Development of an automatic sign language interpreter model

Kalinin Daniil Evgenevich, Radkin Kirill Alexeevich

«Lyceum № 97 of Chelyabinsk », 10-M1 form

Scientific adviser:

Sakanov Damir Muratovich,

teacher of additional education

«Lyceum № 97 of Chelyabinsk »,

Chelyabinsk city

Content

[Chapter 1. The Study of literature 5](#_Toc9626219)

[1.1. The study of similar scientific articles 5](#_Toc9626220)

[1.2. The study of sign language 5](#_Toc9626221)

[1.3. Study of the Arduino microcontroller 5](#_Toc9626222)

[1.4. Development of the scheme of the model 6](#_Toc9626223)

[Chapter 2. Model Assembly 7](#_Toc9626224)

[2.1. Creating parts 7](#_Toc9626225)

[2.1.1. 3D-Model 7](#_Toc9626226)

[2.2. Build process 7](#_Toc9626227)

[2.2.1. Fingers assembly 7](#_Toc9626228)

[2.2.2. Forearm and wrist assembly 7](#_Toc9626229)

[2.3. Hardware connection 8](#_Toc9626230)

[2.4. “Strela” 8](#_Toc9626231)

[Chapter 3. Software part 8](#_Toc9626232)

[3.1. Hardware part programming 8](#_Toc9626233)

[3.1.1. Choosing the programming language 8](#_Toc9626234)

[3.2. Creating the library 9](#_Toc9626235)

[3.2.1. Navigation matrix 9](#_Toc9626236)

[3.2.2. Symbol translate function 10](#_Toc9626237)

[3.2.3. Sentence translate function 10](#_Toc9626238)

[3.3. Develop of the Android application 10](#_Toc9626239)

[3.3.1. Text control mode 11](#_Toc9626240)

[3.3.2. Control due to speech recognition 11](#_Toc9626241)

[3.3.3. Manual control mode 11](#_Toc9626242)

[3.3.4. Localization of application 12](#_Toc9626243)

[3.4. Data transfer implementation 12](#_Toc9626244)

[3.4.1. Implementation from Android application 12](#_Toc9626245)

[3.4.2. Implementation from the microcontroller 13](#_Toc9626246)

[Conclusion 16](#_Toc9626247)

[Bibliography 18](#_Toc9626248)

Abstract

People can accomplish anything. That means that there are no obstacles for the one who really wants to do something. Unfortunately, there is a social group of people with hearing disabilities. We believe that “disabilities” should be only a formal name without any relations to reality. Therefore, our purpose is to do our best to bring this idea to life.

The problem relevance is in the fact that such works have not been written in Russia yet, however, people with hearing disabilities exist in Russia too. The project objective is to develop the first in Russia automatic sign language interpreter to remove restrictions of communication between all people’s social groups. We have set ourselves the number of task. Here it is:

1. To develop the project of programmable sign-language interpreter;
2. To write a program for the prototype;
3. To “teach” the model to recognize speech automatically and translate it into sign language.

# The Study of literature

## The study of similar scientific articles

Before the development of the project search and analysis of articles were conducted on such bases as Scopus and Google Scholar. No scientific articles, describing the similar project, were found. Therefore, we studied forks of foreign authors. It’s turned out to be not as much as we expected.

## The study of sign language

After studying research papers in the database we have begun to study the sign language. It turned out that there are many different variations of this language, used to show whole words and phrases. A head, a body, shoulders and hands are required for images of these gestures. But as each gesture can be thought of as a few simple gestures denoting just one letter, it was decided not to take into account such “hard” gestures while creating and training the model. In Fig. 1[[1]](#footnote-1) "Sign alphabet" is presented, the Russian alphabet translated into sign language.

## **Study of the Arduino microcontroller**

After studying sign language, we started to study the literature related to our model’s acting device – Arduino microcontroller and Arduino IDE – Integrated Development Environment. This microcontroller manufacturer was chosen for meeting all project requirements. Exactly, it had high computer power, enough ports to connect to a variety of third-party modules and was easy to learn.

Moreover, Arduino development environment is based on C++ language, which is known by us from the school program. Despite similar logic and syntax, the Arduino IDE and C++ are two completely different languages. We have to study a lot of literature to learn the Arduino language.

## Development of the scheme of the model

We have studied the existing scheme of models which our foreign colleagues used. 3D-files for printing on 3D printer were a large part of them. After studying seven of these models we concluded that despite the simplicity of manufacturing (all parts are just printed on 3D printer without people participating and then just gathered as usual Lego) they do not have a sufficient ratio of strength to lightness. ABS-plastic from which models are printed is quite breakable and we have to print models using thick layer of plastic to achieve sufficient strength (or we must use more expensive printing technology), as a result, we have a too heavy prototype.

While we were finding model drawings on the WEB, we prepared several variants of our own drawings for developing the model without 3D technologies. But as the comparative analysis and open databases have showed, using 3D printer while creating a model is the easiest way. Firstly, it simplifies the task, secondly, it’s the most accurate method which is really important in our project. Moreover, if you create parts of model according to our drawings, it turns out that Styrofoam, which we expected to exploit because of its lightness, heavily proceed without special tools. Also Styrofoam is destroyed by the twine, which, as the design suggests, should rotate parts (fingers of a hand). That’s why we have decided to use drawings from French open-sours 3D-drawings database named “InMoov”.

# Model Assembly

## Creating parts

### 3D-Model

Besides finding drawing, it was necessary to find 3D-printer where we can print details. There is one in our school. It uses Fusing Deposes Modeling technology. This technology is widespread for its availability and cheapness. That’s why we have decided to use this technology in our project.

## Build process

### Fingers assembly

The first step in the process of assembly was fingers assembly.

Fingers are built from three phalanx and every phalanx is built from two parts. Each phalanx has holes for connecting to each other. Hard wire, which is used as a rotation axis, is put in these holes. Also there are two string inside each finger. They are used for fingers’ bending and extending.

### Forearm and wrist assembly

The wrist is made up from several parts: the massive detail is used as the wrist’s base, three moving details are attached on this base and fingers are attached on them. The wrist is attached on the forearm with system of gears, which allow is to rotate around its axis (Fig. 2).

Forearm is made up from two parts, looking as a half of cylinder. The model becomes more compact and easier to use due to integrated electronics.

## Hardware connection

It was necessary to connect Arduino, Bluetooth module, servo drivers and LCD display (it shows data about current fingers rotation and which letter the model is being showed at the point) for correct auto-interpreter work.

Power from a regular power bank is enough for Arduino, Bluetooth module, LCD display, but it is not enough for five servo drivers.

We had to resolve the problem with servos power. A power bank’s amperage was too small for five drivers. Solution was found in additional power module installation.

## “Strela”

“Strela” is the product of “Amperka” company. This is a platform, manufacturing as an additional expansion board. Strela is an another Arduino microcontroller with several shields[[2]](#footnote-2) on it. We use “strela” as an additional power resource. Taking typical for Arduino voltage, it powers five servos.

# Software part

## Hardware part programming

### Choosing the programming language

We have chosen C++ language to program the hardware part. The compiler, which is used to convert program code to binary code (microcontroller processor can only read this type of code), was adapt for C++ (Arduino language is based on C++).

As programming environment we have chosen the one, which is purposed by Arduino manufacturer, Arduino IDE. This IDE was chosen because it has, except a standard function that exist in all kind of environment, special features for working with Arduino. For example, it has settings to setup connection to microcontroller, options to view the streaming port, many built-in libraries for simplifying the work with Arduino, and also it has parts of ready-to-use code for different tasks.

## Creating the library

It was decided to write our own library of functions to control the model, while writing the software. Firstly, library, which unites all function to control the model, makes our work easier, secondly, it increases code readability (deletes unnecessary functions and unites others), and finally, after the end of the work this library with code and description of building process can be posted on different open-source sites (e.g. GitHub, HabraHabr etc.), for everyone’s ability to take the same model at home.

Let’s talk about library’s functions, representing the greatest interest.

### Navigation matrix

Navigation matrix is the main part of the library and the symbol-translation system (we’ll talk about it later). Navigation matrix is the two-dimensional array of size 32 by 6. There are symbols (letters) codes in the first column in CP1251-table (it describes below). So, the number of table rows considers with the number of letters in Russian alphabet. There are only lowercase codes in the table (uppercase letters is converted to lowercase by the program code). The number of columns was not chosen by chance. Starting with the second column, there are rotation angles for each finger (in the second column there are data for thumb, in the third for index finger etc.). So in each rows we have a following information: there is letter CP1251-code in the first column of each row, and there are rotation angle for each model’s finger to show this letter in other five columns of each row.

### Symbol translate function

As we have said before, symbol translation function is one of the main functions in the library. It works by the following algorithm. It takes one letter (symbol), which you want to translate to sign language, as an input data. Then it converts the letter to the code of this letter, according to the CP1251 table, and finds the code of this letter in the first column of the navigation matrix. When the code is found, the program remembers the number of the column where this code was and rotates each servo driver to the rotation angle, which is wrote in the second, third, fourth, fifth, and sixth column of this row. This is the fastest way to translate each symbol, despite that the data volume is really big (six numbers for each later).

### Sentence translate function

For simplify work with the model we decided to develop the sentence translate function. This function is based on work of previous function, which was described above, and, in fact, this is an add-in function. Here, as input data, you can give a whole sentence, which you want to be translated. Using cycle, named “for”, this function splits sentence into individual letters, expects the spaces and turns uppercases into lowercases. Then this function calls symbol translate function for each letter. This is important, that if this function “sees” the space in the sentence, it will make a pause (separate two words to simplify reading), because there is no sign for space in sign language.

## Develop of the Android application

We have decided to develop the remote model controller android application, which sends users commands from a phone to the model, because of several reasons. Firstly, it helps to control the model without many buttons on it. Secondly, it makes model control easier. The last reason, for someone, who wants to build our model in home from open resources, it is easier to download our app than to build a difficult remote control board.

When you start the application, you can see the home screen with three buttons for selection of control mode (Fig. 3).

There are three control modes. A control by entering the text, a control by speech recognition (voice mode), and the manual mode, where a user can choice rotation angle for each finger manually. The last mode could be used when a user wants to use hand to show a sign, which is not provided in our library, or to use model as a manipulator.

### **Text control mode**

If user presses the button “Text mode”, a new screen will be opened. Text control mode is carried out as follows. User enters the text, which he wants to be translated to sign language, using integrated virtual keyboard, then closes it and presses the button “Translate text”. Text is read from special field by program, then application connects to hand prototype, using Bluetooth (if something goes wrong, user will be noticed by the following message), and, as the last step, text is converted to binary type and sent to the model.

### Control due to speech recognition

If user presses the button “Voice mode” on the home screen, another screen will be opened. In this mode user can type text manually, like it goes in previous mode, but also he can press the button “Press and speak” and tell the phone a text which he wants to be translated. User’s speech will be recognized by phone, so text will be typed at the special field automatically. The last step is to press the button “Translate”.

### Manual control mode

After pressing the button “Manual mode”, the new screen with five Seek-bar[[3]](#footnote-3) will be opened. There is a little box for the right of each scale with scale value, which is written inside it, and a finger name, which rotation angle is controlled by scale. Using this scale, user can choose an individual angle for each model’s finger. After choosing angles, user should press the “Send” button and all angles will be sent to the model according to algorithm, which we will describe later.

### Localization of application

One thought came to us, while we were working with this project. What if foreigners, who don’t know Russian, will be interested in our project? If so, they all will have problems with model control. Therefore, we have decided to translate our app into English. It was a really easy task, especially to do it in Android Studio[[4]](#footnote-4) because there are special developer translation settings.

So, if user changes his phone’s system language to English, app will be available on English too.

## Data transfer implementation

### Implementation from Android application

After writing a basic part of the program, which included interface design (development the image of program components), communication interface with a part of button depression treatment, switching between screens of application, etc., we have taken up the implementation of Bluetooth data transmission channel. After studying the literature on the subject of "communication Android device and microcontroller" we implement this as follows.

To run the application, user's device must have a module capable of implement Bluetooth data transfer, also this device must be enabled. Every time you start the application, these two conditions are checked, and if the Bluetooth is off, the user is prompted to activate it without leaving the application (Fig. 4). When you start any activity (one of the three modes of control) operation of Bluetooth connection is started in a separate stream[[5]](#footnote-5), attach a standard Bluetooth adapter of the device, and then you install the Bluetooth connection and the programm sends to the streaming port to 1 or 0, depending on which screen is open (in manual mode 0 is sent , in the other two 1 is sent). After the user enters the text or sets values on scales and presses the button "Translate text" or "Send data" (depending on the chosen control mode). At this point, the system processes the click: splits the entered string into an array of bytes (one character is encoded as one byte and written out to the element array), then each element of the array is transfered sequentially through streaming Bluetooth output port. Then the port is closed and the user gets the toast[[6]](#footnote-6) that the transfer was successful. In case of an error with the establishment of links with standard Bluetooth adapter, user also gets toast message with a detail information about error.

### Implementation from the microcontroller

At first glance, the implementation of data reception from microcontroller seemed to be easier than on an Android device. At first it was really so. It’s required to connect Bluetooth module[[7]](#footnote-7) to input/output streaming port, and continue working with it as with a linear device, i.e. open streaming port and take data from there. The problem was the following. Data are transferred in numeric form, so it was impossible to determine for sure what kind of data we have transmitted from an Android device: numeric encoding of letters from the first two modes (text and voice control mode), or the rotation angle from the third control mode (manual mode Moreover, it was soon discovered that the character data (text) are not corresponding to any encoding table (so was due to differences in Android and encodings used Arduino).

The problem with different encodings has been being resolved for a long time, but a solution is quite simple. To get the character code, it was necessary to add to the each character code the number 176 (determined experimentally), in this case we get the character code for CP1251-table.

It was proved much more difficult to determine what we share: character code or numerical value. To resolve this problem, data flags were used. When you start your activity on an Android device after receiving a connection with Bluetooth-adapter opens streaming output port, where is recorded 1 or 0, according to the chosen control mode (for text and voice modes it’s 1, for manual it’s 0). By microcontroller side streaming port opens, and the flag is recorded in the corresponding variable, then the port became null and void until the user clicks the button "Translate text" or "Send data ". Receiving the data, the microcontroller will already know the kind of it, targeting the flag.

The logic of the Arduino IDE language is designed so that it has two types of program operation: setup and loop. The commands in the setup run only once when the controller is switched on, the commands in the loop are repeated cyclically while the controller works. In this regard, it was necessary to suspend the reading of data from streaming port after sending a flag, but before sending the data. Otherwise the program would receive a flag, and after reading data would start to where they do not exist (the user has not sent the data, therefore, streaming port is empty), this program would work correctly.

The rewriting of the flag was an another problem. The problem also arose because of the language logic discussed above. While sending text or numeric data again, the program started reading the flag again and taking the first bit (the first character or the first number) of the received data as a flag because of the block loop in which the functions were placed. As a result of the work we have such a program where if we resend the word "Hello" from an Android device on the microcontroller, we will get the word "ello" (if you can call it like that), and the flag will be equal to 63, which corresponds to the numerical encoding of the letter "H" on the table CP1251 (the encoding of the letter "H" on this table – 239), from which 176 was taken away (this phenomenon was mentioned above), i.e. the program received an incomprehensible word, and in addition did not know the word or number: the flag matched neither 0 nor 1. Fortunately, the problem was solved quite simply: before writing the flag, it was necessary to check whether the number that we want to write to the flag is 1 or 0 (letters with such encodings do not exist, and the rotation degrees were sent starting from 5 in order not to confuse with the flag).

Another technological nuance was that the sign language is not divided into upper and lower case letters, but, despite this, the code of the same upper and lower case letters will be different[[8]](#footnote-8). The solution was that due to the peculiarities of the CP1251 code table (its Cyrillic characters are arranged in order, which is not present in all code tables), the code of the lowercase and uppercase letters differ by a constant value equal to 32. Thus, if we know for sure that we have received a character string (the conclusion can be made based on the value of the flag), and if the character code lies in the range from 192 inclusive to 224 inclusive (Fig. 5), we can be sure that the given symbol is an uppercase letter, and when we add this symbol to the code 32, we will get the same letter, but a lowercase one.

The last problem, unfortunately, is not fully solved yet. It’s about the fact that the space character was transmitted as 32 (this is really the code of the character "space", without an offset of 176). The code equal to 32 for the space is the UNICODE standard, the same for all devices (Cyrillic is not included in this standard, where the problem with different encodings came from), but it is easy to notice that when you add a space to the 32 – code – 176, you get 208, which corresponds to the symbol "R" in the CP1251 table. That’s a pity, but, taking another flag every time you use a space will be too resource-intensive for a controller with a small (relative to smartphones and computers, for which such an operation would not be difficult) computing power. Therefore, it was decided to replace code 32 with a space character, since it was used more frequently and was more difficult to avoid using it than the capital letter "R". Thus, the working program while sending from Android-smartphone string "River" displays the word "iver". Several ways of solving this problem have been tried, but each of them harmed the work of the program more than this error. In the end, it was decided to leave it and notify a user from the Android device that the use of capital letters may cause errors in the operation of the program.

# Conclusion

In the process of working a model of the sign language interpreter manipulator was created from scratch, its software was implemented, convenient management was provided (a library for working with the manipulator was written). All tasks are solved, the purpose of work is reached. Despite the still unsolved shortcomings, the prototype is fully working and successfully functioning: it performs the transmitted commands, translates the text, and recognizes speech. In addition, the Android-application was written, which is also a finished product and allows you to implement model management functions.

In the future, it is planned to continue working on the project, including improving and upgrading the translator's software, for example, to add the ability to create your own gestures and use them in the future. It is also planned to improve the operation of the Android application by adding new functions of translation into sign language. There is a possibility of reverse translation realization in the plans: for example, a certain glove, which could translate gestures that a person, who is wearing it, shows into a text or a voice message displayed on the screen of his smartphone.

At the end of the work, an article was written (with the application of program code, Assembly instructions, explanations to them) on the resources of GitHub and HabraHabr. By placing the information in open-source sources, we tried to spread the project as much as it’s possible.

# Bibliography

1. Arduino Russia, [Electronic resource]// http://arduino.ru

2. Arduino, [Electronic resource]//https://www.arduino.cc

3. Cxem.net, [Electronic resource]//http://cxem.net/master/45.php

4. Google Academy, [Online]]// https://scholar.google.ru.

5. Mypractic, [Electronic resource]// http://mypractic.ru/urok-23-podklyuchenie-zhk-lcd-indikatorov-k-arduino-biblioteka-liquidcrystal.html

6. Amperka, [Electronic resource]// http://amperka.ru

7. Arduino hardware platform, [Electronic resource]//http://arduino.ru

8. Database of abstracts and citations and papers, [Electronic resource]//  
https://www.scopus.com.

9. Bloom Jeremy. Studied Arduino: tools and techniques of technical wizardry: Per. from English. — SPb.: BHV-Petersburg, 2015. — 336 p.: Il.

10. Fradkina R. N. Talking hands: M.: "Refl-book", 2001.– 402 pages.

**Appendix**



Fig. 1 Russian sign language.



Fig. 2 Model’s forearm.



Fig. 3. The main app activity.



Fig. 4. Activating the Bluetooth.



Fig. 5. Code page CP1251.

1. This and all others figures are in «Appendix» section under its own number. [↑](#footnote-ref-1)
2. Shield – Some chip, put on a microcontroller or other chip, expanding (or any other way complementing) it. [↑](#footnote-ref-2)
3. Seek bar – one of integrated android system widgets. It is just value selection scale. [↑](#footnote-ref-3)
4. Development environment for android developing. [↑](#footnote-ref-4)
5. The decision to place the installation of a Bluetooth connection in a separate stream was made due to the fact that this operation is quite resource-intensive (requires a lot of RAM), and if this action performs in the main stream, the application will "hang" (stop responding to user actions) for the duration of the Bluetooth connection operation. [↑](#footnote-ref-5)
6. Toast – A professional name for a notification type in which a notification, like a bread toast from a toaster, pops up from the bottom and disappears after a few seconds. [↑](#footnote-ref-6)
7. The HC-06 model was chosen because it is one of the newest models of Bluetooth modules, supporting Bluetooth version 2.1+EDR specification, which guarantees reduced power consumption, increased level of data protection and easy connection of Bluetooth-devices, in addition, stable reception with the module is guaranteed within 10 meters. [↑](#footnote-ref-7)
8. The justification for the fact that this "transformation" is necessary for the correct operation of the program can be found by understanding the principle of the functions of translating characters into sign language, recorded in the library we created. [↑](#footnote-ref-8)