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

Coz ntent

[Chapter 1. The Study of literature 5](#_Toc9446248)

[1.1. The study of similar scientific articles 5](#_Toc9446249)

[1.2. The study of sign language 5](#_Toc9446250)

[1.3. Study of the Arduino microcontroller 5](#_Toc9446251)

[1.4. Development of the scheme of the model 6](#_Toc9446252)

[Chapter 2. Model Assembly 7](#_Toc9446253)

[2.1. Creating parts 7](#_Toc9446254)

[2.1.1. 3D-model 7](#_Toc9446255)

[2.2. 2.2. Процесс сборки 7](#_Toc9446256)

[2.2.1. 2.2.1 Сборка пальцев 7](#_Toc9446257)

[2.2.2. 2.2.2 Сборка предплечья и запястья 7](#_Toc9446258)

[2.3. 2.3. Подключение аппаратуры 8](#_Toc9446259)

[2.4. 2.4 «Стрела». 8](#_Toc9446260)

[Chapter 3. Глава 3. Программная часть 9](#_Toc9446261)

[3.1. 3.1. Программирование аппаратной части 9](#_Toc9446262)

[3.1.1. 3.1.1. Выбор языка и среды программирования 9](#_Toc9446263)

[3.2. 3.2. Написание библиотеки. 9](#_Toc9446264)

[3.2.1. 3.2.1. Навигационная матрица 10](#_Toc9446265)

[3.2.2. 3.2.2. Функция перевода символов 10](#_Toc9446266)

[3.2.3. 3.2.3. Функция перевода предложений 11](#_Toc9446267)

[3.3. 3.3. Создание Android-приложения 11](#_Toc9446268)

[3.3.1. 3.3.1 Управление посредством ввода текста 12](#_Toc9446269)

[3.3.2. 3.3.2. Управление посредством распознавания речи 12](#_Toc9446270)

[3.3.3. 3.3.3. Ручной режим управления 12](#_Toc9446271)

[3.3.4. 3.3.4. Локализация приложения 13](#_Toc9446272)

[3.4. 3.4. Реализация передачи данных 13](#_Toc9446273)

[3.4.1. 3.4.1. Реализация со стороны Android-приложения. 13](#_Toc9446274)

[3.4.2. 3.4.2. Реализация со стороны микроконтроллера 15](#_Toc9446275)

[Chapter 4. Заключение 18](#_Toc9446276)

[Chapter 5. Список литературы 20](#_Toc9446277)

[Chapter 6. Приложение 21](#_Toc9446278)

Abstract

People can accomplish anything. That means that there are not 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 formal name without any relation to reality. Therefore, our purpose is do our best to bring that to live.

The problem relevance is in the fact that such works had not written in Russia yet, however people with hearing disabilities exist in Russia too. The project objective is to develop 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 was conducted search and analysis of articles on such bases as Scopus and Google Scholar. Was not found any scientific articles, describing the similar project. 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. For images of these gestures require the head, body, shoulders, and hands. But because 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 when creating and training the model. In Figure 1[[1]](#footnote-1) presented "Sign alphabet"-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 lern.

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. Large part of them was 3D-files for printing on 3D printer. 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 which models are printed from is quite breakable and we must print models using thick layer of plastic to achieve sufficient strength (or we must use more expensive printing technology), as a result, we have too heavy prototype.

While we finding model drawings on the WEB, we have 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 easiest way. Firstly, it simplifies the task and, secondly, it’s the most accurate method which is really important in our project. Moreover, if you are creating parts of model according to our drawings, it’s turned 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 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 is why we have decided to use this technology in our project.

## Build process

### Fingers assembly

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

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

### Forearm and wrist assembly

The wrist is made up from a several parts: the massive detail is using as the wrist’s base, three moving details are attached on this base and fingers are attached on these The wrist is attached on 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 what letter the model is showing at the point) for correct auto-interpreter work.

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

We have 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 platform, manufacturing as 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 using 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. Данная среда была выбрана за то, что помимо стандартных функций, присущих любой среде разработки, она обладает опциями для настройки подключения к микроконтроллеру, опциями просмотра потокового порта (монитора порта), множеством встроенных библиотек для упрощения работы с микроконтроллером, а также части готового встраиваемого кода для разных задач. Последнее, в частности, больше полезно для тех, кто только начал знакомиться с программированием, мы же данной опцией не пользовались в силу уникальности необходимого нам программного кода.

## 3.2. Написание библиотеки.

В ходе программирования аппаратной части было принято решение написать библиотеку функций для управления манипулятором. Во-первых, единая библиотека упростит нашу дальнейшую работу, улучшит читабельность программного кода: избавит его от лишнего объявления функций, а, во-вторых, после окончания работы данную библиотеку вместе с программным кодом и описанием процесса сборки можно будет выложить на различные open-source источники (источники открытого доступа к различным данным) для того, чтобы каждый желающий мог собрать такой же манипулятор.

Рассмотрим функции библиотеки, представляющие наибольший интерес.

### 3.2.1. Навигационная матрица

Главной частью библиотеки является разработанная нами навигационная матрица и система перевода символов (о ней сказано ниже). Навигационная матрица представляет собой двумерный массив размером 32 на 6. В первом столбце записаны коды символов по CP1251 (о ней будет сказано ниже) таблице. Таким образом, количество строк массива совпадает с количеством букв в русском алфавите (за исключением буквы «ё»). В матрицу занесены только кодировки строчных символов (кодировки одной и той же строчной и заглавной буквы окажутся разными). Обработка заглавных букв и приведение кодировки заглавной буквы к кодировке строчной происходит на микроконтроллере и будет рассмотрено далее. Количество столбцов тоже выбрано неслучайно: начиная со второго столбца (в первом записана кодировка символа), в каждом элементе массива записано число – угол поворота сервопривода. Таким образом, в одной строке навигационного массива содержится следующая информация: в первом элементе строки находится код символа по CP1251-таблице, в последующих пяти содержится угол поворота сервопривода для каждого пальца модели руки.

### 3.2.2. Функция перевода символов

Основной функцией в библиотеке является функция перевода символов. Работает она следующим образом. В качестве аргумента этой функции передается один символ, который необходимо перевести на язык жестов. Далее функция находит код переданного ей символа по таблице CP1251, а после начинает искать полученное число в навигационной матрице. Найдя число в некоторой строчке, программа записывает номер этой строки, а после устанавливает сервоприводы в соответствии с элементами матрицы. Таким образом обеспечена наиболее быстрая работа по переводу символов, несмотря на то, что объем данных (пять чисел для каждого символа) довольно большой.

### 3.2.3. Функция перевода предложений

Для простоты работы с устройством нами также была разработана функция перевода целых предложений. Данная функция основана на принципе работы предыдущей и, по сути, является ее надстройкой. В качестве параметра данной функции передается не отдельный символ, а целое предложение. Функция, используя цикл, разбивает его на отдельные символы, а затем передает каждый символ в функцию перевода символов. Важным технологическим нюансом является то, что в тот момент, когда функция, разбивая предложение, наткнется на пробел, она не станет отправлять его в функцию перевода символов (пробел не предусмотрен в языке жестов), а сделает задержку работы программы для имитации того, что отдельное слово закончилось и сейчас последует другое. Данное решение значительно упростит «чтение» жестов, показанных рукой.

## 3.3. Создание Android-приложения

Для того, чтобы упростить управление манипулятором и избавить его от многочисленных кнопок, было решено осуществлять управление через Android-приложение, передавая команды с Android-устройства по Bluetooth. Приложение при запуске представляет собой стартовый экран с возможностью выбора типа управления (рис. 3).

Всего предусмотрено три типа управления: управление через вводимый текст, управление через распознанную речь (управление голосом), а также режим ручного управления, когда пользователь имеет возможность вручную задать положение каждого пальца искусственной руки, что может быть полезно, например, при возникновении необходимости изобразить жест, отсутствующий в написанной библиотеке, или при использовании модели в качестве манипулятора.

### **3.3.1 Управление посредством ввода текста**

При нажатии на кнопку «Перевод текстом» перед пользователем появится новый экран. Управление посредством ввода текста осуществляется следующим образом. В специально отведенное текстовое поле пользователь вводит текст, который должен быть переведен на язык жестов, используя встроенную виртуальную клавиатуру устройства (обязательно имеется на каждом устройстве), после чего закрывает клавиатуру и нажимает на кнопку «Перевести текст». Программа устанавливает Bluetooth-соединение с прототипом руки (в случае ошибки установки соединения пользователь будет уведомлен всплывающим сообщением), после преобразует введенный текст к двоичному типу и передает на микроконтроллер.

### 3.3.2. Управление посредством распознавания речи

При нажатии на кнопку «Управление голосом» на стартовом экране откроется новое окно. В данном режиме пользователь имеет возможность как записать текст, вручную используя клавиатуру, так и нажать на кнопку «Нажмите и говорите», после чего откроется сервис распознавания речи, и пользователь сможет произнести тот текст, который должен быть переведен на язык жестов. После распознавания текст окажется в текстовом поле, и пользователь сможет нажать на кнопку «Перевести текст».

### 3.3.3. Ручной режим управления

После того, как пользователь нажмет кнопку «Ручной режим», откроется окно с пятью Seek bar[[3]](#footnote-3), на которых можно выбрать значение посредством перемещения ползунка. Справа от каждой шкалы находится небольшое поле, в котором указано то, каким пальцем искусственной руки управляет шкала, а также находящееся на ней значение. С помощью данных шкал пользователь может указать необходимое ему значение для каждого пальца отдельно. После чего он сможет нажать кнопку «Передать данные», и они будут переданы на микроконтроллер по описанному ниже алгоритму.

### 3.3.4. Локализация приложения

В процессе работы над Android-приложением у нас возникла следующая мысль: а что, если наш проект захотят повторить иностранцы, не знающие русского языка. В таком случае у них возникнет проблема с управлением моделью. Было решено перевести приложение на английский язык. Это оказалось не так сложно, как ожидалось, тем более, в Android Studio[[4]](#footnote-4) предусмотрен специальный редактор для перевода приложений

Таким образом, если в настройках устройства сменить системный язык на английский, приложение будет полностью на английском.

## 3.4. Реализация передачи данных

### 3.4.1. Реализация со стороны Android-приложения.

После написания базовой части программы, в состав которой входили: дизайн интерфейса (разработка образов компонентов программы), связь интерфейса с исполнительной частью кода, обработка нажатий на кнопки, переключение между экранами приложения и т.д., мы взялись за реализацию Bluetooth-канала передачи данных. После изучения литературы по теме «Общение Android-устройства и микроконтроллера» было реализовано следующим образом.

Для работы приложения на устройстве пользователя должно присутствовать устройство, способное реализовать Bluetooth-передачу данных, также это устройство должно быть включено. При каждом запуске приложения эти два условия проверяются, и в том случае, если Bluetooth выключен, пользователю предлагается его включить, не выходя из приложения (рис. 4).

При запуске какой-либо активности (открытии одного из трех режимов управления) в отдельном потоке[[5]](#footnote-5) запускается операция подключения стандартного Bluetooth-адаптера устройства, после чего устанавливается Bluetooth-соединение, а в потоковый порт отправляется 1 или 0, в зависимости от того, какой экран открыт (в ручном режиме – 0, в двух остальных – 1). После чего пользователь вводит текст или устанавливает значения на шкалах и нажимает кнопку «Перевести текст» или «Отправить данные» (в зависимости от выбранного режима управления). В этот момент система обрабатывает нажатие на кнопку: разбивает введенную строку на массив байтов (один символ кодируется одним байтом и записывается в свой элемент массива), после чего каждый элемент массива последовательно передается через потоковый Bluetooth-порт вывода. Затем порт закрывается, и пользователь получает тост[[6]](#footnote-6) о том, что передача прошла успешно. В случае ошибок с установлением связи со стандартным Bluetooth-адаптером, при выключенном Bluetooth на устройстве (если пользователь отказался его включать), при отсутствии Bluetooth-модуля на устройстве, в случае ошибок во время открытия потокового порта (процесс сам по себе очень сложный, поэтому даже несмотря на то, что данная функция реализована профессиональными разработчиками компании «Google», в ней могут возникать ошибки из-за разных независящих от пользователя условий), и в случае других ошибок пользователь будет уведомлен тостом.

### 3.4.2. Реализация со стороны микроконтроллера

На первый взгляд, реализация приема данных на микроконтроллере казалась во много раз проще, чем на Android-устройстве. Поначалу действительно было так: необходимы было подключить Bluetooth-модуль[[7]](#footnote-7) к портам потокового ввода-вывода, а далее работать с ним, как с линейным устройством, т.е. открыть порт потокового ввода-вывода и принимать оттуда данные. Проблема оказалась в следующем: данные передавались в числовом виде, поэтому нельзя было наверняка определить, что мы передали с Android-устройства: числовую кодировку буквы из первых двух режимов или угол поворота из третьего режима управления. Мало того, вскоре было обнаружено, что символьные данные (текст) передаются, не соответствуя ни одной кодировке (так происходило из-за различия используемых кодировок в Android и Arduino).

Проблема с разными кодировками решилась не быстро, но довольно просто: для того, чтобы получать код символа, необходимо было прибавлять к полученным числам число 176 (определено экспериментально), в этом случае мы получим код символа по CP1251-таблице.

Намного сложнее оказалось определять, что мы передаем: код символа или цифровое значение. Для решения этой проблемы использовались флаги данных. При старте активности на Android-устройстве после получения связи с Bluetooth-адаптером открывался потоковый порт вывода, куда записывалось 1 или 0, в соответствии с выбранным режимом управления (для текстовых режимов – 1, для ручного – 0). Со стороны микроконтроллера открывался потоковый порт ввода-вывода, и полученный флаг записывался в соответствующую переменную, после чего порт становился недействительным до тех пор, пока пользователь не нажмет кнопку «Перевести текст» или «Отправить данные». Получая данные, микроконтроллер уже будет понимать, что это за данные, ориентируясь на флаг.

Логика языка Arduino IDE разработана таким образом, что в нем присутствуют два типа работы программы: setup и loop. Команды, записанные в setup, выполняются только один раз при включении контроллера, команды, записанные в loop, повторяются циклически, пока контроллер работает. В связи с этим нужно было приостановить чтение данных из потокового порта ввода-вывода после отправки флага, но перед отправкой самих данных. В противном случае программа получила бы флаг, а после начала бы чтение данных оттуда, где их еще нет (пользователь не отправил данные, следовательно, потоковый порт пуст), из-за этого программа работала бы некорректно.

Еще одной проблемой стала перезапись флага. Проблема возникла также из-за логики языка, о которой говорилось выше. При повторной отправке текстовых или числовых данных программа из-за цикличности блока loop, в который помещены выполняемые функции, заново начинала считывать флаг и принимать за флаг первый бит (первый символ или первое число) полученных данных. В результате работы такой программы при повторной отправке слова «Привет» с Android-устройства на микроконтроллере мы получим слово (если это можно так назвать) «ривет», а флаг будет равен 63, что соответствует числовой кодировке буквы «п» по таблице CP1251 (сама кодировка буквы «п» по данной таблице – 239), из которой отняли 176 (об этом феномене говорилось выше), т.е. программа получала непонятное слово, а вдобавок не знала слово это или число: флаг не соответствовал ни 0, ни 1. К счастью, проблема решилась довольно просто: перед записью флага необходимо было проверять, является ли то число, которое мы хотим записать в флаг 1 или 0 (букв с такими кодировками не существует, а цифры было принято отправлять, начиная с 5 для того, чтобы не спутать с флагом).

Еще одним технологическим нюансом явилось то, что язык жестов не подразделяется на заглавные и строчные буквы, но, несмотря на это, код одной и той же заглавной и строчной буквы будет отличаться[[8]](#footnote-8). Решением выступило то, что в силу особенностей кодовой таблицы CP1251 (символы кириллицы в ней расположены по порядку, что присутствует далеко не во всех кодовых таблицах), код строчной и заглавной буквы отличается на постоянную величину, равную 32. Таким образом, если мы точно знаем, что получили символьную строку (вывод об этом можно сделать, опираясь на значение флага), и если код символа лежит в диапазоне от 192 включительно до 224 не включительно (рис. 5), то можем быть уверены, что данный символ – заглавная буква, и при прибавлении к коду данного символа 32 мы получим ту же букву, но строчную.

Последней, к сожалению, пока не до конца решенной проблемой оказалось то, что символ пробела передавался как 32 (это действительно код символа «пробел», без смещения на 176). Код, равный 32 для пробела – это стандарт UNICODE, одинаковый для всех устройств (кириллица в этот стандарт не входит, откуда и взялась проблема с разными кодировками), но нетрудно заметить, что при прибавлении к 32 – коду пробела – 176, получится 208, что соответствует символу «Р» по таблице CP1251. К сожалению, принимать еще один флаг каждый раз при использовании пробела окажется слишком ресурсозатратно для контроллера с небольшой (относительно смартфонов и компьютеров, для которых подобна операция не составила бы труда) вычислительной мощностью. Поэтому было решено заменять код 32 на символ пробела, поскольку он используется чаще и от его использования отказаться труднее, чем от использования заглавной буквы «Р». Таким образом рабочая программа при отправке с Android-смартфона строки «Река» выведет слово « ека». Перепробовано несколько способов решения этой проблемы, но каждый из них вредил работе программы больше, чем данная ошибка. В конечном итоге принято решение оставить ее и уведомлять пользователя с Android-устройства о том, что использование заглавных букв может повлечь ошибки в работе программы.

# Заключение

В процессе работы была с нуля создана модель манипулятора-сурдопереводчика, реализовано его программное обеспечение, обеспечено удобное управление (написана библиотека для работы с манипулятором). Решены все поставленные задачи, достигнута цель работы. Несмотря на нерешенные еще недочеты, прототип является полностью рабочим и успешно функционирует: выполняет переданные команды, переводит текст, распознает речь. Кроме того, было написано Android-приложение, которое также является готовым продуктом и позволяет реализовывать функции управления моделью.

В будущем планируется продолжать работу над проектом, в том числе улучшить и модернизировать ПО переводчика, например, добавить возможность создавать свои жесты и использовать их в будущем. Также планируется улучшить работу Android-приложения, добавив новые функции перевода на язык жестов. В планах присутствует возможность реализации обратного перевода: например, некая перчатка, надевая которую, человек мог бы перевести жесты, которые он показывает, в текст или голосовое сообщение, отображающееся на экране его смартфона.

По окончании работы написана статья (с приложением программного кода, инструкций по сборке, пояснений к ним) на ресурсах GitHub и HabraHabr. Разместив информацию в open-source источниках, мы стремились максимально распространить проект.

# Список литературы

1. Arduino Russia, [Электронный ресурс]//   
   <http://arduino.ru>
2. Arduino, [Электронный ресурс]//  
   <https://www.arduino.cc>
3. Cxem.net, [Электронный ресурс]//  
   <http://cxem.net/master/45.php>
4. Google академия, [Электронный ресурс]//  
    <https://scholar.google.ru>.
5. Mypractic, [Электронный ресурс]//   
   <http://mypractic.ru/urok-23-podklyuchenie-zhk-lcd-indikatorov-k-arduino-biblioteka-liquidcrystal.html>
6. Амперка, [Электронный ресурс]//  
    <http://amperka.ru>
7. Аппаратная платформа Arduino, [Электронный ресурс]//  
   <http://arduino.ru>
8. База данных рефератов и цитирования «Scopus», [Электронный ресурс]//  
   <https://www.scopus.com>.
9. Блум Джереми. Изучаем Arduino: инструменты и методы технического волшебства: Пер. с англ. — СПб.: БХВ-Петербург, 2015. — 336 с.: ил.
10. Фрадкина Р.Н. Говорящие руки: М.: «Рефл-бук», 2001.– 402 стр.

# Приложение



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 – один из виджетов, применяемых в разработке Android-приложений. Представляет собой шкалу с выбором значения. [↑](#footnote-ref-3)
4. Среда разработки приложений для устройств на базе операционной системы Android, в которой создавался проект. [↑](#footnote-ref-4)
5. Решение поместить установку Bluetooth-соединения в отдельный поток было принято в связи с тем, что данная операция довольно ресурсозатратна (требует много оперативной памяти устройства), и при выполнении в главном потоке приложение будет «подвисать» (переставать реагировать на действия пользователя) на время выполнения операции подключения Bluetooth-соединения. [↑](#footnote-ref-5)
6. Tóast (англ.) – тост. Профессиональное название типа уведомлений, в котором уведомление, подобно хлебному тосту из тостера, выскакивает снизу и пропадает через несколько секунд. [↑](#footnote-ref-6)
7. Была выбрана модель HC-06, т.к. это одна из новейших моделей Bluetooth-модулей, поддерживающая спецификацию Bluetooth версии 2.1+EDR, которая гарантирует пониженное потребление энергии, повышенный уровень защиты данных и легкое соединение Bluetooth-устройств, кроме того, устойчивый прием с модулем гарантирован в пределах 10 метров. [↑](#footnote-ref-7)
8. Оправдание того, что данное «превращение» необходимо для корректной работы программы можно найти, разобравшись в принципе работы функций перевода символов на язык жестов, записанных в созданной нами библиотеке. [↑](#footnote-ref-8)