# Содержание

[Содержание 2](#_Toc104497085)

[Введение 3](#_Toc104497086)

[1 Аппаратная часть 4](#_Toc104497087)

[2 Программная часть 6](#_Toc104497088)

[2.1 описание программной части 6](#_Toc104497089)

[2.2. Программа микроконтроллера 7](#_Toc104497090)

[2.2.1. Применяемые технологии 7](#_Toc104497091)

[2.2.3 Модуль работы с сервоприводами 10](#_Toc104497092)

[2.2.4 Модуль работы с UART 13](#_Toc104497093)

[2.3. Управляющая программа 17](#_Toc104497094)

[2.3.1 Применяемые технологии 17](#_Toc104497095)

[2.3.2 Компьютерное зрение, захват движения 18](#_Toc104497096)

[2.3.3 Разработка программы 19](#_Toc104497097)

[3. Тестирование 23](#_Toc104497098)

[3.1 Описание тестового стенда 23](#_Toc104497099)

[3.2 тестирование программы 25](#_Toc104497100)

[Заключение 26](#_Toc104497101)

[Список литературы 27](#_Toc104497102)

Список сокращений

МК – микроконтроллер

УП – управляющая плата

ШД – шаговый двигатель

АЦП - аналого-цифровой преобразователь

ШИМ/PWM – широтно-импульсная модуляция

GPIO – конфигурируемый интерфейс ввода-вывода микроконтроллера

IDE – среда разработки

# Введение

Можно с уверенностью заявить, что мы живем в век высоких технологий. Сегодня мы являемся свидетелями событий, которые несколько десятилетий назад считались невозможными, были показаны лишь в фантастических фильмах и книгах писателей фантастов. От человекоподобных роботов и квадрокоптеров до знаменитого марсохода – возможности кажутся безграничными.

Хочется особо подчеркнуть развитие технологий в области искусственного интеллекта и машинного обучения. Компьютерное зрение – это пример новаторской технологии, которая потенциально может изменить способ управления машинами. В данной работе представлен пример использования машинного зрения для захвата движений рук человека и передачи управляющих сигналов на сервоприводы робота-манипулятора.

1 Анализ предметной области

* 1. Описание предметной области

Задачей робототехники является создание робототехнических систем различного назначения. В основе робототехники лежат такие науки, как кибернетика и механика.

При создании роботов, человек всегда ориентировался на воплощение своих собственных физиологических возможностей. Ведь одной из целей создания роботов является замена труда человека на тяжелых физических работах, а также на работах, выполняемых в условиях опасных сред.

Одними из наиболее распространенных типов роботизированных систем, являются роботы-манипуляторы, специальные устройства выполняющие функции человеческой руки. Такие роботы нашли широкое применение в таких сферах, как автомобилестроение, электротехника, строительство, химическая промышленность, фармацевтика и др.

На рисунке TODO представлена типичная функциональная схема робота

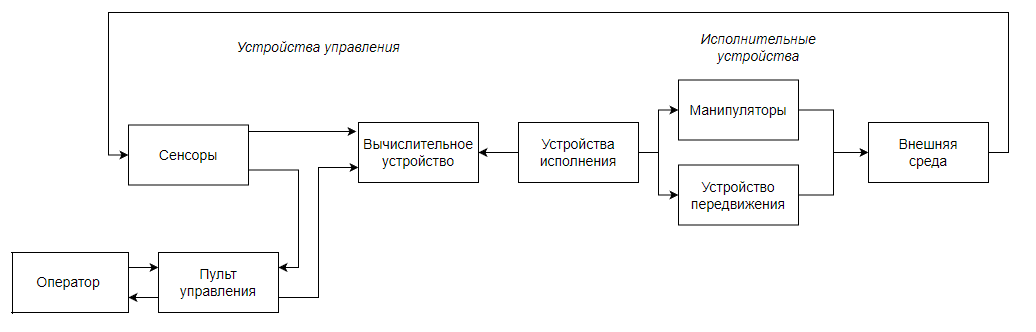


Рис. 1.

По схеме видно, что выделяют две основные группы устройств:

* Устройства управления, которые осуществляют взаимодействие с пользователем, прием и обработку команд, считывание показаний сенсоров и датчиков и передачу управляющих команд на устройства исполнения;
* Исполнительные устройства, осуществляют взаимодействие с внешней средой.

Одним из современных направлений робототехники является внедрение систем искусственного интеллекта и машинного обучения для повышения производительности и качества выполняемых работ. Примерами могут служить системы распознавания и классификации объектов, применяемые при сортировке мусора, либо для расстановки электронных компонентов на печатной плате. А также, различные способы, упрощающие взаимодействие оператора и исполнительных устройств, распознавание движений либо жестов.

* 1. Анализ рынка существующих систем

На рынке представлено огромное количество как роботов-манипуляторов, так и систем управления. Однако, существенным минусом является то, что подавляющее большинство из имеющихся устройств производятся за рубежом, основными производителями являются Китай, Япония и Германия. Отечественные же решения на рынке представлены слабо.

Одним из лидеров на рынке роботизированных систем для обучения школьников и студентов является компания VEX Robotic, на нее мы обратили свое внимание, когда выбирали аппаратное решение для реализации поставленных задач, к сожалению, качество документации оставляет желать лучшего, а также сомнительно качество исполнения, одна из управляющих плат вышла из строя после подключения к компьютеру.

* 1. Вывод

Исходя из вышеизложенного, было принято решение применить инженерный подход, и разработать аппаратно-программный комплекс самостоятельно. В качестве робота-манипулятора была выбрана модель, находящаяся в свободном доступе, ее элементы были напечатаны на домашнем 3д-принтере, также была разработана и электронная плата. Разработано программное обеспечение для управляющей платы и программа управления для компьютера.

2. Проектирование

2.1 Общая структура проекта

Для реализации поставленных задач, проектирование было решено поделить на несколько этапов.

С этапами проектирования можно ознакомиться на рис.



2.2 Выбор аппаратной платформы

2.2.1 Назначение

Под аппаратной платформой в данном случае понимается вид робота, подходящего под наши задачи. Определение степеней свободы, необходимых характеристик.

2.2.2. Функциональные требования

Одним из основных требований к выбору робота-манипулятора было наличие не менее 5 степеней свободы. Это обусловлено важностью апробации как можно большего числа жестов.

Также, не менее важным требованием была легкость обслуживания и ремонтопригодность робота, так как в процессе построения и отладки велик риск повреждения функциональных узлов.

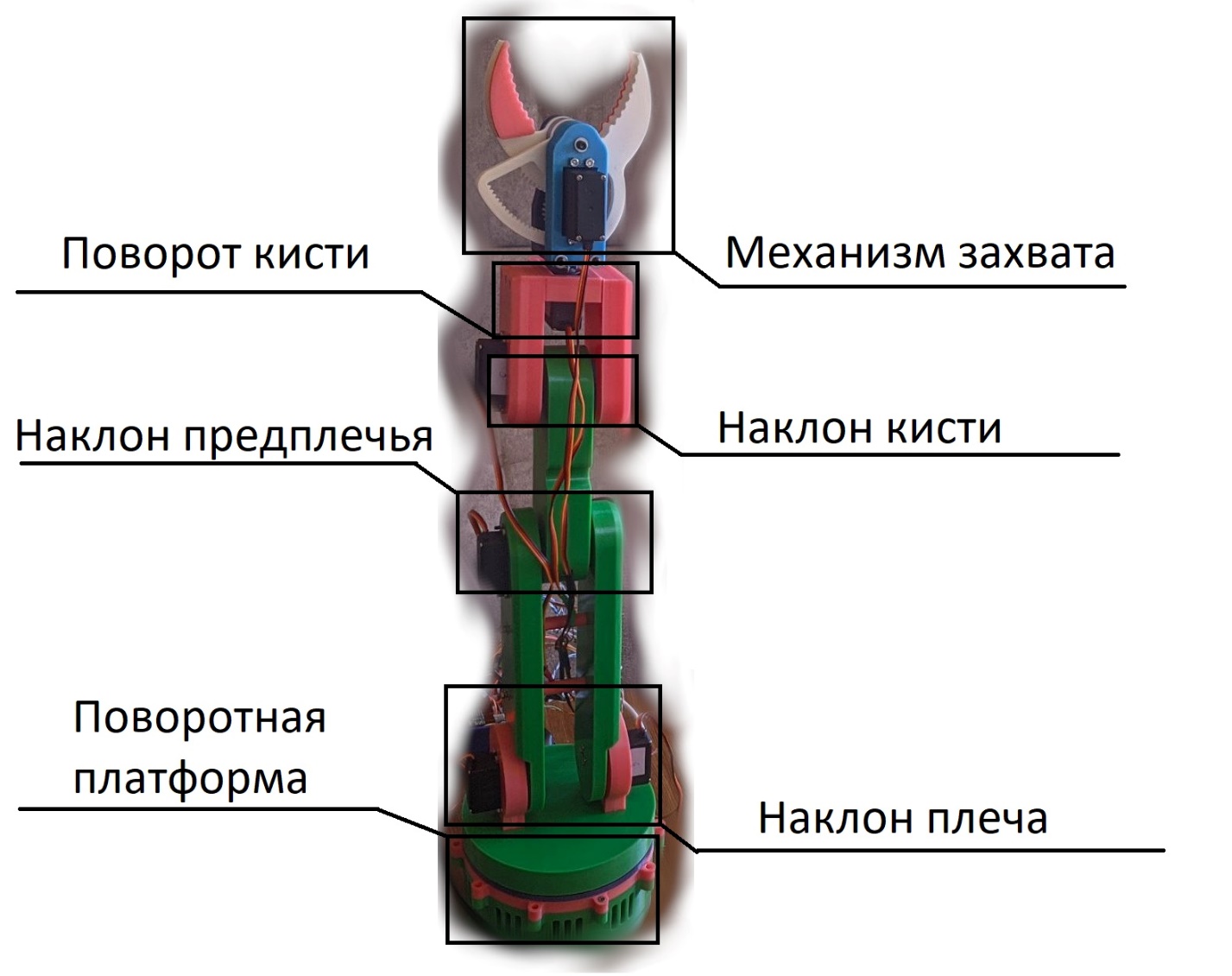
Важно отметить и легкость сборки, доступность комплектующих, в проекте не должно быть дорогостоящих элементов.

2.2.3. Реализация

Исходя из вышеизложенных требований, было принято решение выбрать свободно распространяемую модель робота-манипулятора. Ее структурные элементы были распечатаны на 3д-принтере. Основными функциональными узлами робота являются:

* Механизм захвата;
* Механизм поворота кисти на 180 градусов;
* Механизм наклона кисти;
* Механизм наклона предплечья;
* Механизм наклона плеча;
* Поворотная платформа.

Ознакомиться подробнее с функциональными элементами робота-манипулятора можно на рис.



2.3 Электронная часть

2.3.1 Электронная часть – назначение

Электронная часть проекта включает в себя различные электронные компоненты – модули. Их взаимосвязь обеспечивает функционирование всего устройства. Разберемся со структурой и выделим основные элементы:

1. Управляющая плата – предназначена для коммуникации между электронными модулями и микроконтроллером, также на ней расположены выводы для подключения моторов и другой периферии.
2. Сервоприводы
3. Драйвер
4. Шаговый двигатель
5. Источники питания
6. Проводники

Представим электронную часть проекта в виде схемы:



Рис.

Как видно из приведенной выше схемы, центральным элементом системы является управляющая плата (далее УП). Выбор УП и микроконтроллера, являющегося ее основой, был непростым решением. Были рассмотрены различные решения доступные на рынке, такие как VEX Robotics и LEGO Mindstorms. Мы пришли к выводу о том, что готовые решения содержат существенные недостатки и не удовлетворяют нашим требованиям.

К недостаткам можно отнести:

1. Невозможность вмешательства «извне». Прошивка контроллеров, как правило, закрыта и не допускает модификации;
2. Ни одна из существующих УП не поддерживает совместное подключения шагового двигателя и сервоприводов, из этого следует, что реализация проекта невозможна в полной мере, либо требует существенной переработки аппаратной платформы;
3. Использование производителями «специфических» протоколов для связи с исполняющими устройствами, требуется существенное углубление в аппаратно-программную начинку, что опять же невозможно, см. п. 1;
4. Зачастую отсутствует возможность подключения периферии, такой как экраны, дополнительный набор клавиш – клавиатуры;
5. Отсутствие документации, ее низкое качество, либо устаревание.

Изложенные выше проблемы «ставят крест» на использование УП стороннего производителя. Было принято решение разработать собственную УП, которая бы удовлетворяла нашим требованиям.

2.3.2 Управляющая плата – функциональные требования.

Для того, чтобы решить поставленные задачи, к УП предъявляется ряд требований:

1. Наличие на плате достаточного количества разъемов, для подключения 6 сервоприводов;
2. Коммуникация с персональным компьютером по средствам UART;
3. Поддержка интерфейса i2c;
4. Стабилизированное питание;
5. Размещение на плате драйвера ШД и разъема для подключения ШД;
6. Наличие экрана для вывода сообщений пользователю;
7. Наличие органов управления приводами;
8. Возможность подключения дополнительных органов управления, клавиатур;
9. Возможность подключения дополнительных датчиков, например концевых выключателей, датчиков расстояния и т.д.
10. Модульность конструкции;
11. Простота в эксплуатации, документированность.

2.3.3. Управляющая плата – выбор микроконтроллера

Базисом любой УП является микроконтроллер. Эти микросхемы, предназначенные для управления электронными устройствами, лежат в основе практически любой современной техники. На своем кристалле они содержат процессор, ОЗУ, ПЗУ, порты ввода-вывода и различную периферию, таймеры. АЦП и прочие компоненты.

Выбор микроконтроллера – задача не из простых. На рынке представлено огромное количество различных решений в разных ценовых сегментах, отличающееся между собой производительностью, периферией, средами разработки и качеством доступной документации.

Одним из наиболее популярных микроконтроллеров является AVR Atmega 328p, который является «сердцем» известной платформы для начинающих радиолюбителей – Arduino.

Остановимся на технических характеристиках микроконтроллера Atmega 328p, они представлены в таблице

Таблица

Технические характеристики

|  |  |
| --- | --- |
| Наименование | Значение |
| Питание | 1,8 – 5,5 В |
| Разрядность | 8 Бит |
| Тактовая частота | 16 МГц |
| Объем ОЗУ | 2 Кбайт |
| Объем ПЗУ | 32 Кбайт |
| Объем EEPROM | 1 Кбайт |
| Интерфейсы | I2c, spi, uart |
| Периферия | Pwm, wdt |
| АЦП | 6 каналов по 10 бит |
| Количество входов-выходов | 23 |

Именно его мы выбрали для реализации нашего проекта. Этому способствовало:

1. Большое количество портов ввода-вывода;
2. Наличие интерфейсов UART, I2C, SPI;
3. Наличие АЦП достаточной разрядности;
4. Качественная документация;
5. Большое количество пользовательских библиотек для работы с различной периферией;
6. Многочисленные электронные модули, представленные на рынке разработаны именно под Arduino;
7. Возможность работы напрямую с «железом», в обход высокоуровневой библиотеки, работа с регистрами, ассемблерные вставки и т.д.;
8. Компилятором является кроссплатформенный GCC Toolchain, следовательно разработчик может игнорировать IDE производителя и вести разработку в любой устраивающей его IDE.
9. Большое сообщество разработчиков.

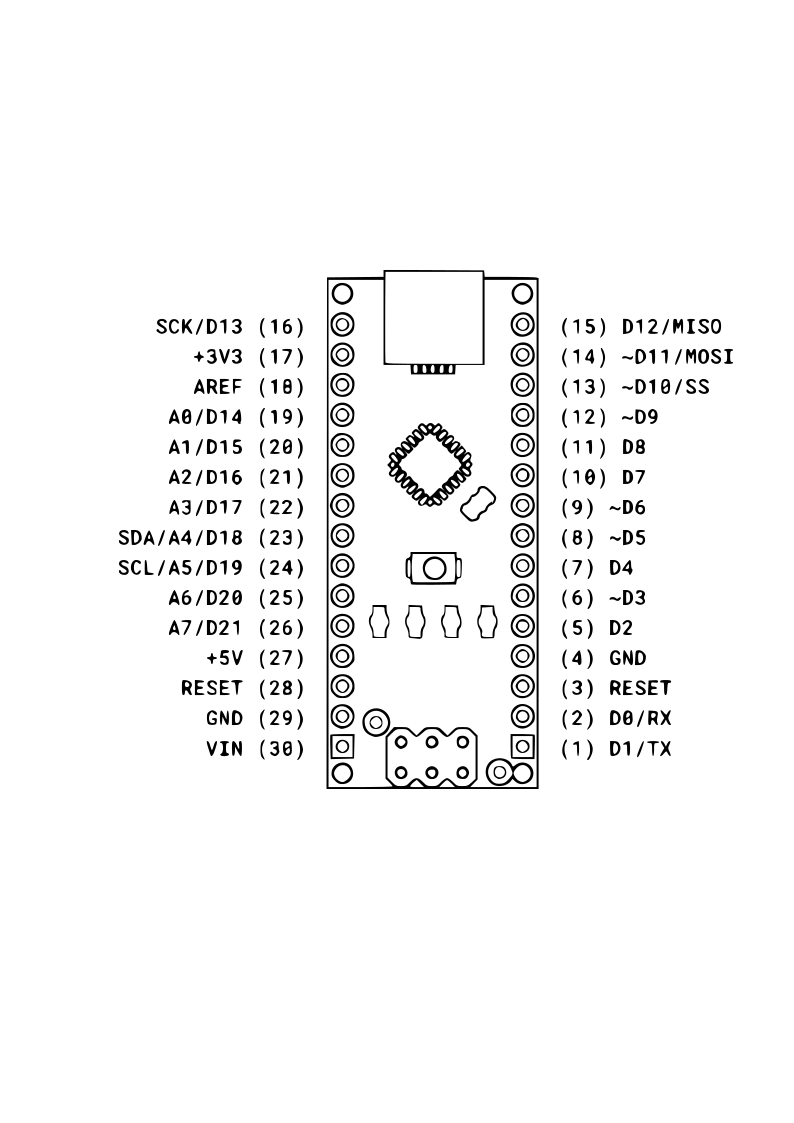
Остановимся на производительности микроконтроллера. Для человека недостаточно ознакомленным с миром микроконтроллеров может показаться что данное устройство обладает весьма скромной производительностью. Это действительно так, особенно учитывая современные тенденции развития вычислительной техники. Однако, такой производительности достаточно для решения поставленной нами задачи, в проекте реализована многозадачность, а также уделялось особое внимание оптимизации (см. раздел Оптимизация).

2.3.3.1 Микроконтроллер – конфигурация интерфейса ввода-вывода

Интерфейс ввода-вывода общего назначения (далее GPIO) осуществляет связь между микропроцессором и периферийными устройствами.

Считаю важным определиться с назначением каждого из доступных выводов GPIO – пинов на этапе проектирования.

На рисунке представлена распиновка (цоколёвка) выводов микроконтроллера.



Как видно на рисунке всего на отладочной плате имеется 30 выводов различного назначения. Не все они являются конфигурируемыми, часть отвечает за питание платы, аппаратный сброс (RESET), вывод для подачи опорного напряжения (AREF), и выводы для питания подключаемой периферии (5V, 3.3V).

Конфигурируемые выводы GPIO могут быть цифровыми и аналоговыми, которые соответствуют D и A на схеме. Аналоговые выводы подключены к аналого-цифровому преобразователю (далее АЦП).

Разберем какие выводы необходимы для подключения необходимой нам периферии. В нашем проекте не используется драйвер сервоприводов, соответственно необходимо занять 6 цифровых выводов – для каждого серводвигателя своя сигнальная линия.

Управляются двигатели с помощью широтно-импульсной модуляции (далее ШИМ/PWM). Несмотря на то, что каждый из цифровых выводов может эмулировать работу ШИМ программным способом, мы будем задействовать только аппаратный ШИМ. Выводы аппаратного ШИМ обозначены знаком «~».

К нашему контроллеру будут подключены 5 потенциометров, с их помощью осуществляется управление каждым из приводов. Как вы можете заметить, мы используем 5 потенциометров для управления 6ю двигателями, ошибки здесь нет, за подъем плеча манипулятора отвечают одновременно два привода управляемые одним органом управления. Потенциометры представляют из себя делители напряжения и подключаются к аналоговым выводам МК.

Также стоит отметить и задействованный в проекте интерфейс I2C, коммуникация между устройствами I2C осуществляется по 2м проводам, выводам I2C соответствуют аналоговые пины A4 и A5.

Интерфейс UART используемый для коммуникации с компьютером задействует выводы D0 и D1.

Как можно заметить, остался 1 свободный аналоговый вывод, отведем его для подключения резистивной-клавиатуры на 5 кнопок.

Для драйвера шагового двигателя требуется как 3 свободных цифровых пина.

Сведем данные в таблицу.

Таблица

Используемые выводы GPIO и их назначение

|  |  |
| --- | --- |
| Вывод GPIO | Назначение |
| A0, A1, A2, A3, A6 | Органы управления – потенциометры |
| A4, A5 | I2C – oled-Дисплей |
| A7 | Резистивная клавиатура |
| D0, D1 | UART |
| D3, D5, D6, D9, D10, D11 | Серводвигатели |
| D7, D8, D12 | Драйвер ШД |

2.3.4 Шаговый двигатель и драйвер шагового двигателя

Считаем необходимым остановиться подробнее на теме шаговых двигателей.

Шаговый двигатель – бесщеточный электродвигатель, ток, подаваемый на обмотки статора, вызывает фиксацию ротора. Следовательно, последовательная активация обмоток статора вызывает угловые перемещения – шаги.

Со строением шагового двигателя можно ознакомиться на рисунке

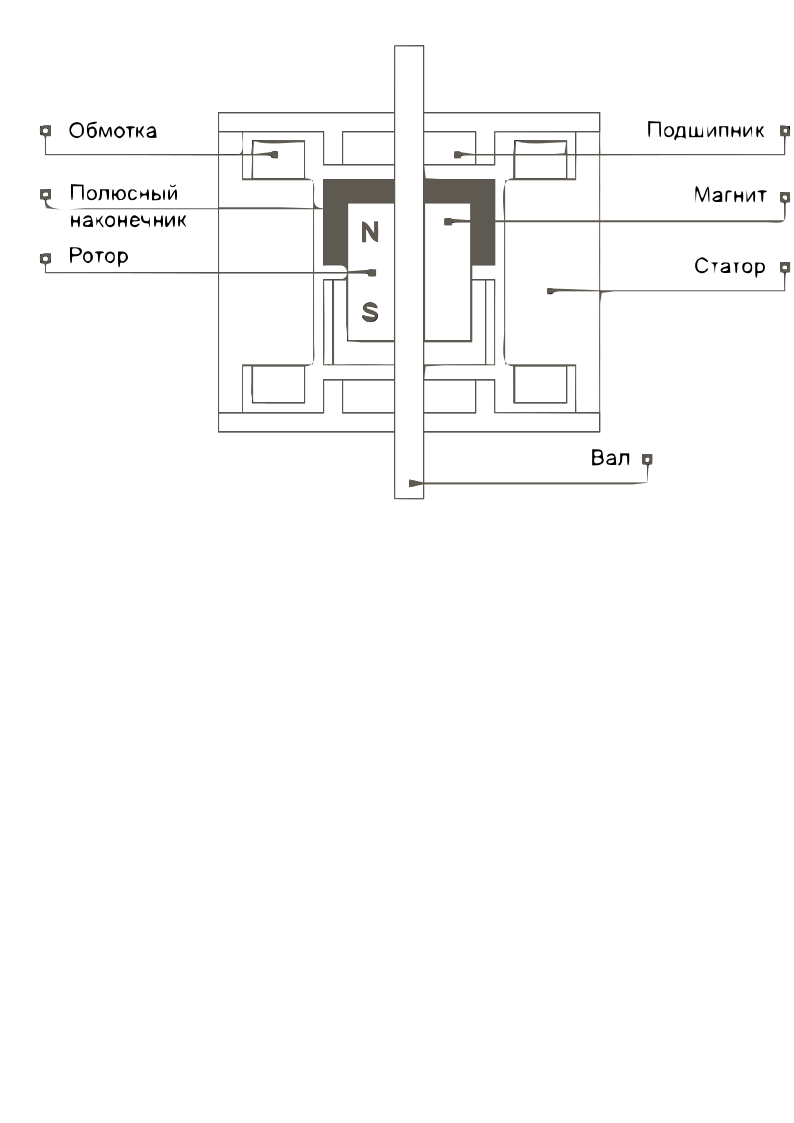


Рис Шаговый двигатель – осевой разрез.

На рынке представлено огромное количество шаговых двигателей различных размеров и форм-факторов. В нашем проекте используется двухфазный двигатель типоразмера nema 17, с длинной вала 24 мм.

Для управления током, подаваемым на обмотки статора, применяют специальное электронное силовое устройство – драйвер ШД. На плате драйвера расположен микроконтроллер, который принимает цифровые сигналы – импульсы, и на их основании управляет сильноточными и высоковольтными обмотками ШД, поочередно запитывая их. От величины подаваемого тока на обмотки зависит крутящий момент двигателя, но также увеличивается и нагрев как драйвера, так и ШД.

Драйверы сильно различаются по сложности. В нашем проекте используется относительно простой и широко распространенный драйвер А4988. Ознакомится с ним можно на рис

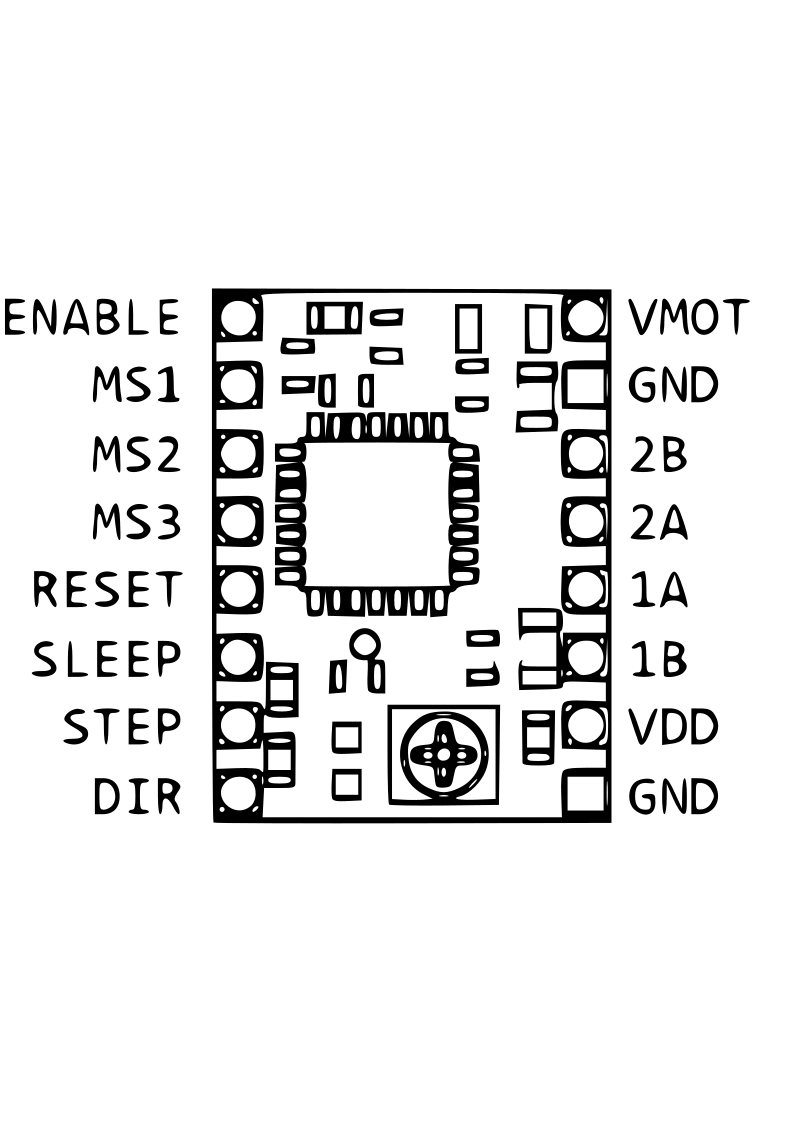


Рис драйвер A4988 и обозначение его выводов

Как видно на рисунке выше, драйвер представляет из себя плату, на которой расположен чип микроконтроллера, электронная обвязка, подстрочный резистор для регулировки опорного напряжения и выводы GPIO.

# 1 Аппаратная часть

В общем виде аппаратная часть представлена на рисунке 1.

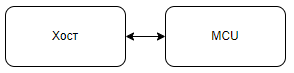


Рис. 1. Аппаратная часть – общий вид

Где хост – управляющее устройство, ПК под управлением ОС Windows, или мини-компьютер на подобие Rasberry pi под управлением ОС Linux.

MCU - micro-controller unit. То есть это исполняемое устройство, через которое в дальнейшем можно получить доступ к его портам, шинам, GPIO.

Более подробный вид схемы представлен на рисунке 2.

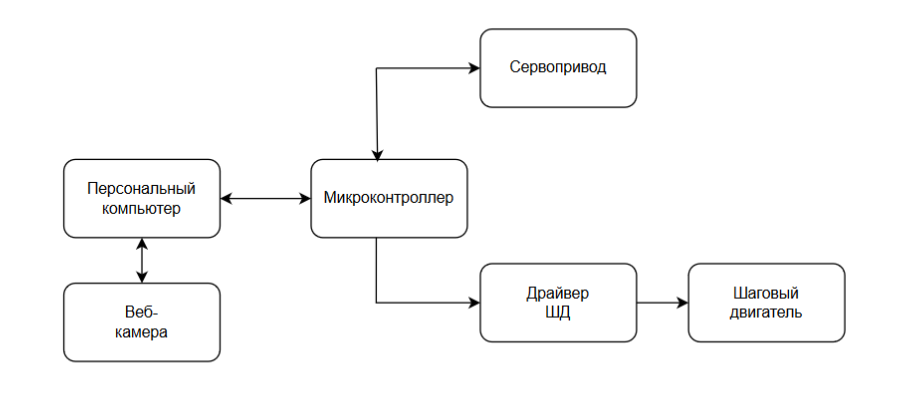


Рис. 2. Аппаратная часть – подробно

Примечание – здесь и далее, стрелками помечается наличие или отсутствие обратной связи.

Опишем основные узлы подробнее:

1. Веб-камера. В качестве веб-камеры будет использоваться смартфон под управлением ОС Android.
2. Персональный компьютер, на котором запущен скрипт, отвечающий за захват движений оператора и передачу управляющих команд по UART - универсальный асинхронный приёмопередатчик.
3. Микроконтроллер. Мы не будем вдаваться в подробности схемотехнического устройства управляющей платы и ее функциональных узлов. Отметим только, что в качестве микроконтроллера выступает испытанный временем чип Atmel Atmega328p.
4. Драйвер шагового двигателя — это специальное устройство, которое на основании импульсов, подаваемых с микроконтроллера, управляет сильноточными либо высоковольтными обмотками шагового двигателя. Используется драйвер А4988.
5. Шаговый двигатель – в проекте используется двигатель типоразмера nema 17.
6. Сервопривод – механический привод с отрицательной обратной связью.

# 2 Программная часть

## 2.1 описание программной части

Программная часть состоит из двух основных элементов. Управляющей программы – скрипта, который на основании пред обученной модели распознает на изображении, полученном с веб-камеры руку человека. И на основании заложенных в программу жестов передает управляющие команды на микроконтроллер через COM – порт компьютера. Со схемой программной части проекта можно ознакомиться на рис. 3.

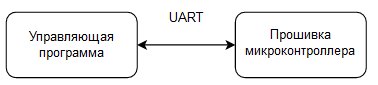


Рис. 3. Схема программной части проекта

Исходя из вышеизложенного можно сделать вывод, что для реализации проекта требуется разработать две программы. Управляющую и управляемую.

## 2.2. Программа микроконтроллера

### 2.2.1. Применяемые технологии

Разработка прошивки микроконтроллера будет осуществляться в редакторе кода VS Code.

Будет использоваться расширение PlatformIO – это специальная среда разработки, включающая в себя средства программирования и отладки, различные компиляторы и библиотеки для работы с микроконтроллерами.

Разработка осуществляется на языке программирования С++.

В проекте применяются следующие библиотеки:

* EEPROM – библиотека для работы с энергонезависимой памятью микроконтроллера.
* ServoSmoth – данная библиотека предоставляет алгоритмы и интерфейсы взаимодействия с сервоприводами
* Wire - библиотека предоставляет возможность работы с интерфейсом I2C, который используется для связи контроллера с oled экраном.
* Goled – библиотека для вывода информации на oled дисплей.
* Gstepper2 – библиотека для работы с шаговыми двигателями
* WDT – библиотека для работы со сторожевым таймером контроллера
* MyCycle – библиотека собственной разработки для работы с программными прерываниями и таймерами.

Разработка ведется с использованием парадигмы Объектного ориентированного программирования

2.2.2 Структура прошивки микроконтроллера и описание основных узлов

При разработке прошивки для микроконтроллера были поставлены следующие задачи:

* Управление приводами как вручную, с использованием потенциометров, так и с помощью жестов оператора;
* управление шаговым двигателем;
* возможность отображения информации на oled-дисплее. Такой как приветствие, координаты приводов, отклик на действия оператора, вывод отладочной информации;
* сохранение в энергонезависимой памяти координат и их последующее восстановление из памяти;
* поддержка резистивной клавиатуры и ввода команд с использованием тактовых клавиш;
* осуществление взаимодействия с ПК по последовательному интерфейсу UART. Прием команд с ПК, отклик на них, вывод отладочной информации в com-порт компьютера;
* система управления питанием контроллера, отслеживание напряжения питания, перезагрузка при падении напряжения.

Ознакомиться со структурой программной части микроконтроллера можно на рисунке 4.

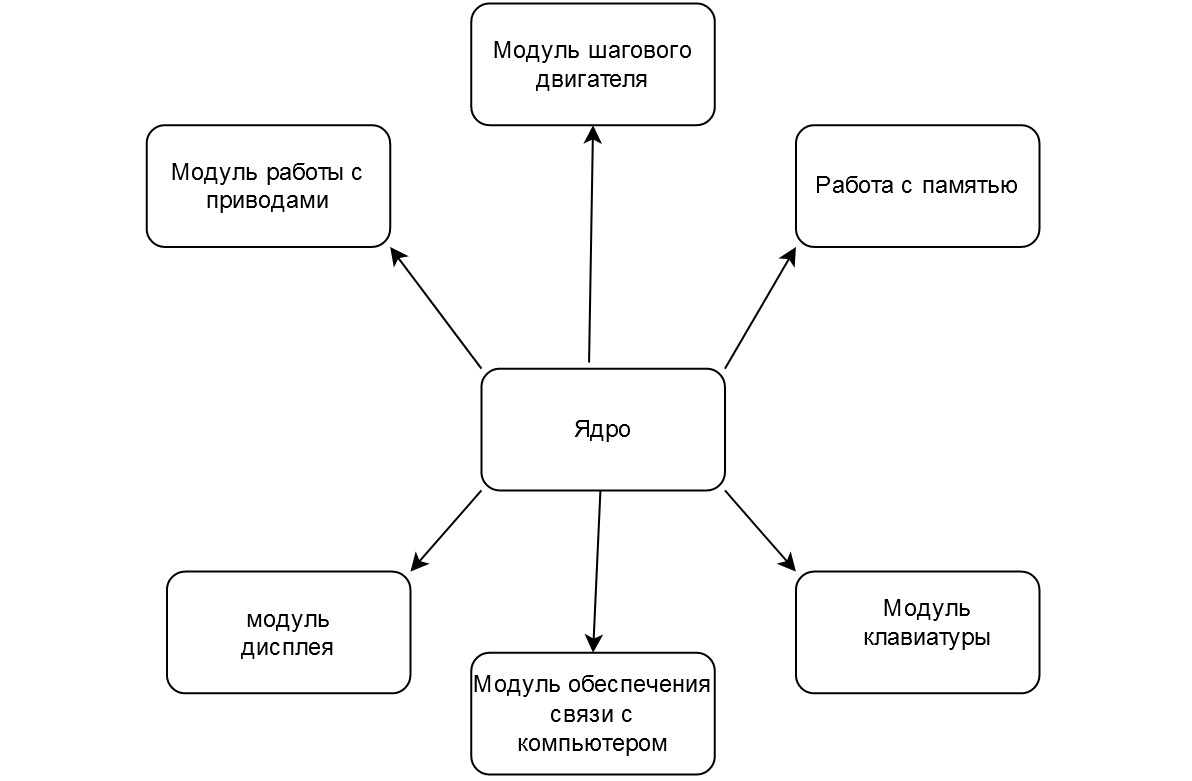


Рис. 4. Модульная структура программы микроконтроллера

Подробное описание каждого из модулей будет представлено в отчете по выпускной квалификационной работе. Здесь же будет представлено описание основных частей программы.

### 2.2.3 Модуль работы с сервоприводами

Сервопривод представляет собой механизм, в данном случае мотор постоянного тока, имеющий в своем составе специальный датчик, осуществляющий контроль и поддержание параметров во время работы, передаваемых отдельный момент времени.

К устройству по присоединенному к нему проводу подается управляющий сигнал, представляющий собой импульсы постоянной частоты и переменной ширины. При подаче сигнала в проводимую схему генератор производит свой импульс, размер которого устанавливается с помощью потенциометра. Другая часть схемы проводит анализ всех постигаемых сигналов, и если он разный, то происходит включение сервопривода. Если размеры импульсов равнозначные, электромотор отключается [1].

Исходя из вышеизложенного становится ясно, что для управления сервомотором нам нужно подать на его контроллер импульс определенной ширины. Подключим мотор к одному из цифровых выводов микроконтроллера с поддержкой генерации ШИМ-сигнала. Подключим библиотеку для работы с приводом и создадим объект.

#include <ServoSmooth.h>

#define AMOUNT 5 // указываем колличество приводов используемых в проекте

#define impulsMin  500  //600

#define impulsMax 2400 //2600

#define servosSpeed 100

#define servosAccel 0.5

#define homePosition 180

#define maxDeg 180

ServoSmooth servos[AMOUNT];

Подключаем сервопривод, указываем цифровой пин микроконтроллера к которому производится подключение сигнального провода, минимальное и максимальное значение ширины импульса а так же начальную точку в которую будет перемещен привод при подаче питания и инициализации (это значение будет взято из энергонезависимой памяти контроллера):

servos[0].attach(11, impulsMin, impulsMax, servo0InitialPoint);

Зададим максимальную скорость и ускорение развиваемые приводом, мы хотим предотвратить износ шестерен и не желаем наблюдать резкие рывки, поэтому скорость будет ограничена (см. блок кода инициализации).

for (int i = 0; i < AMOUNT; i++)

  {

    servos[i].setAccel(servosAccel);

    servos[i].setSpeed(servosSpeed);

    servos[i].setMaxAngle(maxDeg);

    servos[i].smoothStart();

  }

Вызываемый метод smoothStart() означает что движение к стартовой позиции будет осуществлено плавно, без рывка.

Добавим возможность управлять мотором с помощью потенциометра

if (millis() - tmr3 >= 40) {

      tmr3 = millis();

      int pos0 = map(analogRead(A0),0,1023,impulsMin,impulsMax);

      servos[0].setTarget(pos0); }

Обращаем ваше внимание на то, что опрос аналогового пина, которому подключен вывод потенциометра происходит с задержкой, раз в 40 миллисекунд. Это сделано для того, чтобы привод успел получить координату и начать движение до прихода следующей координаты. Иначе возможны рывки и некорректное обрабатывание команд оператора. Потенциометр представляет собой делитель напряжения, вращая ручку мы можем изменять уровень напряжения на его выводе от 1в до 5в. Микроконтроллер, с помощью аналого-цифрового преобразователя считывает напряжение с аналогового пина и представляет его в цифровом виде, затем мы преобразуем это значение в интервале от минимальной ширины импульса до максимальной, что практически соответствует крайне левому и крайне правому положению вала привода. Нужно оговориться, что крайних положений мотор не достигнет из-за того, что потенциометр не идеален, он имеет «мертвые зоны» и собственное сопротивление. Затем мы заносим получившееся значение в целочисленную переменную, и устанавливаем ее в качестве цели для поворота.

Следует отметить, что управлять приводом мы можем двумя способами. Устанавливая нужный угол или задавая значение в виде ширины импульса. Последний случай считаем более правильным, поскольку значение в «углах» лишь является надстройкой, и переводит значение импульсов в углы тратя процессорное время.

Модуль работы с приводами готов, мотор инициализирован, мы можем управлять им с помощью метода SetTarget(x).

TODO Шаговый двигатель

### 2.2.4 Модуль работы с UART

UART, или по-русски УАПП (Универсальный Асинхронный передатчик) – является наиболее распространенным на сегодняшний день физическим протоколом передачи данных. Наиболее известен из семейства UART – протокол RS-232 (в народе – COM-порт). Несмотря на свой почтительный возраст он дожил до наших дней и до сих пор применяется для коммуникации многих промышленных устройств.

Интерфейс имеет две рабочие линии – RX – принимающая и TX – передающая. Выход передатчика соединен со входом приемника и наоборот [2].

Передача происходит битами, в начале передачи устанавливается стартовый бит, завершает передачу стоп-бит.

На отладочной плате контроллера установлена специальная микросхема, называемая USB-TTL-конвертер. Она позволяет подключать к компьютеру по средствам USB-кабеля любое устройство с последовательным интерфейсом.

Инициализируем работу с UART и зададим скорость передачи данных.

Serial.begin(9600);

Serial.setTimeout(10);

Serial.print(F(“Hello Word”));

Задаем скорость передачи равной 9600 бод – бит в секунду. Этого будет достаточно для решения нашей задачи, величина скорости передачи (бодрейт) зависит от качества микросхемы преобразователя, тактовой частоты, а также от качества и экранирования проводников.

Задаем таймаут (10 секунд), дело в том, что передача происходит не сразу, устройство-приемник и передатчик должны инициализироваться.

Затем, мы передаем на компьютер сообщение “Hello world”, обратите внимание на макрос F, он сообщает о том что передаваемая строка будет храниться не в оперативной, а во Flash-памяти контроллера, это один из многочисленных методов оптимизации которым мы будем пользоваться.

Перейдем к реализации коммуникации контроллера и хоста. Представим как будет выглядеть эта связь в виде блок схемы. В данном отчете будет представлена реализация только двух комманд: «Выполнить захват» и «разжать». Блок схема представлена на рис 5.

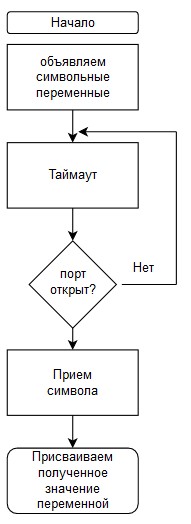


Рис. 6. Блок-схема модуля приема информации по UART

Реализуем блок-схему в виде функции:

 char receivedData(){

  if (millis() - tmr\_stp >= 40) {

    tmr3 = millis();

    if (Serial.available()) {

      ch = Serial.read(); }

    return ch;  }}

Мы будем передавать по UART символьные значения. Опрашивать COM-порт сотни тысяч раз в секунду нет никакой необходимости, поэтому мы задаем таймер, опрос будет происходить раз в 40 миллисекунд. Если порт открыт и доступен, то присваиваем переменной полученное значение. Возвращаем его. Вызываем функцию в теле главного цикла программы.

Переходим к реализации обработчика. Добавляем условие, если значение, которое вернула функция равно, например «С», то выполняем определенное действие, если «О», то другое действие, а если какое либо иное значение, то просто игнорируем его и ничего не делаем. Реализуем алгоритм в коде, воспользуемся конструкцией switch-case:

 switch (res)

  {

    case 'C':{

      servos[0].setTarget(impulsMax);

    }

    break;

    case 'O':{

      servos[0].setTarget(impulsMin);

    }

    break;

  default:

    break; }

Если символ «С» то даем цель первому приводу повернуться на максимальный угол, если «O» то на минимальный, можно усложнить, взяв текущее положение и прибавляя на определенное значение, но в данном случае в этом нет необходимости.

Мы разобрали основные элементы программы микроконтроллера, которые нужны для решения нашей задачи.

Перейдем к управляющей программе.

## 2.3. Управляющая программа

### 2.3.1 Применяемые технологии

Разработка будет вестись на языке Python в среде PyCharm. Выбор языка программирования обусловлен простотой и «дружелюбием» для разработчика, а также большим выбором доступных библиотек и очень качественной документацией.

Используемые библиотеки:

* OpenCV – основная библиотека, предоставляет методы для работы с компьютерным зрением.
* Mediapipe – содержит пред обученные модели распознавания кистей рук, глаз, лица, тела человека и т.д.
* CVzone – прослойка между ними, облегчает работу с mediapipe
* Serial – работа с COM-портом.

### 2.3.2 Компьютерное зрение, захват движения

Способность воспринимать форму и движение рук может стать жизненно важным компонентом для улучшения пользовательского опыта в различных технологических областях. Например, она может стать основой для понимания языка жестов и управления жестами рук, а также для наложения цифрового контента и информации поверх физического мира в дополненной реальности. Несмотря на то, что люди воспринимают руки естественным образом, надежное восприятие рук в реальном времени представляет собой сложную задачу компьютерного зрения, поскольку руки часто закрывают сами себя или друг друга и не имеют высококонтрастных моделей.

MediaPipe Hands - это решение для отслеживания рук и пальцев с высокой точностью. Оно использует машинное обучение (ML) для определения 21 метки руки на основе всего одного кадра. Метки (landmarks) представлены на рисунке 7.

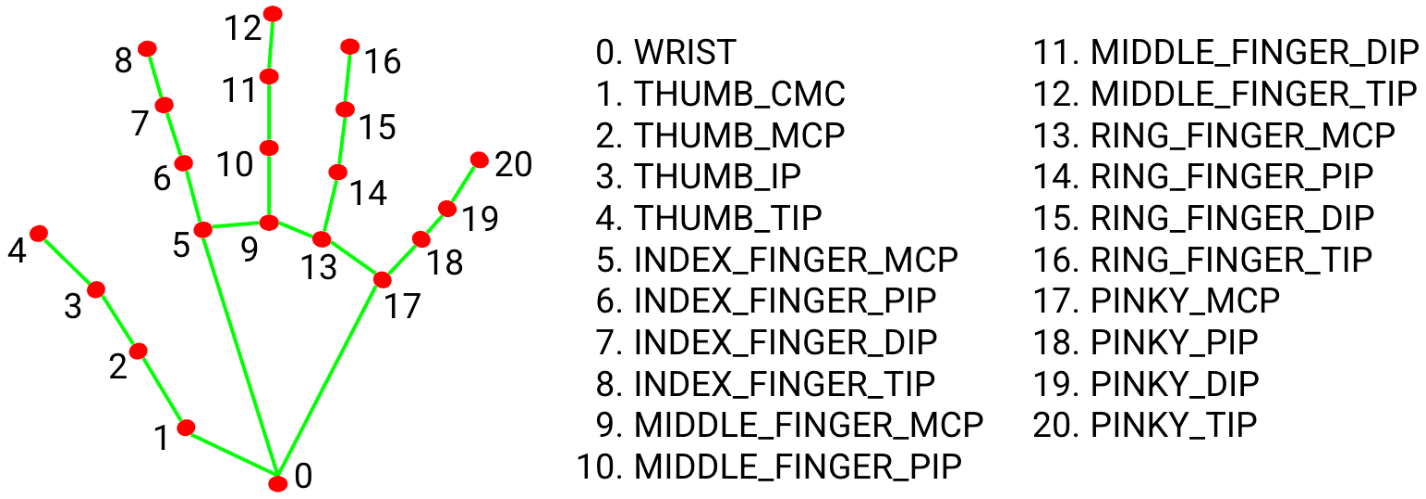


Рис. 7. Метки руки

### 2.3.3 Разработка программы

Добавим вывод изображения на экран

cap = cv2.VideoCapture(0)  
detector = HandDetector(detectionCon=0.8, maxHands=2)  
  
*while True*:  
 success, img = cap.read()  
 hands, img = detector.findHands(img) *# With Draw*

Результат:



Рис. 8.

Добавим счетчик кадров:

cTime = time.time()  
fps = 1 / (cTime - pTime)  
pTime = cTime  
cv2.putText(img, *str*(*int*(fps)), (10, 70), cv2.FONT\_HERSHEY\_PLAIN, 3, (255, 0, 0), 3)

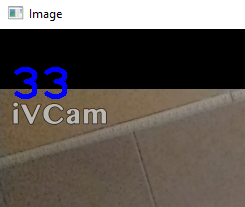


Рис. 9

Добавим распознавание изображения ладони и отображение меток

detector = HandDetector(maxHands=2, detectionCon=0.7)  
*while True*:  
 success, img = cap.read()  
 hands, img = detector.findHands(img, flipType=*False*)  
  
 *# hands = detector.findHands(img, draw=False) #no draw  
  
 if* hands:  
 hand1 = hands[0]  
 lmList1 = hand1["lmList"] *# list of 21 landmarks points* bbox1 = hand1["bbox"]  
 centerPoint1 = hand1["center"] *# center of the hand cx cy* handType = hand1["type"]

right = detector.fingersUp(hand1)

Результат:

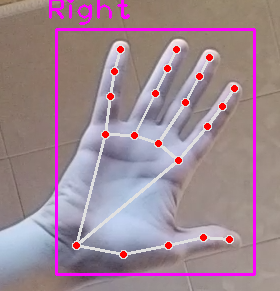
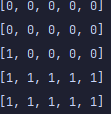


Рис. 10

Теперь перед нами стоит задача определить какой из пальцев поднят а какой опущен.

fingers = detector.fingersUp(hand1)  
*print*(fingers)

Смотрим на вывод



Создается список из пяти элементов, каждый элемент соотносится со своим пальцем и принимает 2 значения, 0 и 1, 0 означает что палец опущен, а 1 – палец поднят. Мы рассмотрим 2 жеста, ладонь раскрыта и ладонь сомкнута. Раскрытая ладонь будет означать что нужно установить сервопривод в положение 0, а сомкнутая ладонь – повернуть вал на 180 градусов.

Из вышесказанного следует, что нас интересует следующая комбинация цифр [0, 0, 0, 0, 0,] – ладонь закрыта, [1, 1, 1, 1, 1] – ладонь раскрыта.

*if* left == [0, 0, 0, 0, 0]:  
 port.write('C'.encode())  
*if* left == [1, 1, 1, 1, 1]:  
 port.write('O'.encode())

Обратите внимание, мы проверяем условие для левой руки, и если получили верное значение то отправляем в COM-порт символ С, или О.

Не забудем создать объект Serial

mySerial = SerialObject("COM19", 9600, 1)

Зададим порт к которому подключен контроллер, бодрейт и задержку.

Перейдем к тестированию.

# 3. Тестирование

## 3.1 Описание тестового стенда

Тестовый стенд представляет собой разработанную отладочную плату (см. рисунок 10). И подключенный к ней один из узлов робота манипулятора отвечающий за захват предметов (рис. 11).



Рис. 10. Управляющая плата

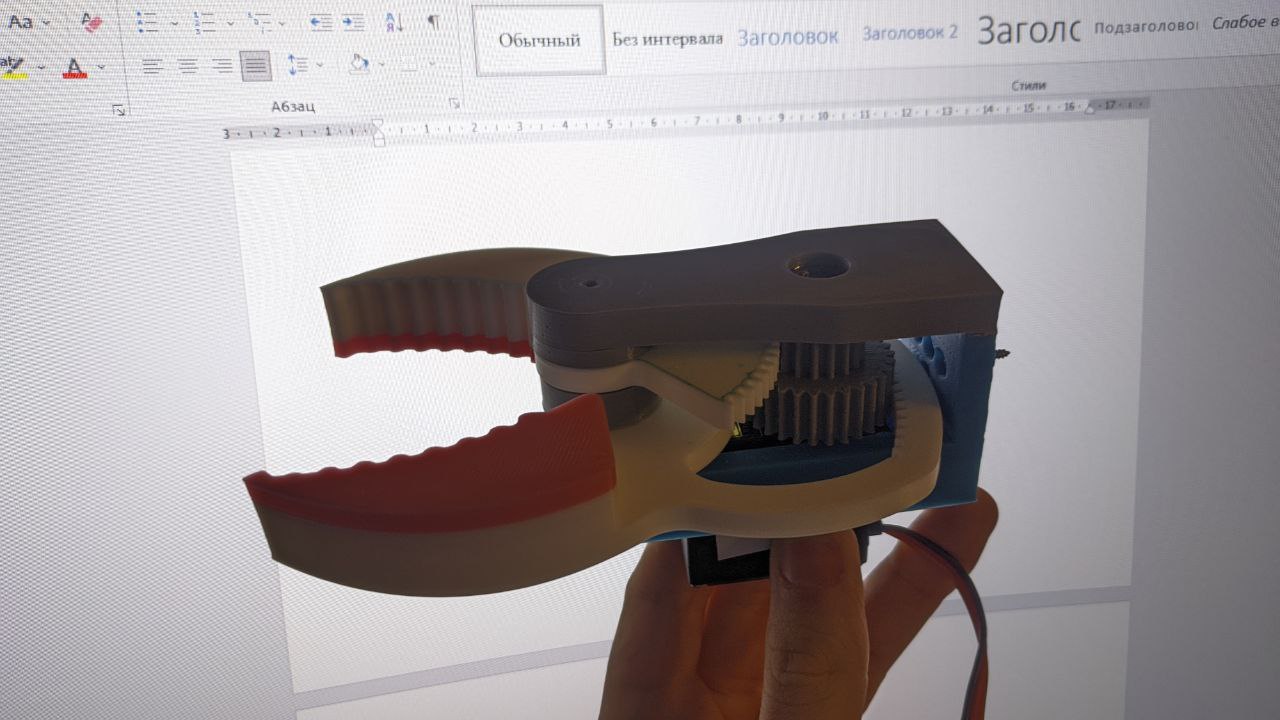


Рис. 11. Один из узлов манипулятора

## 3.2 тестирование программы

Прошивка загружена в контроллер, он подключен к компьютеру и определен как COM19. Подключаем питание двигателя и запускаем скрипт.

На рисунке 11 показано выполнение первого жеста. Жест – открыть, захват – разомкнут.

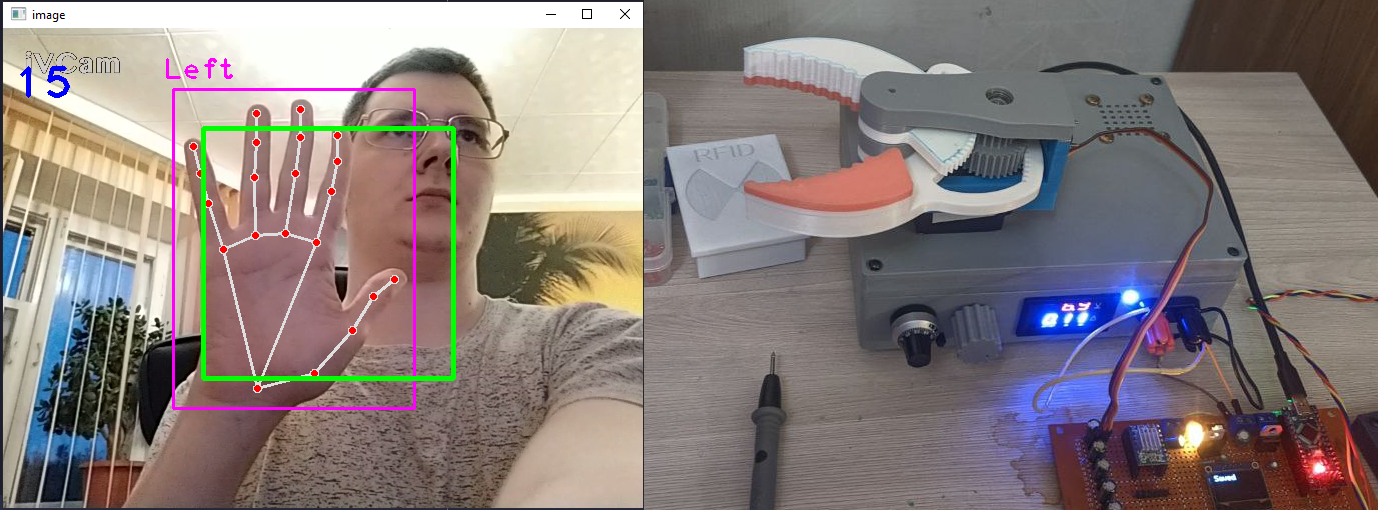


Рис. 11. Жест – открыть

На рисунке 12 показано выполнение второго жеста. Жест - закрыть – ладонь сжата в кулак, захват – замкнут.

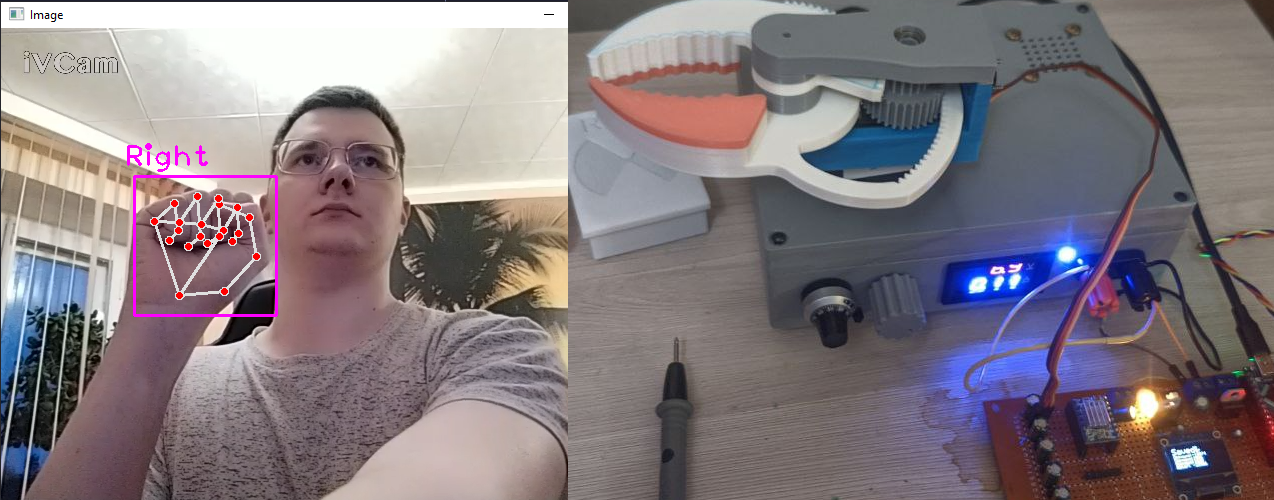


Рис. 12. Жест- закрыть

В результате тестирования программы, мы можем прийти к выводу что программа функционирует нормально, и выполняет заложенный в нее функционал.

# Заключение

В ходе выполнения производственной практики был создан прототип робота манипулятора, разработана управляющая плата. Разработана управляющая программа и исполнительная программа. Налажена работа сервопривода и связь контроллера с компьютером. Подобный способ управления техникой имеет потенциал для дальнейшего изучения.

# Список литературы

1 Бычков М.Г., Ладыгин А.Н. Современный сервопривод ― классифи-  
кация и терминология

2. Михаил Гук «Аппаратные интерфейсы ПК»