Создание модели манипулятора-сурдопереводчика

Работу выполнили

Калинин Даниил Евгеньевич

Радькин Кирилл Алексеевич

Россия, Челябинская область

Город Челябинск

Ученики 10м1 класса

МАОУ «Лицея №97 г. Челябинска»

Научный руководитель проекта

Саканов Дамир Муратович

педагог дополнительного образования

МАОУ «Лицей №97 г. Челябинска»

Работа к защите допущена «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Саканов Д.М

Оглавление

[Введение 4](#_Toc510562186)

[Глава 1. Изучение литературы. 5](#_Toc510562187)

[1.1. Изучение аналогичных научных статей. 5](#_Toc510562188)

[1.2. Изучение языка жестов. 5](#_Toc510562189)

[1.3. Изучение микроконтроллера Arduino. 6](#_Toc510562190)

[1.4. Выбор подходящего микроконтроллера линейки Arduino. 6](#_Toc510562191)

[1.5. Изучение существующих схем манипуляторов. 6](#_Toc510562192)

[1.6. Выбор чертежей для сборки манипулятора. 7](#_Toc510562193)

[Глава 2. Сборка модели. 8](#_Toc510562194)

[2.1. Создание деталей. 8](#_Toc510562195)

[2.1.1 3D-модель. 8](#_Toc510562196)

[2.1.2 Ручная сборка. 8](#_Toc510562197)

[2.2. Процесс сборки. 10](#_Toc510562198)

[2.2.1 Сборка пальцев. 10](#_Toc510562199)

[2.2.2 Сборка запястья и предплечья. 10](#_Toc510562200)

[2.2.3 Крепление сервоприводов. 11](#_Toc510562201)

[2.3. Подключение аппаратуры. 11](#_Toc510562202)

[2.4 «Стрела» 11](#_Toc510562203)

[Глава 3. Программная часть. 13](#_Toc510562204)

[3.1. Программирование аппаратной части. 13](#_Toc510562205)

[3.1.1. Выбор языка и среды программирования. 13](#_Toc510562206)

[3.2. Написание библиотеки. 13](#_Toc510562207)

[3.2.1. Навигационная матрица. 14](#_Toc510562208)

[3.2.2. Функция перевода символов. 14](#_Toc510562209)

[3.2.3. Функция перевода предложений. 14](#_Toc510562210)

[3.2.4. Функция чтения флага. 15](#_Toc510562211)

[3.3. Создание Android-приложения. 15](#_Toc510562212)

[3.3.1 Управление посредством ввода текста. 16](#_Toc510562213)

[3.3.2. Управление посредством распознавания речи. 17](#_Toc510562214)

[3.3.3. Ручной режим управления. 18](#_Toc510562215)

[3.3.4. Локализация приложения. 19](#_Toc510562216)

[3.4. Реализация передачи данных. 21](#_Toc510562217)

[3.4.1. Реализация со стороны Android-приложения. 21](#_Toc510562218)

[3.4.2. Реализация со стороны микроконтроллера. 23](#_Toc510562219)

[Заключение. 27](#_Toc510562220)

[Список литературы. 28](#_Toc510562221)

# Аннотация

Целью работы является создание механической роботизированной модели человеческой руки, выполняющей функцию перевода естественного языка (мы остановились только на русском языке) на невербальный язык – язык жестов, а также написание мобильного приложения, позволяющего управлять моделью и сопутствующего программного обеспечения на самой модели. Работа представляет технический интерес, поскольку, как рассмотрено ниже, в первой главе, нами не было найдено русскоязычных аналогов данного проекта. В результате выполнения работы мы достигли поставленной цели, собрав и настроив рабочий прототип механической модели человеческой руки. Прототип полностью исправен и работает строго по заданному алгоритму, но, тем не менее, в ходе выполнения работы мы столкнулись с некоторыми проблемами, как по аппаратной, так и по программной части, которые, не были решены нами в силу сложности технической реализации проекта, об этих проблемах подробно рассказано в третьей главе. В ходе работы мы прибегали к анализу данных о количестве людей с ограниченными возможностями, анализу и подробному изучению различных невербальных способов общения, анализу документации электроники, а именно микроконтроллеров и сервоприводов, для выбора оптимального по производительности, кроме того, нами были изучены и проанализированы различные чертежи из свободных источников, о чем сказано в первой главе. Теоретическая информация, необходимая для написания части программного кода, а также некоторые чертежи для печати макета выбирались из свободных источников, а именно из интернета, подробнее информационные ресурсы рассмотрены в первой главе. Список используемой литературы имеется в конце работы.

# Введение

Люди могут все. Так говорят, подразумевая, что для человека, который действительно чего-то хочет, нет ничего невозможного. Но, к сожалению, в современном обществе существует такая социальная группа, как люди с ограниченными возможностями. Мы считаем, что «ограниченные возможности» должны быть просто формальным названием, не имеющим ничего общего с действительностью. И наша цель – сделать все, чтобы это было так.

Актуальность проблемы заключается в том, что подобных проектов до сих пор не существует в России, в отличие от людей с ограниченными возможностями слуха. Поэтому мы считаем, что наша главная цель – разработать первый в России автоматический сурдопереводчик для снятия границ общения между всеми людьми.

Для реализации нашей цели мы поставили себе ряд задач:

* Разработать проект программируемого сурдопереводчика.
* Написать программное обеспечение для установки.
* «Научить» модель автоматически распознавать речь и переводить ее на язык жестов.

# Глава 1. Изучение литературы.

## 1.1. Изучение аналогичных научных статей.

Перед тем, как начать разрабатывать проект было принято решение изучить сопутствующую литературу. Был проведен поиск и анализ статей на таких базах, как Scopus и Google Scholar. Не было найдено ни одной научной статьи российских ученых, описывающей проект, аналогичный нашему. Поэтому мы ограничились изучением работ зарубежных авторов, которых оказалось не так много, как мы ожидали.

## 1.2. Изучение языка жестов.

После изучения научных работ в открытой базе, мы приступили к изучению так называемого языка жестов. Оказалось, что существует много различных вариаций этого языка – жесты, обозначающие не отдельные буквы, а целые слова и фразы. Для изображения таких жестов требуется голова, тело, плечи, и обе руки. Но, поскольку каждый такой жест можно представить, как несколько простых жестов – каждый из которых обозначает одну букву–, было принято решение не учитывать такие жесты при создании и обучении модели. На рисунке 1 представлен «Жестовый алфавит» – русский алфавит, переведенный на язык жестов.



Рис. 1. Язык жестов.

## 1.3. Изучение микроконтроллера Arduino.

Изучив язык жестов, мы приступили к изучению литературы, связанной с исполнительным устройством нашей модели – микроконтроллером Arduino и средой разработки – Arduino IDE. Микроконтроллер этого производителя был выбран за то, что отвечал всем требованиям нашего проекта, а именно:

* Имел высокую производительность
* Имел достаточное количество портов для подключения к нему различных сторонних модулей
* Был прост в освоении

Кроме того, среда разработки Arduino IDE основана на языке С++, с которым мы знакомы из школьной программы. Но, несмотря на схожую логику и синтаксис, Arduino IDE и С++ – два совершенно разных языка. Пришлось изучить немало литературы для того, чтобы освоить язык Arduino.

## 1.4. Выбор подходящего микроконтроллера линейки Arduino.

В линейке Arduino существует много различных микроконтроллеров. Для проекта были выбраны Arduino Mega 2560 (сочетает в себе высокую производительность и дешевизну стоимости), Arduino Due (самый мощный микроконтроллер во всей линейке Arduino; как заявляет производитель, этот микроконтроллер способен управлять настоящим заводским станком небольших размеров) и Arduino Uno (наиболее популярный и недорогой контроллер в линейке). Так как нам необходима максимальная производительность, мы выбрали Arduino Due.

## 1.5. Изучение существующих схем манипуляторов.

Для того, чтобы избежать ошибок наших иностранных коллег, мы изучили уже существующие схемы сборки манипулятора. Большая их часть представляла собой файлы 3D-моделей для печати на 3D-принтере. После изучения прототипов этих моделей мы пришли к выводу, что несмотря на невероятную простоту изготовления (все детали проектируются на компьютере, после чего необходимо распечатать их и собрать, как обычный конструктор), они не обладают достаточным соотношением прочности к легкости: ABS пластик, из которого печатаются такие модели достаточно хрупок, и для достижения необходимой прочности необходимо печатать довольно толстым слоем пластика (либо использовать более дорогостоящую технологию порошковой печати), в итоге прототип получится слишком тяжелым.

Было изучено 7 различных прототипов манипуляторов, печатаемых на 3D-принтере. Основным минусом среди всех прототипов была изложенная выше проблема. Тем не менее, встретился и более удачный проект, расположенный в французском открытом банке 3D проектов «InMoov», который мы решили использовать.

## 1.6. Выбор чертежей для сборки манипулятора.

В процессе написания работы встал вопрос о выборе чертежей, по которым мы будем собирать манипулятор. Выбор стоял между чертежами, созданными нами (не предполагающими использование 3D-технологий) и чертежами, взятыми со свободных библиотек чертежей в интернете.  
Разумеется, параллельно с поиском соответствующих условиям, изложенным выше, чертежей в интернете, мы подготовили несколько вариантов своих чертежей для разработки модели манипулятора, как уже говорилось раньше, без 3D технологий. Но, как оказалось при сравнительном анализе наших чертежей и чертежей из открытых баз в интернете, использование 3D-принтера при создании прототипа, во-первых, намного упрощает задачу, а, во-вторых, является наиболее точным методом, что также немало важно в нашем проекте. Тем более, при создании нескольких частей-прототипов по нашим чертежам оказалось, что пенополистирол, который мы предполагали использовать за его легкость и прочность, удовлетворяет не всем требованиям нашей конструкции, а именно: довольно тяжело обрабатывается (имеется ввиду качественная обработка) без специальных станков, а также, режется леской, которая, как предполагала конструкция, должна осуществлять сгиб пальцев руки-манипулятора. В связи с вышеизложенными фактами, было решено взять готовые чертежи. В случае нахождения недостатков конструкции при сборке, они, во-первых, были бы несущественны, так как был проведен детальный анализ конструкции, во-вторых, они были бы легко исправимы благодаря тому, что файл 3D-макета всегда можно отредактировать. Для того, чтобы «не изобретать велосипед» в нашем проекте мы решили использовать макет манипулятора, о котором говорилось в предыдущем параграфе.

# Глава 2. Сборка модели.

## 2.1. Создание деталей.

### 2.1.1 3D-модель.

Одним из важнейших этапов сборки был выбор материала и поиск нужной нам модели. Как было сказано выше, модель была взята в открытом банке 3D проектов – “InMoov”.

Также, нам было необходимо было найти 3D-принтер, на котором были бы распечатаны детали. В этом нам помог наш научный руководитель. В ведении образовательной организации, в которой преподает Дамир Муратович, находится 3D принтер, печатающий по технологии «Послойное наплавление (FDM)». Данная технология очень распространена из-за своей доступности и дешевизны. Именно ее мы решили использовать для печати модели.

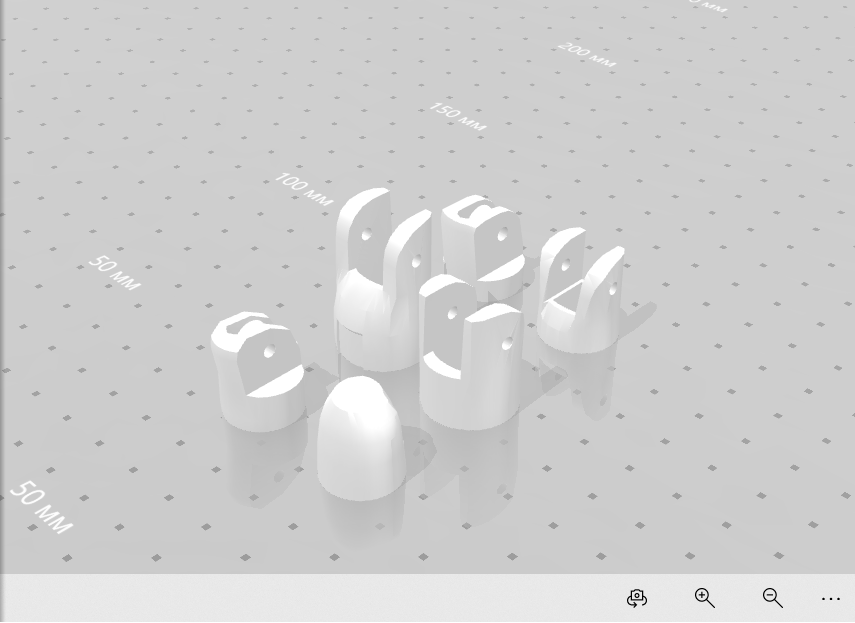


Рис 2. Одна из моделей для печати.

На рисунке 2 приведен пример одного из чертежей, по которым должна была быть проведена печать.

### 2.1.2 Ручная сборка.

За несколько недель до сдачи работы выяснилось, что принтер, на котором должны были быть распечатаны детали, серьезно сломался и на его починку требуется около месяца. Чтобы закончить проект, было принято решение самостоятельно собрать манипулятор из подручных материалов. Разумеется, в будущем проект будет продолжен, детали распечатаны, а манипулятор собран из них.  
 В качестве материала для пальцев было решено использовать тонкую фанеру. Небольшие резинки выступали в роли возвращающей силы, которая стремится удержать палец в исходном положении.

На рисунке 3 (см. ниже) приведена условная схема сборки одного пальца в двух проекциях. Слева изображено горизонтальное сечение пальца, справа, соответственно, его вертикальное сечение. Красным цветом обозначены подвижные стержни, которые являются осями для вращения двух соседних фаланг. Зеленым цветом выделены резинки, возвращающие палец в исходное положение после окончания действия.  
 Фаланги пальцев размечались на фанере, соответственно, 3 варианта фаланг.   
 Первый тип – самая первая и самая короткая фаланга. В ней проделывалось два отверстия: одно для закрепления нити, она продевалась через него и закреплялась снаружи, второе отверстие для подвижных стержней (см. выше), в роли подвижных стержней выступили болты. В связи с этим, второе отверстие делалось чуть шире основания болта (для обеспечения свободного равномерного движения). Также, на первом типе фаланги закреплялся тонкий стержень с кольцом на конце: с помощью этого кольца закреплялась резинка, стягивающая фаланги друг к другу.

Второй тип – фаланга среднего размера. В ней также проделывалось два отверстия, но оба из них предназначались для осей вращения. Кроме того, между двумя составляющими фаланги закреплялся тонкий стержень, под которым проходила нить. Использование такого метода проложить нить вдоль пальца, не вынося ее далеко за пределы руки.

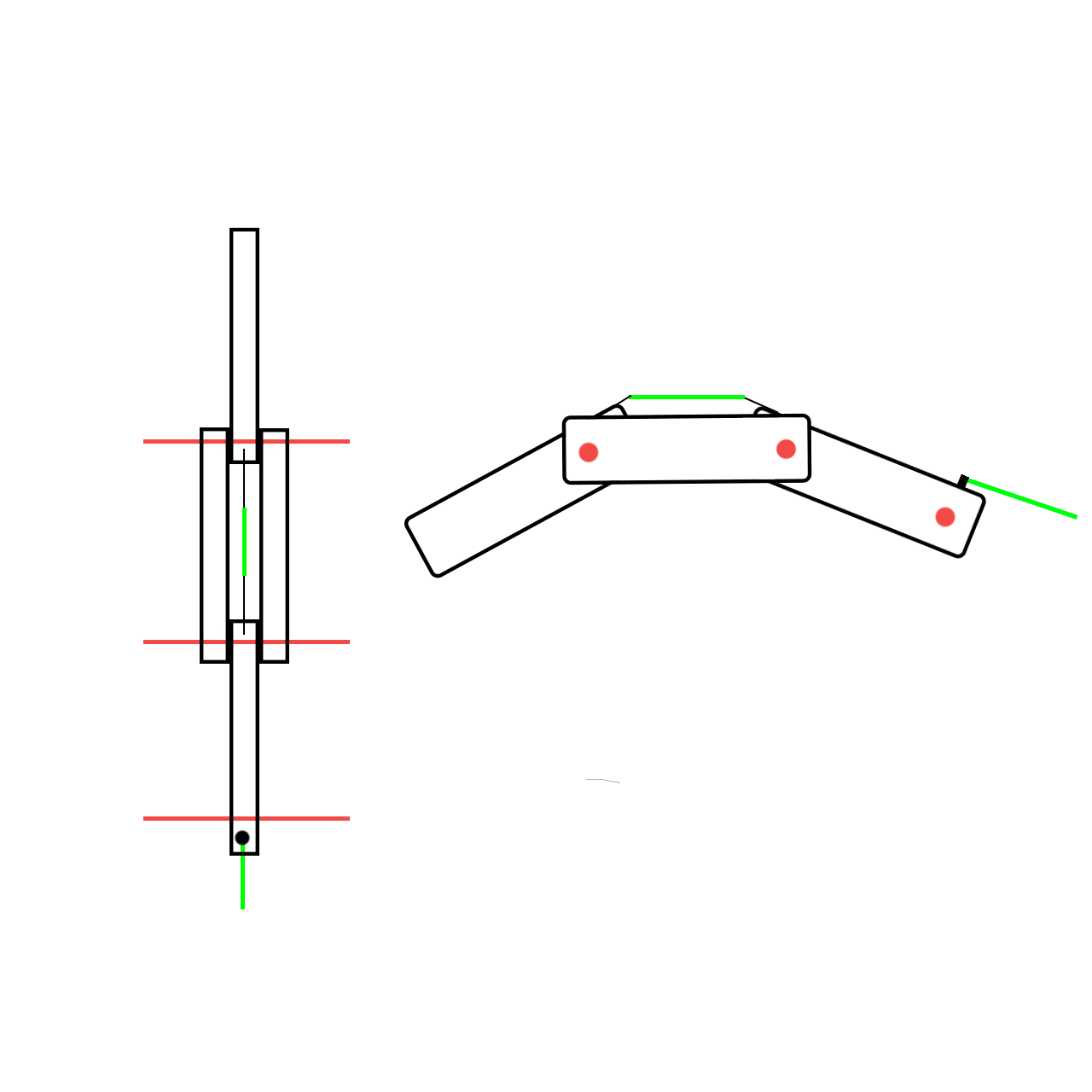


Рис. 3. Схема сборки пальца манипулятора (слева – в горизонтальном сечении; справа – в вертикальном сечении).

Третий тип – самая длинная фаланга. Аналогично со вторым типом, два отверстия для осей. Одна из осей позволяет закрепить весь палец к руке и обеспечить его наклон. Также, к фаланге крепилась резинка, второй конец которой находился на специально вынесенной для этого (за пределы запястья) части руки.

## 2.2. Процесс сборки.

### 2.2.1 Сборка пальцев.

Самым сложным этапом в процессе сборке была сборка пальцев, особенно первого, т.к. именно на первом, часто пробном объекте совершаются все возможные ошибки.

Для каждого пальца было необходимо вырезать по две фаланги каждого типа. Они размечались на фанере, после вырезки и обработки детали каждого типа складывались попарно, и в каждой паре просверливались отверстия, необходимые им. Попарная сортировка и сверление были необходимы для обеспечения совпадения отверстий.  
 Далее, фаланги 1-го и 3-го типов склеивались между собой и соединялись с фалангами 2-го типа на осях вращения. В первую фалангу продевалась нить, закреплялась на конце, продевалась под тонким стержнем, вклеенным между частями второй фаланги, продевалась через небольшое кольцо, закрепленное на третьей фаланге, и привязывалась вторым концом к сервоприводу.  
 Между 1-ой и 3-ей фалангами закреплялась резинка. Такая же резинка крепилась с другого конца 3-ей фаланги, притягивая его к внешней стороне запястья.

### 2.2.2 Сборка запястья и предплечья.

Запястье и предплечье собирались из металлических деталей (пластин различной длины, детали в виде «углов»). В таких деталях проделано несколько отверстий, позволяющих легко фиксировать детали друг на друге с помощью болтов и гаек.

Каждый палец крепился к запястью с помощью «детали-уголка». Также, через такие детали продевались нити, идущие к сервоприводам. Это сделано для того чтобы направить все нити параллельно друг другу, вдоль предплечья.

### 2.2.3 Крепление сервоприводов.

Сервоприводы закреплялись на дальнем конце предплечья. То, что ранее нити были перенаправлены параллельно друг другу, значительно облегчало крепление сервоприводов на руку. Также, сложность заключалась в том, что при вращении своей верхней частью, сервоприводы могут помешать друг другу, и, возможно, повредить манипулятор, необходимо было исключить эту возможность.

## 2.3. Подключение аппаратуры.

Для работы сурдопереводчика, было необходимо подключить Arduino, Bluetooth-модуль, сервоприводы и LCD-монитор (на него выводятся данные о текущих углах наклона сервоприводов и какой символ показывает манипулятор в данный момент).

С подключением Arduino не возникло никаких проблем, контроллеру не были обходимы какие-либо особенные условия. Запитать его можно от обычного powerbank-а[[1]](#footnote-1), который используются для зарядки мобильных устройств.

Аналогичная ситуация с Bluetooth-модулем и LCD-монитором. Им обоим хватает питания 5V, которое предоставляет им контроллер.

Гораздо сложнее было подключить сервоприводы. Для подключения одного сервопривода достаточно питание 5V от Arduino, но для 5-ти серво его уже недостаточно, т.к. для быстрой и четкой работы сервоприводу необходим определенный ток и напряжение. Но, т.к. у нас 5 серво, при подключении к Arduino, либо ток, либо напряжение на каждом серво окажутся недостаточными для нормальной работы. Чтобы решить эту проблему, было необходимо установить источник питания (или его стабилизатор), выдающий необходимое напряжение, которое при подаче на серво давало нужный ток, не слишком большой, иначе серво может сгореть.

### 2.4 «Стрела»

«Стрела» - продукт компании «Амперка», Arduino-совместимая платформа, выпускаемая как материнская плата для постройки роботов. «Стрела» заменяет собой Arduino с несколькими «шилдами». Из них был использован только один «шилд» - для сервомоторов. Фактически, «Стрела» была использована в качестве источника питания(см.выше). Потребляя обычное напряжение(характерное для Arduino), она могла обеспечить питанием 5 сервоприводов. Именно поэтому «Стрела» используется как серво-«шилд» для Arduino.

# Глава 3. Программная часть.

## 3.1. Программирование аппаратной части.

### 3.1.1. Выбор языка и среды программирования.

Для программирования аппаратной части был выбран язык программирования C++. Дело в том, что компилятор, преобразующий программный код в бинарный – код, который может воспринять процессор микроконтроллера – адаптирован для языка C++ (язык программирования Arduino IDE основан на C++). Кроме того, данный язык изучается в рамках школьной программы, что упрощает его использование.

В качестве среды для разработки была выбрана среда, предложенная производителем микроконтроллеров линейки Arduino – 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 о написании библиотеки функций для управления манипулятором. Данное решение было принято в связи с тем, что, 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[[2]](#footnote-2).

В последующих пунктах будут рассмотрена работа функций 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»). Кроме того, стоит отметить, что 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 не случайно: 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.

### 3.2.4. Функция чтения флага.

Подробнее о данной функции будет сказано в пункте о реализации Bluetooth-общения между Android-устройством и микроконтроллером линейки Arduino. Тем не менее, о некоторых аспектах расскажем сейчас.  
Функция при помощи специального алгоритма читает флаг, необходимый для правильной работы с поступающими далее данными, и возвращает true, если флаг – 0, и false,если флаг ­– 1. Кроме того, функция предусматривает еще несколько технологических аспектов для правильного чтения флага (например, проверяет, действительно ли значение флага соответствует одному из его возможных значений: не была ли часть самих данных по ошибке записана во флаг). Таким образом, данная функция упрощает обработку значения флага: в условный оператор можно не прописывать множество условий, а просто подставить возвращенное значение данной функции.

## 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).



Рис 4. Главный экран приложения.

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 (рис. 4).

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.

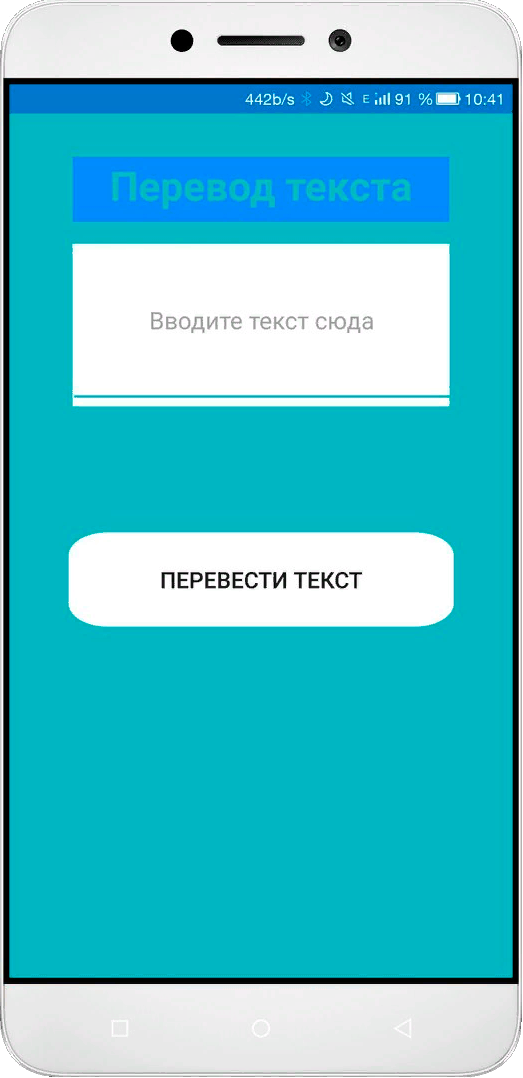


Рис. 5. Окно работы в режиме перевода текста.

### 

### 3.3.2. Управление посредством распознавания речи.

При нажатии на кнопку «Управление голосом» на стартовом экране откроется новое окно (рис. 5).

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», после чего, как уже было описано выше установится Bluetooth-соединение, и данные будут переданы.

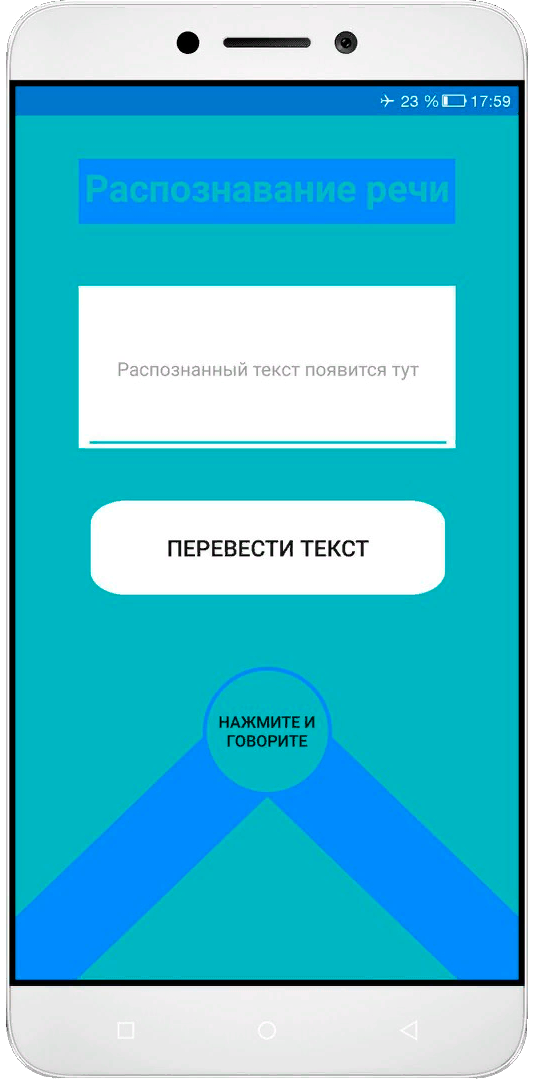


Рис 6. Окно работы в режиме распознавания речи.

### 

### 3.3.3. Ручной режим управления.

После того, как пользователь нажмет кнопку «Ручной режим» откроется окно (рис. 6) 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. Таким образом, с помощью данных шкал пользователь может указать необходимое ему значение для каждого пальца отдельно. После чего он сможет нажать кнопку «Передать данных», по нажатию которой данные будут переданы на микроконтроллер по описанному ниже алгоритму.

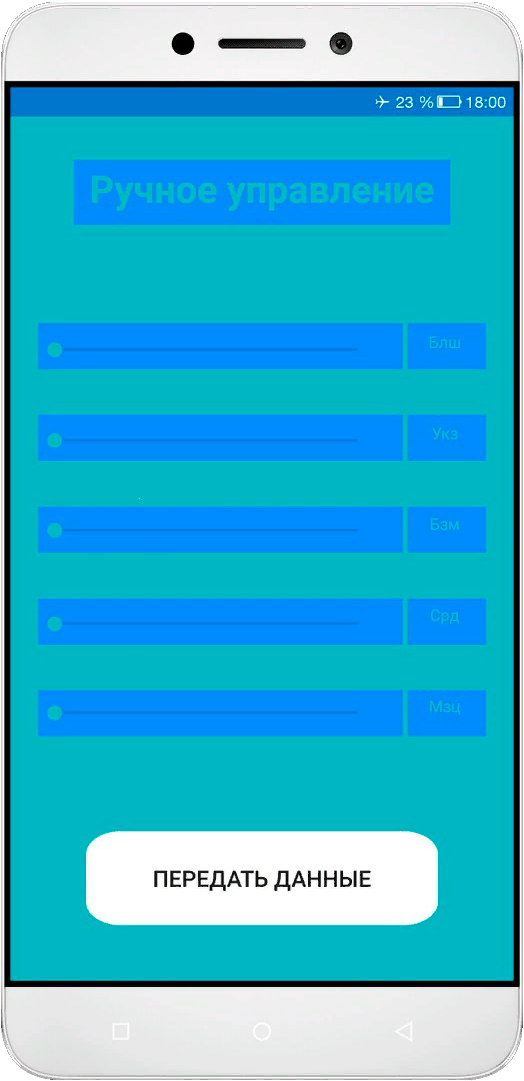


Рис 7. Окно работы с Seek bar.

### 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: приложение полностью на русском языке будет им непонятно. В связи со сказанным выше, 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 (рис.7).



Рис.8. Окно редактора локализаций Android Studio.

Таким образом, если в настройках устройства сменить системный язык на английский, приложение будет полностью на английском (рис. 8).

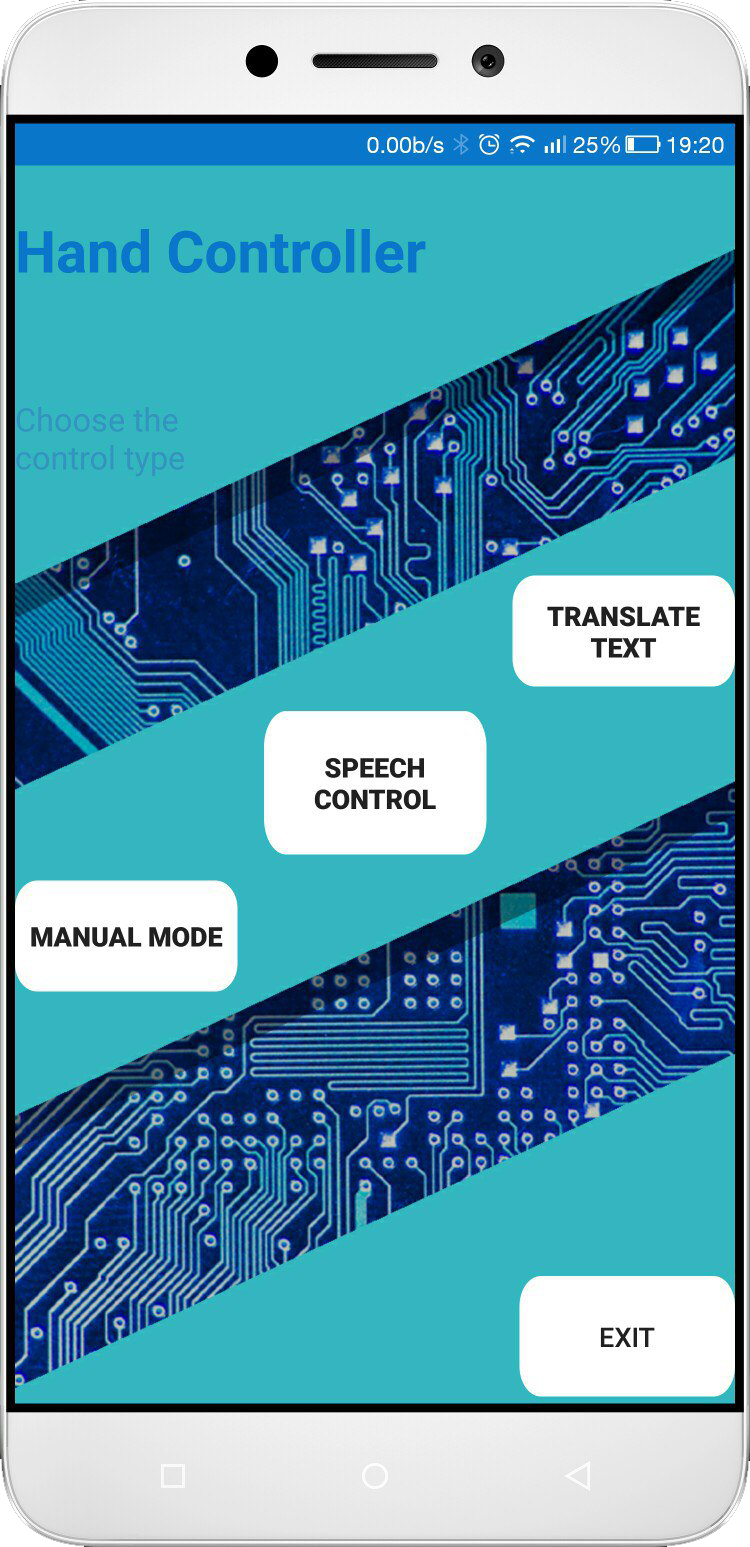


Рис. 9. Локализованный вариант приложения.

## 3.4. Реализация передачи данных.

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

После написания базовой части программы, в состав которой входило: 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. Пришлось изучать множество учебной литературы по теме, поскольку в рамках IT-школы Samsung данная тема не была пройдена на момент реализации проекта. 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. 9).



Рис. 10. Окно включения Bluetooth из приложения.

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 (рис. 10).

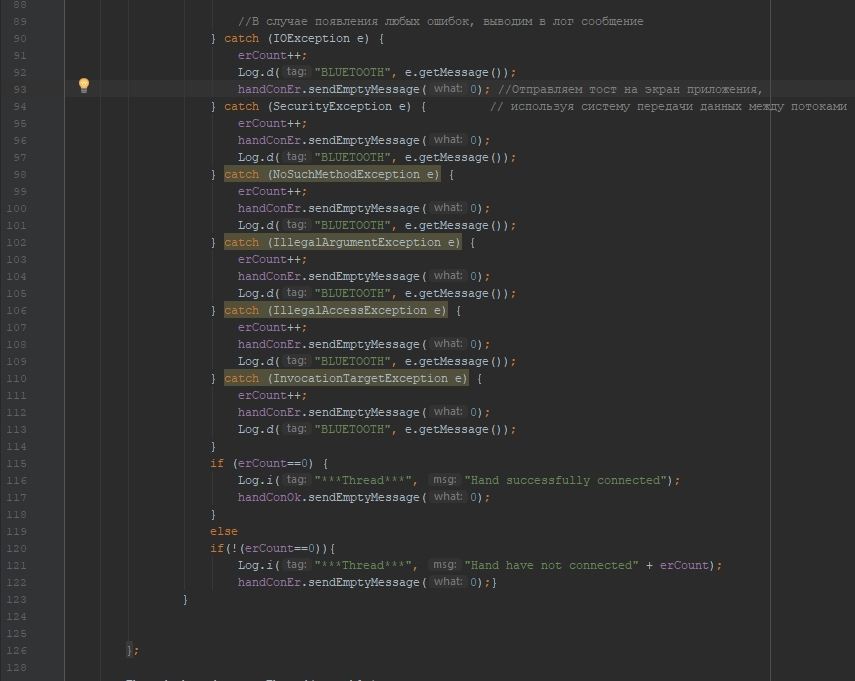


Рис. 11. Реализация обработки ошибок приложения.

### 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 (была выбрана модель HC-06 т.к. это одна из новейших модель Bluetooth-модулей, поддерживающая спецификацию Bluetooth версии 2.1+EDR, которое гарантирует пониженное потребление энергии, повышенный уровнем защиты данных и лёгкое соединение Bluetooth-устройств, кроме того устойчивый приём с модулем гарантирован в пределах 10 метров), 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 setupвыполняются только один раз при включении контроллера, команды, записанные в loop повторяются циклически, пока контроллер работает. В связи с этим явилось необходимостью организовать приостановление чтения данных из потокового порта ввода-вывода в промежуток времени 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 выполняемы 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).

Очередной проблемой, а, вернее, 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[[7]](#footnote-7). 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, включая до 224, не включая (рис. 11), то мы можем быть уверенны, что данный символ – заглавная буква, и при прибавлении к коду данного символа 32, мы получим ту же букву, но строчную.

Последней, к сожалению, не до конца решенной проблемой оказалось то, что символ пробела передавался как 32 (это действительно код символа «пробел», без смещения на 176), происходило так из-за того, что 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». Было перепробовано несколько способов решения этой проблемы, но каждый из них вредил работе программы намного больше, чем данная ошибка. В конечном итоге, было принято решение оставить эту ошибку, и уведомлять пользователя с Android-устройства о том, что использование заглавных букв может повлечь ошибки в работе программы.



Рис. 12. Кодовая таблица CP1251.

# Заключение.

Подведем итоги. В процессе работы была с нуля создана модель манипулятора-сурдопереводчика, реализовано его программное обеспечение, обеспечено удобное управление (написана библиотека для работы с манипулятором). То есть, были выполнены все поставленные задачи, достигнута цель работы. Хотелось бы отметить, что, несмотря на некоторые недостатки в работе программной части, а также недостатки в конструкции модели, прототип является полностью рабочим и нормально 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.

Мы считаем, что проект должен получить рецензию от человека с ограниченными возможностями слуха. Это необходимо не только для нахождения новых ошибок и неудобств, их исправления, но и для повышения статуса проекта.

В будущем планируется продолжать работу над проектом, в том числе распечатать и собрать версию манипулятора на 3D-принтере, возможно улучшить и модернизировать ПО переводчика, например, добавить возможность создавать свои жесты и использовать их в будущем. Также возможно улучшение работы Android-приложения, добавления новых функций перевода на язык жестов. Кроме того, 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 планируется написать статью (с приложением программного кода, инструкций по сборке, пояснений к ним и т.д.) на таких ресурсах, как GitHub и HabraHabr. То есть, максимально распространить проект, сделав доступ к нему открытым, добавив в open-source источники.

# 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. Шестернинов Е.Е., Арцев М.Н. Спутник Исследователя. –М.: Некоммерческая организация Благотворительный фонд наследия Менделеева, 2017. –53 с.:ил.
11. Гейльман И. Ф. Знакомьтесь: ручная речь: М.: Загрей, 2001.   
    – 172 с.
12. Фрадкина Р.Н. Говорящие руки: М.: «Рефл-бук», 2001г.  
    – 402 стр.

1. Powerbank (англ.) – портативный аккумулятор. Устройство, используемое для зарядки смартфонов, планшетов и другой портативной техники. [↑](#footnote-ref-1)
2. Здесь следует отметить, что наш проект является уникальным для России. В связи с этим, по окончании работы, будет поставлена цель максимально распространить проект, добавив его в open-source источники. [↑](#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. Здесь и далее в работе будет использована профессиональная лексика с целью обеспечения корректности сказанного. [↑](#footnote-ref-6)
7. 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-7)