Bachelor Thesis

Voice-Enabled Smart Home Modules

Author:
Josef Šanda

Supervisor:
Ing. Martin Bulín, MSc.

A thesis submitted in fulfillment of the requirements for the degree of Bachelor (Bc.)

in the

Department of Cybernetics

May 12, 2021

Declaration of Authorship

I, Josef Šanda, declare that this thesis titled, “Voice-Enabled Smart Home Modules” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

“Something supersmart.”

Your hero

UNIVERSITY OF WEST BOHEMIA

Abstract

Faculty of Applied Sciences

Department of Cybernetics

Bachelor (Bc.)

Voice-Enabled Smart Home Modules

by Josef Šanda

Your abstract goes here...

Acknowledgements

Your acknowledgements go here...

Contents

List of Figures

List of Tables

List of Algorithms and Code Parts

List of Abbreviations

ASR	**A**utomatic **S**peech **R**ecognition
BSON	**B**inary **JSON**
CLI	**c**ommand-**l**ine **i**nterface
HMM	**H**idden **M**arkov **M**odels
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
JSON	**J**ava**S**cript **O**bject **N**otation
MQTT	**M**essage **Q**ueuing **T**elemetry **T**ransport
REST	**RE**presentational **S**tate **T**ransfer

Chapter 1

Introduction

Your intro... Thesis ref example: Bulín, 2016, Misc ref example: Šmídl, 2017, Article ref example: McCulloch and Pitts, 1943, Online webpage ref example: Bradley, 2006

1.1 State of the Art

The most famous intelligent personal assistants include Alexa, Siri, Google Assistant. These virtual assistants work on a very similar principle as follows. The assistant constantly listens in his surroundings to see if a wake-up word has been spoken (listening analysis is processed on its hardware). After saying the wake-up word, the assistant starts recording a sound and analyzes simultaneously if no one is talking anymore. This recorded sound then send to the appropriate servers for processing. The server then handles the relevant device or service according to the processed user command and sends back the voice assistant an audio response.

Siri's first assistant was created by Apple in 2010 and shortly followed by Cortana by Microsoft in 2013 and Alexa by Amazon in 2014. The growing power of computers and advancing cloud technology allows scientists and software engineers to train voice assistants more easily. Over time, voice assistants can respond to the user more naturally and give the user the feeling of talking to a person.

In addition to these tasks, the user can connect the voice assistant to web services (see Fig. 1.1) like Tasker, IFTTT and other features (often called "skills") developed by third-party developers. By these additions, the user adds a new palette of commands such as automating social media posts, ordering a usual drink from a local Starbucks or summoning an Uber or Lyft using connected account data.

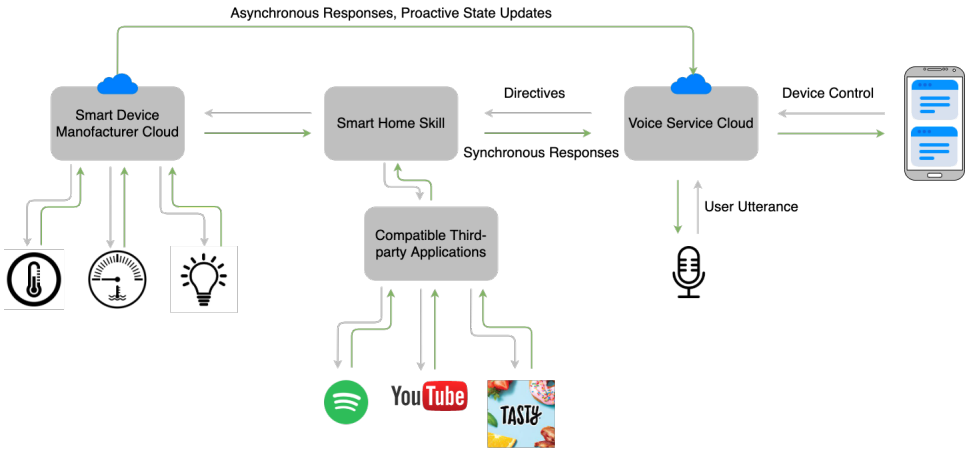

Figure 1.1: Connection schema of voice assistant service

Although each currently available voice assistant has unique features, they share some similarities and are able to perform the following basic tasks (Hoy, 2018):

- send and read text messages, make phone calls, and send and read email messages;
- answer basic informational queries (“What time is it? What’s the weather forecast? How many ounces are in a cup?”);
- set timers, alarms, and calendar entries;
- set reminders, make lists, and do basic math calculations;
- control media playback from connected services such as Amazon, Google Play, iTunes, Pandora, Netflix, and Spotify;
- control Internet-of-Things-enabled devices such as thermostats, lights, alarms, and locks; and
- tell jokes and stories.

1.1.1 Comparison between Google Assistant, Siri, Alexa

Because each voice assistant is developed independently, and each company protects its own. Despite their common ground, these assistants are quite different. In Fig. 1.2, is determined the most capable assistant by asking 800 questions that consist of categories like (Munster, 2019):

- Local – Where is the nearest coffee shop?
- Commerce – Order me more paper towels.
- Navigation – How do I get to Uptown on the bus?
- Information – Who do the Twins play tonight?
- Command – Remind me to call Jerome at 2 pm today.

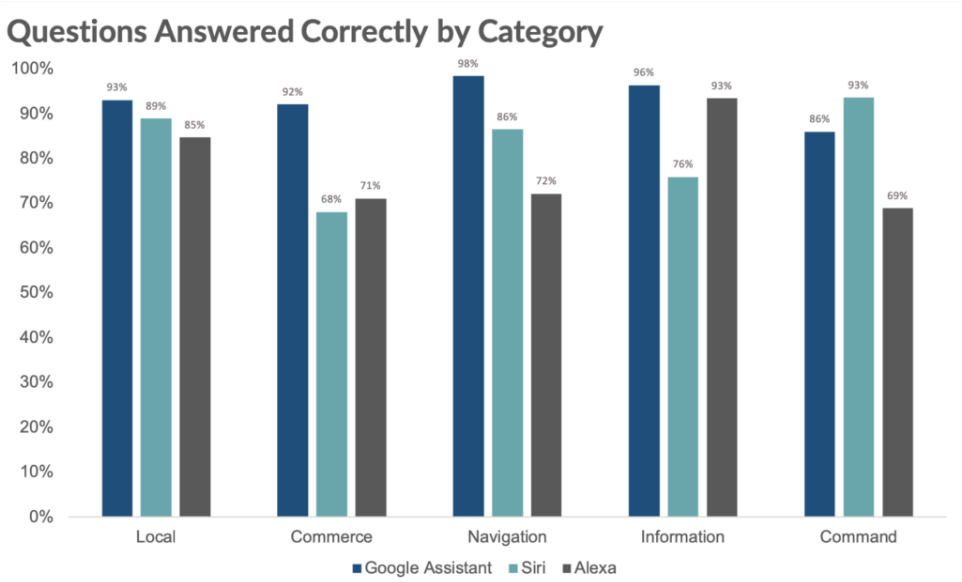

Figure 1.2: Voice assistent comparison by types of questions

Google Assistant has answered 93% correctly and has understood all 800 questions correctly. Siri has been next, has answered 83% correctly and has misunderstood only two questions. Alexa has answered 80% correctly and has misunderstood only one. According to the data shown in Fig. 1.2, Google Assistant has better results overall but lacks in the command category. Amazon Alexa has excellent results only in the information category, where it climbs just below the results of Google Assistant. Siri is brilliant in the command category for such functions as a calling, sending SMS or playing music.

If several users occupy the room, each voice assistant has its way of handling this situation. For example, Amazon Alexa and Google Assistant creating multiple voice profiles, which allows the user to train the assistant to recognize his voice specifically and therefore offer different data and use separate accounts for services. This is a very complex task, and not one can cope with it at the desired level.

1.2 Thesis Objectives

1.3 Thesis Outline

Chapter 2

Dialog Systems

2.1 Automatic Speech Recognition

ASR (Automatic Speech Recognition) is a way of converting sound into text.

Sound is nothing more than vibrations of the air that we humans are trained exceptionally well to decode. Moreover, now, we are teaching our computers how to do this. In the beginning, we have a stream of words that a person has uttered. The sound is picked up by a microphone and converted to a digital signal through a sound card, which means a stream of ones and zeroes.

One of the possible approaches in ASR modelling is, for example, at the level of phonemes or the level of whole words. We will only give an example here at the phoneme level, as the other approaches are very similar.

The first step the ASR system do is process the sound. It steps the sound to have chunks of speech (shown in Fig. 2.1) that can be worked with and that can be mapped to letters. These chunks are called phones.

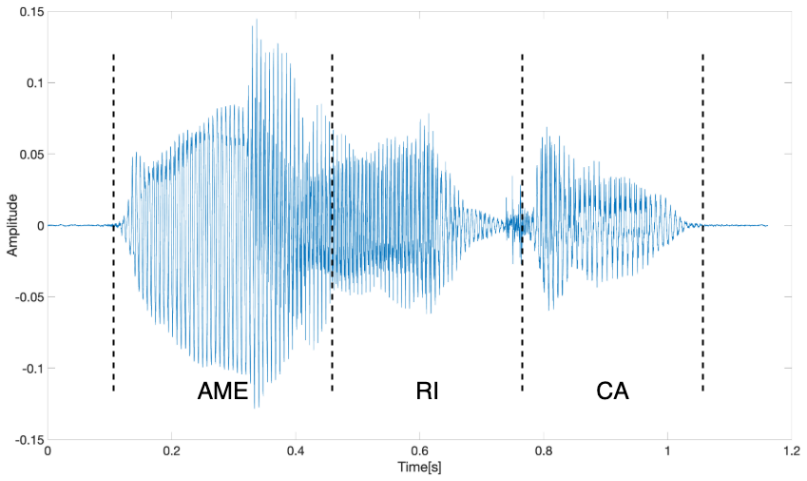

Figure 2.1: Chunked speech signal

The part of ASR responsible for mapping sound to phones is called the *acoustic model* as a set of building blocks, boxes which contain models for all phones in a given language as showing in Fig. 2.2.

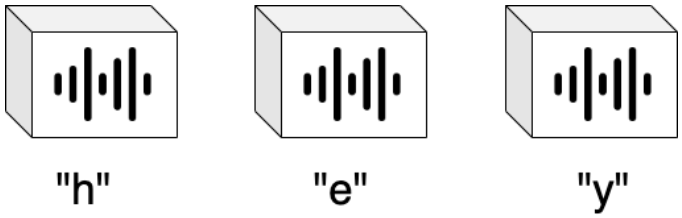

Figure 2.2: Phones boxes

There are boxes labelled, for example, A, B, C, depending on which phones are used in the particular language. On top of that, part of this construction set is also contextual probabilities. It means how likely a phone is to follow another. The acoustic model's task is to guess which phones have been pronounced and how they combine into a word. The acoustic model processes the sound and compares it to the models of individual phones from its boxes. Since speech is very complex in a real scenario, the chunks that a person uttered will be similar to more than one box. The acoustic model takes this into account and also looks at the neighbouring chunks and their contextual probabilities. For example, in the string "HELLO", the second phoneme that a person uttered might have been E. However, it also could have been ə, A or even I, with different degrees of certainty. The next phoneme is probably L, but it also could be R. There are different probabilities of these phones in context, for example, H followed by E is more likely, at least in English, than H followed by I. The ASR system combines these bits of information and outputs the most likely result - a string of phones.(Stanislav, 2020)

The next step is to convert it into words. Nevertheless, this part can be tricky because the ASR does not know when a word starts or ends. Contrary to popular belief, there are no pauses between words in fluent speech. This particular string "heloumaj..." of phones can constitute several different phrases, for example, "hell oh my nay miss" or "hello mine aim is", or "hello my name is". The part of ASR responsible for mapping phones to words and phrases is called the language model.

Hidden Markov Models are widely used for the statistical approach for automatic speech recognition. Suppose that $W = \{w_1, w_2, \ldots, w_N\}$ is a sequence of words, and $O = \{o_1, o_2, \ldots, o_N\}$ is a sequence of phones. These sequences are taken with a period of 10 ms for segments of speech of length from 20 to 40 ms. The Bayes Theorem for conditional probability is used to figure out which phones have been pronounced and how they combine into a word (see Fig. 2.3).

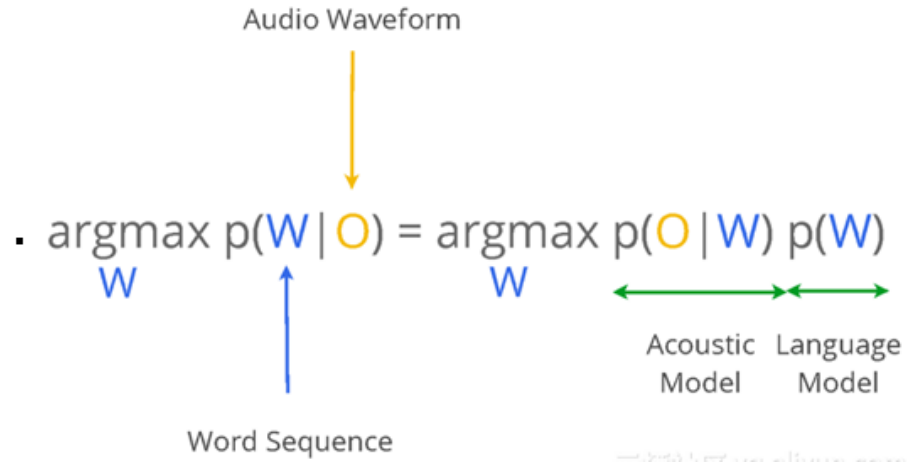

Figure 2.3: The relation among aucoustic model, language model and Bayes theorem

where P(W) is the a priori probability of word W, P(O|W) is the probability that the sequence of phones O will be generated under the conditions of pronouncing the sequence of words W, P(O) is the a priori probability of the sequence of phones O.

Since the probability P(O) is independent of the sequence of words W, it is possible to modify the equation into the form:

$$\boldsymbol{W}' = \underset{w}{\operatorname{argmax}} P(\boldsymbol{W} \mid \boldsymbol{O}) = \underset{w}{\operatorname{argmax}} P(\boldsymbol{W}) P(\boldsymbol{O} \mid \boldsymbol{W}) \tag{2.1}$$

2.2 Automatic Speech Synthesis

The task of generating a speech out of text information has originally two approaches:

1. concatenative (unit selection);
2. statistical parametric.

With concatenative synthesis is based on sequential combining of shot prerecorded samples of the speech. These samples can be stored in a database as of whole sentences, phrases, words and different phonemes. It depends on the application of the solutions. Building the unit selections synthesis model consists of three steps:

1. Recording of the whole selected speech units in no possible context.
2. Labelling segmentation of units.
3. Choosing the most appropriate units.

The concatenative method is the most straightforward approach to the speech generation. Disadvantages include the requirement to have

an ample storage for recorded units and an inability to apply various changes to a voice.

The statistical parametric synthesis consists of two parts, as shown in Fig. 2.4. The training step's approach is to extract excitation parameters like fundamental frequency and dynamic features, and spectral parameters from the speech database. Then we estimate them using one of the statistical models. The Hidden Markov model (HMM) is the most widely used for this task. It should be noted that HMM is conscious dependent. It means that in this step, in addition to phonetic context, linguistic and prosodic context is taken into account. In the synthesis part, at first given sentence is converted into points with a dependent label sequence, and then their chance HMM is constructed according to this sequence. Next, spectrum and excitation parameters are generated from the utterance HMM, and finally, speech waveforms are synthesized from these parameters using excitation generation and the speech synthesis filter. The advantages of the statistical parametric approach are:

1. Small footprint
2. No need to store the speech waveforms, only statistics language independence.
3. Flexibility in changing voice characteristics speaking styles and emotions.

The most noticeable drawback is the quality of a synthesized speech.

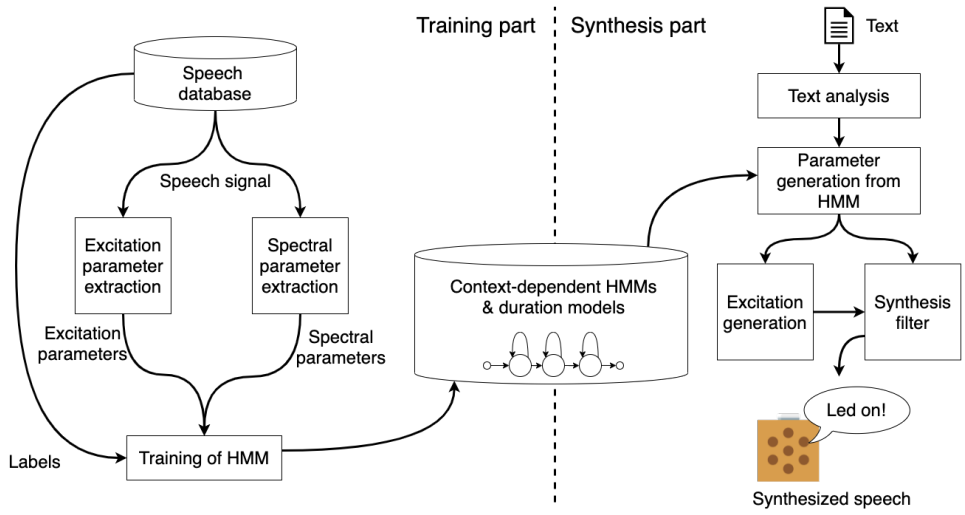

Figure 2.4: Statistical parametric speech synthesis (Zen, Tokuda, and Black, 2009)

2.3 The SpeechCloud Platform

The SpeechCloud platform, developed at the Department of Cybernetics of the University of West Bohemia, is a system that connects ASR and TTS systems operating together via one interface. It is then possible to use these systems by many applications simultaneously through

this interface. An independent instance is created for each dialogue system, allowing a client to create a characteristic language model, send a speech record to recognize, and receive the synthesized speech.

SpeechCloud provides the same services to all clients unless limited or specified otherwise. Each client should have the same functions, but each device, experiment or project is separated from the others, so the results are not affected by the unwanted intervention.

The architecture of the SpeechCloud and the connection to the client is briefly visualized in Fig. 2.5. The SpeechCloud using the module SCAPIServer as a primary point to establish a connection with the client application. Thus, the module negotiates with the client a specific application configuration, a control communication channel and the authentication of the session. The SCAPIServer then provides these pieces of information to other modules. The SIPSwitch module mediates the audio data transfer service between the SCWorker component and the client application. One instance of the SCWorker component is reserved for each client that holds one ASR and TTS instance. The SCWorker component has access to a TCP/IP network connection to collect additional data sources.

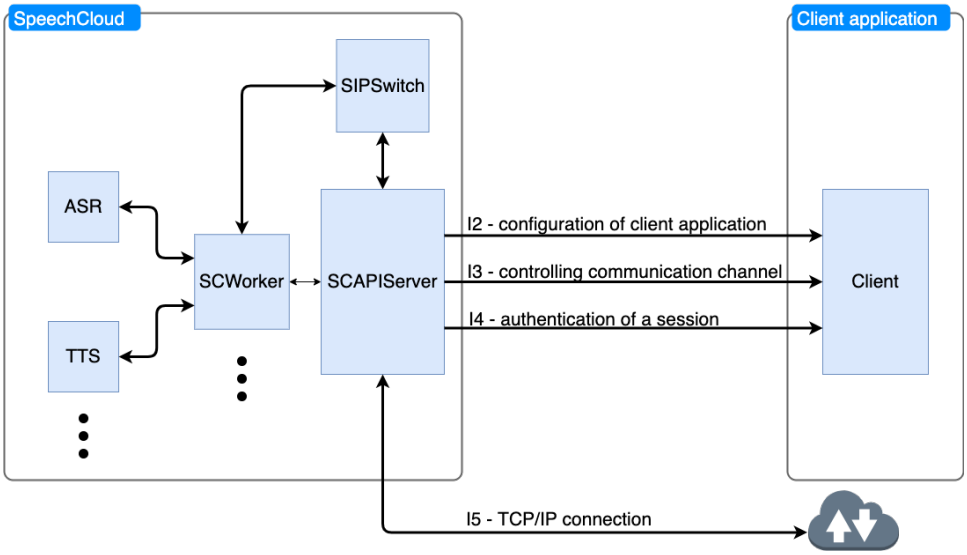

Figure 2.5: SpeechCloud schema

Solving the subject of the connection and transmission of data to the SpeechCloud via Internet communication protocols is not the content of this work hence are used ready-made software components and the SpeechCloud platform is used as a service.

Chapter 3

Backend

Own engine running on Raspberry Pi 4 has been developed and serves as the backend for the project. The whole engine is coded in Python, and adheres to the following principles:

- *Simplicity*: write a straightforward code that is easily understandable for later rewriting.
- *Modifiability*: write a code with the ability to admit changes due to a new requirement or detect an error that needs to be fixed.
- *Modularity*: write a well-encapsulated code of modules, which do particular, well-documented functions.
- *Robustness*: write a code focusing on handling unexpected termination and unexpected actions.

3.1 Diagram description

This section briefly describes the architecture of the engine that is figured on a diagram - see Fig. 3.1.

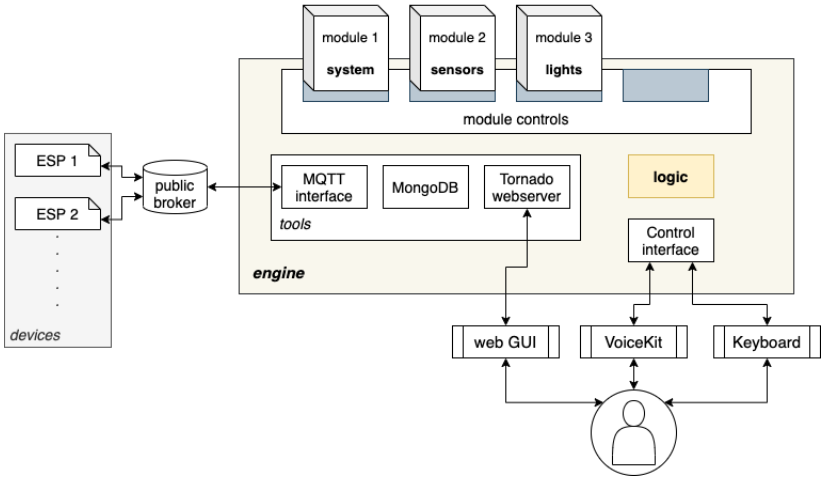

Figure 3.1: Project architecture

Engine uses tools like MQTT, MongoDB, Tornado web server that is described later. Each of them runs in its thread and concurrently. These

tools create a basis for modules and mediate main functionalities such as database, web server and communication.

The engine is designed to easily remove, add or update any mutually independent modules that define functions used by a user interface. Each module is described in the Chap. 4.

The engine also contains a separate block for *logic*. This block captures a command from the VoiceKit or keyboard interface, then browsing a predefined list of each module's commands and determines the best match for the user voice command or command written on the keyboard. If it does not find the voice command in lists, it replies that the command has not found with the recognized command.

3.2 Database

MongoDB is an open-source document database built upon a NoSQL database and written in C++. Database's horizontal, scale-out architecture support vast volumes of both data and traffic. One document can have others embedded in itself, and there is no need to declare the structure of documents to the system - documents are self-describing.(Jayaram, 2020)

Before using this type of database, we have to be familiar with different terminology compare to traditional SQL databases:

SQL Server	MongoDB
Database	Database
Table	Collection
Index	Index
Row	Document
Column	Field
Joining	Linking & Embedding
Partition	Sharding (Range Partition)
Replication	ReplSet

Table 3.1: MongoDB terminology

We use this type of database because it is famous for its use in agile methodologies, and the project tends to enlarge in the future. The main benefits are:

- MongoDB is easy to scale.
- Schema-less database: we do not need to design the database's schema because the code we write defines the schema, thus saves much time.
- The document query language supported by MongoDB is simplistic as compared to SQL queries.

- There is no need for mapping application's objects to database's objects in MongoDB.
- No complex joins are needed in MongoDB. There is no relationship among data in MongoDB.
- Because of using JSON[1] format to store data, it is effortless to store arrays and objects.
- MongoDB is free to use. There is no cost for it.
- MongoDB is simple to set up and install.

For adding a new field, the field can be created without affecting all other documents in the collection, without updating a central system catalog, and without taking the system offline.

In the project, we save all incoming messages from MQTT to MongoDB to a collection based on a name of interest module.

3.3 Communication

Communication is the backbone of the whole project among several devices over the internet. Therefore, it had to be found robust, scalable, and cost-effective protocols that transmit messages and data securely.Based on the survey, we choose three protocols that, in combination, satisfy all our requirements, and we will delve deeper into them in the following sections.

3.3.1 MQTT

MQTT is a standardized protocol by the OASIS MQTT Technical Committee used for message and data exchange. The protocol is designed specifically for the Internet of Things. The protocol is developed in vast language diversity from low-level to high-level programming language and designed at light versions for low-performance devices. Hence, it suits our use-case perfectly because each module possesses tons of various devices with limited resources that are already included or will arise later on. (Malý, 2016)

[1] **J**ava**S**cript **O**bject **N**otation

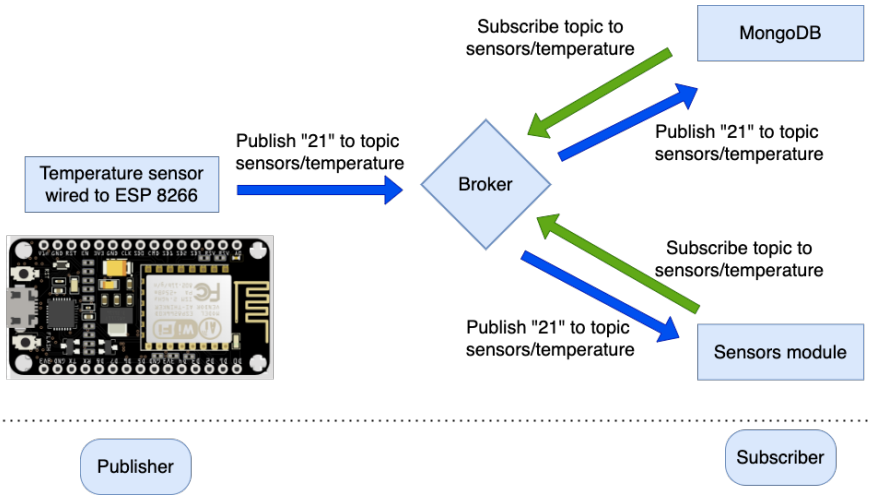

Figure 3.2: MQTT publisher/subscriber pattern

The design principles are to minimize network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. The protocol determines errors by TCP and orchestrates communication by the central point - broker. The protocol architecture uses a publish/subscribe pattern (also known as pub/sub) shown in Fig. 3.2, which provides an alternative to traditional client-server architecture. Architecture decouples publishers and subscribers who never contact each other directly and are not even aware that the other exist. The decoupling give us the following advantage:

- *Space decoupling*: publisher and subscriber do not need to know each other.
- *Time decoupling*: publisher and subscriber do not need to run at the same time.
- *Synchronization decoupling*: operations on both components do not need to be interrupted during publishing or receiving.

When the publisher sends his message, it is handled by the broker who filters all incoming messages and distributes them to accredited subscribers. The filtering is based on topic or subject, content and type.

In the case of MQTT, the filtering is subject-based and therefore, every message including a subject or a topic. The client subscribes to the topics he is interested in, and the broker distributes the messages accordingly as shown in Fig. 3.3.

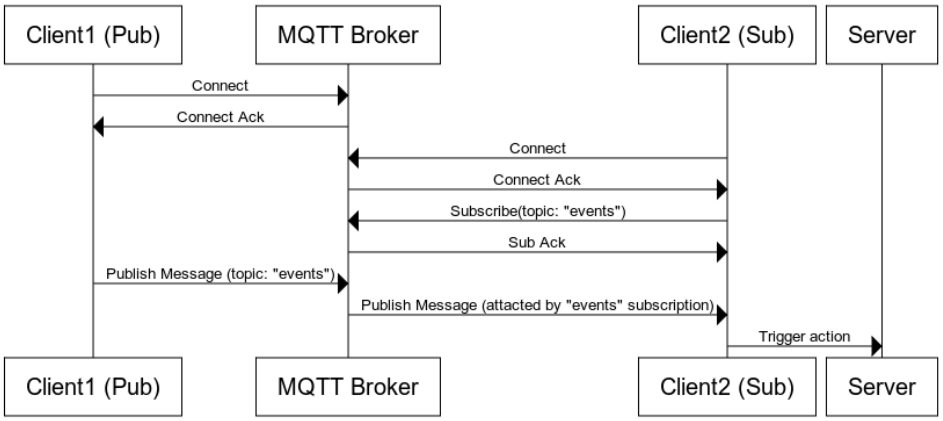

Figure 3.3: Diagram illustrating how communication in MQTT flow.

The topics are generally strings with a hierarchical structure that allow different subscription levels. It is feasible to use wildcards to subscribe, for example, sensors/# to receive all messages related to the sensors, for example, sensors/temperature or sensors/illuminance.

The MQTT protocol has the Quality of Service (QoS) levels essential to any communication protocol. The level of QoS can be specified for each message or topic separately according to its importance.

In MQTT, there are 3 QoS levels:

- *QoS 0*: This level is often called "fire and forget" when a message is not stored and retransmitted by a sender.
- *QoS 1*: Is is guarantees that a message is delivered at least one time to the receiver. The message is stored on a sender until it gets a PUBACK packet from a receiver.
- *QoS 2*: It is the highest level, and it guarantees that each message received only once by the intended recipients.

It is vital to mention MQTT have the feature retained messages that are mechanisms where the broker stores the last retained message for a specific topic. This feature allows a client does not have to wait until a new message is published to know the last known status of other devices.

3.3.2 WebSocket

In this work, WebSockets are used to provide communication between the client and the engine. WebSocket provides a low-latency, persistent, full-duplex connection between a client and server over TCP. The protocol is chiefly used for a real-time web application because it is faster than HTTP concerning more transfers by one connection. The protocol belongs to the stateful type of protocols, which means the connection between client and server will keep alive until either client or web server terminate it. The protocol fits for us in use between client and web server in case of real-time response.(Wang, Salim, and Moskovits, 2013)

The main benefits are:

- *Persistent*: After an initial HTTP handshake, the connection keeps alive using a ping-pong process, in which the server continuously pings the client for a response. It is a more efficient way than establishing and terminating the connection for each client request and server response. Server terminating connection after an explicit request from the client, or implicitly when the client goes offline.
- *Secure*: WebSocket Secure uses standard SSL and TLS encryption to establish a secure connection. Although we do not pursue this issue in our work, it is a valuable feature to add later.
- *Extensible*: Protocol is designed to enabling the implementation of subprotocols and extensions of additional functionality such as MQTT, WAMP, XMPP[2], AMQP[3], multiplexing and data compression. This benefit makes WebSockets a future-proof solution for the possible addition of other functionalities.
- *Low-latency*: WebSocket significantly reduces each message's data size, drastically decreasing latency by eliminating the need for a new connection with every request and the fact that after the initial handshake, all subsequent messages include only relevant information.
- *Bidirectional* - This enables the engine to send real-time updates asynchronously, without requiring the client to submit a request each time, as is the case with HTTP.

We will apply this protocol for transfer between clients such as VoiceKit, keyboard or web interface and engine in case of real-time response.

3.3.3 REST

In other cases like fetching data only once or data that is not required very frequently, we use RESTfull web service on a web server. This service enables us to transfer a lightweight data-interchange format JSON trivially and reliably - see Fig. 3.4. We use a standard GET REST request on a defined URI and then decode it like JSON for fetching data.

[2] E**x**tensible **M**essaging and **P**resence **P**rotocol - messaging and presence protocol based on XML and mainly used in a near-real-time exchange of structured data.

[3] **A**dvanced **M**essage **Q**ueuing **P**rotocol - an open standard application layer protocol for message-oriented middleware.

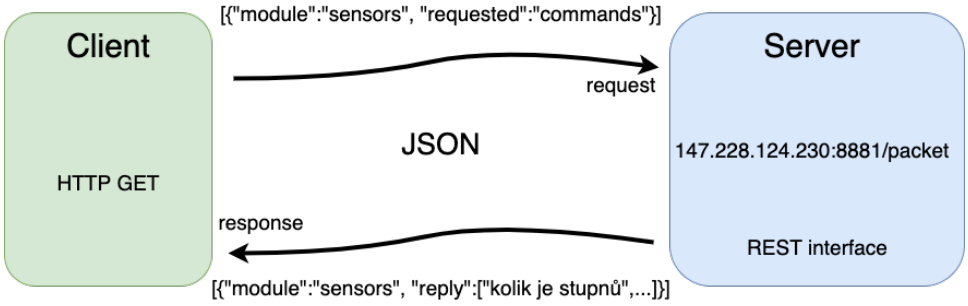

Figure 3.4: REST principle

3.4 Controllers

3.4.1 Keyboard

The keyboard is a python script with a particular purpose for developing new voice commands. This script opens up a CLI built upon a voicehome controller. The developer can quickly type a voice command with high accuracy through the command-line and debug the command thoroughly in various forms.

3.4.2 VoiceKit

VoiceKit is a building kit made by Google that lets users create their natural language processor and connect it to the Google Assistant or Cloud Speech-to-Text service. By pressing a button on top, users can ask questions and issue voice commands to their programs. All of this fits in a handy little cardboard cube powered by a Raspberry Pi.(*Voice Kit*)

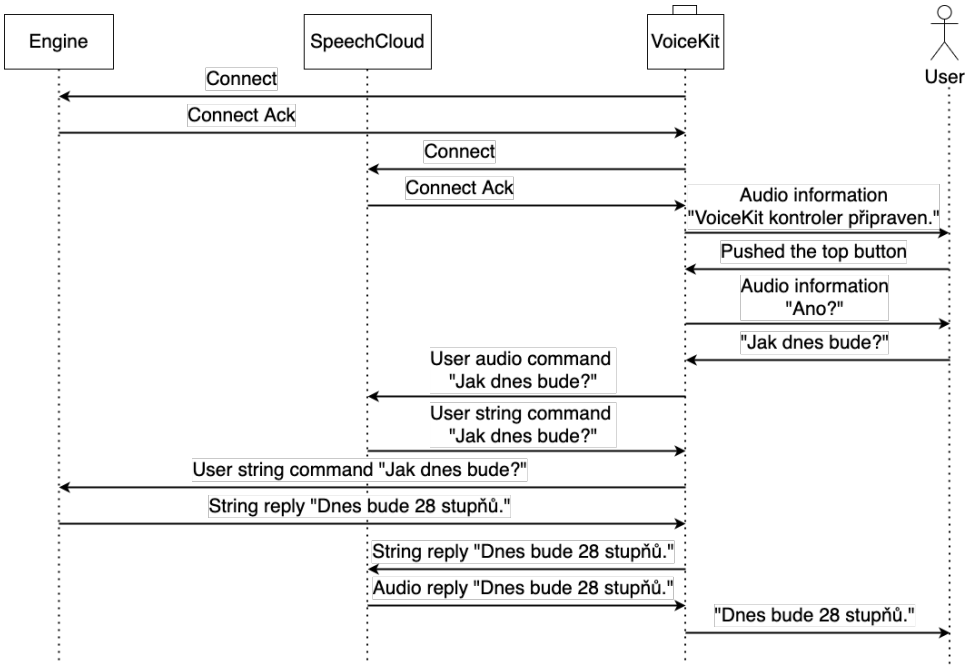

Figure 3.5: Diagram of messages flows during a conversation

Fig. 3.5 show a diagram of messages flows during a conversation. It is evident from the diagram that all communication with a user and the SpeechCloud mediate VoiceKit thus engine can manipulate just with a text.

3.4.3 Website

As the second interface next to the already mentioned Voice Kit is a website. The web server is implemented in Python using the Tornado framework.

The website's architecture aims to use it via a portable device like a smartphone and tablet or touch screen attached to the wall. Therefore the website is constructed to be responsible, straightforward and touch-friendly. The website's use-cases are to able the user to monitor ESP, sensors, lights, weather and voice commands, display historical sensors data, feasible voice commands and description of them, trigger lights and modules.

The website communicates with the engine by the already mentioned WebSocket. Figure 3.6 show an example of communication between the web site and the ESP to turn on an onboard led. Communication uses JSON as a data format and is evident from the figure that web site and engine use for communicating protocol WebSocket, whereas ESP and engine use MQTT.

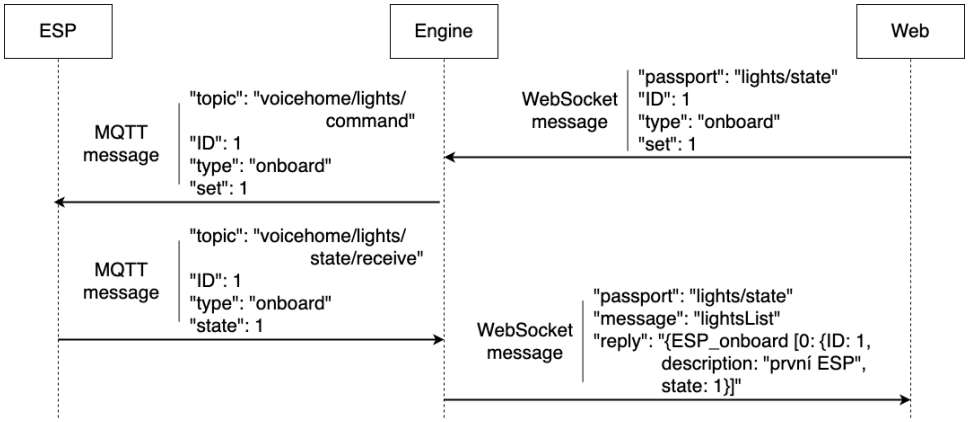

Figure 3.6: Diagram of message flows to turn on/off led on ESP by the website.

Chapter 4

Modules

Modules are well-encapsulated code written to provide functional and control elements (moves) above home to the user. Each module inherits from VoicehomeModule class that provide communicating interface to each module by WebSocket, MQTT and mediate writing and reading from MongoDB. Each module defines its topic for MQTT and passport for WebSocket that subscribe from these services. Messages containing these topics or passports are passed through the engine to the modules.

Thus each module has to be created by the following approach:

1) Create a new folder in "voicehome\modules\", the name of this folder is the module's name.
2) create two mandatory files
    1) "voicehome\modules\<module_name>\metadata.json" that include object with following variables.
        - "module_id": contain the name of the module
        - "description": contain a string with a brief description of the module
        - "mqtt_topics": contain a list of MQTT topics module wants to subscribe
        - "websocket_passports": contain a list of WebSocket passports passing messages to the module
        - "moves": list of objects that define moves this module is capable of
            - "move_id": a unique ID that follows the convention <module_name>_<order_in_this_list>
            - "method_name": contain the name of a Python function in <module_name>.py called when this move is activated
            - "description": contain a brief description of the move
            - "calls": list of calls (voice commands to VoiceKit) activating this move; it is a list of lists of words (must fit the chosen logic algorithm)

2) "voicehome/modules/<module_name>/<module_name>.py" that define class of module. This class have to inherit from VoicehomeModule and include all methods listed in the previous file metadata.json.

```
1 from modules.voicehome_module import VoicehomeModule
2
3
4 class Module_name (VoicehomeModule):
5
6     def __init__(self, engine, dir_path):
7         VoicehomeModule.__init__(self, engine,
              dir_path)
```

Part of Code 4.1: Template for creating a new module

After accomplishing these requirements, it is unnecessary to restart the entire engine but use the voice command "načti moduly" from the System module. Each particular module can be turned off or on using the web interface in the Modules tab, which is specified in more detail in Chap. ***.

4.1 Lights

The system module provides the user commands to control lights by voice. The user not only turns on, off or blinks lights but can also identify the development boards by lighting a onboard LED on a specific board. The module keeps in memory a list of all lights with their current status and detailed description.

The onboard LEDs are mounted on the board from the factory on pin 2. The other lights have their specific wiring, but one LED is prepared for demonstration purposes, which by our definition is located in the living room and is wired according to the diagram in Fig. 4.1.

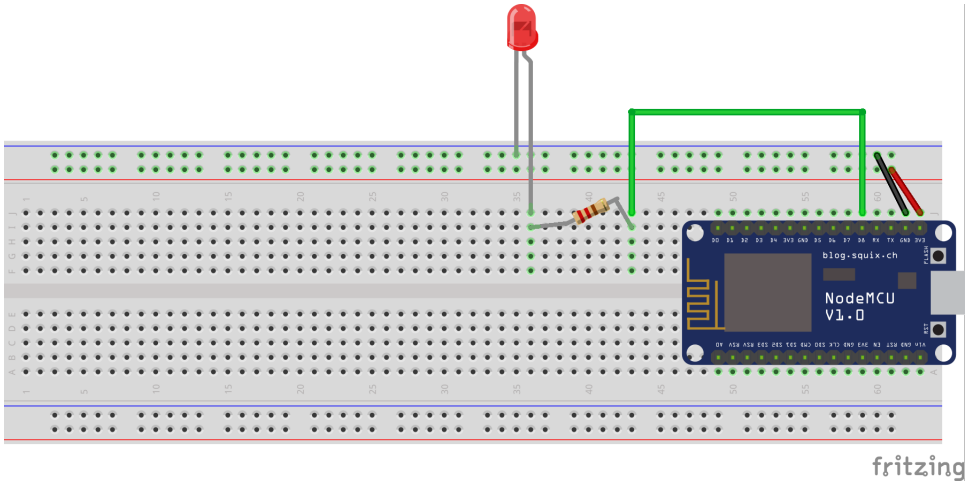

Figure 4.1: LED "living room" wiring diagram

4.1.1 Voice commands

The module responds to the following questions:

- Turn on all the onboard LEDs
  Voice commands
  - "Rozsviť všechny vývojové desky."
  - "Rozsviť všechny vestavěné ledky."

  Reply
  - Module confirm each light separately - "Vývojová deska jedna je rozsvícena.", "Vývojová deska dva je rozsvícena.", etc.
- Turn off all the onboard LEDs
  Voice commands
  - "Zhasni všechny vývojové desky."
  - "Zhasni všechny vestavěné ledky."

  Reply
  - Module confirm each light separately - "Vývojová deska jedna je zhasnuta.", "Vývojová deska dva je zhasnuta.", etc.
- Turn on the light 1
  Voice commands
  - "Zapni obýváku světlo."
  - "Rozsviť obýváku světlo."
  - "Zapni obývacím pokoji světlo."
  - "Rozsviť obývacím pokoji světlo."

  Reply
  - "Světlo v obývacím pokoji rozsvíceno."
- Turn off the light 1
  Voice commands
  - "Vypni obýváku světlo."
  - "Zhasni obýváku světlo."
  - "Vypni obývacím pokoji světlo."
  - "Zhasni obývacím pokoji světlo."

  Reply
  - "Světlo v obývacím pokoji zhasnuto."
- Turn on the onboard LED number 1
  Voice commands
  - "Rozsviť vestavěnou ledku vývojové desky číslo jedna."

  - "Rozsviť vývojovou desku číslo jedna."

  Reply

  - "Vývojová deska číslo jedna rozsvícena."

- Turn off the onboard LED number 1
  Voice commands

  - "Zhasni vestavěnou ledku vývojové desky číslo jedna."
  - "Zhasni vývojovou desku číslo jedna."

  Reply

  - "Vývojová deska číslo jedna zhasnuta."

- Turn on the onboard LED number 2
  Voice commands

  - "Rozsviť vestavěnou ledku vývojové desky číslo dva."
  - "Rozsviť vývojovou desku číslo dva."

  Reply

  - "Vývojová deska číslo dva rozsvícena."

- Turn off the onboard LED number 2
  Voice commands

  - "Zhasni vestavěnou ledku vývojové desky číslo dva."
  - "Zhasni vývojovou desku číslo dva."

  Reply

  - "Vývojová deska číslo dva zhasnuta."

- Turn on the onboard LED number 3
  Voice commands

  - "Rozsviť vestavěnou ledku vývojové desky číslo tři."
  - "Rozsviť vývojovou desku číslo tři."

  Reply

  - "Vývojová deska číslo tři rozsvícena."

- Turn off the onboard LED number 3
  Voice commands

  - "Zhasni vestavěnou ledku vývojové desky číslo tři."
  - "Zhasni vývojovou desku číslo tři."

  Reply

  - "Vývojová deska číslo tři zhasnuta."

- Voicekit answer which lights are turned on
  Voice commands

- "Která světla svítí."

Reply

- "Aktuálně nejsou rozsvícena žádná světla."
- "Aktuálně jsou rozsvícena tyto světla první ESP, druhé ESP..."

4.1.2 Messages structure

The engine uses for maintain lights following topics and messages:

- "voicehome/lights/command" - to turn the light on/off

```
{
    "ID":1,
    "type":"ESP_onboard",
    "set":0
}
```

Part of Code 4.2: Structure of JSON message to turn on/off the light in module *Lights*

- "voicehome/lights/state/command" - to ask the light for state

```
{
    "ID":1,
    "type":"ESP_onboard"
}
```

Part of Code 4.3: Structure of JSON message to asking for the state of the light in module *Lights*

- "voicehome/lights/state/receive" - to receive state of the light

```
{
    "type":"light",
    "state":0,
    "ID":1
}
```

Part of Code 4.4: Structure of JSON message to receive state of the light in module *Lights*

4.2 Sensors

The sensors module provides the user commands to communicate directly with sensors wired to the ESP development board or ask for statistics information such as average. The sensors connected to the module are bme280 (see Fig. 4.2), ds18b20 (see Fig. 4.3) and tsl2591 (see Fig. 4.4). The module uses the Python library to obtain old data from the MongoDB database to calculate statistical data.

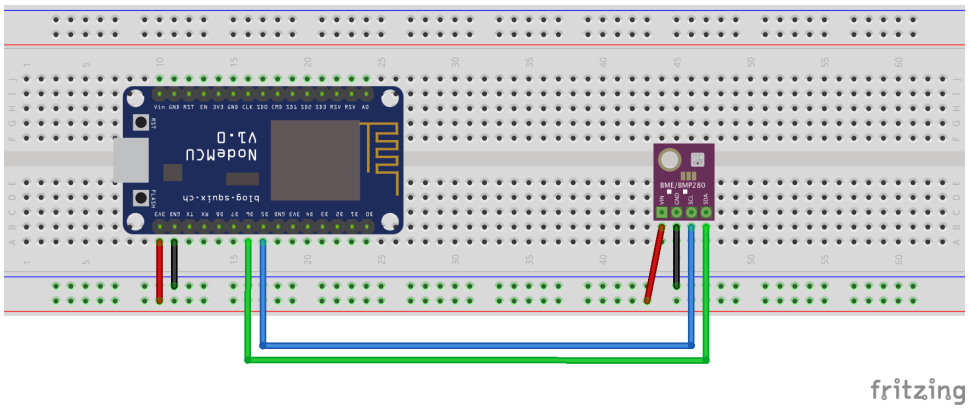

Figure 4.2: BME 280 wiring diagram

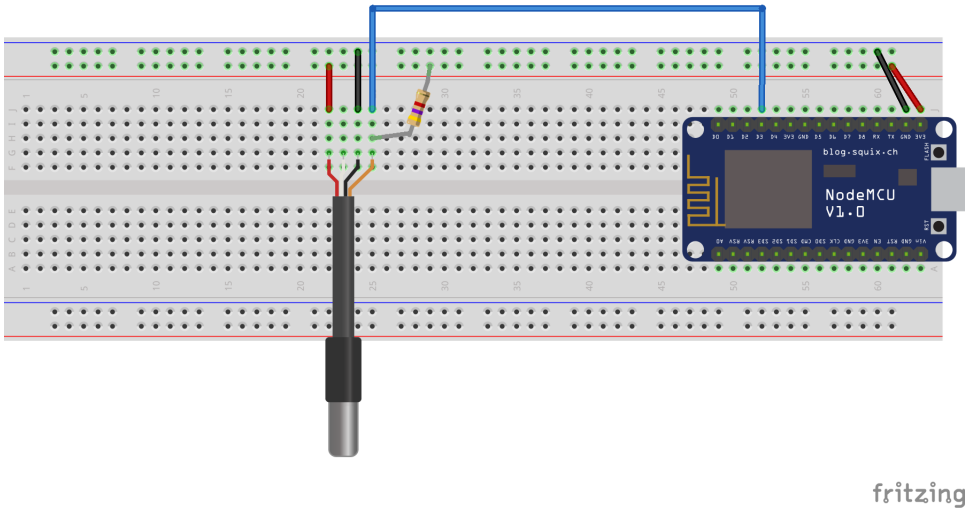

Figure 4.3: DS18B20 wiring diagram

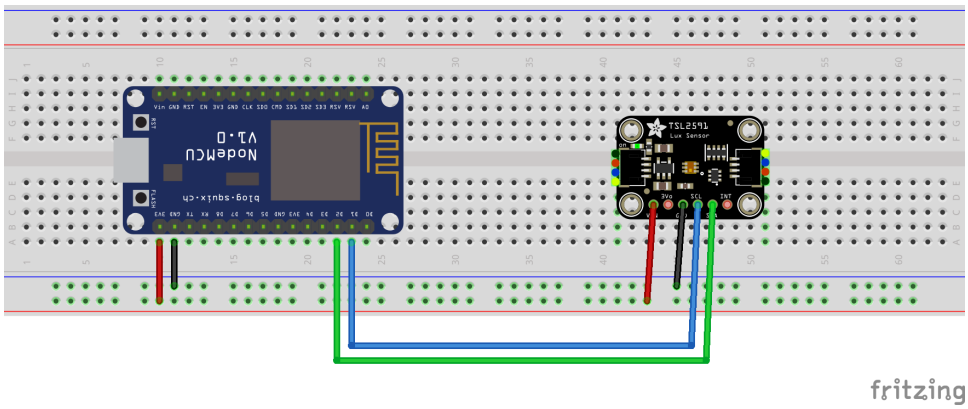

Figure 4.4: TSL2591 wiring diagram

- Sends a command via MQTT to measure current temperature
  Voice commands
  - "Kolik je stupňů?"
  - "Jaká je teplota?"

  - "Změř teplotu."

  Reply

  - "Na senzor je odeslán dotaz. Aktuální teplota je dvacet."

- Sends a command via MQTT to measure current pressure
  Voice commands

  - "Kolik je tlak?"
  - "Jaký je tlak?"
  - "Změř tlak."

  Reply

  - "Na senzor je odeslán dotaz. Aktuální tlak je devět set devadesát devět."

- Sends a command via MQTT to measure current humidity
  Voice commands

  - "Kolik je vlhkost?"
  - "Jaká je vlhkost?"
  - "Změř vlhkost."

  Reply

  - "Na senzor je odeslán dotaz. Aktuální vlhkost je deset."

- Sends a command via MQTT to measure current illuminance
  Voice commands

  - "Kolik je intenzity světla?"
  - "Jaká je intenzita světla?"
  - "Změř světlo."

  Reply

  - "Na senzor je odeslán dotaz. Aktuální intenzita světla je třicet."

- Voicekit answer average temperature for the last day
  Voice commands

  - "Průměrná teplota za poslední den."
  - "Dnešní průměrná teplota."

  Reply

  - "Teplotu nebylo možné vypočíst."
  - "Průměrná teplota za poslední den je dvacet."

- Voicekit answer average pressure for the last day
  Voice commands

  - "Průměrný tlak za poslední den."
  - "Dnešní průměrný tlak."

  Reply

  - "Tlak nebylo možné vypočíst."
  - "Průměrný tlak za poslední den je devět set devadesát."

- Voicekit answer average humidity for the last day
  Voice commands

  - "Průměrná vlhkost za poslední den."
  - "Dnešní průměrná vlhkost."

  Reply

  - "Vlhkost nebylo možné vypočíst."
  - "Průměrná vlhkost za poslední den je devět set devadesát."

- Voicekit answer average illuminance for the last day
  Voice commands

  - "Průměrná intenzita světelnosti za poslední den."
  - "Dnešní průměrná intenzita světelnosti."

  Reply

  - "Světelnost nebylo možné vypočíst."
  - "Průměrná intenzita světelnosti za poslední den je dvanáct."

- Voicekit answer average temperature for the last week
  Voice commands

  - "Průměrná teplota za poslední týden."
  - "Týdenní průměrná teplota."

  Reply

  - "Teplotu nebylo možné vypočíst."
  - "Průměrná teplota za poslední týden je dvacet."

- Voicekit answer average pressure for the last week
  Voice commands

  - "Průměrný tlak za poslední týden."
  - "Týdenní průměrný tlak."

  Reply

  - "Tlak nebylo možné vypočíst."
  - "Průměrný tlak za poslední týden je devět set devadesát."

- Voicekit answer average humidity for the last week
  Voice commands
  - "Průměrná vlhkost za poslední týden."
  - "Týdenní průměrná vlhkost."

  Reply
  - "Vlhkost nebylo možné vypočíst."
  - "Průměrná vlhkost za poslední týden je devět set devadesát."
- Voicekit answer average illuminance for the last week
  Voice commands
  - "Průměrná intenzita světelnosti za poslední týden."
  - "Týdenní průměrná intenzita světelnosti."

  Reply
  - "Světelnost nebylo možné vypočíst."
  - "Průměrná intenzita světelnosti za poslední týden je dvanáct."

4.3 Time

This module provides commands for manipulation with the time, such as asking for time, date, set timer. The module does not communicate with other devices. The module's functions exploit system information and information available on the Internet. To reach this information from the web is used technique call web scraping that can run the web site and suck desired pieces of information from this site. A simple example of this technic is shown in **??**

4.3.1 Voice commands

The module responds to the following questions:

- Send command to ask the server for the current time
  Voice commands
  - Kolik je hodin?
  - Čas

  Reply
  - Právě je pět hodin dvacet minut a pět sekund.
- Send command to ask the server for the current day of year
  Voice commands
  - Kolikátého dnes je?
  - Datum

  Reply

  - Dnes je 4. 5. 2021

- Send a command to the server to start timer on 3 minute
  Voice commands

  - Zapni časovač

  Reply

  - Časovač je nastaven na 3 minuty

- Send a command to the server to stop timer
  Voice commands

  - Vypni časovač
  - Zastav časovač

  Reply

  - Časovač je vypnut

- Ask the server for today's day of the week
  Voice commands

  - Co je za den v týdnu?
  - Co je za den?

  Reply

  - Dnes je pondělí.

- Ask the server for today's sunrise time
  Voice commands

  - Kdy vychází slunce?
  - Východ slunce

  Reply

  - Nebylo možno získat data ze serveru meteogram.cz
  - Slunce vychází v šest hodin a třicet minut.

- Ask the server for today's sunset time
  Voice commands

  - Kdy zapadá slunce?
  - Západ slunce

  Reply

  - Nebylo možno získat data ze serveru meteogram.cz
  - Slunce zapadá v osmnáct hodin a třicet minut.

- Ask the server for today's nameday
  Voice commands

  - Kdo má dnes svátek?

  - Svátek

  Reply

  - Nebylo možno získat data ze serveru svatky.centrum.cz
  - Podle serveru svatky.centrum.cz Renata.

4.4 System

The system module provides the user commands to test the functionality and adjust some settings of the engine. The module communicates primarily with the engine, but it is possible to establish this communication with other devices. Like other technology, the module uses the MongoDB library from python to test the engine's database.

- Reloads all the modules again. A fresh refresh.
  Voice commands

  - Načti moduly
  - Aktualizuj moduly
  - Přenačti moduly

  Reply

  - Moduly byly znovu načteny.

- Makes a testing write and read with the database.
  Voice commands

  - Otestuj databáze
  - Test databáze
  - Otestuj databázi

  Reply

  - Modul System: Databáze otestována. Vyhledáno dat jeden.
  - Modul System: Chyba při testování databáze

- Makes a testing MQTT publish to voicehome/system/test, which this module is also subscribing
  Voice commands

  - Otestuj MQTT
  - Test MQTT
  - Vyzkoušej MQTT

  Reply

  - Na mqtt nebylo možné odesla zprávu
  - Zpráva na mqtt odeslána

- Sends a testing websocket message with passport system/test.
  Voice commands
  - Otestuj WebSocket
  - Test WebSocket
  - Vyzkoušej WebSockety

  Reply
  - Na websoket nebylo možné odesla zprávu
  - Zpráva na websoket odeslána

4.5 Weather

The weather module provides the user commands to answer questions about the weather. The module's functions exploit the information available on the Internet. By preprocessing the information from the Internet and replating characters like "°C" to "stupnů" or "-" to "mínus", we can then send fully synthesizable text to SpeechCloud and answer the question to the user. To reach this information from the web is used technique call web scraping that can run the web site and suck desired pieces of information from this site. Preprocessing the information uses the technique regex and essential functions such as finding text and selecting text — a simple example of how regex is used shown in code part 4.1.

- Getting forecast for today from www.chmi.cz.
  Voice commands
  - Dnešní předpověď
  - Jak dnes bude?

  Reply
  - Nebylo možno získat data ze serveru chmi.cz
  - Server chmi.cz předpovídá pro dnešek. Polojasno až oblačno, místy přeháňky...
- Getting forecast for tomorrow from www.chmi.cz.
  Voice commands
  - Předpověď zítra
  - Jak zítra bude?

  Reply
  - Nebylo možno získat data ze serveru chmi.cz
  - Server chmi.cz předpovídá pro zítřek. Polojasno až oblačno, místy přeháňky...

- Getting forecast for monday from www.chmi.cz if it is up to four days and not today.
  Voice commands
  - Předpověď na pondělí
  - Jak bude pondělí?

  Reply
  - Nebylo možno získat data ze serveru chmi.cz
  - Server chmi.cz předpovídá na pondělí. Polojasno až oblačno, místy přeháňky...
- Getting forecast for tuesday from www.chmi.cz if it is up to four days and not today.
  Voice commands
  - Předpověď na úterý
  - Jak bude úterý?

  Reply
  - Nebylo možno získat data ze serveru chmi.cz
  - Server chmi.cz předpovídá na úterý. Polojasno až oblačno, místy přeháňky...
- Getting forecast for wednesday from www.chmi.cz if it is up to four days and not today.
  Voice commands
  - Předpověď na středu
  - Jak bude středu?

  Reply
  - Nebylo možno získat data ze serveru chmi.cz
  - Server chmi.cz předpovídá na středu. Polojasno až oblačno, místy přeháňky...
- Getting forecast for thursday from www.chmi.cz if it is up to four days and not today.
  Voice commands
  - Předpověď na čtvrtek
  - Jak bude čtvrtek?

  Reply
  - Nebylo možno získat data ze serveru chmi.cz
  - Server chmi.cz předpovídá na čtvrtek. Polojasno až oblačno, místy přeháňky...

- Getting forecast for friday from www.chmi.cz if it is up to four days and not today.
  Voice commands
  - Předpověď na pátek
  - Jak bude pátek?

  Reply
  - Nebylo možno získat data ze serveru chmi.cz
  - Server chmi.cz předpovídá na pátek. Polojasno až oblačno, místy přeháňky...
- Getting forecast for saturday from www.chmi.cz if it is up to four days and not today.
  Voice commands
  - Předpověď na sobotu
  - Jak bude sobotu?

  Reply
  - Nebylo možno získat data ze serveru chmi.cz
  - Server chmi.cz předpovídá na sobotu. Polojasno až oblačno, místy přeháňky...
- Getting forecast for sunday from www.chmi.cz if it is up to four days and not today.
  Voice commands
  - Předpověď na neděli
  - Jak bude neděli?

  Reply
  - Nebylo možno získat data ze serveru chmi.cz
  - Server chmi.cz předpovídá na neděli. Polojasno až oblačno, místy přeháňky...

Chapter 5

GUI

Chapter 6

Examples

6.1 XOR Function

Chapter 7

Discussion

Discussion starter...

7.1 Recapitulation of Methods

7.2 Summary of Results

Chapter 8

Conclusion

Conclusion text...

MongoDB project's application is as simple as possible because it is not the topic of the thesis. Is there plenty of room for improvement and streamlining.

8.1 Future Work

Outlook...

Bibliography

[1] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[2] Peter Bradley. *The XOR Problem and Solution*. 2006. url: http://mind.ilstu.edu/.

[3] Heiga Zen, Keiichi Tokuda, and Alan W. Black. “Statistical parametric speech synthesis”. In: *Speech Communication* 51.11 (2009), pp. 1039–1064. issn: 0167-6393. doi: https://doi.org/10.1016/j.specom.2009.04.004. url: https://www.sciencedirect.com/science/article/pii/S0167639309000648.

[4] Vanessa Wang, Frank Salim, and Peter Moskovits. “The WebSocket API”. In: *The Definitive Guide to HTML5 WebSocket* (2013), 13–32. doi: 10.1007/978-1-4302-4741-8_2.

[5] Martin Bulín. “Classification of terrain based on proprioception and tactile sensing for multi-legged walking robot”. MA thesis. Campusvej 55, 5230 Odense M: University of Southern Denmark, June 2016.

[6] Martin Malý. *Protokol MQTT: komunikační standard pro IoT*. 2016. url: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/.

[7] Luboš Šmídl. personal communication. supervision of the thesis. 2017.

[8] Matthew B. Hoy. “Alexa, Siri, Cortana, and more: An introduction to voice assistants”. In: *Medical Reference Services Quarterly* 37.1 (2018), 81–88. doi: 10.1080/02763869.2018.1404391.

[9] Gene Munster. *Annual Digital Assistant IQ Test*. 2019. url: https://loupventures.com/annual-digital-assistant-iq-test/.

[10] Prashanth Jayaram. *When to Use (and Not to Use) MongoDB - DZone Database*. 2020. url: https://dzone.com/articles/why-mongodb.

[11] Petr Stanislav. “Speech recognition of patients after total laryngectomy communicating by electrolarynx”. PhD dissertation. Západočeská univerzita v Plzni, 2020.

[12] *Voice Kit*. url: https://aiyprojects.withgoogle.com/voice/.

Appendix A1

Structure of the Workspace