МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«Национальный исследовательский ядерный университет «МИФИ»**
**(НИЯУ МИФИ)**

**Утверждаю**
Проректор НИЯУ МИФИ

_____Е.Б. Весна

«____»_____ 2021 г.

# «Робототехника (11 класс)»

Авторы:
Климов В.В., кандидат технических наук, заместитель директора Института интеллектуальных кибернетических систем НИЯУ МИФИ
Барышев Г.К., старший преподаватель кафедры конструирования приборов и установок НИЯУ МИФИ
Бирюков А.П., инженер кафедры конструирования приборов и установок НИЯУ МИФИ
Сатдарова М.Л., лаборант кафедры конструирования приборов и установок НИЯУ МИФИ

Москва 2021

**Содержание**

# 1. Введение в робототехнику

## 1.1. Основные понятия

Для начала введем общее определение изучаемой нами науки.

Робототехника - раздел фундаментальной и прикладной науки, который занимается вопросами проектирования, производства и применения автоматических и автоматизированных технических систем - роботов.

Так как робототехника является прикладной наукой, она содержит в себе множество направлений развития и составляющих, а потому использует крайне широкий набор терминов. Рассмотрим самые часто встречающиеся из них

Робот — это перепрограммируемое механическое устройство целевого назначения, способное действовать без помощи человека. Такое определение совмещает в себе сразу две черты любого робототехнического изделия – оно должно иметь управляющий центр, то есть, быть программируемым на нужные создателю задачи; и оно должно иметь механические составляющие: опоры, манипуляторы, шарниры, датчики и прочие компоненты. Сочетание механического и программируемого аппарата позволяет достичь исполнения каких-либо команд без помощи человека, либо автоматически, либо под удаленным управлением.

Часто вместе с понятиями робота и робототехники используют термин мехатроника – область науки и техники, основанная на синергетическом объединении узлов точной механики с электронными, электротехническими и компьютерными компонентами, обеспечивающими проектирование и производство качественно новых механизмов, машин и систем с интеллектуальным управлением их функциональными движениями.

Существует также подразделение роботов на два основных типа – промышленные и обслуживающие. В последнее время в связи с развитием направления во многих сферах жизни классификация роботов постоянно дополняется. Все устройства такого типа обладают схожей структурой, изображенной на рисунке.

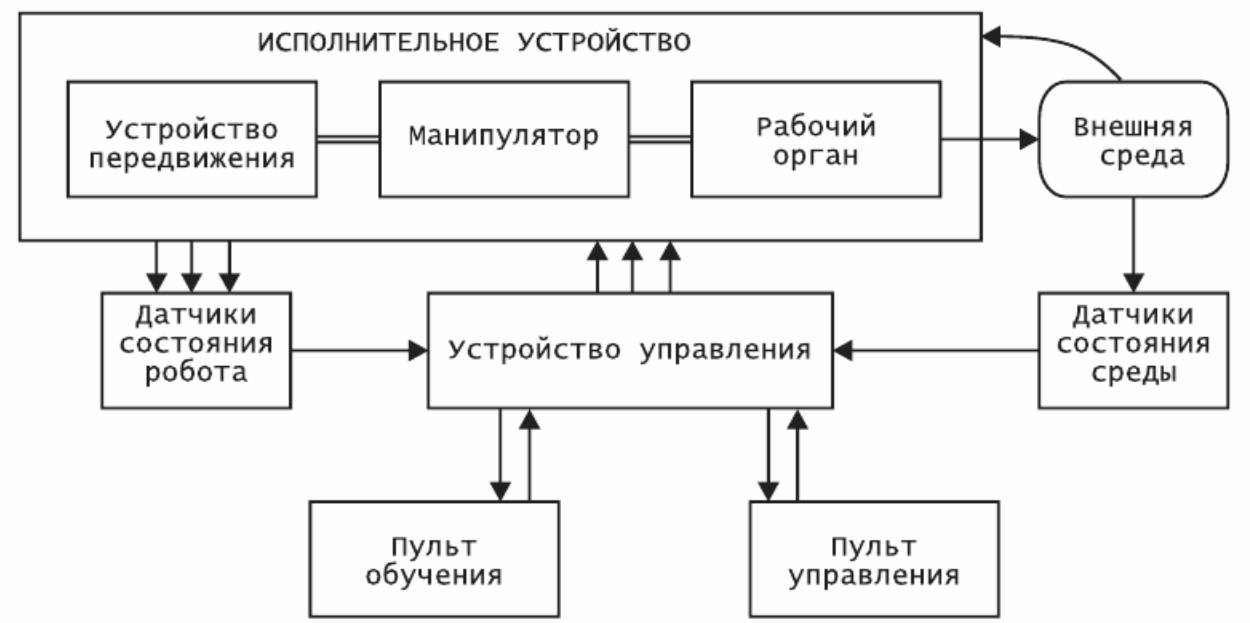


Рисунок 1 - Схема составных элементов робототехнической системы

Как видно из иллюстрации, основная часть робота – это исполнительное устройство, которое содержит в себе модули перемещения, манипуляторы и рабочие органы, которые непосредственно взаимодействуют с окружающей средой. Успешное взаимодействие невозможно без комплекса датчиков состояний робота и окружающей среды, коммуницирующих между собой с помощью общего устройства управления. Посылая команды на устройство управления, можно настраивать систему и манипулировать ей.

Далее в тексте будут рассмотрены основные разделы робототехники, позволяющие успешно координировать и подбирать все составные части робототехнических систем.

## 1.2. Разделы робототехники

Условно робототехническое устройство можно разделить на две большие логические части – механическую и управляющую. Исходя из них можно назвать основные разделы того, что должна включать в себя робототехническая система. По этим разделам и будет строиться изучение курса:

1. Программирование микроконтроллеров. Это большой и важный раздел, целиком посвященный написанию управляющих программ будущего устройства, их откладке, способам оптимизации системных ресурсов. В данном пособии упор сделан на наиболее часто встречающийся в робототехнических задачах язык С. Для обработки данных и прочих задач последнее время популярность набирает язык Python, который оказывается для многих проще в освоении.

2. Схемотехника - самостоятельная инженерная наука, занимающаяся электрическими платами и цепями. Использование ее в робототехнике необходимо для создания самостоятельных оптимизированных устройств. Уверенное знание принципов построения электрических цепей, строения электронных плат и умение обращаться с ними сыграет большую роль в успехе любого робототехнического проекта. Специализированные программные средства позволяют проводить инженерные расчеты максимально удобно, так же как и проектировать и отлаживать электрические схемы и платы. Данный курс помимо теоретических основ схемотехники включает полезные прикладные навыки, в том числе методику создания плат и процесс их заказа на производстве для дальнейшего использования в работе.

3. 3D-моделирование, деталирование - еще один важнейший раздел. С помощью систем автоматизированного проектирования появляется возможность создать полный прототип своего проекта. Помимо простого прототипирования, моделирование в данном случае включает в себя правила оформления чертежей и минимума конструкторской документации. Наличие

этого раздела позволяет свободно оформить свою идею, представить ее и вести работу наиболее систематизированно

4. Механика, инженерия - область знаний, которая будет задействована при создании механических элементов устройства. Она включает в себя основные принципы создания механизмов, способы соединения различных деталей. В ходе этого раздела будут изучены часто используемые решения и области их применения: шестерни, ремни планетарная и червячная передачи. Задача механики - позволить создавать конкурентоспособные и логичные инженерные решения механической части робототехнической системы.

Изучение перечисленных разделов представляется достаточно затруднительным без наличия специальных программных средств. Многие из них обладают либо открытой, либо доступной по ученической подписке лицензией и позволяют комфортно работать над заданием. Далее рассмотрим наиболее популярные и удобные из них.

### 1.3. Обзор ПО для обучения

Мы уже познакомились с основными понятиями в робототехнике и узнали, из какие основные разделы она в себя включает. Для дальнейшего обучения полезно ознакомиться с основными программами, которые можно использовать как для обучения, так и для дальнейшего развития и работы в сфере робототехники.

Робототехника, как мы уже выяснили, состоит из трёх основных частей. Это электроника, программирование и инженерия. Мы рассмотрим программы для каждого из направлений по отдельности. Все представленное ниже ПО можно скачать и использовать бесплатно на различных условиях. В виде учебных версий, сокращенных, либо же данное ПО является полностью бесплатным.

- **Tinkercad**

Для базового обучения электронике, программированию (и, при желании, 3D моделированию) очень удобно использовать такой сервис как Tinkercad. Данная программа является бесплатной и позволяет быстро начать обучение, так как не требует установки. Всё, что нужно, — это пройти регистрацию на сайте. Tinkercad обладает хорошим функционалом и встроенными уроками по электронике и программированию.

По данным урокам и заранее сделанным проектам можно самостоятельно начать осваивать азы электроники и программирования. В Tinkercad существует несколько видов программирования. Во-первых, это обычный текстовый формат кода на Си-подобном языке. Во-вторых, это блочное программирование, которое позволяет создавать базовые программы, не прибегая к написанию текстового кода. Это даёт возможность на начальных этапах быстро получать готовые схемы, устройства без необходимости разбираться в синтаксисе, что значительно увеличивает скорость работы.

Однако, набор блоков, из которых можно писать программу, достаточно небольшой. Поэтому нет возможности написать программы для подключения некоторых датчиков, экранов и так далее, но для начальных этапов обучения ПО подойдет. Помимо этого, есть промежуточный вариант, в котором программа представлена сразу в виде блоков и текста. Этот формат позволяет сделать плавный переход от начального обучения написанию программ в виде блоков к более применимому в реальности. При этом понимание структуры программы, работы циклов, логические операции не нужно будет осмысливать заново.

Tinkercad позволяет не только заниматься самостоятельно, но и работать в составе класса с преподавателем. Помимо этого, существует еще два раздела, которые позволяют строить 3D модели по двум различным принципам. Однако, с точки зрения обучения проектированию и 3D моделированию, не рекомендуется использовать данное ПО, так как принципы построения модели отличаются от большинства общепринятых и используемых в других, более профессиональных системах проектирования. Для обучения 3D моделированию рекомендуется использовать иные программы, которые будут приведены далее.

- **Ardublok**

  Помимо Tinkercad, для начального обучения программированию возможно использование Ardublok. Данная программа представлена в двух версиях. При необходимости ее можно установить на компьютер. Однако, как и в случае с Tinkercad, существует версия, которая позволяет работать онлайн.

  Данное ПО даёт возможность использовать простое программирование в виде блоков кода для построения программы, которая далее будет переведена в формат обычного текстового кода. Текстовый код, в свою очередь, можно загрузить в обычную схему или в виртуальную в Tinkercad.

  Однако, существуют некоторые изъяны, которые требуют небольших изменений в коде программы. При этом удобство ее использования и обилие блоков для различных специализированных модулей позволяют писать любые, даже сложные программы. Такой подход даёт возможность быстро обучиться структурированно писать программный

код, понять основные принципы и закономерности работы программ, не отвлекаясь на написание текста.

- **Arduino IDE**

  Основное ПО, которое используется для программирования микроконтроллера Arduino. Arduino IDE является самым популярным и первым ПО, которое появилось для этих целей. Его можно использовать как для обучения, так и для дальнейшей работы. Arduino IDE специально разработана для написания программ под Arduino и схожие контроллеры.

  Язык разработки является Си-подобным. Данная программа обладает широким спектром специализированных библиотек для подключения различных модулей, что позволяет достаточно просто начать писать для сложных компонентов, не вникая глубоко в принцип их работы.

  Помимо этого, существует множество уроков, примеров написания программ в Arduino IDE. Также присутствуют встроенные образцы с подробными комментариями разработчиков. Как правило, данные комментарии написаны на английском языке, поэтому для работы с Arduino, как и в целом для программирования других платформ, было бы плюсом знание английского языка или уверенное пользование переводчиком.

Помимо перечисленных, существует еще некоторое количество программ, которые позволяют писать код уже под более серьезные проекты, основанные на контроллерах семейства STM 32. Нужно отметить, что данные контроллеры являются более сложными для освоения и имеют достаточно высокий порог вхождения. Поэтому не рекомендуется приступать к их

изучению до того момента, пока нет понимания Arduino на достаточно хорошем уровне.

В данном учебнике мы приведем лишь краткую справку по работе с данными контроллерами, чтобы имелось представление об их существовании и том, куда стоит двигаться далее. Для работы с STM можно использовать различные виды ПО и их комбинации. Например, CubeMX и Keil, либо же CubeMX и Cude IDE, а также и Keil, Cube IDE по-отдельности.

Для начала необходимо разобраться с самим принципом работы в программах. В отличии от Arduino, при работе с STM у разработчика имеется значительно больше возможностей по настройке и работе с тем или иным контроллером. Это, соответственно, усложняет процесс работы. В случае с Arduino все пины (входы платы, на которые можно подключать устройства) заранее разделены на группы: цифровые, аналоговые и поддерживающие ШИМ (широтно-импульсная модуляция). В STM инициализация, то есть определение типа пинов, производится каждый раз во время написания кода.

Тем не менее, существуют программы, которые могут упростить этот процесс.

- **CubeMx**

  Позволяет с помощью графического интерфейса произвести инициализацию пинов. После того, как собран нужный вариант распиновки (то есть, сочетания нужных входов) платы, программа автоматически формирует основу, в которой уже созданы все пины в соответствии со схемой. После этого написание кода становится не сильно сложнее, чем написание программы под Arduino.

Для написания программ под микроконтроллеры STM чаще всего используется язык Си. Однако, в отличие от Arduino, существует и множество других достаточно популярных языков программирования, на которых можно

писать код под контроллеры серии STM. Можно выбрать язык, который будет понятнее и удобнее в использовании.

Рассмотрим программы, которые используются для разработки электроники, составления электронных схем. Для базового освоения удобно использовать Tinkercad, упомянутый ранее. Данный сервис позволяет составлять несложные электронные схемы как с использованием микроконтроллеров, так и без. В этом случае основным достоинством является простота работы. В отличие от других программ, которые будут рассмотрены далее, все электронные компоненты представлены в виде рисунков самих компонентов, а не условных обозначений. Это упрощает разработку. Данную программу рекомендуется использовать именно для начального обучения. В Tinkercad имеются заранее созданные схемы, которые расположены в специальной последовательности и позволяют пройти мини курс обучения уже в самой программе.

Помимо практики в электронике и программировании, полезным может оказаться проектирование и создание собственных плат. Для того, чтобы создавать их и принципиальные схемы к ним же, можно использовать различные виды программ.

- **Fritzing**

  Одной из наиболее удобных для создания своих собственных плат и схем является программа Fritzing. В ней достаточно просто сделать переход от работы с платами, как с изображениями самих компонентов, к уловным обозначениям, принятым в электронике. В программе Fritzing можно собирать схемы по аналогии со схемами в Tinkercad, однако существует большее разнообразие компонентов, что позволяет делать более сложные схемы.

  После составления схемы возможно перейти к отображению этой же схемы в виде условных обозначений. Помимо этого, существует вариант

отображения в виде гербер файлов, которые можно использовать для создания печатной платы. При этом все три вида связаны в один проект, что позволяет производить редактирование схемы в любом из видов и получать результат сразу во всех отображениях. Данное ПО отлично подходит как для обучения, так и выполнения проектов любого уровня.

- **EasyEDA**

  Позволяет создавать более сложные платы, чем Fritzing, а также сразу же заказать их изготовление на производстве после разработки.

  EasyEDA является бесплатной, обладает широким спектром компонентов. При этом существует возможность перенести в EasyEDA проект из какой-либо другой программы по созданию принципиальных схем и отредактировать его. Это позволяет производить совместную разработку схемы, даже с использованием различного ПО. Помимо этого, программа является облачной, то есть, разработку можно вести браузере. Также в программе есть сеть встроенных уроков, которые позволяют быстро начать работать и создавать свои собственные проекты.

Помимо программ, которые мы перечислили выше, есть еще множество ПО, позволяющих создавать те или виды разработки. В разделе в качестве примеров приведены те продукты, которые подходят для обучения. Помимо них можно рассматривать Altium designer, TinyCad, ZenitPCB и другие. В целом, важнее не хорошо подобранное обеспечение, а знание основных принципов работы в нужной области.

Для создания механики какого-либо механизма используются такие программы, как САПРы (системы автоматизированного проектирования). САПР — это программа, которая позволяет создать 3D модель, сборку и чертеж какого-либо изделия. Иногда данное ПО также позволяет произвести

какие-либо расчеты (например, расчет прочности изделия). Помимо этого, некоторые программы обладают специальными модулями для подготовки управляющей программы для фрезерного, токарного станка с ЧПУ, либо же для 3D принтера.

Tinkercad имеет модули и для создания 3D моделей. В программе есть два модуля для этих целей. Первый — “Блоки кода’’, где разработка 3D моделей производится за счет соединения различных блоков. В качестве блоков представлены геометрические фигуры (параллелепипед, конус, пирамида), а также блоки по изменению геометрии (объединение, разделение, перемещение). Данную часть программы, как уже говорилось ранее, не рекомендуется использовать, так как навыки, полученные тут, не пригодятся в дальнейшем при работе с более профессиональным ПО, но могут пригодиться при написании программ.

Структура построения модели аналогична структуре написания программы. Помимо этого, в Tinkercad есть модуль, позволяющий работать с 3D моделями уже в более привычном виде. В нем возможно создание 3D моделей также с помощью базовых форм и различных операций над ними. Однако, процесс их создания сильно отличается от процесса создания моделей в первом случае. Данную часть уже можно использовать для начального обучения и для быстрого создания несложным форм.

Перечислим основные программы, использующиеся для 3D моделирования.

- **Компас-3D**

  Программа позволяет выполнять весь цикл разработки какой-либо конструкции — от создания детали, сборки, до чертежей и спецификаций. Компас-3D — это программа, созданная в России, поэтому она адаптирована под нормы ЕСКД (единая система конструкторской документации — объединение нормативных документов, регулирующих порядок разработки тех или иных изделий)

и позволяет создавать чертежи и спецификации намного удобнее, чем аналогичное ПО от иностранных производителей.

Есть и недостатки. Например, при работе со сложными элементами, имеющими множество поверхностей, либо же со сборками, состоящими из большого количества компонентов, программа может перегружаться и экстренно завершать работу. В целом, это характерно для многих ресурсоемких программных продуктов.

Программа имеет встроенные уроки Азбука Компас, что позволяет достаточно легко научиться пользоваться ей. При этом разработкой ПО занимается компания Аскон, что означает наличие понятного русскоязычного интерфейса и множества статей по его использованию.

- **Autodesk Inventor**

  Перейдем к программе со схожими параметрами. Она, как и Компас 3D, позволяет создавать детали, сборки, оформлять чертежи и спецификации. Программа иностранная, но при этом существуют версии, позволяющие делать оформление по ЕСКД, хотя и не совсем эталонное.

  Все недостатки при необходимости можно исправить вручную. Преимуществом данной программы является ее оптимизация, возникает меньше проблем при создании сложных объектов.

  Данное ПО обладает тем же функционалом, что и Компас 3D, однако, есть еще и ряд дополнительных модулей, которые позволяют дополнить создание модели еще и расчетами.

Есть и другой способ решения проблемы с оформлением чертежей по ЕСКД. Для моделирования используют Autodesk Inventor либо SolidWorks (еще одна CAD-система), а для оформления чертежей и остальной документации применяют Компас 3D.

- **Fusion 360**

  Данная программа также является продуктом компании Autodesk. Она позволяет создавать модели, сборки и чертежи, но принцип построения отличается от многих аналогов.

  Принцип построения модели более прост и понятен, поэтому данная программа подойдет для базового обучения 3D моделированию. При этом многие продолжают использовать ее и при дальнейшей работе, так как программа предназначена и для профессионалов.

Описанные программы имеют сходные принципы создания и обработки моделей, поэтому решение по поводу того, что именно использовать в работе, остается за пользователем. В данном пособии многие примеры будут разобраны в программах из этого списка, потому что они наиболее наглядны.

# 2. Микроконтроллеры

Микроконтроллер - управляющая плата, которая отвечает за координацию и обмен данных между периферийными устройствами. Периферийные устройства бывают самые разные. К ним относятся и дисплеи, и датчики, и переключатели, и движки в том числе. Корректное программирование контроллера позволяет получить отлаженный алгоритм работы устройства.

Как правило, при планировании создания устройства автор волен выбрать любой подходящий по мощностям микроконтроллер, однако большинство из них программируется на языке С, чистом или в том или ином виде упрощенным, дополненным с помощью различных библиотек.

В этом модуле будет разобрана база языка С, описаны основные его составляющие и алгоритмы, т.е. готовые решения для работы с датчиками и данными, которые часто встречаются в робототехнических задачах.

## 2.1. Язык Си. Синтаксис, простейшие понятия

Язык Си – один из наиболее распространенных языков программирования на сегодняшний день, несмотря на обилие новых высокоуровневых языков. Он включает в себя основные конструкции, требуемые для хорошо структурированных программ: группирование операторов, принятие решений (IF), циклы с разной постановкой условия (в начале или в конце), выбор одного из множества. Реализована система указателей и методы адресной арифметики.

Функционал Си включает в себя много различных программных решений. Например, аргументы функций передаются посредством копирования значения аргумента, так, вызванная функция не сможет изменить фактический аргумент. Реализованы инструменты для работы с массивами.

Так как Си разработан достаточно давно и имеет большую историю, есть набор очевидных несовершенств языка, с которыми приходится мириться – несколько не совсем совместимых между собой версий языка, например, или некоторые неочевидные нюансы синтаксиса.

Рассмотрим основы языка последовательно. Возьмем классическую тестовую программу вывода на экран компьютера сообщения «Hello, world». С помощью нее можно понять, как в самом примитивном случае выстраивается программа в данном языке программирования.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World");
6
7     return 0;
8 }
```

Попробуем разобраться с происходящим. Некоторые моменты впоследствии будут разобраны подробнее и с конкретными примерами.

Начнем с первой строки – команда #include нужна, чтобы подключить к программе библиотеку – файл с набором команд, нужным для выполнения определенных функций. Файл разрешения .h называется заголовочным и указывается в угловых скобках.

Тело самой программы, включающее объявление переменных, все операторы, функции и процедуры, заключается в глобальной функции main(). Все, что должна делать программа, должно быть заключено в фигурных скобках этой функции. Замыкает любую программу как правило команда return 0, завершающая выполнение функции main.

Операторы в языке Си обязательно разделяются точкой с запятой. Без этого знака компилятор выдаст ошибку.

Стандартный вывод сообщения на экран осуществляется командой printf(). В качестве аргумента, в скобках, указывается то, что пользователь собирается вывести на экран. В данном случае мы выводим строку – специальный тип данных, с сообщением «Hello world».

Рассмотрим функцию printf() подробнее. Вывод на экран правильнее называть выводом в стандартный поток вывода. В общем случае функция выводит на экран строку, переданную первым аргументом, заменяя в ней специальные комбинации символов преобразованными данными, переданными ранее в аргументах. Каждый тип данных имеет свое обозначение.

Таким образом записи

```
printf("Hello World");
printf("%s", "Hello World");
```

равнозначны. Во втором варианте перед указанием самой строки, которую нужно отобразить, указывается спецификатор строкового типа данных - %s. Для целых чисел используется %d. Если нужно выделить больше знакомест, чем занимает число, нужное число мест указывается перед буквенным обозначением, а выравнивание выставляется автоматически по правому краю. Если выровнять нужно с левой стороны, перед числом ставится знак минус (%10d, %-10d).

Если в одной строке идет вывод сразу несколько блоков данных, они указываются и перечисляются последовательно:

```
printf("%d %s, %d %s.\n", 6, "students", 1, "teacher");
```

### Типы данных

Далее разумно разобрать типы данных. Для корректной работы с переменной языку Си требуется знать, что в ней хранится – целое число в

определенном диапазоне, число с плавающей точкой, символ или строка. Для целых чисел существует несколько типов, которые отличаются друг от друга объемом отводимой на хранение переменной памяти компьютера.

Наиболее распространенный целочисленный тип данных – int, на который отводится 4 байта памяти. Следует быть внимательным и избегать ситуаций переполнения, когда вводимое или получаемое число превышает максимальное значение для данного типа – программа будет работать некорректно.

Помимо типа int существуют и некоторые другие модифицированные целочисленные типы данных – short, long, unsigned, unsigned short, unsigned long. Они отличаются количеством памяти, выделяемым на хранение числа.

Еще один важный и часто встречающийся тип данных – char. Он нужен для хранения в тексте программы символов. Он подразумевает под собой числа в диапазоне от -128 до 127. У каждого символа имеется соответствующее число по таблице символов ASCII. Каждому числу диапазона в соответствие ставится определенный знак, который можно посмотреть по специальной таблице.

Помимо целочисленных типов данных в программировании используются и вещественные типы данных, которые нужны для хранения дробных чисел в формате с плавающей точкой. К таким типам относятся float и double.

**Массивы**

Для хранения отдельных блоков данных в языке Си предусмотрена реализация массивов – матричных структур, которые могут содержать в себе различные переменные. В общем случае массивы можно разделить на две большие категории: статические и динамические. В первой категории

количество элементов массива задается при объявлении его в программе, во второй – оно может быть изменено в зависимости от конкретной ситуации использования, когда общее количество переменных, которые должны входить в структуру, предсказать невозможно. Ниже приведен пример объявления массива. Сначала указывается тип данных, которые будут храниться, затем – название, после него в квадратных скобках указывается количество элементов:

```
int array[60];
```

Каждому элементу присваивается порядковый номер, нумерация начинается с нуля. К каждому элементу, используя этот порядковый номер, можно обратиться, чтобы использовать его или поменять значение хранящейся переменной. Пример такого обращения расположен ниже:

```
points[k] = 0.1 + points[k-1];
```

Как и во многих других языках программирования реализован условный оператор if-else. Он достаточно часто применяется в кодах управляющих программ для устройств на базе микроконтроллера Arduino, поэтому далее мы обратим на него некоторое внимание.

Оператор if-else позволяет установить выполнение определенного набора действий при появлении какого-либо условия и в случаях, когда поставленное ранее условие не выполняется. Синтаксис достаточно прост:

```
if (логич_выражение)
{
//набор действий, если условие истинно
}
else
{
//набор действий, если условие ложно
}
```

Для записи логических выражений так же существуют вполне определенные правила. Если необходимо реализовать конструкцию логического «И», между логическими условиями используются знаки «&&», для «ИЛИ» - «||». Оператор if-else может ветвиться, то есть, в ходе выполнения проверять истинность сразу нескольких условий. Пример множественного ветвления:

```
if (…) {
 …
} else if (…) {
 …
} else if (…) {
 …
} else {
 …
}
```

## Циклы

Важное место в робототехнике занимают циклы. Они имеют простой синтаксис и позволяют настраивать выполняемые действия в зависимости от условий не один раз, как простой условный оператор, а с каким-либо счетчиком, например. Главная сложность циклов – правильные условия. Если условие будет неверным, программа зациклится и не закончит свое выполнение, что может привести к серьезным последствиям.

## While

Цикл while больше всего похож на рассмотренный ранее условный оператор. В начале каждого шага (еще шаги называют итерациями) программа проверяет выполнение введенного условия, и, если оно истинно, выполняет блок действий. Как только условие в начале перестает выполняться, программа выходит из цикла. Синтаксис достаточно прост:

```
while (логич_выражение)
{
```

```
   // набор действий, который будет выполняться до тех пор, пока условие
истинно
}
```

## Do while

Вариация структуры while, которая отличается только моментом проверки выполнения условия цикла – это происходит в конце каждой итерации, а не в начале.

```
do {
   // набор действий, который будет выполняться до тех пор, пока условие
истинно
} while (логич_выражение);
```

## For

Используется в робототехнике примерно с той же частотой, что и While, возможно, даже чаще. Отличается от остальных тем, что количество итераций цикла может быть зафиксировано пользователем, для этого вводится параметр-счетчик, и внутри цикла возможно указать условия его изменения. К синтаксису нужно привыкнуть: заголовок цикла состоит из трех частей, у каждой из которых своя задача. Сначала указывается, какая переменная передается в цикл в качестве счетчика, возможно задать ее начальное значение. Через точку с запятой указывается порог этой переменной, сверху или снизу, это логическое условие. В последней части указывается условие изменения переменной в конце каждого шага – отдельно в теле цикла это прописывать не нужно, достаточно указать именно в заголовке.

Далее все уже привычно – в фигурных скобках указывается тело цикла. Выглядит это примерно так:

```
for (объявление счетчика и значения; диапазон изменения; правило изменения
счетчика)
{
// набор действий
```

```
}
```

Помимо описанных выше структур, язык содержит множество других. Больше о них можно узнать в справочниках языка программирования, либо в самоучителях. Далее мы будем обращаться к Си уже в контексте программирования микроконтроллеров типа Arduino – в основе так же будет лежать Си, но модернизированный и упрощенный относительно классического варианта.

## 2.2. Микроконтроллер Arduino

Как уже ранее упоминалось, основополагающая часть любой робототехнической системы – это ее управляющее устройство. Главное его назначение – координировать работу всех периферийных устройств, получать, обрабатывать данные, иногда – пересылать их на другие устройства. В качестве таких устройств используют микроконтроллеры, микропроцессорные системы, те же компьютеры. Для сложных устройств конструируют собственные сложные разветвленные платы управления.

Если в системе не должно быть больших вычислительных мощностей, то есть, она не должна самостоятельно обучаться, обрабатывать и анализировать данные, целесообразно использовать микроконтроллеры. Микроконтроллеры – это такие устройства, которые созданы для объединения нескольких периферийных компонентов с целью увеличения их взаимодействия и общего быстродействия системы. В отличие, например, от микропроцессора, который содержит только самые необходимые элементы (память, источник тактирования и пр.), микроконтроллеры могут быть оснащены самой разной периферией, которая превращает их в самостоятельные производительные устройства. Существует определенный набор компонентов, которые встречаются у большинства микроконтроллеров, например. Зачастую в робототехнике используются именно микроконтроллеры – благодаря их

универсальности и дешевизне относительно полноценных микропроцессорных систем.

На этапе обучения робототехнике необходимости использовать сложные персонализированные решения, как правило, не наблюдается. Системы начального уровня не требуют больших вычислительных мощностей для управления, потому появились более-менее универсальные решения. Одно из самых простых и популярных – система Arduino.

Arduino – это семейство микроконтроллеров, которое объединяет схожая структура, широкий набор всевозможных комплектующих, популярность и простота изучения. Все они программируются на одном и том же языке в одной и той же среде Arduino IDE. Все они подключаются к компьютеру через USB интерфейс и не требуют обязательного использования программатора, который нужен для работы с большинством аналогичных устройств. Многие, начиная свой пусть с Arduino, далее переходят на более сложные решения, используя усвоенные ранее принципы и методы построения управляющих систем. В этом параграфе рассмотрим основные параметры этого микроконтроллера, его применение, программирование.

Семейство Arduino содержит в себе микроконтроллеры с различными характеристиками, размерами и назначением. Рассмотрим самые известные из них.

### Arduino UNO

Бессменный лидер среди типовых микроконтроллеров, самый распространенный и популярный. Построен на базе контроллера Atmega328. Существует несколько версий, каждая из которых в чем-то усовершенствована относительно прошлых.

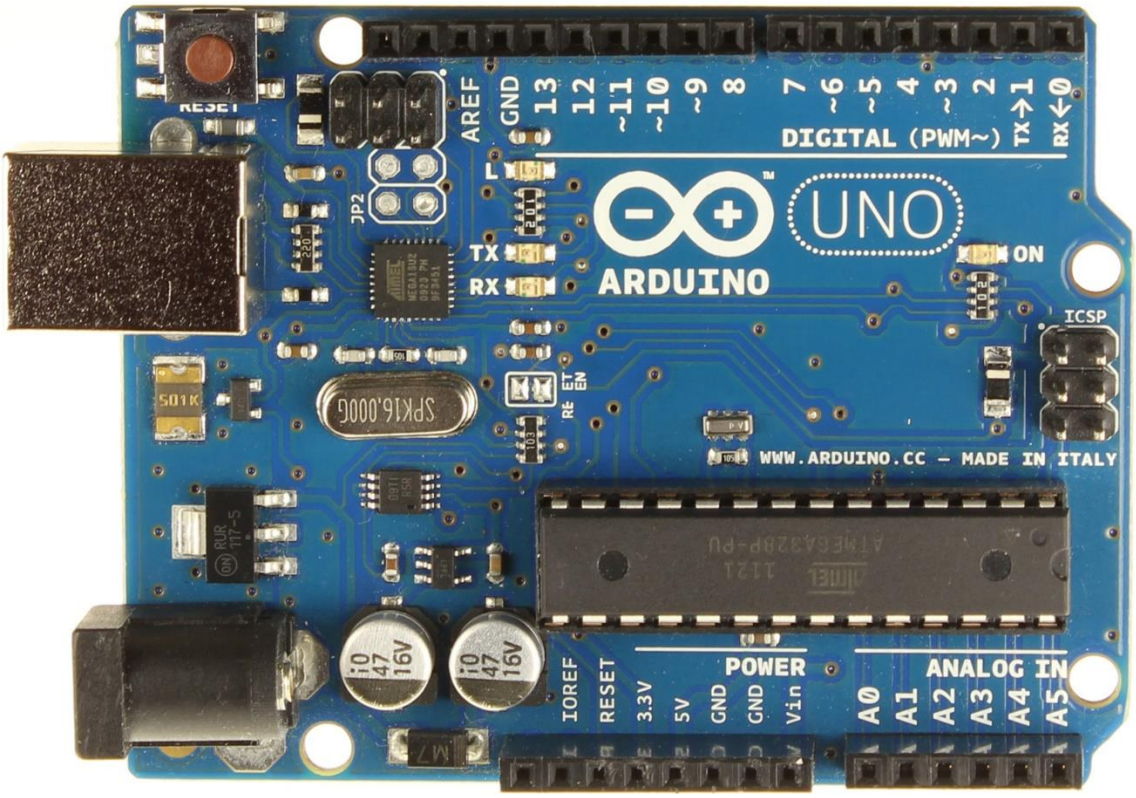


Рисунок 2 - Микроконтроллер Arduino UNO R3

## Arduino Leonardo

Основан на микроконтроллере ATmega 32U4. Отличается увеличенным числом выходов и усовершенствованным USB-подключением – помимо определения в качестве COM-порта, контроллер может определяться как мышь или клавиатура.

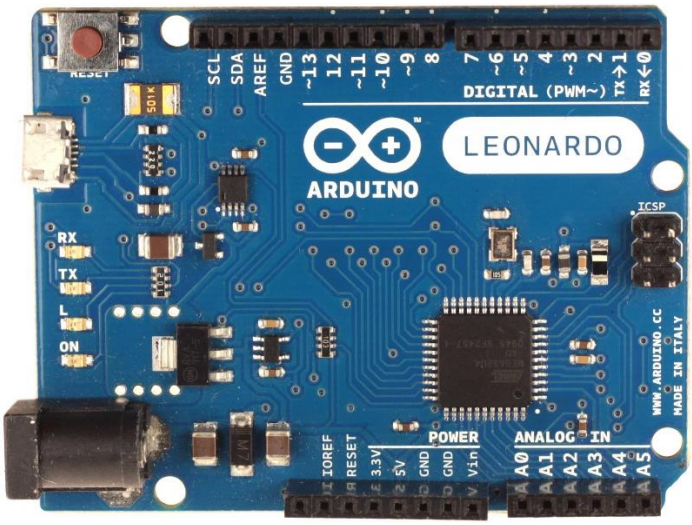


Рисунок 3 - Микроконтроллер Arduino Leonardo

## Arduino Nano

Маленький собрат остальных контроллеров, тем не менее представляющий собой полноценное устройство, незаменимое в проектах, в которых важна портативность и производительность. Построен на базе ATmega 328, как и UNO. Разъем внешнего питания отсутствует, USB-кабель подключения к компьютеру отличается от стандартного.


Рисунок 4- Микроконтроллер Arduino Nano

## Arduino Mega 2560

Еще один микроконтроллер семейства, отличается самым большим количеством выходов и характерными размерами. Спроектирован на основе микроконтроллера ATmega 2560, от чего и получил свое название. Имеет 4 приемопередатчика для реализации последовательных (обычно, для соединения с компьютером) интерфейсов, в то время как многие собратья имеют только один-два таковых. Так же имеет много версий.

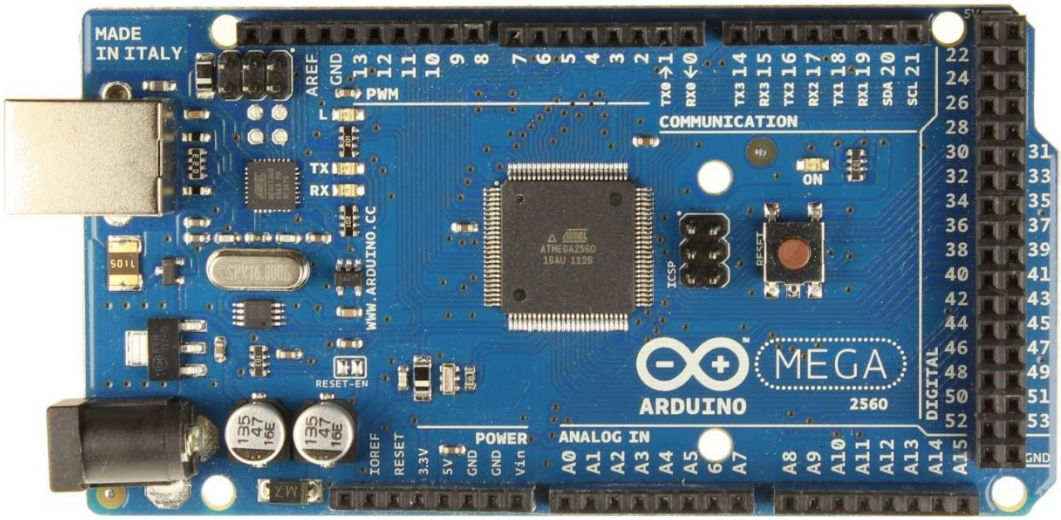


Рисунок 5- Микроконтроллер Arduino Mega 2560

Ниже представлена сравнительная таблица основных характеристик данных контроллеров.

Таблица 1 - характеристики контроллеров Arduino

| **Модель** | **Uno** | **Leonardo** | **Nano** | **Mega 2560** |
|---|---|---|---|---|
| **Контроллер** | ATmega328 | ATmega32u4 | Atmel ATmega168 или ATmega328 | ATmega2560 |
| **Цифровые входы/выходы** | 14 | 20 | 14 | 54 |
| **Из цифровых ШИМ** | 6 | 7 | 6 | 15 |
| **Аналоговые входы** | 6 | 12 | 8 | 16 |
| **Размеры, ДхШ** | 6,9x5,4 | 6,9x5,4 | 1,85x4,3 | 10,2x5,4 |

Контроллеры Arduino – далеко не единственный вариант управляющего устройства в робототехнической системе, однако они обладают ключевыми

характеристиками для начинающих электронщиков. Рассмотрим основные преимущества и недостатки системы.

К преимуществам можно отнести следующее:

- Распространенность и доступность. Контроллеры можно приобрести во многих магазинах электроники по достаточно демократичной цене по всей стране;
- Большое количество совместимых устройств. Существуют готовые наборы для робототехнического творчества, в целом с Arduino можно использовать крайне широкий спектр уже адаптированных к контроллеру датчиков и устройств;
- Обширное сообщество. Многие люди используют этот контроллер, поэтому в широком доступе большое количество уже готовых уроков и проектов, позволяющих отточить собственные навыки;
- Низкий порог вхождения. В образовательных целях многие электротехнические и программные нюансы в системе спрятаны так, чтобы даже начинающий пользователь смог без особого труда быстро собрать свою первую схему и запустить управляющую программу к ней;
- Наличие программ-эмуляторов. Благодаря широкому распространению, разработано несколько программных продуктов, которые позволяют симулировать работу с оборудованием, не имея к нему физического доступа.

Есть и несовершенства:

- Относительно невысокие мощности. На Arduino-системах не получится обрабатывать данные или работать с их большими объемами. Для этих целей используются микропроцессорные устройства, которые более сложны и более дороги;

- Ограниченность. Зачастую сложность проекта и его исполнение приходится подстраивать под микроконтроллер, что может отразиться на качестве конечного результата;
- Частота обработки входных сигналов аналоговых входов. Если плата должна быстро считывать сигналы высокой частоты, могут возникнуть проблемы.

**Компоненты**

К плате Arduino можно подключить крайне широкий спектр различных компонентов. В этом разделе рассмотрим основные категории устройств, которые могут работать с этими микроконтроллерами. Стоит упомянуть, что несколько контроллеров при необходимости могут работать в связке между собой.

Датчики

Под датчиками можно понимать любые элементы, которые каким-либо образом собирают информацию об окружающем мире и способны передавать ее управляющему устройству. К ним относятся фоторезисторы, термисторы, различные дальномеры, датчики влажности почвы, датчики Холла. Датчики как правило имеют провода питания и как минимум один сигнальный. К каждому такому устройству прилагается техническая документация со справкой по подключению к контроллеру и основными программными командами, при необходимости, с информацией о необходимости подключения тех или иных библиотек.

Модули передачи данных

С помощью этих устройств в систему можно добавить возможность работы с SD-картами, Wi-Fi и Bluetooth, присоединить контроллер к смартфону, тем самым настроив удаленное управление. Существуют

специальные GPS-модули на случай, если стоит задача построения маршрутов и ориентирования робота в пространстве. Например, данные, собираемые различными датчиками, можно отправлять на обрабатывающее устройство, не нагружая контроллер операциями, для которых он не совсем предназначен.

Shields или платы расширения

Также в просторечии называются «шилдами». Это специальные устройства, которые размещаются на плате соответственно ее выходам и выполняют какую-то определенную задачу. Как правило служат переходниками между микроконтроллером и каким-либо другим устройством. Например, на плате Arduino Mega 2560, используя плату расширения, можно построить и запрограммировать 3D-принтер.

Механические и радиоэлектронные компоненты

В этот раздел входят все движки, насосы, сервоприводы, реле, механические и сенсорные кнопки, всевозможные диоды, пьезоэлементы, вибромоторы. Это категория элементов, которые завязаны на механическом взаимодействии со средой, с пользователем, частей между собой, либо обеспечивают взаимодействие нескольких сходных элементов, например, с помощью модулей реле и им подобных можно построить колесные тележки.

Эта классификация далеко не полная, как минимум, потому что она не учитывает различные варианты дисплеев, которые могут быть подключены к микроконтроллеру.

В дальнейшем процедуру программирования и сбора электронных схем мы будем рассматривать на примере Arduino UNO, поэтому ознакомимся с этим микроконтроллером подробнее.

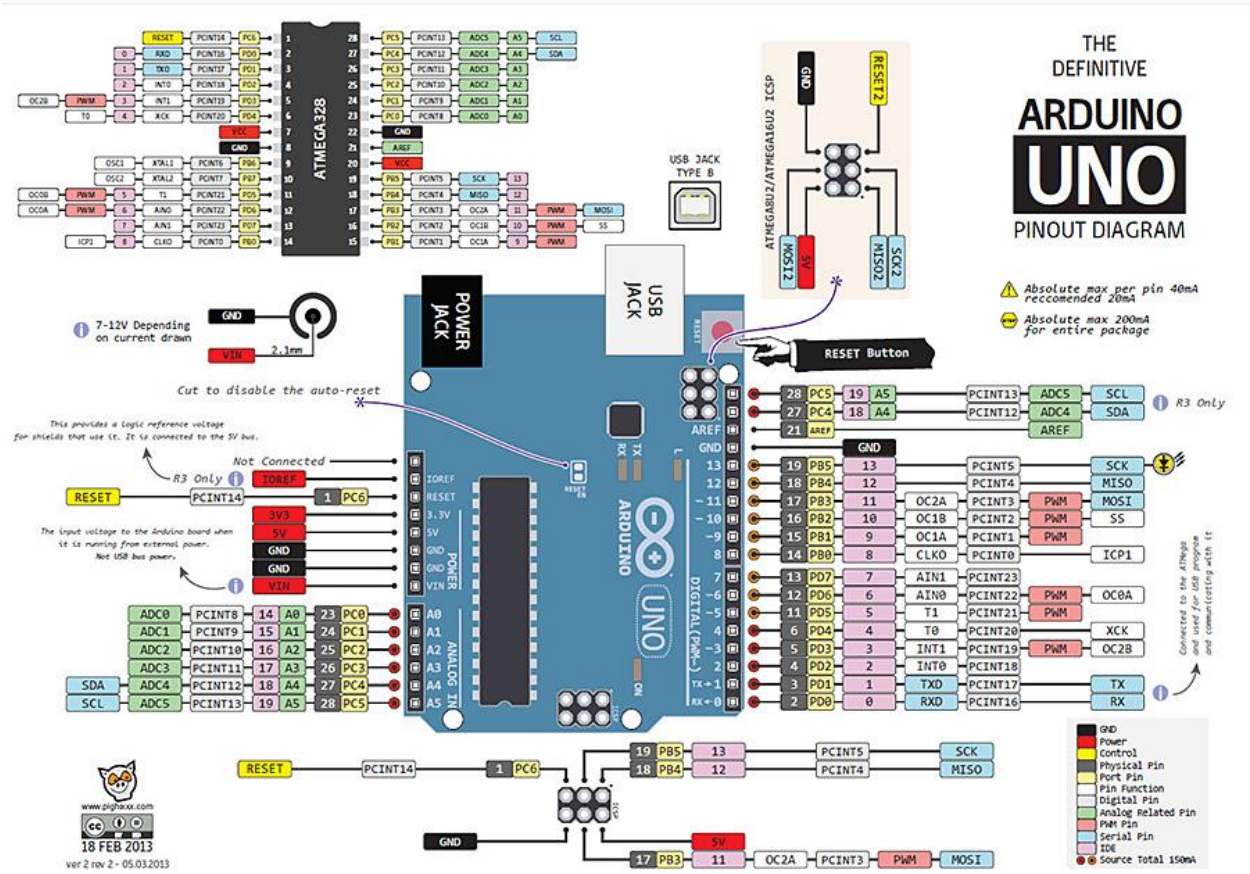


Рисунок 6- Расположение и назначение входов Arduino UNO

На представленном рисунке изображена «распиновка» этого микроконтроллера – подписан каждый выход самой платы и отдельно блока микроконтроллера. Обозначены его основные параметры и функции. На иллюстрации также указаны предупреждения о максимальном токе на один вход и на всей цепи. Несмотря на наличие в плате встроенного предохранителя, если эти значения превысить, больше микроконтроллер использовать не получится.

Каждый выход платы подписан и обозначен определенным цветом, отвечающим за назначение. Легенда указана на самом рисунке. Плата UNO оснащена 14 цифровыми выходами и 6 аналоговыми.

Цифровые порты обозначаются обычными числами от 0 до 13 и нужны для подключения к плате различных датчиков и устройств. Их отличительная особенность состоит в том, что они имеют только два значения для приема и

передачи – HIGH (соответствует логической единице) и LOW (соответствует нулю). Таким образом напряжение больше условных трех вольт такой вход считает как единицу, а близкое к нулю – как ноль. Существует два основных режима работы цифрового порта – на вход, если на него приходят данные с устройства и на выход, если данные отправляются на устройство с самого микроконтроллера.

Аналоговые порты подключены к аналого-цифровому преобразователю, который в Arduino UNO 10-разрядный, то есть, может оперировать значениями в диапазоне от 0 до 1023. Они носят обозначения A0-…-A5. В отличие от цифровых портов, на аналоговых напряжение можно регулировать, либо считывать зависимость напряжения от времени с определенной частотой, равной частоте работы преобразователя. Превышение этой частоты будет приводить к ошибкам и искажениям. Тем не менее, могут использоваться в качестве цифровых.

Рассмотрим основные обозначения.

VIN – разъем для питания от внешнего источника, поддерживает диапазон 7-12 В.

USB – разъем для подключения микроконтроллера к компьютеру или к питанию.

5V – может использоваться как источник питания для подключаемых устройств. С помощью макетной платы можно последовательно подключить несколько компонентов. Кроме того, через него же можно подать питание на плату, минуя внутренний преобразователь напряжения.

3.3V – порт с пониженным напряжением, который требуется для подключения некоторых компонентов.

GND – порты для подключения земли.

AREF – порт опорного напряжения для аналоговых входов. По умолчанию опорное напряжение составляет 5В, это и есть диапазон, в котором меняется сигнал. Оно может быть как уменьшено для точности преобразования, так и увеличено при необходимости.

IOREF – порт опорного напряжения для цифровых входов, через него можно регулировать и узнавать рабочее напряжение микроконтроллера.

Reset – чтобы полностью перезагрузить микроконтроллер, на этот вход нужно подать низкий уровень.

SCL, SDA – специальные входы, которые используются в протоколе обмена данными I2C, очень удобном для одновременного подключения многих устройств, работы с дисплеями и высокоскоростной передачи данных с малым процентом ошибок и искажений.

Цифровой выход 13 – помимо обычного выхода имеет встроенный светодиод, который пользователь может использовать, просто обратившись к нему.

Цифровые выходы 0 (RX), 1 (TX) – выводы интерфейса UART, обычные датчики на них подключать не рекомендуется.

Arduino UNO – упрощенная и облегченная версия обычных контроллеров, для программирования и прошивки которых на самом деле нужно еще одно буферное устройство, называемое программатором. К нашему контроллеру его так же можно подключать при необходимости. За это отвечают шесть контактов напротив разъемов питания платы.

Работа с микроконтроллером Arduino UNO

Деятельность по созданию каких-либо проектов состоит из трех основных этапов: подбора компонентов, создания схем и программирования. Подбор компонентов на начальном этапе состоит просто в поиске необходимых датчиков, создание схем заключается в правильном

подключении элементов соответственно инструкции, а программирование будет рассмотрено в данном разделе подробнее.

Как было сказано ранее, подключение контроллера к компьютеру осуществляется через USB. Компьютер как правило определяет плату как COM-порт. Далее вся работа с контроллером производится через ПО Arduino IDE – среду разработки и откладки Arduino программ, которая находится в свободном доступе. Вместе с обеспечением сразу устанавливаются необходимые для работы драйверы.

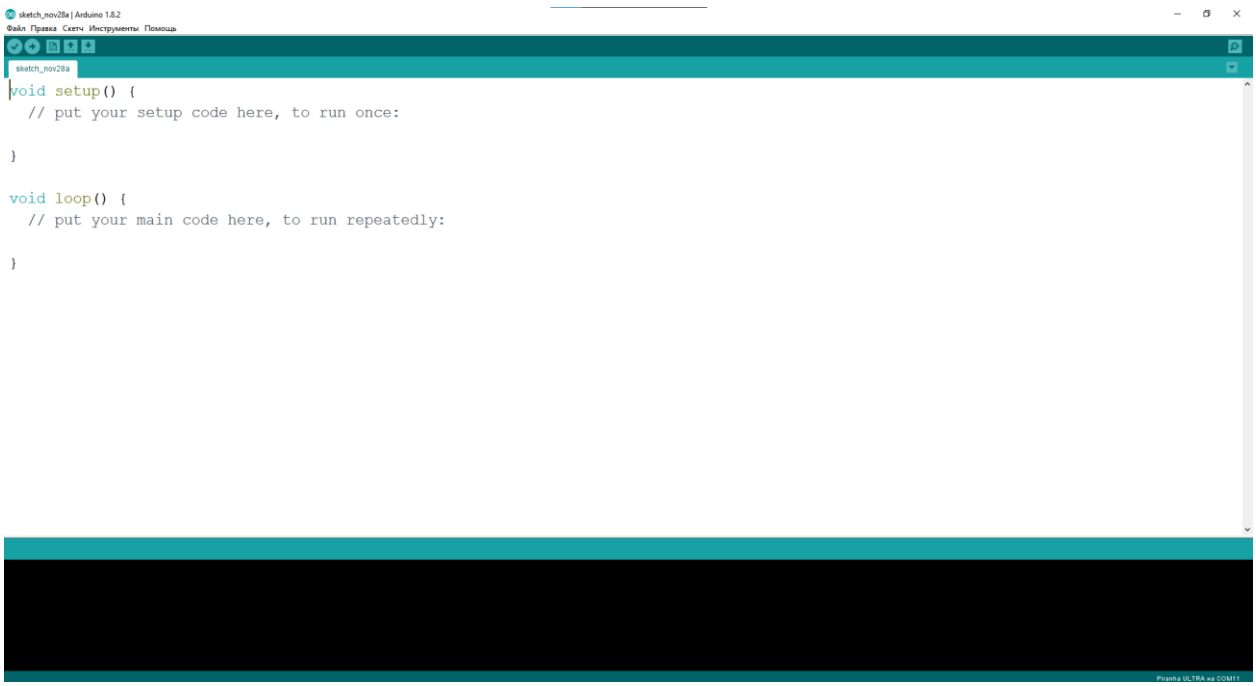


Рисунок 7- Интерфейс программы Arduino IDE

Интерфейс программы не сложен. В большом белом поле вводится код, черное поле под ним нужно для вывода ошибок и сообщений компилятора, который запускается в левом верхнем углу значком галочки. Для загрузки программы в плату используется соседняя кнопка со стрелкой. Перед загрузкой код будет еще раз проверен на ошибки и скомпилирован, в черном поле выведено, какой процент от памяти микроконтроллера заняла программа при ее загрузке в плату. Настройки классически расположены в верхнем меню.

При подключении платы к контроллеру нужно во вкладке «Инструменты» выбрать подменю «Порт» и выбрать тот, к которому подключена плата. После этого в соседнем подменю «Плата» выбрать используемое устройство. Программа настроена и готова к работе.

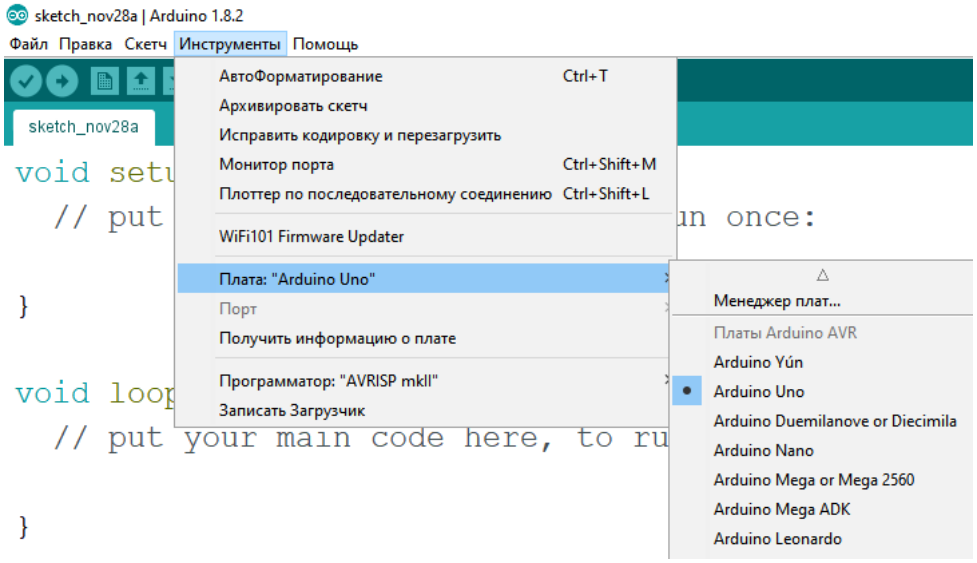


Рисунок 8 - Настройка программы для выбранной платы

Управляющий код имеет несложную структуру, состоящую из трех блоков:

- Подключение библиотек, объявление функций, процедур и глобальных переменных, введение констант. По сути, это подготовительный этап, в котором программист объявляет все, что может пригодиться при выполнении программы.
- Действия, которые выполнятся единожды. Программы Arduino, в отличие от обычных программ на языке Си, обладают важной особенностью – они содержат раздел, в котором указанные операторы будут выполняться как и в Си, единожды до завершения, а так же раздел, все действия в котором будут повторяться безостановочно на протяжении всего времени работы микроконтроллера, пока у него не будет отключено питание, либо не будет загружена другая управляющая программа;

- Действия, которые будут зациклены. Это то, о чем говорилось в предыдущем пункте.

Рассмотрим структуру на классическом примере мигания встроенным в плату светодиодом.

```
1  void setup() {
2    pinMode(13, OUTPUT);     // Инициализируем цифровой вход/выход в режиме
3 //выхода.
4 }
5
6 void loop() {
7    digitalWrite(13, HIGH);    // зажигаем светодиод
8    delay(1000);               // ждем секунду
9    digitalWrite(13, LOW);     // выключаем светодиод
10   delay(1000);               // ждем секунду
11 }
```

Переменные в программе отсутствуют, никаких библиотек не используется. Это нормально, код сразу начинается с выполнения разовых действий. Такие действия обязательно заключаются в функцию setup, синтаксис которой описан в первой строке. Тело функции, то есть, все действия, выполняемые в ней, находятся в фигурных скобках. В данном случае с помощью оператора pinMode мы задаем режим работы 13 цифрового порта на выход, потому что собираемся управлять диодом с контроллера, а не наоборот.

В шестой строке объявляется третья часть программы, действия в которой будут зациклены. Она обозначается как loop, синтаксис у нее такой же, как и у setup. Важно заметить, что часть с объявлением переменных и подключением библиотек пропустить можно, а вот функции setup и loop должны быть обязательно. Если внутри них нет никаких действий, то между фигурными скобками просто не будет никакого текста. Без них программа не скомпилируется.

Обратим внимание на подаваемые в этой программе команды. В целом любую информацию о том или ином операторе или функции можно посмотреть во внутренней справке языка. Для этого нужно зайти во вкладку «Помощь» верхнего меню и выбрать пункт «Справочник».

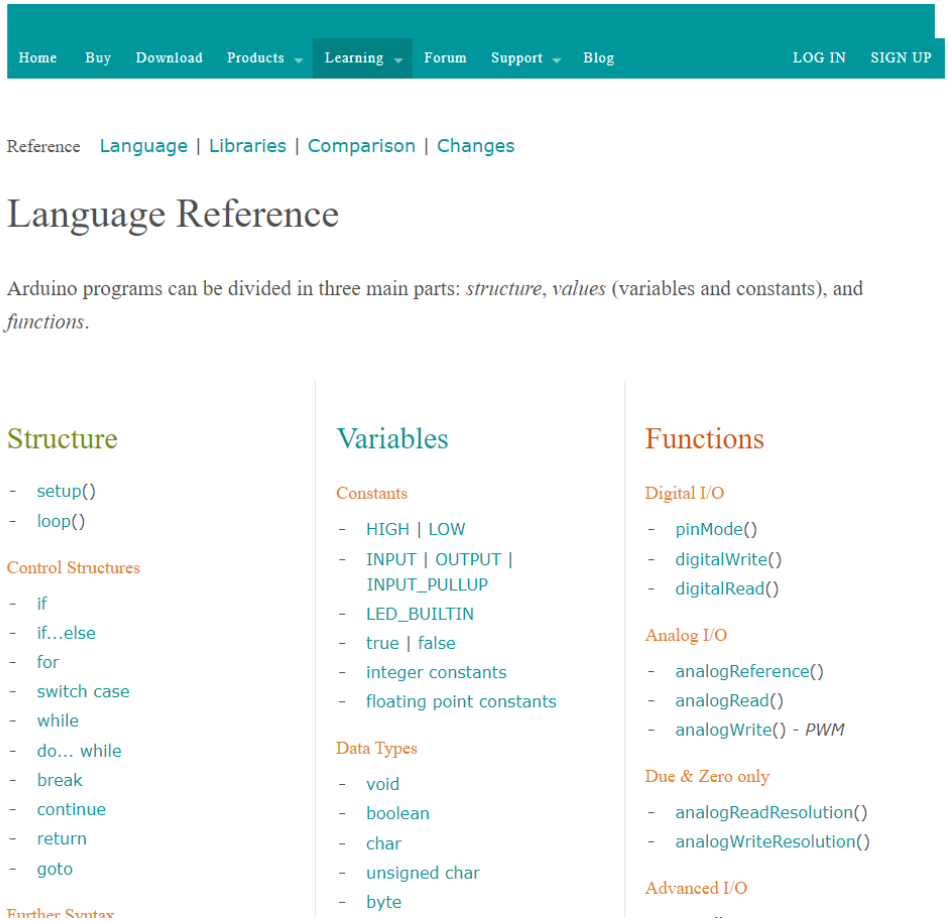


Рисунок 9- Интерфейс справочника языка

На этой странице можно найти много прикладной информации касательно программирования контроллеров – описания типов данных, условного оператора, циклов, специфических команд управления цифровыми и аналоговыми входами.

Одни из самых важных и часто используемых команд будут разобраны далее.

pinMode(номер входа, режим OUTPUT/INPUT) – оператор, устанавливающий режим работы выбранного цифрового разъема.

digitalWrite(номер входа, HIGH/LOW или 1/0) – подает на указанный вход высокий или низкий логический уровень. Чтобы зажечь диод, например, достаточно подать на разъем, к которому он подключен, единицу.

digitalRead(номер входа) – считывает состояние указанного цифрового входа. Возвращает 0 или 1.

analogWrite(номер входа, значение от 0 до 1023) – работает по аналогии с digitalWrite, только диапазон подаваемых значений изменен из-за подключения этих портов к аналого-цифровому преобразователю.

analogRead(номер аналогового входа) – аналогична команде для цифровых разъемов. Возвращает число в диапазоне от 0 до 1023.

delay(время в миллисекундах) – полностью приостанавливает выполнение программы на указанное время. Существует вариант команды в микросекундах.

Рано или поздно возникает потребность передавать данные с платы на компьютер, и для этого нужно задействовать последовательный интерфейс (UART). Делается это следующим образом: в разделе setup объявляется команда Serial.begin(скорость передачи данных в бит/с). Скорость можно выбрать из нескольких опций в зависимости от ситуации. В стандартных случаях используется значение 9600. Сам вывод данных осуществляется командой Serial.print с учетом правил форматирования.

Для того, чтобы посмотреть результат вывода, на компьютере в подменю «Инструменты» нужно выбрать опцию «Монитор порта», либо воспользоваться комбинацией горячих клавиш.

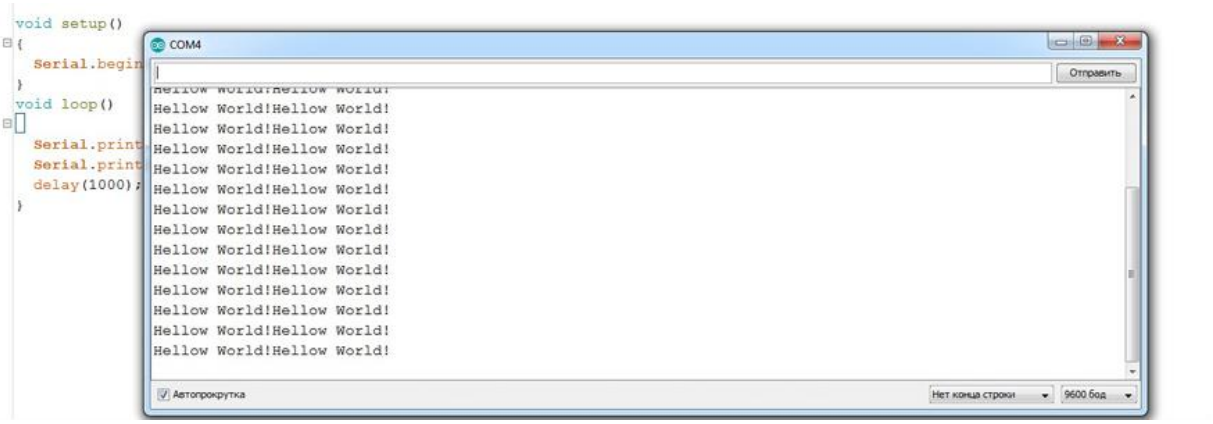


Рисунок 10 - Окно монитора порта с выводимыми данными

## Примеры робототехнических программ

В этом разделе ознакомимся с примерами использования Arduino в качестве управляющего устройства тех или иных робототехнических систем.

Программирование двухколесной платформы

В разрезе робототехники программирование любого движения занимает важное место в работе. Рассмотрим подключение и управление двумя двигателями при помощи контроллера. Для управления моторами в данном случае используется специальная плата-расширение, к клеммам которой подсоединяются движки. Обращаясь к конкретным клеммам, можно регулировать их работу.

```
1  // Моторы подключаются к клеммам M1+,M1-,M2+,M2- платы расширения
2  // Motor shield использует четыре контакта 6,5,7,4 для управления моторами
3
4  #define SPEED_LEFT      6
5  #define SPEED_RIGHT     5
6  #define DIR_LEFT        7
7  #define DIR_RIGHT       4
8
9  void go(int speed, bool reverseLeft, bool reverseRight, int duration)
10 {
11     // Для регулировки скорости `speed` может принимать значения от 0 до 255,
12     // чем больше, тем быстрее.
13     analogWrite(SPEED_LEFT, speed);
14     analogWrite(SPEED_RIGHT, speed);
```

```
15      digitalWrite(DIR_LEFT, reverseLeft ? LOW : HIGH);
16      digitalWrite(DIR_RIGHT, reverseRight ? LOW : HIGH);
17      delay(duration);
18  }
19
20  void setup()
21  {
22      // Настраивает выводы платы 4,5,6,7 на вывод сигналов
23      for(int i = 4; i <= 7; i++)
24          pinMode(i, OUTPUT);
25  }
26
27  void loop()
28  {
29      // Задержка 5 секунд после включения питания
30      delay(5000);
31      // Секунда движения вперед
32      go(150, false, false, 1000);
33      // Разворот на 180 градусов
34      go(125, true, false, 1350);
35      // Секунда движения вперед
36      go(150, false, false, 1000);
37      // Разворот на 180 градусов в другую сторону
38      go(125, false, true, 1300);
39      // Движение назад
40      go(100, true, true, 1500);
41      // Остановка
42      go(0, false, false, 0);
43      while (true)
44          ;
45  }
```

В первой части программы записью #define описаны константы, соответствующие номерам входов, к которым подключены контакты платы расширения. Это сделано для удобства и последующего обращения к нужным контактам не цифрой, а более простым для усвоения и отслеживания текстом.

С 9 строки начинается описание функции, которая в этой программе отвечает за само передвижение двухколесной оси. Void означает, что функция не должна возвращать никакое значение, go – это ее имя. В скобках указаны

аргументы, которые будут в нее передаваться с указанием требуемого типа данных. Числовыми значениями передается скорость и длительность движения, логические переменные нужны для настройки движения вправо или влево.

В командах, расположенных в теле функции, в качестве параметров служат значения, которые были переданы в качестве ее аргументов. С помощью analogWrite на порту устанавливается определенный уровень, связанный со скоростью движения (чем больше напряжения подается, тем выше скорость, прямая пропорциональная зависимость).

digitalWrite в данном случае нужен для «разрешения» левому или правому движку двигаться. В качестве передаваемого в нее уровня в данном случае служит результат работы условного оператора, записанного в кратком виде. Записи, приведенные ниже, аналогичны.

```
reverseLeft ? LOW : HIGH
if (reverseLeft == LOW) reverseLeft = HIGH;
```

Если в функцию передается значение false, то соответствующий двигатель будет работать; если значение истинное, движение будет запрещено – таким образом осуществляется поворот или разворот в определенную сторону.

В разделе setup с помощью цикла for, который нам уже встречался, описана процедура установления режима работы портов с четвертого по седьмой на выход.

Loop содержит выполняемую последовательность функций go с разными аргументами, они все образуют схему движения такой колесной пары. В конце стоит цикл while(true). Он вечный, поэтому действия в разделе выполнятся единожды – программа не войдет во вторую итерацию и мини-робот проедет прописанный путь только один раз.

Цикл while(true) может вызывать вопросы. Дело в том, что в качестве логического условия в нем фигурирует выражение типа (1==1), а тело цикла

пустое. Равенство числа самому себе всегда истинно, поэтому цикл будет бесконечным и остановит дальнейшее выполнение программы.

Такой способ управления двигателями на самом деле не очень удобен, потому что каждый маршрут приходится программировать и загружать вручную, а чтобы узнать, какое расстояние устройство будет проходить, например, за секунду времени, нужно проводить измерения. В таком ключе часто реализуют удаленное управление, например, при помощи радиопередатчика. Рассмотрим подобный пример.

Для реализации радиосвязи на Arduino требуется два микроконтроллера – один в качестве приемника, другой – передатчика. Задача в данном случае – считать значение с переменного резистора, преобразовать его, передать на другую плату и пропорционально полученному значению повернуть сервомотор. Помимо резистора и сервопривода, естественно, используется комплект радиомодулей.

Программа для передатчика в данном случае:

```
#include <VirtualWire.h> // Библиотека передатчика

void setup()
{
    // Запуск передатчика
    vw_set_ptt_inverted(true);
    vw_setup(1000); // биты в секунду
}

void loop()
{
    // чтение показаний с переменного резистора
    int sensorValue = analogRead(A0);

    // отправка значения
    send(sensorValue);
}

```

```
19 void send(int param)
20 {
21     // конвертация int в массив из 2 байт
22     uint8_t msg[2];
23     int len = 2;
24     msg[0] = highByte(param);
25     msg[1] = lowByte(param);
26 
27     // функция отправки из библиотеки
28     vw_send(msg, len);
29     // ожидание полной отправки
30     vw_wait_tx();
31 }
```

В первой строке описано подключение библиотеки, содержащей необходимые инструменты и команды для работы с радиопередачей. Используя эти команды, в setup запускается передатчик и устанавливается скорость передачи данных (шестая и седьмая строки). В разделе loop, строка 13, считывается значение на переменном резисторе и функцией send отправляется по радиоканалу.

Функция send не относится к функциям, описанным в библиотеке, ее описание дано после кода самой программы – с точки зрения синтаксиса это допустимо. Описание функции может идти либо в первом разделе, либо в конце всей программы.

Рассмотрим send. Она не возвращает никакое значение, имеет один целочисленный аргумент – то самое число, которое будет отправлено.

Основная ее задача – конвертировать число в байтовый формат, допустимый для отправки, отправить в радиоканал и дождаться отправки. Конвертация происходит путем преобразования числа в массив из двух значений – старшего и младшего байта числа, которые могут быть отправлены по радиоканалу и по получении преобразованы обратно в исходное число.

Программа для приемника может выглядеть следующим образом:

```
#include <VirtualWire.h>

// Библиотека для сервомотора, не конфликтующая с wire
#include <ServoTimer2.h>

// Создаем объект сервомотора
ServoTimer2 myservo;

void setup()
{
    // Запуск приемника
    vw_set_ptt_inverted(true);
    vw_setup(1000); // бит в секунду
    vw_rx_start(); // запуск приемника

    // объявляем сервопривод на 6 входе
    myservo.attach(6);
}

void loop()
{
    uint8_t msg[2];
    uint8_t len = 2;

    if (vw_get_message(msg, &len)) {
        // переводим байты в int
        int value = word(msg[0], msg[1]);

        // подгоняем под диапазон входных данных сервопривода
        int sValue = map(value, 0, 1023, 600, 2400);
        myservo.write(sValue); //сервопривод двигается
    }
}
```

В ней так же подключается библиотека для работы с радиопередатчиком, но к ней добавлена еще одна, она нужна для программирования сервомотора – для управления им недостаточно просто подать питание в нужном диапазоне. Эта библиотека несколько отличается от стандартной servo.h, но она не вступает в конфликт с virtualWire.

В 7 строке содержится объявление объекта – это стандартная практика в библиотеке, позволяющая объявить в программе что-либо, что будет подчиняться правилам этого заголовочного файла. Объекту в данном случае присваивается имя myservo.

12-14 строки содержат команды включения радиопередатчика на прием аналогично примеру с передатчиком. В 17 строке расположена команда привязки объекта сервомотора к конкретному порту, к которому он подключен физически.

В разделе loop создается массив, чтобы в него записать полученные от передатчика значения в том же формате, в котором они были присланы. Задается его длина – 2 элемента. Следующий за объявлением массива условный оператор нужен для того, чтобы при получении информации принять ее и обработать.

Используется обратная функция перевода числа в байтах в обычное, десятичное, с помощью функции map можно масштабировать исходный диапазон в удобный для передачи в сервомотор. У нее 5 аргументов – число, которое нужно перевести, исходный диапазон изменения числа и конечный, который понадобится в дальнейшем.

Команда write, обращенная к сервоприводу (31 строка) вызывает поворот сервопривода на угол, определенный введенным числом.

Дополнительно рассмотрим пример из немного иной категории – подключение к плате дисплея и вывод на него данных с какого-либо датчика, например, дальномера. Именно он выбран не случайно – его часто используют в робототехнических системах для создания системы ориентации устройства в пространстве.

Схема такова: к Arduino UNO подключается по протоколу I2C (встречается чаще на данный момент) двустрочный дисплей LCS1602 и ультразвуковой дальномер HC-SR04 на цифровых входах. Программа несложная:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);  // Создание объекта дисплея
int echoPin = 2; // Первый вход дальномера
int trigPin = 3; // Второй вход дальномера

void setup() {
    pinMode(trigPin, OUTPUT); //Отсюда отправляем импульс
    pinMode(echoPin, INPUT);  //Здесь принимаем импульс
    lcd.init(); // Инициализация дисплея в системе
    lcd.backlight();// Включаем подсветку дисплея
  }

void loop() {
   lcd.setCursor(0, 1); //Ставим курсор в начало
    int duration, cm;
    digitalWrite(trigPin, LOW); //Выключаем источник импульса
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);  //Включаем источник импульса
    delayMicroseconds(10);  // Посылаем импульсы в течение 10 сек
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH); //Считаем время полета импульса
    cm = duration / 58; //Делим на скорость звука и пополам
    lcd.println(cm);  // Выводим расстояние до объекта
    delay(100);
}
```

Для работы с дисплеем подключаются сразу две библиотеки, одна отвечает за возможность общения по протоколу I2C, вторая – за управление конкретным дисплеем. Далее в программе создается объект с именем lcd. В скобках задается его адрес в системе, количество элементов в строке и количество строк, соответственно, в нашем случае 16 элементов в 2 строках.

Для подключения дальномера используется два цифровых порта. Происходит это вот почему: дальномер содержит в себе два модуля – излучатель и приемник, которыми можно управлять отдельно. Подавая напряжение на излучатель, мы провоцируем подачу ультразвуковых импульсов. Они проходят какое-то расстояние, отражаясь от препятствия, попадают на приемник. Если засечь время, которое пройдет с момента отправления

импульса до его возвращения, зная скорость звука в воздухе, можно без особого труда рассчитать пройденное импульсом расстояние.

В разделе setup снова задаются режимы работы цифровых выходов. Тот, который работает на подачу импульса, выставлен на выход, приемник – на вход. В 10 и 11 строках включается дисплей и его подсветка.

В повторяющейся части программы мы каждый раз ставим курсор в начало дисплея, вводим переменные, которые понадобятся для расчета расстояния. В строках 17-22 описан процесс отправления и приема ультразвуковых импульсов дальномером.

Функция pulseIn предназначена для получения интервала времени, за который импульс долетит до препятствия, отразится от него и вернется обратно. В первом аргументе указан вход, за которым мы «наблюдаем», во втором – значение, которое мы ждем. Время будет засечено с момента подачи импульса до того, как на входе приемника появится высокий уровень, то есть, в приемник попадет импульс. Пересчитываем длительность в расстояние, выводим его на дисплей, между измерениями задаем задержку в сто миллисекунд.

**Справочные данные**

Как уже говорилось ранее, Arduino это достаточно популярная платформа, которая обросла большим количеством справочной литературы, готовых проектов и вспомогательного ПО. Полезные для работы программы были упомянуты в начале при разговоре о вспомогательных инструментах для изучения робототехники.

В итоге помочь в изучении электроники в робототехнике могут:

- Внутренняя справка среды Arduino IDE;
- Справка языка программирования Си;
- Tinkercad;

- Ardublock;
- Fritizing;

Оказать посильное содействие может большое количество готовых проектов и методических пособий по электронике в разрезе платформы Arduino. Существуют тематические наборы, касающиеся определенных тем: сборки мобильных роботов, медицинской физики (ЭКГ, ЭЭГ и пр.), радиоэлектронные комплектующие, наборы для «Умного дома».

## 2.3. Микроконтроллер ATmega

Микроконтроллер Arduino – отличный вариант для того, чтобы начать обучение, однако, он подходит далеко не всегда. Рано или поздно возникает необходимость использовать более гибкие и более сложные управляющие устройства. В этом разделе рассматривается еще один микроконтроллер, на базе которого, собственно, и создан Arduino, но уже как самостоятельное устройство.

Сердцем многих Arduino-устройств выступает микроконтроллер ATmega 328PU. Напомним, что микроконтроллер — микросхема, предназначенная для управления электронными устройствами. Типичное исполнение сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это однокристальный компьютер, способный выполнять относительно простые задачи. Он отличается от микропроцессора интегрированными в микросхему устройствами ввода-вывода, таймерами и другими периферийными устройствами.

Логично, что для работы контроллера на него нужно подать питание, причем пинов (входов) питания достаточно много. Это связано с топологией чипа на уровне производства. Каждый пин держит токовую нагрузку, поэтому

микроконтроллер должен быть запитан равномерно со всех сторон. С точки зрения принципиальных схем, подвод питания к контроллеру выглядит следующим образом:

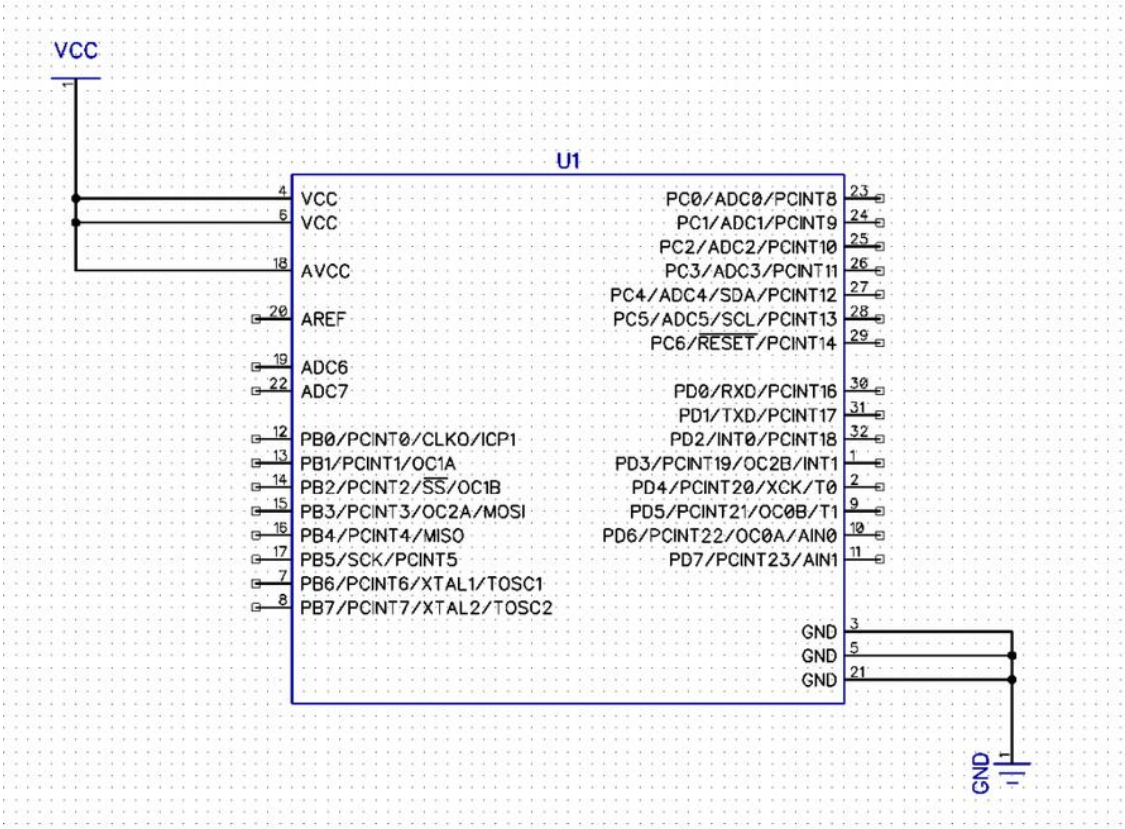


Рисунок 11 - Принципиальная схема питания ATmega328PU

Продолжим напрашивающуюся аналогию с Arduino. Этот контроллер имеет AREF-вход, на который подается опорное напряжение аналого-цифрового преобразователя, как правило 5В. Если этого достаточно, пин «подтягивается» к питанию конденсатором. Второй конденсатор, подключаемый на питание, нужен для того, чтобы стабилизировать напряжение. С учетом этих поправок, получаем такую схему:

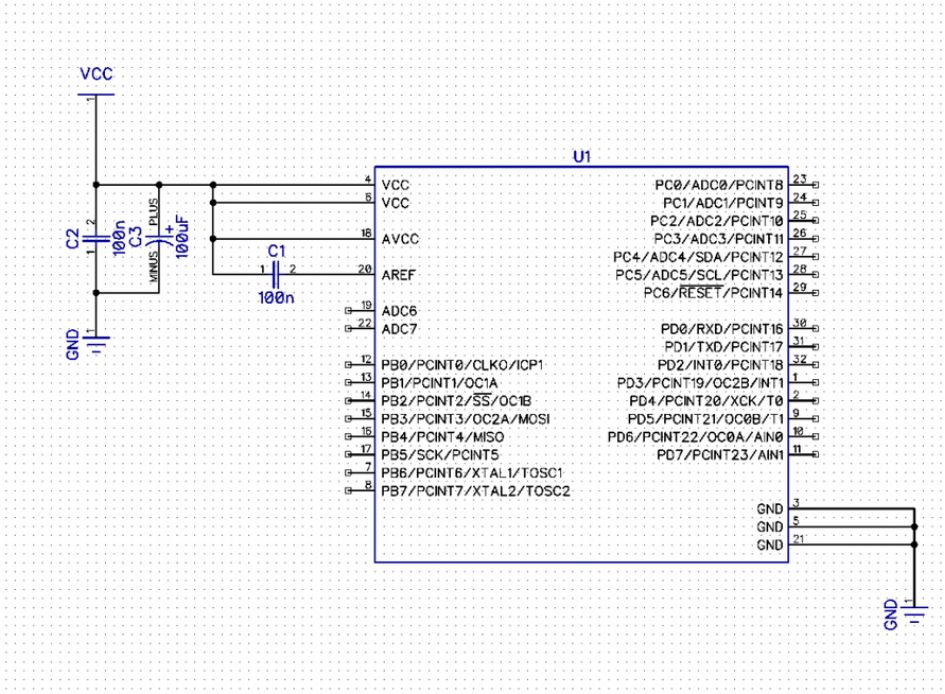


Рисунок 12 - Подвод питания к контроллеру с учетом конденсаторов

Перезагрузка и сброс Arduino обычно производится кнопкой RESET. На самом деле это еще один выход контроллера, с помощью которого в случае использования чистого ATmega можно настроить периферийные устройства, хоть и с ограничениями. Он «подтянут» внутренним резистором сопротивлением 10 кОм. С учетом этих портов, схема снова усложняется:

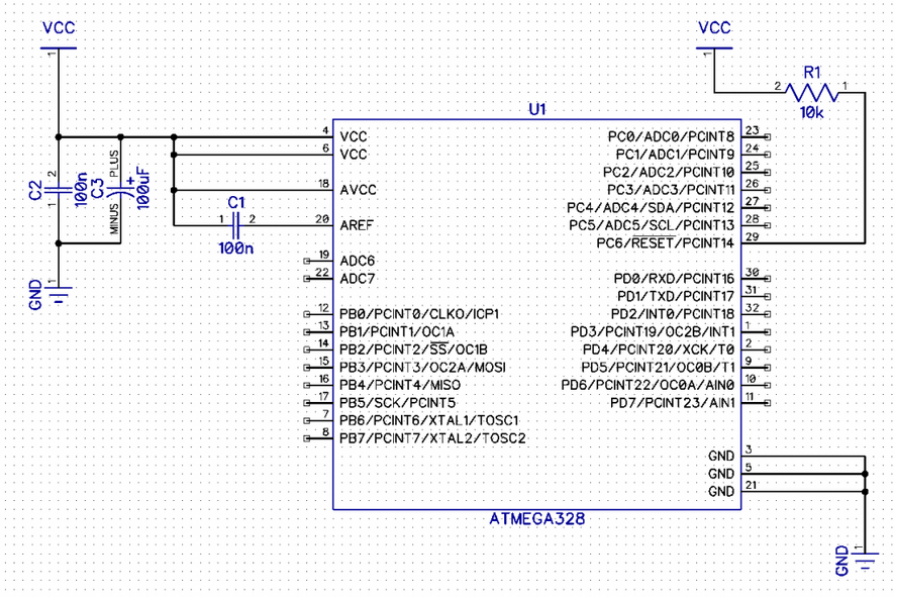


Рисунок 113 - Принципиальная схема с учетом портов RESET

Введем новое понятие. Тактирование — тактовый сигнал, использующийся для согласования операций одной или более цифровых схем. Тактовая частота микроконтроллера — это количество тактов в секунду, которые выполняет микроконтроллер. Обычно Arduino работает на частоте 16 МГц от внешнего кварцевого резонатора. Для того, чтобы и другой контроллер работал на той же самой частоте, резонатор нужно подключить к микроконтроллеру вместе с конденсатором на 22 пФ:

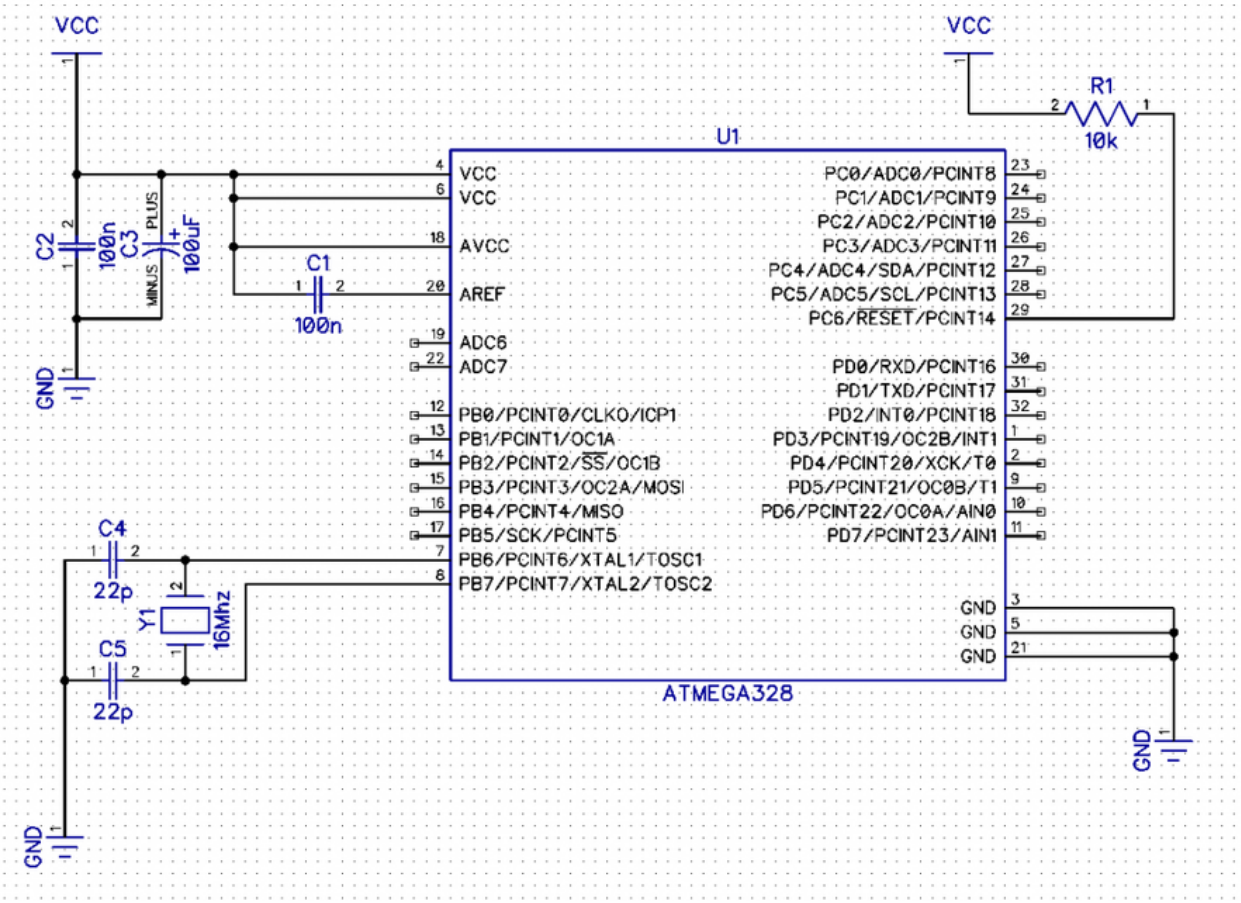


Рисунок 14 - Принципиальная схема с учетом кварцевого резонатора

Если необходимо сэкономить место на плате, то кварцевый резонатор вместе с соответствующим конденсатором устанавливать не обязательно. Можно тактироваться от встроенной RC цепи с частотой в 8МГц.

Все необходимые для работы микроконтроллера компоненты рассмотрены, поэтому можно переходить к его программированию. Для этого есть

несколько вариантов, например, можно использовать специальный программатор USB ASP. Схема подключения представлена ниже:


Рисунок 15 - Подключение программатора USB ASP

Подключение производится по шине SPI, для этого существует 6 контактов: MISO, MOSI, SCK, RST, GND, 5V. Уточним расположение на принципиальной схеме:

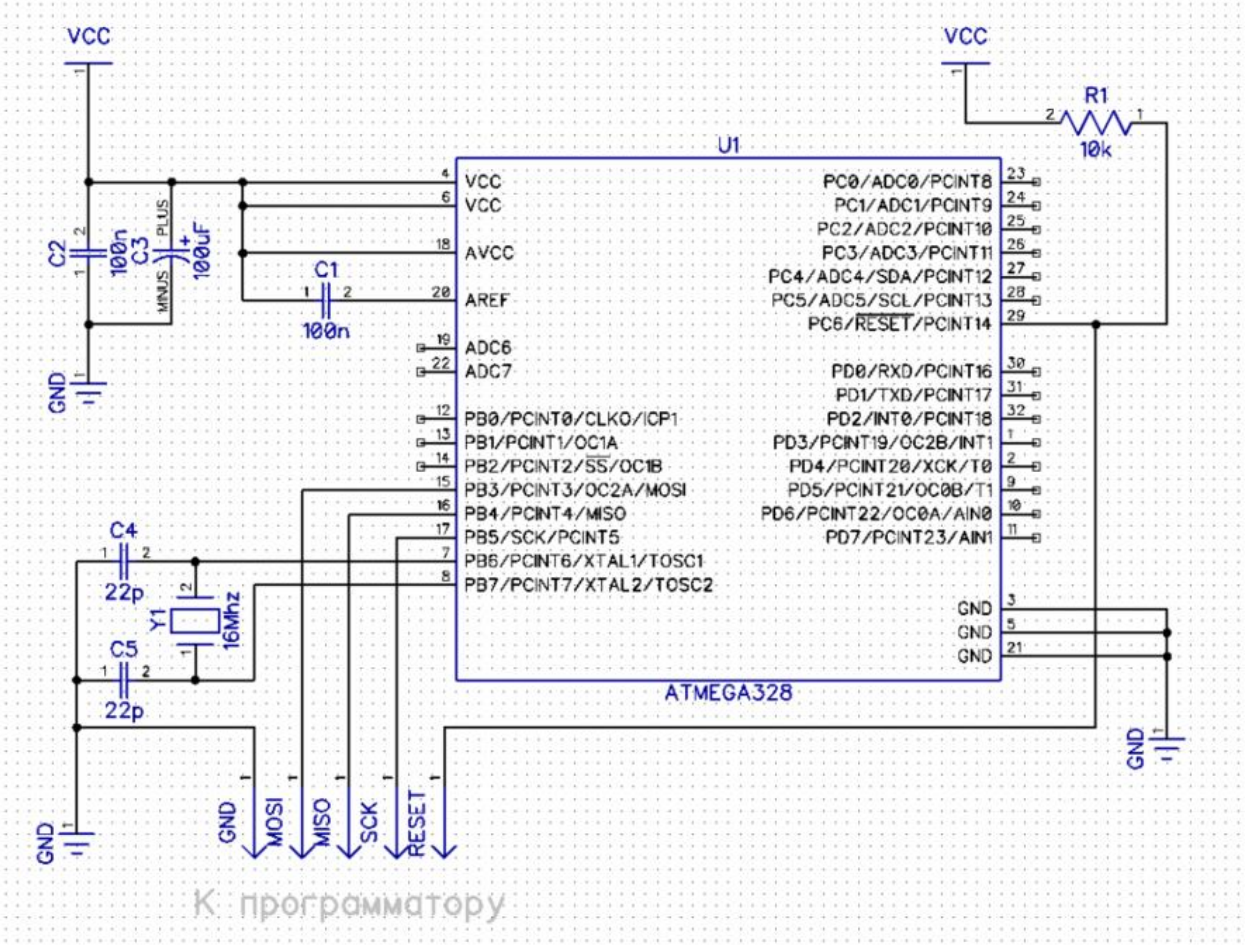

Рисунок 16 - Принципиальная схема с учетом USB-ASP

Подключенный программатор позволяет начать прошивку контроллера. В качестве замены программатору можно рассмотреть и обыкновенную плату Arduino. Принцип прошивки через программатор достаточно простой, состоит из нескольких этапов. Во-первых, нужно подключить программатор к контроллеру, установить необходимое обеспечение для работы с ним. С помощью Arduino IDE выбрать нужный программатор в списке, микроконтроллер и частоту. Далее программу можно просто загрузить в плату, как при работе с Arduino.

Работа без программатора чуть сложнее. Перед использованием плату Arduino нужно проверить, уточнив номера входов MOSI, MISO, SCK. Если их нумерация совпадает с нумерацией, приведенной на схеме ниже, плату можно использовать. Если имеются отличия, лучше выбрать для этих целей другую плату или программатор.

| pin name: | платы на основе mega168 или 328: | mega(1280 and 2560) |
|---|---|---|
| // MOSI: | 11: | 51 |
| // MISO: | 12: | 50 |
| // SCK: | 13: | 52 |

Рисунок 17 - Схема соответствия для использования Arduino в качестве программатора

Плату нужно подготовить для работы в качестве программатора. Подключив ее к компьютеру, нужно открыть Arduino IDE, зайти в меню «Файл», подменю «Образцы» и выбрать «ArduinoISP». Код загрузить в плату.


Рисунок 18 - Программирование Arduino для работы в качестве программатора

Следующий шаг – подключение прошиваемого микроконтроллера к плате. Выходы интерфейса SPI контроллера подключаются к одноименным выводам Arduino, выход RESET необходимо соединить к 10 выходу платы (53 для плат на основе ATmega1280 и 2560).

Прежде чем продолжить, введем еще одно определение. Загрузчик – небольшая программа, прошитая в памяти микроконтроллера, которая позволяет загружать в него код без внешних аппаратных средств. Он активизируется на несколько секунд после сброса устройства. Его наличие дает возможность прошивать микроконтроллер напрямую через последовательный порт. Для записи загрузчика нужно подключить Arduino с микроконтроллером к компьютеру, зайти в меню «Инструменты» и выбрать опцию «Записать Загрузчик».

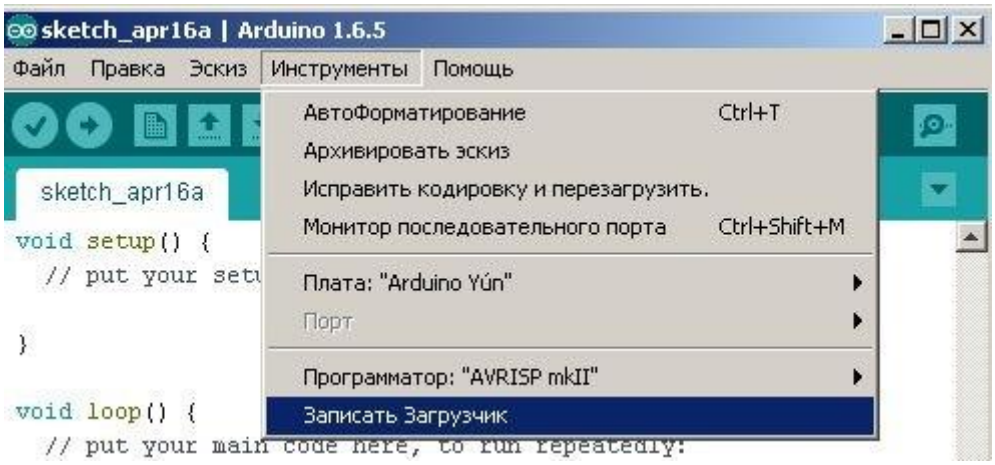


Рисунок 19 - Запись загрузчика в плату

На этом подготовка к использованию контроллера Atmega328 закончена. Дальнейшие действия совершенно такие же, как при использовании Arduino.

## 2.4. Микроконтроллеры семейства STM32

Продолжим рассмотрение работы с микроконтроллерами, обратившись к семейству STM32. Оно объединяет большую серию 32-битных микроконтроллеров производства компании STMicroelectronics. Чипы STM32 группируются в серии, в рамках каждой из которых используется одно и то же 32-битное ядро ARM, например, Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0+ или Cortex-M0. Каждый микроконтроллер состоит из ядра процессора, статической RAM-памяти, флеш-памяти, отладочного и различных периферийных интерфейсов.

Микроконтроллеры этого типа обладают большими возможностями в сравнении с Arduino, поэтому используются в более комплексных и требовательных задачах. Этим объясняется их частое использование в робототехнике.

Для ознакомления с этой программной платформой понадобится несколько элементов. Порог вхождения тут значительно выше, чем у Arduino.

Во-первых, это отладочная плата – макетная плата, на которой уже установлены основные элементы схемы, например, отвечающие за питание или основные разъемы передачи данных. Помимо этого, на отладочной плате отводят специальное место для установки электронных компонентов, микросхем или модулей. Отладочные платы существенно разнятся в зависимости от выполняемых ими задач.

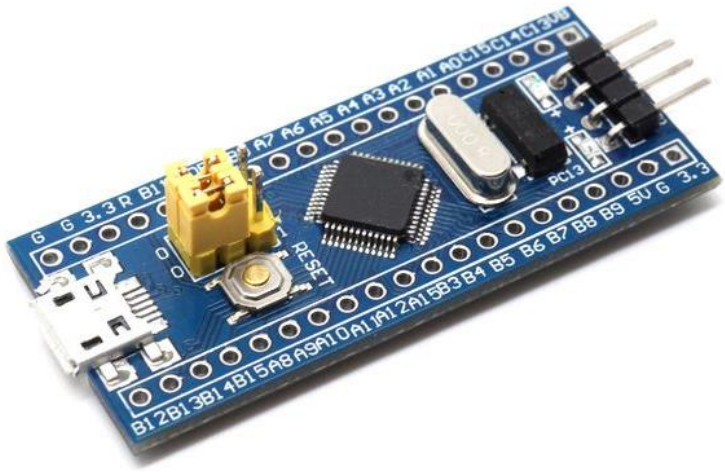


Рисунок 20 - Отладочная плата без встроенного программатора

Программатор – устройство, выполняющее функцию трансфера программного кода с компьютера на плату, это – ещё одно необходимое дополнение для работы с STM32.


Рисунок 21 - Программатор

С его помощью можно осуществлять откладку контроллера и обмен данными с компьютером. От Arduino эта система отличается уже тем, что в учебной плате программатор был сразу встроен в нее.

Рисунок 22 - Отладочная плата со встроенным программатором

С сайта официального производителя скачивается специальное ПО для работы с STM32. Чтобы сформировать программный код для загрузки, понадобится Cube MX. Arduino IDE использовать не получится, потому что STM32-контроллеры обладают возможностью гибкой настройки функции

входов, чего у Arduino не наблюдается. Cube MX предусматривает работу с этой особенностью.

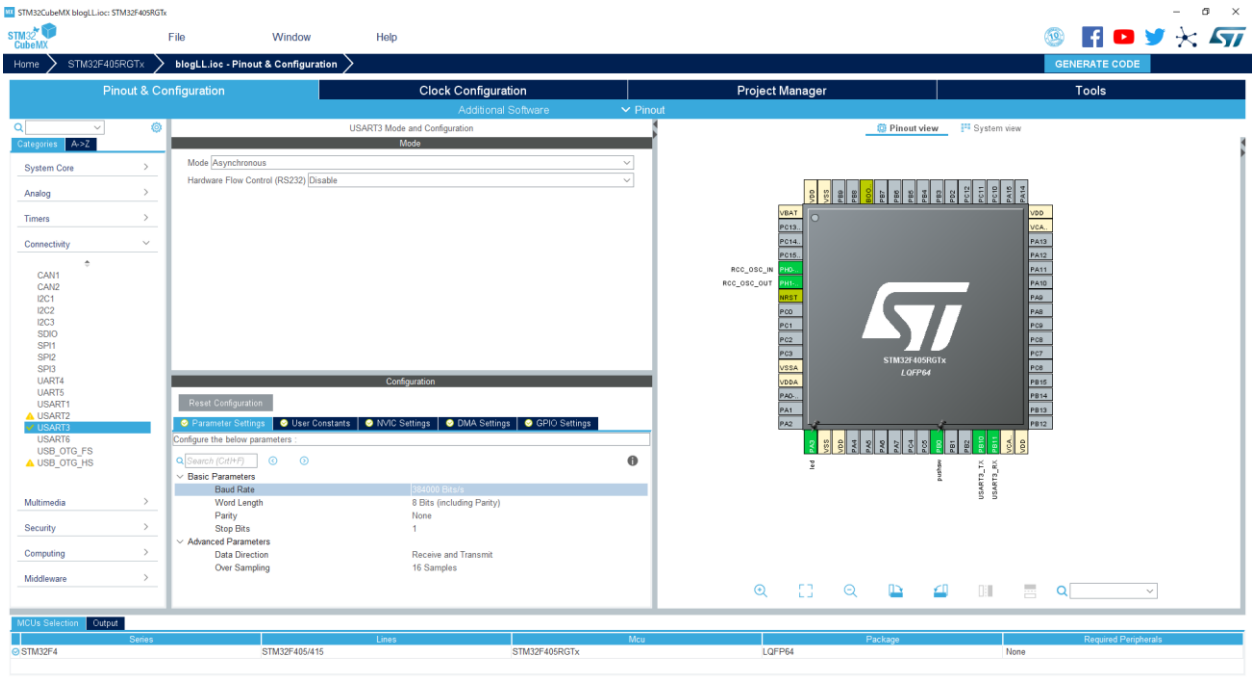


Рисунок 23 - Интерфейс программы CubeMX

Для написания программы пользователю необходимо освоить ещё одну программу – Cube IDE, которая напоминает Arduino IDE.

Написание программ осуществляется, тем не менее, сходным образом с Arduino. Изучим на примере моргания светодиодом.

Чтобы настроить работу единственного входа, куда будет подключаться диод, нужно воспользоваться Cube MX – определить вход, который будет использован, указать его в программе. Помимо этого, необходимо указать, что отладка будет осуществляться с помощью компьютера.

Следующим шагом станет формирование файла с информацией о режиме работы входов и передача его в Cube IDE. Останется добавить в сформированный автоматически код строки, отвечающие за мигание светодиодом и загрузить итоговый вариант в плату.

```
1 while(1)
2     {
```

```
3         GPIO_SetBits(GPIOC, GPIO_Pin_13);
4         Delay(2000000);
5         GPIO_ResetBits(GPIOC, GPIO_Pin_13);
6         Delay(2000000);
7     }
```

На этом и заканчивается краткое описание алгоритма работы с контроллерами семейства STM32.

# 3. Схемотехника

Приступим к освоению другого, не менее важного раздела, необходимого для комплексного изучения робототехники. Знания в схемотехнике нужны для того, чтобы понимать принцип работы электронных компонентов, основные области и случаи их применения. Все это нужно для создания собственных электронных схем и для совершенствования разрабатываемых систем.

Данный модуль не ставит своей целью подробно изучить основы науки, он нужен для того, чтобы предоставить необходимую для практической работы теорию и некоторые советы по ее грамотному применению.

## 3.1. Основные понятия

В основном работа ведется с цепями постоянного тока. Напомним основные физические величины, которыми оперирует электроника и схемотехника.

Заряд q [Кл] — это физическая скалярная величина, определяющая способность тел быть источником электромагнитных полей и принимать участие в электромагнитном взаимодействии.

Сила тока I [А] — отношение протекшего заряда q [Кл] ко времени t [с].

Напряжение U [В] — скалярная физическая величина, значение которой равно работе эффективного (включающего сторонние заряды) электрического поля, совершаемой при переносе единичного пробного электрического заряда из одной точки в другую.

Сопротивление R [Ом] — физическая величина, которая показывает свойство проводника противодействовать прохождению через него электрического тока.

Закон Ома для участка цепи связывает три этих параметра.

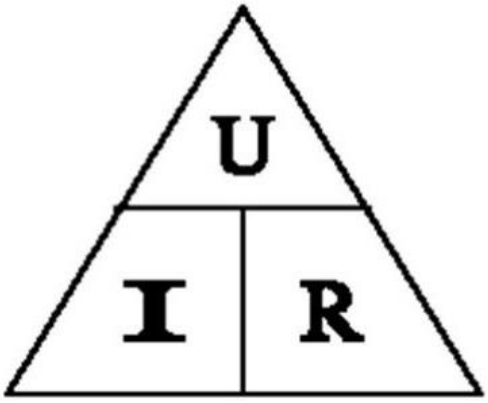


Рисунок 24 - Треугольник величин закона Ома

Помимо силы тока, сопротивления и напряжения, напомним про емкость, заряд и индуктивность.

Электрическая емкость C [Ф] — величина, характеризующая способность тела накапливать электрический заряд.

Индуктивность L [Гн] — это физическая величина, характеризующая магнитные свойства электрической цепи. В некоторых источниках её называют коэффициентом самоиндукции, так как она зависит от текущего в замкнутом контуре тока и создаваемого им магнитного потока.

## 3.2. Базовые компоненты

Перейдем к изучению базовых электронных компонентов. Рассматривать будем по группам. Деление производится исходя из принципов применения и строения.

Сначала рассмотрим блок, включающий в себя всевозможные переключатели и кнопки.

Кнопка — компонент, в основе того лежит принцип размыкания и замыкания электрической цепи при ее нажатии. Это обеспечивается за счет наличия двух несоединенных между собой проводников. При нажатии на кнопку третий проводник замыкает цепь с первыми двумя. После нажатия кнопки он сам поднимается с помощью пружины в исходное положение.

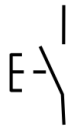

Рисунок 25 - Условное обозначение кнопки

Выключатель отличается от кнопки главным образом тем, что при одном нажатии фиксирует свое положение, а не возвращается в исходное. Замыкание происходит в результате сдвига проводника, замыкающего эти контакты.

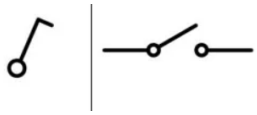

Рисунок 26 - Условное обозначение выключателя

Переключатель может находиться сразу в трех положениях. Работает так же, как выключатель, но имеет больше неподвижных контактов, и свободный проводник может замыкать поочередно обе группы, либо находиться в разомкнутом положении.

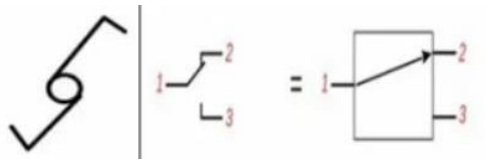


Рисунок 27 - Условное обозначение переключателя

Перейдем к рассмотрению резистивных компонентов. Их главная задача – контроль электрического тока, протекающего в цепи. К резистивным компонентам относятся и датчики, сопротивление которых каким-либо образом зависит от параметров окружающей среды.

Начнем с базы — резистора. Это элемент электрической цепи, имеющий определенную постоянную величину сопротивления, предназначенный для регулировки параметров цепи согласно законам Ома.

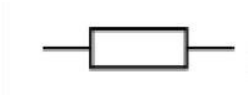

Рисунок 28 - Условное обозначение резистора

Потенциометр – компонент на базе резистора, который позволяет менять напряжение на участке цепи из-за изменения параметров внутреннего проводника. Определения достаточно для успешной работы с ним, но для полного понимания существует много упрощенных аналогий.

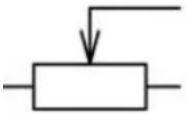

Рисунок 29 - Условное обозначение потенциометра

К этой же категории компонентов относится и реостат, который по сути является составной частью потенциометра. Он имеет два контакта в отличие от потенциометра, и сопротивление между ними меняется при передвижении ползунка. Фоторезистор меняет сопротивление в зависимости от яркости падающего на чувствительный элемент света, термистор, соответственно, от температуры.

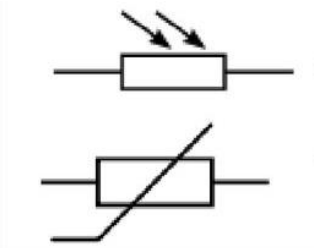

Рисунок 30 - Условное обозначение фоторезистора (сверху) и термистора (снизу)

Разберем методы световой индикации. Раньше чаще всего использовались простые лампы накаливания. Внутри корпуса лампы расположена нить накала, при пропускании тока через которую она раскаляется и начинает излучать свет. Полярности такой источник освещения не имеет.


Рисунок 31 - Условное обозначение лампы накаливания

На сегодняшний день гораздо чаще используются светодиоды. Светодиод – полупроводниковый элемент, в котором при прохождении электрического тока возникает электромагнитное излучение видимого спектра. Он обладает полярностью, то есть, порядок подключения проводов питания и земли в данном случае важен. Как правило, у диода разные длины контактов, более длинный подключается к «плюсу». Для ограничения величины тока, который будет протекать через светодиод (это обусловлено его строением и параметрами) необходимо рассчитать необходимое сопротивление ограничивающего резистора и подключить его.

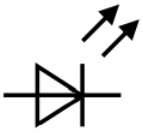

Рисунок 32 - Условное обозначение светодиода

RGB-светодиод это популярная разновидность светодиода, небольшое устройство, которое объединяет в себе три диода базовых цветов: синий, красный и зеленый. Управляя ими, можно получать различные оттенки цветов. Подавая высокий уровень на все управляющие контакты, получаем белый.

Светодиод имеет 4 контакта – по одному для подвода питания и управления каждым из базовых цветов и общий контакт для «земли».

Еще один индикаторный элемент – семисегментный индикатор. Его достаточно удобно использовать для вывода информации в числовом формате: показания датчиков, время, количество итераций и так далее.

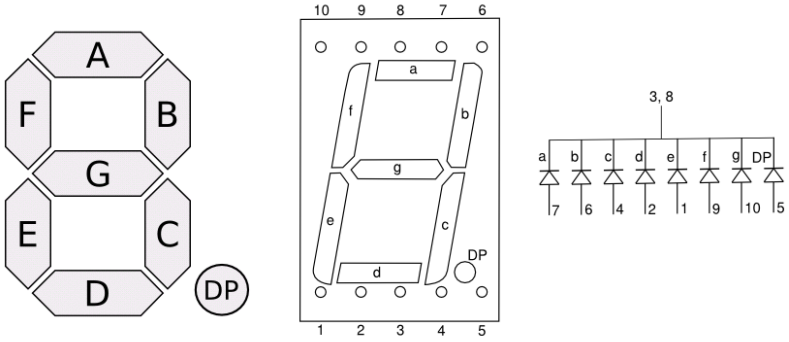


Рисунок 33 - Семисегментный индикатор

Компонент состоит из 8 отделенных друг от друга областей, они используются для поэтапного отображения цифр, дополнительно к ним идет значок точки. Каждый сегмент – это отдельный диод, который можно программировать самостоятельно и подключать независимо. Компонент имеет десять выходов, как и с RGB-светодиодом – общую землю и отдельные подводы питания для каждого из сегментов. Отличие в том, что выходов для «земли» два: по одному на четыре контакта питания.

Помимо перечисленного набора элементов существует большое количество других вариантов индикации, но, тем не менее, основные рассмотрены.

Продолжим пошаговое рассмотрение основных компонентов транзистором. Транзистор – электронный ключ, позволяющий с помощью подачи небольших токов на один контакт из трех разрешать протекание и управлять более большими токами на остальных. Такое устройство нужно для работы с мощными светодиодами, которые управляются с помощью микроконтроллера, например. Контакты транзистора называются базой, коллектором и эмиттером. При подаче малого по величине управляющего

сигнала на контакт базы между коллектором и эмиттером начинает протекать ток.

Транзистор, как базовое устройство, может быть модифицирован в зависимости от специфики использования.

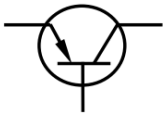

Рисунок 34 - Условное обозначение транзистора

Конденсатор – устройство, способное накапливать электрический заряд и мгновенно его отдавать. Используется для построения цепей с частотно-зависимыми свойствами: фильтров, цепей обратной связи, колебательных контуров и проч. Возможность быстрого разряда конденсатора позволяет получить импульс большой мощности, который применяется в лазерной технике, например. Кроме того, в электронных устройствах помимо функции хранения электроэнергии конденсаторы используют как элементы памяти.

В промышленной электротехнике конденсаторы используются для компенсации реактивной мощности и в фильтрах высших гармоник.

Как датчики малых перемещений: малое изменение расстояния между обкладками очень заметно сказывается на ёмкости конденсатора.

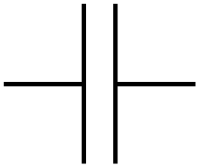

Рисунок 35 - Условное обозначение конденсатора

На этом перечисление основных компонентов электрических цепей можно считать завершенным. Рассмотрим самые важные связки, которые постоянно встречаются в робототехнике.

### 3.3. Часто используемые схемы

Схемы с резистором

У резистора гораздо больше применений, чем одно, но основное, конечно, – ограничение тока в цепи. Самый яркий пример – подключение перед входом светодиода, чтобы ограничить проходящий ток и не повредить элемент.

Подтягивающий резистор – так называют схему, в которой резистор включен между проводником и питанием цепи. Такое подключение нужно, чтобы, допустим, гарантировать на логическом входе, с которым соединен проводник, высокий уровень в следующих случаях:

- Проводник не подключен к логическому выходу;
- Разомкнут ключевой элемент на присоединённом логическом выходе, который устроен как открытый вывод ключевого элемента.

В случае нахождения между проводником и «землей» говорят, что резистор стягивающий. Он нужен для обеспечения низкого уровня на логическом входе в изложенных выше случаях.

Любой логический вход имеет ёмкость относительно земли. Если сигнал формируется на открытом выводе ключевого элемента, то чем выше сопротивление подтягивающего резистора, тем больше время нарастания или спада сигнала при размыкании ключевого элемента. Если подтяжка к питанию, то надо учитывать время нарастания сигнала. Если подтяжка к земле, то время спада. Время спада или нарастания — это время между размыканием ключа и достижением сигнала порогового напряжения.

Пороговое напряжение — это напряжение, при достижении которого логическим входом фиксируется изменение логического состояния.

При проектировании логических схем приходится рассчитывать сопротивление подтягивающего резистора, при этом известны ёмкость входа и пороговое напряжение. Время спада или нарастания пропорционально сопротивлению подтягивающего резистора, то есть, например, при увеличении сопротивления вдвое время спада или нарастания так же увеличится вдвое.

Следующая частая схема совместной работы устройств – матричное подключение. Это могут быть матрицы светодиодов или любая клавиатура. Под таким соединением принято понимать расположение компонентов в порядке строк и столбцов, где, например, ряд элементов имеет один набор общих контактов, а столбец – другой. Этот способ позволяет существенно сократить необходимое количество контактов для подключения многоэлементных систем. На рисунке ниже представлено устройство матричной клавиатуры:

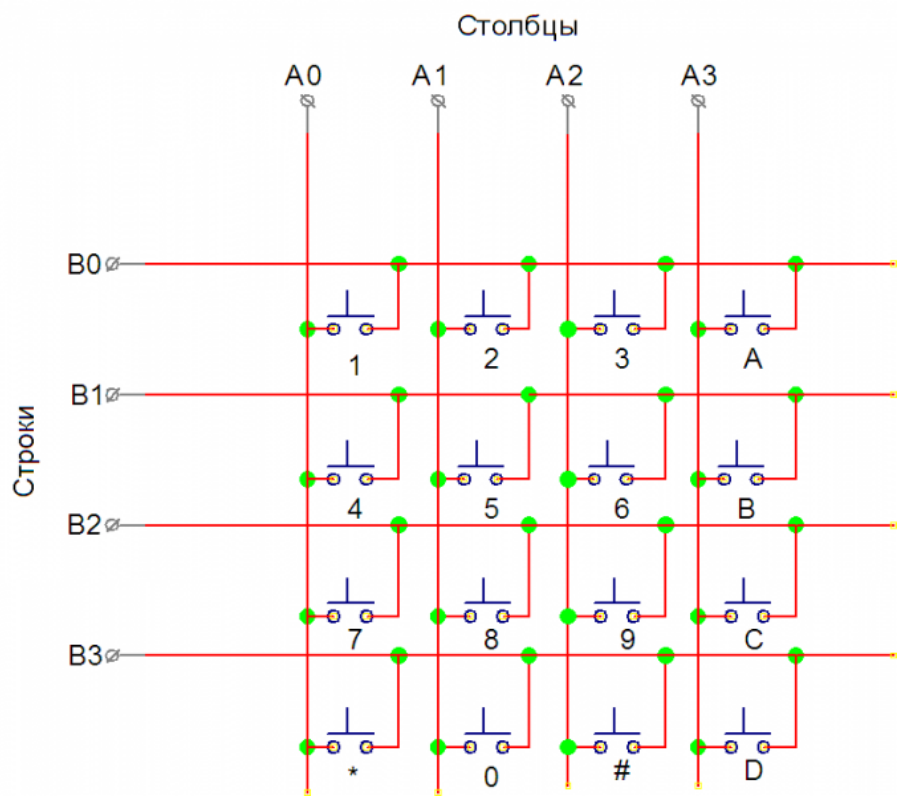


Рисунок 36 - Схема подключения матричной клавиатуры

### 3.4. Методики изготовления электронных плат

Рассмотрим процесс изготовления собственных электронных плат. В основе каждого устройства лежит принципиальная схема, в которой указано расположение элементов на будущей плате. Для того, чтобы создать ее, необходимо иметь представление о функции будущей платы и о количестве различных элементов, которые будут ее составлять.

Работу со встраиваемыми логическими схемами, контроллерами, модулями памяти и так далее можно упросить использованием технической документации, к ним прилагающейся. Как правило, эта литература содержит достаточное количество информации с описанием входов, рекомендуемыми параметрами тока и напряжения.

Составление принципиальных схем могут упростить некоторые программы, например, EasyEDA. После переноса схемы соединения всех компонентов в виде принципиальной схемы в программу можно перейти к разработке схемы самой платы. Чаще всего в программах по разработке электроники в одном проекте находятся сразу две схемы, одна принципиальная (в виде условных обозначений), вторая — в виде самих компонентов, которые будут расположены на плате. После создания принципиальной схемы необходимо будет перейти ко второму виду отображения платы, то есть. в виде самих компонентов, которые будут изображены на плате. Далее в этом отображении компоненты необходимо расставить так, как они будут расположены на плате. После этого нужно произвести трассировку, то есть провести дорожки между всеми необходимыми контактами компонентов. Затем можно будет сохранять плату в различных форматах для производства.

Существует несколько методик изготовления электронных плат. С некоторыми из них познакомимся далее. Выбранные способы отличаются относительной простотой и распространенностью использования.

Начнем с того, что для создания платы нужны готовые гербер-файлы, то есть изображения дорожек, контактов для подключения, шелкографии. Создание принципиальной схемы и ее перенос в нужный формат, рассматривать не будем. Начнем с предположения, что плата уже разведена и сохранена в нужном формате.

Для начала познакомимся с основными способами изготовления плат. Это ЛУТ (лазерно-утюжная технология), фоторезист, фрезеровка и заказ на производстве. Самый простой вариант — это заказать изготовление готовой платы по вашим гербер-файлам. Он требует только выбора параметров платы, заказа и нескольких дней-недель ожидания. Для заказа можно использовать JCB PCB, либо подобные онлайн-сервисы. Перейдём к наиболее популярным способам создания плат в домашних условиях.

**ЛУТ**

Этот способ популярен из-за своей простоты и возможности сделать платы достаточно хорошо качества. Для создания платы не требуется специфических знаний, оборудования, инструментов и так далее. Всё, что понадобится, — это:

- Кусочек стеклотекстолита с медным покрытием (материал, из которого будет сделана основа платы);
- Среда для травления. Как правило, это хлорное железо, но бывают и другие специфические варианты;
- Напечатанная на фотобумаге, либо на журнальной бумаге заготовка платы. Печать должна быть осуществлена на лазерном принтере, иначе ничего не получится;
- Немного ручного инструмента, утюг.

На заранее обезжиренную поверхность текстолита, вырезанного по размерам платы, необходимо с помощью утюга перевести напечатанный шаблон, после чего под струей холодной воды аккуратно удалить бумагу,

чтобы на текстолите осталась краска в форме дорожек платы. Если получились какие-либо пропуски в дорожках, необходимо либо всё сделать заново, либо (в случае, если пропусков не так уж и много) перекрыть их с помощью специального маркера.

После этого плату помещаем в раствор для травления. Медь будет постепенно уходить с платы. При этом под слоем краски она останется. Как только вся медь, которая находится не под краской, будет удалена, необходимо достать плату и тщательно промыть под струей воды, чтобы удалить все остатки раствора для травления.

На следующем этапе с помощью растворителя удаляем краску с платы. Теперь можно просверлить в ней отверстия. Далее нужно припаять все контакты и залудить дорожки. После этого можно покрыть плату лаком для защиты.

**Фоторезист**

Метод фоторезиста немного похож на ЛУТ, но при этом есть и существенные отличия. Для этого метода потребуется:

- Фоторезист пленочный негатив;
- Прозрачная пленка для струйного или лазерного принтера;
- Принтер (для соответствующей пленки);
- Фольгированный стеклотекстолит;
- Ультрафиолетовая лампа;
- Кальцинированная сода (пищевая не подойдет).

Для начала разберемся с тем, что такое фоторезист пленочный негатив. Пленочный негативный фоторезист представляет собой полимерный светочувствительный материал, покрытый с обеих сторон тонкой защитной пленкой (схема изображена на рисунке 37). Воздействие света на него либо разрушает полимер (позитивный фоторезист), либо, наоборот, вызывает его полимеризацию и понижает его растворимость в специальном растворителе

(негативный фоторезист). При последующей обработке происходит травление в «окнах», образованных засвеченными (позитивный фоторезист) или не засвеченными (негативный фоторезист) участками полимера.


Рисунок 37 - Фоторезист

Теперь перейдем к самому процессу изготовления. Первым делом необходимо распечатать фотошаблон для фоторезиста. Для этого открываем заранее разработанный файл платы, выбираем печать негатива (желательно в настройках указать повышенный расход краски) и печатаем на кусочке пленки, проверяем на просвет. Если просвечивает, то увеличиваем расход краски и печатаем заново, либо печатаем поверх уже напечатанной заготовки.

Фотошаблон нужно оставить до высыхания и в это время подготовить основу для изготовления платы. Для этого отрезается кусок стеклотекстолита нужного размера и обезжиривается его поверхность, после чего отрезается соответствующий размерам кусок фоторезиста. Важно хранить фоторезист в недоступном для света месте, иначе он станет непригоден для использования.

К плате нужно приклеить фоторезист, для этого убрать матовую пленку и пригладить, чтобы не было пузырей и складок. Так же важно не трогать руками часть, с которой уже убрали пленку. После этого уже высохший шаблон прикладывается сверху на фоторезист, выравнивается и прижимается куском стекла.

После этого плата ставится под ультрафиолетовую лампу на расстоянии 10-15 см над стеклом и остается в таком состоянии под включенной лампой на 7

минут. Далее заготовка достается и вторая защитная пленка убирается (аккуратно, чтобы не удалить слой самого фоторезиста). После этого фоторезист проявляется, для чего нужно опустить плату в раствор кальцинированной соды на 30 секунд.

Далее следует промывка проточной водой и сушка. После чего, как и в случае с ЛУТ, травление в растворе  хлорного железа. После этого плата еще раз отмывается, сверлятся отверстия и впаиваются компоненты. На этом изготовление платы можно считать законченным.

# 4. Моделирование

Проектирование любого робототехнического устройства включает в себя задачу 3D-моделирования его составных частей. Помимо моделирования, часто возникает необходимость показать результаты работы в графическом формате – для этого используются рабочие и сборочные чертежи, остальная конструкторская документация. Цель данного раздела – разобрать основные принципы работы автоматизированных систем проектирования, закономерности единой системы конструкторской документации и донести способы создания моделей устройств, которые могут пригодиться.

## 4.1. Введение в САПР – основные методы и операции проектирования

Системы автоматизированного проектирования – специальный класс ПО, существующий для конструирования комплектующих, деталей, изделий и конструкций. Все они используют принципы 3D-моделирования и принципы оформления конструкторской документации в том или ином виде.

К основным составным частям САПР можно отнести следующие модули:

- Система трехмерного проектирования. Позволяет создавать трехмерные модели различными способами, проектировать сборочные единицы, содержащие в том числе и стандартные конструктивные элементы, как правило, находящиеся в базе САПР. Существует очень много упрощающих моделирование функций, часть из которых будет рассмотрена;
- Чертежно-графический редактор нужен для автоматизации проектно-конструкторских работ, облегчения создания и выпуска графической и текстовой документации;

- Модуль проектирования спецификаций часто нужен на предприятиях. С его помощью оформляются многие сопроводительные документы конструкторских проектов;
- Текстовый редактор позволяет разработать стандартизированную документацию.

Применение САПР и основы работы с конструкторской документацией необходимы в робототехнической работе, когда речь идет о создании какого-либо устройства. Его можно сначала спроектировать в САПР, перевести в вид чертежей, по ним, если нужно, изготовить и собрать физический прототип.

На самом деле, все чертежи, разрабатываемые на любом отечеством предприятии, должны соответствовать нормам ЕСКД – единой системы конструкторской документации, набору нормативных документов, полностью регулирующих оформление, содержание и назначение различных чертежей.

Далее работа в САПР будет рассматриваться на примере программы КОМПАС-3D – российского продукта, обладающего полным перечнем необходимых для выполнения различных конструкторских задач инструментов.

Программа позволяет вести работу с несколькими типами документов. К ним относятся детали (неделимые единицы изделия), сборки (модули из нескольких деталей), чертежи (графические интерпретации деталей, сборочных единиц, изделий), текстовые документы, фрагменты.

Рассмотрим алгоритм создания 3D-моделей. В КОМПАС-3D любая модель представляет собой совокупность геометрических объектов: различных тел, пространственных точек и кривых, в основе которых могут лежать эскизы. Геометрические объекты строятся с помощью примитивов – стандартных моделей вершин, ребер и граней.

Модели всегда можно поделить на два основных типа – твердотельные, представленные телами и обладающими ненулевой массой и поверхностные, представленные поверхностями и не обладающие массой. Кроме того, возможны их сочетания.

Для построения модели и ее частей часто пользуются эскизами. Эскизы – объекты трехмерного моделирования, созданные с помощью чертежно-графического редактора. Могут располагаться как на координатной или вспомогательной плоскостях, так и на плоских гранях. Применяются для задания формы сечения тела или поверхности, траектории перемещения сечений, положения элементов массива.

В КОМПАС-3D эскизы строятся в специальном режиме эскизов. Как только эскиз построен, с помощью специальных операций, свойственных большинству САПР, можно преобразовать его в тело.

Для построения тел используется основной набор команд:

- Выдавливание – тело образовано путем перемещения сечения вдоль прямолинейной траектории на заданное расстояние;
- Вращение – тело образовано поворотом сечения вокруг оси на заданный угол;
- По траектории – тело образовано путем перемещения сечения вдоль произвольной траектории;
- По сечениям – соединение нескольких сечений;
- Листовое тело – особый тип команд для образования листовых тел.

**Выдавливание**

Тело, полученное с помощью выдавливания, может быть как самостоятельным, так и объединено с другими телами в более сложное. Операцию можно применять к граням, эскизам, ребрам или пространственными кривым. Как правило, при подборе элементов

выдавливания, возможен выбор между сплошным (твердотельным) и тонкостенным результирующим элементом.

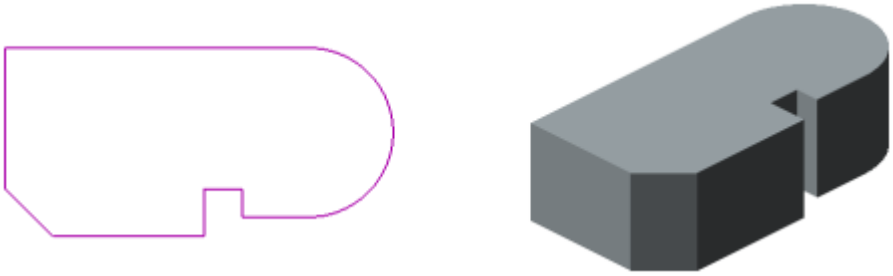

Рисунок 38 - Эскиз и элемент, полученный операцией его выдавливания

В зависимости от настроек, если сечение представляет собой плоскую грань, контур, построенный по эскизу или на плоской грани и выталкивается в противоположном самом себе направлении, то возможен уклон боковых граней элемента.

При выдавливании пространственной кривой или ребра возможно создание исключительно тонкостенного элемента.

Рисунок 39 - Результат выдавливания пространственной кривой

**Вращение**

При использовании операции вращения необходимо задать не только эскиз, который будет вращаться, но и ось, вокруг которой это будет происходить. Вращать можно грани, эскизы, ребра или пространственные кривые.

Рисунок 40 - Эскиз, ось, элемент, образованный операцией вращения

Также можно выбрать, какой элемент должен получиться – сплошной или тонкостенный. При использовании разомкнутого сечения, а также при вращении ребра и пространственной кривой, возможно построение только тонкостенных элементов.

Рисунок 41 - Вращение спирального сечения

**По сечениям**

В логике этой операции тело образуется путем соединения нескольких сечений произвольных расположения и формы.

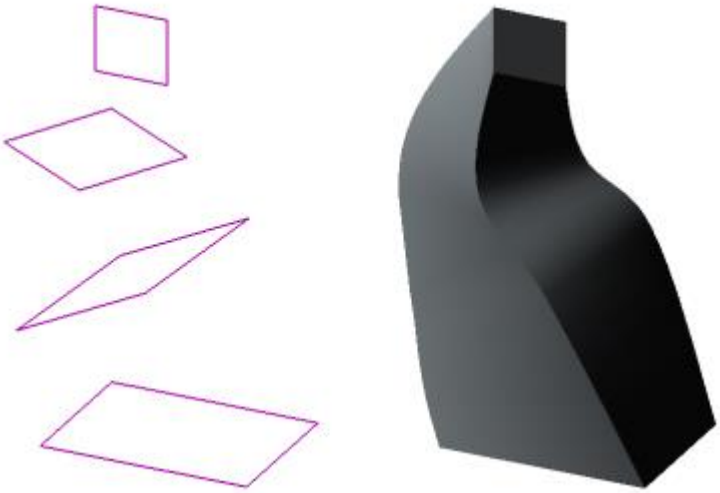

Рисунок 42 - Сечения и полученный на их основе элемент

В качестве используемых сечений могут быть использованы, как и в большинстве случаев, эскизы, контуры, грани и пространственные кривые. Крайние сечения могут быть точками.

Форма элемента может быть изменена, если задать направляющие кривые, их количество не ограничено.

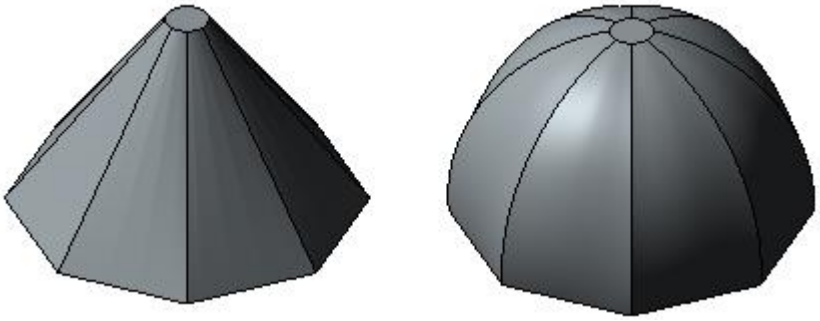

Рисунок 43 - Изменение тела заданием направляющих кривых

Построение элемента по траектории происходит следующим образом:

Рисунок 44 - Сечение, направляющая, тело, полученное выдавливанием по траектории

Тела, созданные на основе этих операций, можно комбинировать, объединять, вырезать одно из другого и так далее. Используя такое поэтапное построение модели, можно получить нужную деталь. В этом и состоит базовый принцип 3D-проектирования в САПР, и, в частности, в КОМПАС-3D.

**Создание эскизов**

Эскиз, лежащий в основе многих 3D-объектов, является основополагающим элементом моделирования. Рассмотрим закономерности его создания.

Начнем с того, что эскиз, как правило, - это комбинация большого количества геометрических элементов: прямых, окружностей, кривых, различных

многоугольников. Используя принципы построения, можно создать самый широкий спектр эскизов, а на их основе – подробные модели. Перечислим основные операции для работы с эскизами:

- Автолиния – объединение цепочек объектов в одно целое;
- Прямоугольник – построение по двум заданным вершинам, либо по центру и вершине, либо по трем вершинам, либо по центру и двум точкам;
- Отрезок – соединение двух произвольных точек плоскости. Создавать можно сразу указывая параллельность или перпендикулярность, касательную через внешнюю точку или касательный к двум кривым;
- Окружность – создание произвольной окружности разными способами (через центр и точку на окружности, по трем точкам, с центром на объекте, касательная к кривой, касательная к двум кривым, к трем кривым, окружность по двум точкам (диаметральным));
- Дуга – изображение произвольной дуги по заданным параметрам;
- Вспомогательная прямая – произвольная прямая по одной точке;
- Фаска – отрезок пересечения двух пересекающихся кривых;
- Скругление – замена угла пересечения объектов дугой;
- Точка – создание точки в любом месте плоскости;
- Сплайн по точкам – создание сплайна, проходящего через указанные вершины (сплайн – одна из форм кривых);
- Спроецировать объект – создает в текущем эскизе проекцию указанного объекта в модели. Крайне полезная функция, если часть детали уже спроектирована, и ее, например, необходимо продлить. Функция не только проецирует объекты, но и фиксирует их, поэтому дальнейшие построения можно выполнять на основе этих фиксированных фрагментов;
- Эквидистанта – построение элементов на одинаковых расстояниях от указанной оси;

- Эллипс – создание эллипса по центру и полуосям;
- Коническая кривая – построение кривой конического сечения.

В конструкторской работе линии подразумевают под собой определенные обозначения. Их нужно соблюдать при работе не только на бумаге, но и в САПР. Все параметры линий (их толщина, расстояние между штрихами) указаны в соответствующих ГОСТ по оформлению конструкторской документации. Требования оттуда перенесены в КОМПАС-3D. Рассмотрим основные стили:

- Основные линии. Используются по умолчанию и отображаются во всех видах. По контуру этих линий происходит операция выдавливания;
- Тонкие линии. Могут быть использованы для вспомогательных операций при построении. Они отображаются только в окне эскизов и не участвуют в дальнейшем построении деталей;
- Осевые линии. Ими обозначаются оси любых симметричных объектов, они используются для применения операции вращения.

Для получения твердотельной фигуры после применения операций выдавливания, вращения и пр. к эскизу, необходимо, чтобы контур эскиза был замкнут и отсутствовали самопересечения. Если это условие не будет соблюдено, получится тонкостенный элемент. Это стоит всегда учитывать при работе и возможные причины ошибок искать именно в некорректных эскизах.

**Булевые операции**

Операции такого рода нужны для арифметической работы с создаваемыми телами. Одно тело возможно вычесть из другого, или наоборот, объединить два отдельных в одно единое.

Булевых операций можно насчитать 3 вида:

- Объединение. Выбирается несколько тел, применяется операция, тела становятся одним целым. Объединить тела, которые не пересекаются, невозможно;
- Вычитание. Результатом операции является удаление одного тела из другого. В случае, если уменьшаемое и вычитаемое тела пересекаются не полностью, вычитаемое тело все равно будет удалено полностью. Поясним, уменьшаемое тело – тело, из которого будет убираться элемент. Вычитаемое – то, что будет вырезано из уменьшаемого;
- Пересечение. Операция оставляет только пересекающуюся часть двух тел, удаляя исходные;

**Массивы**

При необходимости создать ряд однотипных геометрических элементов с определенным расположением используют такой инструмент, как массивы.

При работе с массивами исходный элемент достаточно создать только единожды. Создаваемые дубликаты имеют четкую связь с оригиналом – при изменении исходника, автоматически изменения будут применены к остальным моделям.

Существуют различные алгоритмы создания массивов, касающиеся геометрического расположения элементов. Массивы бывают по сетке, по концентрической сетке, вдоль кривой, по точкам, по таблице и зеркальные.

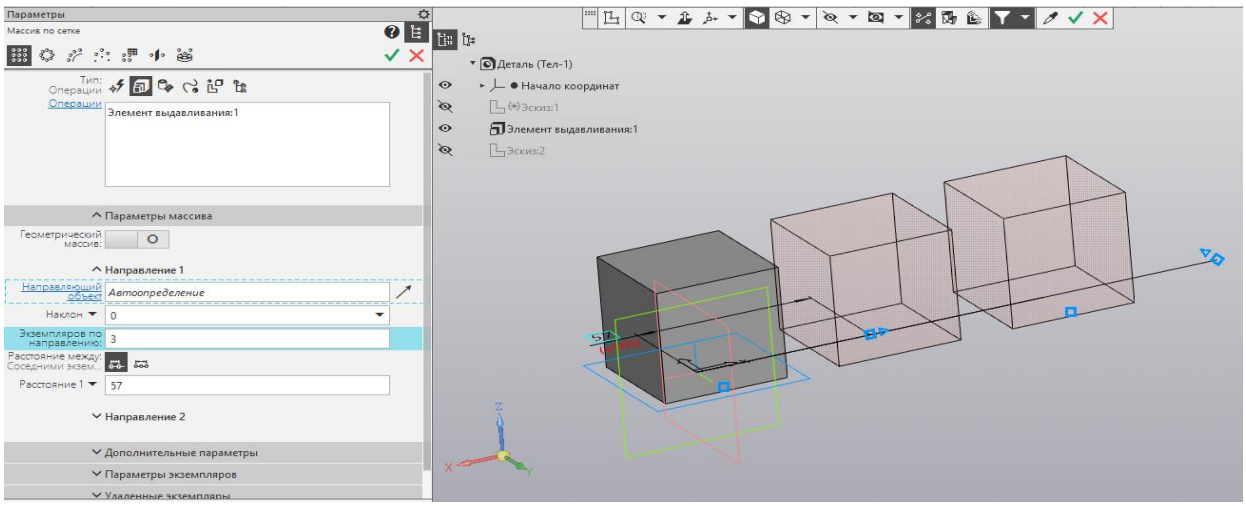


Рисунок 45 - Интерфейс создания массива элементов вдоль прямой в одном направлении в КОМПАС-3D

## 4.2. Параметрическое моделирование

Такой способ построения моделей подразумевает задание взаимных связей и ограничений, позволяющих для изменения свойств модели регулировать один-два параметра, а не каждое измерение по отдельности с перспективой нарушить изначальную геометрию тела. В этом разделе рассмотрим основные методы реализации такого способа и его преимущества.

Необходимость параметризовать модель возникает при проектировании сложных деталей в случае, если необходимо поменять, например, один эскиз из многих. Без параметризации придется переделывать все составные элементы, с ней – зависимые компоненты изменятся соответственно. Кроме того, детали, которые проектируются для дальнейшего производства, могут масштабироваться или уточняться в размерах. Без параметрических связей между элементами тела это затруднительно.

Таким образом, корректным, правильным эскизом можно назвать обладающий всеми необходимыми размерами, взаимосвязями между элементами, привязками к началу координат.

В большинстве САПР параметрическое моделирование доступно, и существуют инструменты, помогающие его реализовать. Например, на панели

эскиза можно проверить, правильно ли параметризован эскиз. О правильной параметризации говорят, когда на все возможные степени свободы наложены ограничения и взаимосвязи и указаны все необходимые размеры элемента.

В КОМПАС-3D это реализуется через механизм работы с переменными. Каждому размеру можно присвоить свою переменную и дальнейшие размеры указывать относительно первого.

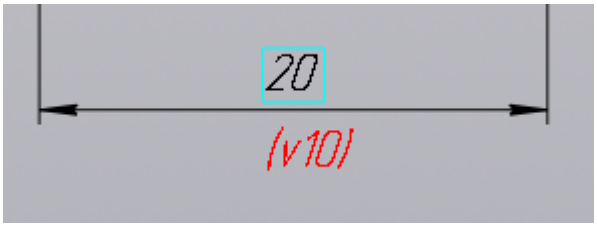


Рисунок 46 - Параметру v10 присвоено значение 20

Переменных в чертеже может быть достаточно много. Для того, чтобы посмотреть их полный список, значения и отношения между собой, если они заданы, нужно попасть в раздел «Переменные». Тут же можно поменять заданные значения переменных при необходимости.

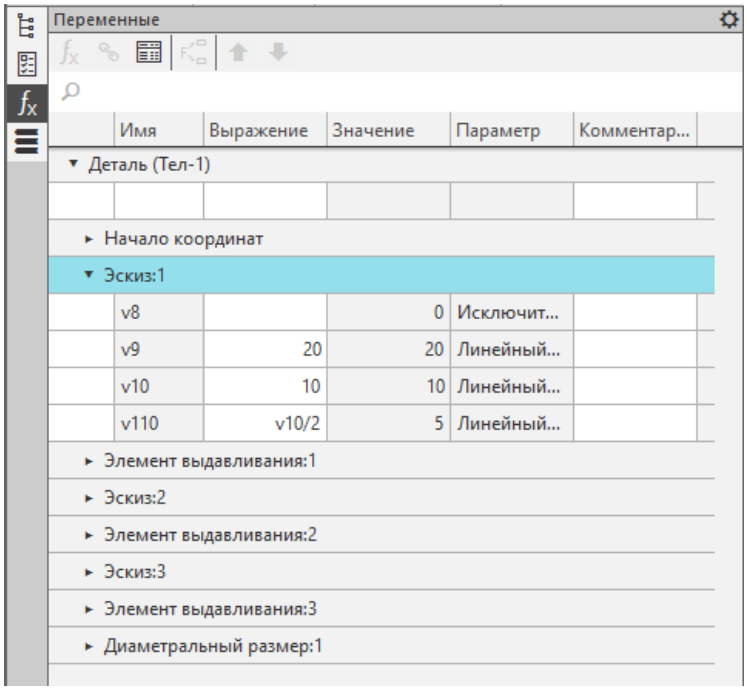


Рисунок 47 - Меню "Переменные"

В общем случае существует несколько способов реализации параметрического моделирования, которые можно встретить в различных САПР. Рассмотрим основные.

- Табличная параметризация. Параметры типовых деталей заносятся в специальную таблицу, новые объекты можно моделировать на их основе и дополнять таблицу. Возможности немного ограничены, так как тяжело задавать произвольные значения и геометрические отношения элементов. Тем не менее, такой метод значительно ускоряет работу в САПР и часто используется конструкторами при создании чертежей;
- Иерархическая параметризация (на основе истории построений). Как правило, в отдельном окне САПР находится вся последовательность построения модели в виде «дерева построения». Она включает в себя все вспомогательные элементы, эскизы, примененные к ним операции в хронологическом порядке. С помощью такого типа параметризации возможно адаптивное изменение параметров детали на основе изменения конкретного ее параметра. Используется практически во всех САПР;
- Вариационная (размерная) параметризация. Именно та, которая заключается в наложении на элементы эскизов специальных связок и ограничений (соосность, параллельность, перпендикулярность и прочее);
- Геометрическая параметризация. Вид параметрического моделирования, при котором геометрия каждого объекта, задаваемого подобным образом, может быть пересчитана автоматически в зависимости от положения родительских объектов, его параметров и переменных. Модель в случае геометрической параметризации состоит из элементов построения и элементов изображения. Первые еще называют конструкторскими линиями – они задают все параметрические связи. Ко второй группе элементов относятся все

линии изображения и оформления (размеры, надписи и штриховки). Редактирование модели таким образом наиболее гибкое – в случае необходимости корректировок и перестроений не обязательно удалять исходные построения, достаточно заменить их, и модель подстроится самостоятельно.

В некоторых САПР существует возможность загрузить эскизы изделия в растровом формате, перенести их в системные модели либо сразу построить на их основе модель, которую потом можно будет дополнить собственными элементами.

### 4.3. Основы ЕСКД

Цель этого раздела, в первую очередь, ознакомиться с основными правилами оформления конструкторской документации. Это может быть полезно в работе для корректной и грамотной демонстрации чертежей своих устройств. Далее будут рассмотрены самые основные понятия, относящиеся к этой теме.

ЕСКД – единая система конструкторской документации. Она в общем случае нужна для того, чтобы все создаваемые на различных предприятиях чертежи имели общую структуру, и работник одной отрасли мог без особого затруднения прочесть и понять чертеж другого конструктора.

Конструкторские документы нужны для полного определения состава и устройства изделия и должны содержать все необходимые для его разработки, изготовления, контроля, эксплуатации и ремонта сведения. К ним относятся графические и текстовые документы: рабочие чертежи деталей, сборочные чертежи, общие виды, схемы, спецификации и пояснительные записки.

Основные конструкторские документы – чертежи и спецификация к изделию.

Чертеж детали – это документ, содержащий изображение детали и другие данные, которые необходимы для ее изготовления и контроля. На нем указываются размеры, их предельные отклонения и шероховатости поверхностей, так же технические требования и дополнительная информация, которую конструктор счел необходимой.

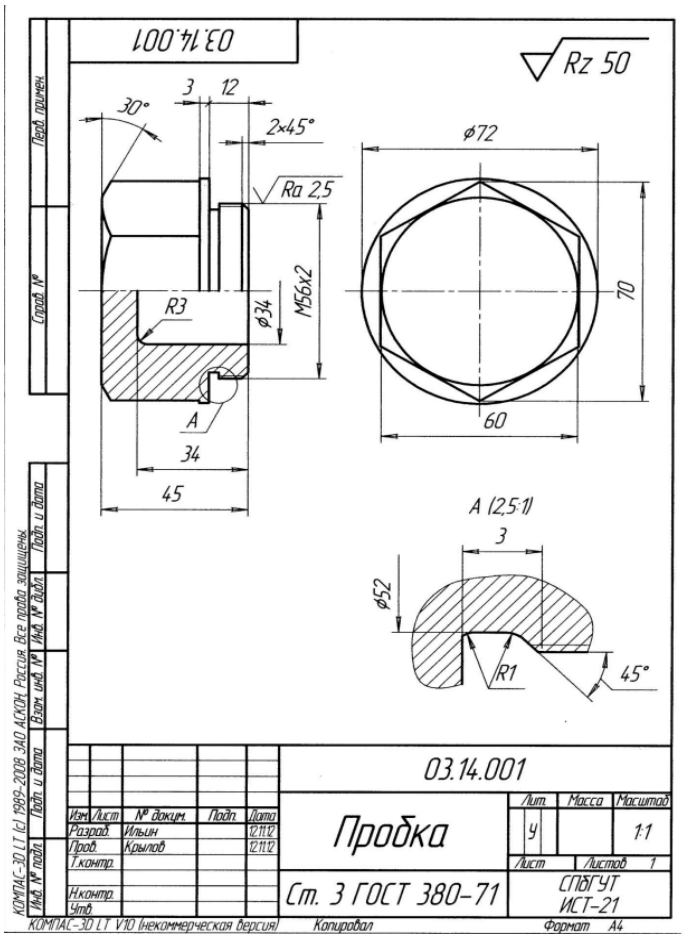


Рисунок 48 - Пример чертежа детали

Любой чертеж включает в себя рамку, и она всегда делается по строго установленному шаблону. Информация в ней – код детали в учете предприятия или учебного заведения, название детали, материал, из которого она изготавливается, масса, служебные надписи, для чертежа обязательно указание масштаба. Так же в штампе указывается конструктор чертежа, проверяющий и даты. Чертеж содержит виды детали, достаточные для представления ее функциональности, в их качестве выступают проекции на одну из трех осевых плоскостей при условии, что деталь, как правило,

центрирована относительно начала координат. Для того, чтобы показать внутренний рельеф деталей, допустимо применять сечения и разрезы. Штриховкой обозначается тело детали.

На чертеже должны быть указаны все необходимые для изготовления детали размеры, и не должно быть лишних, чтобы не возникало путаницы. Помимо размеров, указываются предельные отклонения и допуски (определяют, насколько точно необходимо выполнить производство детали), допуски формы и расположения (устанавливают ограничения, например, на соосность или на перпендикулярность составных элементов детали). Отдельно для функциональных поверхностей (такие, которые могут быть задействованы в сборке или в эксплуатации) указываются параметры шероховатости поверхности (требование, насколько она должна быть ровной). Для всех неуказанных поверхностей устанавливается общее значение шероховатости и предельных отклонений размеров, которое определяется соответствующими ГОСТами и функцией разрабатываемой детали.

Сборочный чертеж – документ, который содержит изображение сборочной единицы (изделия, состоящего из нескольких отдельных деталей) и другие данные, необходимые для ее сборки, изготовления и последующего контроля.

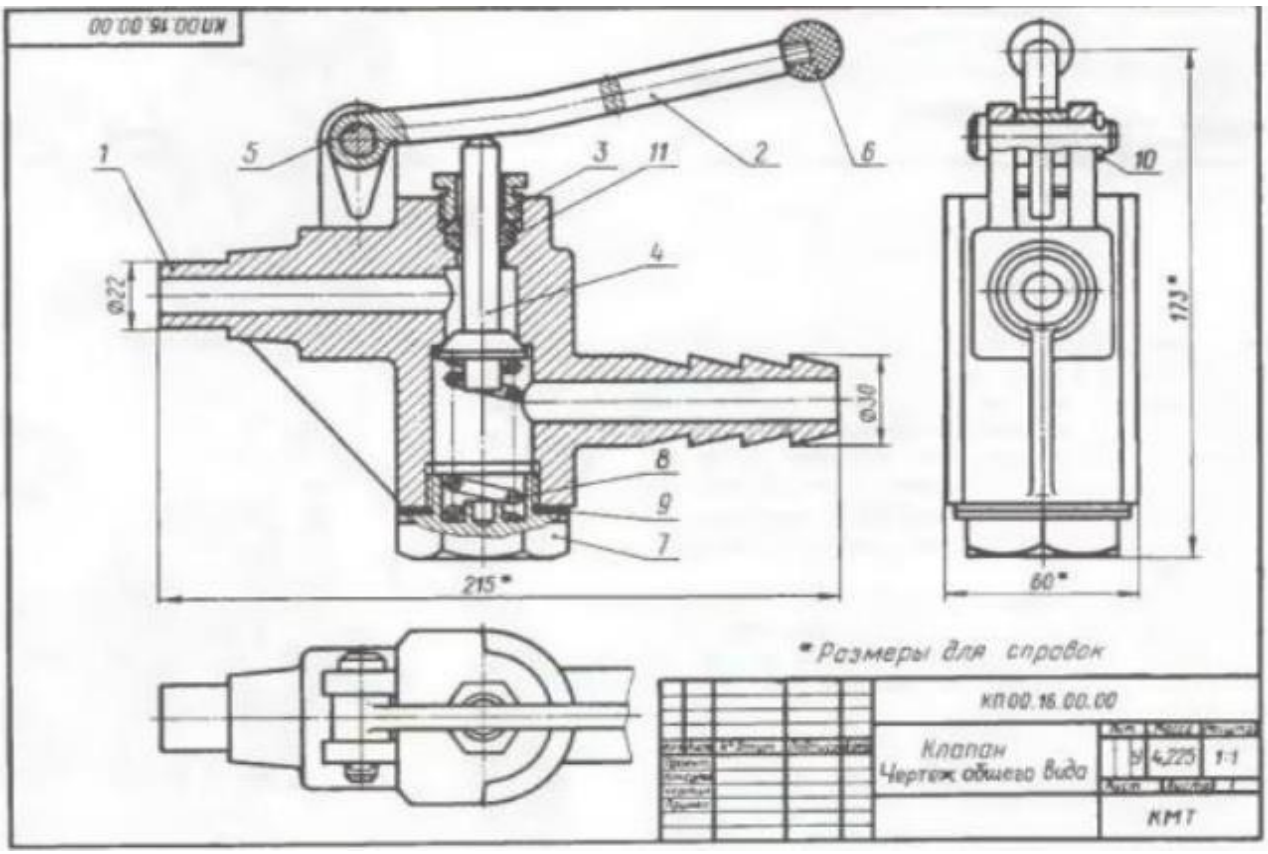


Рисунок 49 - Сборочный чертеж

Схема – документ, где в виде условных изображений или обозначений показаны все составные части изделия и связи между ними.

Спецификация – документ, в котором полностью перечислен состав сборочной единицы с указанием наименований чертежей, количества деталей и прочее.


Рисунок 50 - Спецификация к сборочному чертежу

При создании объектов конструкторской документации руководствуются требованиями, установленными государством в специальных документах – ГОСТ (государственный стандарт). В них указаны все приемлемые масштабы документов, правила оформления рамок и подписей, прописаны требования к используемым шрифтам, дана расшифровка значений тех или иных типов линий, регламентированы их

параметры (толщина, расстояние между штрихами). Линии, например, бывают: сплошные толстые основные, сплошные тонкие, сплошные волнистые, штриховые, штрих-пунктирные, разомкнутые.

Отдельной конструкторской задачей зачастую является эскизирование. Эскизом называется чертеж, выполненный от руки, в котором может быть не соблюден стандартный масштаб, но должны соблюдаться пропорции изделия. По содержанию требования к эскизам такие же, как и к чертежам. К ним прибегают в случаях первичной разработки новой конструкции, при составлении рабочих чертежей уже физически имеющихся деталей, если нужно изготовить подобную деталь, а ее чертежей нет.

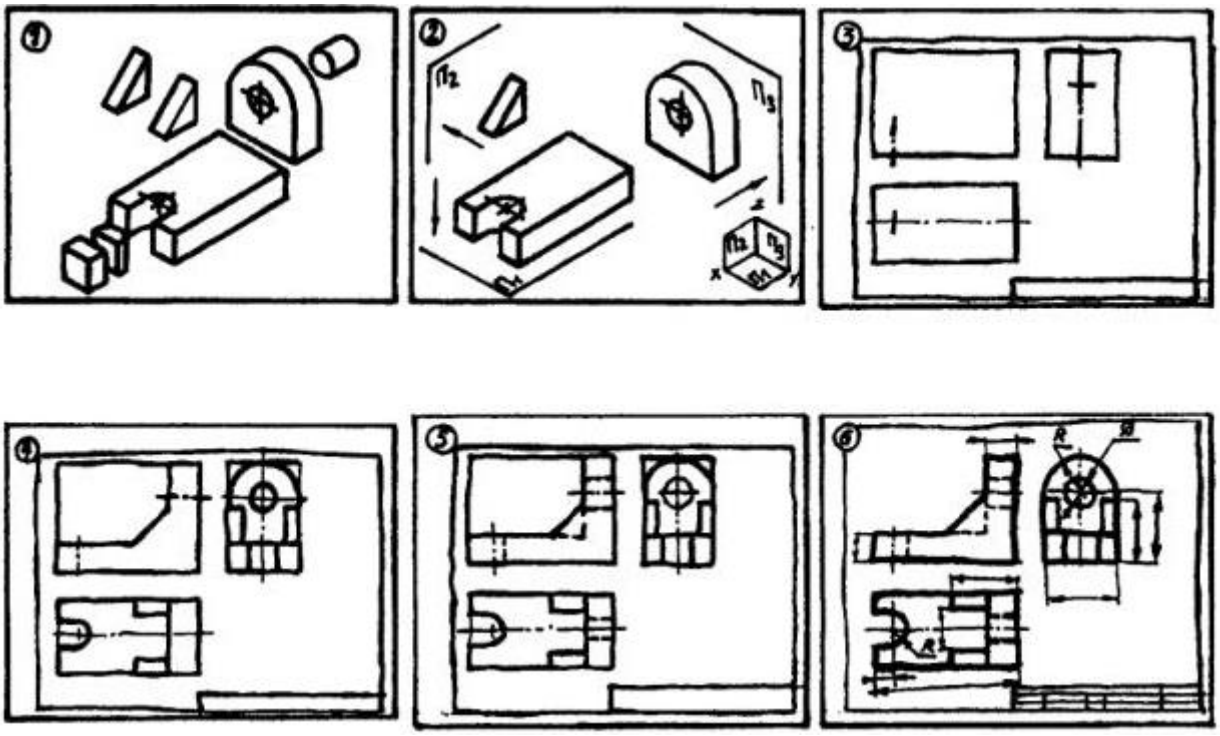

Рисунок 51 - Этапы построения эскиза

Чтобы правильно создать эскиз, нужно работать поэтапно: внимательно осмотреть деталь и изучить ее конструкцию, заметить выступы, фаски, резьбы и другие элементы, мысленно разделить деталь на геометрические примитивы, это пригодится при дальнейшем ее моделировании в САПР. Полезно попытаться установить материал детали и способ ее изготовления – это может помочь понять какие-либо конструкционные особенности изделия.

Следующая задача – выбор главного изображения детали. Главный вид должен максимально полно отражать ее геометрию и давать наиболее полную информацию. Определяют, какие разрезы будет полезно выполнить, чтобы показать внутренние части при необходимости.

### 4.4. Работа с поверхностями

Поверхностное моделирование – еще один способ создания деталей. Он применяется для проектирования объектов сложной формы, которую тяжело составить с помощью обыкновенно используемых геометрических примитивов. Один из подходов такого моделирование – использование набора опорных профилей и ободных кривых, на основе которых строятся поверхности.

Поверхностная модель основана на представлении объекта как совокупности нескольких поверхностей-граней, она может быть описана системой уравнений составляющих поверхностей. Она может иметь вершины, ребра и грани.

Основные инструменты при создании моделей:

- Базовые поверхности. Например, плоскости, получаемые разверткой отрезка прямой или конические поверхности, которые можно получить, развернув в трехмерном пространстве конические и цилиндрические фигуры;
- Поверхности вращения получаются вращением плоских граней вокруг оси;
- Поверхности сопряжений и пересечений. Они позволяют построить сложные геометрические объекты, при этом сохранив точность контактного соединения элементов. Образованы на линиях сопряжений и пересечений других типов поверхностей;

- Аналитические поверхности, заданные с помощью уравнений с тремя переменными – x, y и z.
- Поверхности свободной формы (скульптурные) – сложные виды поверхностей, описывающиеся системой математических уравнений.

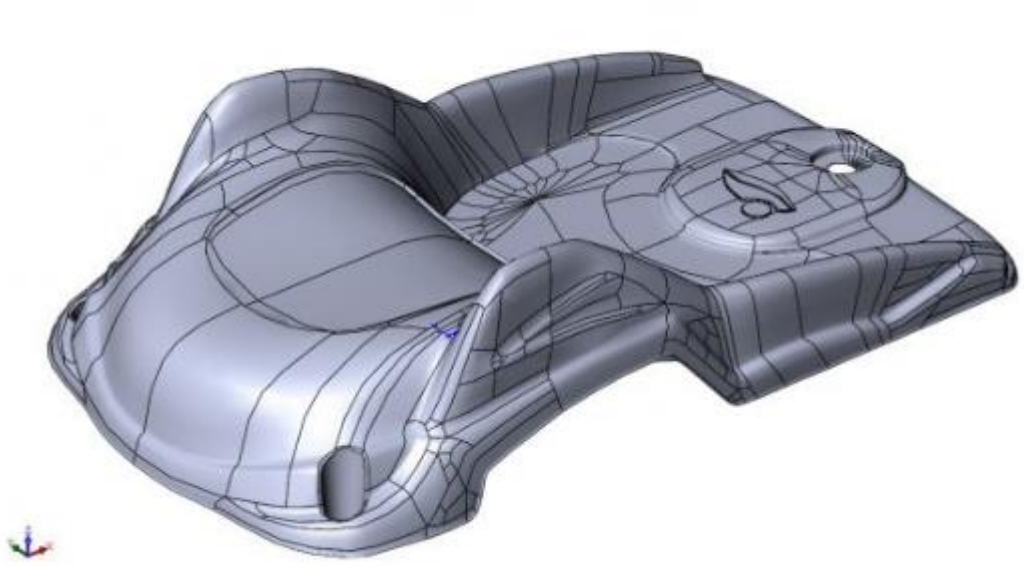

Рисунок 52 - Модель, спроектированная с помощью поверхностного моделирования

При таком виде моделирования в основном приходится работать с составными поверхностями. Составная поверхность – поверхность, составленная из нескольких элементарных поверхностей. Деталь, к примеру, может состоять из плоскостей, из цилиндрических поверхностей, соединяемых гранями.

Иногда поверхность можно полностью определить, покрыв ее сеткой четырехугольников. При помощи интерполяции может быть определена внутренняя область каждого четырехугольника, а изображение, полученное таким наложением, будет отображаться как многогранный каркас, на который автоматически аппроксимировано натяжение некой гладкой криволинейной поверхности.

Базовые операции такого способа моделирования – продление, соединение и обрезка. Все изделие должно быть описано как семейство

различных поверхностей. Рассмотрим основные преимущества поверхностной модели в сравнении со стандартным методом:

- Упрощенное создание криволинейных поверхностей и объектов на их основе;
- Инструменты для распознавания граней, чтобы получать качественные трехмерные изображения, пригодные к дальнейшей обработке;
- Распознавание особых построений, например, отверстий;
- Совместимость интерфейсов со станками ЧПУ при имитации траектории движения инструмента в пространстве.

Есть и некоторые недостатки, из-за которых создание моделей с помощью геометрических примитивов все еще популярно:

- Неоднозначность моделирования реальных твердых тел;
- Недостаток точности для последующего изготовления;
- Сложности с отображением внутренних областей и удалением вспомогательных скрытых построений.

Многие САПР имеют инструменты для поверхностного моделирования. Где-то они реализованы лучше, где-то хуже. Этот метод находит применение при укрупненном анализе геометрии деталей, в случае необходимости оценить возможность ее изготовления методом литья, а также при анализе тонкостенных элементов.

# 5. Механика и инженерия

Цель данного раздела – объяснить основные закономерности и методы соединения деталей, создания механизмов, познакомить читателя с основами механических узлов и их применением в робототехнике.

## 5.1. Основные понятия

Для введения в курс дела, обозначим некоторые определения.

Соединение – фиксация различных элементов конструкции относительно друг друга с применением различных вспомогательных материалов (свинчивание, склеивание и т.д.). Соединения могут классифицироваться по разным признакам.

В первой группе соединения разделяются на неразъемные, которые невозможно разобрать без деформации компонентов и разъемные, для которых возможность разборки предусмотрена.

Вторая группа разделяет соединения по типу подвижности. Соединение можно назвать подвижным, если детали в составе сборки могут перемещаться друг относительно друга без потери целостности конструкции. Неподвижное соединение – детали нельзя сместить друг относительно друга без повреждения конструкции.

Последняя группа признаков относится к форме сопрягаемых поверхностей. Так, соединения делятся на плоские, конические, сферические, винтовые и другие.

Выбор типа соединения определяется условиями взаимодействия деталей, требованиями к прочности соединения, предполагаемыми условиями эксплуатации, требованиями к надежности и долговечности.

## 5.2. Соединения

Подвижные и неподвижные соединения деталей для различных узлов, агрегатов и механизмов подбираются с учетом наибольшей целесообразности, исходя из запрашиваемых прочностных характеристик, особенностей монтажа, экономичности в изготовлении и эксплуатации.

Сварные соединения, например, применяются для неподвижного закрепления деталей из металла, когда требуется, чтобы соединение было прочным, долговечным и герметичным (например, соединение фланца и трубы).

Пайка, в общем-то, по технологии и характеристикам сходна со сваркой, но отличается тем, что для пайки применяются специальные составы (припои), как правило на основе олова, свинца и флюсовых добавок. Наиболее широко пайка применяется в радиотехнике, электронике, при соединении деталей гидравлических систем (пайка трубок и штуцеров) и т.д.

Заклепочное (клепаное) соединение применяется, когда детали испытывают знакопеременные нагрузки, за исключением нагрузок на срез. Например, соединение обшивки и силового каркаса самолета.

Резьбовые соединения применяются чаще всего в качестве разборных соединений.

Надежность резьбового соединения обеспечивается за счет силы трения в витках резьбы. Коэффициент трения в правильно соединенных деталях должен превышать коэффициент сдвига основных деталей. Величина коэффициента трения зависит от момента затяжки резьбового соединения, размеров и свойств резьбовой пары. Примером резьбовых соединений являются болты, шпильки, винты.

Шпоночные и шлицевые соединения применяются при соединении деталей совместного вращения. Чаще всего это валы и зубчатые колеса, валы и шкивы, валы и муфты, а также валы и всевозможные рукоятки, толкатели и т.п.

Шлицевое соединение обеспечивает передачу значительно большего момента, чем шпоночное и применяется в более нагруженных узлах.

Штифтовое соединение обеспечивает неподвижность и точное позиционирование деталей относительно друг друга. Может применяться для фиксации деталей при сварке. Такие соединения обеспечивают соосность отверстий.

Каждая группа соединений на самом деле включает в себя большое количество специфических и не очень подвидов, из них всегда можно выбрать наиболее подходящий для выполнения стоящей перед создателем робототехнической системы вариант.

### 5.3. Передачи и кинематические схемы

Проектирование многих механизмов обычно решает задачу передачи движения конкретному элементу. Реализация может быть совершенно разная: передача от двигателя к колесам, поступательного движения от поршней двигателя в коленвал и так далее. В этом разделе описаны самые часто встречающиеся типы приводов, которые могут оказаться полезными для использования в робототехнике.

Выделяют три основных вида передаваемого различными способами движения:

- Вращательное;
- Прямолинейное или возвратно-поступательное;
- Движение по определенной траектории.

Чаще всего стоит вопрос о передаче именно вращательного движения. Проще всего его решить системой шкив-ремень. Это реализовано следующим образом: шкивы, то есть, специальные колеса с канавкой или ободком по всей окружности, передающие движение приводному канату или ремню и использующие для этого энергию силы трения, взаимодействуют с помощью

ремня, передающего вращение с одного шкива на другой. Разные диаметры шкивов порождают разные скорости вращения.

Угловая скорость вращения шкивов остается одинаковой, особенно при допущении, что ремень не растягивается. С учетом их разных диаметров, линейная скорость движения точек обода отличается пропорционально отношению радиуса ведущего (вращающегося) и ведомого (вращаемого) валов.

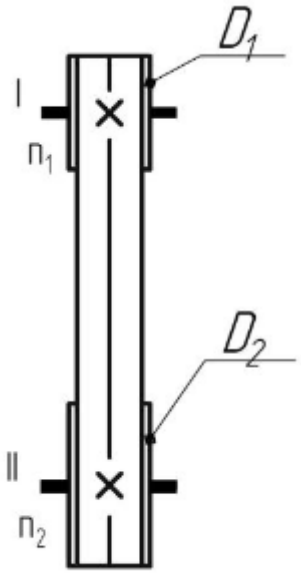


Рисунок 53 - Ременная передача

Кроме шкивов, используют зубчатые передачи – механизмы, которые с помощью зубчатого зацепления передают или преобразуют движение с изменением угловых скоростей и моментов. В отличие от ременных могут передавать движение не только в одной, но и в перпендикулярных плоскостях. Для этого разработаны специальные шестерни, передающие вращение между валами, находящимися под углом друг к другу.

Как и в случае с ременными передачами, возможно изменение скорости вращения вала. Для этого используется различное количество зубьев на шестернях. Определить частоту вращения можно из простой пропорции,

потому что произведения частоты вращения зубчатой оси на количество ее зубьев для ведомой и ведущей шестерней равны между собой.

Зубчатые передачи между параллельными валами осуществляются цилиндрическими колесами с прямыми, косыми и шевронными зубьями. Передачи между валами с пересекающимися осями осуществляются обычно коническими колесами с прямыми и круговыми зубьями, реже тангенциальными зубьями. Зубчатые передачи для преобразования вращательного движения в поступательное и наоборот осуществляются цилиндрическим колесом и рейкой.

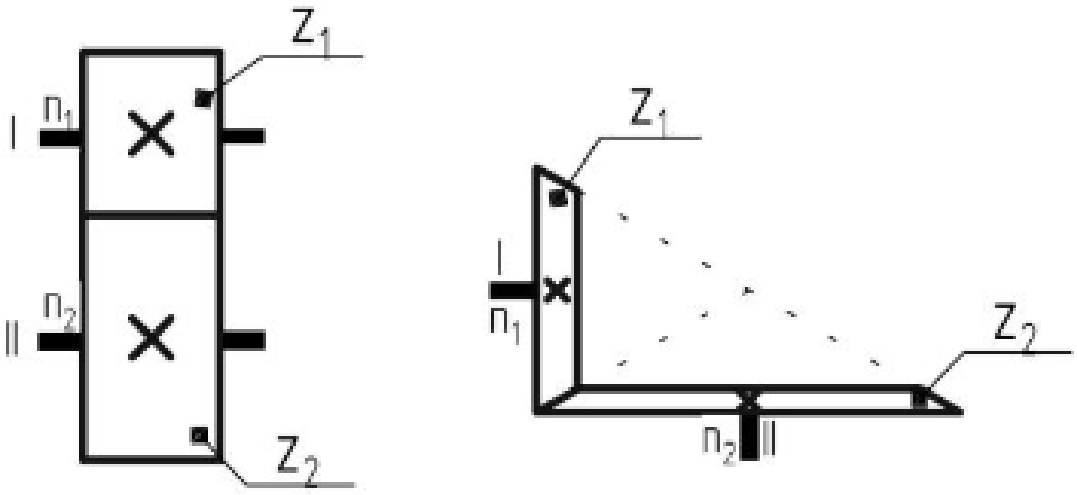


Рисунок 54 - Зубчатая передача

Червячная передача (зубчато-винтовая передача) — механическая передача, осуществляющаяся зацеплением червяка и сопряжённого с ним червячного колеса (для преобразования угловой скорости и усилия вращения) или гайки (для линейных перемещений).

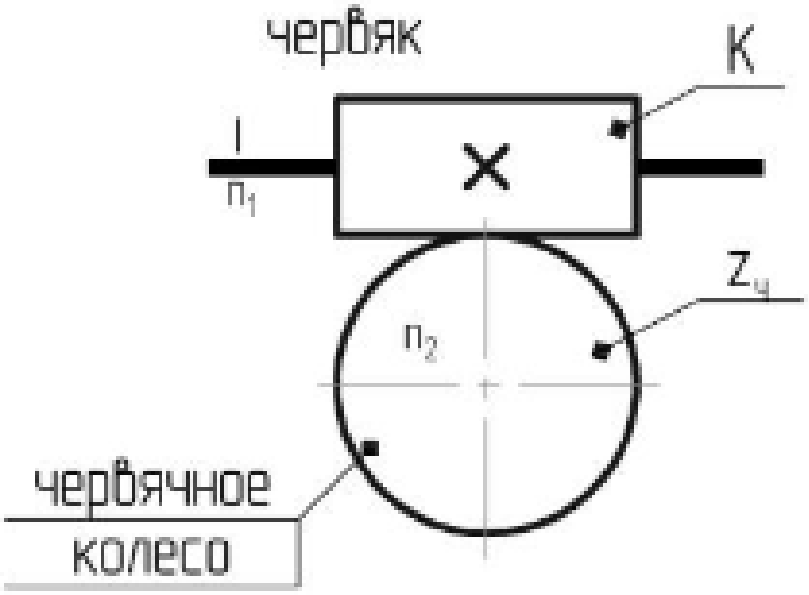


Рисунок 55 - Червячная передача

Дадим определение кинематической цепи. Это связанная система объектов, образующих между собой кинематические пары. Цепи бывают простые (каждое из звеньев входит в состав одной или двух кинематических пар) и сложные (имеются звенья, входящих в трех и более кинематических пар). Звено – какой-либо вид получения или передачи движения.

Задачи по этой теме будут изложены в практическом модуле.

# 6. Практика

Робототехника – достаточно прикладная наука, поэтому изучение теоретических основ многих составляющих, из которых она состоит, не несет особого смысла без какой-либо практики. Данный раздел учебника посвящен практической части изучаемого материала, содержит разборы тематических задач по разделам и предложения по усовершенствованию исполнения этих задач, которыми можно заняться самостоятельно.

Отдельное место в тексте будет занимать проектная деятельность. Умение правильно выстроить свою работу, поставить цель и сформировать задачи, выполнение которых приводит к ее достижению в том или ином виде, очень важно не только в школе, но и в университете, и в дальнейшей жизни вне зависимости от выбранной профессии и рода деятельности. Опишем основные этапы работы, способы ее грамотной реализации и презентации.

Первый раздел модуля посвящен практике с микроконтроллером Arduino UNO, основы работы с которым были изложены в соответствующей главе.

## 6.1. Создание Arduino-устройств

Как мы условились ранее, в данном пособии примеры разбираются на базе микроконтроллера Arduino UNO, как одной из самых базовых и распространенных моделей, на основе которой строится обучение чаще всего. Переход к другим версиям контроллера не очень сложен, потому что они имеют одинаковый принцип программирования и архитектуру. Все возникающие детали, как правило, решаются чтением сопроводительной документации к платам и комплектующим.

В ходе модуля будут подробно рассмотрены несколько примеров использования Arduino. Подключение и программирование выполнено на базе эмулятора Tinkercad от компании AUTODESK, так как это бесплатный

свободно распространяемый продукт, для работы с которым необходимо только стабильное интернет-подключение.

Tinkercad отлично подходит для начальных этапов изучения Arduino – он содержит базовый набор компонентов для начала работы, несколько готовых сборок для изучения, вводные уроки для ознакомления с интерфейсом и некоторыми принципиальными моментами.

Сервис также подойдет, если хочется попробовать работу с микроконтроллером, но при этом не приобретать его и комплектующие, а для начала изучить его возможности, некоторые нюансы и саму логику построения систем на его основе. Кроме того, сервис совершенствуется, расширяется спектр его возможностей и применений.

Обратимся к интерфейсу. Пользователя встречает предложение зарегистрироваться или войти с существующим аккаунтом, чтобы получить доступ к своему аккаунту, хранилищу своих проектов и сообществу единомышленников. После авторизации пользователь попадает на стартовую страницу.

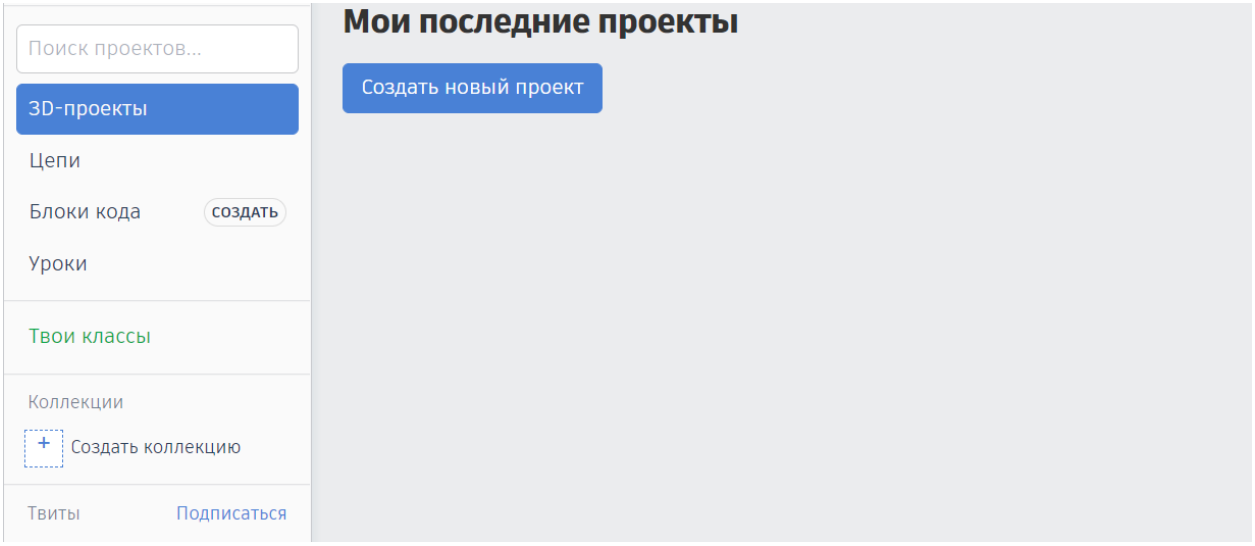


Рисунок 56 - Стартовая страница пользователя

Для работы с Arduino следует выбрать раздел «Цепи». После нажатия кнопки «Создать цепь» пользователь переходит непосредственно в интерфейс

работы с платой. В верхней панели, рядом с логотипом сервиса, можно выбрать название проекта.

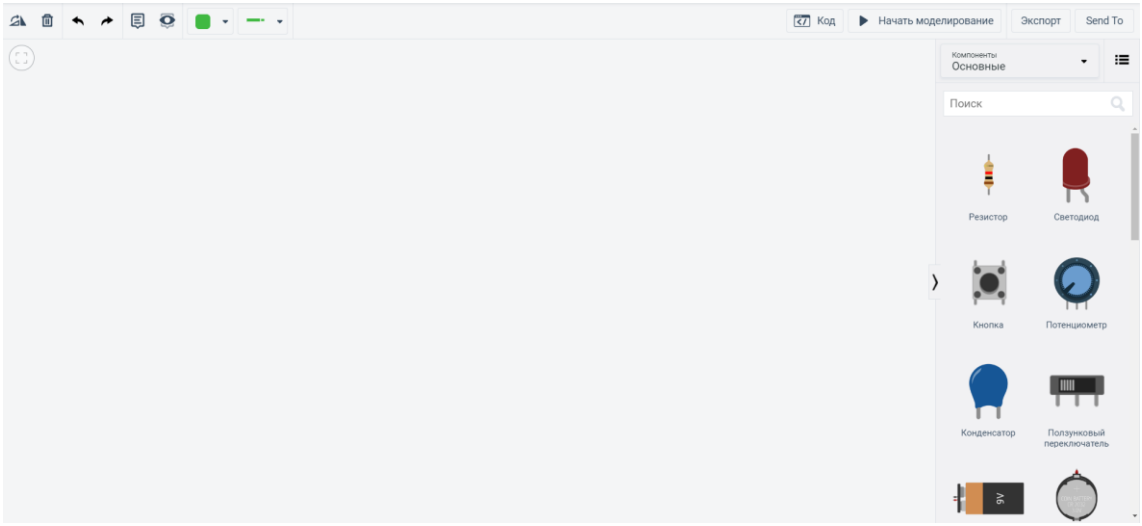


Рисунок 57 - Рабочая область

Листая меню компонентов, нетрудно найти Arduino UNO R3. Чтобы разместить ее в рабочей области достаточно кликом мыши перетащить изображение платы из меню на поверхность. В меню основных компонентов расположены самые частые. Чтобы посмотреть полный список, нужно перейти в раздел «Все» меню компонентов.

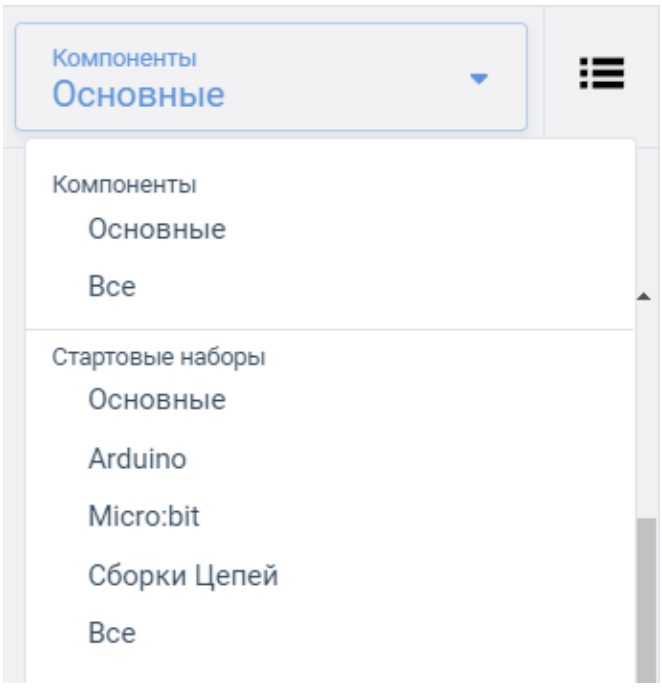


Рисунок 58 - Подменю выбора компонентов

Провода для соединения элементов создаются простым соединением нужных клемм. В контекстном меню можно выбрать вид провода и его цвет при необходимости. Клавишей Del ненужный элемент можно удалить.

Чтобы перейти к написанию программы, нужно попасть в раздел «Код», расположенный в верхнем меню правее.

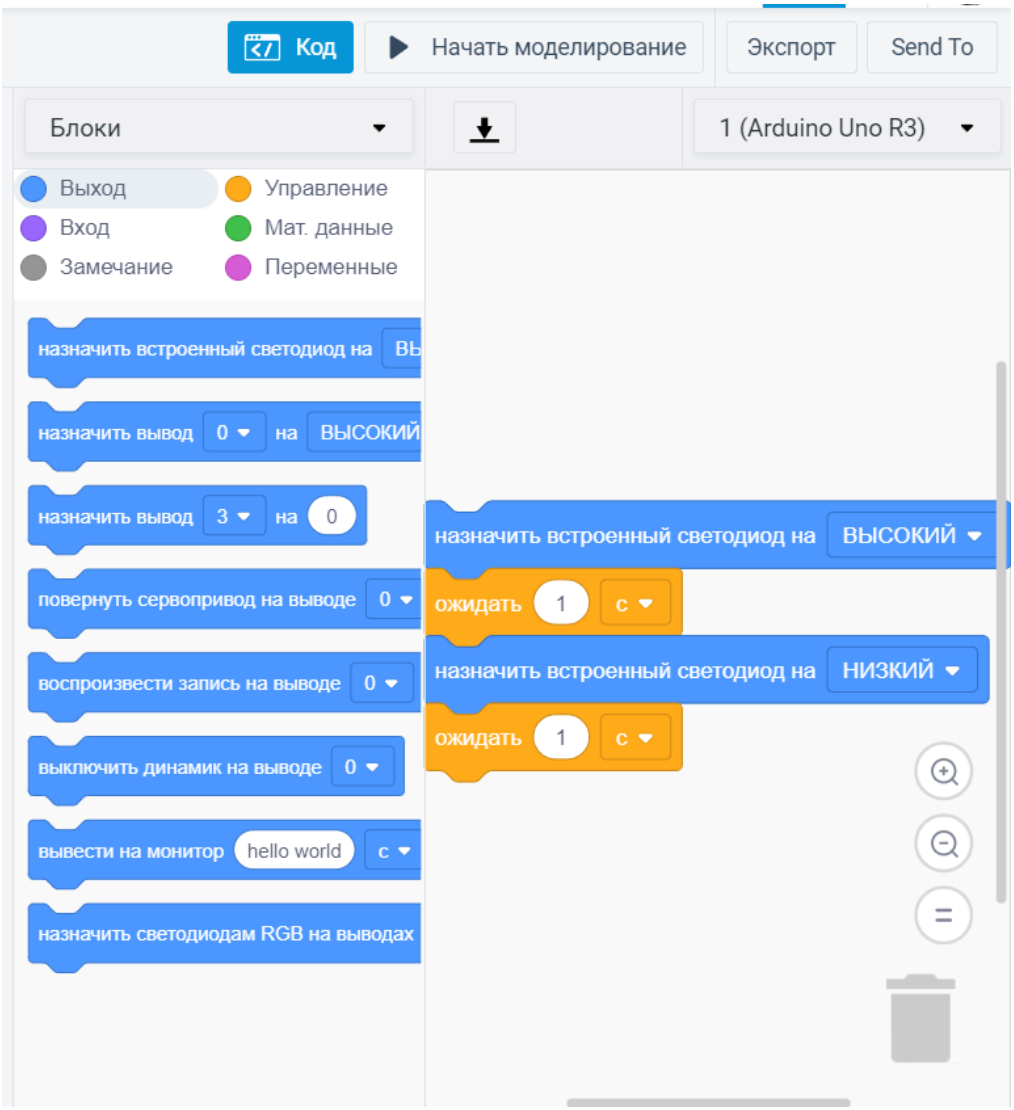


Рисунок 59 - Первоначальное меню кода

Несложно догадаться, что это не очень похоже на то, что было рассмотрено на этапе изучения Arduino, поэтому режим блоков нужно переключить на текст, предварительно согласившись в сплывающем предупреждении о том, что все, что было сделано ранее, сбросится. После переключения интерфейс станет гораздо более привычным. Написанный код можно скачать, библиотеки

подключаются в специальном меню, доступен монитор последовательного интерфейса. По умолчанию доступен код программы, которая раз в секунду мигает встроенным в плату диодом. Этой программы достаточно, чтобы при нажатии кнопки «Начать моделирование» начался отсчет времени и симуляция работы схемы.

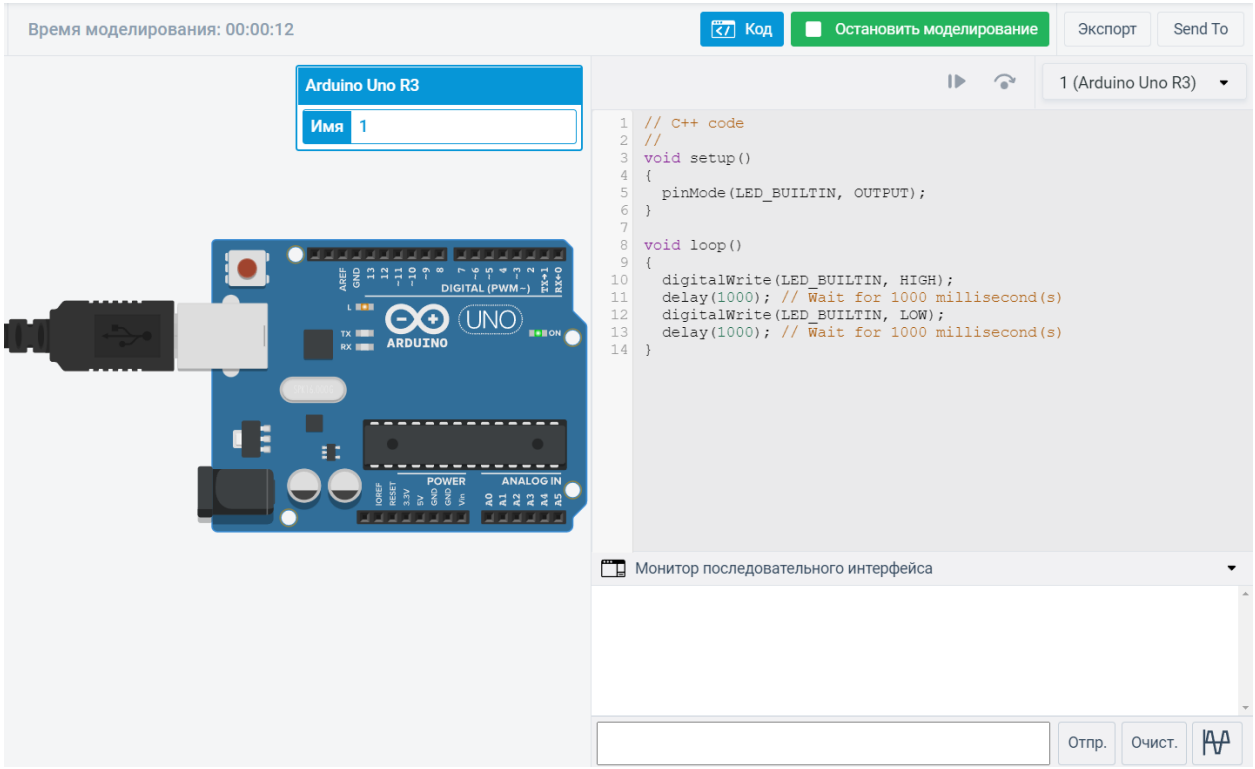


Рисунок 60 - Простейшая симуляция

Во время моделирования редактировать код и схему невозможно. То же относится и к физическим сборкам – все подключения можно производить только при отключенном питании. Моделирование не всегда происходит в режиме реального времени, в этом нет ничего страшного, просто нужны определенные мощности, чтобы воспроизвести все происходящие процессы.

Если в программе предусмотрен вывод значений в монитор последовательного интерфейса, можно сразу же их визуализировать в виде графика нажатием соответствующей кнопки.

Интерфейс достаточно прост, поэтому ознакомление с его основными элементами тоже проходит быстро. Можно переходить к рассмотрению

конкретных примеров. Разберем один из самых простых проектов, через который проходят, наверное, все новички.

### Пример 1. RGB-светодиод

Управлять трехцветным диодом интереснее, чем тремя отдельными, потому что управляющие программы для этих случаев друг от друга почти не отличаются, а в первом случае можно смешивать цвета и создавать некое подобие цветовой палитры – устройства, которое может помочь пользователю подобрать цвета. Пусть наша задача – написать программу, в результате работы которой красный цвет включается на полторы секунды, зеленый – на одну, синий – на половину секунды, а потом две секунды горят все три диода.

Клеммы на диоде помечены. Те, которые называются как соответствующие цвета, нужно через резисторы подключить к цифровым выходам. Оставшаяся клемма подключается к разъему GND. Схема несложная, показана на рисунке ниже.

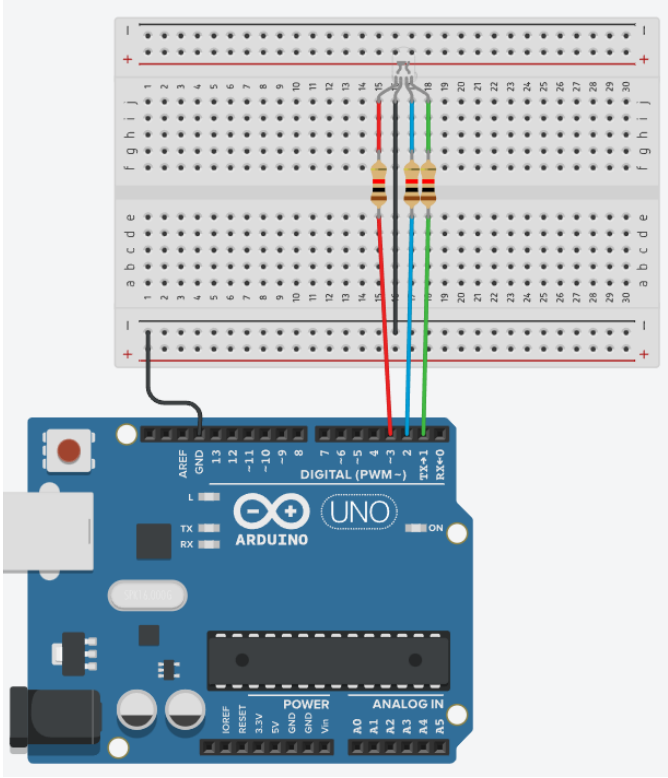


Рисунок 61 - Подключение RGB-светодиода

Программа, которая позволяет выполнить поставленную задачу, крайне несложная и однообразная:

```
#define RED 3  // присваиваем имя RED для пина 1
#define GRN 1 // присваиваем имя GRN для пина 2
#define BLU 2 //присваиваем имя BLU для пина 3

void setup() {
   pinMode(RED, OUTPUT);  // используем Pin11 для вывода
   pinMode(GRN, OUTPUT); // используем Pin12 для вывода
   pinMode(BLU, OUTPUT);  // используем Pin13 для вывода
}

void loop() {

   digitalWrite(RED, HIGH); // включаем красный свет
   digitalWrite(GRN, LOW);
   digitalWrite(BLU, LOW);

   delay(1500); // устанавливаем паузу для эффекта

   digitalWrite(RED, LOW);
   digitalWrite(GRN, HIGH); // включаем зеленый свет
   digitalWrite(BLU, LOW);

   delay(1000); // устанавливаем паузу для эффекта

   digitalWrite(RED, LOW);
   digitalWrite(GRN, LOW);
   digitalWrite(BLU, HIGH); // включаем синий свет
  delay(500);

   digitalWrite(RED, 1);
   digitalWrite(GRN, 1);
   digitalWrite(BLU, 1); // включаем синий свет
  delay(2000); // устанавливаем паузу для эффекта
}
```

Рисунок 62 - Код программы управления RGB-светодиодом

В данной схеме нет ничего, чего не было бы разобрано ранее в главе, посвященной микроконтроллерам. Пример этот нужен для лучшего ознакомления с интерфейсом и знакомства с небольшой рутинной задачей. Похожие часто возникают на этапе программирования собственных проектов.

**Пример 2. Проверка скорости реакции человека**

Следующий пример иллюстрирует то, как с помощью контроллера можно создавать устройства, близкие, например, к медицинской физике. Цель данной схемы – узнать время реакции человека на раздражитель, в данном случае – диод, который загорается в случайное время с момента начала цикла программы. Все, что понадобится для данной системы из компонентов, помимо контроллера – тактовая кнопка и диод. Подключение диода не

отличается от рассмотренного ранее. Кнопка же имеет четыре клеммы. Правая нижняя клемма подключается к разъему GND. Верхнюю левую можно с помощью макетной платы присоединить одновременно к цифровому входу и к разъему 5V через резистор. Такое подключение выполняется для устранения возможных ложных срабатываний.

Кнопка работает просто – нажатие замыкает электрическую цепь, на цифровом выходе появляется высокий уровень. В остальное время, когда кнопка не нажата, уровень остается низким. Возможная схема сборки выглядит так:

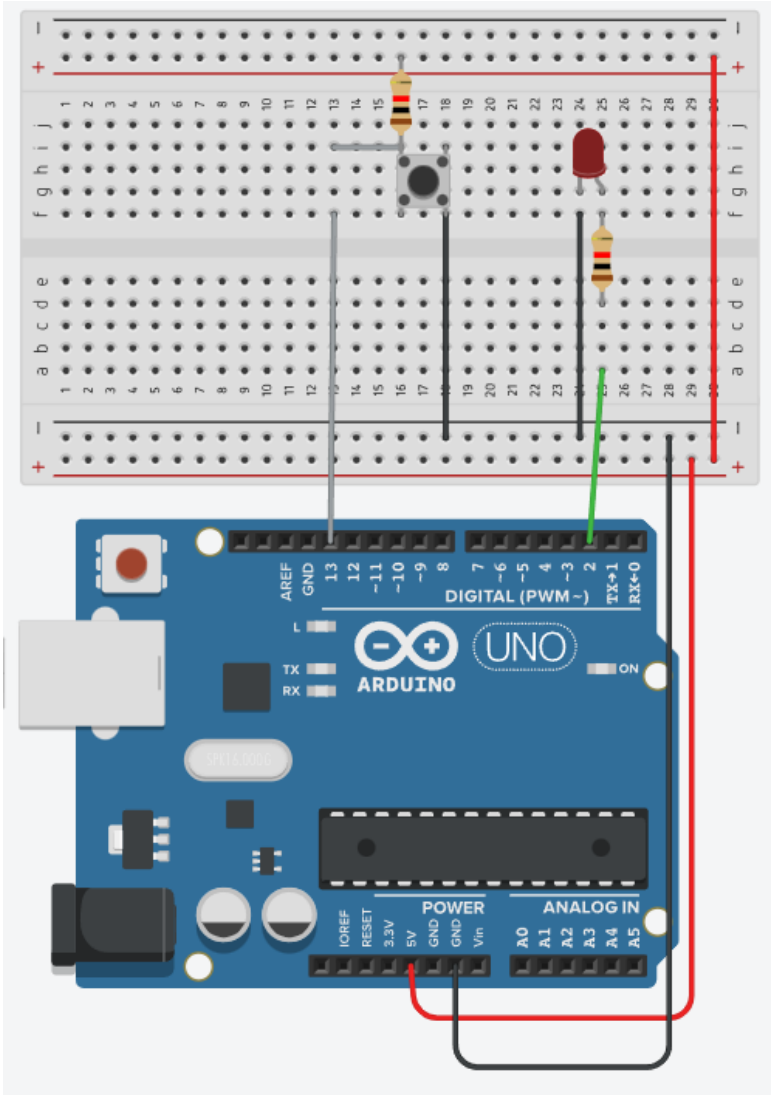


Рисунок 63 - Схема подключения устройства проверки скорости реакции на раздражитель

Программа, позволяющая реализовать поставленную задачу, так же не отличается сложностью:

```
int button = 13;
int led = 2;
int t, react;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(button, INPUT);
  randomSeed(A0);
  Serial.begin(9600);
}

void loop() {
  delay(random(500, 2000));
  digitalWrite(led, HIGH);
  t = millis();
  while (digitalRead(button)==HIGH) {}
  digitalWrite(led, LOW);
  react = millis() - t;
  Serial.println(react);
}
```

Рисунок 64 - код программы к примеру 2

Здесь входу диода и кнопки даются свои названия, заводятся переменные для хранения данных о времени. Порт диода устанавливается на выход, порт кнопки – на вход, потому что с него мы собираемся читать данные. Функция randomSeed(A0) нужна для генерации в программе случайных чисел. Если диод постоянно будет загораться через фиксированный промежуток времени от начала итерации программы, пользователь привыкнет, и эксперимент не будет честным. В данном случае числа будут получаться действительно случайными, их источником выбираются помехи на аналоговом порту A0, к которому в данном случае ничего не подключено.

Функция Serial.begin нам уже знакома – она открывает передачу данных в последовательный интерфейс, потому что мы собираемся выводить полученное время реакции в миллисекундах в монитор порта.

Случайное время задержки перед включением сигнального диода достигается командой delay, в аргументе которой стоит random, результатом выполнения которого будет случайное число в указанном диапазоне. После ожидания в

течение этого времени диод загорается, и начинается отсчет времени, которое понадобится человеку, чтобы среагировать.

Команда millis() возвращает количество миллисекунд, прошедшее с загрузки программы в плату и начала ее выполнения. Мы сохраняем это значение в переменную t, когда включаем светодиод.

Цикл while помогает поймать момент, когда пользователь все же нажмет кнопку. Когда условие появления высоко уровня на входе кнопки будет выполнено, можно погасить диод и сразу же посчитать время, которое прошло с момента его включения. Для этого переменной react присваивается значение, равное разности текущего количества миллисекунд, прошедшего с момента первого выполнения программы и переменной t, хранящей время включения диода. Именно эта разность нам и нужна. Полученное значение выводится в монитор порта. Результат работы программы таков:

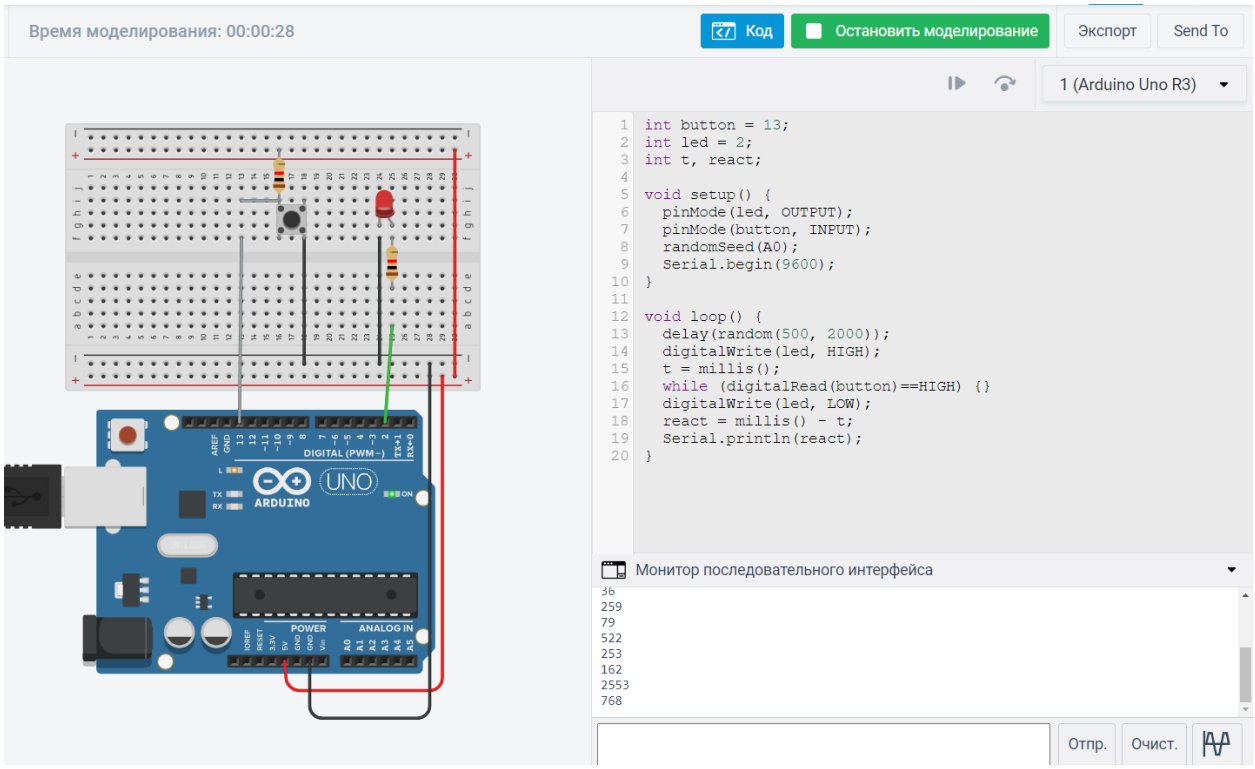


Рисунок 65 - Результат выполнения

Таким образом эмулятор позволяет смоделировать прототип реального устройства. Программу можно дополнить выводом данных на дисплей, а не на

компьютер, и тогда, при наличии внешнего питания, устройство сможет работать автономно. Раздражитель так же можно заменить другими элементами, например, пьезоэлементом или вибромотором.

**Пример 3. Тоновая клавиатура**

В качестве примера подключения нескольких кнопок к одному устройству, то есть, по сути, интерфейса взаимодействия системы с пользователем, сделаем это на основе тоновой клавиатуры. В этом случае источником звука является пьезоэлемент – элемент, который благодаря своей структуре может преобразовывать электрические импульсы в колебания воздуха, то есть, звук, и наоборот.

В Arduino есть отдельная команда для программирования такого устройства и она называется tone. В нее передается три аргумента – номер выхода, к которому подключен пьезоэлемент, частота звука, которую он должен произвести, и длительность импульса. На самом деле, длительность указывать необязательно, функция сработает и с двумя аргументами.

В данном случае мы делаем маленькую вариацию клавиатуры музыкального инструмента, поэтому частоты будут совпадать с известными частотами нот. Нажатие на определенную кнопку будет вызывать звук определенной частоты.

Подключение кнопки было рассмотрено в прошлом примере, здесь нужно продублировать его еще два раза. Стоит обратить внимание на крайне распространенную ошибку – если одинаковые элементы схемы отвечают разным задачам, как в нашем случае, их все равно нужно подключать к разным входам, иначе микроконтроллер не поймет никакого отличия между ними. Поэтому три кнопки обязательно должны быть подключены к различным цифровым входам контроллера. Еще одно замечание – аналоговые входы контроллера могут работать как цифровые, но обратное утверждение несправедливо. К аналоговому входу в общем случае применимы все команды для цифрового.

Пьезоэлемент подключается одной клеммой на разъем GND, другой – на любой цифровой выход. Схема устройства, сделанная с помощью беспаечной макетной платы, представлена на рисунке ниже.

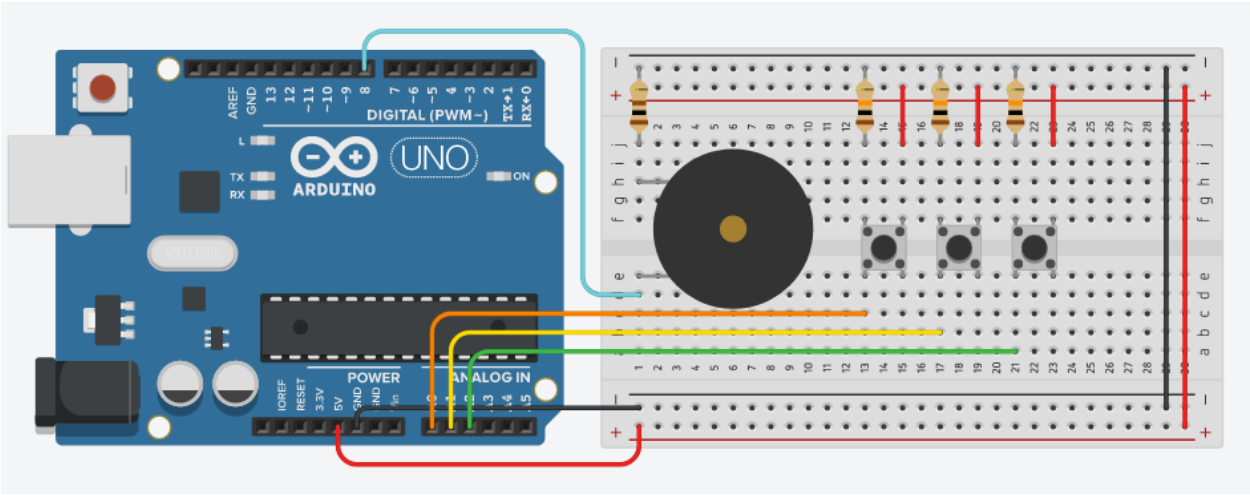


Рисунок 66 - Электронная схема подключения для примера 3

В данном разделе все примеры собираются с использованием беспаечной макетной платы. Это удобно не только с точки зрения визуализации логики подключения, но и при физической сборке схемы, потому что при использовании платы отпадает почти любая необходимость паять провода. Без оной, например, чтобы подключить четыре компонента к одному разъему 5V, пришлось бы выкручиваться.

```
1  void setup()
2  {
3    pinMode(A0, INPUT);
4    pinMode(8, OUTPUT);
5    pinMode(A1, INPUT);
6    pinMode(A2, INPUT);
7  }
8
9  void loop()
10 {
11   if (digitalRead(A0) == HIGH) {
12     tone(8, 440, 100);
13   }
14   if (digitalRead(A1) == HIGH) {
15     tone(8, 494, 100);
16   }
17   if (digitalRead(A2) == HIGH) {
18     tone(8, 523, 100);
19   }
20   delay(10);
21 }
```

Рисунок 67 - Код управляющей программы к примеру 3

Программа оказывается совершенно тривиальной и может быть однообразно развита до любого количества кнопок на тоновой клавиатуре. В разделе setup объявляются режимы работы выходов. Там нет ничего неожиданного – разъемы, к которым подключены кнопки, работают на вход, потому что с них значения считываются, разъем, на котором находится пьезоэлемент работает на выход, потому что на него с помощью функции tone мы будем отправлять импульсы, чтобы получить звуковой сигнал.

Единственная достойная внимания конструкция в разделе loop находится в строках 11-13 и повторяется в последующих, просто в аргументах фигурируют другие порты. Эти строки содержат в себе условный оператор – если кнопка нажата, то есть, цепь замкнута, и на ней высокий логический уровень, то на пьезоэлемент нужно подавать импульсы для получения звука определенной частоты.

Существует множество способов улучшить, преобразовать и использовать в измененном виде похожую программу. Можно соотнести длительность звукового сигнала с длительностью нажатия клавиши, добавить большее число клавиш. Часто в таких случаях используются операторы множественного выбора, которые могут помочь, если прописывать условия для каждого компонента в отдельности слишком сложно или трудозатратно.

**Пример 4. Датчик температуры**

Следующий пример интересен с точки зрения небольшой обработки приходящего на аналоговый порт сигнала и выполнения какого-либо набора действий, на основе получаемых данных. Более того, в этом случае можно построить график. Иногда в робототехнической системе возникает необходимость получения данных о внешней среде, их обработки и какого-то относительно форматированного вывода. Существуют такие дисплеи, например, на которых можно выводить графики и строить достаточно подробные графические интерфейсы. Бывают и случаи, когда получаемые

данные удобно загружать на внешний носитель вроде SD-карты, либо пересылать их по беспроводным интерфейсам. Пересылка данных с помощью радиопередатчика была рассмотрена в соответствующем разделе.

Сейчас задача такова: есть аналоговый датчик температуры и три диода. В зависимости от температуры загорается определенный индикатор состояния. Кроме того, данные отображаются в мониторе последовательного интерфейса, на их основе строится график.

Модель аналогового датчика температуры в эмуляторе основана на датчике TMP36. У него три выхода – два для питания, 5V и GND и один сигнальный, который подключается к аналоговому входу. Физический принцип работы изящен – по сути, это резистор, сопротивление которого очень сильно зависит от температуры. Такие резисторы называются термисторами и чем-то в концепции схожи с фоторезисторами. Подключение диода уже было рассмотрено ранее.

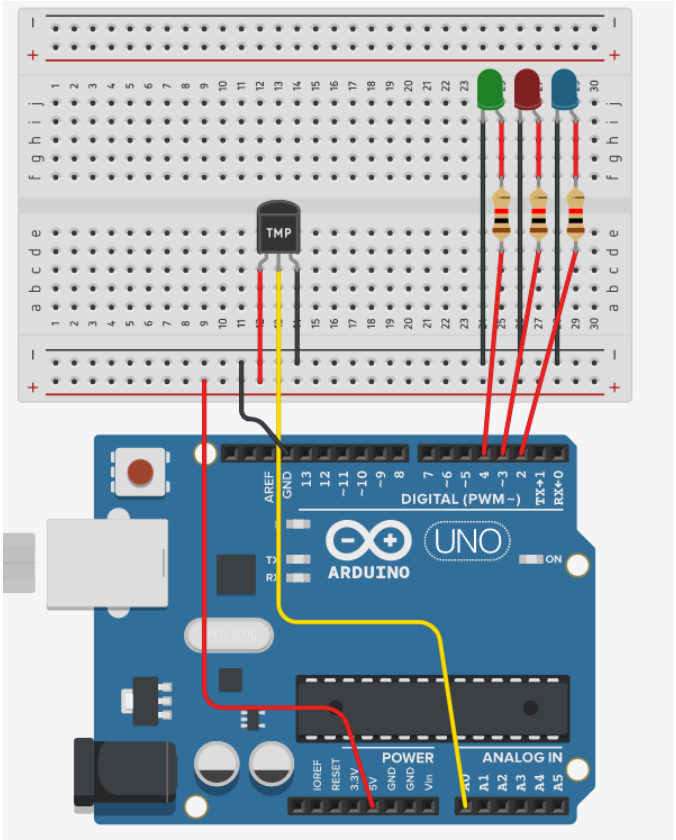


Рисунок 68 - Электронная схема к примеру 4

Управляющая программа не требует никаких специальных библиотек, но есть один несложный нюанс. Значения, приходящие с датчика, находятся в диапазоне от 0 до 1023, как и положено, и их нужно перевести в температуру, учитывая, что она может быть и отрицательной тоже, учитывая диапазон измерений. Согласно технической документации, диапазон измеряемых температур составляет значения от -40°C до +150°C. Немного окольным путем мы вводим поправочный коэффициент, с помощью которого можно переводить аналоговые значения в температуру.


Рисунок 69 - Код управляющей программы примера 4 и вывод данных в монитор последовательного интерфейса

В начале, как обычно, мы вводим нужные для работы переменные. Целочисленные нас уже не устраивают – мы имеем возможность получать более точные дробные значения, поэтому используем тип данных float. В переменные этого типа будут записываться значения температуры,

напряжения и промежуточные значения с учетом поправочных коэффициентов. Они все одного типа, потому что в вычислениях лучше не допускать взаимодействия переменных разного рода – это может привести к ошибкам и некорректным результатам, казалось бы, верных вычислений.

Среди установочных параметров программы указываем только начало передачи данных в последовательный интерфейс, больше ничего не понадобится. В циклической части первым делом считываем значение с датчика и переводим его в значение из нужного нам диапазона. Иногда алгоритм перехода значений указан в технической документации, иногда нужно немного подумать над ним самостоятельно. Измерения проводим раз в сто миллисекунд, чтобы была возможность быстро получать достаточно плавные графики, итоговые значения выводим в монитор порта.

Не забываем, что в системе расположены три диода, поэтому далее в тексте программы следуют условия их включения. В зависимости от значения температуры, которое было рассчитано, загорается тот или иной индикатор. Реализовано это при помощи простого условного оператора, который встречался в ходе повествования уже не раз.

В эмуляторе значения температуры, которая измеряется датчиком, задаются вручную с помощью ползунка, а потому оказывается очень несложно проверить правильность своих расчетов. В монитор порта выводятся полученные значения, на их основе тут же строится график, отражающий их изменение с течением времени.

Рассмотренный пример, как и остальные, призван обобщить какой-то случай реальной робототехнической задачи, которую зачастую приходится решать. Фигурировать может другой датчик, с другим принципом измерения, могут быть другие способы оповещения о его состоянии, но важен сам факт создания взаимосвязи между показанием прибора и ответными действиями на него, прописанными в управляющей программе.

## 6.2. Разработка электронных схем

Практический раздел модуля электроники будет посвящен созданию печатных плат для устройств. В качестве базовой программы разработки предполагается использовать EasyEDA, о которой говорилось ранее. Она не требует установки и обладает достаточным функционалом. Помимо этого, существует командный режим работы, в котором сразу несколько людей могут разрабатывать одну плату, совершенствовать ее, вносить правки, отслеживать версии.

### Начало работы

Для доступа к интерфейсу достаточно либо зарегистрироваться на соответствующем интернет-ресурсе, либо установить ПО на компьютер и запустить. После этого можно попасть в личный кабинет пользователя, где будут храниться все проекты и библиотеки корпусов и компонентов.

Интерфейс программы достаточно прост и не требует отдельного изучения, тем более, существуют встроенные уроки, которые могут разъяснить возникающие вопросы. Перейдём к рассмотрению примера.

### Создание электрической принципиальной схемы

В качестве примера для создания платы соберём преобразователь напряжения с 9 до 180 В, схема которого приведена ниже:

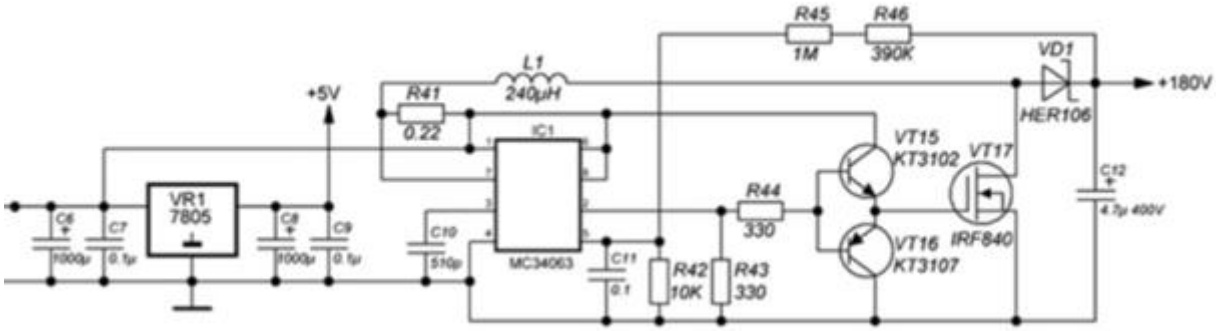


Рисунок 70 - Схема электрического преобразователя напряжения 9-180В

Для создания принципиальной схемы нужно вернуться к начальному экрану и создать новый проект: **Документ→Новый→Проект** (необходимо указать только название проекта). Для удобства можно настроить формат рабочего листа, используя модальное окно **Инструменты рисования**.

Далее размещаются все необходимые компоненты на рабочем поле при помощи вкладки **EELib** (находится левее рабочего поля).

Для поиска компонентов, которых нет во вкладке **EELib (**микросхемы MC34063, стабилизатора LM7805 и так далее) нужно воспользоваться библиотеками. В левой части рабочего поля находится вкладка **Поиск библиотек**. В появившемся окне достаточно ввести название искомого компонента. Далее из предложенного списка выбрать нужный (компоненты могут отличаться по типу корпуса и УГО). Также в данной вкладке сразу же можно купить необходимый компонент, а также посмотреть документацию, что иногда бывает необходимо, особенно при работе со сложными компонентами (микроконтроллерами, АЦП, модулями памяти).

После того, как все компоненты найдены, необходимо соединить их между собой в соответствии с принципиальной схемой. Для соединения используется операция **Провода**, которая находится в меню с выпадающим списком

**Соединение**. По аналогии соединяются все необходимые компоненты. В результате должна получиться схема, подобная рисунку ниже.

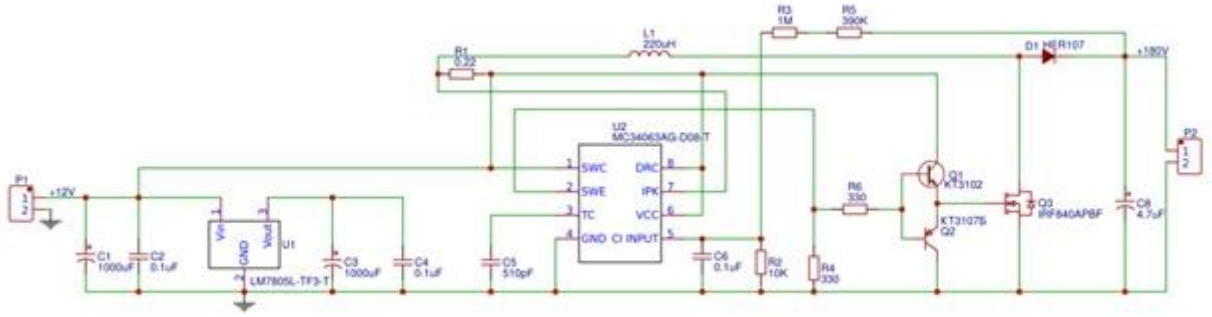


Рисунок 71 - Принципиальная схема преобразователя напряжений

На этом создание принципиальной схемы завершается. Можно приступать к работе с платой.

**Компоновка и трассировка**

Для создания схемы печатной платы по принципиальной схеме необходимо на верхней панели на вкладке **Конвертировать** нажать на ссылку **Конвертировать в печатную плату**. Следующим шагом будет компоновка посадочных мест всех компонентов и определение границ самой платы. Распределение компонентов происходит вручную, так как автокомпоновка отсутствует. Распределяем все компоненты таким образом, чтобы было удобно в будущем протягивать между ними дорожки. При компоновке связи между компонентами, которые будет необходимо в будущем соединить, подсвечиваются, что помогает расположить компоненты оптимальным образом.

После компоновки переходят к трассировке. Трассировка печатных плат — это один из шагов проектирования, который представляет собой процесс определения места и реализации проводящего рисунка платы. При осуществлении разводки плат выполняется прокладка проводников, которые

соединяют между собой те или иные компоненты. Трассировку можно выполнить вручную, соединяя все необходимые компоненты между собой. Можно упростить процесс с помощью функции автотрассировки. Тогда дорожки создаются автоматически по заранее определенным связям между компонентами. Для автотрассировки на верхней панели во вкладке **Разводка** переходим к **Автотрассировщику**. Никаких параметров, кроме ширины дорожки 0,35 мм, задавать не нужно, нажимаем на кнопку **Запустить**. Ширина дорожки определяется исходя из тока, который будет по ней протекать, способа изготовления платы, плотности установки компонентов.

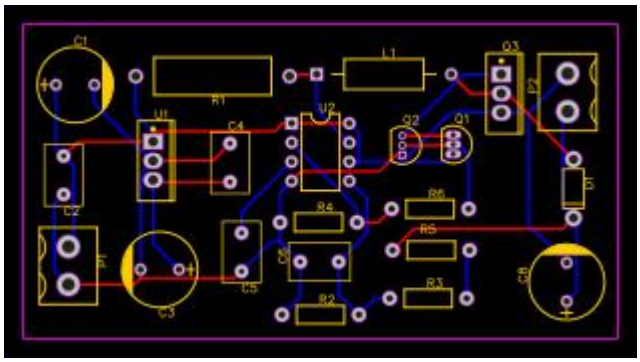


Рисунок 72 - Результат автотрассировки в EasyEDA

Считается, что самый лучший способ трассировки – ручной, когда можно предусмотреть все нюансы будущего изделия самостоятельно. Автотрассировка полезна в случае изготовления стандартных несложных компонентов, которые не требовательны к размеру и не предполагают большой токовой нагрузки.

## 6.3. Работа в CAD

В этом разделе будет разобран несложный пример создания 3D-модели и оформления ее неполного чертежа в программе КОМПАС-3D. Цель этого разбора – поэтапно описать принцип создания модели и логику построения чертежа, а также ознакомиться с интерфейсом программы.

Сразу стоит упомянуть – в КОМПАС-3D многие вопросы можно решить, используя внутреннюю справку программы, а нужную операцию найти в поиске в специальном окошке. Предложенный способ выполнения работы не является единственно верным – можно сделать и проще, и более примитивно.

Ознакомимся с заданием. На рисунке ниже изображен эскиз детали. Нужно построить 3D-модель и оформить ее чертеж. Приведен именно эскиз, потому что на нем, в отличие от обычного чертежа, отсутствует указание масштаба. Пунктирными линии – невидимые, они показывают профиль детали изнутри.

Деталь симметричная. Построение будет вестись с помощью геометрических примитивов. Размеры на эскизе, как и на всех чертежах по правилам, указаны в миллиметрах. За основу можно взять цилиндр высотой 15 мм, внешним диаметром 126 мм, внутренним – 30.

Далее на цилиндре расположен еще один – внешний диаметр его 84 мм, внутренний – переменный. На расстоянии 60 мм от верхней грани он будет составлять 50 мм, ниже – 30 мм. У детали присутствуют ребра жесткости. При работе с ними следует учитывать, что если выполняется разрез, и ребро попадает под него, оно не штрихуется.

У детали есть два паза – один прямоугольный, шириной 30 миллиметров и глубиной 10, на верхнем цилиндре и сквозной полукруглый на расстоянии 56 мм от верхней грани детали.

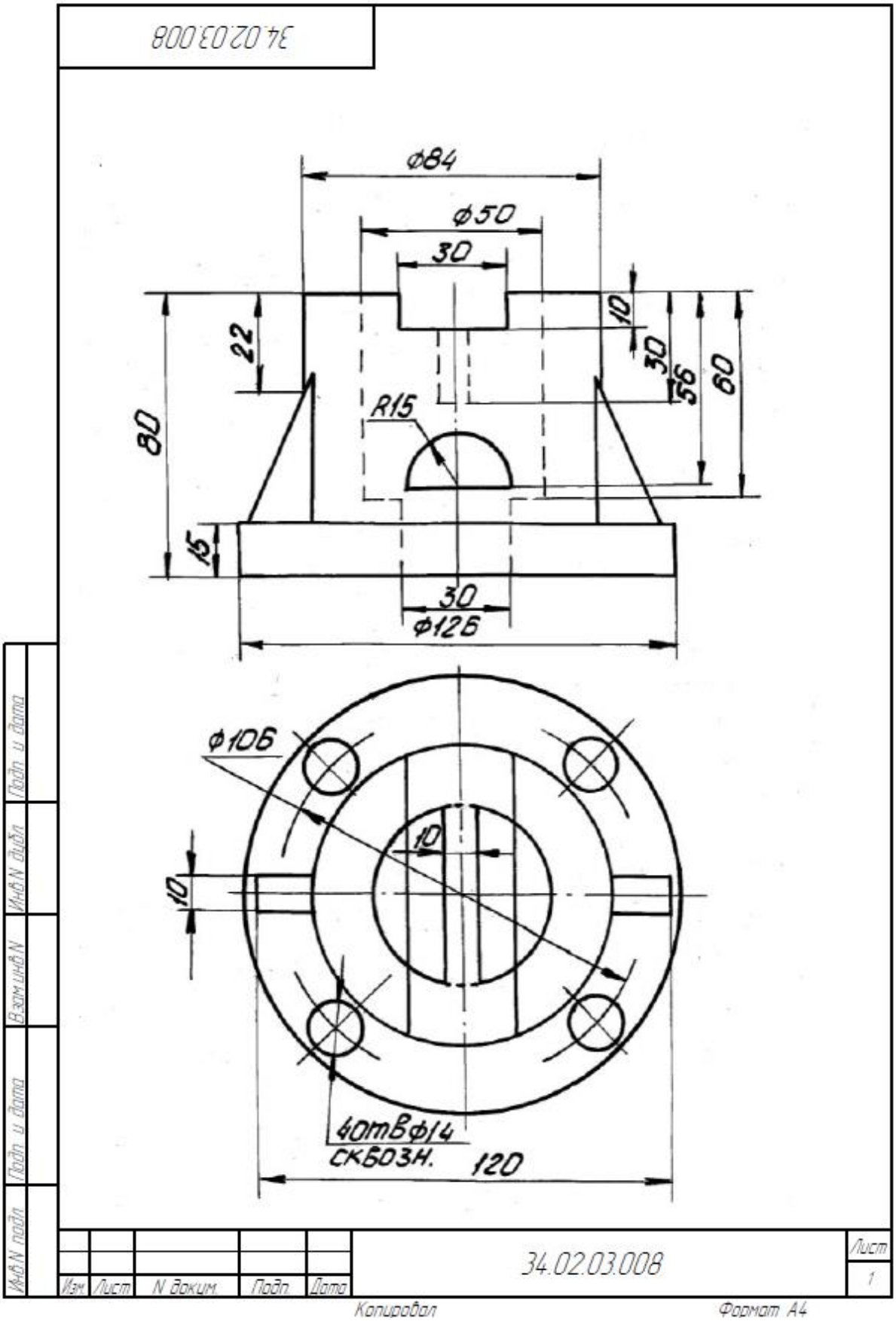


Рисунок 72 - Эскиз детали

Цилиндр с диаметром 126 мм условимся называть основанием детали. В основании расположены четыре сквозных отверстия. В верхнем цилиндре расположена полочка, ее тоже нужно будет учесть при построении.

Приступим к построению. При открытии программы нужно выбрать создание новой детали. Откроется интерфейс программы – система координат и рабочее меню. Выполним корпус детали с помощью операции вращения. Создадим эскиз, который будет вращаться вокруг оси. Перейдем к плоскости ZX. Для этого на нее нужно нажать и во всплывшем меню выбрать опцию «Создать эскиз».

Построим на расстоянии 15 мм от начала координат вспомогательную вертикальную прямую. Для этого сверху в меню «Отрезок» сначала можно построить горизонтальный отрезок нужной длины, затем, по нему – вертикальный. Из размеров детали его длина должна составлять 20 мм. Построим сразу горизонтальный длиной в 10 мм. Из конца этого отрезка ведем еще один вертикальный длиной 60 мм. Затем еще один горизонтальный на 17 мм. За ним горизонтальный отрезок в 65 мм, отступ в 21 мм и еще 15 мм вниз. Замыкаем с началом эскиза, вспомогательный отрезок от начала координат удаляем. Получаем эскиз корпуса детали.

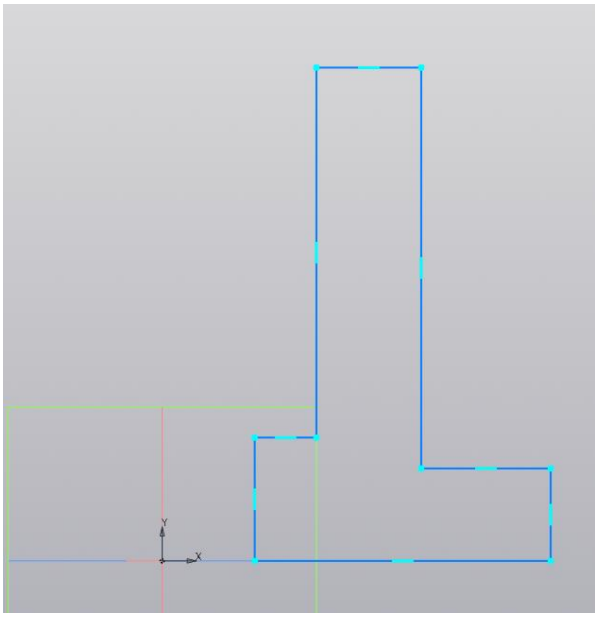

Рисунок 74 - Эскиз корпуса детали

Проведем вертикальный отрезок от начала координат, предварительно изменив стиль линии на «Осевая». Теперь можно применить к эскизу вращение.

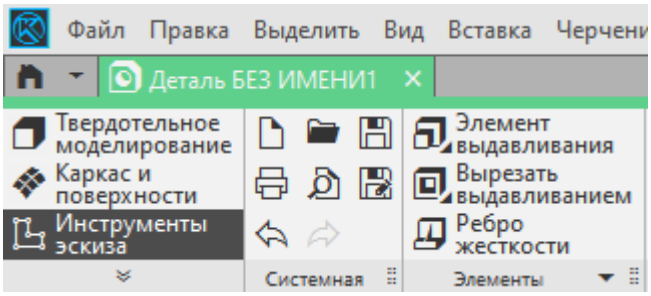


Рисунок 75 - Операция вращения

В верхнем левом углу находится панель, на ней – операции, применяемые к эскизам. Долгое нажатие мышкой на пункт «Элемент выдавливания» позволяет открыть еще одно меню, в котором нужно выбрать «Элемент вращения».

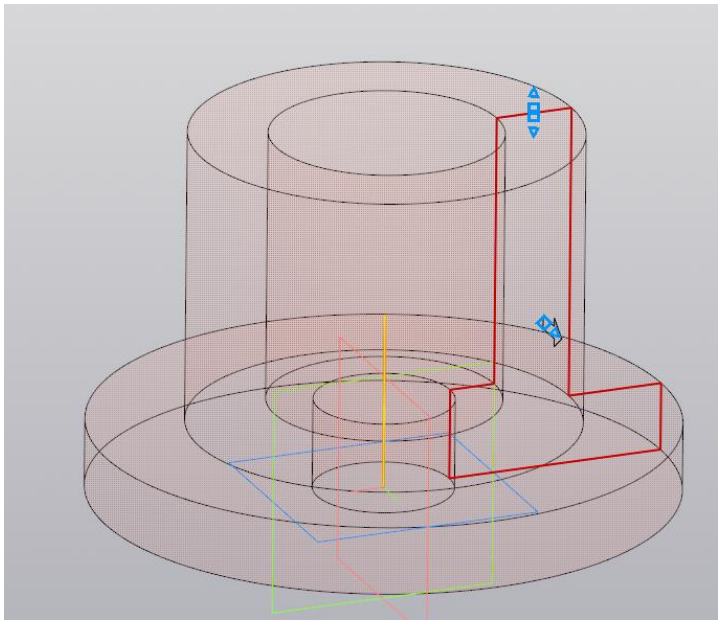

Рисунок 76 - Применения вращения к эскизу

Программа определила эскиз и ось автоматически и показывает предварительный результат. Для его применения в меню слева нужно нажать на галочку.

Построим полочку. Вернемся к плоскости ZX. Построим от начала координат вспомогательный отрезок высотой 50 мм. От его конца – горизонтальный на 5 мм. Потом проведем вертикальный отрезок на 30 мм вверх и еще один горизонтальный на 5. Выделим эти отрезки и применим к ним операцию «Зеркально отразить». В качестве центра симметрии можно выбрать центр вертикального отрезка. Относительно него создаем вторую половину полочки. Вспомогательный отрезок от начала координат удаляем.

Теперь приступим к выдавливанию. Выбираем операцию «Элемент выдавливания». Выдавливать будем в обе стороны, поэтому выбираем опцию «Симметрично». Расстояние выбираем такое, чтобы грани полочки находились в стенке цилиндра, не выпирая из нее. 60 мм будет достаточно. Выбираем, нажимаем на галочку.

Остались пазы, ребра жесткости и отверстия. Ребра жесткости построим с помощью соответствующей операции. Для ее применения необходимо построить только эскиз, определяющий форму внешнего края ребра и тело, к которому строится ребро. Нам достаточно построить только одну наклонную линию по размерам с эскиза-задания. После этого выбирается операция «Ребро жесткости», нажимается флажок «Симметричная толщина» и в качестве ее указывается значение 5 мм – в одну и в другую сторону. Применяем операцию, получаем ребро жесткости.

Чтобы получить второе ребро, находим операцию «Массив по сетке», долго удерживаем нажатой левую кнопку мыши и выбираем опцию «Зеркальный массив». Выбираем ребро жесткости и плоскость, относительно которой будет произведено зеркальное построение. Применяем операцию.

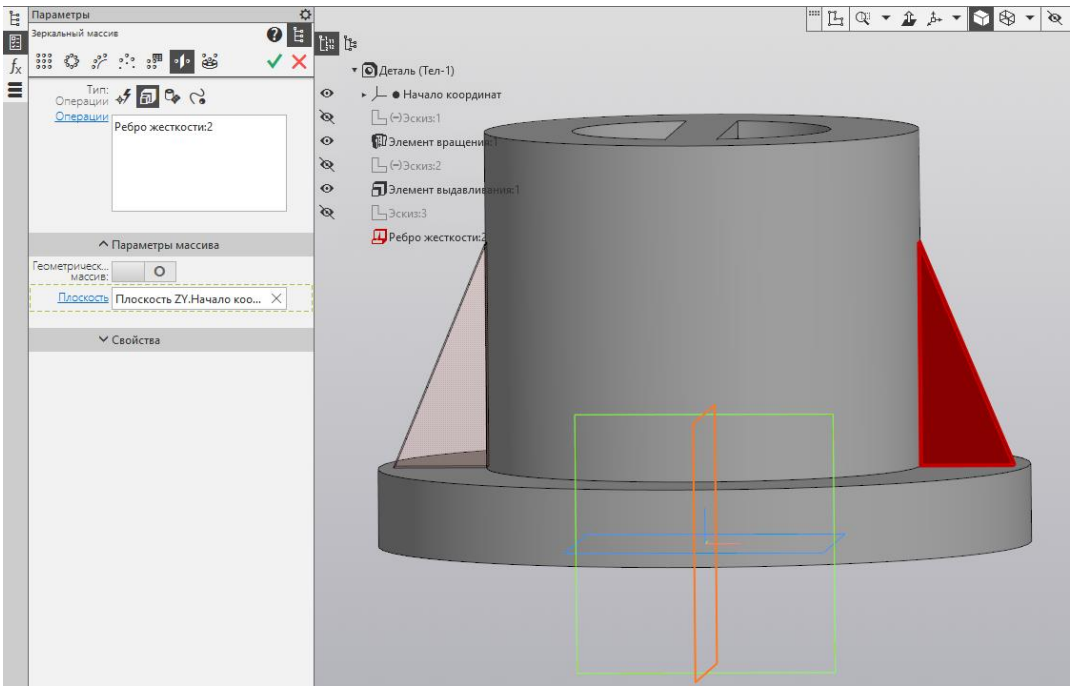


Рисунок 77 - Предпросмотр результатов зеркального массива

Создадим отверстия. Аналогично предыдущим построениям спроектируем сквозное отверстие нужных параметров в основании цилиндра. Чтобы не строить одно и то же четыре раза, применим операцию «Массив по концентрической сетке». Выберем отверстие как элемент для массива, выберем ось – ось симметрии всей детали, угол между экземплярами 90 градусов и количество экземпляров – 4.

Применим операцию. Теперь осталось только два паза – прямоугольный и полукруглый. Создадим их вместе. Снова вернемся к плоскости ZX и построим эскизы отверстий. Применяем операцию «Вырезать выдавливанием», настраиваем параметры «Через все» для двух направлений.

На этом построение 3D-модели завершено. Можно переходить к созданию чертежа.

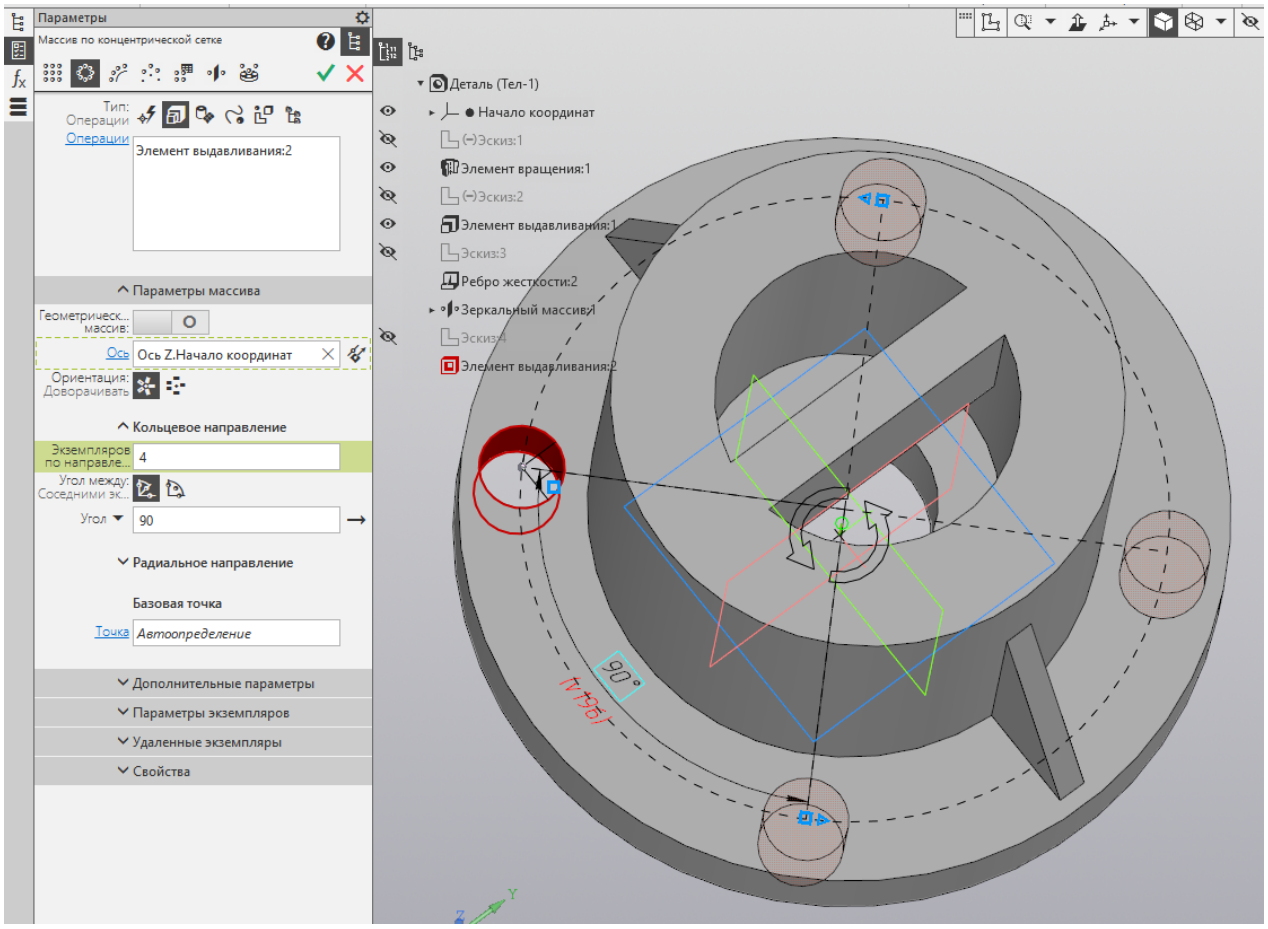


Рисунок 78 - Массив по концентрической сетке

Ищем справа сверху меню «Чертеж» и выбираем «Создать чертеж по модели». Откроется меню чертежа с уже готовым оформлением – по умолчанию чертеж выполняется на листе А4. Нормы и форматы выполнения чертежей можно уточнить в соответствующем ГОСТ. Сейчас мы можем добавить виды детали, воспользуемся тремя основными. Первым выберем вид спереди, затем вид сверху и вид слева. Разместим их в проекционной связи. Чтобы они поместились на лист А4, поставим масштаб 1:2.

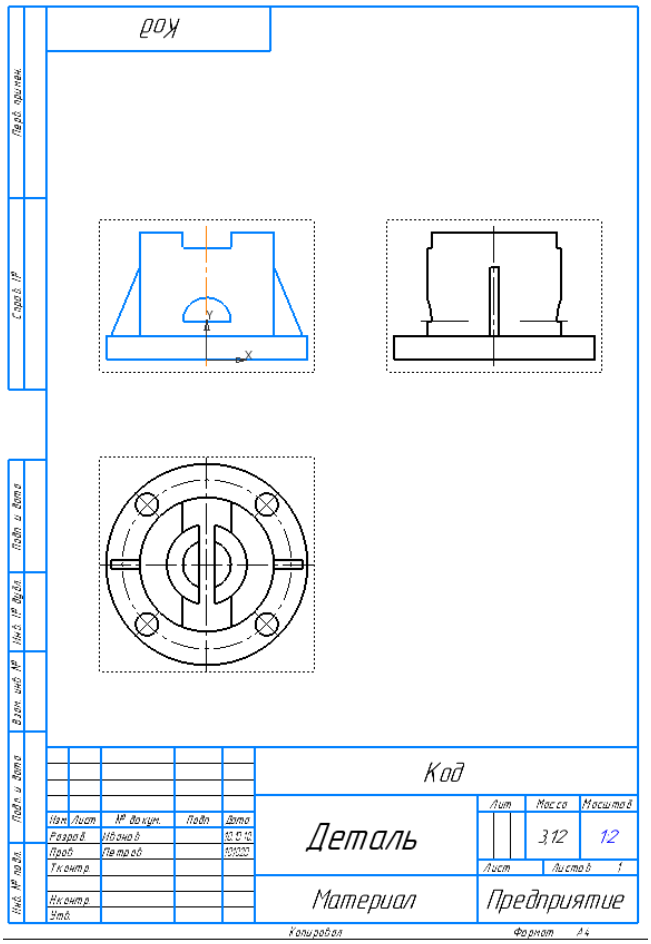


Рисунок 79 - Макет чертежа

Эти три вида не дают полного представления о геометрии детали, поэтому их стоит дополнить разрезами. Удалим вид спереди. Найдем обозначение «Линия разреза/сечения» и разместим его на виде сверху как на картинке. Разрезанный вид спереди поместим на место старого.

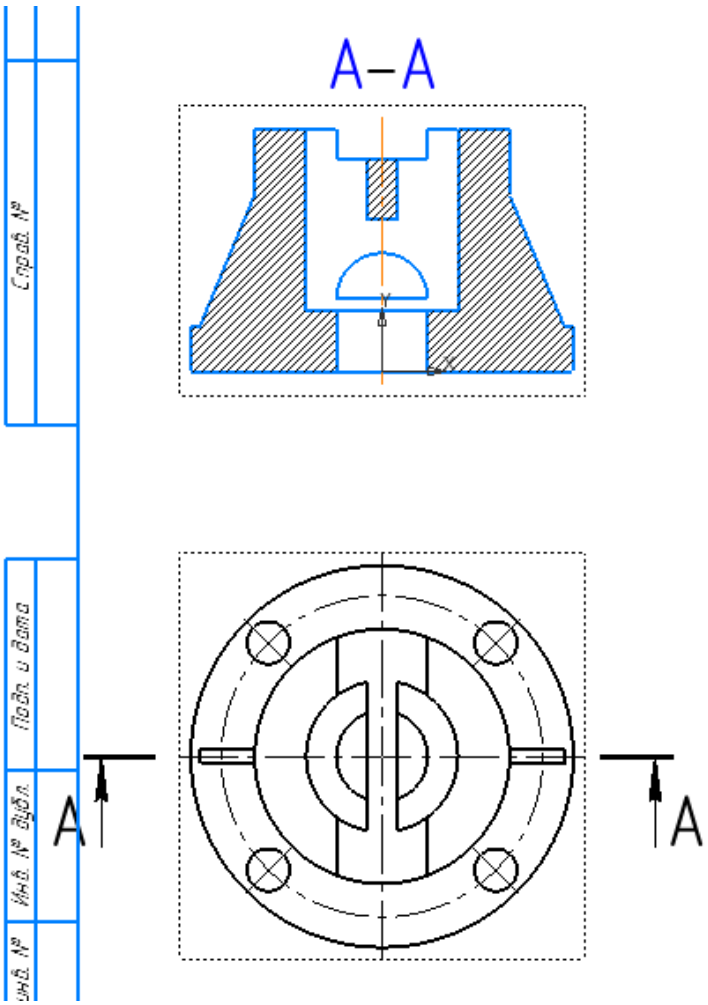


Рисунок 80 - Добавление фронтального разреза

Кликнем на стрелки правой кнопкой мыши и выберем «Разрушить». Удалим обозначение вида А-А над разрезом вместе со стрелками. Теперь нужно отредактировать получившийся вид – ребра жесткости не должны быть заштрихованы.

Далее нужно расставить размеры. Начинать следует с габаритных – ширины, длины, высоты. Размеры выставляются операцией «Линейный размер». Размеры, касающиеся цилиндров и окружностей, нужно обязательно обозначать значком диаметра или радиуса, который можно выбрать в меню размера.

Обо всех правилах постановки размеров рассказывает соответствующий ГОСТ. Тем не менее, они не должны быть расположены вплотную друг к другу, наезжать на линии рамки.

Еще один важный момент – осевые линии. Если деталь имеет какую-либо симметрию, обязательно обозначить осевую линию на ее оси. То же касается отверстий или любых других ее симметричных элементов. Примерный вид чертежа представлен ниже:

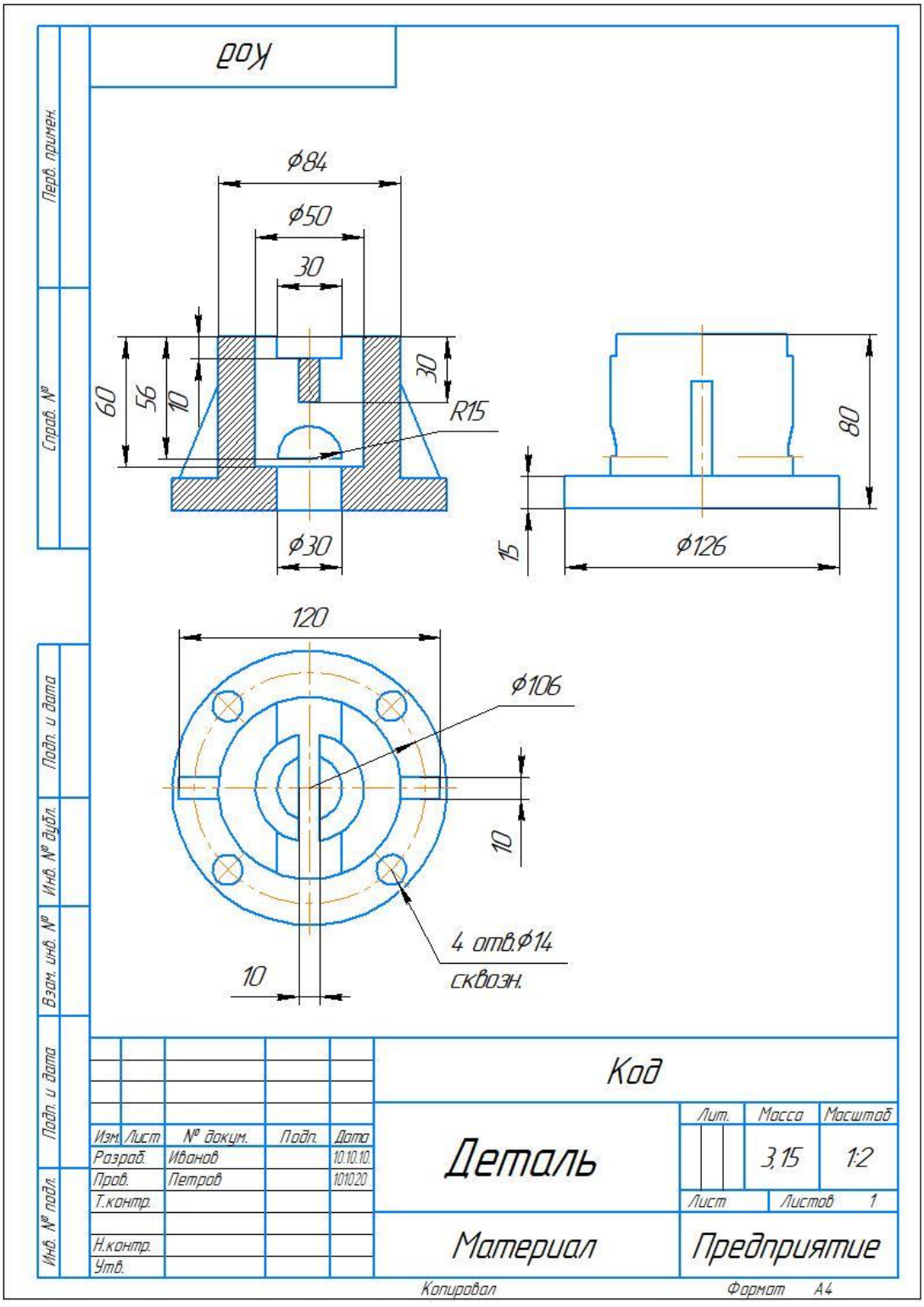


Рисунок 81 - Макет чертежа

Рассмотренный пример дает понятие о принципе работы в САПР, касающейся моделирования и построения чертежей. В рамках практической деятельности моделирование, скорее всего, пригодится больше и успех в нем достигается некоторым количеством практики.

Моделирование в робототехнике в основном пригождается при использовании технологий 3D-печати. Созданную модель достаточно перевести в специальный формат, настроить параметры печати и запустить ее. Таким образом и создаются корпуса устройств, их детали и другие компоненты.

## 6.4. Расчет кинематических цепей

Данный практический модуль предназначен для работы над расчетами кинематических цепей. Это полезно с инженерной точки зрения при наличии задачи расчета характеристик используемых технических решений. Цель раздела – дать больше информации касательно инженерных механизмов и методик их расчета.

Для понимания инженерной практики в сфере механики нужно затронуть тему зубчатых передач и передаточных механизмов. Часто задачи в сфере мехатроники и робототехники напрямую связаны с понятиями понижающих или повышающих передач. Как пример, можно рассмотреть ременные передачи. В основном они используются для того, чтобы добиться определенной частоты вращения какого-либо звена либо усилить его крутящий момент. Так же их называют понижающими или повышающими передачами.

Передача является повышающей, если на выходном звене в системе увеличивается частота вращения, а понижающей — если уменьшается. Чаще всего используют именно понижающие передачи, так как побочным эффектом уменьшения частоты вращения выходного звена является прямо-пропорциональное увеличение крутящего момента с небольшой погрешностью. Погрешность обусловлена тем, что часто КПД такой системы около 90-95%.

Обычно систему, которая работает как понижающая передача, называют редуктором, а систему, которая работает как повышающая передача, – мультипликатором. Параметр, который показывает, насколько система понижающая или повышающая, называют передаточным отношением системы. У редукторов передаточное отношение строго меньше 1, у мультипликаторов — строго больше.

Напомним, что простейшая передача состоит из двух компонентов, ведомого и ведущего. Передаточное отношение напрямую зависит от характеристик, которыми обладает ведущий и ведомый компонент. Если конкретизировать передаточное отношение, то это отношение диаметра (числа зубьев в случае с шестернями) ведущего компонента (колеса, звездочки), к диаметру (числу зубьев) ведомого компонента. Если же рассматривать многоуровневую систему, состоящую из нескольких передач, как на Рисунке 82, то можно определить так называемое «полное» передаточное отношение системы. Для этого необходимо взять произведение всех передаточных чисел всех передач в системе.

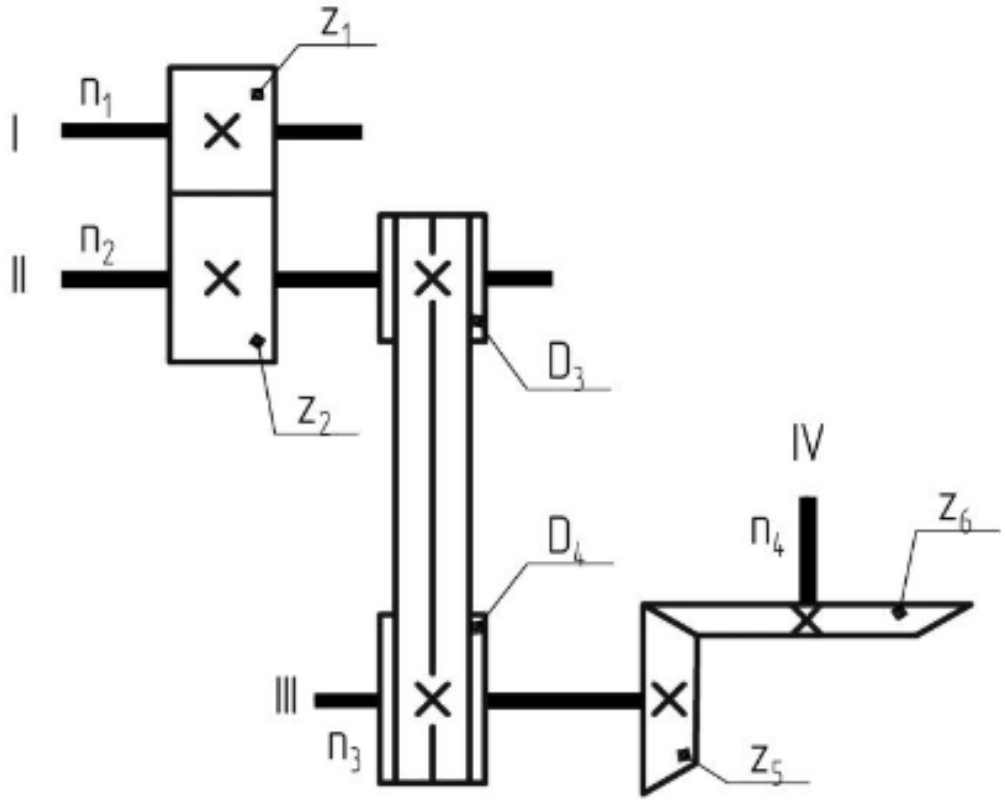


Рисунок 8229 - Многоуровневая понижающая система

Ознакомимся с решением задачи на многоуровневую понижающую систему или трехступенчатый редуктор.

Рассмотрим первую передачу, $n_1$ и $n_2$ – частоты вращения ведущего и ведомого вала соответственно. На валах жестко закреплены шестерни, $z_1$ и $z_2$ – количество зубьев ведущей и ведомой шестеренки соответственно. В задаче нам известна частота вращения выходного вала, и количество зубьев ведущих и ведомых шестеренок. Исходя из определения передаточного отношения можно обозначить формулу.

$n_2 = n_1 \times {}^{z_1}/_{z_2}$ – расчет частоты вращения ведомого вала первой передачи

Как только посчитана частота входного вала первой передачи, можно перейти ко второй, так как вал $n_2$ является ее ведущим компонентом, а $n_3$ — ведомым. В данном случае вторая передача представляет из себя ременное соединение, и как следствие передаточное отношение в ней будет определяться через диаметры $D_3$ и $D_4$ ведущего и ведомого шкивов, на которых закреплен ремень.

$n_3 = n_2 \times {}^{D_3}/_{D_4}$ – расчет частоты вращения ведомого вала второй передачи

Перейдем к третьей передаче, которая представляет из себя коническую зубчатую передачу, все работает по аналогии с первой. Так как $n_3$ теперь является частотой вращения ведущего компонента в этой передаче, а $z_5$ и $z_6$ количеством зубьев ведущей и ведомой шестеренки соответственно, можно достаточно просто посчитать $n_4$, частоту вращения ведомого вала всей системы.

$n_4 = n_3 \times {}^{z_5}/_{z_6}$ – частота вращения ведомого вала системы

Вот через такие, достаточно простые отношения можно посчитать все кинематические параметры передач любой сложности. Однако необходимо внести одну ремарку, в данном расчете не учитывается КПД зубчатой передачи, о котором говорилось ранее. На практике частоты вращения

ведомых компонентов, а также их крутящие моменты будут отличаться, но незначительно.

## 6.5. Проектная работа

В этом разделе рассматривается важность проектной работы, ее правильного оформления и сопровождения. Здесь изложены основные ее этапы и правила выполнения. Умение правильно организовать деятельность оказывается невероятно важным в любой дальнейшей работе, и проекты – один из многих способов научиться этому.

Вне зависимости от направления работы, способность ставить цели и задачи к ним, достигать их, выражать их корректно устно и письменно, грамотно демонстрировать ход и результаты своей работы зачастую имеет критическое значение.

### Цель

Любая работа должна иметь смысл, цель, к которой стремится ее исполнитель. Это может быть разработка чего-либо, исследование явлений, написание программы. Правильная формулировка цели сама по себе задает некоторый вектор дальнейшего развития работ по ее достижению.

### Задачи

Как правило, просто взять и достичь поставленной цели с первого шага невозможно. Для того, чтобы систематизировать фронт работ, назначают задачи – подцели по каждому из нескольких направлений деятельности, которые предусматривает работа. В случае робототехники задачами будут действия, связанные с проектированием, изготовлением составных частей, подбором компонент, написанием управляющих программ, сборкой конечного изделия, его тестами, проверками и испытаниями.

Если проект достаточно обширный и многопрофильный, что можно отнести к робототехнике, задачи рекомендуется разбить на подзадачи – набор необходимых для выполнения задачи целей.

К постановке цели и задач следует относиться очень ответственно. Они могут изменяться в ходе работы, но чем больше времени и размышлений будет потрачено на осмысление, тем проще потом будет выстроить свою деятельность или деятельность команды.

В случае командной работы уже на этапе постановки задач стоит разделить обязанности. Если все участники команды будут заниматься одним и тем же, работу будет очень сложно организовать. В случае, если у проекта много исполнителей, можно назначить координатора – человека, в чьи обязанности будет входить контроль выполнения поставленных задач и подзадач.

Не стоит пытаться описать всю работу досконально подробно – по настолько подробному плану не получится сделать все, при несоблюдении одного пункта придется корректировать остальные. Действия должны быть обозначены вполне четко, но без прямых указаний по их выполнению.

**Литературный обзор**

Эта часть работы очень пригодится при дальнейшем обучении, как только возникнет необходимость писать какие-либо отчетные работы. Основная задача обзора – собрать как можно больше информации по теме проекта, разобрать аналоги и существующие прототипы. В случае работы в команде с обзором должны ознакомиться все участники, чтобы не войти в ступор, не повторить уже сделанную кем-то работу или не упереться в недостаток знаний в нужной области.

Дополнительно к обзору полезно составить список литературы, к которой можно будет обращаться по мере выполнения проекта. Как правило, это могут быть статьи и учебники, в которых можно уточнить теоретические, иногда даже практические моменты.

На этапе литературного обзора можно корректировать ожидания от проекта и сформировать его актуальность. Актуальность задачи – это причина, по

которой этим проектом занимаются. Говоря об этом, нужно обязательно сказать, какая проблема привела к теме проекта, и как прогнозируемый результат его выполнения сможет помочь эту проблему решить. Для робототехнических проектов актуальность часто включает в себя потребность в автоматизации рутинных задач.

### Списки оборудования

В случае, если проект практический, то есть его результатом является физическое устройство, выполняющее определенную функцию, то в процессе его исполнения возникнет необходимость в закупке оборудования и комплектующих. Такие списки, сметы, очень помогают систематизировать этот процесс, вести учет задействованных компонентов и планировать приобретение следующих.

Как правило, в таких списках указывается наименование оборудования, требуемое количество (кстати, полезно закупать запасные комплектующие), его стоимость и название магазина (либо ссылка), где его можно приобрести. Удобно отмечать, в наличии ли нужная комплектующая, насколько скоро ее нужно закупить. Отдельно можно вести учет аналогов в случае, если рассматриваются разные варианты исполнения чего-либо – какой-нибудь конструкции или детали.

### План работы

План работы будет служить картой исполнения проекта. Нужно расставить приоритеты, возможно, обозначить какие-то сроки исполнения. Будет большой удачей их придерживаться.

План включает в себя и подготовку к выполнению задачи, и примерное описание действий, которые должны привести к ее успешному завершению. Наверное, план – это самая непостоянная часть проекта, но она позволяет предугадать и предсказать его развитие, что нужно для правильной оценки возможности получения желаемого результата.

Отрицательный результат – это тоже результат. Если цель оказалась недостижимой, стоит провести рефлексию – определить причины такого результата, разобраться, что к нему привело, какие действия либо же бездействия привели к неудаче.

**Презентация**

Каков бы ни был представляемый проект, наибольшее внимание аудитория будет уделять презентации и тому рассказу длиной 5-7 минут, который подготовит выступающий. Вникать во все мелочи и детали работы, особенно если она объемная или содержит большое количество теоретических выкладок, никто не будет.

Правильная презентация деятельности позволяет внешним людям правильно и по достоинству ее оценить. Рассмотрим основные требования к презентациям, которые вытекают из ГОСТ.

Первый слайд всегда должен быть титульным. На нем должна находиться следующая информация:

- Учреждение, в котором выполнялась работа (школа или университет, например);
- Название работы;
- Имена авторов;
- Имена руководителей;
- Контактные данные;
- Место и дата выполнения проекта.

Далее идет введение в работу. Называется ее цель, задачи и краткий вывод об их выполнении. Все слайды, кроме титульного, нумеруются. Основная часть презентации идет за введением, и, как правило, содержит несколько подразделов.

Стиль презентации должен быть единообразным, иллюстрации подписаны и в хорошем качестве, фон лучше брать светлый. Что касается шрифтов – выбирается любой читаемый. Для основного текста размер шрифта может варьироваться от 24 до 28, для заголовков – от 28-32 и более.

Последние слайды презентации обязательно содержат заключение, в котором еще раз объявляются результаты выполнения работы. Самый последний слайд, обычно, содержит благодарность за внимание к работе.

В целом презентация должна содержать необходимый максимум для понимания сути проекта, используемых средств достижения целей и результатов всей работы.

**Устное выступление**

Рассказ презентации сопровождается устным докладом, длительность которого как правильно жестко регламентирована. За эти 5-10 минут, которые участники получают на выступление, грамотно поставленной и подготовленной речью нужно изложить все, что было сделано, зачем это было сделано, и какими способами участники добились таких результатов.

Речь полагается строить соответственно презентации, акцентируя внимание на самостоятельной работе участников, на конкретно их вклад в проект. Начинается все со введения, где главные черты – актуальность и цель работы. Основная часть работы должна занимать наибольшую часть речи. Подготовить такое выступление, после которого ни у кого не останется вопросов, конечно, невозможно, но, тем не менее, нужно сконцентрировать всю полезную информацию о работе в эти несколько минут. В заключении хорошим тоном является перефразированное повторение полученных в работе результатов, сравнение их с поставленными целями и еще одна отсылка к актуальности выполняемой задачи.

**Тезисы**

Если работа отправляется на какой-либо конкурс или конференцию, организаторы вправе организовать отбор на эти мероприятия, попросив участников прислать тезисы своих работ. Что это такое?

Тезисы – сжатое описание сделанного проекта, которое опять же должно содержать организацию, на базе которой выполнена работа, указание авторов, тему, актуальность, цели, задачи, действия для их достижения и краткие выводы.

Суть тезисов в том, что это, как правило, документ размером в 1-2 страницы, который позволяет читающему быстро оценить адекватность и примерно качество проекта для того, чтобы отобрать его для дальнейшего участия или же отправить отказ.

**Выводы**

Проектная деятельность – крайне интересное направление работы, которое способно дать практических навыков и теоретических знаний даже больше, чем любой прочитанный слушателю курс. Однако такое возможно только в том случае, когда подход к техническому творчеству ведется крайне ответственно, работа выполняется поэтапно и имеет не только цели, но и план их реализации.

Выполнение конкурсных и проектных задач не только дает возможность получить отличное образование в высшей школе, но и развивает важные социальные навыки: стратегическое мышление, ответственность, умение презентовать и защитить свою идею перед коллегами и экспертами, креативность и многие другие.