TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Control

Andrea Di Florio

VOICE REMOTE CONTROL WITH ARDUINO

Project

Instructor: Dr. Juri Belikov

Tallinn 2014

Table of contents

Introduction

In the last few years, there has been a constant development of technologies based on speech recognition, which has been employed to write messages, to manage smartphones as well as to write texts. My project is based on the idea of developing a system which, by using the speech recognition, would be able to command a television remotely. A similar technology (Magic Remote Control) developed by lg is already on sale and it is compatible only with televisions of the same brand and it cost is around €60.

Objectives

My project aims at developing a vocal remote control of reduced costs and able to:

- Recognize vocal commands produced by the user;
- Send simple commands, like those sent by pressing a button on the remote control;
- Send complex commands(signals), like those sent by pressing a series of buttons on the remote control.

The essay is structured as follows:

- In Chapter 1 the structure of the system will be described, both the components and the developed app will be analysed in detail.
- In Chapter 2 the Use Case Diagram, Activity diagrams, Sequence Diagrams will be described.
- In Chapter 3 possible usage scenarios of the system will be presented.
- In Chapter 4 possible future development of the system as well as various usage modalities will be discussed.

Chapter 1

Structure of the system

In order not to increase the costs, the system is based on an Android smartphone (owned by millions of people, and an Arduino Uno[2] whose cost is around 15 Euros (cheaper than LG remote control). The smartphone is used for the speech recognition by using the API provided by Google[9][10], whereas Arduino is used to send the infrared signal. The communication between the two takes place through the Bluetooth[1]. The advantage of this structure relies on the fact that the majority of smartphones Android are provided with a Bluetooth port but not with an infrared one.

The developed app realizes a speech recognition and sends the signals through the Bluetooth. The signals sent from the smartphone are decoded by Arduino and used to select the command to be send to the television or to the selected device.

In particular, the components used are the following:

Figure 1 Smartphone Android

Figure 2 Arduino Uno

Figure 3 Bluetooth module HC-05

Figure 4 IR sender led

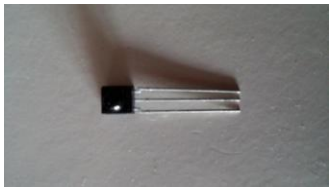


Figure 5 IR receiver TSOP4838

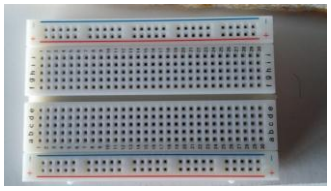


Figure 6 Breadboard

Figure 7 Resistance

Figure 8 Connection cables

1.1 Decoding signals

In order to use a Samsung television the infrared signals (IR) proceeding from a compatible remote control have been decoded. For this aim, the IR library[3] for Arduino has been used.

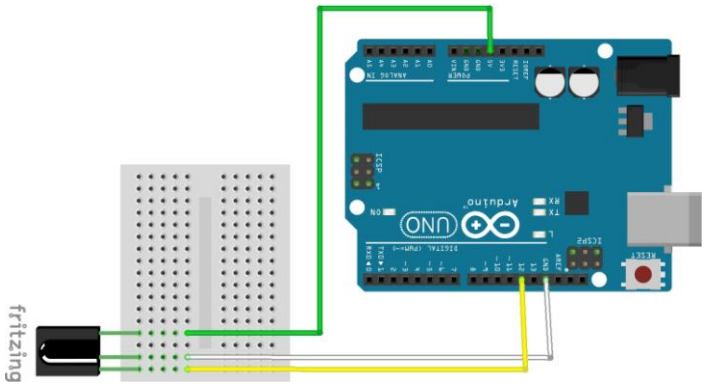


Figure 9 Arduino scheme to receive signal

The code used to decode the signals is the following:

```
1. #include <IRremote.h>
2.
3. int PIN = 12;
4.
5. IRrecv irrecv(PIN);
6.
7. decode_results risultato;
8.
9. void setup(){
10. Serial.begin(9600);
11. irrecv.enableIRIn(); //Set receiver
12. }
13.
14. void loop() {
15. if (irrecv.decode(&risultato)) { //Decode the value
16. Serial.println(risultato.value, HEX);
17. irrecv.resume(); //Receive the next value
18. }
19. }
```

Line number 5 of the code shows the parameter irrecv as the receiver of the infrared signal from the remote control and consider as input parameter the pin used for the reception. In lines 10 the frequency for the communication with Arduino USB port is set up; whereas line 11 concerns the reception of the signal. The instruction in line 15 decodes the value of the sent signal (in hexadecimal notation) and subsequently print to video the hex value (line16)[4]

The following table contains the values of the main commands: the first column shows the button pressed on the remote control, the second the respective action and the third one the decoded hexadecimal value.

COMMAND	ACTIVITY	HEX VALUE
Turn on/Turn off	TV turn on/turn off	E0E040BF
1	Channel one	E0E020DF
2	Channel two	E0E0A05F
3	Channel three	E0E0609F
4	Channel four	E0E010EF
5	Channel five	E0E0906F
6	Channel six	E0E050AF
7	Channel seven	E0E030CF
8	Channel eight	E0E0B04F
9	Channel nine	E0E0708F
0	Channel zero	E0E08877
Volume up	Raise the volume	E0E0E01F
Volume down	Reduce the volume	E0E0D02F
Channel up	Change channel forward	E0E048B7
Channel down	Change channel back	E0E008F7
Mute	Disable audio	E0E0F00F
Telvideo	Access to Televideo	E0E034CB
Channel list	See the channel list	E0E0D629
Tools	Show the menu of settings	E0E0D22D
Info	Show the info about televison program	E0E0F807
Back	Go back	E0E01AE5
Exit	Exit from any menu	E0E0B44B
Up	Arrow up	E0E006F9
Down	Arrow down	E0E08679
Guide	Show the channel guide	E0E0F20D
Left	Arrow left	E0E0A659
Right	Arrow right	E0E046B9
Ok	Allows you to enter and confirm the change of an option, enter into folders that contain media files or open applications	E0E016E9
Smart	Access to the application of Smart TV	E0E09E61
Play	Play the movie	E0E0E21D
Pause	Pause the movie	E0E052AD
Stop	Stops the movie	E0E0629D
Forward	Forward the movie	E0E012ED
Go back	Go back the movie	E0E0A25D
Menu	Show menu	E0E058A7
Subtitle	Show subtitle	E0E0A45B
Source	Show the source of TV	E0E0807F

Figure 10 Remote control used

1.2 Signal Sending

Once collected the signals of the remote control, the Android smartphone has been used for the speech recognition. The smartphone has sent by Bluetooth the characters to the Arduino card, which is connected to the Bluetooth module.

Every character is identified through a 6 values encoding and each encoding is associated with a signal to be sent to the television. When the Arduino card receives an encoding of a character, it verifies what encoding is it and send the respective signal through the infrared led (IRSender) to the television.

In order to manage the Bluetooth connection the Arduino SoftwareSerial library has been used[6]. The working instructions are showed in the following code:

```
1.   #include <SoftwareSerial.h>
2.
3.   #define rxPin 10
4.   #define txPin 11
5.    […]
6.    void setup(){
7.    //line to sent pin to send and to receive
8.    pinMode(rxPin, INPUT);
9.    pinMode(txPin, OUTPUT);
10.   mySerial.begin(38400);
12. }
```

Lines 3 and 4 define the Arduino pin codes used for the reception and sending of Bluetooth signals. Such pin codes are used in lines 8 and 9 as parameters for the pinMode functions. Finally, in line 10 the frequency for the Bluetooth communication is set up.

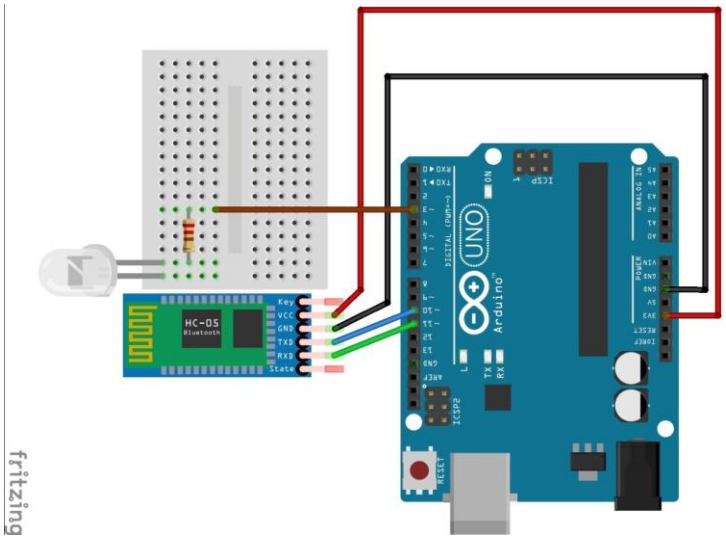


Figure 11 Arduino schema of the system

In order to optimize the management of Arduino memory, every command has been memorized through one variable instead of through 6. In particular, the value included in the variable is obtained through this formula1:

$$Id\ command=(val1*1) + (val2*2) + (val3*3)$$

Instead, for the other commands the formula is:

$$(val1*1) + (val2*2) + (val3*3) + (val4*4) + (val5*5) + (val6*6).$$

In the following table the first column indicates the command expressed by the user, in the second one the character sent by the smartphone to Arduino, the following columns represents the decoding of the character in Arduino and the last column the id associated to the command.

Vocal command	Character	Val 1	Val 2	Val 3	Val 4	Val 5	Val 6	Id command
Turn on/Turn off	a	120	0	248	255	255	255	864
1	1	120	0	128	255	255	255	504
2	2	128	128	128	255	255	255	768
3	3	248	128	128	255	255	255	888
4	4	0	248	128	255	255	255	880
5	$	0	128	128	255	255	255	4465
6	6	128	248	128	255	255	255	1008
7	7	248	248	128	255	255	255	4953
8	£	128	0	248	0	248	255	3642

[1] Formula valid for characters which have as last 3 values 255 and val(I) identifies the i-th value received.

9	9	120	224	128	255	255	255	4777
0	0	248	248	248	255	255	255	5313
Volume up	+	248	120	30	248	255	255	4375
Volume down	–	248	120	254	248	120	254	4366
Channel up	^	128	120	254	255	255	255	1130
Channel down	}	120	254	248	255	255	255	1372
Mute	b	128	0	248	255	255	255	872
Televideo	n	128	248	248	255	255	255	1368
Channel list	=	120	254	128	255	255	255	1012
Tools	/	248	120	224	255	255	255	1160
Info	h	0	248	248	255	255	255	1240
Back	§	128	0	248	128	248	255	4154
Exit	à	248	0	0	248	255	255	4045
Up	y	120	224	248	120	224	248	4400
Down	d	0	128	248	255	255	255	1000
Left	.	128	120	224	255	255	255	1040
Right	’	0	120	224	255	255	255	912
Ok	k	248	120	254	248	255	255	5047
Smart	ù	248	0	120	224	248	255	4274
Play	ò	248	0	128	128	248	255	3914
Pause	ì	248	0	0	120	254	255	3528
Stop	-	120	254	120	224	255	255	4689
Forward	]	248	248	120	254	255	255	4925
Go back	[	120	254	120	254	255	255	4809
Menu	m	120	254	248	248	255	255	5169
Subtitle	°	128	0	0	248	255	255	3925
Source	#	248	0	128	255	255	255	632
Guide	g	248	128	248	255	255	255	1248
Record	*	128	120	30	248	255	255	4255

In addition, special commands have been created. These commands allow the user to realize more complex actions by using a combination of the commands previously described. In particular, vocal commands able to change the channel just by saying its name have been created. Such commands avoid to send single numbers in sequence to the speech recognizer. Moreover, two other interesting commands have been created. These are: “Start Youtube” and “Channel Tuning”. The former allows to open the app Youtube and the latter enables a automatic syntonization of the television channels. This has several advantages: let’s figure an old person that needs to synchronize the channels on his/her television and has difficulties in finding or simply remembering the button sequence to carry out the operation. The problem can be solved easily thanks to only a vocal command: “channel tuning”.

In the following table are listed the special commands that have been created. In the first column, the name of the vocal command is indicated. The second column shows the character sent from the smart

phone to Arduino. The following six columns list the decoding of the character by Arduino and finally, in the last column, there is Id paired to the command:

Vocal command	Character	Val 1	Val 2	Val 3	Val 4	Val 5	Val 6	Id command
Channell Sintonization	u	120	30	255	248	255	255	4742
CNN	c	248	0	248	255	255	255	992
Euronews	e	120	30	248	248	255	255	4721
Start YouTube	i	120	224	248	248	255	255	5109

As it was for the reception of the signal, also for the sending the library IRremote has been used[5]. The recognition of the commands has been managed through a switch. The following code fragment recognize the command "ON/OFF" and send the corresponding signal to the television.

```
1. case 864:
2.       Serial.println("Turn on/Turn off");
3.       irsend.sendSamsung(0xE0E040BF, 32);
4.       delay(100);
5.       break;
```

Line 3 contains the function sendSamsung which is used to send the signal to the television. Such function takes in input the hexadecimal value of the command as parameter (in this case ON/OFF) and the relative number of bits. It is also important noticing the function delay(100) in line 4 which allows to manage the delay between the signal just sent and the following signal to be sent.

1.3 Class Diagram of Android app

The app is composed of 3 principal classes:

1. BluetoothChatService
2. BluetoothViewer
3. Device List Activity

The BluetoothChatService class is used for the creation and the management of the Bluetooth connections. In order to improve the realization if this two functions the two subclasses ConnecThread and Connected Thread. The former is used to establish the connection, whereas the second to manage it. Of fundamental importance is also the method write() used to send the command to Arduino.

The Main Activity class of the app is Bluetooth Viewer which allows:

1. The creation of the Bluetooth connection;
2. The activation of the speech recognition;
3. The management of the vocal command;
4. The sending of the command to Arduino.

The creation of the Bluetooth connection takes place by creating a BluetoothChatService object, the speech recognition is activated by pushing the button “mSendButt0n” and the inserted command is managed by the method whatsend(string) which select the command to be sent through the Bluetooth. The sending of the message to Arduino takes place activating the method sendmessage(string) which uses the method write() of the BluetoothChatService object.

The last class (Device List Activity) is an activity which allows the user to visualize the list of the devices paired and to pair.

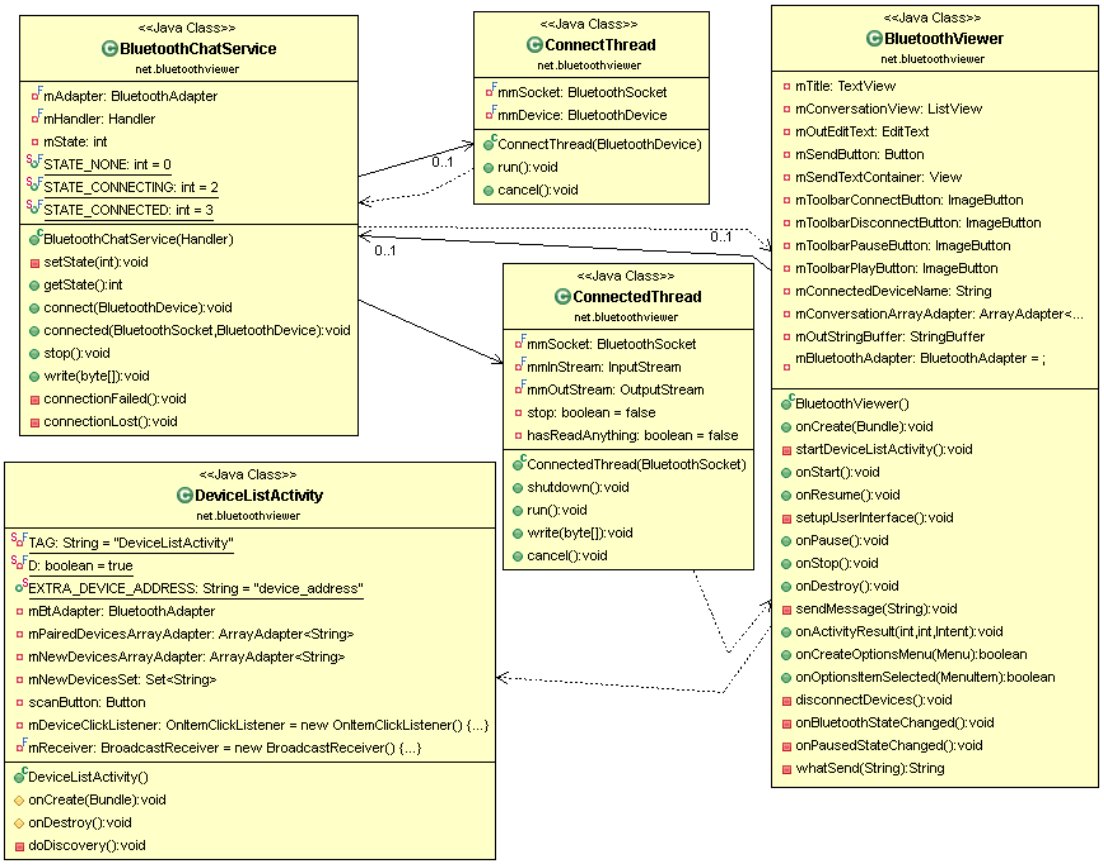


Figure 12 Class diagram of application

Chapter 2

Use Case Diagram, Activity diagrams, Sequence Diagrams

2.1 Use Case

In the following tables are presented the different use cases. The only actor to be considered in this system is the user, that is the person who wants to use the vocal remote control.

Use case	Bluetooth activation
Actors	User
Precondition	Launch application Bluetooth disabled
Sequence of events	1. The system asks to user if he wants to activate the bluetooth 2. If the user does not authorize the activation of bluetooth: 2.1 The system closes the Android application 2.2 The use case ends 3. The user authorizes the bluetooth activation 4. The system activates the bluetooth
Post condition	The device has been paired

Use case	First connection smartphone and Arduino bluetooth
Actors	User
Precondition	Bluetooth activation
Sequence of events	1. The user requests a connection to a device 2. The system asks to user if s/he wants to search for new devices 3. The user selects search devices 4. The system displays the devices to be paired 5. The user selects the device 6. The system requires the entry of the pin for the paired 7. The user enters the pin 8. The system makes the connection
Post condition	The device has been connected

Use case	Connection smartphone and Arduino bluetooth
Actors	User
Precondition	Bluetooth activation done First connectiom smartphone and Arduino bluetooth done
Sequence of events	The user requests a connection to a device The system offers the user the list of devices previously connected

	The user selects the device The system connects the device
Post condition	The smartphone is connected

Use case	General command
Actors	User
Precondition	Bluetooth connection done
Sequence of events	1. The user activates the speech recognition 2. The user sets a command 3. If the command is not recognized: 3.1. The system displays on the screen of the smartphone that the command statement was not recognized 3.2. The use case ends 4. The system displays on the screen of the smartphone the command spelled 5. The system sends the command statement through Arduino
Post condition	The command has been executed

Use case	Channel change from 0 to 9
Actors	User
Precondition	Bluetooth connection done
Sequence of events	1. The user activates the speech recognition 2. The user sets a number between 0 and 9 3. If the command is not recognized: 3.1. The system displays on the screen of the smartphone that the command spelled was not recognized 3.2. The use case ends 4. The system displays on the screen of the smartphone command spelled 5. The system sends the command spelled through Arduino
Post condition	The channel has been changed

Use case	Change the channel from 10 to 999
Actors	User
Precondition	Bluetooth connection done
Sequence of events	1. < Channel change from 0 to 9> 2. < Channel change from 0 to 9> 3. If the user wants to display a 2-digit channel: 3.1 The use case ends 4. < Channel change from 0 to 9>
Post condition	Displays a channel from 10 to 999

Use case	Channel tuning
Actors	User
Precondition	Bluetooth connection done
Sequence of events	1. The user activates the speech recognition 2. The user spells out the command "Tuning Channels" 3. If the command is not recognized: 3.1. The system displays on the screen of the smartphone that the command spelled was not recognized 3.2. The use case ends 4. The system displays on the screen of the smartphone command spelled 5. The system, through Arduino, sends commands to the TV to make the operation indicated by the user
Post condition	The tuning has been carried out
Alternative sequence of events	1. < Generic command > menu 2. < Generic command > Down 3. < Generic command > Down 4. < Generic command > Ok 5. < Generic command > Ok 6. < Generic command > Ok 7. < Generic command > Ok 8. < Generic command >Down 9. < Generic command > Down 10. < Generic command > Down 11. < Generic command > Ok

Use case	Change channel by spelling the name
Actors	User
Precondition	Bluetooth connection done
Sequence of events	1. The user activates the speech recognition 2. The user spells the channel name 3. If the command is not recognized: 3.1. The system displays on the screen of the smartphone that the command spelled was not recognized 3.2. The use case ends 4. The system displays on the screen of the smartphone the command spelled 5. The system, through Arduino, sends the command to the television to perform the operation indicated by the user
Post condition	The change of the channel has been carried out

Use case	Activation of YouTube
Actors	User
Precondition	Bluetooth connection done
Sequence of events	1. The user activates the speech recognition 2. The user spells the command "Start YouTube" 3. If the command is not recognized: 3.1. The system displays on the screen of the smartphone that the command spelled was not recognized 3.2. The use case ends 4. The system displays on the screen of the smartphone the command spelled 5. The system, through Arduino, sends commands to the TV to carry out the operation indicated by the user
Post condition	The smart TV shows Youtube
Alternative sequence of events	1. < Generic command > Smart 2. < Generic command > Right 3. < Generic command > Right 4. < Generic command > Right 5. < Generic command > Right 6. < Generic command > Ok

Use case	Increase/Decrease the volume
Actors	User
Precondition	Bluetooth connection done
Sequence of events	1. The user activates the speech recognition 2. The user spells the command "volume up" / " volume down " 3. The system displays on the screen of the smartphone the command spelled 4. The system, through Arduino, sends commands to the TV to carry out the operation indicated by the user
Post condition	The volume has been turned up/down

2.2 Activities diagrams

In the following section the activity diagrams of some previously described use cases are presented. The activity diagrams represent the different system states:

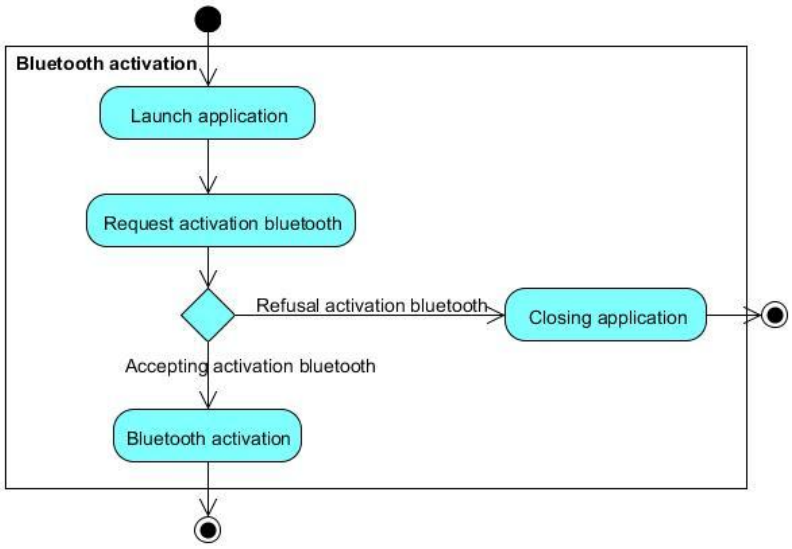


Figure 13 Activity diagrams of Bluetooth activation

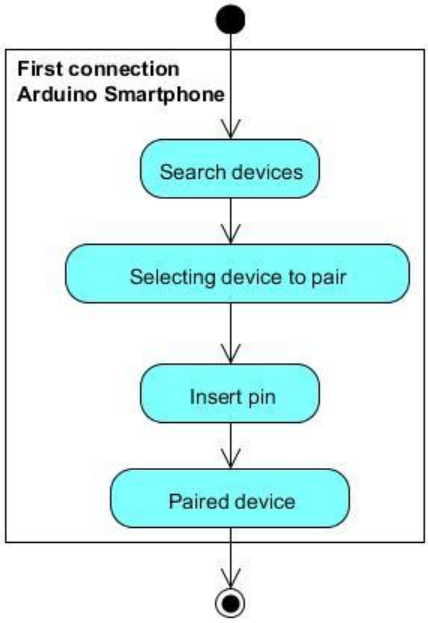


Figure 14 Activity diagrams of First connection Arduino Smartphone

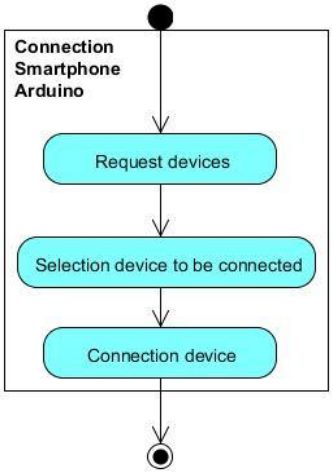


Figure 15 Activity diagrams of connection smartphone Arduino

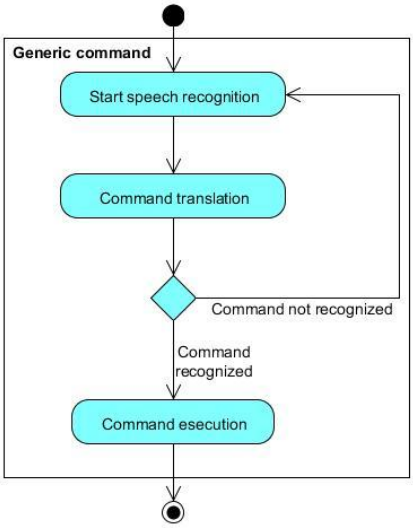


Figure 16 Activity diagrams of generic command

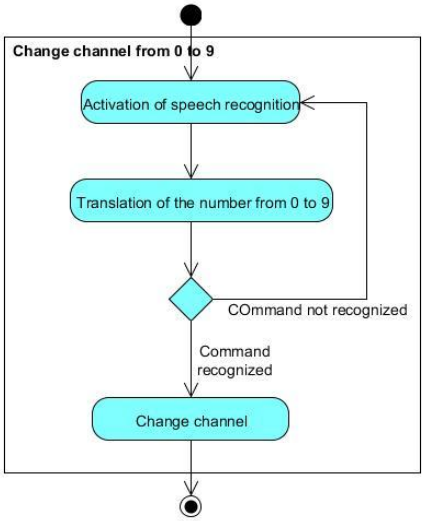


Figure 17 Activity diagrams of change channel from 0 to 9

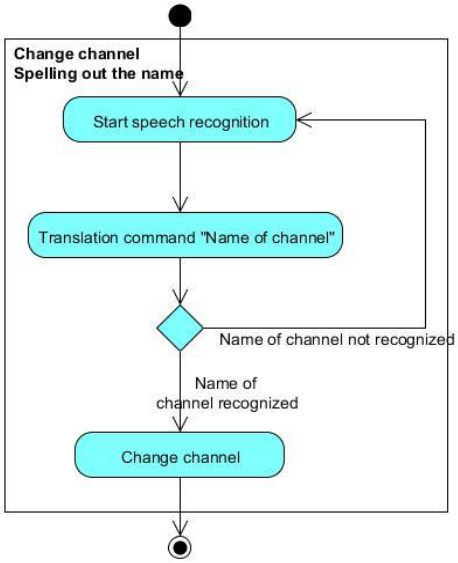


Figure 18 Activity diagrams of change channell spelling out the name

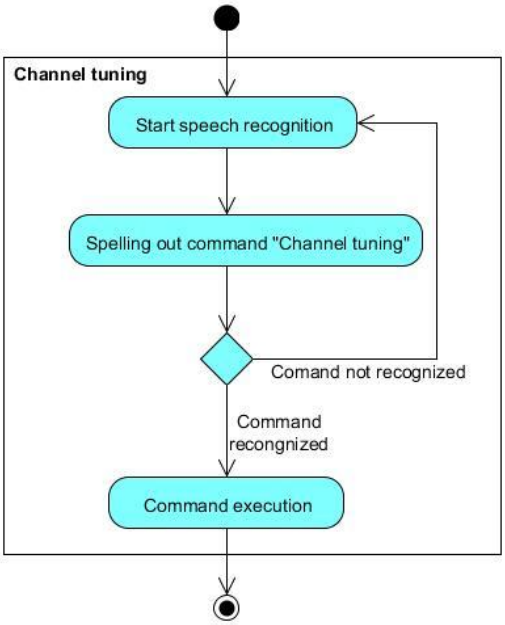


Figure 19 Activity diagrams of channel tuning

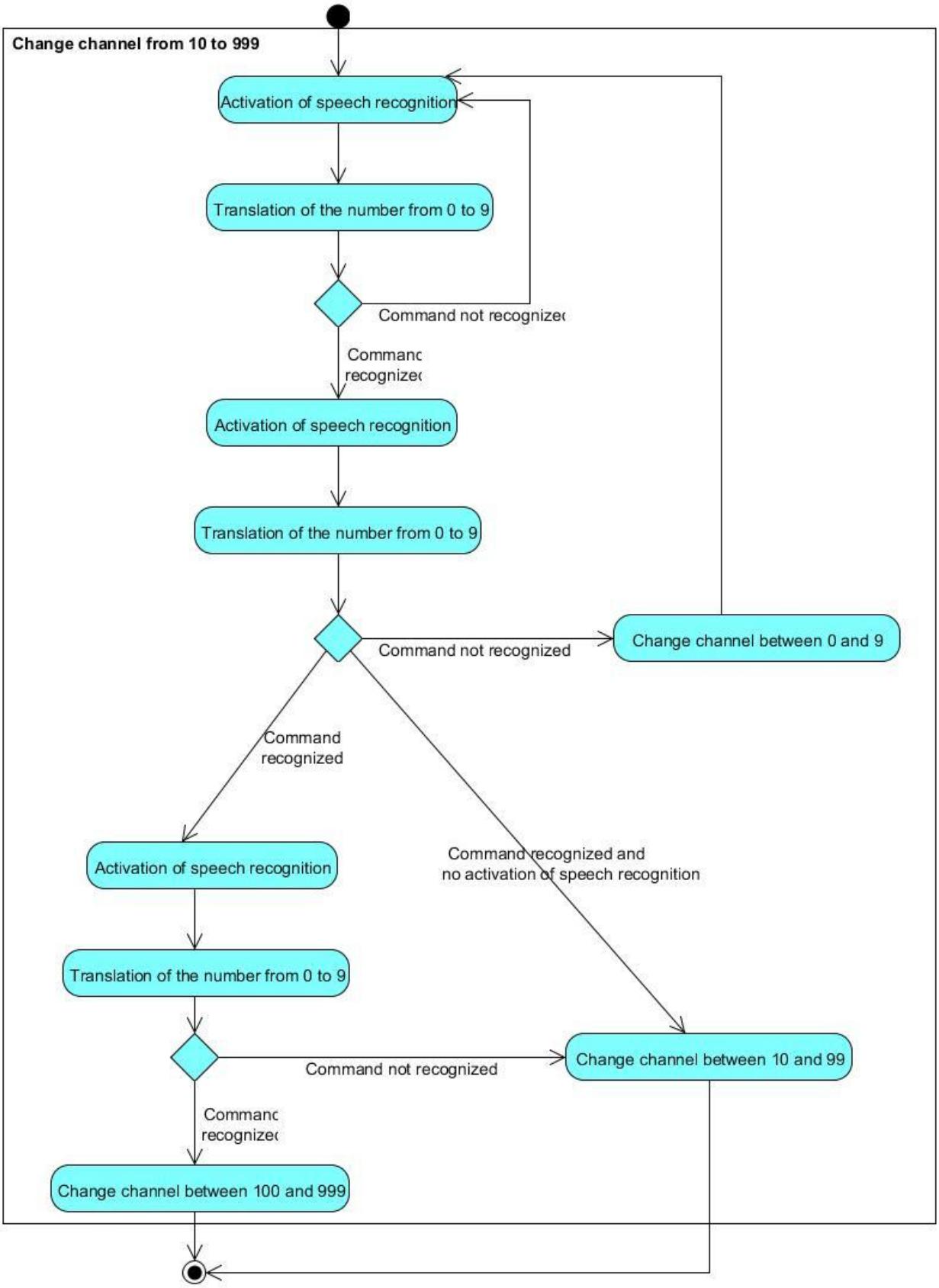


Figure 20 Activity diagrams of change channel from 10 to 999

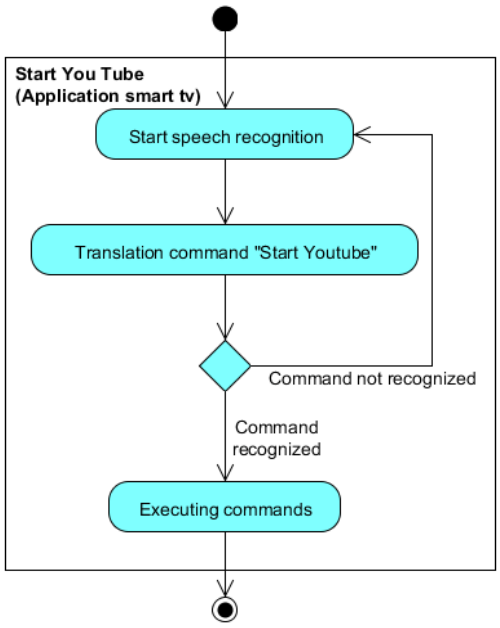


Figure 21 Activity diagrams of command start you tube

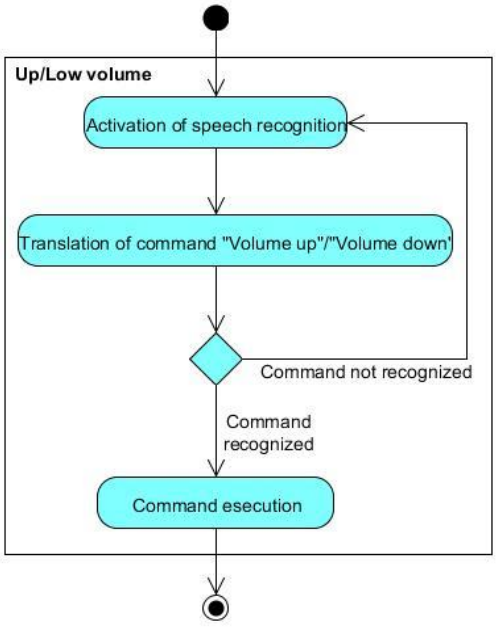


Figure 22 Activity diagrams of change volume

2.3 Sequence Diagrams

In the following schemes the sequence diagrams relative to some previously described cases are presented. The sequence diagrams represents the interaction among the objects of the system:

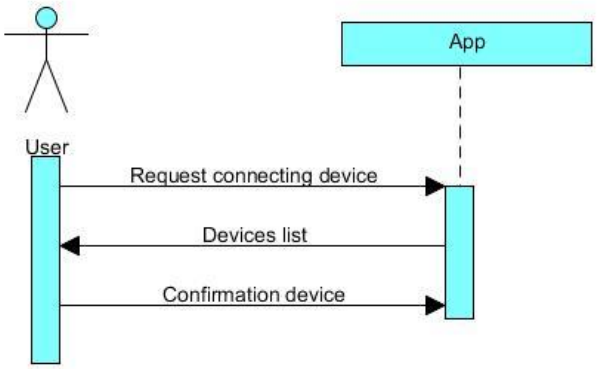


Figure 23 Sequence Diagrams of bluetooth activation

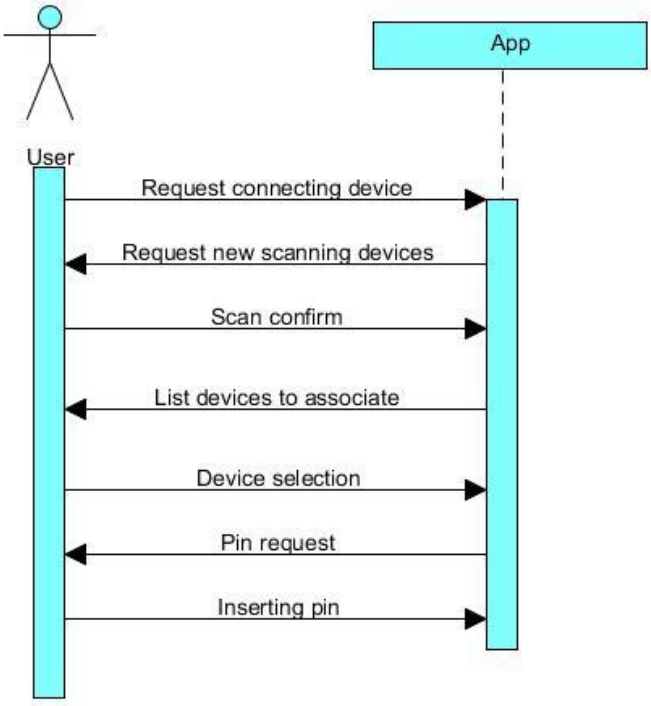


Figure 24 Sequence Diagrams of the first bluetooth connection

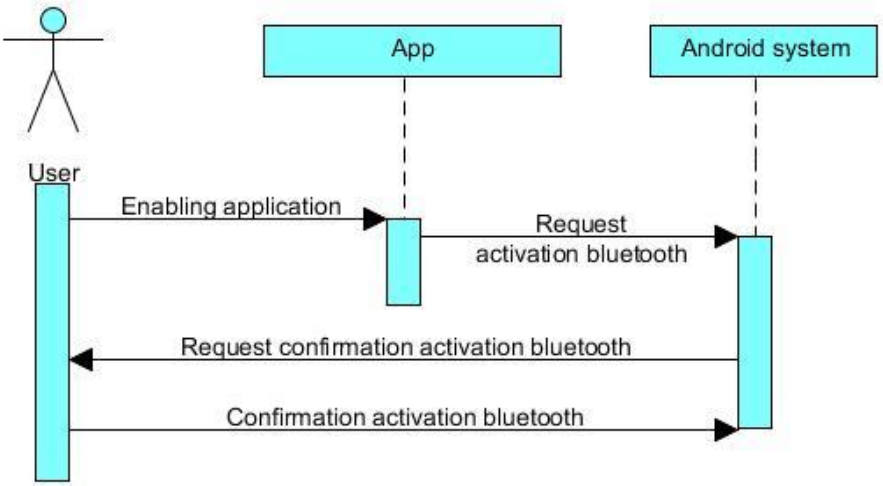


Figure 25 Sequence Diagrams of Bluetooth connection

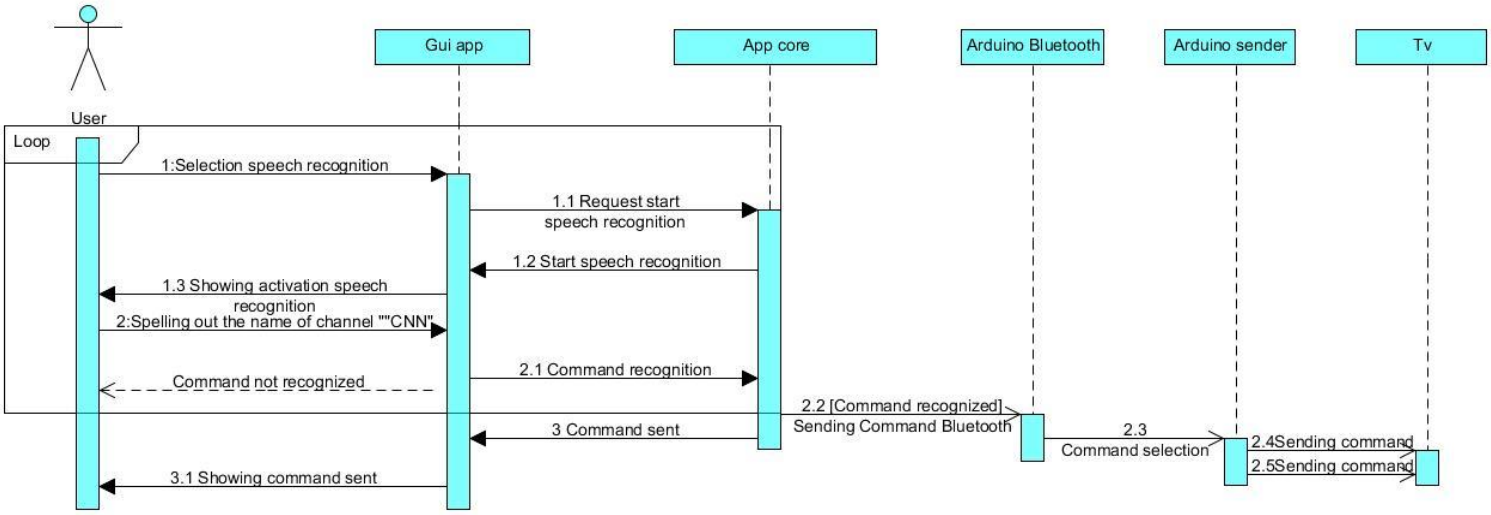


Figure 26 Sequence Diagrams of change channel with the name

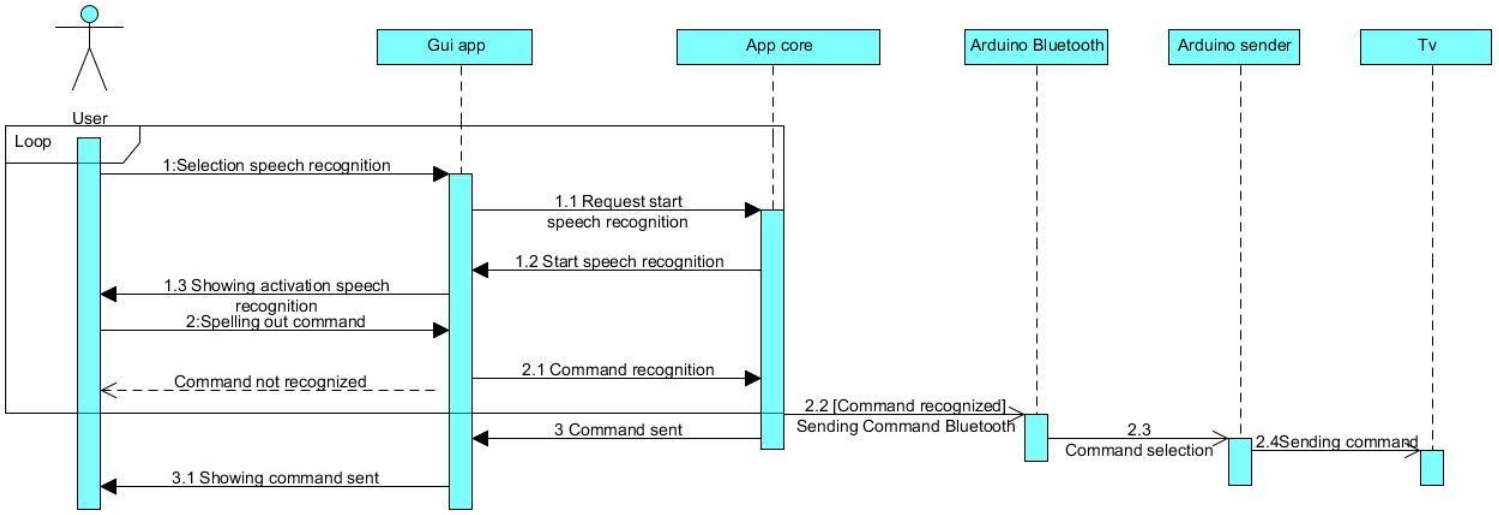


Figure 27 Sequence diagram of generic command

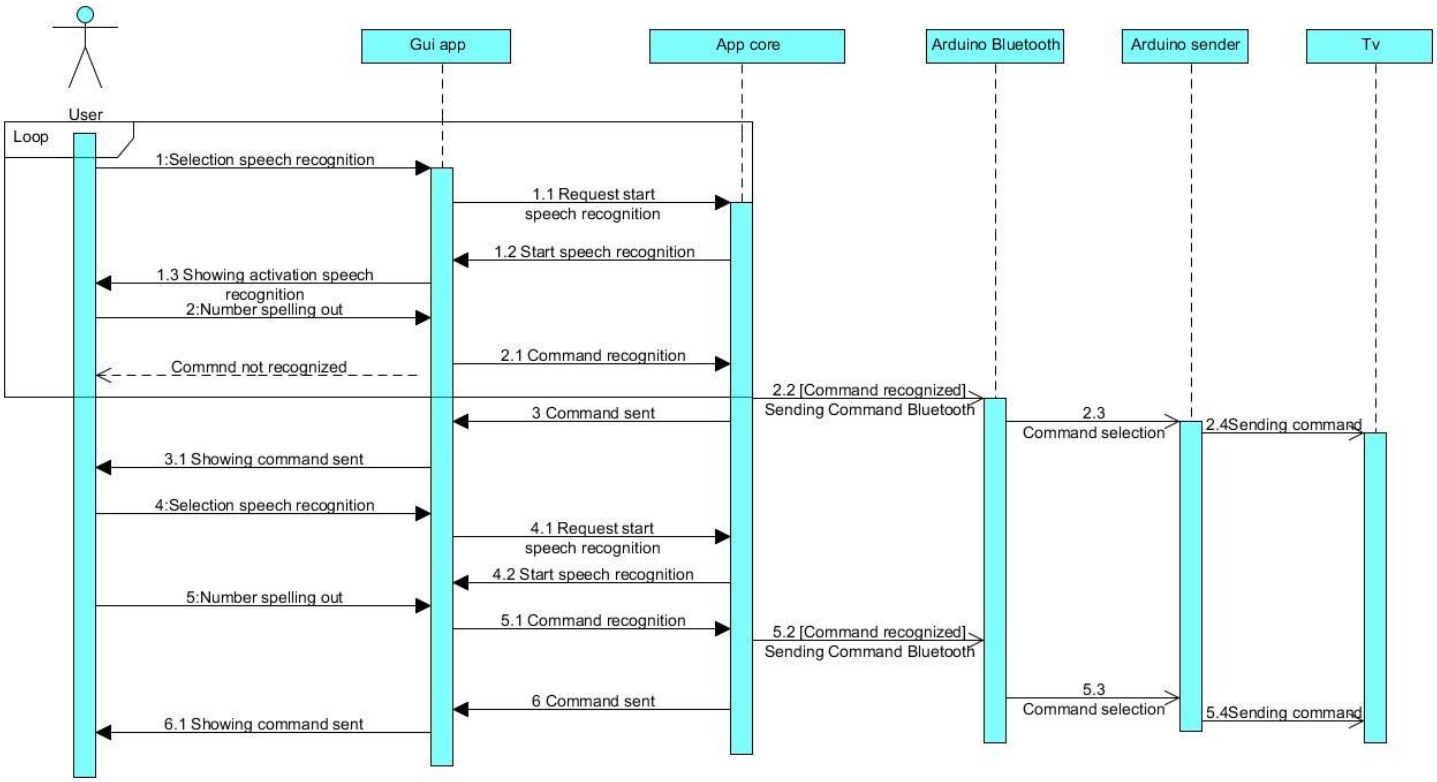


Figure 28 Sequence Diagrams of change channel from 10 to 999

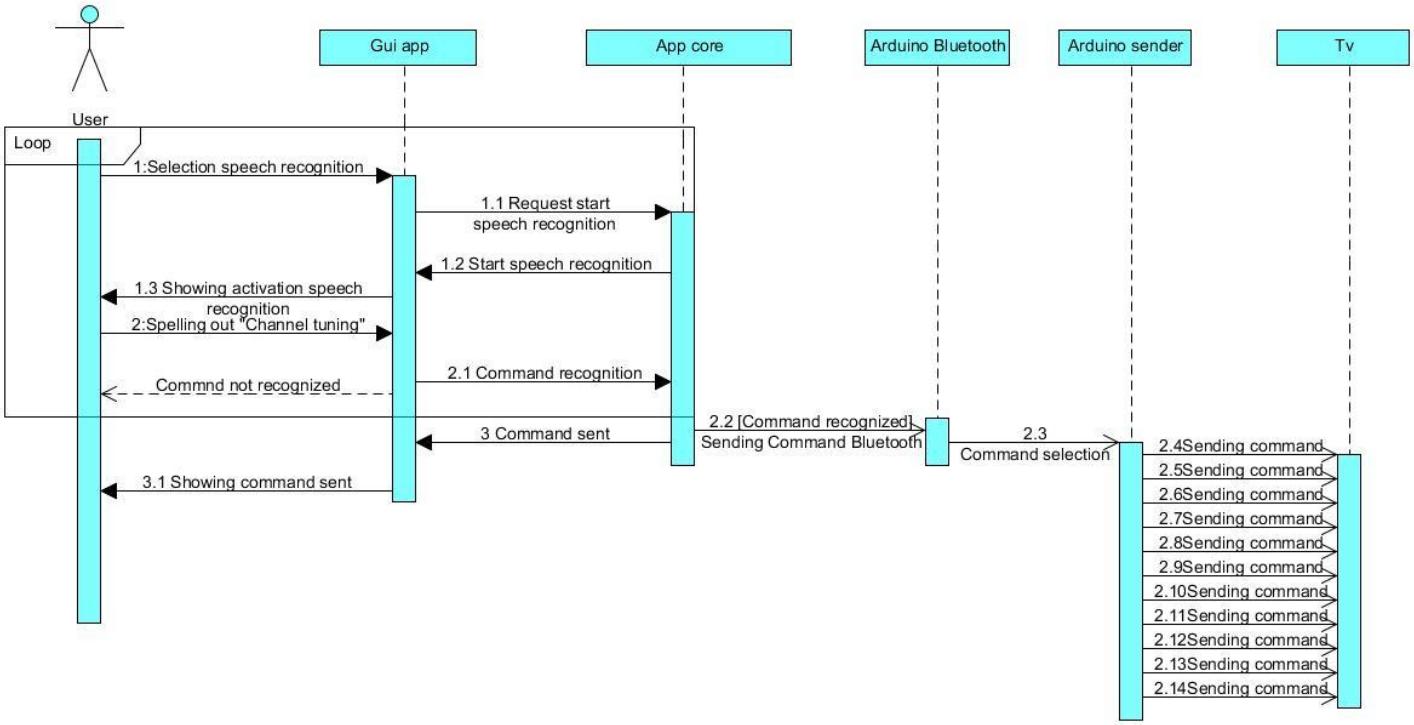


Figure 29 Sequence Diagrams of channel tuning

Chapter 3

Scenario

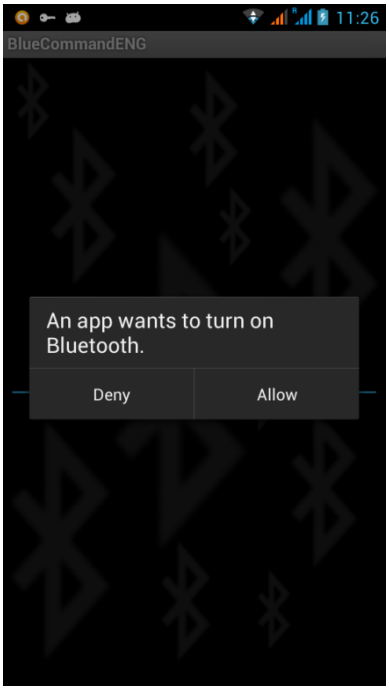
In this chapter some possible scenarios for the usage of the app are presented. The scenarios described are the followings:

Scenario 1: Access to the application and learning of the first functions

Scenario 2: Interaction with an app already present on the smart TV[7].

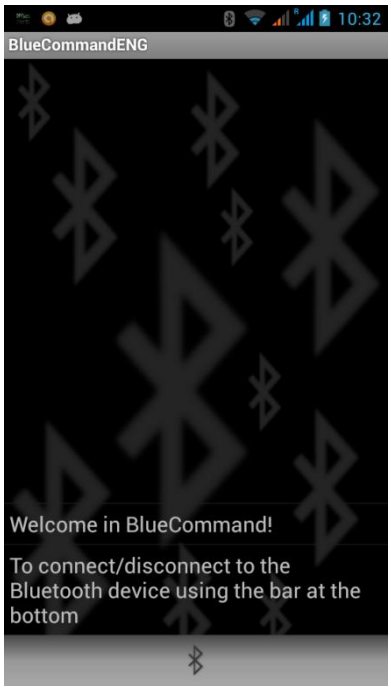
Scenario1: Access to the app and learning of the first functions

Kristo has just installed the app BlueCommand on his smartphone and he wants to start learning its basic functions.



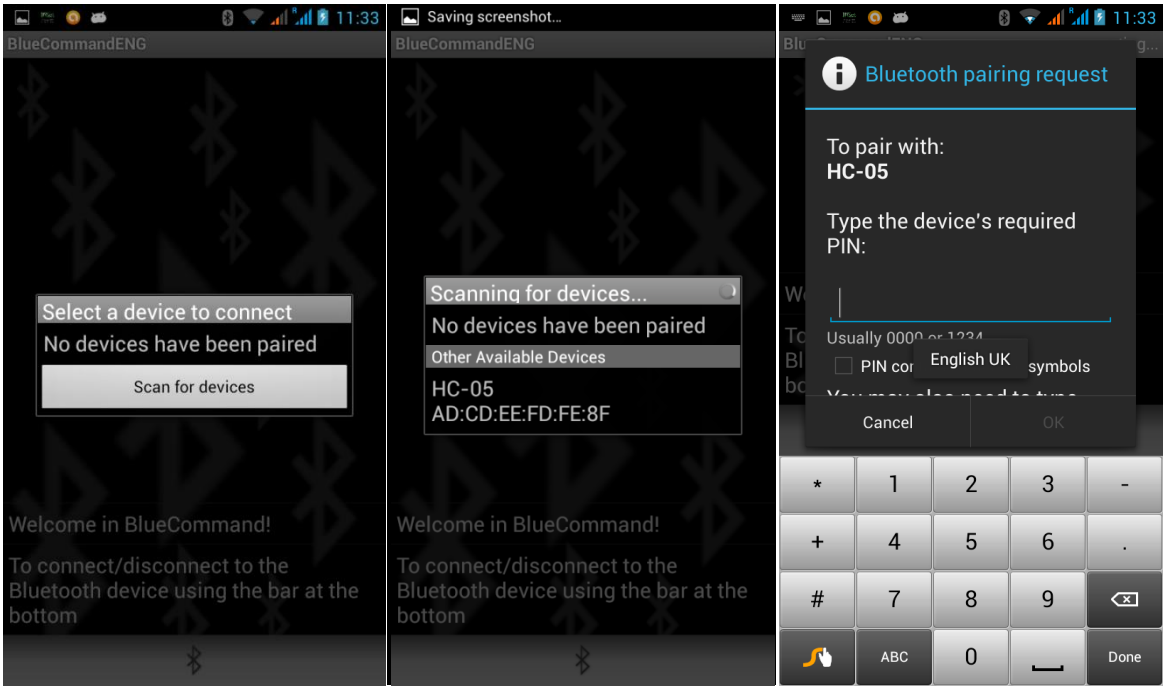
Screen 1 App – App opening screen. A message asking the user for the authorization to open the Bluetooth connection. If the user pushes the button "Allow" he/she can move to the second screen, whereas if it pushes "Deny" the app closes itself.

Now Kristo can push the button to associate the Bluetooth module of Arduino to his smatphone.



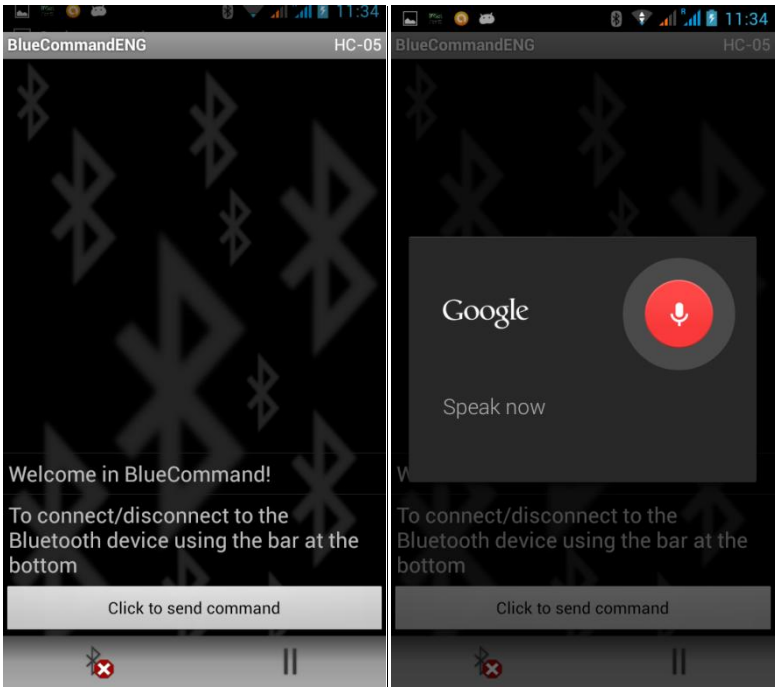
Screen 2 App – This screen show the welcome message for the user. In the lower part of the screen there is a button with the Bluetooth symbol to connect the phone to the device.

Still no devices have been paired. Kristo, then, decides to search for a near device and waits for the Bluetooth devices to be visualized on the screen. Later, he associates the Bluetooth module (HC-05) of Arduino to his smartphone by typing the PIN.



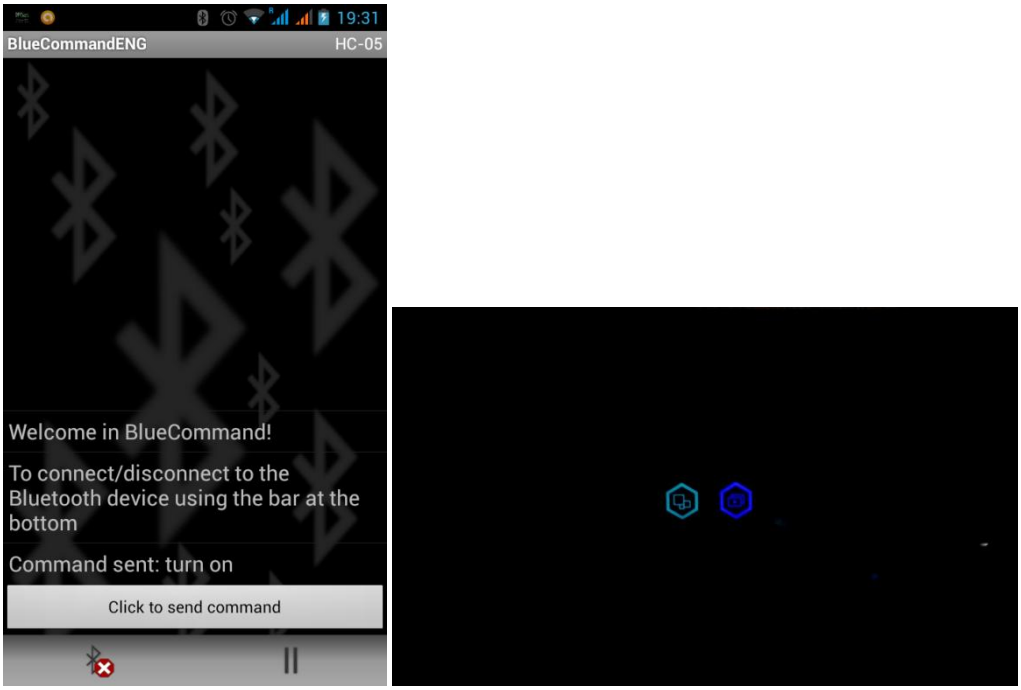
Screens 3,4,5 App – Screen number 3 shows a message that list the devices that are currently paired. If no device has been paired that the app suggests the user to search for near devices through the button "Search for devices". Screen number 4 shows the found devices. The user by pushing the name of the device chose what device he wants to pair. Screen 5 allows the user to type the secret PIN to be paired to the device.

Now Kristo can click on the button to activate the speech recognizer and the program waits for a vocal command.



Screen 6,7 App – Screen number 6 shows the button "Press to send the command" through which the user can activate the speech recognition. Screen number 7 shows an example of a screen with the voice recognition ready to receive the vocal command from the user.

Kristo wants to turn on his Samsung television and spell the command "Turn on". The television turns on.



Screen 8 App, Screen 9 television – Screen number 8 shows the vocal command "Turn on" spelled by the user on the smartphone screen. Screen number 9 shows the television turning on.

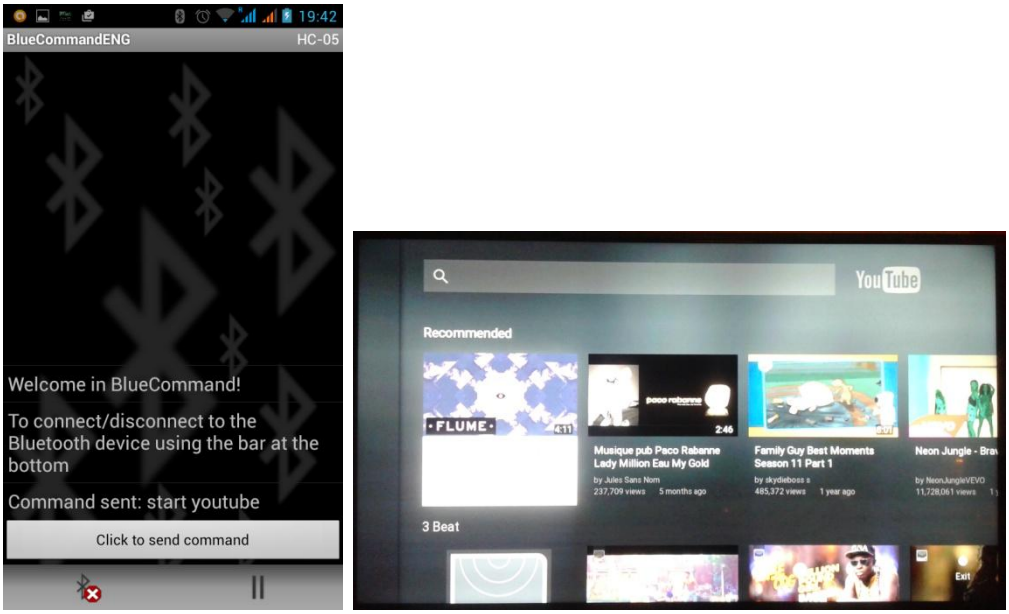
Scenario 2: Interaction with an application already present on the smart TV

Piret has a Smart TV Samsung and wants to see a video on Youtube.

She has two possibilities:

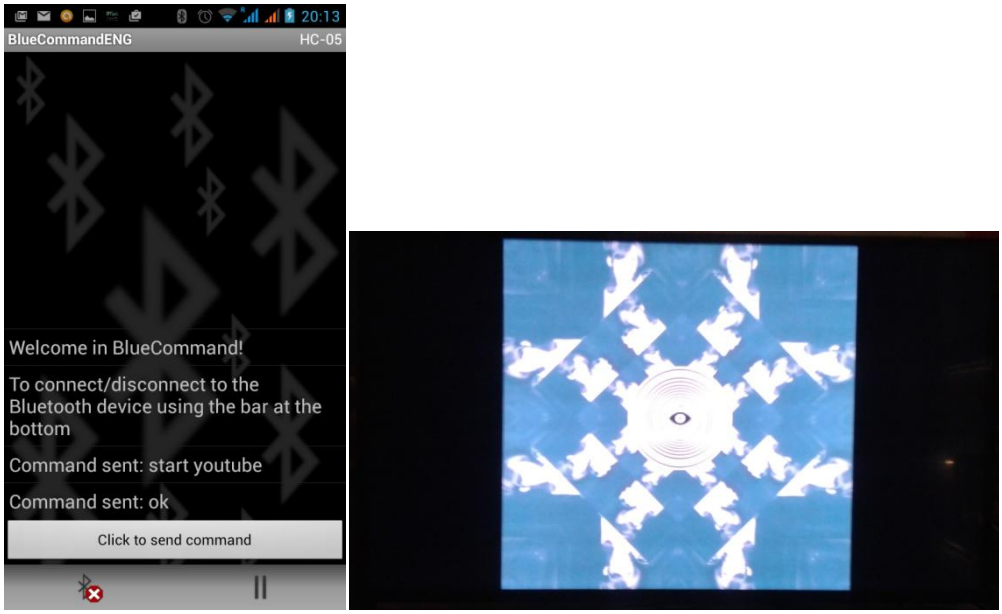
1. Spell the sequence of vocal commands to turn on the app;
2. Spell directly the command “StartYoutube”.

Piret choses the second option and, after waiting for the application to open finds immediately the video she was looking for.



Screen 1 App, 2 Television – Screen number 1 shows the vocal command “StartYoutube” spelled by the user on the app screen. In screen number 2 the app “Youtube” has been opened and the user can chose the video by spelling the desired sequence of commands.

She then decides to pick that video by spelling “OK”.



Screen 3 App, 4 Television – Screen number 3 shows the vocal command “ok” spelled by the user on the app screen. In screen number 4 the video has been opened.

Conclusions

The development of the system has been a great opportunity to deepen the development of Android apps in general as well as some aspects of Arduino.

The goal of the project has been that of creating a vocal remote control which allows to manage a Smart TV Samsung. To reach such a goal an Arduino and a smartphone Android have been used. The former is used to send the IR signals to the TV whereas the latter to carry out the speech recognition. One important element in my project has been the desire to facilitate the usage of the TV by the user. Goal that has been achieved by notably simplifying some complex functions. As a matter of fact the user, by simply spelling one command. Carries out actions that with a simple remote control would be difficult.

Possible future developments

The system that has been developed is a prototype to manage a Samsung TV; however in the future could be used also for other TV brands by changing part of the Arduino code.

In addition to the TV, this programme could also be used to manage other devices like decoders, home theatre and DVD reader. In particular, for the decoder the system will be used as remote control repeater. Since the Bluetooth allows the signal to pass through the walls, it will be enough to position the Arduino card near the decoder and use the smartphone in another room where both the audio and video signals already arrive.

Figure 30 Schema of remote control repeater

The developed system will be able to manage whatever device able to receive commands through an IR receiver. It will be possible to manage several devices at the same time through the simple use of the voice. For example: TV, decoder, rolling shutter and air conditioning.

Overall, it is possible to state that such project represents a useful starting point for useful developments in different fields.

Bibliography

[1] Android bluetooth Api [Last Access 03/01/2015]. [Online]. Avaible: http://developer.android.com/guide/topics/connectivity/bluetooth.html

[2] Arduino uno [Last Access 03/01/2015]. [Online]. Avaible: http://arduino.cc/en/Main/arduinoBoardUno

[3] Library iremote [Last Access 03/01/2015]. [Online]. Avaible: https://github.com/shirriff/Arduino-IRremote

[4] Tutorial Arduino ir sender ir receiver [Last Access 03/01/2015]. [Online]. Avaible: http://garagelab.com/profiles/blogs/tutorial-arduino-ir-sender-and-receiver

[5] IR management [Last Access 03/01/2015]. [Online]. Avaible: https://learn.sparkfun.com/tutorials/ir-communication/all

[6] Android and Arduino bluetooth communication [Last Access 03/01/2015]. [Online]. Avaible: http://bellcode.wordpress.com/2012/01/02/android-and-arduino-bluetooth-communication/

[7] Smart TV samsung [Last Access 03/01/2015]. [Online]. Avaible: http://www.samsung.com/it/consumer/TV-audio-video/smart-TV/

[8] Bluetooth HC-05 module [Last Access 03/01/2015]. [Online]. Avaible: http://www.exp-tech.de/service/datasheet/HC-Serial-Bluetooth-Products.pdf

[9] Api speech recognition [Last Access 03/01/2015]. [Online]. Avaible: http://developer.android.com/reference/android/speech/package-summary.html

[10] Android speech recognition exemple [Last Access 03/01/2015]. [Online]. Avaible: http://android-developers.blogspot.it/2009/09/introduction-to-text-to-speech-in.html