jamk.fi

Voice control of smart home by using Google Cloud Speech-To-Text API

Jozef Magdolen

Bachelor’s thesis
May 2018
School of Technology, Communication and Transport
Information and Communication Technology
Software Engineering

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s) Magdolen, Jozef	Type of publication Bachelor's thesis	Date May 2018
		Language of publication: English
	42	Permission for web publication: yes

Title of publication
Voice control of smart home by using Google Cloud Speech-To-Text API

Degree programme
Software Engineering

Supervisor(s)
Manninen, Pasi

Assigned by
-

Abstract

This work explains the process of creation two applications. The first application is an Android application that uses Speech-To-Text transcription by implementation of web service Google Speech API and Bluetooth communication as the client. The second application is a Windows application written in C# language using Bluetooth web service as the server. The applications show how Google Speech API works and how to implement these web services. The thesis consists of three main parts: Google Speech API, Bluetooth communication between the two devices, and the method to build an application using these features.

The first part describes various ways to convert Speech into text form using Google Speech API and its features. It also explains what Speech to text conversion means and how this technique works.

The second part describes the principle of operation of Bluetooth communication; how this communication works; how to implement Bluetooth service to a Windows application as the server and to an Android application as the client side. It also introduces its benefits but also its restrictions and its shortcomings.

The third part deals with creating of the Windows and Android applications. This part describes how to implement these services to the specific applications, what is needed to set and how these applications work.

The goal of the thesis has been met, both applications were created successfully and the whole system works correctly.

Keywords/tags
Android, Bluetooth, Google Speech API, Windows application, C#

Miscellaneous

Contents

Figures

Tables

Acronyms

AI	Artificial intelligence
API	Application Programming Interface
APK	Android Application Package
GHz	Gigahertz
gRPC	Remote Procedure Calls
ID	Identification number
IDE	Integrated Development Environment
ISM band	Industrial, scientific and medical radio bands
JSON	JavaScript Object Notation
Kb/s	Kilobit per second
MAC	Media Access Control
Mb/s	Megabit per second
MHz	Megahertz
mW	Milliwatt
RF	Radio Frequency
RFCOMM	Bluetooth transport protocol
SDK	Software Development Kit
UUID	Universally Unique Identifier
Wi-Fi	Wireless local area networking technology

1 Introduction

There are many services and extensions in the web, which deal with the fluency and simplicity of the use of Internet connection as well as applications based on Internet connection such as Facebook, Instagram, WhatsApp and others. Companies such as Google, Microsoft, Apple or Amazon are investing a great amount of resources to develop the best Voice assistant, which would able to respond to different commands.

The goal of this thesis was to create two applications, one of which could record a voice command, subsequently process it to text and send to another device for the next processing. The second application should be able to receive this voice command, process it and execute it if possible.

The second chapter is dedicated to Google Speech Cloud Speech-To-Text API, which provides the function of voice transcription into text form. The chapter shows different techniques of voice transcription and the options, which the Google service offers. Besides that, it shows, how to set an application to use this service.

How to create a Bluetooth service in an Android application or in a Windows application is explained in the third chapter. The third chapter also includes a brief overview through Bluetooth technology, comparison of the versions and available technologies.

In the fourth chapter all theoretical knowledge is put together and the process of implementation is explained. it is also shown, how both applications were developed and how these applications work.

2 Google Cloud Speech-to-Text API

2.1 Speech recognition services

The Speech recognition and its translation to digital text version is an area that many programmers and developers have been dealing with for some years. It is a wide area, which interferes with many scientific disciplines such as linguistics, mathematics and computer science. To reach correct results of spoken words or sentences complex diagnostic of voice is needed; for example, identification of language, recognition of intonation for correctly specifying of sentence patterns, or statistical processing of verbalism.

Speech recognition is generally applied to services, where is not convenient to use ordinary input methods. Besides this group of devices, voice control is also used in common devices as mobile phones or computers for the purpose of simplification, facilitation or making work more effective. Today’s smart devices are dispose among other things this feature. This group includes devices such as televisions, computers, mobile phones, watches, domestic appliances such as fridges, freezers, intelligent controlling of households, in-car systems, offices as well as devices in industry or development. Speech recognition is also used for people with some type of handicap to help them to create text files and documents. It is also used in military, especially in air forces, transportation, education or autonomous systems (Most Common Uses of Voice Recognition Software 2016).

There are many ways to create an application using some Speech to text APIs. Major tech companies such as Google, Microsoft, Amazon or Apple use their own algorithm for voice processing and invest considerable resources to improve it. These services are no longer just about voice recording and trying to transpose it to the text form. The algorithms of speech recognition are very sophisticated. That is one of the reason, why Artificial intelligence is used. The algorithms must be able to separate speech and noise in the background, find out not just correct language but also specific dialect with intonation. These services are built to become more like a personal assistant; they should become some “new member” of household in the future. AI can learn itself and so after a while, it is able to find out, when its user will come home, what to buy, how to set a thermostat and much more (The Past, Present, and Future of Speech Recognition Technology). It is not easy to build a software like this, and the best people in this field of research needed to be employed. To be more progressive and open for all possibilities of voice control, these companies offer their services for other developers, however, of course, with some restrictions, e.g. such as the length of one record or number of free minutes on their servers in a month.

There are three most used methods for Speech recognition (A Review on Different Approaches for Speech Recognition System 2018):

- HMM – Hidden Markov Model
- DTW – Dynamic Time Warping
- Neural networks

"A Hidden Markov Model is a type of graphical model often used to mode temporal data. The HMM assume that the data observed is not the actual state of the model, but instead generated by the underlying hidden states. Because of the flexibility and computational efficiency, HMM have found wide application in many different fields. They are known for their use in temporal pattern recognition and generation, such as speech recognition, handwriting recognition, and speech synthesis." (Hidden Markov Models 2018).

Dynamic Time Warping is an algorithm for measuring similarity between two sequences that may vary in time or speed. It is possible to compare it to two video records. For instance, similarities in walking patterns would be detected, even if in one video the person walked slowly and if in the other video the person was walking more quickly, or even if there were accelerations and deceleration during one observation. In general, DTW is a method that allows a computer to find an optimal match between two given sequences with certain restrictions (International Journal of Computer Applications 2015).

Simply put, while HMMs uses mechanism of parsing of voice signal and probability of occurrence parts of sentence, DTW uses recognition of individual words.

Neural networks are capable of solving more complicated recognition tasks but could not perform as excellent as HMM when it comes to large vocabularies. Neural network technology is used due the following reasons; It reduces the modeling unit to advance the recognition rate, depth learning, can be used to develop combine a hybrid system. This method of speech recognition has been significantly developed in recent years (Ibid).

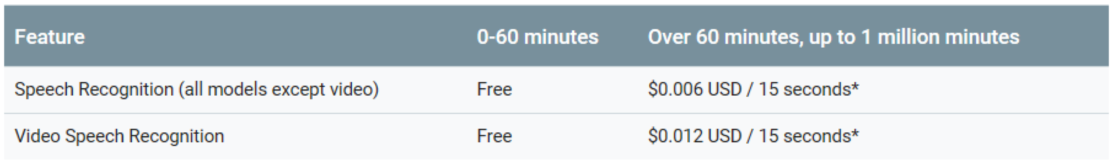
2.2 About Google Cloud Speech API

Google Cloud Speech API enables developers to convert audio to text by applying powerful neural network models in an easy to use API. The API recognizes over 120 languages and variants to support a global user base. It also allows transcribing the users' text by dictating to an application's microphone. It also filters inappropriate content in text results for all languages, enables command-and-control through voice, or transcribes audio files, among many other use cases. Google Cloud Speech service enables to recognize the audio uploaded in the request, and integrates it to audio storage on Google Cloud Storage by using the same technology Google uses to power its own products (Documentation to Cloud Speech-to-Text API 2018).

Google Cloud Speech API applies most advanced deep learning neural network algorithms on user's audio for speech recognition with high accuracy. The advantage is that Speech API accuracy improves over time as Google company improves the internal speech recognition technology and software (Ibid).

One of the features is that Speech API can stream text results (return immediately), as they become available, so the recognized text appears immediately while speaking. There is also option to recognize text from audio file. The Google Cloud service can also handle noisy audio, so there is no need to filter audio record before processing. This service has been available for third-part developers since summer 2016, when Google allowed access to its speech recognition technology. After this step, Google Speech Cloud API has become the most used web recognition service (Ibid).

Regarding pricing, the Google Cloud Speech API is priced monthly on the amount of audio or video successfully processed by the service, measured in increments rounded up to 15 seconds (Figure 1). For example, three separate requests, each containing 7 seconds of audio are billed as 45 seconds (3 x 15 seconds) of audio. Fractions of seconds are included, when rounding up to the nearest increment of 15 seconds. That is, 15.14 seconds are rounded up and billed as 30 seconds. Pricing tiers are based on the total amount of audio processed by the service per month (Documentation to Cloud Speech-to-Text API 2018).



Feature	0-60 minutes	Over 60 minutes, up to 1 million minutes
Speech Recognition (all models except video)	Free	$0.006 USD / 15 seconds*
Video Speech Recognition	Free	$0.012 USD / 15 seconds*

Figure 1. Pricing of Google Speech Cloud service

2.3 Types of Speech requests

The Speech API has three main methods to perform speech recognition (Ibid):

- Synchronous Speech Recognition
- Asynchronous Speech Recognition
- Streaming Speech Recognition

Synchronous Speech Recognition returns recognized text for short audio files (less than ~1 minute) in the response as soon as it is processed. Audio content can be sent directly to Cloud Speech-to-Text, or it can process audio content already residing in Google Cloud Storage. Synchronous Speech Recognition is suitable for shorter audio records or for audio records stored locally because this method of Speech Recognition is the fastest and the simplest (Documentation to Cloud Speech-to-Text API 2018).

Asynchronous Speech Recognition starts a long running audio or video processing operation. Asynchronous Speech Recognition is used to recognize audio or video file, which is longer than a minute. Audio content can be sent directly to Cloud Speech-to-

Text, or it can process audio content that already resides in Google Cloud Storage (Ibid).

Streaming Speech Recognition allows streaming audio to the Cloud Speech API and receiving the stream of speech recognition results in real time as the audio is processed. Streaming Speech Recognition is available only via gRPC. gRPC is explained later (Ibid).

The application is based on Streaming Speech Recognition method. The reason of using this method of speech recognition is that user can say such as short so long or multiple commands as well and the process of translating speech to text should be as fast as possible. The best way how to ensure this feature is to use Streaming Speech Recognition method, where the results are returned immediately.

2.4 Streaming Speech Recognition and gRPC

To work with real-time applications such as recording live audio from a microphone the streaming speech recognition has been developed. This method returns interim results while an audio sample is recorded, therefore allows a result to appear, while a user is speaking. Streaming recognition method uses gRPC bi-directional stream for recognition on an audio data (Cloud Speech-to-Text, Basic 2018).

gRPC (Remote Procedure Calls) is a modern open source remote procedure call system initially developed at Google runs on HTTP/2 network protocol. It enables client and server applications to communicate transparently and makes it easier to build connected systems. As in many RPC systems, gRPC is based around the idea of defining a service, specifying the methods to be called remotely with their parameters and return types. On the server side, the server implements this interface and runs a gRPC server to handle client calls. On the client side, the client has a stub that provides the same methods as the server (gRPC Overview 2018).

The communication between server and client side is shown in Figure 2.

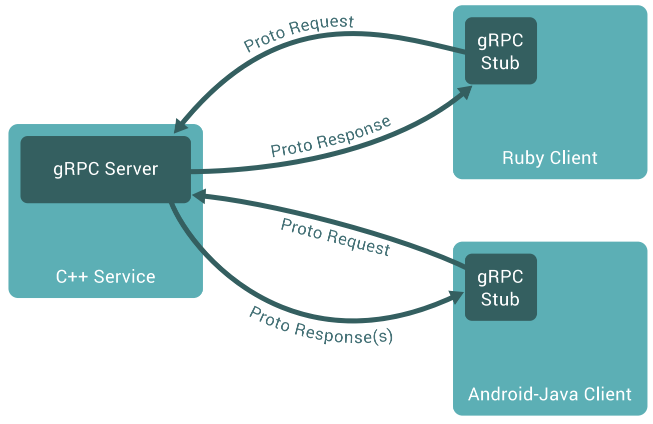
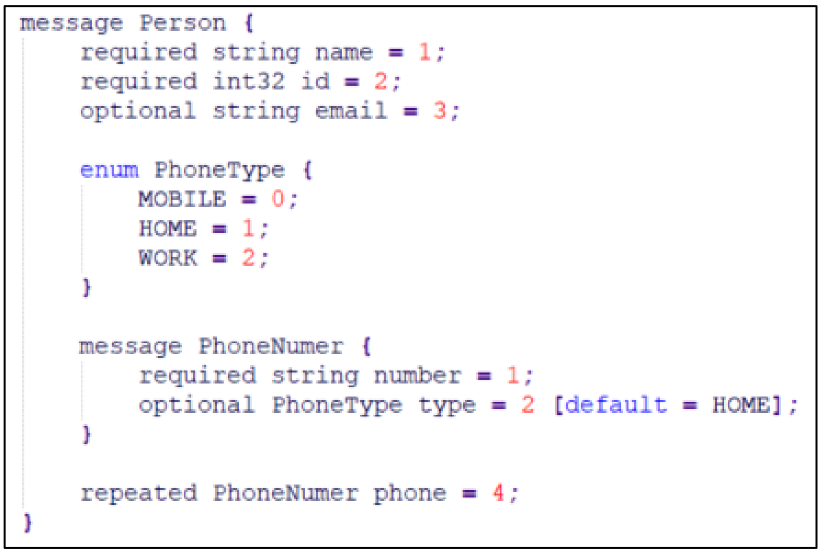


Figure 2. gRPC Client - Server communication

gRPC clients and servers can run and talk to each other in a variety of environments and can be written in any of gRPC supported languages thus for example, there can be gRPC server side created in JAVA language and the client side in Go, Python or C++ language. By default, gRPC uses protocol buffer, Google's mature open source mechanism for serializing structured data, however, it can be used also with other data formats, such as JSON (Ibid).

Protocol buffers are a flexible, efficient, automated mechanism for serializing structured data. Each protocol buffer message is a small logical record of information, containing a series of name-value pairs called fields. This information is saved in .proto file, as below is shown (gRPC Developer Guide 2018).

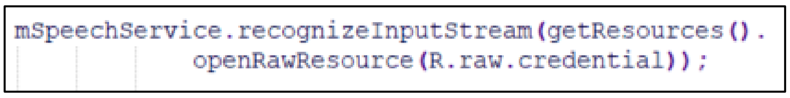

```
message Person {
    required string name = 1;
    required int32 id = 2;
    optional string email = 3;

    enum PhoneType {
        MOBILE = 0;
        HOME = 1;
        WORK = 2;
    }

    message PhoneNumer {
        required string number = 1;
        optional PhoneType type = 2 [default = HOME];
    }

    repeated PhoneNumer phone = 4;
}
```

2.5 How to build Google Cloud Speech Application

2.5.1 Implementation of Google Cloud Speech library

To enable API speech in an application, there has to be a unique API key first and next service account key. These keys are used in an application for authentication to gain access to Google server and to use with Google Cloud Speech service. To gain these keys, sign-in to Google Cloud Platform is needed. There is necessary to set up a project first and choose right service. There are two options, how to use Service Account Key. It is possible to download this key as JSON format or P12 format, which is there for backward compatibility. Before downloading the Key, a user is redirected to Google page, where is required to fill in personal information and information about credit card as well, in the case of data limit overdrawing. The downloaded key contains information about users account as well as project id, generated private key, client id, security key and other important information for server (Documentation to Cloud Speech-to-Text API 2018).

The next code shows, how to implement .json credential file to an Android application.



```
mSpeechService.recognizeInputStream(getResources().
        openRawResource(R.raw.credential));
```

There is Google Cloud API v1 Client library, which allows developers a connection to Google Cloud service and use of all Google Speech-to-Text features. The library is free and there are more options on how to implement it to a project. The user can choose whether to use this library as a downloaded .zip file, define it in Android Maven file or in Gradle file (Ibid).

In this case, Google Cloud API v1 Client library is used and defined in Android build.grandle file.

As first, the new plugin must be added. The plugin specifies type of the project and indicates which plugin should be used.

```
apply plugin: 'com.google.protobuf'
```

After that the path of the class must be added to inform the system, where the project and used libraries are situated and.

```
classpath 'com.google.protobuf:protobuf-gradle-plugin:0.8.3'
```

To ensure that protocol buffer will used the plugin to the module needs to be add and the protbuf-lite library as well.

```
implementation 'com.google.protobuf:protobuf-lite:3.0.0'

protobuf {
    protoc {
        artifact = 'com.google.protobuf:protoc:3.0.0'
    }
    plugins {
        javalite {
            artifact = 'com.google.protobuf:protoc-gen-javalite:3.0.0'
        }
    }
    generateProtoTask {
        all().each { task ->
        task.builtins {
            remove java
        }
        task.plugins {
            javalite { }
        }
        }
    }
}
```

“A streaming Speech-to-Text API recognition call is designed for real-time capture and recording of audio within a bi-directional stream. That means that the application can send audio on the request stream and receive interim and final recognition results on the response stream in real time. Interim results represent the current recognition result for a section of audio, while the final recognition result represents the lasts, best guess for that section of audio.” (Ibid). This is an advantage of gRPC service because bidirectional communication cannot be used in REST service.

2.5.2 Streaming requests

In contrast with synchronous and asynchronous communication, where it is possible to send an audio sample and a configuration within a single request, Streaming Speech recognition requires to send these items in multiple requests. The first request – StreamingRecognizeRequest must contain a configuration without any added audio sample. As follow another request – StreamingRecognizeRequest is sent over the same stream and this request contains raw audio bytes (Documentation to Cloud Speech-to-Text API 2018).

A StreamingRecognitionCofnig consists of the following fields (Ibid):

- config – (required) contains configuration information about the audio sample
- SingleUtterance – (optional, defaults to “false”) indicates whether the request should end immediately after any speech is no longer detected. If is set, after Speech-to-Text will detect silence or pause in the recorded audio, the recognition will end. If it is not set, the stream continues with listening and processing an audio, until the stream is closed directly, or the limit of the stream recognition is reached. This implies, that to set single_uterrance to true is advisable for processing voice commands.
- InterimResults – (optional, defaults to “false”) by the setting this field to “true”, temporary results will be return, and these results may be modified after processing more audio. By setting InterimResults to “false”, temporary results will be noted inside responses by the setting of is_final to “false”

Used StreamingRecognitionConfig method is shown in the following code snippet.

```
StreamingRecognitionConfig streamingConfig =
        StreamingRecognitionConfig.new Builder()
        .setConfig(config)
        .setInterimResults(true)
        .setSingleUtterance(false)
        .build();
```

2.5.3 Streaming responses

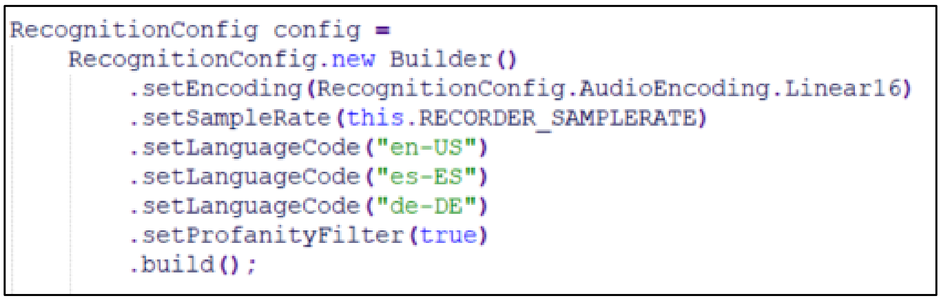
Streaming speech recognition results are returned inside a several of responses of type StreamingRecognizeResponse and comprise of following fields (shown in the next code snippet) (Documentation to Cloud Speech-to-Text API 2018):

- speechEventType – "value of this event indicates when a single utterance has been determined to have been completed."
- result – contains the list of results, which were returned and may be interim or final. This field contains of following sub-fields:
  - alternatives – includes a list of alternative transcriptions
  - isFinal – notifies whether the returned results inside this list are interim or final
  - stability – indicates defectiveness of obtained results, where value 0.0 indicates total instability while value 1.0 indicates entire stability

```
@Override
public void onNext(StreamingRecognizeResponse response) {
    int numOfResults = response.getResultsCount();

    if (numOfResults > 0) {
        for (int i = 0; i < numOfResults; i++) {
            StreamingRecognitionResult result = response.getResultList().get(i);
            String text = result.getAlternatives(0).getTranscript();

            if (result.getIsFinal()) {
                Log.d("Final", text);
                mScreen.onFInal(text);
            } else {
                Log.d("Partial", text);
                mScreen.onPartial(text);
            }
        }
    }
}
```

Besides these settings and attributes, it is possible in the configuration of API to set more important properties such as audio encoding, sample rate, supported languages or profanity filter as well. In this case, every profanity word is replaced by asterisks, except the first character (Ibid).

The setting of these attributes is the next code snippet.



```
RecognitionConfig config =
    RecognitionConfig.new Builder()
        .setEncoding(RecognitionConfig.AudioEncoding.Linear16)
        .setSampleRate(this.RECORDER_SAMPLERATE)
        .setLanguageCode("en-US")
        .setLanguageCode("es-ES")
        .setLanguageCode("de-DE")
        .setProfanityFilter(true)
        .build();
```

3 Bluetooth service

3.1 Overview

Bluetooth is a short-range wireless communication technology standard allows to exchange data over short distances using RF transceiver operates in the unlicensed ISM band (How does Bluetooth work? 2017). Bluetooth technology was invented by telecom vendor Ericsson in 1994 as an idea to replace the cables connecting electronic devices trough RS-232 data cables. Now the Bluetooth service is managed by the Bluetooth Special Interest Group (SIG), that publishes Bluetooth specification, administers the qualification program and administrates Bluetooth wireless technology. SIG is a global community of over 30,000 member companies (Learn about the history of and people behind Bluetooth SIG 2018). Due to low price, small dimensions and low consuming of electricity, Bluetooth devices have become globally very popular

3.2 Operating bands, speed and topology

The Bluetooth RF transceiver operates in the unlicensed ISM band centered at 2.4GHz, the same range of frequencies as used by microwaves and Wi-Fi. Hence, it supports multiple radio options that enable developers to build products with the unique connectivity. Bluetooth uses radio technology called Frequency-Hopping-Spread-Spectrum (FHSS). This technology divides transmitted data into packages and transmit each packet on one of the 79 designed Bluetooth channels. Each channel has bandwidth of 1 MHz. During the data transmission, up to 1600 hops per second occur. The adaptive hopping technique improves Bluetooth technology's coexistence with static (non-hopping) ISM systems when these are located in the vicinity of piconet (How does Bluetooth work? 2017).

Bluetooth provides two solutions of connection depending on power consuming (Bluetooth Radio Versions 2018):

- Bluetooth Low Energy (LE) – designed for very low power operation. To enable reliable operation in the 2.4 GHz frequency band, it leverages a frequency hopping spread spectrum approach, that transmits data over 40 channels. The Bluetooth Low Energy radio provides data rate from 125 Kb/s up to 2 Mb/s and power levels from 1 mW to 100 mW

- Bluetooth Basic Rate/Enhanced Data Rate – designed for low power operation and leverages Adaptive Frequency Hopping approach, transmitting data over 79 channels. Bluetooth Basic Rate radio supports data transfer rate from 1 Mb/s to 3 Mb/s and power levels from 1 mW to 100 mW. This technology supports a point-to-point network topology, that is optimized for audio streaming.

From the first Bluetooth version much has changed in speed rate and range. Table 1 shows some differences between individual versions (Bluetooth 4 vs Bluetooth 5: A Future comparison 2017; How does Bluetooth work? 2017).

Table 1. Differences in speed rate between different Bluetooth versions

Version	Speed rate [Mbit/s]
1.2	1
2.0 + EDR	3
3.0 + HS	24
4.0 + LE	24
5.0	48

To best meet the wireless connectivity a diverse developer population is needed. Bluetooth technology supports multiple topology options, from simple point-to-point connections to broadcast connections and connection called mesh (Bluetooth Topology Options 2018).

- Point-to-point is a network topology used for establishing one-to-one device communication. The point-to-point topology available on Bluetooth Basic Rate is optimized for audio streaming and is ideally suited for a wide range of wireless devices, such as speakers, headsets or handsfree car kits. The point-to-point topology available on Bluetooth LE is optimized for data transfer and is well suited for connected device products, such as fitness trackers, health monitors or PC peripherals and accessories.

- Broadcast is the network topology used for establishing one-to-many device communication. The broadcast topology available on Bluetooth LE is optimized for localized information sharing and is ideal for location services such as retail point-of-interest information and indoor navigation as well as asset tracking.

- Mesh is a network topology used for establishing many-to-many device communication. The mesh topology available on Bluetooth LE enables the creation of large-scale device networks and is ideally suited for control, monitoring and automation systems, where more devices need to reliably and securely communicate with each other.

3.3 Implementation of Bluetooth service in Android and Windows applications

3.3.1 Basic information

“Android platform includes support for Bluetooth network stack, which allows a device to wirelessly exchange data with other Bluetooth devices. The application framework provides access to the Bluetooth functionality through Android Bluetooth APIs.”. The Bluetooth API performs many actions; scanning for other Bluetooth devices or for already paired Bluetooth devices, connecting to other, transferring data, establishing RF channels and managing multiple connections (Bluetooth Overview 2018).

“In order for Bluetooth-enabled devices to transmit data between some other device, the device must first form a channel of communication using a pairing process.”. The process is as follow: first, a discoverable device must set up itself as “available device” for incoming connection requests. By using a service discovery process can other devices find the discoverable device. After the discoverable device accepts request to pair with the device, the bonding process is finished, security keys and the information about devices are exchanged. “When the session is complete, the device that initiated the pairing request releases the channel that had linked it to the discoverable device, however both devices remain bonded, so they can reconnect automatically during a future session as long as they are within each other’ range” and neither device has removed the bond (Ibid).

It is important to remember that there is a difference between being paired and being connected. “To be paired means, that two devices are aware of each other’s existence, have a shared link-key used for authentication and can establish an encrypted connection with each other. To be connected means that the devices currently share an RFCOMM channel and are able to transmit data with each other.”. It is required to be paired before an RFCOMM connection is established (ibid).

In order to use Bluetooth features in an application, it is necessary to declare permissions in the Android Manifest file. The first of these is BLUETOOTH. This permission allows a device to communicate via Bluetooth; performing of connection requests, accept this requests or transfer data. The next permission is ACCESS_COARSE_LOCATION. This permission is required because Bluetooth scans can collect information about the user location. And as last, BLUETOOTH_ADMIN permission is required. This one enables to initiate device discovery or manipulate with Bluetooth settings (Ibid).

Which permissions are needed and are illustrated in the next code snippet.

```
<manifest ... >
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    ...
</manifest>
```

Before an application can communicate over Bluetooth, it is needs to be verified whether the device supports Bluetooth service. If so, to ensure it is enabled.

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) {
    //Device doesn't support Bluetooth
}
```

If Bluetooth is not supported, then is needed to disable the Bluetooth service; if the device supports Bluetooth service, but it are disabled, then is possible to set up request that the user enables or disables without leaving a current application.

```
if (!mBluetoothAdapter.isEnabled()) {
    Intent enableByIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

The Figure 3 shows the dialog windows invoked by the Bluetooth request to allow or deny turn on the Bluetooth.

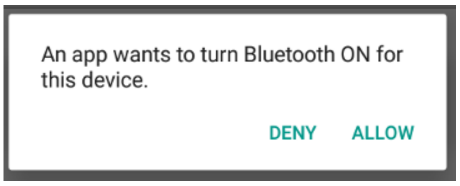

An app wants to turn Bluetooth ON for this device.

Figure 3. Dialog window to allow Bluetooth service

In order to receive information about each device discovered, a BroadcastReceiver has to be registered first for the intent ACTION_FOUND. After that, this intent is broadcasts by the system to each device. How to register the BroadcastReceiver is shown in the next code snippet.

```
//Create a BroadcastReceiver for ACTION_FOUND
private BroadcastReceiver mBroadcastReceiver3 = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        final String action = intent.getAction();
        Log.d(TAG, "onReceive: ACTION FOUND.");
        if (action.equals(BluetoothDevice.ACTION_FOUND)) {
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            if (!mBTDevices.equals(device)) {
                mBTDevices.add(device);
                Log.d(TAG, "onReceive: " + device.getName() + ": " + device.getAddress());
                mDeviceListAdapter = new DeviceListAdapter(context, R.layout.device_adapter_view,
                                        mBTDevices);
                lvNewDevices.setAdapter(mDeviceListAdapter);
            }
        }
    }
};
```

To initiate a Bluetooth connection between two devices, all that is needed from the associated BluetoothDevice object is a MAC address. This address is returned by calling getAddress() method. In order to create a connection between two devices, it is necessary to implement server and client side as well because “one device must open the server socket and the other one must initiate the connection using the server device’s MAC address.”. After the connection is accepted, the server receives information from the client about the socket. The client provides socket information after RFCOMM channel is opened to the server. When both (client and server) devices have connected Bluetooth socket on the same RFCOMM channel, they are considered connected, and data transfer can begin (Ibid).

3.3.2 Server side of Bluetooth windows application

The role of the server is to listen for an incoming connection by holding an open Bluetooth server listener. After connection, the server listener should be discarded if the user wants the device to accept more connections. To uniquely identify a Bluetooth service, it is necessary to use a specific ID. This ID is called Universally Unique Identifier – UUID and it is standardized as 128-bit format string ID. UUID is big enough that random generated ID cannot have the same ID with other application.

UUID can be defined in the code as next code snippet shows.

```
Guid mUUID = new Guid("00001101-0000-1000-8000-00805f9b34fb");
```

To set up a server listener and accept a connection are needed following steps:

- get a BluetoothListener by calling new BluetoothListener()
- start listening for connection requests by calling Start()

How to implement all these steps is shown below.

```
BluetoothListener btListener = new BluetoothListener(localAddress, mUUID);
btListener.Start();
```

After a listener detects new connection and connection passed successfully, there a stream is sets and the receiving data can be stored in a buffer and so shown in the received text box.

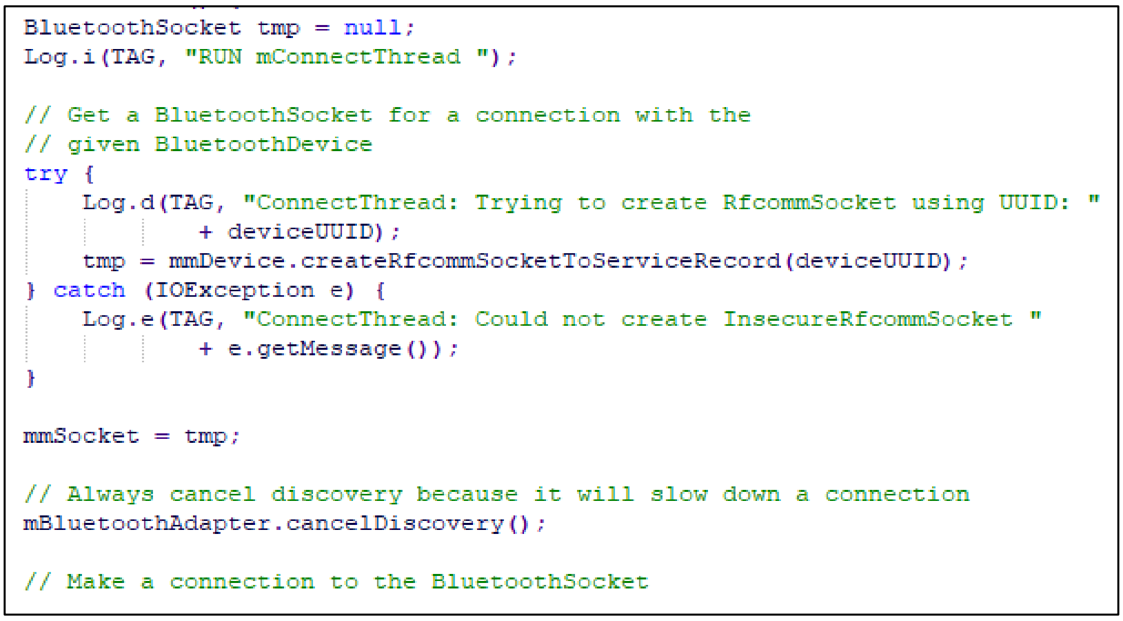
3.3.3 Client side of Android application

To initiate connection with a remote device that is accepting a connection on an open server socket, it must be BluetoothDevice object obtained first.

To set up the client side of Bluetooth connection service is needed to implement following steps (Bluetooth Overview 2018):

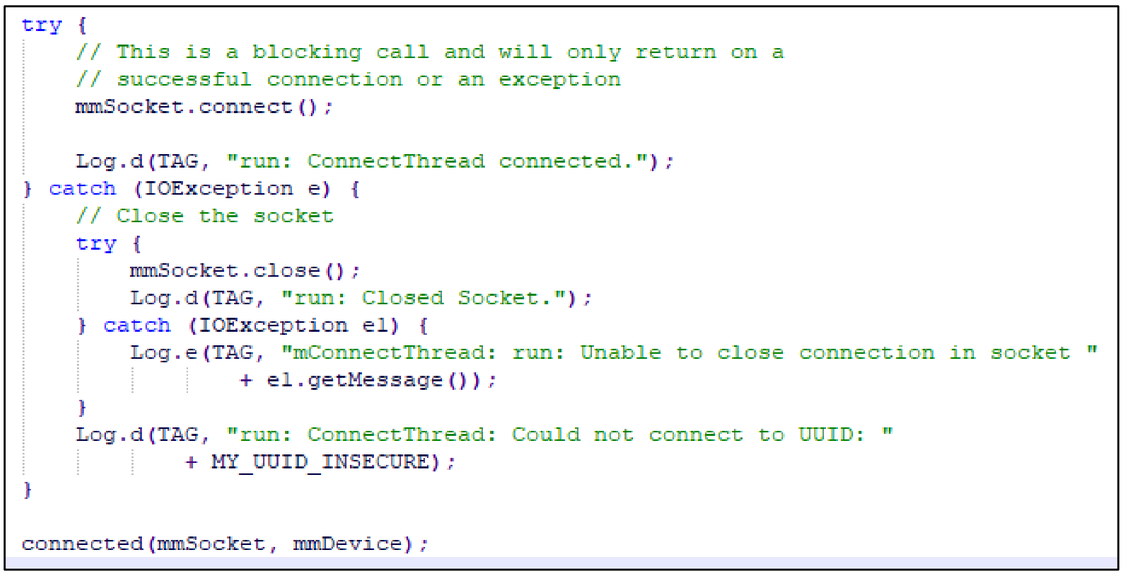
- to use BluetoothDevice object and get a BluetoothSocket by calling createRfcommSocketToServiceRecord(UUID)
- Initiate connection by calling connect()

After the Bluetooth socket is given, discovering of new devices should be canceled, because it will slow down a connection (see the code snippet below).



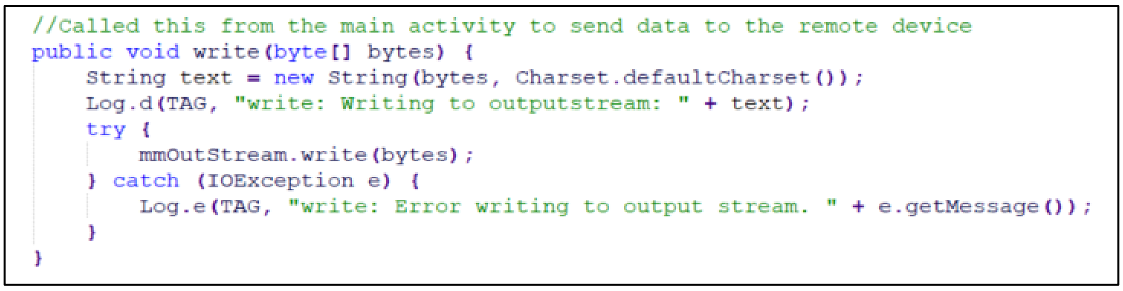
```
BluetoothSocket tmp = null;
Log.i(TAG, "RUN mConnectThread ");

// Get a BluetoothSocket for a connection with the
// given BluetoothDevice
try {
    Log.d(TAG, "ConnectThread: Trying to create RfcommSocket using UUID: "
            + deviceUUID);
    tmp = mmDevice.createRfcommSocketToServiceRecord(deviceUUID);
} catch (IOException e) {
    Log.e(TAG, "ConnectThread: Could not create InsecureRfcommSocket "
            + e.getMessage());
}

mmSocket = tmp;

// Always cancel discovery because it will slow down a connection
mBluetoothAdapter.cancelDiscovery();

// Make a connection to the BluetoothSocket
```

If the attempt to connect to another device times out or it fails, the system will try to close the given socket. If this try fails, an exception will be throw by the system, that the connection ended by an error. That is shown in the next code snippet.



```
try {
    // This is a blocking call and will only return on a
    // successful connection or an exception
    mmSocket.connect();

    Log.d(TAG, "run: ConnectThread connected.");
} catch (IOException e) {
    // Close the socket
    try {
        mmSocket.close();
        Log.d(TAG, "run: Closed Socket.");
    } catch (IOException e1) {
        Log.e(TAG, "mConnectThread: run: Unable to close connection in socket "
                + e1.getMessage());
    }
    Log.d(TAG, "run: ConnectThread: Could not connect to UUID: "
            + MY_UUID_INSECURE);
}

connected(mmSocket, mmDevice);
```

After the connection is established successfully, each device has a connected BluetoothSocket. After that, the general procedure to transfer data begins by setting getInputStream() and getOutputStream() methods that handle transmissions through the socket. To write data to the stream is possible by using write(byte[]) methods as seen in the code snippet below.



```
//Called this from the main activity to send data to the remote device
public void write(byte[] bytes) {
    String text = new String(bytes, Charset.defaultCharset());
    Log.d(TAG, "write: Writing to outputstream: " + text);
    try {
        mmOutStream.write(bytes);
    } catch (IOException e) {
        Log.e(TAG, "write: Error writing to output stream. " + e.getMessage());
    }
}
```

4 Developing of Application

4.1 Introduction

The developed application consists of two parts; namely, the Android application and the Windows application Both applications work separately and are independent of the used technologies and the running environment. The communication between these two applications provides Bluetooth service.

The Windows application is written in programming language C# and uses 32feet.NET library which allows fast use of Bluetooth service. This application is designed to provide the server side of Bluetooth service. The application is developed in IDE called MS Visual Studio 2015, which offers many tools that help during coding or code refactoring.

The Android application is developed as the client side of Bluetooth service and as programming language JAVA language was used. The application was developed in IDE called Android Studio. It is an official integrated development environment for Google's Android operating systems, so it provides many useful tools for development of an Android application. As mentioned, Google Speech Cloud was used as speech-to-text translator. How is build entire system and how it works is illustrated in Figure 4.

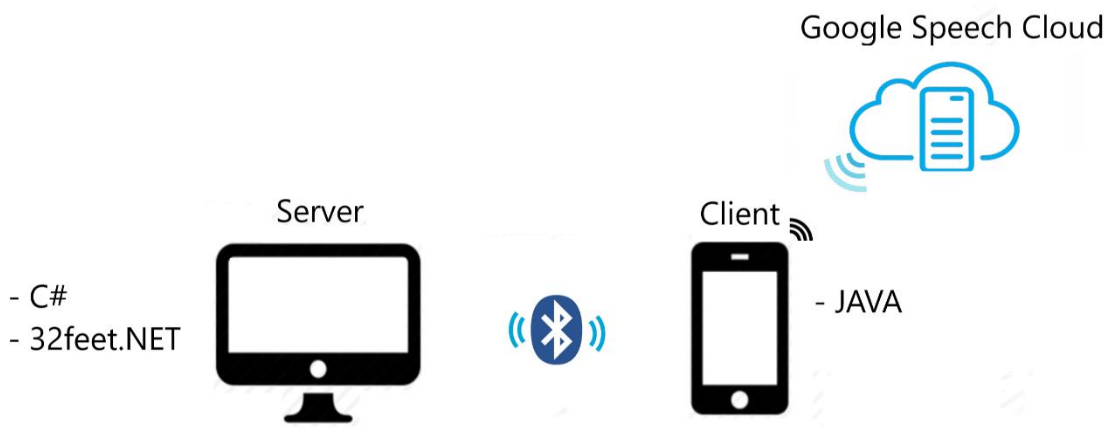


Figure 4. The application's schema

4.2 Windows application

4.2.1 Overview

“Microsoft Visual studio is a suite of tools for creating software, from the planning phase through UI design, coding, testing, debugging, analyzing code quality and performance, deploying to customers, and gathering telemetry on usage. These tools are designed to work together as seamlessly as possible.” (Visual Studio IDE 2015).

“Visual Studio by default provides support for C#, C and C++, JavaScript, F# and Visual Basic. That enables developers to create many kinds of applications, from single store applications and games for mobile clients, to large, complex systems that power enterprises and data centers. This enables to create applications and games that run on not only Windows operation systems but also Android and iOS platform. It is possible to create websites and web services as well based on ASP.NRT, JQuery, AngularJS and other popular frameworks.”. Besides that, developers can create applications for platform and devices as diverse as Azure, Office, SharePoint, virtual and augmented reality, Internet of Things or applications using high graphical performance such as Xbox games, using DirectX module. Visual Studio enables also to sign in to users Microsoft, work or school account. That allows a developer to synchronize settings and windows layouts across multiple devices (Ibid).

4.2.2 The project configuring

Before a user wants to start with coding an application, is a project needs to be created. “A Visual Studio project is a collection of files and resources that are compiled to a single binary executable file for application; such as .an, .exe, .DLL and others. Visual Studio has a concept of the solution, which can contain multiple projects or websites.” (Visual Studio IDE 2015). When a first project is created, the solution is also created. Thus, it is possible to add more projects to that solution if needed. The project is created by clicking in the left top corner on item “File”->”New”->”Project”. After that, the user can choose form plenty of different programming languages, frameworks and application forms as shown in Figure 5. The Bluetooth server application was created as Windows Forms Application.

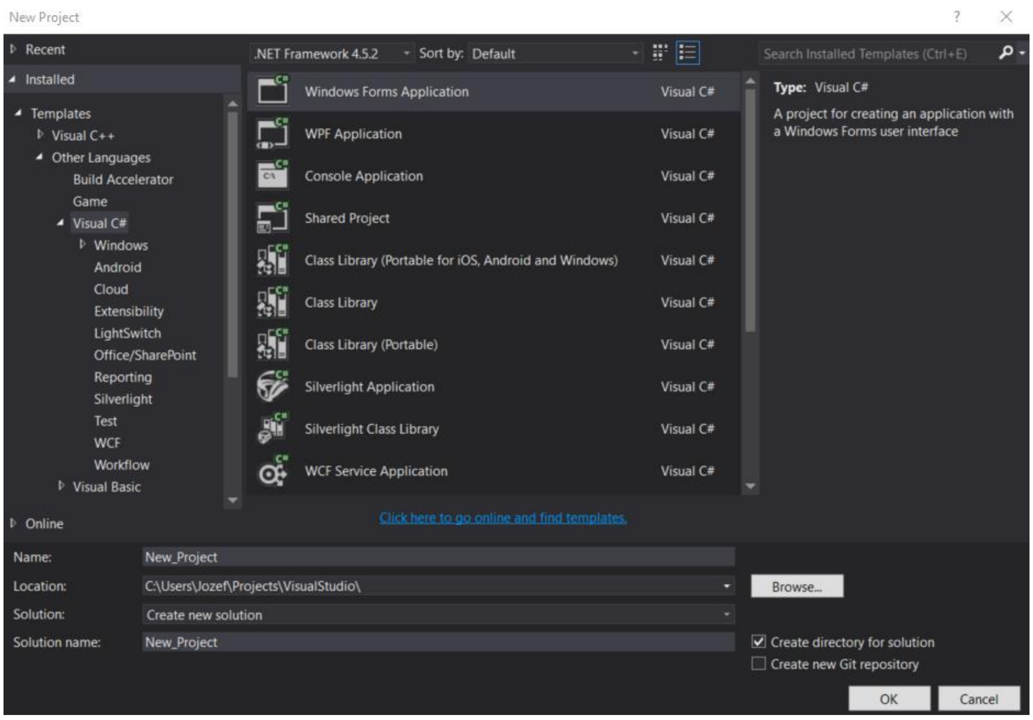
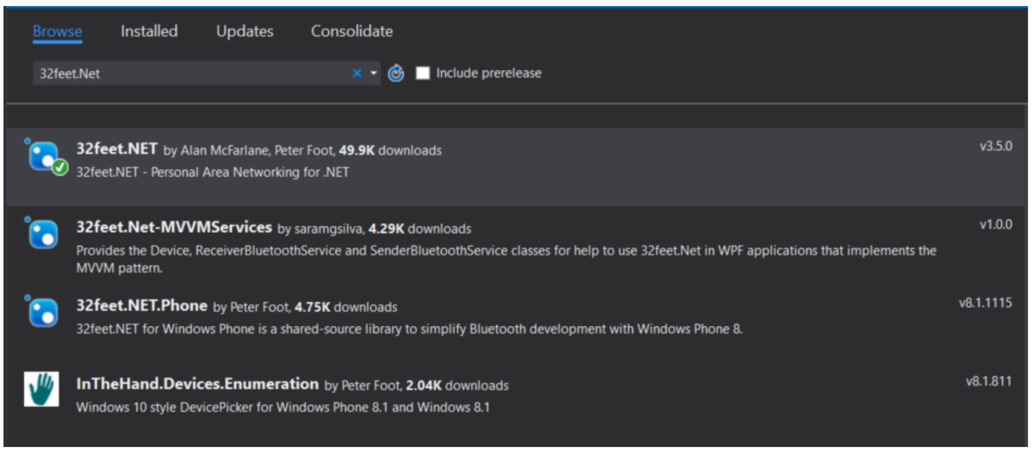

New Project
Recent
Installed
Templates
Visual C++
Other Languages
Build Accelerator
Game
Visual C#
Windows
Android
Cloud
Extensibility
LightSwitch
Office/SharePoint
Reporting
Silverlight
Test
WCF
Workflow
Visual Basic
Online
.NET Framework 4.5.2
Sort by: Default
Search Installed Templates (Ctrl+E)
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Shared Project Visual C#
Class Library (Portable for iOS, Android and Windows) Visual C#
Class Library Visual C#
Class Library (Portable) Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
WCF Service Application Visual C#
Type: Visual C#
A project for creating an application with a Windows Forms user interface
Click here to go online and find templates.
Name: New_Project
Location: C:\Users\Jozef\Projects\VisualStudio\
Solution: Create new solution
Solution name: New_Project
Browse...
Create directory for solution
Create new Git repository
OK
Cancel

Figure 5. Setting up a new project

Before the implementation of Bluetooth service can start, the correct library needs to be installed. That can be done by downloading this library from 32feet.NET web site or easily through the NuGet package manager. The NuGet package manager can be opened by right-clicking on the project solution and then selecting “Manage NuGet packages for solution” option. After that, the 32feet.NET library can be found in the opened window, as in Figure 6 illustrates.



32feet.NET by Alan McFarlane, Peter Foot, 49.9K downloads v3.5.0
32feet.NET - Personal Area Networking for .NET

32feet.Net-MVVMServices by saramgsilva, 4.29K downloads v1.0.0
Provides the Device, ReceiverBluetoothService and SenderBluetoothService classes for help to use 32feet.Net in WPF applications that implements the MVVM pattern.

32feet.NET.Phone by Peter Foot, 4.75K downloads v8.1.1115
32feet.NET for Windows Phone is a shared-source library to simplify Bluetooth development with Windows Phone 8.

InTheHand.Devices.Enumeration by Peter Foot, 2.04K downloads v8.1.811
Windows 10 style DevicePicker for Windows Phone 8.1 and Windows 8.1

Figure 6. Installing 32feet.NET library in NuGet Package Manager

"32feet.NET is a shared-source library to make personal area networking technologies (PAN) such as Bluetooth, Infrared (IrDA) and more, easily accessible from .NET code. ". Besides desktop systems, this library also supports mobile and embedded systems. There are no restrictions for developers, because 32feet.NET is free for non-commercial and commercial use as well. It is possible to use this library as is, however, developers may make modifications if needed, but the Licence.txt document must be included. (32feet.NET 2012).

The currently library consists of the following parts (Ibid):

- Bluetooth
- IrDA
- Object Exchange

To use this library, a device with Microsoft, Widcomm, BlueSoleil or Stonestreet One Bluetopia Bluetooth stack is required. Besides that, there are required .NET Compact Framework 3.5 version or higher and .NET Framework 3.5 version for desktop Windows XP, Window Vista, Windows 7, Window 8, Windows 10 operation system or Windows CE.NET 4.2. It is also possible to use the functionality of this library for Windows Phone 8 in the InTheHand.Phone.Bluetooth.dll sub-library (Ibid).

4.2.3 The application

The application is based on an idea of "smart home". "Smart home" is the term used to define a residence that has appliances, lighting, heating, air conditioning, TVs, computers, entertainment, audio & video systems, security and camera systems, that are capable of communication with one another and can be controlled remotely by a time schedule, from any room in the house, as well as remotely from any location in the world by phone or internet (What is smart home 2018).
However, the created application is not so "clever" as it was mentioned.
The interface of the application consists of several household appliances such as television, refrigerator, lights and microwave, which are possible to control. Besides that, the interface contains two buttons to turn the server on and off and two text boxes. One text box responses to changes of the server or the client side and shows their actual state. The second one shows a message, which was sent from the client

to the server as request to execute this message. The application interface is illustrated in Figure 7.

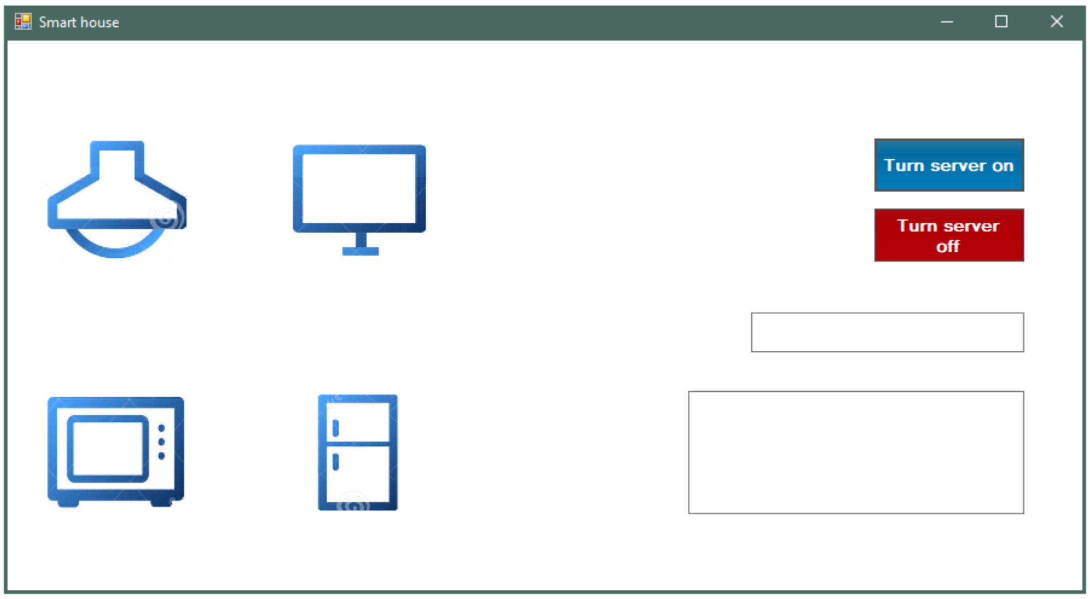
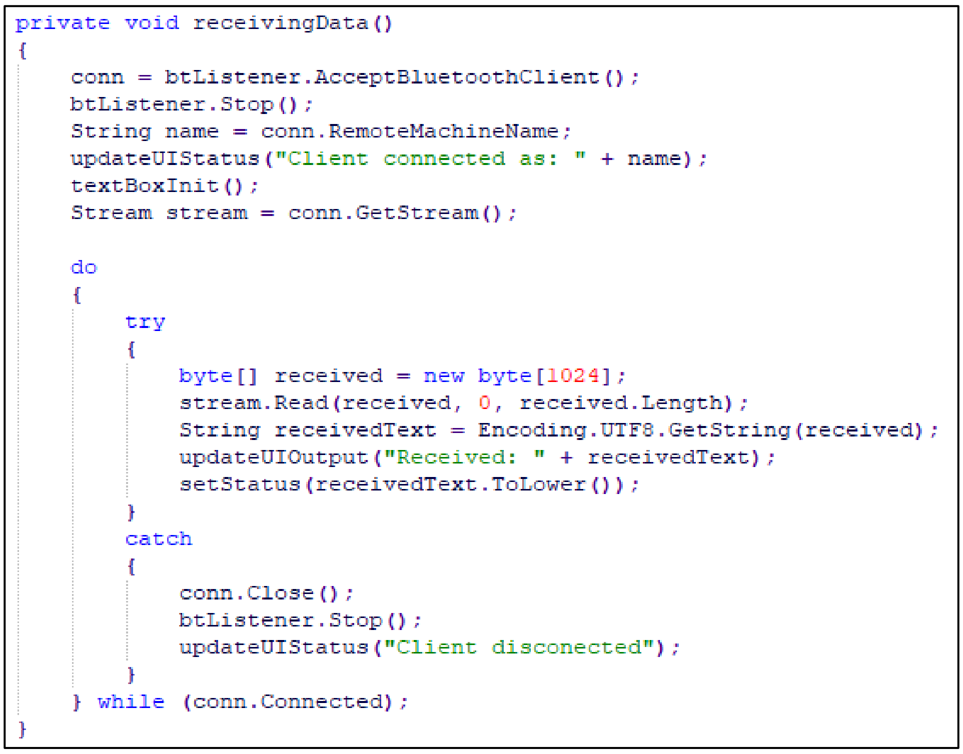


Figure 7. Application interface

The application works as follows. When the server is turned on, the new Bluetooth listener is created, and the server is launched with specific UUID. After that, the Bluetooth listener “listens” to some incoming requests for a connection.

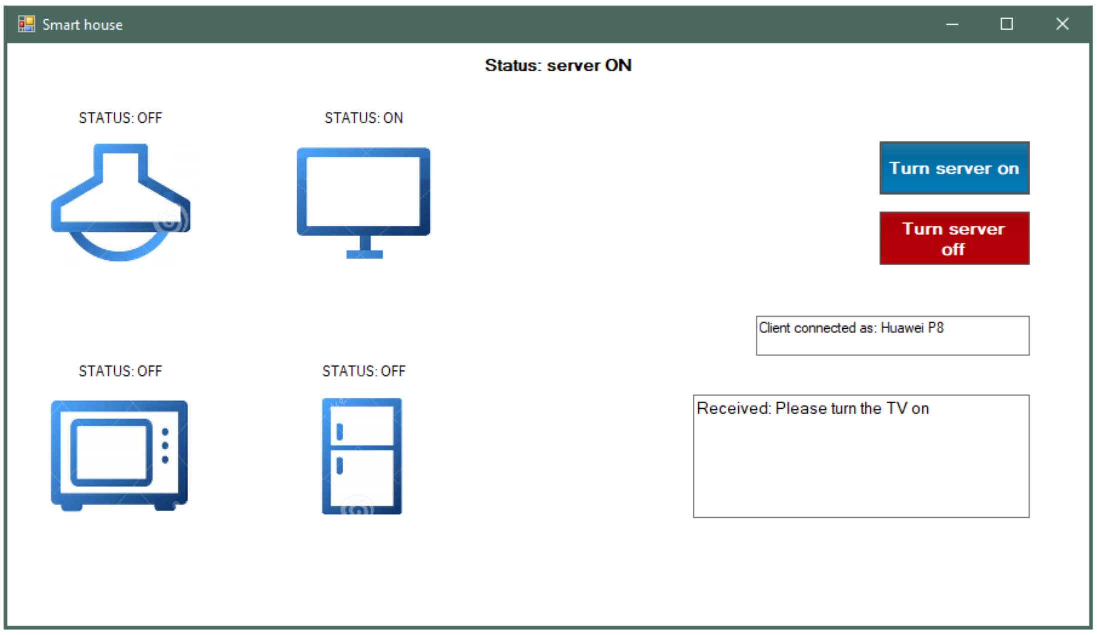
If the client with the same UUID is detected, the server accepts his/her request and tries to establish the connection. If the connection is established successfully, the listener is stopped to enable to listen for another client’s requests and it is shown, which device is connected to the server. Next, the actual state of the appliances is loaded. The application does not have memory for saving the states of appliances from previous session, so when the server turns on, the state of all appliances is set to “OFF”. After that the stream is initialized. “The stream, in programming language, is an abstract class that provides standard methods to transfer bytes such as read and write to the source” (C# Stream 2018). This stream reads the received data and converts it to a form to show it on the server side and for next processing until the client is connected. If some unexpected error occurred, the connection will close, and the server shows that the client was disconnected. This logic next code snippet illustrates.

When the “Turn server off” button is clicked, the connection with the client is closed as well as Bluetooth listener and the server turns off.



```
private void receivingData()
{
    conn = btListener.AcceptBluetoothClient();
    btListener.Stop();
    String name = conn.RemoteMachineName;
    updateUIStatus("Client connected as: " + name);
    textBoxInit();
    Stream stream = conn.GetStream();

    do
    {
        try
        {
            byte[] received = new byte[1024];
            stream.Read(received, 0, received.Length);
            String receivedText = Encoding.UTF8.GetString(received);
            updateUIOutput("Received: " + receivedText);
            setStatus(receivedText.ToLower());
        }
        catch
        {
            conn.Close();
            btListener.Stop();
            updateUIStatus("Client disconected");
        }
    } while (conn.Connected);
}
```

Basically, the user has an overview of the household appliances and can control them or see their actual state; whether they are “ON” or “OFF”, and eventually change the state. The application “is listening” to what the user is saying and based on which appliance was mentioned in the request and what the user wants to do with it, the application will execute specified steps. How it works is illustrated in Figure 8.



Client connected as: Huawei P8

Received: Please turn the TV on

Figure 8. Remote controlling of appliances

4.3 Android application

4.3.1 Overview

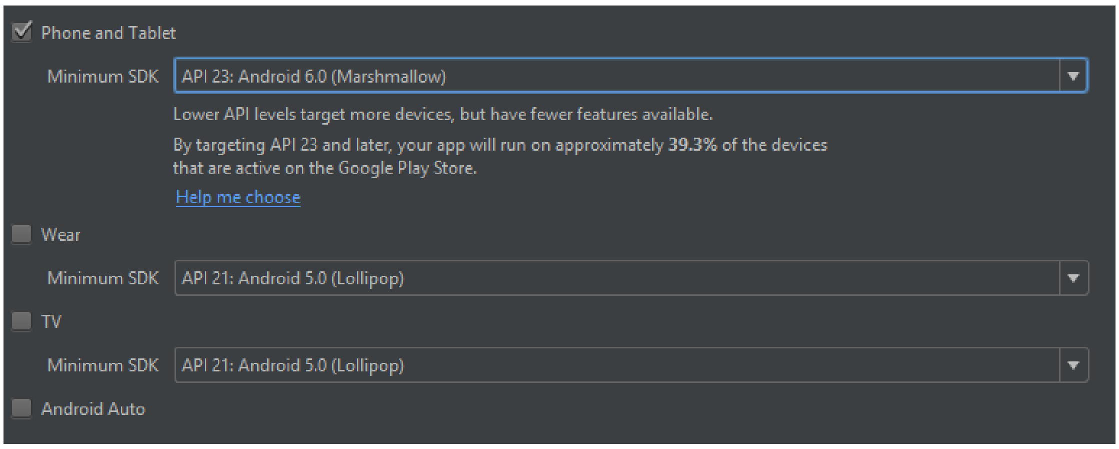
The entire application is developed in the program called Android Studio. “Android Studio is the official Integrated Development Environment for Android application development, based on IntelliJ IDEA.” Because of IntelliJ’s code editor and others helpful tools, the Android Studio offers much more features to enhance the productivity of developers, while an application is developed, such as fast and feature-rich emulator, instant run to push changes to the running application without building a new APK, integrated GitHub, support for C++ language or build-in support for Google Cloud Platform (Meet Android Studio 2018).

4.3.2 Configuration of project

Each project in Android Studio contains of one or more modules. These modules include all source code files and resource files. “All the build files are visible at the top level under Gradle Scripts” and each application’s module contains folders (Meet Android Studio 2018):

- manifest: the AndroidManifest.xml file is located here
- java: contains the JAVA source code files
- res: all non-code resources are situated here, such as UI strings, XML Layouts or images

To create a new project is possible by clicking to “File”->”New”->”New Project”. After the name of project is entered and the location of the project as well, the user can choose between Android platforms such as Phone and Tablet, Wear, TV or choose Android Auto option. Besides that, one of the offered Androids SDKs, from Android 2.3 to newest one Android P needs to be chosen. It is important to choose the correct one because some of the features from higher APIs are not supported for lower; therefore, the created application may not work. This is shown in Figure 9. To make coding easier, the developer can pick one from some offered Activities; for example, Google Map Activity, Login Activity, Setting Activity or Basic whether Empty Activity.



[✓] Phone and Tablet
Minimum SDK: API 23: Android 6.0 (Marshmallow)
Lower API levels target more devices, but have fewer features available.
By targeting API 23 and later, your app will run on approximately **39.3%** of the devices that are active on the Google Play Store.
Help me choose

[ ] Wear
Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] TV
Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] Android Auto

Figure 9. Choice of Android platform

4.3.3 Application

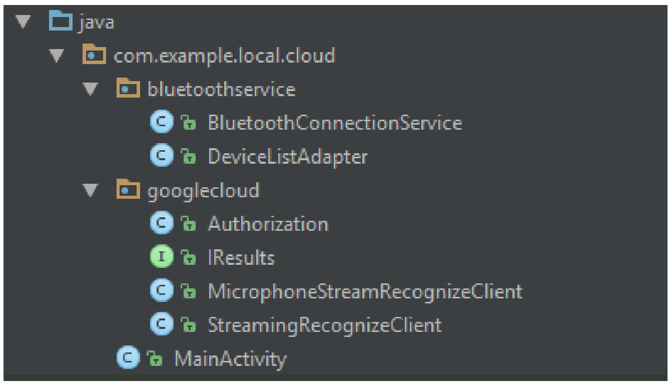
The application is created as a client, which can be connected to a server. The application is composed of three different parts:

- Bluetooth service
- Google cloud service
- Main Activity

Bluetooth service consists from two JAVA classes, BluetoothConnectionService and DeviceListAdapter, and from one layout file, device_adapter_view. BluetoothConnectionService class includes the entire Bluetooth client service, which is used to connect to the Bluetooth server and the data sending. DeviceListAdapter class contains the method to hold information about available Bluetooth devices and as follows, the layout file can display them.

Google cloud service consists of three JAVA classes and one Interface. StreamingRecognizeClient class serves for setting the whole Google service configuration as was explained in chapters 2.5.2 and 2.5.3 as well as the microphone of the device. MicrophodeStreamRecognizeClient is the client for audio steaming initialized and in the Authorization class there is access to the Google cloud server set. The interface IResults is a part of Google Cloud service where the options of streamed text are set such as onPartial and onFinal.

MainActivity class serves for controlling the entire application such as layout behavior, initialization of buttons, menu, configurating of the Bluetooth service. This project structure is illustrated in Figure 10.



```
java
  com.example.local.cloud
    bluetoothservice
      BluetoothConnectionService
      DeviceListAdapter
    googlecloud
      Authorization
      IResults
      MicrophoneStreamRecognizeClient
      StreamingRecognizeClient
    MainActivity
```

Figure 10. Project structure

The application's interface contains several buttons, switch, two text views and one list view. Using the buttons the user can turn on or off the Bluetooth adapter, find some available Bluetooth devices displayed in list view, record the voice and send the record. Using the switch, the user can choose between automatic or manual sending of the record. One of the text views displays the server where the device is connected and another of the recorded text.

The interface of the application is illustrated in Figure 11.

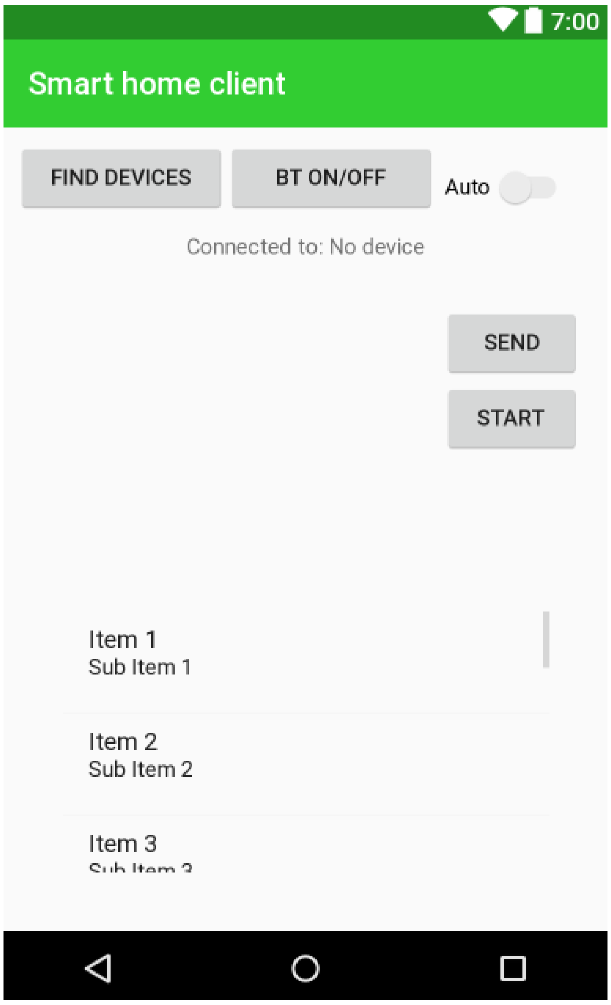


Figure 11. Android application layout

The application works as follows. In the start the Bluetooth needs to be turned on with the button designed for that. By pressing this button, the program checks itself if Bluetooth is supported and if it is, Bluetooth turns on. After that, the user can press “FIND DEVICES” button. If Bluetooth was turned off, the message will appear that Bluetooth is off, and Bluetooth service cannot search for other Bluetooth devices. The Bluetooth broadcast receiver will be initialized and will listen for available devices. This is shown in the next code snippet.

```
private BroadcastReceiver mBroadcastReceiver3 = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        final String action = intent.getAction();
        Log.d(TAG, "onReceive: ACTION FOUND.");
        if (action.equals(BluetoothDevice.ACTION_FOUND)) {
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            if (!mBTDevices.equals(device)) {
                mBTDevices.add(device);
                Log.d(TAG, "onReceive: " + device.getName() + ": " + device.getAddress());
                mDeviceListAdapter = new DeviceListAdapter(context,
                                   R.layout.device_adapter_view, mBTDevices);
                lvNewDevices.setAdapter(mDeviceListAdapter);
            }
        }
    }
};
```

Subsequently, all available devices will be displayed in the list view. By long-pressing on the selected device, the context menu is opened. The context menu offers three options: to connect to the selected device, to pair with the selected device or to disconnect from the connected device. The context menu is shown in Figure 12.

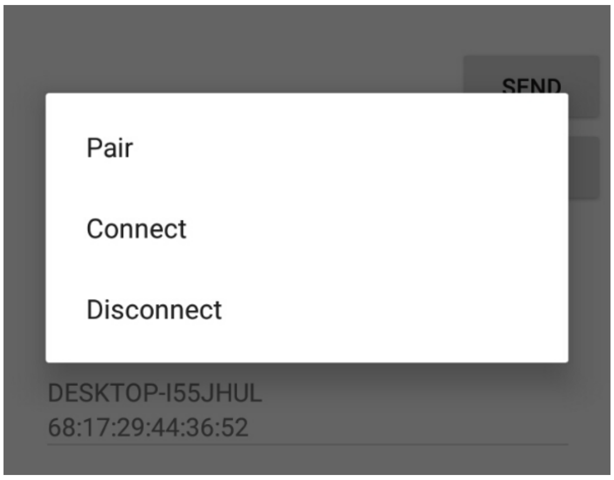
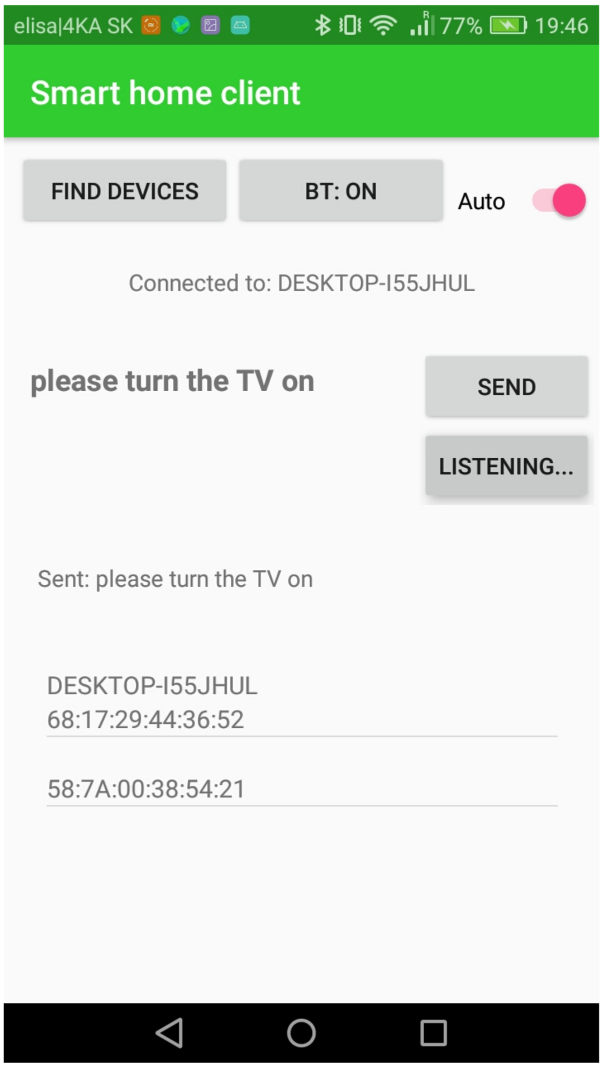


Figure 12. Context menu

Before the device is connected to another device, it is checked, whether the devices are already paired with each other. If not, the message is displayed that the devices must pair first. When the connection is passed, the message about connected devices and the name of the connected device are displayed in the text box as well. From this point, the user can send an order to the server by pressing the “START” button. When the button is pressed, Google Speech service is initialized and while it is pressed, the speech is sent to Google Cloud server for the transcription. After releasing the button, Google Speech service is stopped, and the entire text is displayed. If the “Auto” switch is on, the transcribed text is immediately sent to the server, if not, the user has to press “Send” button to send this text to the server. How it works is shown in Figure 13.



Connected to: DESKTOP-I55JHUL

please turn the TV on

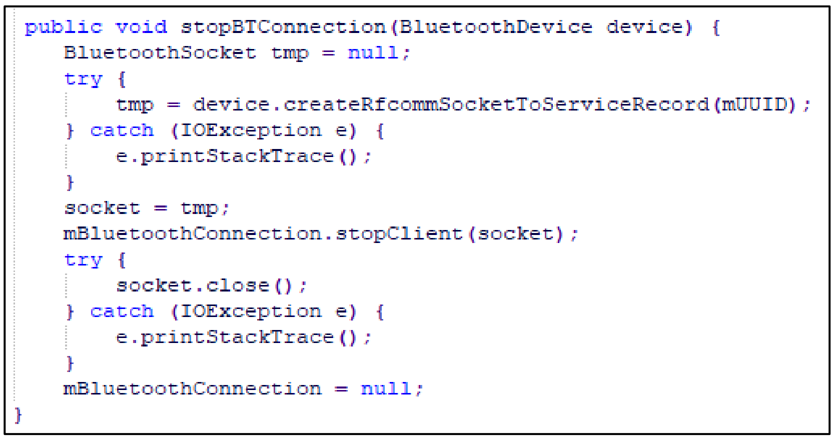
Sent: please turn the TV on

DESKTOP-I55JHUL
68:17:29:44:36:52

58:7A:00:38:54:21

Figure 13. Application after a record is sent

The sending of records works until the device is connected to the server. If the connection is interrupted, the message about the disconnected client will display and the user must initialize the new connection. When the user wants to disconnect from the server, the long-pressing on the connected device in list view displays the context menu and by pressing on “Disconnect” option the device is disconnected. In this moment, the method for disconnecting is called where the client socket is closed, and Bluetooth connection service is closed. This method following code snippet illustrates.



```
public void stopBTConnection(BluetoothDevice device) {
    BluetoothSocket tmp = null;
    try {
        tmp = device.createRfcommSocketToServiceRecord(mUUID);
    } catch (IOException e) {
        e.printStackTrace();
    }
    socket = tmp;
    mBluetoothConnection.stopClient(socket);
    try {
        socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    mBluetoothConnection = null;
}
```

5 Results

The result of several months of work is the application consists from two separate applications; the Windows application on the server side and the Android application on the client side. The application works as a simple concept of “smart home”, where the user can control by voice some household appliances through an application.

During the test phase I have not met with some unexpected errors or sudden fails and both applications work properly.

However, there are some features I would add into the application in the future. For example, the user cannot see the current status of appliances in the Android application layout and the confirmation about executed command is missing as well. There is also no option to choose some another language, although the application supports not just English language, but German and Spanish as well.

Despite these restrictions, the application met all selected objectives, which were given at the beginning of the thesis.

6 Conclusion

The main goal of the thesis was to develop a project consisting of two separate applications based on Google Speech Cloud API service which can communicate via Bluetooth service. The purpose of the project was to develop a simple client-server application, which can simulate “smart home” technology.

At the beginning, I was challenged by the new technologies such as Bluetooth service or C# programming language. I have had some previous knowledge about Google Speech Cloud service but not much. The development of an Android application was also relatively new for me. While exploring and gathering new knowledge about these technologies, very well written official documentation by Google and Microsoft helped me.

After the first weeks of obtaining new technologies, I started to develop the Android application and the Bluetooth service. Besides the official documentation, many on-line tutorials and some previous school projects helped me much. After the first working version of the Bluetooth Android application, I started with the implementation of Google Speech Cloud service into the application. When the Google Speech API worked correctly, I merged both separate Android applications to one and changed the design. It took a while to rectify all bugs and test the application by sending the transcribed voice record to the PC via Bluetooth. I have used PC’s serial port for that and Tera Term PC application for controlling the received results.

The second step was to develop a Windows application. After some research I decided to create the application in Microsoft Visual Studio and use C# programming language for that because I found great 32feet.NET library for that. Surprisingly, despite of few complications, I developed the Windows application comparatively fast.

The result of this work is the application meeting all the set objectives, even though there are some features, I would like to implement into the application in the future.

References

Bluetooth SIG. 2018. *Radio Versions.* Accessed on 12.5.2018. Retrieved from: https://www.bluetooth.com/bluetooth-technology/radio-versions

Bluetooth SIG. 2018. *Solution Areas.* Accessed on 12.5.2018. Retrieved from: https://www.bluetooth.com/bluetooth-technology/solution-areas

Bluetooth SIG. 2018. *Topology Options.* Accessed on 12.5.2018. Retrieved from: https://www.bluetooth.com/bluetooth-technology/topology-options

Brilliant.org. 2018. *Hidden Markov Models.* Accessed on: 10.5.2018. Retrieved from: https://brilliant.org/wiki/hidden-markov-models/

Google. 2018. *Bluetooth overview.* Accessed on 24.4.2018. Retrieved from: https://developer.android.com/guide/topics/connectivity/bluetooth

Google. 2018. *Cloud Speech-to-Text Basic.* Accessed on 9.4.2018. Retrieved from: https://cloud.google.com/speech-to-text/docs/basics

Google. 2018. *Meet Android Studio.* Accessed on: 8.5.2018. Retrieved from: https://developer.android.com/studio/intro/

In The Hand Ltd. 2012. *32feet.NET – Personal Area Networking for .NET*. Accessed on 17.4.2012, Retrieved on: https://archive.codeplex.com/?p=32feet

Lawson, S. 2018. *The Past, Present, and Future of Speech Recognition Technology*. Accessed on 19.3.2018. Retrieved from: https://www.clickz.com/report-past-present-future-speech-recognition-technology/209060/

Microsoft. 2018. *Visual Studio IDE.* Accessed on 12.5.2018. Retrieved from: https://msdn.microsoft.com/library/dn762121(v=vs.140).aspx

Mukhtar, M. 2017. *Bluetooth 4 Vs. Bluetooth 5: A feature comparison.* Accessed on 5.5.2017. Retrieved from: https://www.itechtics.com/bluetooth-4-vs-bluetooth-5-feature-comparison/

Saksamudre, S & Shrishrimal, P.P & Deshmukh,R.R. 2015. *A Review on Different Approaches for Speech Recognition System.* Accessed on 22.4.2015. Retrieved from: https://pdfs.semanticscholar.org/b909/3377c6579b97ab8bd5d4dd9947d372dddc2e.pdf

Scientific American. 2017. *How does Bluetooth work?.* Accessed on 5.5.2018. Retrieved from: https://www.scientificamerican.com/article/experts-how-does-bluetooth-work/

SmartHomeUSA. 2018. *What Is A Smart Home.* Accessed on: 17.5.2018. Retrieved from: https://www.smarthomeusa.com/smarthome/

SparkFun. 2013. *Bluetooth Basics.* Accessed on 26.4.2018. Retrieved from: https://learn.sparkfun.com/tutorials/bluetooth-basics?_ga=2.176954469.1557557617.1526634123-229761359.1525950169

Total Voice Technologies. 2018. *Most Common Uses of Voice Recognition Software.* Accessed on: 2.4.2018. Retrieved from: http://www.totalvoicetech.com/Most-Common-Uses-of-Voice-Recognition-Software.html

Tutorials Teacher. 2018. *C# Stream:*. Accessed on: 1.5.2018. Retrieved from: http://www.tutorialsteacher.com/csharp/csharp-stream-io