



**FACULTY  
OF INFORMATION  
TECHNOLOGY  
CTU IN PRAGUE**

## ASSIGNMENT OF BACHELOR'S THESIS

**Title:** Office IoT  
**Student:** Mikuláš Formánek  
**Supervisor:** Ing. Jan Šedivý, CSc.  
**Study Programme:** Informatics  
**Study Branch:** Computer engineering  
**Department:** Department of Digital Design  
**Validity:** Until the end of summer semester 2018/19

### Instructions

Design and implement an IoT system for monitoring the presence and movement of office personnel and visitors. The system will use the following wireless sensors and actuators; camera, motion detector, microphone, speaker, socket on/off switch. The sensors will be selected by the supervisor. Review, compare, select and implement a suitable embedded processor for a communication hub. Use WiFi for communication with sensors and actuators. Review, select and use appropriate communication protocol. Use the Google cloud REST services and Dialog Flow tools to provide a voice user interface. On a voice request the system will inform about a presence of a selected person.

### References

Will be provided by the supervisor.

doc. Ing. Hana Kubátová, CSc.  
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.  
Dean

Prague February 8, 2018





**FACULTY  
OF INFORMATION  
TECHNOLOGY  
CTU IN PRAGUE**

Bachelor's thesis

## **Office IoT**

*Mikuláš Formánek*

Department of Digital Design  
Supervisor: Ing. Jan Šedivý, CSc.

May 14, 2018



---

# Acknowledgements

Thanks Jan Šedivý for supporting me all the way ...



---

# Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on May 14, 2018

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2018 Mikuláš Formánek. All rights reserved.

*This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).*

### **Citation of this thesis**

Formánek, Mikuláš. *Office IoT*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.



---

# Abstract

Bachelor thesis looks for the design of modern office from saving administrative tasks point of view. The main goal was to design and implement attendance system, set of actuators and find proper communication protocol. Attendance solution is using face recognition and the whole system has been connected with voice assistant.

**Klíčová slova** Hlasový asistent, Vestavěné systémy, Raspberry Pi, Rozpoznávání obličeje, rozpoznávání hlasu, intent, docházkový systém, biomentrické senzory, autentifikace, cloud computing, Turris Omnia



---

# Abstrakt

Bakalářská práce se zabývá návrhem moderní kanceláře z hlediska ulehčení administrativy a běžných úkonů v kanceláři. Cílem bylo navrhnout a implementovat docházkový systém, systém aktuátorů a vhodné komunikační rozhraní. V rámci řešení bylo využito rozpoznávání obličeje, docházkový systém byl napojen na hlasový asistent s komunikačním protokolem HTTP.

**Keywords** Voice Assistant, Embedded system, Raspberry Pi , Face recognition, Speech to text, intent, Attendance monitoring, Authentication, Cloud Computing, Turris Omnia



---

# Contents

<b>Introduction</b>	<b>1</b>
<b>Goals</b>	<b>3</b>
<b>1 Analysis</b>	<b>5</b>
1.1 Existing solutions of attendance monitoring . . . . .	5
1.2 Chat Bot . . . . .	8
1.3 Speech to Text . . . . .	11
1.4 Communication protocols for IoT . . . . .	13
<b>2 Implementation</b>	<b>17</b>
2.1 Comparision of single board computers . . . . .	17
2.2 Selecting proper sensors . . . . .	19
2.3 Face Recognition . . . . .	21
2.4 Attention manager . . . . .	28
2.5 Remote control of actuators . . . . .	31
2.6 Voice Assistant . . . . .	31
2.7 Implementation on Raspberry Pi . . . . .	34
<b>Conclusion</b>	<b>39</b>
<b>Bibliography</b>	<b>41</b>
<b>A Acronyms</b>	<b>43</b>
<b>B Contents of enclosed CD</b>	<b>45</b>



---

## List of Figures

1.1	P9000-SD . . . . .	5
1.2	Realand-ZDC60 . . . . .	6
1.3	BROADWAY 3D BR . . . . .	7
1.4	SYSF203TP . . . . .	8
1.5	Google Home Assistant . . . . .	8
1.6	Voice Commerce Sales in 2017 . . . . .	9
1.7	Voice Assistant (VA) architecture . . . . .	10
1.8	Speech Recognition Cycle . . . . .	11
1.9	MQTT exchange . . . . .	16
2.1	Cortex A53 . . . . .	18
2.2	Matrix Voice . . . . .	21
2.3	Face recognition diagram . . . . .	22
2.4	Picture with recognized person . . . . .	23
2.5	Nucleus Profile page . . . . .	29
2.6	Nucleus presence dashboard . . . . .	30
2.7	Turris scheme with jablotrone sensors . . . . .	31
2.8	TP-83N(thermostat), AC-88(switch relay) . . . . .	32
2.9	Dialog Flow - create intent . . . . .	37
2.10	Dialog Flow - entities and answers . . . . .	37





---

## List of Tables

1.1	Comparsion of P9000 . . . . .	6
1.2	Comparsion of Realand ZDC60 . . . . .	6
1.3	Comparsion of BROADWAY 3D BR . . . . .	7
1.4	Comparsion of SYSF203TP . . . . .	8
2.1	Single Board Computer (SBC) CPU . . . . .	18
2.2	SBC RAM . . . . .	18
2.3	Time to take photo . . . . .	19
2.4	Comparsion of Face Recognition Api . . . . .	20
2.5	Comparsion of Face Recognition Api . . . . .	22



---

# Introduction

In these days our society is mainly focused on speed and results, is required to automate basic operation like monitoring attendance.

In time of writing theses I spend lot of time in the new building of CIIRC and work under organization named eClub. We were finding unexisting application for new technologies and monitoring attendance was one thing we strongly need for ourselves.

Thesis shows usage voice-assistant like replacement for reception also application of using face-recognition for attendance monitoring, brings comparisons for sensors and embedded processors from usability point of view.

In theoretical part I am dealing with speech recognition and also taking look at voice-assistant principle.



---

## Goals

Main goal of the thesis is creating attendance monitoring system with using face recognition technology and implement voice assistant, which could replace reception in the future. Also thesis should bring comparison of Single board PC.



---

# Analysis

## 1.1 Existing solutions of attendance monitoring

In these days there are lot of types of monitor attendance on market. It's useful instrument how control your employer's time. We can divide them into these categories by type of authentication: biometric and batch.

### 1.1.1 Batch solution

Batch systems are the oldest one, each user has unique chip/paper/key and manually put them to some machine which recognize current user.

Company Jablotron offers system named PS-9000SD, using RFID reader.



Figure 1.1: P9000-SD;<sup>1</sup>

### 1.1.2 Biometric solution

Biometric sensors capture some unique part of our body for example fingerprint, iris, bloodstream or face to recognize particular person. And also you

---

<sup>1</sup>Online2018-04-16; <https://www.jabloshop.cz/bezkontaktni-vstupni-system-ps-9000-sd-vnitрни-pamet-slot-na-sd-kartu>

## 1. ANALYSIS

---

Table 1.1: Comparison of P9000

Pros	Cons
Ready to use	Cannot be extended
Price( 100USD)	Manual interaction
	Another device in pocket
	Easy to copy chip

Table 1.2: Comparison of Realand ZDC60

Pros	Cons
Hard to clone fingerprint	No access to fingerprints
Price( 500USD)	Manual interaction

cannot simply foist your co-worker's fingerprint easily.

On the market are getting more popular fingerprint sensors, for example Realand ZDC60 offers fingerprint and also batch authentication.



Figure 1.2: Realand-ZDC60;<sup>2</sup>

Another possible biometric sensor is dealing with 3D scan. Product BROADWAY 3D BR is 3D scan which can be used at reception, able to scan from distance up to 1,2m. On the other hand product is extremely expensive.

---

<sup>2</sup> Online 2018-04-16 ; [https://eshop.nejdochazka.cz/24-thickbox\\_default/dochazkovy-system-realand-biometrie-rfid.jpg](https://eshop.nejdochazka.cz/24-thickbox_default/dochazkovy-system-realand-biometrie-rfid.jpg)

<sup>3</sup>Online 2018-04-17; [http://katalog.abbas.cz/galerie/1\\_2675/broadway-3d-br-original.jpg](http://katalog.abbas.cz/galerie/1_2675/broadway-3d-br-original.jpg)



## 1.1. Existing solutions of attendance monitoring

---



Figure 1.3: BROADWAY 3D BR;<sup>3</sup>

Table 1.3: Comparison of BROADWAY 3D BR

Pros	Cons
Secured	Expensive
Without interaction	

Company SYSDO offers attendance system SYSF203TP with face-recognition, it uses double camera with IR light for bad light conditions, one thing which is not positive - you have to be really close - works on maximum distance of 25cm.

---

<sup>4</sup>Online 2018-04-17; <https://www.sysdo.cz/files/2016/10/sysf2032.png>



Figure 1.4: SYSF203TP,<sup>4</sup>

Table 1.4: Comparison of SYSF203TP

Pros	Cons
Price	Maximum distance
No extra device needed	

## 1.2 Chat Bot

Chatbot (sometimes referred to as a chatterbot) is a computer program that attempts to simulate the conversation or "chatter" of a human being via text or voice interactions. A user can ask a chatbot a question or make a command, and the chatbot responds or performs the requested action.[1]

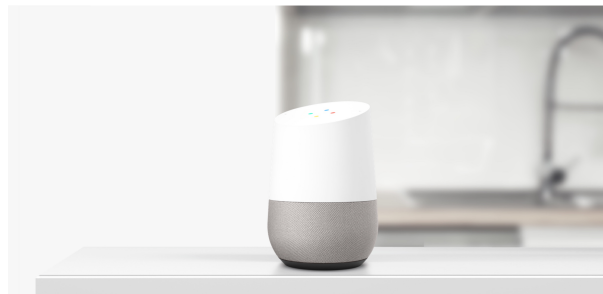


Figure 1.5: Google Home Assistant<sup>5</sup>

VA are being incorporated into a wide range of consumer products, and nearly half of U.S. adults (46%) say they now use these applications to interact with smartphones and other devices, according to a Pew Research Center survey conducted this spring.

<sup>5</sup> Online 2018-04-17; <https://madeby.google.com/static/images/home/hero.jpg>

Voice assistants are present on a wide range of devices, but the most common way for Americans to use them is on a smartphone: 42% of U.S. adults use voice assistants in this way. Some 14% of the public has used a voice assistant on a computer or tablet, while 8% say they use them on a stand-alone device such as an Amazon Echo or Google Home.[2]

### 1.2.1 Commerce usage

Nowadays are chatbot(also called voice-assistant(“VA”) widely used in many forms. For example every smartphone with Android 5.0 or higher can simply use Google Assistant, company Apple provides Siri for their devices, Alexa is product of Amazon.

#### Voice Commerce Sales 2017, 2022 in U.S. & U.K.

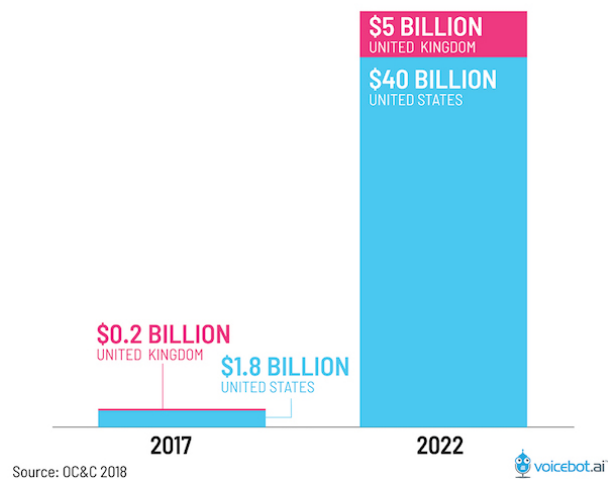


Figure 1.6: Voice Commerce Sales in 2017<sup>6</sup>

Many application are developed for VA, they can provide new “easy to use” - for example Uber has already application in Google Assistant, which can help make the process faster. You don’t need to type where you want to go, just say that. Unfortunately this feature is not available in Czech Republic.

Another way of usage shows Domino’s application - you can directly order through your VA, so it looks that VA overtakes some of the automated jobs like receptionist, shop assistant, and many others . . .

Good example is Question and Answer systems, which could be typically used in store, where customer don’t need to wait for shop assistant and just ask for help. I am expecting this become reality in some shops until 2019.

<sup>6</sup>Online 2018-04-18; <https://www.voicebot.ai/wp-content/uploads/2018/03/voice-commerce-sales-2017-2022-02.jpeg>

### 1.2.2 Intelligence Voice Assistant architecture

We can divide mechanism of Voice Assistant into 3 steps:

1. Speech to text
2. Natural language processing
3. Intent to action

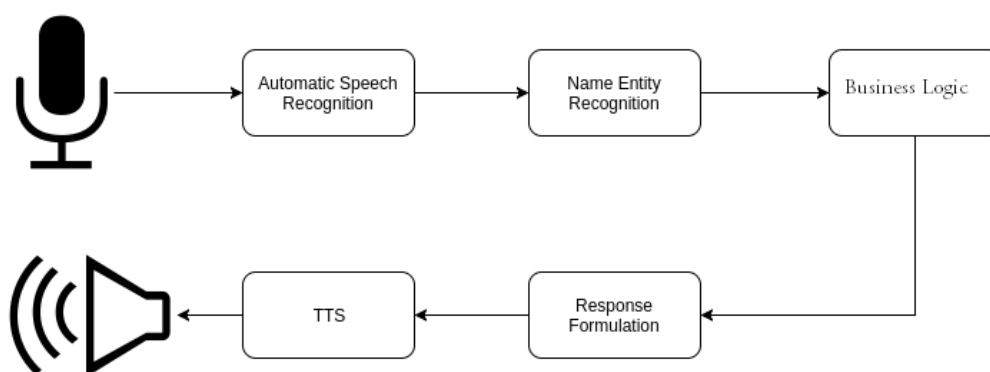


Figure 1.7: VA architecture

**Speech to text** is process which take voice input from microphone and try to recognize to text. This particular problem is the main reason why we still don't have voice assistant for every language. Special algorithms are using convolution neural networks, which are pre-trained on millions voice samples.

The software breaks your speech down into tiny, recognizable parts called phonemes — there are only 44 of them in the English language. It's the order, combination and context of these phonemes that allows the sophisticated audio analysis software to figure out what exactly you're saying, like the bread, cheese and sauce that differentiate a pizza from a calzone or a sandwich. For words that are pronounced the same way, such as eight and ate, the software analyzes the context and syntax of the sentence to figure out the best text match for the word you spoke.[3]

**Natural language processing (NLP)** is trying recognize what really user want to say, for example "How to say good night in german". Voice Assistant needs to detect we want translate sentence "good night" to german

“bis später”. There are many methods how to solve this ranking problem, for example TF-IDF, Word2Vec etc. . .

**Intent to action** fulfills user’s request, in our case VA call translate api to sentence “good night” and responds to user with “bis später”.

### 1.3 Speech to Text

Speech to text (STT) convert speech from a recorded audio signal to text. Humans convert words to speech with their speech production mechanism. An STT system aims to infer those original words given the observable signal. The most common and as of today best method is the probabilistic approach. A speech signal corresponds to any word (or sequence of words) in the vocabulary with a certain probability. Therefore, assuming a word  $x$  or word sequence  $X$  was spoken, we compute a score for matching these words with the speech signal. This score is calculated from the acoustic properties of speech sub-units (phonemes in the acoustic model), linguistic knowledge about which words can follow which other words. Including additional knowledge as the pronunciation score proposed in this work has also shown to be helpful. Finally, we sort the possible word sequence hypotheses by score, and pick the hypothesis with the highest score as recognition results. The process of speech recognition can be divided into the following consecutive steps.[4]

1. pre-processing
2. feature extraction
3. decoding
4. result post-processing

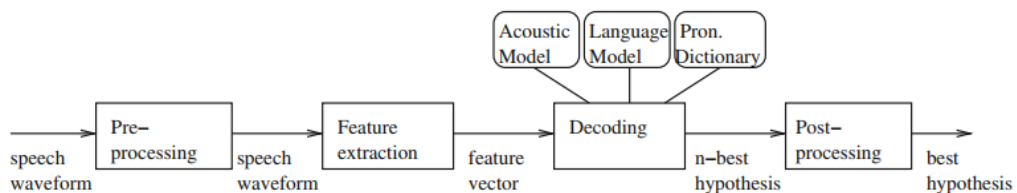


Figure 1.8: Speech Recognition Cycle<sup>7</sup>;

<sup>7</sup>Online 2018-05-05; [https://doi.org/10.1007/978-3-642-19586-0\\_2](https://doi.org/10.1007/978-3-642-19586-0_2)

### 1.3.1 Pre-processing

Speech is recorded with a microphone and the signal is sampled with 16 kHz. The Shannon sampling theorem states that a bandwidth limited signal can be perfectly reconstructed if the sampling frequency is more than double of the maximum frequency. That means that in the sampled data, frequencies up to almost 8 kHz are constituted correctly. While this is not the total frequency range of human speech, it is more than double of what is transmitted over telephone networks. In these are typically limited to the 5 Hz–3.7 kHz range, and has shown in research to be sufficient for speech recognition applications. It is possible to remove frequencies below 100 Hz with a high-pass filter as they tend to contain noise but can be considered of little relevance for speech recognition. An important part of pre-processing is also speech/non-speech segmentation. As speech recognition systems will classify any sound to any phoneme with some (even if very low) probability, background noise can cause insertions of phonemes or words into the recognition result if the noise resembles the parameters of a phoneme model better than those of a silence model. Such insertions can be reduced by removing areas from the speech signal between the start of the recording and the point of time when the user starts to speak, and after the end of the utterance.[4]

### 1.3.2 Feature extraction

Features extraction is process where acoustic observations are extracted over time frames of uniform length. These frames, the speech signal is stationary. The length of these frames is typically around 25 ms, for the acoustic samples in this window one multi-dimensional feature vector is calculated. The time frames are overlapping and shifted by typically 10 ms. On the time window, a fast Fourier transformation is performed, moving into the spectral domain. Human ears do not perceive all frequency bands equally. This effect can be simulated with band-pass filters of non-uniform frequency band widths. Until 500 Hz, the width of the filters is 100 Hz, after that it increases logarithmically. The spectrum is decorrelated with a discrete cosine transformation. Of the resulting coefficients, the first coefficients carry the most significance. Therefore only the first e.g. ten coefficients are selected as feature vector.[4]

### 1.3.3 Decoding

Decoding is the process to calculate which sequence of words is most likely to match to the acoustic signal represented by the feature vectors.

## 1.4 Communication protocols for IoT

Currently in hype of IoT world is necessary to consider connection between sensors, since every device has different performance, power consumption, it would be uneconomical to use same communication protocol for each application and device.

### 1.4.1 HTTP

Hyper Text Transfer Protocol (HTTP) is one of the oldest protocol, which has been introduced in 1991 for exchanging hypertext. HTTP operate under TCP protocol. It provides (subset)following operations:

- GET - request for specific resources
- POST - request, where usually server accepts data insides
- DELETE - request for delete specific resources
- PUT - creates specified resources if does not exists
- and many others...

#### 1.4.1.1 GET request header

Typical request and response for an webserver is:

```
GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtd4654
Pragma: no-cache
Cache-Control: no-cache
```

Listing 1: Code segment - Get header

## 1. ANALYSIS

---

```
HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Sat, 28 Nov 2009 04:36:25 GMT
Server: LiteSpeed
Connection: close
X-Powered-By: W3 Total Cache/0.8
Pragma: public
Expires: Sat, 28 Nov 2009 05:36:25 GMT
Etag: "pub1259380237;gz"
Cache-Control: max-age=3600, public
Content-Type: text/html; charset=UTF-8
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT
X-Pingback: http://net.tutsplus.com/xmlrpc.php
Content-Encoding: gzip
Vary: Accept-Encoding, Cookie, User-Agent
Data...
```

Listing 2: Code segment - Get header



Pros:

- widespread
- libraries support
- well known

Cons:

- implementation is not trivial
- talkative
- client is the one who still has to be active

As we can see the biggest problem of usage HTTP for IoT world is fact that header is keep growing with information which are not useful.

### 1.4.2 MQTT

MQ Telemetry Transport (MQTT) is publish-subscribe based protocol, where aim is to provide protocol to devices with limited bandwidth and small footprint. Publisher-subscriber model means, there is any central point in network(let say broker), where is device which is handling all exchange of messages. Messages are sorted by topic - for example: building, floor, park place ... Device could publish some data through this broker - PUBLISH /park-place/1 free. Or subscribe for receiving messages of chosen topic.[5] Also MQTT allow 3 level Quality of Service:

- level 1 - fire and forget. Publisher just send message to broker.
- level 2 - at least once. Publisher sends message and waits for PUBACK from broker.
- level 3 - exactly once. Same as level 2, but publisher has to claim he received PUBACK message with PUBREL.

Pros:

- small footprint
- not so hard to implementation
- QoS

Cons:

- limited support in libraries
- broker is required

---

<sup>8</sup>Online 2018-05-05; <https://www.hivemq.com/Screenshot-2014-10-22-at-12.21.07.png>

## 1. ANALYSIS

---

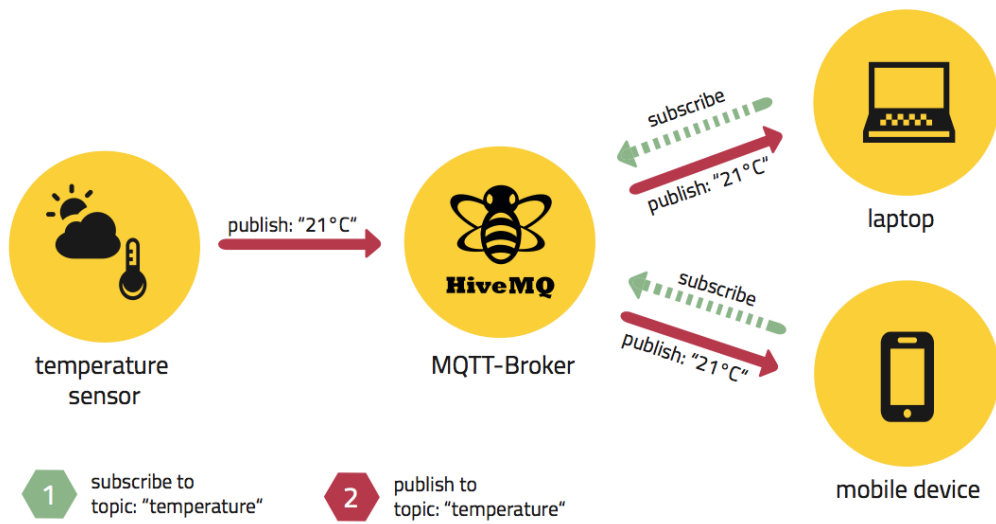


Figure 1.9: MQTT exchange<sup>8</sup>;

---

# Implementation

I have decided to try mix some authentication options for my monitor attendance system. First of all, I am going to use face-recognition to identify newcomers into workspace, since we have obtained Turris Omnia for internet connection. I will use data from ARP table, every user will have device with unique MAC address so I can easily observe who is inside workspace and due to face-recognition I am going to benefit from 2-step authentication.

## 2.1 Comparison of single board computers

SBC is small form factor computer which is built on single circuit board and includes all functional computer components.[6]

They become popular since 2012, caused by releasing Raspberry Pi by Raspberry Pi Foundation for education purposes. For comparing SBC I take these representatives:

- Raspberry Pi 3 (RPI)
- Banana Pi M64 (BRPI)
- ROCK64 3 (RRPI)
- Orange Pi Plus2E H3 (ORPI)

### 2.1.1 CPU

Every of these SBC includes popular ARM CPU, as mentioned in table only ORPI offers has different CPU: H3 Quad-core Cortex-A7 and others has Quad-Core ARM Cortex A53. Main difference between them is fact that A53(released in 2012) is 64 bit and A7(Released in 2011) is 32 bit. Also A53 unlike A7 offers D-Cache with ECC and I-Cache with parity, and is improved for AES instructions. From this point I choose A53.

## 2. IMPLEMENTATION

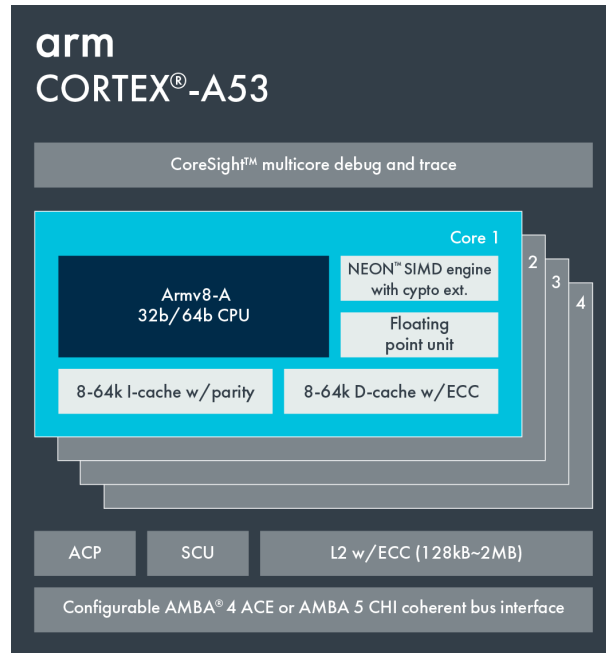


Figure 2.1: Cortex A53;<sup>9</sup>

Table 2.1: SBC CPU

Raspberry Pi 3	Quad-Core ARM Cortex A53
Banana Pi M64 3	Quad-Core ARM Cortex A53
ROCK64 3	Quad-Core ARM Cortex A53
Orange Pi Plus2E H3	H3 Quad-core Cortex-A7

Table 2.2: SBC RAM

Raspberry Pi 3	1 GB DDR3
Banana Pi M64 3	2 GB LPDDR3
ROCK64 3	4GB DRAM
Orange Pi Plus2E H3	1GB DDR3

### 2.1.2 RAM

As compared in table each SBC offers different size and type of RAM. BRPI seems to be more energy saving due to used LPDDR3 which compared to DDR3 saves about 30% of energy when active and 90% on standby.[7]

<sup>9</sup>Online 2018-04-25; <https://developer.arm.com/-/media/developer/Block%20Diagrams/Cortex-A%20Processor%20Block%20Diagrams/CortexA53.png>

Table 2.3: Time to take photo

	RPI cam V2(640x480)	RPI cam V2(3264x2448)	PlayStation Eye(640x480)
Init time	0.351855039597	0.350429058075	0.00145101547241
Capture time	0.958659172058	1.08443903923	2.35704493523

### 2.1.3 Peripherals

All of them offers GPIO pins, which I will use to connect for motion sensor, also usb would be needed, RRPI has USB 3.0, which is “nice to have” feature, but not so many microphones/webcams supports third version. . .

BRPI with RPI and ORPI includes CSI port for camera. Each of them offers HDMI ports.

### 2.1.4 Conclusion

As I realized in test cameras connected through CSI port are much faster than USB, so although 4GB from RRPI seems pretty interesting in my needs will be 1GB enough. Main impact for my decision for choosing RPI was fact about community support and quality, where RPI makes “standard”.

## 2.2 Selecting proper sensors

For my implementation of Face Recognition attendance and VA I needed to choose proper microphone, camera and stereo.

### 2.2.1 Camera

There plenty of options to capture picture, in our case using Raspberry PI I have a few options. First one is using CSI Camera Port with Raspberry Camera Module V2. Second one is finding some USB camera. Last one is using ip camera.

Our goal is choose camera with optimal resolution and with lowest time to capture. For these I prepared simple script (this one for camera pi module, but it doesn't really differs from usb camera).

There are currently being sold two modules for raspberry camera v1 and v2, these differs from maximum resolution - v2 has 8Mp and v1 5Mp, also each model has separately sold version “NOIR” without IR filter for capturing in bad light conditions.

For representative of USB camera I choosed Sony PlayStation Eye with resolution 640x480 with 60 frames per second and 320x240 120 frames per second.

From these results I choose Raspberry Camera V2 for my purposes.

Table 2.4: Comparison of Face Recognition Api

	Number of microphones
Google Home Mini	2
Google Home	2
Echo Dot	7
Amazon Echo	7
Home Pod	6

### 2.2.2 Microphone

For good speech recognition is necessary to have perfect microphone. I researched what microphones are used in commercial VA, they mainly use microphone arrays with various number of microphones and benefits from beamforming.

A beamformer is a processor used in conjunction with an array of sensors to provide a versatile form of spatial filtering. The sensor array collects spatial samples of propagating wave fields, which are processed by the beamformer. The objective is to estimate the signal arriving from a desired direction in the presence of noise and interfering signals. A beamformer performs spatial filtering to separate signals that have overlapping frequency content but originate from different spatial locations. This paper provides an overview of beamforming from a signal processing perspective, with an emphasis on recent research. Data independent, statistically optimum, adaptive, and partially adaptive beamforming are discussed. [8]

I found 2 interesting options for me, first one is Sony Playstation Eye, webcam with 4 microphones, each channel is sampled with 16bit decoder on rate of 48kHz.

Other option was Matrix Voice. “MATRIX Voice is an open-source VOICE RECOGNITION platform consisting of a 3.14-inches in diameter dev board, with a radial array of 8 MEMS microphones connected to a Xilinx Spartan 6 FPGA & 64 Mbit Flash, a 512Mbit SDRAM and 18 RGBW LED’s.” [9]

Matrix Voice could be really usefull, already implemented Google Home or Alexa with Rapsberry Pi, or LED array could be used for feedback, but in time of creating own VA wasn’t available. . . For these reason I decided for Sony Playstation Eye, maybe later, when Matrix Voice will be available l would like to change microphone.

### 2.2.3 Motion sensor

I chose basic sensor, which is often used in comunity - HC-SR501, which normally operates on 5V and output signal has 3,3V logic. It also offer adjustable

length of pulse.

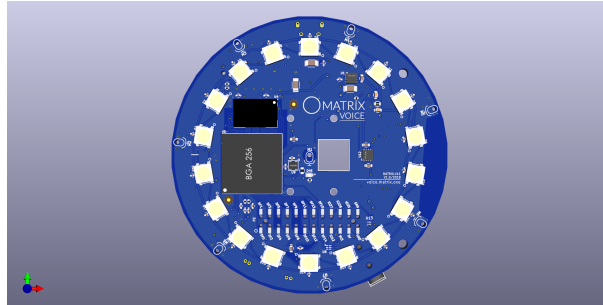


Figure 2.2: Matrix Voice;<sup>10</sup>

```
from picamera import PiCamera
import time
start_time = time.time()
camera = PiCamera()
camera.resolution=(640,480)
print("INIT: \t\t %s seconds ---" % (time.time() - start_time))
#pygame.camera.list_camera() #Camera detected or not
camera.capture('image.jpg')
print("CAPTURE: \t %s seconds ---" % (time.time() - start_time))
```

Listing 3: Code segment - time measure

## 2.3 Face Recognition

Due to my previous research, I chose Raspberry Pi 3 as my embedded system. And Raspberry Pi Camera V2 for face recognition and HC-05 as motion sensor.

I was wondering if I make my own neural network for face recognition, but that would exceed the limit of this thesis, anyway I researched a few existing solutions. Almost every big company which provides cloud computing/hosting also has their own solution for face recognition for example: Microsoft Face API, Cloud Vision API from Google or IBM Visual Recognition.

I decided to choose Amazon Face Rekognition (not typo) due to its available free tier and mostly for data protection, they claim that they don't store face pictures on their servers, just classification vectors. In our office we are working about 10 people and the free tier allows you to store 1000 face meta-data and

<sup>10</sup>Online 2018-04-25; <https://matrix-io.github.io/matrix-documentation/matrix-hal/img/matrix-voice-top.png>

## 2. IMPLEMENTATION

---

Table 2.5: Comparison of Face Recognition Api

	Amazon Face Rekognition	Microsoft Face API	Kairos
Free tier	Yes	Yes	Yes
Privacy	Don't store face image on servers	Store face pictures	No information
SDK	Yes	No	Yes
API	Yes	Yes	Yes

process 5000 images per month for first 12 moths. After 12 moths we will be outside free tier so costs are 1\$ per 1000 processed images.

Basic cycle:

1. Wait for signal from Motion HC-501
2. Capture few photo
3. Try to recognize person from picture
4. Send information about entrance to server



Figure 2.3: Face recognition diagram

### 2.3.1 Implementation in Python

I decided use Python to implement behavior, mostly for good libraries to work with GPIO, neural network, raspberry camera.

In `__main__` function we can see first three steps of basic cycle: Waiting for signal from HC-05 and Capturing photo.

Last step - Face Recognition is defined in two functions: `recognize_person`, which calls `recognize_photo` for every taken picture.

Function `recognize_photo` gets response from Face Rekognition, I have used boto3 framework, which communicates with Amazon Web Services(Amazon Web Service (AWS)). If the person is recognized I save the picture for later on SD card of Raspberry and also I paint square around recognized face.

Function `search_faces_by_image` takes few arguments, one of them - most important is `threshold`, which means you can setup threshold for accepting person's identity.



```
    if __name__ == "__main__":
        init_logging()
        load_secrets()
        init_gpio()

    with picamera.PiCamera() as camera:
        while True:
            streams = [io.BytesIO() for i in range(NO_OF_PHOTOS)]           # prepara

            # wait and shoot
            GPIO.wait_for_edge(MOTION_SENSOR_PIN, GPIO.RISING)
            camera.capture_sequence(streams, 'jpeg', use_video_port=True)
            logging.info("SHOTS FIRED\n")

            recognized = recognize_person(streams)
            if recognized is True:
                pass
            else:
                logging.info("Person not recognized.")
```

Listing 4: Code segment - basic cycle



Figure 2.4: Picture with recognized person

## 2. IMPLEMENTATION

---

```
def recognize_photo(stream):
    buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)           # Conver
    image = cv2.imdecode(buff, 1)                                         # Create

    response = get_boto_response(stream)

    if response is False:
        return False

    if len(response.get('FaceMatches')) == 0:
        logging.info('Face detected but not recognized.')
        cv2.imwrite(
            '/var/www/html/good/archiv/unknown/unknown' + time.strftime("%Y-%m-%d-%H
            image)
        return None

    person_id = response['FaceMatches'][0]['Face']['ExternalImageId']

    mark_image_with_response_data(response, image, person_id)

    cv2.imwrite('/var/www/html/good/uprava.jpg', image)

    url = 'good/archiv/' + time.strftime("%Y-%m-%d-%H:%M:%S", time.gmtime()) + perso
    cv2.imwrite('/var/www/html/' + url, image)
    send_id(person_id, 'http://147.32.69.139/' + url)
    return True
```

Listing 5: Code segment - recognize photo

```

def get_boto_response(stream):
    client = boto3.client(
        'rekognition',
        region_name=AWS_DEFAULT_REGION,
        aws_access_key_id=AWS_ACCESS_KEY_ID,
        aws_secret_access_key=AWS_SECRET_ACCESS_KEY
    )

    try:
        return client.search_faces_by_image(
            CollectionId='ciirc',
            Image={'Bytes': stream.getvalue()},
            MaxFaces=123,
            FaceMatchThreshold=50
        )

    except Exception as inst:
        cv2.imwrite('/var/www/html/bad/' + time.strftime("%Y-%m-%d-%H:%M:%S", time.g
        logging.info('No face in the picture.')
        return False

```

Listing 6: Code segment - recognize photo

```

def mark_image_with_response_data(response, image, person_id):
    b_h = response['SearchedFaceBoundingBox']['Height'] * HEIGHT
    b_w = response['SearchedFaceBoundingBox']['Width'] * WIDTH
    b_top = response['SearchedFaceBoundingBox']['Left'] * WIDTH
    b_left = response['SearchedFaceBoundingBox']['Top'] * HEIGHT

    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.rectangle(image, (int(b_top), int(b_left)), (int(b_top + b_h), int(b_left + b_w)), (0, 0, 0), 2)
    cv2.putText(image, person_id, (int(b_top - 20), int(b_left)), font, 1, (255, 255, 255), 1)

```

Listing 7: Code segment - paint square around face

### 2.3.2 Raspberry implementation

First of all, I made a systemd service from code face recognition, mainly from two reasons: autostart on boot and also for debugging.

systemd is a suite of basic building blocks for a Linux system. It provides a system and service manager that runs as PID 1 and starts the rest of the system. systemd provides aggressive parallelization capabilities, uses socket and D-Bus activation for starting services, offers on-demand starting of daemons, keeps track of processes using Linux control groups, maintains mount and automount points, and implements an elaborate transactional dependency-based service control logic. systemd supports SysV and LSB init scripts and works as a replacement for sysvinit. Other parts include a logging daemon, utilities to control basic system configuration like the hostname, date, locale, maintain a list of logged-in users and running containers and virtual machines, system accounts, runtime directories and settings, and daemons to manage simple network configuration, network time synchronization, log forwarding, and name resolution.[10]

I create configuration file located in `/etc/systemd/system/` folder, few interesting parts:

- `Wants=network.target` means that our service requires network connection
- `After=alsa-state.target` service should be start after sound system is running
- `Environment=AWS_ACCESS_KEY_ID=XXXX` systemd pass shared variable to the script
- `ExecStart=/usr/bin/python /home/pi/eclub-camera/motion.py` specify location of script
- `Restart=always` always when service crash will be automatically restarted

```
[Unit]
Description=Motion for raspberry
Wants=network.target
Before=network.target
After=alsa-state.target

[Service]
User=pi
Restart=always
RestartSec=3

Environment=HUB_TOKEN=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Environment=AWS_ACCESS_KEY_ID=XXXXXXXXXXXXXXXXXXXXXXXXXXXX
Environment=AWS_SECRET_ACCESS_KEY=XXXXXXXXXXXXXXXXXXXXXXXX
Environment=XDG_RUNTIME_DIR=/run/user/1000
Environment=AWS_CONFIG_FILE=/home/pi/.aws/config
Environment=AWS_SHARED_CREDENTIALS_FILE=/home/pi/.aws/credentials

ExecStart=/usr/bin/python /home/pi/eclub-camera/motion.py
WorkingDirectory=/home/pi/eclub-camera

[Install]
WantedBy=multi-user.target
```

Listing 8: Code segment - configuration of face recognition service

## 2.4 Attention manager

As mentioned at the beginning, I want to mix facial recognition + arp observation. For these purposes I used starpin project.

starpin (STealthy ARP INformer, developed under eClub) is a simple app meant to be run as a daemon on a networked device. starpin gets a fresh copy of the neighbor (ARP) table every once in a while and evaluates the result, e.g. a device that hasn't been in the table for several ticks is considered to have been disconnected, a device that goes stale and never drops out of the table is considered to be a 'rogue' entry and triggers a flush of the table etc.

starpin can send out two kinds of requests the DELETE request that tells the server to reset its current session and the POST request that informs the server of a client connecting/disconnecting that contains a JSON in the following format:

```
[
  {
    "event"      : "disconnect",
    "mac_address": "AA:AA:AA:AA:AA:AA",
    "ipv4_address": "192.168.0.1"
  },
  {
    "event"      : "connect",
    "mac_address": "BB:BB:BB:BB:BB:BB",
    "ipv4_address": "192.168.0.2"
  }
]
```

Listing 9: Code segment - starpin request

That's becoming useful with some information system, which collects these requests, and updates some user-table. For this particular purpose I have used Nucleus Hub project(<https://gitlab.com/eclubiot/django-nucleus-hub/>), which has been also developed in E-Club. Which combines these applications:

- attendance monitoring - backend for starpin
- webhook - backend for dialog flow
- accounts - profile page

Nucleus Hub is based on Django framework, which is Python framework. Each user which is in E-club and is connected in our Turrus network could

register via Google account(which everybody ows due to CTU emails), we are benefiting from Outh security and also has some access to any shared values - name, surname. There is also last image captured from the camera and registered device for observing arp table.


### Profile

**User information:**

First name:

Last name:

Email:   
We'll never share your email with anyone else.



**Devices:**

**Current device**  
 Device information  
**IPv4 address:** 89.248.248.219  
 Please check if you are connected to Wi-Fi **Turris**.

[Register this device](#)

ID	MAC address	Action(s)
39	ac:72:89:3c:77:d3	<a href="#" style="background-color: #dc3545; color: white; padding: 2px 5px; border-radius: 3px;">Remove</a>
42	40:6f:2a:f8:48:9a	<a href="#" style="background-color: #dc3545; color: white; padding: 2px 5px; border-radius: 3px;">Remove</a>

Figure 2.5: Nucleus Profile page

Also nucleus contains dashboard for actual presence with adjustable granularity of refreshing 1/5/10/15 seconds. Last usage of Nucleus is webhook implementation, it is used for fulfillment, briefly said when VA is asked how many people are inside, then because VA don't store attendance itself, send a POST request to nucleus, nucleus takes request and calls function `get_number_of_people` which looks for tables who is present and responds with parameter `fulfillment-Text`, which could be "There are currently three people". Another way how to communicate with Nucleus Hub is REST API, where you can simply list users, devices and entries, you can use GET method or Json.

## 2. IMPLEMENTATION

---

```
def get_number_of_people(params):
    present_users = User.objects.exclude(devices=None).filter(
        devices__entry__is_connected=True).distinct()

    if not present_users.exists():
        return {
            'speech': f'Nobody is currently present, the place is dark and lonely.',
            'displayText': f'Nobody is currently present, the place is dark and lonely.'
        }

    return {
        'speech': f'There are currently {len(present_users)} people.',
        'displayText': f'There are currently {len(present_users)} people.'
    }
```

Listing 10: Code segment - Nucleus hub webhook get\_number\_of\_people

**Present users:** 1s 5s 10s 15s  
Currently present club's users. 2018-03-05 15:40:17














#	Fullname	Active device(s)	Last photo
	Jakub Konrad	1	<a href="#">Link</a>
	Petr Marek	1	<a href="#">Link</a>
	Jan Uhlík		
	Jan Havránek		
	Jan Šedivý		
	Rostislav Kaufman		
	Mikuláš Formánek		
	Jan Presperín		
	Jan Pichl		
	Martin Matulík		
	Long Hoang Nguyen		
	drabekj.		
	Adam Ukleh		

Figure 2.6: Nucleus presence dashboard



## 2.5 Remote control of actuators

I wanted to implement some kind of lights controlling for people, so I found in our stock some devices from Jablotron namely: TP-83N(thermostat), AC-88(switch relay) also with included Turris Dongle, where you can remotely control these devices from Turris Omnia, however, I want to use rather RPI to control that, specially with voice interface, which Turris Omnia does not really offers...

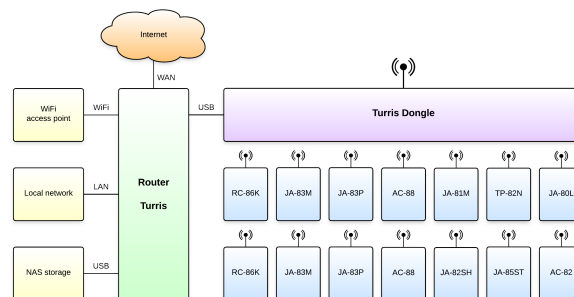


Figure 2.7: Turris scheme with jablotrone sensors;<sup>11</sup>

As long as we have new kitchen in E-club we had to change them and one that was occasion for implement voice control, I selected some lights and used library called jablotron-python-interface (also developed in E-club, main author is Alexej Popovič popoval2@fel.cvut.cz and I am co-author) which provides interface to control jablotron's devices from any machine with implemented python. Usage is really simple, because Turris Omnia communicates through serial link so you need to specify where is dongle located in `/dev/usbX` and bind events(thermostat reports, button switch) to functions.

## 2.6 Voice Assistant

To make own VA was dissatisfaction about current solution, I need to implement just some voice interface for our three or four functions on the same device which implements face-recognition.

First idea was to make own application to Google Home Assistant or Amazon Alexa, but first problem I realized was a fact you have to say attention word - cannot be triggered by motion sensor. Another problem is you have to firstly say: "Talk to my Attendace application" and that decided.

<sup>11</sup> Online 2018-04-29; [https://doc.turris.cz/gadgets/\\_media/bl\\_diagram.png?w=770&tok=665789](https://doc.turris.cz/gadgets/_media/bl_diagram.png?w=770&tok=665789)

<sup>11</sup> Online 2018-04-25; <https://www.jabloshop.cz/image/cache/catalog/jablotron/Regulace%20topen%C3%AD/83n-650x650.jpg>



Figure 2.8: TP-83N(thermostat), AC-88(switch relay);

```
dongle = devices.Dongle(port="/dev/ttyUSB0")
dongle.init()

def switch(event):
    sn = event.invoker.sn
    btn = event.data['btn']
    timestamp = event.timestamp

    print("{0} - btn {1} press from {2}.".format(timestamp, btn, sn))
    dongle.req_send_state(pgx="{0}".format(btn))

    events.bind(events.Event.ev_remote_btn_press, switch)
    dongle.req_send_state(pgx="0") # switch off
    dongle.req_send_state(pgx="1") # switch on
```

Listing 11: Code segment - jablon-python-interface usage

There are basically 2 problems that you have tackle: STT and Text to speech (TTS).

## 2.6.1 Speech to text implementation

There are plenty implementations of speech of text on the Internet, how ever it's mostly available as service. We take a look for options from Raspberry Pi side:

### 2.6.1.1 Google Cloud API

Google Cloud Speech-to-Text offers to translate audio in more than 120 languages, using modern neural networks, also could be used in real-time application.

Result of Speech-to-Text is also compensated with surroundings, for example when you are in the rush street Speech-to-Text can compute the noise around you. Google also implemented his speech-to-text, in some way into Android phone, that means you don't have to need Internet connection for realizing speech-to-text. Google also provides API for lot of platforms.

With Google you have first 60 minutes of using Speech-to-Text free, than 15 seconds is per 0.006 USD.

### 2.6.1.2 Bing Speech Api

Whichever approach developers choose (REST APIs or client libraries), Microsoft speech service supports the following:

- Advanced speech recognition technologies from Microsoft that are used by Cortana, Office Dictation, Office Translator, and other Microsoft products.
- Real-time continuous recognition. The speech recognition API enables users to transcribe audio into text in real time, and supports to receive the intermediate results of the words that have been recognized so far. The speech service also supports end-of-speech detection. In addition, users can choose additional formatting capabilities, like capitalization and punctuation, masking profanity, and text normalization.
- Supports optimized speech recognition results for interactive, conversation, and dictation scenarios. For user scenarios which require customized language models and acoustic models, Custom Speech Service allows you to create speech models that tailored to your application and your users. Support many spoken languages in multiple dialects. For the full list of supported languages in each recognition mode, see recognition languages.
- Integration with language understanding. Besides converting the input audio into text, the Speech to Text provides applications an additional capability to understand what the text

means. It uses the Language Understanding Intelligent Service(LUIS) to extract intents and entities from the recognized text.

### 2.6.1.3 IBM Watson Speech to Text API

IBM also offers their STT, but compared to previous listed their are quite limited, for example you can translate real-time just from 7 languages. And my personal experience wasn't so good, vice versa only them have also implementation through websockets, which are more powerful for real-time than HTTP requests.

### 2.6.2 Text to Speech

Each company mentioned in previous section also provides some text-to-speech, but there is one specialty: Google translator! This is really old feature, free, not so much configurable as other services, where you can choose person voice, intents and so on...

### 2.6.3 Usage

All you need to do is create GET request for url `https://translate.google.com/translate_tts?ie=UTF-8&tl=cs-CZ&client=tw-ob&q=Jak%C3%BD+to+%C3%BA%C5%BEasn%C3%BD+n%C3%A1stroj`

- argument tl - target language
- argument client - this is for bypassing captcha on Google side.
- argument q - is the query

Server simply responds with download able mp3 file, which you can simply run.

### 2.6.4 Usage in Python

## 2.7 Implementation on Raspberry Pi

For implementation simple VA I choose Python for developing. My idea was to create basic cycle which consists of:

1. Wait for interruption from motion sensor
2. Apply Speech to Text
3. Process text via Dialog Flow

```

def tts(q):
    q = q.replace(' ', '+')
    tr_url = "http://translate.google.com/translate_tts?ie=UTF-8
    &client=tw-ob&q=" + q + "&tl=en";
    content = req(tr_url);
    #print( content)
    file = open('tts.mp3', 'wb')
    file.write(content)
    file.close()
    mixer.init()
    mixer.music.load('tts.mp3')
    mixer.music.play()
    while mixer.music.get_busy() == True:
        continue

```

Listing 12: Code segment - TTS using Google Translator

4. Get response from previous step and make an interaction (Speech to Text, Light control)

For STT I choose Google Cloud API and for TTS my favorite Google Translator, but main think remains - how process the text.

### 2.7.1 Dialog Flow(ex API AI)

Dialog Flow (DF) is powerful tool, which even uses Google for their Home Assistants, it let you create dialogs, recognizing intents, storing the context - when told name of your pet in previous sentence, it stores it for future usage in dialog.

When creating intent you have to fill some training phrases, so it can use machine learning to recognizing similar input sentences. Also you can add entities - names, dates etc... Last think you have to mention are answers - what will be the answer for this. Another option are webhooks - all you have to do is create function on your “web-server” - in our case the Nucleus HUB, then Dialog Flow send there which intent and which phrase has user told to microphone and Nucleus HUB responds with Speech Synthesis Markup Language (SSML).

For recognizing text I used package <https://pypi.org/project/SpeechRecognition/>, which supports several STT API including Google STT.

## 2. IMPLEMENTATION

---

```
...
while(True):
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(7, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
    # Read output from PIR motion sensor
    GPIO.wait_for_edge(7, GPIO.RISING) # wait for interrupt
    #tts("Hi there how can help ? ")
    os.system("cvlc --play-and-exit /home/pi/chat-bot/hi.mp3")
    #welcome intent "Hi how can l help you"
    try:
        r = sr.Recognizer()
        #index = pyaudio.PyAudio().get_device_count() - 1;
        i=0
        while (i!=2):
            with sr.Microphone(2) as source:
                print("Say something!")
                audio = r.listen(source)
            # Speech recognition using Google Speech Recognition
            try:
                # break
                text_input = r.recognize_google(audio,language='cs');
                print("translate : " + text_input);
                ai = apiai.ApiAI(CLIENT_ACCESS_TOKEN);
                request = ai.text_request();
                request.lang = 'en' # optional, default value equal 'en'
                request.session_id = "<SESSION ID, UNIQUE FOR EACH USER>";
                request.query = text_input
                response = request.getresponse().read().decode('utf-8')
                text_response = response;
                j = json.loads(text_response);
                response_text = j["result"]["fulfillment"]['speech']; # get file
                tts(response_text)
                i = 0;
            except sr.UnknownValueError:
                i = i + 1;
                print("Google Speech Recognition could not understand audio")
```

Listing 13: Code segment - Voice Assistant - basic cycle

## 2.7. Implementation on Raspberry Pi

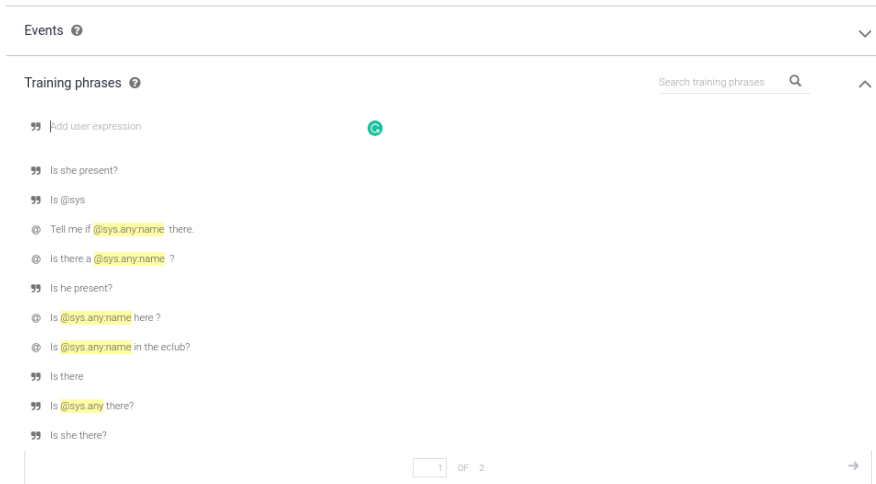


Figure 2.9: Dialog Flow - create intent

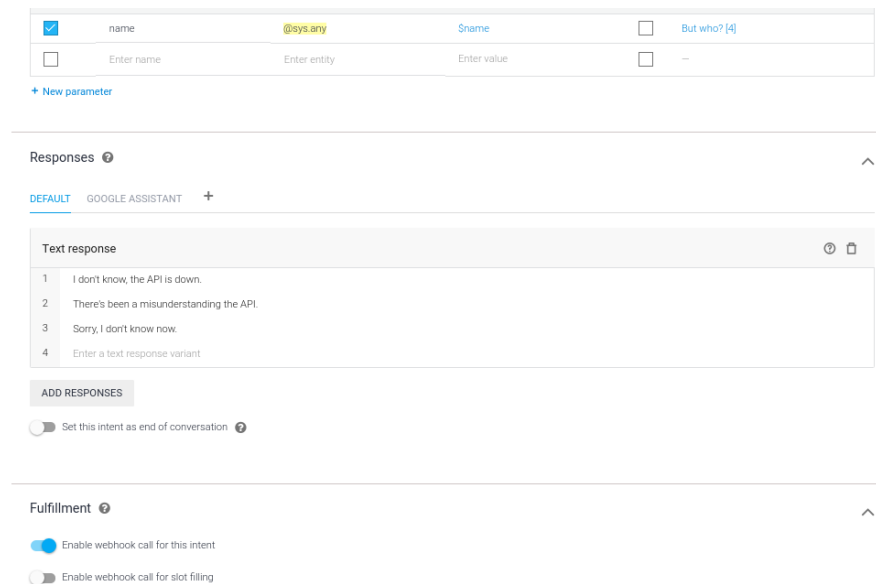


Figure 2.10: Dialog Flow - entities and answers





---

## Conclusion

In this thesis I tackle with implementation of monitoring attendance system, which is connected to simple Voice assistant, which also could control lights. Device is currently in CIIRC building and improves our office life at all. It also brings comparison of Single Board CPU and some theory behind Voice Assistant, especially Speech Recognition.

Although face recognition keeps running (over 9 months) we can say that system itself is degrading, caused probably by our visage changes(length of hair, sunglasses, clothes), this is point which would be improved in recent future - adding automatically new photos of members to AWS. Also there is lot of work on voice interface, I hope I can decrease response time and add some attention lights for user to respect that machine is now processing. Also I would like try to implement authentication trough voice, which seems to be very impressive.



---

## Bibliography

- [1] What is chat bot? - Definition from WhatIs.com. [Cited 2018-04-17]. Available from: <https://searchcrm.techtarget.com/definition/chatbot>
- [2] Olmstead, K. Nearly half of Americans use digital voice assistants, mostly on their smartphones. Dec 2017, [Cited 2018-04-18]. Available from: <http://www.pewresearch.org/fact-tank/2017/12/12/nearly-half-of-americans-use-digital-voice-assistants-mostly-on-their-smartphones/>
- [3] Ossola, A. Ever Wondered: How does speech-to-text software work? > Scienceline. Aug 2014, [Cited 2018-04-17]. Available from: <http://scienceline.org/2014/08/ever-wondered-how-does-speech-to-text-software-work/>
- [4] Gruhn, R. E.; Minker, W.; et al. Automatic Speech Recognition. In *Signals and Communication Technology*, Springer Berlin Heidelberg, 2011, pp. 5–17, doi:10.1007/978-3-642-19586-0\_2. Available from: [https://doi.org/10.1007/978-3-642-19586-0\\_2](https://doi.org/10.1007/978-3-642-19586-0_2)
- [5] Malý, M. Protokol MQTT: komunikační standard pro IoT. Jun 2016, [CITED 2018-05-01]. Available from: <https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/>
- [6] Beal, V. SBC - single-board computer. [CITED 2018-04-25]. Available from: [https://www.webopedia.com/TERM/S/sbc\\_single\\_board\\_computer.html](https://www.webopedia.com/TERM/S/sbc_single_board_computer.html)
- [7] Greenberg, M. Committed to Memory  $\iota$  When is LPDDR3 not LPDDR3? When it's DDR3L... Jan 2014, [CITED 2018-04-26]. Available from: <https://blogs.synopsys.com/committedtomemory/2014/01/10/when-is-lpddr3-not-lpddr3-when-its-ddr3l/>

## BIBLIOGRAPHY

---

- [8] Veen, B. V.; Buckley, K. Beamforming: a versatile approach to spatial filtering. *IEEE ASSP Magazine*, volume 5, no. 2, apr 1988: pp. 4–24, doi:10.1109/53.665. Available from: <https://doi.org/10.1109/53.665>
- [9] Voice. Available from: <https://matrix-io.github.io/matrix-documentation/matrix-hal/datasheets/voice/>
- [10] systemd System and Service Manager. [Cited 2018-04-22]. Available from: <https://www.freedesktop.org/wiki/Software/systemd/>

## Acronyms

**AWS** Amazon Web Service.

**BRPI** Banana Pi M64.

**DF** Dialog Flow.

**HTTP** Hyper Text Transfer Protocol.

**MQTT** MQ Telemetry Transport.

**NLP** Natural language processing.

**ORPI** Orange Pi Plus2E H3.

**RPI** Raspberry Pi 3.

**RRPI** ROCK64 3.

**SBC** Single Board Computer.

**SSML** Speech Synthesis Markup Language.

**STT** Speech to text.

**TTS** Text to speech.

**VA** Voice Assistant.



---

## Contents of enclosed CD

	readme.txt .....	the file with CD contents description
	src .....	the directory of source codes
	scripts .....	implementation sources - python
	configurations .....	implementation sources - systemd
	ltx .....	the directory of L <sup>A</sup> T <sub>E</sub> X source codes of the thesis
	text .....	video directory
	text .....	the thesis text directory
	thesis.pdf .....	the thesis text in PDF format