Создание модели

манипулятора-сурдопереводчика

Калинин Даниил Евгеньевич, Радькин Кирилл Алексеевич

МАОУ «Лицей №97 г. Челябинска», 10м1 класс

Научный руководитель:

Саканов Дамир Муратович

педагог дополнительного образования

МАОУ «Лицей №97 г. Челябинска»

г. Челябинск

Оглавление

[Введение 4](#_Toc535950867)

[Глава 1. Изучение литературы 4](#_Toc535950868)

[1.1. Изучение аналогичных научных статей 4](#_Toc535950869)

[1.2. Изучение языка жестов 4](#_Toc535950870)

[1.3. Изучение микроконтроллера Arduino 4](#_Toc535950871)

[1.4. Разработка чертежа манипулятора 5](#_Toc535950872)

[Глава 2. Сборка модели 5](#_Toc535950873)

[2.1. Создание деталей 5](#_Toc535950874)

[2.1.1 3D-модель 6](#_Toc535950875)

[2.2. Процесс сборки 6](#_Toc535950876)

[2.2.1 Сборка пальцев 6](#_Toc535950877)

[2.2.2 Сборка предплечья и запястья 6](#_Toc535950878)

[2.3. Подключение аппаратуры 6](#_Toc535950879)

[2.4 «Стрела». 7](#_Toc535950880)

[Глава 3. Программная часть 7](#_Toc535950881)

[3.1. Программирование аппаратной части 7](#_Toc535950882)

[3.1.1. Выбор языка и среды программирования 7](#_Toc535950883)

[3.2. Написание библиотеки. 7](#_Toc535950884)

[3.2.1. Навигационная матрица 8](#_Toc535950885)

[3.2.2. Функция перевода символов 8](#_Toc535950886)

[3.2.3. Функция перевода предложений 8](#_Toc535950887)

[3.3. Создание Android-приложения 9](#_Toc535950888)

[3.3.1 Управление посредством ввода текста 9](#_Toc535950889)

[3.3.2. Управление посредством распознавания речи 9](#_Toc535950890)

[3.3.3. Ручной режим управления 10](#_Toc535950891)

[3.3.4. Локализация приложения 10](#_Toc535950892)

[3.4. Реализация передачи данных 10](#_Toc535950893)

[3.4.1. Реализация со стороны Android-приложения. 10](#_Toc535950894)

[3.4.2. Реализация со стороны микроконтроллера 12](#_Toc535950895)

[Заключение 14](#_Toc535950896)

[Список литературы 15](#_Toc535950897)

[Приложение 16](#_Toc535950898)

# Введение

Люди могут все. Так говорят, подразумевая, что для человека, который действительно чего-то хочет, нет ничего невозможного. Но, к сожалению, 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, хотя есть люди с ограниченными возможностями слуха. Цель проекта – 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 разработкой проекта 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 изучили работы 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 требуются голова, тело, плечи, и обе руки. Но поскольку каждый такой жест можно представить как несколько простых жестов, обозначающих одну букву, было принято решение не учитывать такие жесты при создании и обучении модели. На рисунке 1[[1]](#footnote-1) представлен «Жестовый алфавит» – русский алфавит, переведенный на 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 проекта: имел высокую производительность, достаточное количество портов для подключения к нему различных сторонних модулей и был прост в освоении.

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

## 1.4. Разработка чертежа манипулятора

Мы изучили существующие схемы сборки манипулятора у наших иностранных коллег. Большая их часть представляла собой файлы 3D-моделей для печати на 3D-принтере. После изучения 7 прототипов этих моделей мы пришли к выводу, что несмотря на 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 прочности необходимо печатать их довольно толстым слоем пластика (либо использовать более дорогостоящую технологию порошковой печати), в итоге прототип получится слишком тяжелым.

Параллельно с поиском соответствующих чертежей в интернете мы подготовили несколько вариантов своих — для разработки модели манипулятора без 3D-технологий. Но, как оказалось при сравнительном анализе наших чертежей и из открытых баз в интернете, использование 3D-принтера при создании прототипа, во-первых, намного упрощает задачу, а, во-вторых, является наиболее точным методом, что немаловажно в нашем проекте. Тем более, при создании нескольких частей-прототипов по нашим чертежам оказалось, что пенополистирол, который мы предполагали использовать из-за его легкости, тяжело обрабатывается без специальных станков, а также режется леской, которая, как предполагала конструкция, должна сгибать пальцы руки-манипулятора. Было решено взять готовые чертежи. Нам удалось отыскать удачный проект, расположенный в французском открытом банке 3D-проектов «InMoov», который мы решили использовать.

# Глава 2. Сборка модели

## 2.1. Создание деталей

### 2.1.1 3D-модель

Коме проекта модели нам было необходимо было найти 3D-принтер, на котором были бы распечатаны детали. В нашем лицее находится 3D-принтер, печатающий по технологии «Послойное наплавление (FDM)». Технология распространена из-за своей доступности и дешевизны. Именно ее мы решили использовать для печати модели.

## 2.2. Процесс сборки

### 2.2.1 Сборка пальцев

Первым этапом в процессе сборке была сборка пальцев.

Пальцы собирались из фаланг, каждая фаланга – из 2 деталей. На каждой фаланге присутствуют отверстия для соединения деталей между собой. В эти отверстия вставлен жесткий жгут, служащий осью вращения. Также внутри пальца продеваютя две жестких нити или лески, которые подтягиваются с помощью сервоприводов и позволяют сгибать и разгибать палец.

### 2.2.2 Сборка предплечья и запястья

Запястьяе состоит из нескольких деталей: массивная деталь, служит основой для него, на эту деталь закрепляются еще 3 подвижные детали, на которые, устанавливаются пальцы. Запястье закрепляется на предплечье через систему шестеренок, которые позволяют ему вращаться вокруг своей оси.

Предплечье состоит из двух больших деталей, похожих на половины цилиндра. Благодаря внутреннему размещению электроники манипулятор становится более компактным и удобным в использовании.

## 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).

Arduino, Bluetooth-модуль и LCD-монитор можно от обычного powerbank-а[[2]](#footnote-2), который используются для зарядки мобильных устройств.

Гораздо сложнее было подключить сервоприводы. Пяти сервоприводам не хватает силы тока или напряжения микроконтроллера. Решение проблемы мы видели в установке дополнительного источника питания, или его стабилизатора, поскольку при слишком большом напряжении, ток на сервоприводах окажется слишком большой, и они сгорят.

## 2.4 «Стрела».

«Стрела» – продукт компании «Амперка» – платформа, выпускаемая как материнская плата для постройки различных устройств на базе Arduino. «Стрела» представляет собой микроконтроллер Arduino с несколькими «шилдами»[[3]](#footnote-3). Фактически, «Стрела» была использована в качестве источника питания (см. выше). Потребляя характерное для Arduino напряжение, она могла обеспечить питанием 5 сервоприводов.

# 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++. Компилятор, преобразующий программный код в бинарный – код, который может воспринять процессор микроконтроллера – адаптирован для языка C++ (язык программирования Arduino IDE основан на C++).

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 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. 2).

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[[4]](#footnote-4), 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[[5]](#footnote-5) 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. 3).

При запуске какой-либо активности (открытии одного из трех режимов управления) в отдельном потоке[[6]](#footnote-6) 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[[7]](#footnote-7) 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[[8]](#footnote-8) 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[[9]](#footnote-9). 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. 4), 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



Рис. 1. Язык жестов.



Рис. 2. Главный экран приложения.



Рис. 3. Включение Bluetooth.



Рис. 4. Кодовая таблица CP1251.

1. Данный и все последующие рисунки находятся в разделе «Приложение» под соответствующим номером. [↑](#footnote-ref-1)
2. Powerbank (англ.) – Портативный аккумулятор. Устройство, используемое для зарядки смартфонов, планшетов и другой портативной техники. [↑](#footnote-ref-2)
3. Shield (англ. – щит.) – Некоторая микросхема, надеваемая на микроконтроллер или другую микросхему, расширяющая (или любым другом образом дополняющая) его. [↑](#footnote-ref-3)
4. Seek bar – один из виджетов, применяемых в разработке Android-приложений. Представляет собой шкалу с выбором значения. [↑](#footnote-ref-4)
5. Среда разработки приложений для устройств на базе операционной системы Android, в которой создавался проект. [↑](#footnote-ref-5)
6. 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-6)
7. Tóast (англ.) – тост. Профессиональное название типа уведомлений, в котором уведомление, подобно хлебному тосту из тостера, выскакивает снизу и пропадает через несколько секунд. [↑](#footnote-ref-7)
8. 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-8)
9. 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-9)