveď změnu pozice
   - Zařízení provede změnu pozice
   - Po dokončení informuje o svém stavu (Akce 4)
4. Proveď změnu náklonu
   - Zařízení provede změnu náklonu
   - Po dokončení informuje o svém stavu (Akce 4)
5. Ohlášení stavu
   - Zařízení informuje o aktuální pozici a náklonu

- Tuto akci provádí zařízení. V případě, že ji přijme, je ignorována

6. Dokončení kalibrace
   - Zařízení informuje o dokončení kalibrace.
   - Tuto akci provádí zařízení. V případě, že ji přijme, je ignorována

Změna polohy a náklonu. Požadavek na změnu polohy či náklonu bude obsahovat procentuální požadovanou hodnotu. Ta se na základě hodnot získaných kalibrací převede na časový interval, po jehož dobu bude roztočeno servo. Podle aktuální a požadované hodnoty se také určí směr, kterým se servo bude točit. Při pohybu žaluzie nahoru a dolů bude zachován původní náklon.

Díky tomuto návrhu bude mít uživatel možnost využít modul řízení samostatně bez nutnosti připojit ho k serveru, který bude definován později v této práci. Potřebovat k tomu bude MQTT brokera a implementaci zpráv. Přijde tím ale připravenou možnost propojení s Google Home.

3.2 Serverová aplikace

Server bude prostředníkem pro komunikaci mezi žaluziemi, uživatelem a Google Home. Bude navržen jako SaaS, čili všichni uživatelé a jejich zařízení se budou připojovat k tomuto jednomu serveru.

3.2.1 Komunikace s žaluziemi

Metoda komunikace již byla zvolena v předchozí části, a to technologie MQTT. Existují dvě možnosti návrhu spojení mezi moduly a serverem. První možnost je taková, že MQTT broker bude oddělen, tedy server bude MQTT klient a bude se přihlašovat k odběru a publikovat zprávy stejně jako moduly. Druhá možnost spočívá v tom, že server bude MQTT broker, a tudíž všechny zprávy půjdou přes něj. Toto druhé řešení usnadní i identifikaci a ověření klientů, protože data o žaluziích budou na tomto serveru dostupná. Bude tedy využito řešení, kdy MQTT zprostředkovatel je součástí serverové aplikace.

Žaluzie při pokusu o navázání spojení odešlou také své uživatelské jméno a heslo. Uživatelské jméno bude pouze kopírovat klientské ID. Heslo bude spojovací kód. Server tyto údaje porovná s hodnotami v databázi a na základě toho povolí nebo odmítne spojení. V případě povolení označí toto zařízení jako dostupné, aby s ním mohl uživatel v grafickém rozhraní manipulovat.

3.2.2 Propojení s Google Home

Záměry definované v analýze jsou vlastně HTTP požadavky. Server tedy bude disponovat HTTP serverem pro jejich obsluhu. Budou k tomu dostačující tři koncové body, z toho jeden bude pro vykonávání záměrů a dva budou autorizační:

1. **/smarthome**
   - příjem všech záměrů
   - požadavek typu POST, tělo ve formátu JSON
   - lze využít Googlem dodávaného SDK pro různé jazyky
2. **/oauth2/auth**
   - 1. část OAuth autorizace
   - uživatel se v grafickém prostředí přihlásí
   - proběhne přesměrování na definovanou adresu s autorizačním kódem
3. **/oauth2/token**
   - 2. část OAuth autorizace
   - Google požádá o výměnu autorizačního kódu za přístupový token
   - bude mu vydán přístupový a obnovovací token

3.2.3 Příjem požadavků z klientské aplikace

Pro obsluhu požadavků z webové aplikace bude navržena REST API umožňující práci s uživatelovými zařízeními. Všechny požadavky musí obsahovat uživatelův přístupový token. Zařízení odkazované v URL již musí být v databázi a musí být propojení s uživatelem, který odesílá požadavek. Toto neplatí pro požadavek o přidání nového zařízení.

4. **/devices/*id*/setup/connection**
   - žádost o přidání nového zařízení
   - zařízení bude přidáno do databáze
   - v odpovědi bude odeslán spojovací token
5. **/devices/*id*/setup/start**
   - inicializace kalibrační sekvence
6. **/devices/*id*/setup/continue**
   - příjem naměřených parametrů
   - odeslání hodnot do zařízení
7. **/devices/*id*/position**
   - požadavek na změnu pozice
   - v těle bude požadovaná procentuální hodnota
8. **/devices/*id*/tilt**
   - požadavek na změnu náklonu
   - v těle bude požadovaná procentuální hodnota

3.3 Klientská aplikace

Jak bylo zmíněno výše, konfigurace připojení bude probíhat přímo na zařízení. Ke kalibraci žaluzií a jejich ovládání je ale potřeba navrhnout rozhraní, ve kterém to uživateli bude umožněno. Komunikace s žaluziemi bude probíhat skrze server pomocí HTTP požadavků. Vzhledem k tomuto faktu není třeba vyvíjet nativní mobilní aplikaci, ale plně dostačující bude aplikace webová. Aplikace bude umožňovat následující:

1. Přihlášení uživatele
   - uživatel se bude moci přihlásit svým Google účtem
2. Přidání zařízení
   - uživateli bude umožněno vytvořit spojovací kód pro jeho žaluzie
3. Kalibrace zařízení
   - uživatel bude moci spustit kalibrační sekvenci a následně zadat naměřené hodnoty
4. Sledování stavu zařízení
   - uživatel uvidí všechna svá zařízení a jejich stavy
   - stavy jsou: dostupnost zařízení, aktuální poloha a náklon
5. Ovládání zařízení
   - uživatel bude moci změnit polohu a náklon jednotlivých zařízení

Kapitola 4
Implementace a vyhodnocení

V této kapitole bude popsán postup implementace všech tří částí. Byl kladen důraz na rozšiřitelnost kódu o další funkcionalitu a použití moderních běžně používaných technologií. Vyvinutý systém byl testován na sestrojeném modelu žaluzií. Výsledky budou porovnány s parametry existujících řešení.

4.1 Implementace SW modulu žaluzií

Program pro ESP8266 byl vyvíjen v prostředí Arduino, které umožňuje rychlou a nenáročnou práci s vývojovými deskami a zjednodušuje některé aspekty jazyka C++. Program se v Arduinu dělí na dvě hlavní složky/funkce: funkce *setup*, ve které se inicializují součásti potřebné k běhu programu, a funkce *loop*, která se vykonává neustále dokola.

4.1.1 Inicializace

Prvně jde o načtení konfigurace ze souboru, vytvoření přístupového bodu v případě, že konfigurace není validní, nebo připojení k místní WiFi sití, pokud validní je. V případě přístupového bodu se také inicializuje DNS server, který všechny požadavky směřuje na HTTP server, který poskytuje konfigurační rozhraní. V něm lze zadat údaje k WiFi síti, spojovací token a případně změnit adresu MQTT zprostředkovatele. Toho je dosaženo vužitím knihovny WiFiManager od GitHub uživatele Tzapu [26]. Po ukončení konfigurace je proveden pokus o připojení k zadané sítí. V případě úspěchu se dále pomocí spojovacího tokenu naváže spojení s MQTT brokerem a přihlášení k odběru zpráv z tématu *devices/esp-blinds-id/command*. Toho je docíleno pomocí knihovny *PubSubClient* od GitHub uživatele Knolleary [27]. Pokud dojde k chybě, vyresetuje se konfigurace a je opět otevřen přístupový bod. Jestliže nenastanou problémy, přechází se do druhé části programu.

4.1.2 Smyčka

Ve smyčce programu se nejprve zkontroluje, zda má zařízení stále aktivní MQTT spojení, a v negativním případě dojde k pokusu o obnovení. Následně dojde k přijmu zpráv a vyhodnocení obsahujících akcí, které jsou:

1. Začátek kalibrace - provedení pohybu žaluzií
2. Dokončení kalibrace - uložení hodnot
3. Požadavek na změnu polohy - uložení požadované pozice
4. Požadavek na změnu náklonu - uložení požadovaného náklonu

Započnutím kalibrace se na 3 vteřiny spustí servo motor v jednom směru. V dokončení by se měly nacházet tři hodnoty: *wentDistance* – o jakou vzdálenost se žaluzie posunuly, *totalDistance* – jak jsou žaluzie vysoké, *wentUp* – zda-li jela žaluzie nahoru. Pomocí těchto hodnot je vyčítána celková časová vzdálenost žaluzie, tedy jak dlouho musí být servo sepnuto, aby se žaluzie dostala z 0 na 100%, a také aktuální časová poloha, tedy kde se právě nachází.

Sérií pokusů bylo zjištěno, že k náklonu lamel z 0 na 100% je potřeba sepnutí po dobu 600ms. Tato hodnota je tedy přímo vložená do programu a při měnění náklonu je s ní počítáno. Při pohybu žaluzie nahoru respektive dolů se nejprve provede náklon do nejnižší resp. nejvyšší hodnoty a až poté se vykonává časová změna polohy. Po dokončení je proveden návrat do původní hodnoty náklonu.

Příklad. Žaluzie je ve 20% výšky a 50% náklonu (vodorovně). Změní se požadovaná hodnota výšky na 60%. Nejprve dojde k náklonu do 0% (300ms ve směru nahoru), následně samotný pohyb (40% celkové časové vzdálenosti) a nakonec návrat do 50% náklonu (300ms ve směru dolů).

Po dokončení pohybu je odeslána stavová zpráva do tématu *devices/esp-blinds-id/state*, která obsahuje:

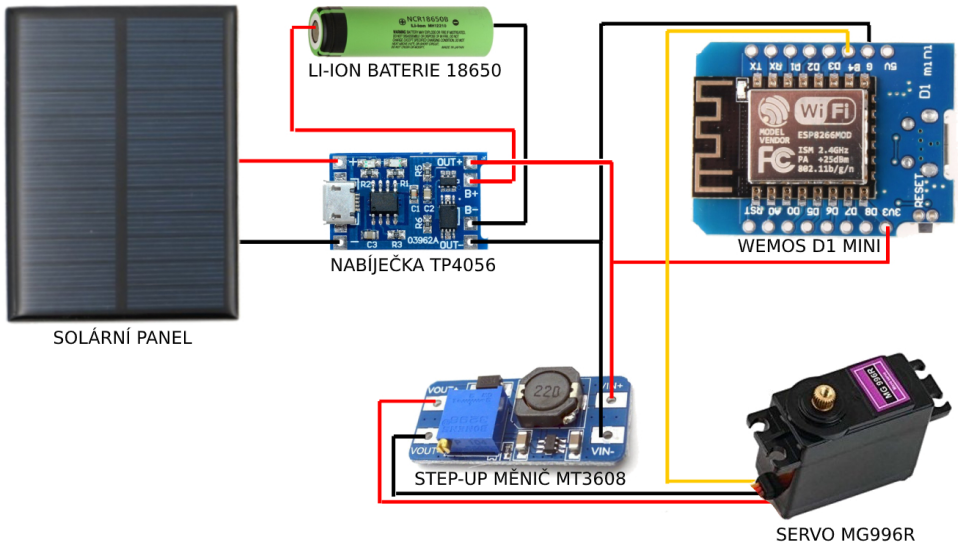
- Typ akce = oznámení stavu
- Aktuální pozice [%]
- Aktuální náklon [%]
- Hodnota vnitřního napětí [mV]

4.2 Zapojení HW modulu řízení

K sestavení hardwarového demonstračního modelu byly využity běžně dostupné komponenty zmíněné v návrhu. Jejich zapojení lze nalézt na obrázku 4.1. Byl využit solární panel velikosti 6x11cm a později také 11x14cm. Oba mají uváděné napětí 6V, přičemž na přímém slunci dosahovaly hodnoty i 7.2V a naopak při zatažené obloze byly hodnoty kolem 4V. Energie je ukládána do Li-Ion baterie 18650. Konkrétně byl využit článek Panasonic NCR18650B s udávanou kapacitou 3350mAh a také dva paralelně zapojené zrecyklované články bez bližších detailů.

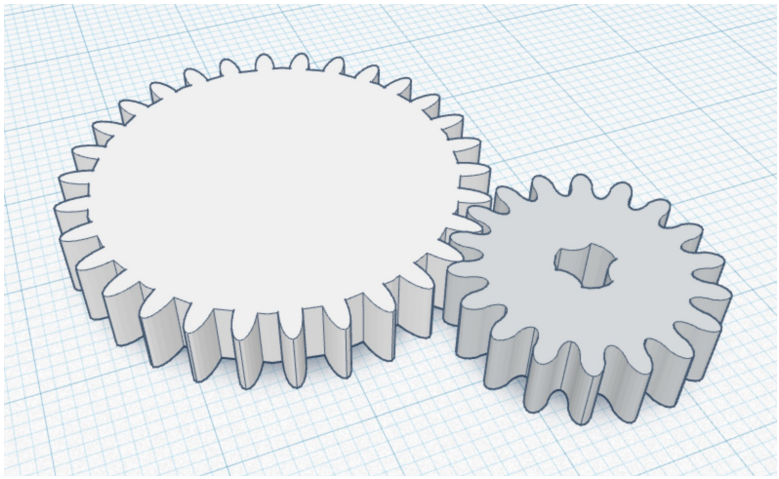
Nabíjení je podporováno 5V mobilním adaptérem a je regulováno obvodem s chipem TP4056, který mimo jiné disponuje i ochranou baterie, která ji chrání proti vybití (<2.4V), přepětí (>4.2V), nadproudu (>3A) a zkratu.

Výstupní napětí baterie se pohybuje od 4.2V při plném nabití, až k 2.4V při zdravém vybití. Desku s ESP lze tedy zapojit přímo, jelikož Wemos D1 mini i NodeMCU disponují regulátorem, který převádí vstupní napětí na 3.3V. Servo MG996 pracuje s napětím 4.8V až 6V, kdy platí – vyšší napětí = vyšší rychlost. Proto je využit step-up boost měnič MT3608, který napětí z baterie konvertuje na 6V, pro dosažení co nejvyšší rychlosti serva.



Obrázek 4.1: Blokové schéma zapojení

Pro napojení serva na osu žaluzie byla vymodelována soustava dvou ozubených kol, viz obrázek 4.2. První má průměr 20mm přímo se nasune na osu žaluzie. Druhé má průměr 32mm a k servu se připevní pomocí dodávaného adaptéru. Modely byly navrženy v online 3D modelovacím programu Tinkercad od společnosti Autodesk. Poté byly vytištěny na 3D tiskárně a kvůli nepřesnostem při měření osy žaluzie musely být upraveny pilníkem. Musel být také proveden permanentní zásah do konzole žaluzie. V místě napojení serva na osu byl vyříznut otvor velikosti přibližně 40x6mm na ozubené kolo. Servo je aktuálně ke konzoli přiděláno velkým množstvím lepící pásky, jak je viditelné na obrázku 4.3. Při dalším vývoji by mohl být navržen kryt, který by obsahoval všechny komponenty a jen by se na konzoli přicvakl, případně přišrouboval.



Obrázek 4.2: 3D modely ozubených kol

Obrázek 4.3: Demonstrační model (detail lepící pásky držící vše pohromadě)

4.3 Serverová aplikace

Pro implementaci serveru byla vybrána platforma NodeJS. Sestává z HTTP serveru a MQTT zprostředkovatele, kdy každý běží na jiném portu, ale jsou součástí jednoho kódu. K implementaci Google Home musel být vytvořen projekt na Google Cloud platformě. Konfigurace Google Home probíhá na Google Actions, kde se zadává URL pro zpracování záměrů *smarthome* a URL spojování účtů, tedy autorizační koncové body *auth* a *token*. Zde se také setkáváme s problémem, který při implementaci nastal. Tyto URL musí být na zabezpečeném protokolu HTTPS s certifikátem s ověřenou autoritou. Server ale při vývoji běžel lokálně na autorově PC.

Tento problém byl vyřešen softwarem *ngrok*, který vytvoří zabezpečený

tunel mezi internetem a lokálním prostředím. Ke koncovým bodům se tak lze dostat pomocí vygenerované URL adresy, která vypadá například takto: *https://651d50ef.ngrok.io/*. Tento nástroj také nabízí prostředí, ve kterém lze vidět příchozí požadavky včetně jejich obsahu a následně i odpovědi, což se velice hodilo pro ladění chyb.

Když už byl vytvořen Google Cloud projekt, rozhodl se autor využít i další Googlem nabízený produkt. Google Firebase je sada nástrojů pro vývoj a běh aplikací. Konkrétní využité nástroje byly Firebase Authentication zprostředkovávající autentizaci a Firebase Firestore – NoSQL dokumentová databáze. Tyto nástroje mají výhodu v jednoduché implementaci pomocí dodávaného SDK. Pro serverovou část je to Firebase Admin SDK zahrnující všechny nástroje v jednom. Konfigurace v kódu probíhá pomocí servisního účtu, který si lze vygenerovat ve webovém prostředí nazývaném Firebase Console, kde lze zároveň nahlížet do registru uživatelů a dat ve Firestoru. Zpočátku bylo záměrem využít další nástroj Firebase Functions, který umožňuje nasazení NodeJS kódu ve formě funkcí, čímž se myslí jeden koncový bod. Toto řešení ale bylo zamítnuto z důvodu nemožnosti nasadit MQTT server.

4.3.1 Uložiště

Jak bylo zmíněno výše, pro ukládání dat je využita dokumentová databáze Firestore. Dokumenty jsou podobné řádkům v SQL databázích, obsahují typované proměnné a jejich hodnoty. Každý dokument může mít jakoukoli strukturu. Dokumenty jsou sdružovány do kolekcí. Pro přístup k datům se určují práva pomocí tzv. pravidel, která definují, jaké podmínky je nutné splnit pro čtení/zápis dat do kolekcí/dokumentů. Tato pravidla neplatí pro administrátora, kterým se stáváme použitím Firebase Admin SDK. Důležitost pravidel a další výhody budou vysvětleny u webové aplikace.

V databázi se aktuálně nachází 5 kolekcí:

1. **auth_code**
   - autorizační kódy pro Google Home
   - dokumenty obsahují položky: ID klienta, ID uživatele, platnost, čas vytvoření
2. **auth_request**
   - autorizační požadavky, vytvořené přístupem na */oauth2/auth*
   - dokumenty obsahují položky: ID klienta, platnost, čas vytvoření, adresu pro přesměrování, typ odpovědi, stav
3. **client**
   - informace o klientech, kteří mohou využít implementovanou OAuth2
   - dokumenty obsahují položky: ID a tajný klíč klienta, povolené adresy pro přesměrování, povolené granty

- aktuálně se v kolekci nachází jediný záznam pro Google Home, jednoduše ale lze rozšířit o dalšího klienta

4. **refresh_token**
   - vygenerované obnovovací tokeny
   - dokumenty obsahují položky: ID klienta, ID uživatele, platnost, čas vytvoření
5. **devices**
   - kolekce jednotlivých zařízení
   - obsahují konfigurační položky: spojovací kód, ID uživatele, ID zařízení, název,
   - a také stavové hodnoty: připojeno, pozice, náklon, v pohybu

HTTP server. HTTP server je implementován pomocí webového frameworku Express, který se stará o obsluhu koncových bodů. Cesty jsou definovány dle návrhu. Na cestu */smarthome* je připojen autorizační middleware, který kontroluje přítomnost a platnost přístupového tokenu vydaného tímto serverem. Podobně tomu je u cest začínajících */devices*, kde je ale ověřování implementováno pomocí Firebase Authentication, protože token je vydáván uživateli Googlem.

MQTT server. MQTT server využívá knihovnu Mosca, která poskytuje kompletní zprostředkování MQTT prostředí. Knihovna umožňuje vlastní implementaci ověřování klientů a autorizaci odběrů a publikování do témat. Toho je využito pro autorizaci zápisu do tématu *devices/esp-blinds-abc123/state*, kam může zapisovat pouze zařízení, které se dříve správně autentizovalo pomocí svého ID a spojovacího kódu. Do tématu *devices/esp-blinds-abc123/command* může zapisovat pouze server. Na obrázku 4.4 je ukázka kódu autentizace MQTT klienta.

Pro přehled, co se na serveru děje, byl napsán jednoduchý logovací systém. Využívá komunikační nástroj Slack, do kterého pomocí webhooků odesílá informace o připojení/odpojení zařízení, přijmu a výkonu HTTP požadavku (z webové aplikace i z Google Home) a MQTT zprávy.

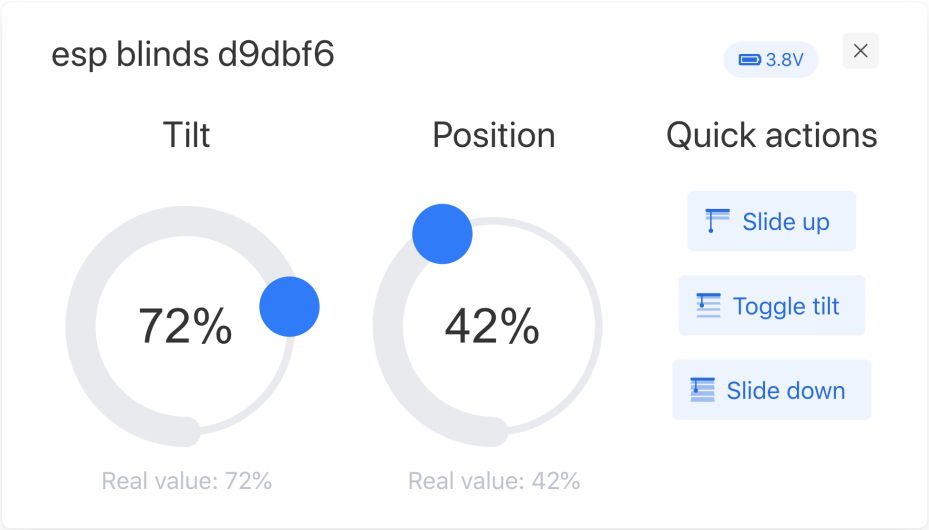
4.4 Webová aplikace

Webová aplikace je naprogramována v javascriptu s použitím knihovny React. Uživatel se do aplikace přihlašuje svým Google účtem, k čemuž byl opět využit Firebase SDK. Po přihlášení jsou z databáze načtena uživatelova zařízení. Zde se dostáváme k využití databázových pravidel, která dovolují zobrazit uživateli pouze ta zařízení, jichž je majitelem. Další velká výhoda Firestoru je možnost přistupovat k datům přímo pomocí javascriptové knihovny, tedy bez serveru. Navíc lze přijímat aktualizace dat v reálném čase, což uživateli umožňuje vidět aktuální stav žaluzií a jejich hodnot.

```
const authenticate =
  async (client, username, password, callback) => {
  password = password.toString()
  if(testDeviceId(client.id)) {
    const device =
      await Firestore.collection('devices').doc(client.id).get()
    if(device.exists) {
      const {connection_token} = device.data()
      if(connection_token === password.toString()) {
        return callback(null, true)
      }
    }
  }
  callback(null, false)
}
```

Obrázek 4.4: Funkce pro autentizaci MQTT klienta

Podle hodnot je určeno, jak zařízení zobrazit. Pokud zařízení není nakalibrováno, je zobrazena možnost započít kalibraci. V opačném případě se vykreslí ovládání prvky zařízení, viz obrázek 4.5. Pokud je ovšem zařízení nedostupné, nebo je právě v pohybu, je ovládání znemožněno a přes ovládací panel je vykreslena zpráva popisující tuto událost. Požadavky na změnu pozice či náklonu jsou odesílány serveru pomocí knihovny Axios, která se stará o odeslání HTTP požadavků s daty ve správném formátu (JSON). Ke stylizaci byl využit open-source CSS framework Bulma, který zajišťuje responzivní design, takže se aplikace dobře používá na mobilu, tabletu i počítači.



Obrázek 4.5: Ovládací prvky žaluzií

Pro správu stavu a dat aplikace byla zvolena knihovna Redux, v javascriptových webových aplikacích velice rozšířená technologie. Data jsou uložena v uložišti (store). Změny jsou působeny odesíláním akcí (actions), které jsou zpracovávány reduktory (reducers). Rozšířením je technologie Redux Saga, která využívá javascriptových generátorů k zpracovávání asynchronních úloh. Přímo k asynchronnímu získávání dat z Firestoru je využita knihovna *redux-saga-firebase*, která, jak název napovídá, využívá technologie Redux Saga k práci nejen s Firestorem, ale i s dalšími Firebase produkty, například Authentication.

Na obrázku 4.6 je ukázka synchronizace dat z Firestoru. Jde o cyklus sestávající z akce úspěšného přihlášení uživatele (řádek 3), přihlášení k synchronizaci uživatelových zařízení (řádek 5), akce odhlášení uživatele (řádek 13) a následné zrušení synchronizace spolu s vymazáním aktuálních dat z paměti (řádek 14).

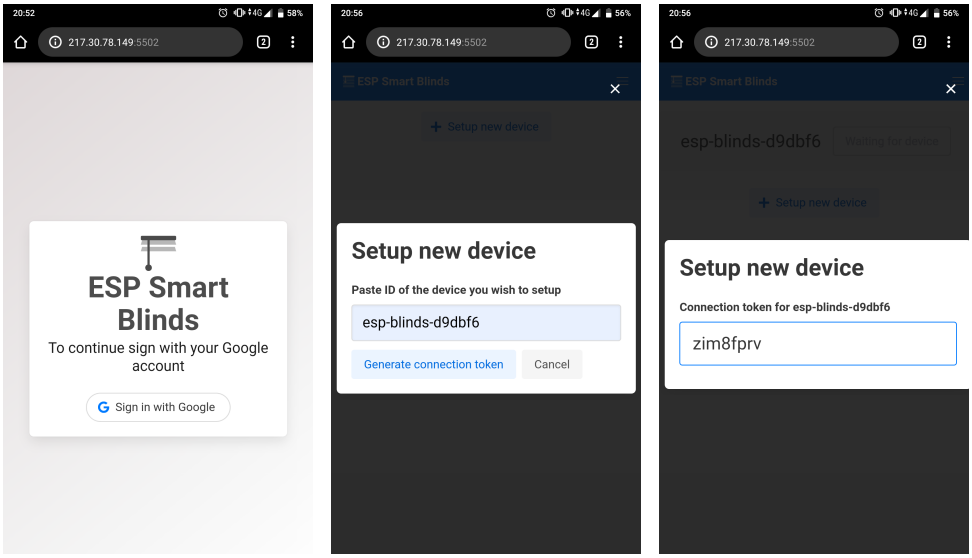
```
1  function* syncDevicesSaga() {
2    while(true) {
3      yield take(types.LOGIN.SUCCESS)
4      let user = yield select(state => state.auth.user)
5      const syncTask = yield fork(
6        rsf.firestore.syncCollection,
7        devicesColRef(user.uid),
8        {
9          successActionCreator: syncDevices,
10         transform: devicesTransformer,
11       }
12     )
13     yield take(types.LOGOUT.SUCCESS)
14     yield all([
15       cancel(syncTask),
16       put(clearDevices())
17     ])
18   }
19 }
```

Obrázek 4.6: Redux Saga změny náklonu

4.5 Kalibrace a propojení s Google Home

V této části bude ukázán postup nastavení zařízení, kalibrace a také propojení s Google Home pomocí implementovaného systému. Nejprve je potřeba, aby si uživatel pro dané zařízení vytvořil spojovací kód. To provede ve webové aplikaci, která je v době implementace dostupná na adrese *217.30.78.149:5502*. Uživatel se přihlásí pomocí svého Google účtu, který používá v Google Home. (Obrázek 4.7a) Po přihlášení se objeví možnost přidat nové zařízení, kde po

kliknutí doplní identifikátor zařízení. (Obrázek 4.7b) Tlačítkem *Generate new token* se odešle požadavek na server, který vrátí spojovací kód, a ten je uživateli zobrazen. (Obrázek 4.7c)

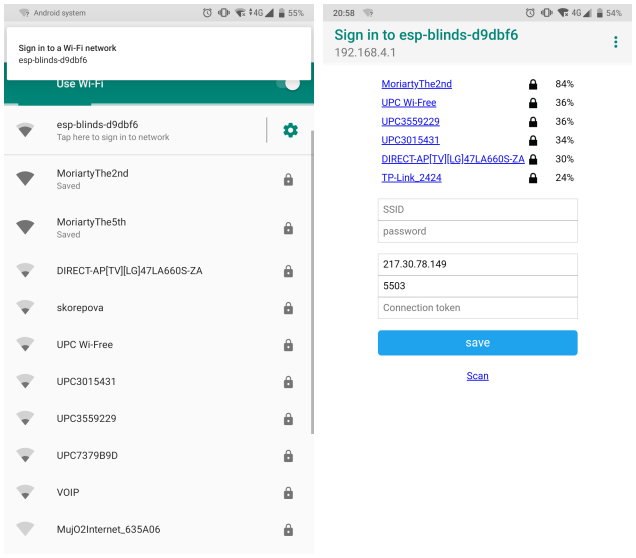


(a) : Přihlášení **(b) :** Doplnění identifikátoru **(c) :** Vytvořený spojovací kód

Obrázek 4.7: Přidání nového zařízení ve webové aplikace

Nyní je potřeba nastavit samotné zařízení. K tomu je třeba připojit se k zařízením vysílanému přístupovému bodu. (Obrázek 4.8a) Uživateli bude nabídnuto přihlášení k síti WiFi, kde ale ve skutečnosti bude nastavovat údaje k jeho lokální WiFi síti a spojovací kód, který si vygeneroval ve webové aplikaci. (Obrázek 4.8b) Na obrázku jsou také vidět další dva údaje. Jedná se adresu a port k MQTT serveru. Možnost změnit je je ponechána pro případ, kdy by se změnil produkční server, nebo by uživatel chtěl propojit žaluzie s jeho vlastním systémem.

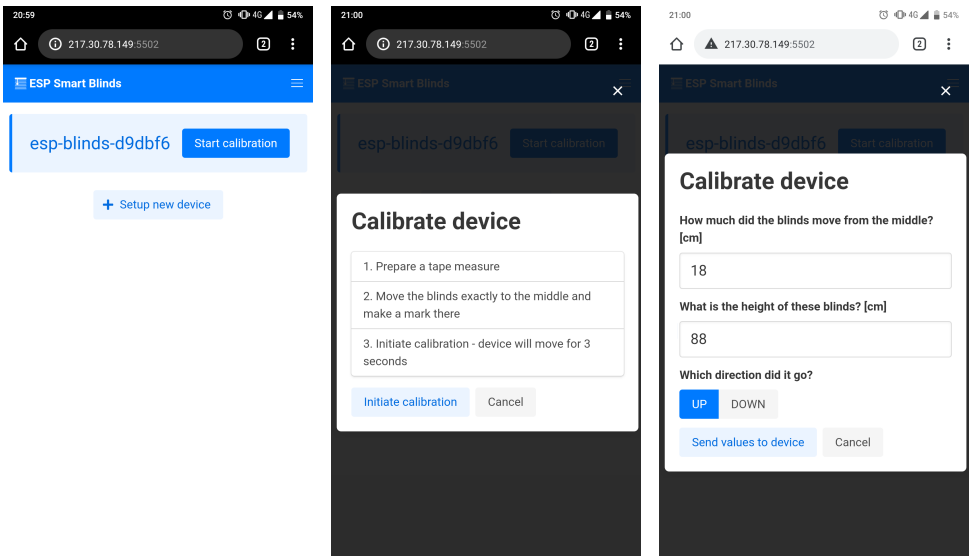
Po uložení údajů se zařízení pokusí přihlásit k WiFi sití a navázat MQTT spojení se zprostředkovatelem. Pokud vše proběhne v pořádku, objeví se zařízení ve webové aplikaci s možností kalibrovat. (Obrázek 4.9a) Po kliknutí na tlačítko *Start calibration* se zobrazí informace o kalibraci a tlačítko na započetí kalibrace. (Obrázek 4.9b) Zařízení provede 3 vteřinový pohyb a uživatel je tázán k zadání naměřených údajů, po jejichž zadání je odešle, a tím dokončí kalibraci. (Obrázek 4.9c)



(a) : Připojení k přístupovému bodu

(b) : Vyplnění WiFi údajů a spojovacího kódu

Obrázek 4.8: Nastavení modulu žaluzie



(a) : Zařízení připravené ke kalibraci

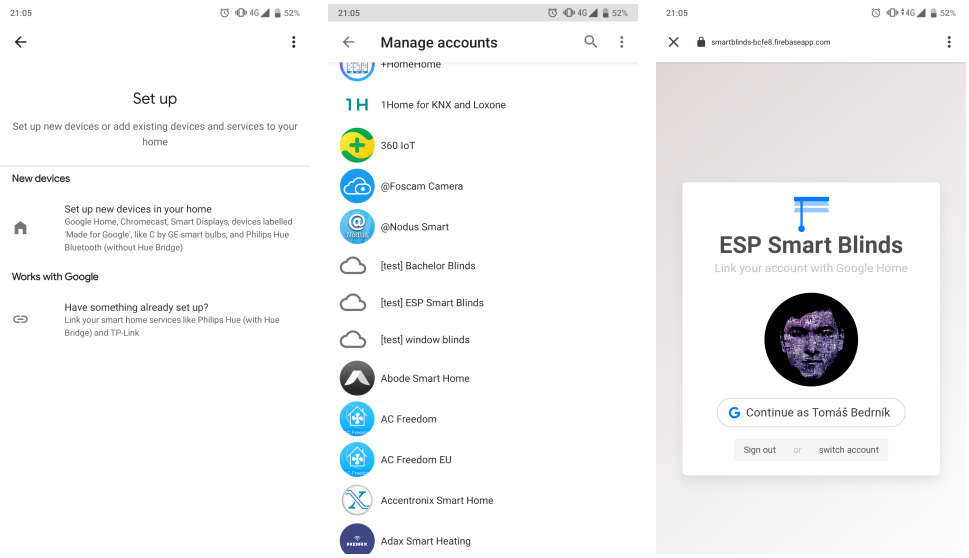
(b) : Instrukce ke kalibraci

(c) : Zadání hodnot kalibrace

Obrázek 4.9: Kalibrace zařízení pomocí webové aplikace

Posledním krokem je propojení s Google Home. Ve skutečnosti může být tento krok proveden kdykoli, ale v tomto stavu uvidí uživatel rovnou nějaké zařízení. V aplikaci Google Home se uživatel standardním způsobem prokliká do pohledu *Set up* (Obrázek 4.10a), kde zvolí propojení existující služby (Obrázek 4.10b, všimněte si položky *[test] ESP Smart Blinds*) a následně se přihlásí svým Google účtem, čímž provede spojení. Ihned poté dojde

k synchronizaci zařízení a uživatelovy žaluzie se objeví na hlavní stránce aplikace Google Home, viz Obrázek 4.11.

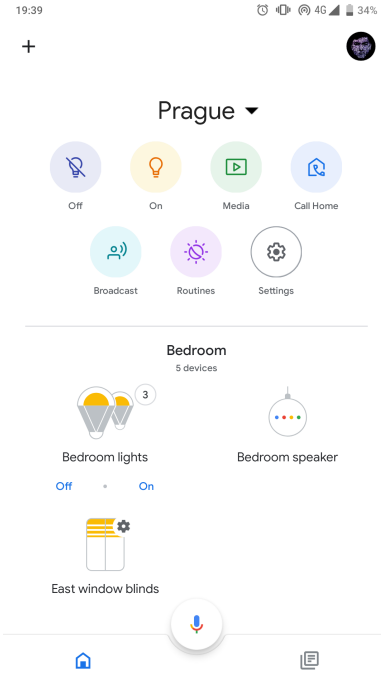


(a) : Pohled Set up

(b) : Seznam dostupných služeb včetně *ESP Smart Blinds*

(c) : Přihlašovací obrazovka OAuth

Obrázek 4.10: Propojení s Google Home 1. část



Obrázek 4.11: Hlavní obrazovka aplikace Google Home

4.6 Požadavek na změnu polohy/náklonu

V této části bude popsán postup zpracování požadavku na změnu polohy či náklonu pomocí webové aplikace.

1. Příjem požadavku na */devices/esp-blinds-abcdef/position* s tělem `{"position": 20}`
2. Middleware zkontroluje pravost a platnost přístupového tokenu a existenci, dostupnost a majitele zařízení
3. V databázi se u tohoto zařízení nastaví hodnota *running* na *true*. Tato informace se poté zpropaguje do webové aplikace jako změna hodnoty a uživateli je zamezeno provádět další požadavky na toto zařízeno.
4. MQTT broker publikuje do tématu *devices/esp-blinds-abcdef/command* zprávu s tělem `{"type": 2, "position": 20}`. Číslo 2 znamená typ akce "změna polohy".
5. Po odeslání MQTT zprávy je odpovězeno na HTTP požadavek stavovým kódem 200
6. Modul žaluzií přijme zprávu a začne vykonávat změnu polohy. Po dokončení publikuje do tématu *devices/esp-blinds-abcdef/state* zprávu s tělem `{"type": 4, "position": 20, "tilt": 50}`. Číslo 4 znamená typ akce "informace o stavu".
7. MQTT broker přijme tuto zprávu a nastaví v databázi aktualizuje hodnoty tohoto zařízení + změní hodnotu *running* na *false*. Tato změna je opět propagována do webové aplikace a uživateli je umožněno provézt další akci.
8. Aktualizovaný stav je třeba také sdělit Googlu požadavkem ReportState.

Podobný proces se odehrává při záměru EXECUTE. V tom se může měnit hodnota více zařízení naráz. Pro celistvost bude uveden celý proces, i když jsou některé kroky shodné.

1. Příjem požadavku na */smarthome* s tělem obsahujícím záměr EXECUTE
2. Middleware zkontroluje pravost a platnost přístupového tokenu.
3. Z databáze se načtou požadovaná zařízení a rozdělí se do skupin:
   - OFFLINE - zařízení není dostupné
   - RUNNING - zařízení je právě v pohybu
   - SETUP - zařízení není kalibrováno
   - ERROR - zařízení neumí tuto akci vykonat
   - PENDING - zařízení vykonává akci

4. Nenulové skupiny jsou odeslány jako odpověď na požadavek
5. V databázi se u zařízení ve skupině PENDING nastaví hodnota *running* na *true*.
6. MQTT broker publikuje do témat všech zařízení ze skupiny PENDING akce na změnu polohy/náklonu.
7. Dále vše pokračuje stejně jako v předchozím procesu od kroku 6

4.7 Porovnání parametrů s komerčními produkty

Pro otestování byl sestrojen demonstrační model okna s žaluziemi o rozměrech 90cm na výšku a 45cm na šířku. Doba potřebná k změně pozice z 0 na 100% je 15 vteřin. Tato hodnota se bude měnit lineárně s rostoucí/klesající výškou žaluzie. Ovládání pomocí hlasových povelů přes Google asistenta funguje spolehlivě. Doba od přijmu hlasového příkazu do začátku pohybu žaluzie je menší než 2 vteřiny.

Vzhledem k neimplementovanému uspávání mikrokontroléru je výdrž baterie necelé dva dny a je proto využit externí 5V adaptér. Dle výpočtů by bylo dosaženo výdrže jeden měsíc na jednu baterii při implementaci spánku 8 hodin denně a celé řešení by mělo být energeticky soběstačné při metodě zpracování požadavků jednou za 5 minut.

V tabulce 4.1 je uveden seznam komponent použitých v modelu včetně ceny. Při vynechání solárního panelu a baterie, která aktuálně slouží spíše jako záložní zdroj, by došlo k snížení ceny na přibližně 400Kč. Naopak přidáním krytu vytištěného 3D tiskárnou by se cena mohla zvýšit o cca 100Kč. Celkově se tedy pořizovací cena finálního produktu pohybuje okolo jednoho tisíce korun s baterií, nebo pěti set bez baterie. Systém může být také rozšiřován o další baterie a solární panely dle potřeby uživatele.

Porovnání parametrů s komerčními produkty je provedeno v tabulce 4.2. Srovnány jsou způsoby připojení, ovládání a cena. V tomto hledisku si navržený výrobek stojí velice dobře. V čem vlastní řešení zaostává je složitost instalace. Konkurenční výrobky vyžadují k instalaci pouze nasazení na kuličkový řetízek, kdežto navržené řešení potřebuje výrobu otvoru do profilu žaluzie.

Označení	Komponenta	Cena
Wemos D1 mini	Vývojová deska s ESP8266	148 Kč
TP4056	Nabíječka Li-ion článku s ochranou	25 Kč
NCR18650B	Li-ion baterie Panasonic, 3350mAh	208 Kč
MT3608	Step-up boost měnič	28 Kč
MG996	Servo s kovovými převody	188 Kč
-	Solární panel 6V 2W	278 Kč
Celková cena		875 Kč

Tabulka 4.1: Seznam komponent HW modelu vč. cen

Produkt	Připojení	Google Home	Výdrž baterie	Cena
Vlastní řešení	WiFi	Ano	1.5 dne	875 Kč
SH-RB230	Bluetooth	Ne	5 hodin	770 Kč
YH002	433MHz/WiFi	Ano	neuvedeno	1667 Kč
Tilt MSB Kit	Bluetooth	Ne	3-4 měsíce	3280 Kč
+ Bridge	WiFi	Ano	-	+2280 Kč

Tabulka 4.2: Parametry zařízení

Kapitola 5

Závěr

Cílem teoretické části této práce bylo seznámit se s problémem motorizace žaluzií, komunikace chytrých zařízení a implementace systému kompatibilního s ekosystémem Google Home. Část byla také věnována analýze trhu s motorizovanými žaluziemi. V praktické části bylo cílem navrhnout přídavný modul k existujícím žaluziím zajišťující motorizaci pohybu a možnost ovládání skrze Google Home.

První kapitola byl jednoduchý úvod do smart zařízení a seznámení s prací.

V druhé kapitole došlo k seznámení s žaluziemi, možnostmi motorizace, mikrokontroléry a jejich komunikace. Velká část byla věnována výzkumu implementace pro Google Home včetně zabezpečení. Dále jsme se podívali na téma napájení IoT zařízení s využitím sluneční energie. Na konci této kapitoly byly představeny na trhu dostupné výrobky.

V kapitole číslo 3 byl rozebrán návrh vlastního řešení sestávajícího ze tří částí. První část byla věnována samotnému modulu řízení, který využívá chip ESP8266, servo motor k otáčení osou žaluzie a solární panel k napájení. V části druhé byl navržen NodeJS server zajišťující kompatibilitu s Google Home a obsluhu webové aplikace. Ta byla představena ve třetí části a umožňuje uživateli správu zařízení včetně jejich kalibrace a ovládání.

Ve čtvrté kapitole byl popsán postup samotné implementace všech tří součástí. Byly vyčteny použité technologie a principy jednotlivých programů. Také byl ukázán uživatelský postup přidání zařízení do aplikace a propojení s Google Home. Nakonec byly představeny dosažené parametry sestrojeného modely a došlo k porovnání s komerčními produkty.

Výsledkem práce je funkční model žaluzií sestavený z běžně dostupných komponent. Po nastavení pomocí webové aplikace je ovladatelný hlasem přes Google Home. Nebyl implementován systém snižující energetickou náročnost. Modul tak lze napájet z jedné baterie pouze jeden až dva dny. V případě rozšíření o tuto funkcionalitu by byl model soběstačný ze solární energie.

Literatura

[1] E-shop Alibaba *Sweethome SH-RB230* [online]. [cit. 2020-1-3]. Dostupné z: https://www.alibaba.com/product-detail/Build-in-Bluetooth-and-APP-Control_62445005308.html

[2] Oficiální web *YiHao Blinds NYHBC002* [online]. [cit. 2020-1-2]. Dostupné z: http://www.yhblinds.com/en/products.php?tid=4

[3] E-shop Alibaba *YiHao Blinds NYHBC002* [online]. [cit. 2020-1-3]. Dostupné z: https://yhblinds.en.alibaba.com/product/60768947527-815537182/KECO_standard_roller_blind_tubular_motor_with_fast_speed_and_budget_offer.html

[4] Oficiální web *YiHao Blinds YH002* [online]. [cit. 2020-1-2]. Dostupné z: http://www.yhblinds.com/en/products.php?tid=2

[5] E-shop Alibaba *YiHao Blinds NYHBC002* [online]. [cit. 2020-1-3]. Dostupné z: https://www.alibaba.com/product-detail/2019-new-China-produce-smart-chain_60825420023.html

[6] Oficiální web *Tilt MySmartBlinds Automation Kit* [online]. [cit. 2020-1-2]. Dostupné z: https://www.tiltsmarthome.com/products/mysmartblinds-automation-kit

[7] Oficiální web *Tilt Smart Hub* [online]. [cit. 2020-1-2]. Dostupné z: https://www.tiltsmarthome.com/products/smart-hub

[8] E-shop Amazon *tilt's Blinds Automation Kit* [online]. [cit. 2020-5-8]. Dostupné z: https://www.amazon.com/MySmartBlinds-Automation-automated-Compatible-Assistant/dp/B078T1X2HM/

[9] E-shop Žaluzie na míru *HORIZONTÁLNÍ ŽALUZIE STANDARD* [online]. [cit. 2020-1-3]. Dostupné z: https://www.zaluzienamiru.cz/zaluzie-standard

[10] MALÝ, Martin *Protokol MQTT: komunikační standard pro IoT* [online]. [cit. 2020-1-3]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/

[11] VOJÁČEK, Antonín *IoT MQTT prakticky v automatizaci - 1.díl - úvod* [online]. [cit. 2020-1-3]. Dostupné z: https://automatizace.hw.cz/iot-mqtt-prakticky-v-automatizaci-1dil-uvod.html

[12] Oficiální dokumentace *MQTT verze 3.1.1* [online]. [cit. 2020-5-20] Dostupné z http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[13] E-shop Protosupplies *NodeMCU devkit v1.0* [online]. [cit. 2020-1-2]. Dostupné z: https://protosupplies.com/product/esp8266-nodemcu-v1-0-esp-12e-wifi-module/

[14] E-shop Laskarduino *Wemos D1 Mini* [online]. [cit. 2020-5-20]. Dostupné z: https://www.laskarduino.cz/wemos-d1-mini-esp8266-wifi-modul/

[15] Obrázek *NodeMCU devkit v1.0* [online]. [cit. 2020-1-3]. Dostupné z: https://github.com/nodemcu/nodemcu-devkit-v1.0/blob/master/Documents/NodeMCU_DEVKIT_1.0.jpg

[16] E-shop Aliexpress *MG996R* [online]. [cit. 2020-1-3]. Dostupné z: https://www.aliexpress.com/item/32355876966.html

[17] Espressif *Technická dokumentace ESP8266EX* [online]. [cit. 2020-5-18]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[18] Espressif *Technická dokumentace ESP32* [online]. [cit. 2020-5-20]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[19] ŘEZÁČ, Kamil *Krokové motory* [online]. [cit. 2020-5-20] Dostupné z https://robotika.cz/articles/steppers/cs

[20] Martin S. *Servo motor* [online]. [cit. 2020-5-20] Dostupné z https://navody.arduino-shop.cz/arduino-projekty/servo-motor.html

[21] Martin S. *Servo motor* [online]. [cit. 2020-5-20] Dostupné z https://navody.arduino-shop.cz/arduino-projekty/servo-motor.html

[22] Obrázek *ESP-12F* [online]. [cit. 2020-5-20] Dostupné z https://www.laskarduino.cz/esp-12f-esp8266-wifi-modul--tcp-ip/

[23] Obrázek *ESP32 WROOM-12* [online]. [cit. 2020-5-20] Dostupné z https://www.laskarduino.cz/espressif-esp32-wroom-32-2-4ghz-wifi-bluetooth-modul/

[24] Oficiální dokumentace *Google Smart Home* [online]. [cit. 2020-5-20] Dostupné z https://developers.google.com/assistant/smarthome/overview

[25] Oficiální dokumentace *Smart home intents* [online]. [cit. 2020-5-20] Dostupné z `https://developers.google.com/assistant/smarthome/concepts/intents`

[26] GitHub repozitář *WiFiManager od Tzapu* [online]. [cit. 2020-5-20] Dostupné z `https://github.com/tzapu/WiFiManager`

[27] GitHub repozitář *PubSubClient od Knolleary* [online]. [cit. 2020-5-20] Dostupné z `https://github.com/knolleary/pubsubclient`

[28] Web Solar Electricity Handbook *Solar Irradiance* [online]. [cit. 2020-5-20] Dostupné z `http://www.solarelectricityhandbook.com/solar-irradiance.html`

Příloha A

Seznam použitých zkratek

API Application Programming Interface

BLE Bluetooth Low Energy

CSS Cascading Style Sheet

DNS Domain Name System

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol Secure

I2C Inter-Integrated Circuit

IDE Integrated Development Environment

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

JSON JavaScript Object Notation

JWT JSON Web Token

LED Light-Emitting Diode

MQTT MQ Telemetry Transport

NFC Near Field Communication

NoSQL Non SQL

PC Personal Computer

QoS Quality of Service

RC Radio-Controlled

REST Representational State Transfer

SaaS Software as a Service

SDK Software Development Kit

SSL Secure Sockets Layer

SPI Serial Peripheral Interface

SQL Structured Query Language

URL Uniform Resource Locator

USB Universal Serial Bus

UTP Unshielded Twisted Pair

Příloha B

Obsah přiloženého CD

- **esp-blinds-module.ino** program pro ESP8266
- **/klient** projekt webové aplikace
- **/server** projekt serverové aplikace
- **/text** adresář s touto prací