Программируемый контроллер ВЕРТОР МЕГА

Артикул ПЭМ10.9194

Технические данные и руководство пользователя

1. Назначение устройства

Программируемый контроллер ВЕРТОР МЕГА (Рис. 1.1) является одним из ключевых элементов системы управляющей электроники «Эвольвектор ВЕРТОР» (далее ВЕРТОР). Он предназначен для создания систем управления старших моделей робототехнических конструкторов «Эвольвектор», оснащенных приводами высокой мощности и потребляющих большое количество электроэнергии. Контроллер рассчитан на применение совместно с электронными модулями, входящими в систему ВЕРТОР (подробная информация о системе представлена в соответствующем разделе сайте https://academy.evolvector.ru).

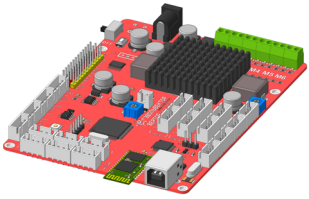

Рис. 1.1

2. Конструкция контроллера и назначение выводов (контактов)

Контроллер выполнен в виде печатной платы, которая оснащена следующими элементами:

- разъемами типа XH-2.54-4P для подключения совместимых электронных модулей (группа разъемов №1, 2, 5);
- разъемами питания под штекер размером 2,1x5,5 для подачи электроэнергии от заряжаемого модуля;
- клеммниками для подсоединения электрических двигателей, соответствующих требованиям номинального напряжения и мощности;
- штыревыми контактами для подключения стандартных RC серводвигателей;
- разъемами типа XH-2.54-4P для подключения устройств, работающих по последовательному протоколу UART (группа разъемов №3);
- разъемами типа XH-2.54-4P для подключения устройств, работающих по протоколу I2C (группа разъемов №4);
- регулятором напряжения питания моторов;
- разъемом USB тип B для подключения контроллера к персональному компьютеру и загрузки в него программ управления (скетчей);
- выключателем питания для клеммников электродвигателей и контактов сервоприводов;
- разъемом для подключения вентилятора, являющегося элементом системы охлаждения нагревающихся частей платы контроллера;
- регулятором скорости вращения вентилятора;
- кнопкой принудительной перезагрузки контроллера в случае его сбоя или некорректной работы;
- bluetooth модулем, подключенным к выводам управляющего микроконтроллера и позволяющим как загружать скетчи в контроллер через bluetooth соединение, так и выполнять дистанционное управление контроллером с помощью мобильного приложения.

Перечисленные элементы представлены на рисунке 2.1.

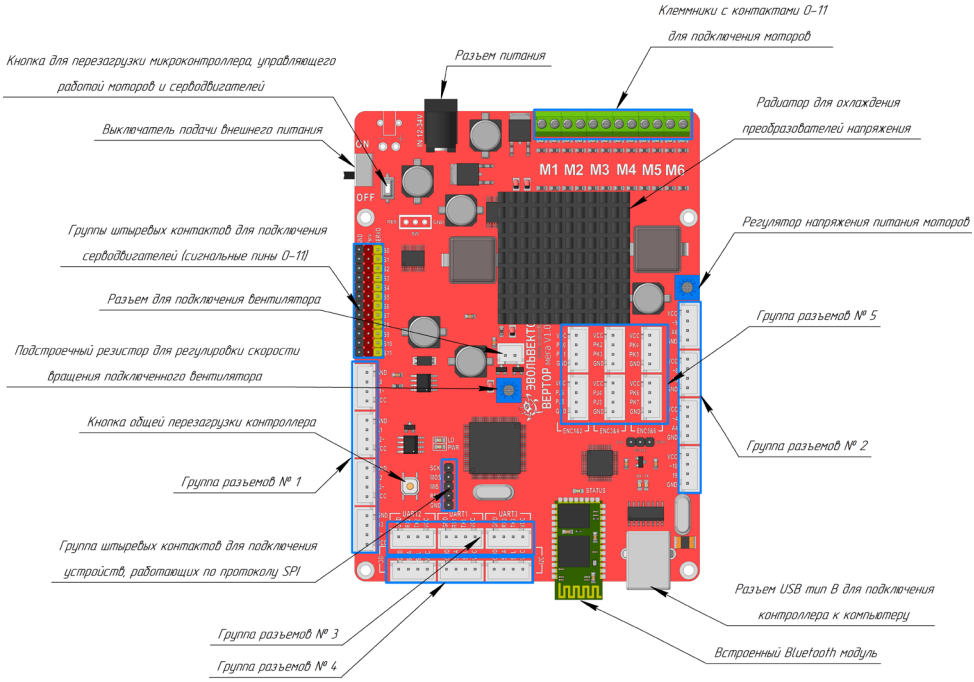

Рис. 2.1

На рисунке 2.2 представлен внешний вид контроллера с указанием габаритных размеров, а также расположение и размеры крепежных отверстий.

Расположение и форма крепежных отверстий на плате контроллера совместимы с таковыми на модулт питания 6-тью аккумуляторами типа 18650, что дает возможность крепить эти устройства с помощью стоек одно над другим. Помимо этого, по расстоянию между крепежными отверстиями (кратно 8 мм) контроллер совместим с конструкторами Эвольвектор, LEGO, MakeBlock, и может прикручиваться к их деталям с помощью стоек.

Смонтированные на контроллере четырехконтактные разъемы типа XH-2.54-4P промаркированы и объединены в группы, исходя из функциональных особенностей подключаемых к ним модулей. Благодаря имеющимся разъемам предусмотрена возможность подключения сразу до 20-ти электронных модулей системы электроники ВЕРТОР. Общей чертой всех разъемов является наличие выводов питания (VCC) и «земли» (GND), которые расположены по краям разъема. В части же сигнальных контактов, размещенных в центре разъема, существуют отличия.

Группы разъемов №1 и №2 являются разъемами общего назначения и размещены по боковым сторонам платы. У части из них присутствует один цифровой (промаркирован просто числовым номером) и один аналоговый (промаркирован буквой A с номером) контакты.

К разъемам с аналоговым пином возможно подключение датчиков, выдающих аналоговый сигнал. Один

разъем из 2-й группы имеет 2 цифровых контакта — к такому возможно подключение исключительно цифровых модулей.

Цифровые контакты, перед номером которых стоит знак тильда («~»), поддерживают широтно-импульсную модуляцию, то есть на них может быть сформирован псевдо аналоговый выходной сигнал.

Также имеется группа разъемов №3, которая предназначена для подключения устройств, работающих по последовательному интерфейсу передачи данных UART.

К группе №4 относятся разъемы, позволяющие подключать модули, работающие по интерфейсу I2C.

А группа разъемов №5 включает в себя дополнительные разъемы, которые можно использовать для подключения энкодеров мотор-редукторов или дополнительных датчиков или устройств.

Штыревые разъемы с маркировкой S0...S11 предназначены для подключения до 12-ти стандартных RC серводвигателей с суммарным токопотреблением не более 5 А. С аналогичной общей токовой нагрузкой возможно подключение до 6-ти обычных коллекторных моторов к клеммникам (контакты промаркированы M1...M6).

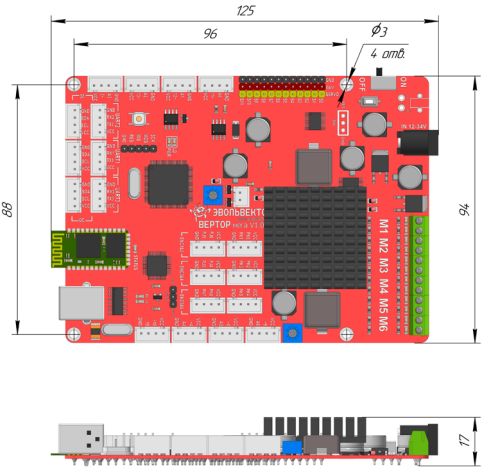

Рис. 2.2

3. Принципы работы с контроллером

3.1 Подсистема питания и ее охлаждение

Контроллер ВЕРТОР МЕГА представляет собой плату, которая является основой для создания систем управления роботами, оснащенных большим количеством датчиков и приводов различного типа. Проще говоря, это контроллер, на базе которого возможно конструирование манипуляционных и мобильно-манипуляционных роботов в рамках образовательных проектов.

Плата наделена мощной подсистемой питания для подключаемых устройств. Для питания серводвигателей доступно токопотребление до 5 А. Аналогичная величина потребляемого тока допустима и для мотор-редукторов. То есть суммарное токопотребление приводов робота может достигать 10 А.

Это приводит к существенному тепловыделению на преобразователях напряжения, расположенных на плате, поэтому возникает необходимость в их дополнительном охлаждении. Оно реализуется с помощью установленного на нагревающиеся элементы радиатора и размещенного на нем вентилятора. В базовой поставке контроллера радиатор уже закреплен на плате, а вентилятор находится в коробке с контроллером отдельно, исходя из условий упаковки и транспортировки. Поэтому, прежде чем приступить к работе с контроллером, требуется установить вентилятор на радиатор и подключить его к контроллеру (Рис. 3.1).

В целях энергосбережения на плате выполнено автоматическое управление вентилятором. Плата оснащена датчиком температуры и вентилятор включается только тогда, когда температура нагревающихся элементов достигает порогового значения.

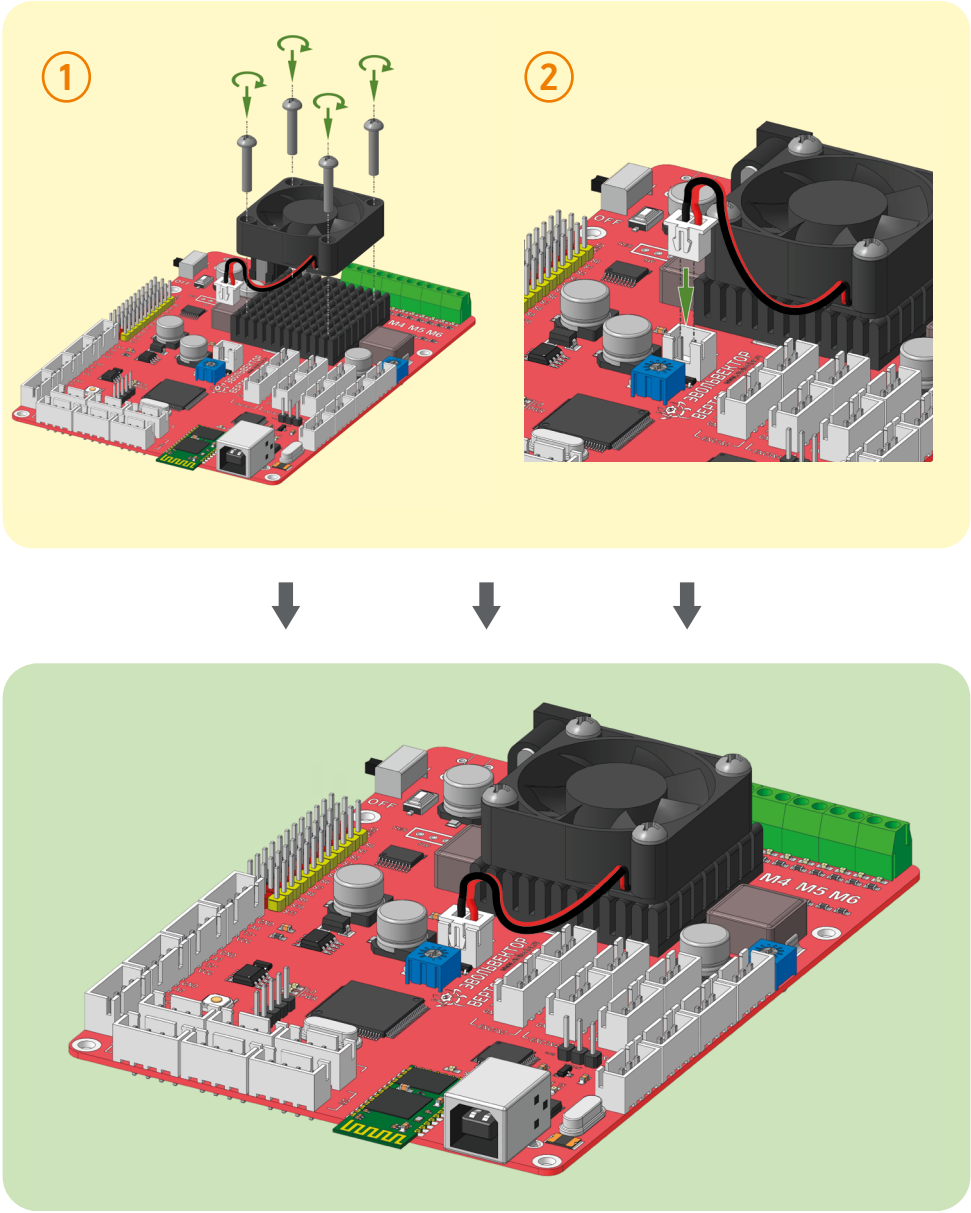

Рис. 3.1

3.2 Подключение контроллера к компьютеру по USB

Контроллер построен на основе платформы Arduino Mega. Поэтому в базовом варианте его программирование осуществляется через среду разработки Arduino IDE. В ней создается программа (скетч) на любом персональном компьютере (далее ПК) с установленной операционной системой (далее ОС) Windows или Linux. После создания скетча контроллер подключается к данному компьютеру и в него загружается написанная программа. Дополнительные программаторы для программирования контроллера не требуются.

Подключение контроллера к ПК возможно двумя способами: с помощью обычного USB кабеля и посредством сопряжения по Bluetooth каналу.

Подключение контроллера к компьютеру по USB выполняется с помощью кабеля «USB – USB тип B». Если на компьютере установлена ОС Windows, то после подключения контроллера операционная система в автоматическом режиме должна обнаружить контроллер и также автоматически установить необходимое программное обеспечение (драйвер com-порта usb-контроллера CH340). На это может потребоваться некоторое время (до 10-15 минут), поэтому не стоит торопиться сразу после первого подсоединения контроллера что-то в него загружать.

Если этого не произошло и драйвер автоматически по каким-либо причинам так и не установился, то его потребуется установить в ОС вручную.

Для этого зайдите в диспетчер устройств и отыщите в списке оборудование USB2.0-Serial с восклицательным знаком (Рис. 3.2).

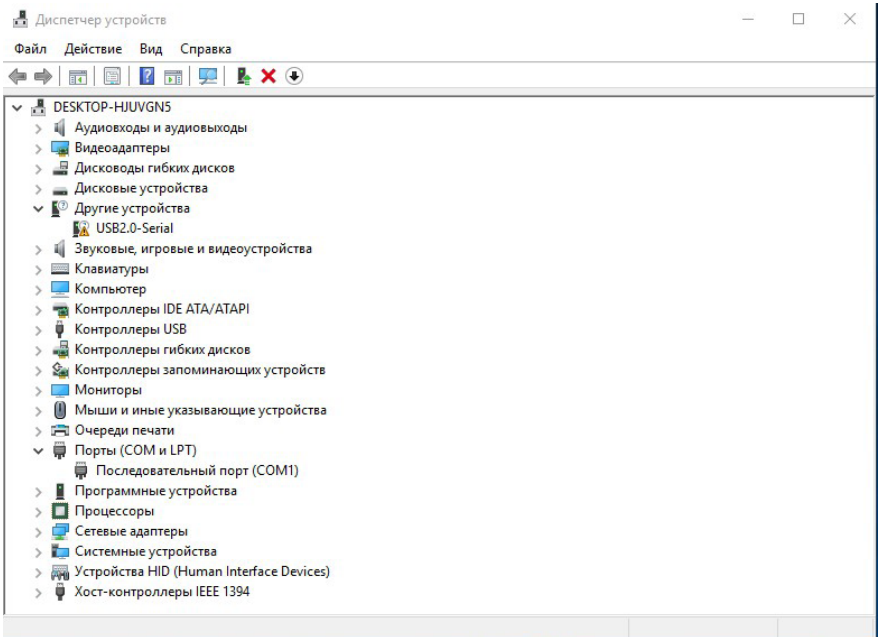

Рис. 3.2

Обновите драйвер, кликнув по данному пункту правой кнопкой мыши, и выбрав пункт «Обновить драйвер». Далее укажите в одном из диалоговых окон путь к папке с файлами драйвера для USB контроллера CH340 (Рис. 3.3).

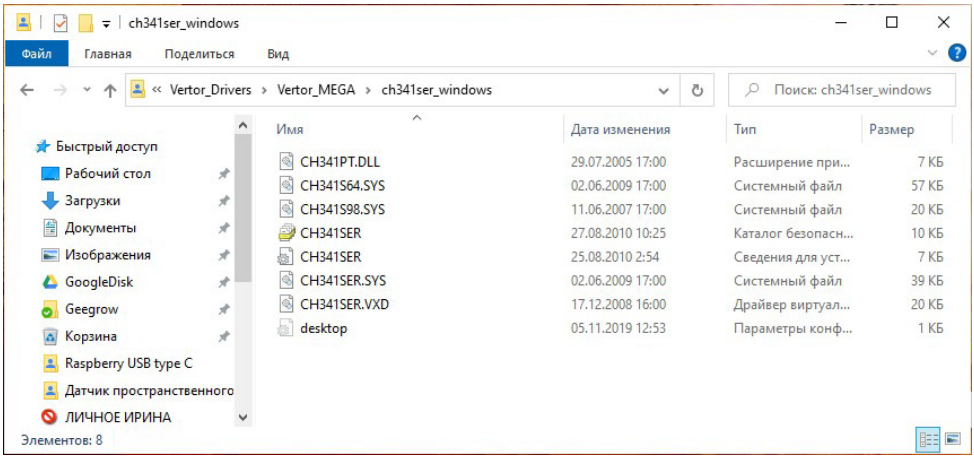

Рис. 3.3

В результате, после установки драйвера в диспетчере устройств появится порт USB-SERIAL CH340, сразу после которого в скобках указывается номер COM порта, который ему автоматически присваивается при установке (Рис. 3.4).

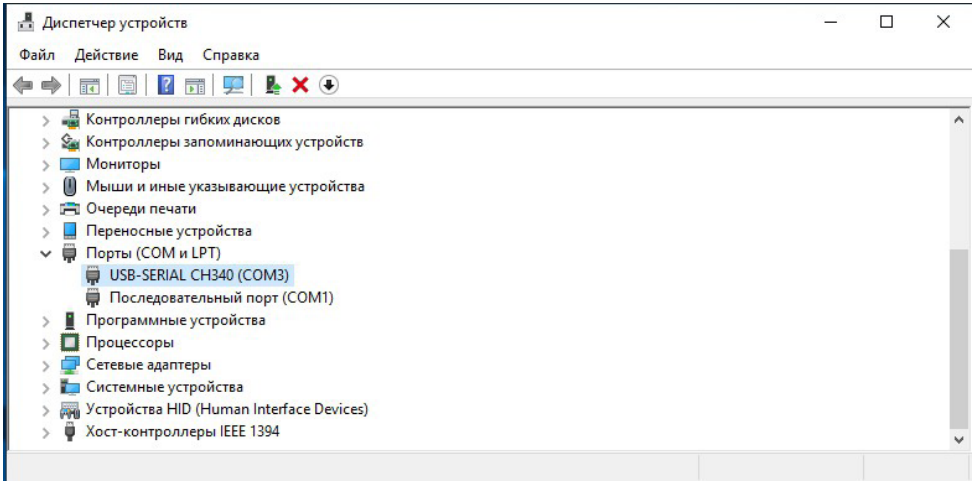

Рис. 3.4

Именно данный порт надо указывать в Arduino IDE при настройке соединения контроллера с компьютером (Рис. 3.6). Однако, предварительно в среде разработки требуется выбрать саму плату контроллера (Рис. 3.5).

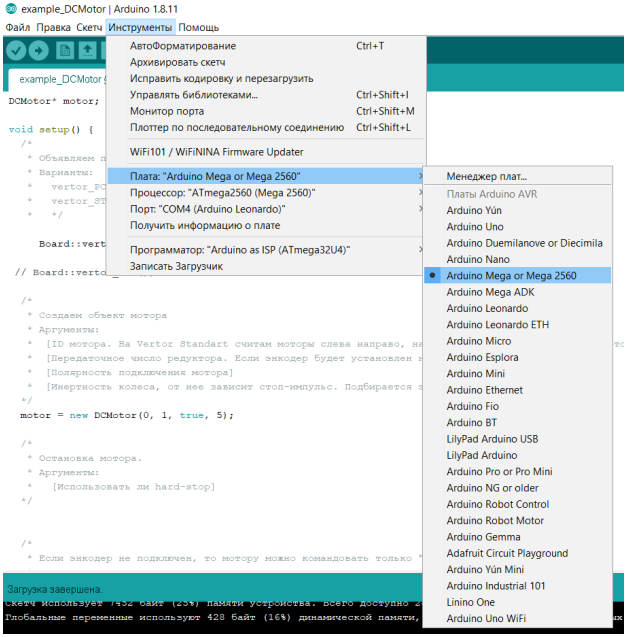

Рис. 3.5

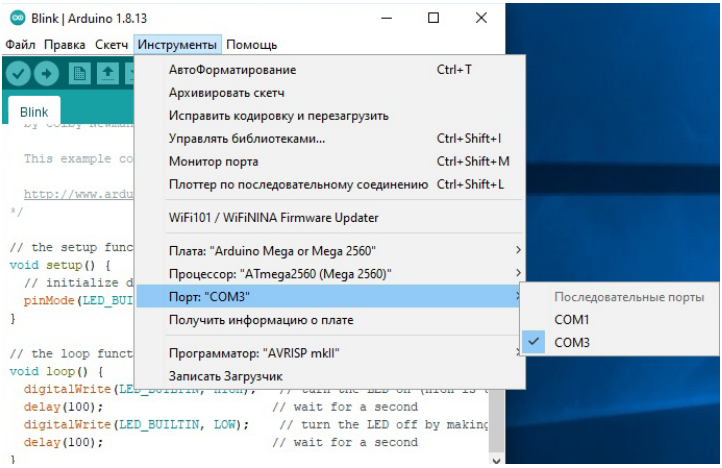

Рис. 3.6

После выполнения указанных настроек контроллер полностью готов к загрузке скетчей.

Если же контроллер подключается к компьютеру с ОС Linux, то все гораздо проще. Все, что необходимо установить для работы — инсталлировать саму Arduino IDE (делается это путем ввода в терминал команды sudo apt-get install arduino) и выполнить простые настройки. Они сводятся к такому же выбору платы и выбору порта, который здесь имеет несколько иной вид (Рис. 3.7).

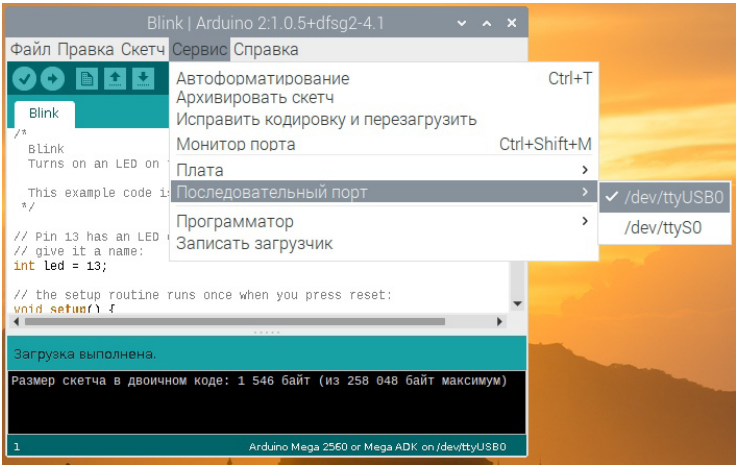

Рис. 3.7

3.3 Подключение контроллера к компьютеру по Bluetooth и простая передача данных

Контроллер оснащен встроенным bluetooth модулем JDY 30. Это делает возможным как загрузку скетчей в контроллер в беспроводном режиме после предварительного сопряжения, так и дистанционное управление моделями, собранными на основе контроллера.

Данный модуль может работать в следующих режимах:

1. Режим поиска устройств и сопряжение с ними, после чего свободно выполняется передача данных между устройствами;

2. Режим передачи данных между модулем и сопряженным устройством. В этом случае модуль способен работать только в качестве подчиненного устройства (slave). В этом случае модуль способен только принимать и передавать данные управляющему устройству, с которым он сопряжен. Например, данный режим используется при подключении модуля к смартфону.

3. Режим программирования. Данный режим позволяет изменять всевозможные настройки модуля: запрашивать и менять его имя, адрес, скорость передачи данных, выполнять изменение кода доступа к модулю, а также менять другие параметры работы.

Для определения того, в каком режиме работает модуль, существует система индикации с помощью индикаторного светодиода STATE, находящегося на поверхности платы:

1. Светодиод мигает с частотой 1 Гц — модуль в режиме программирования или поиска устройств;
2. Светодиод не мигает — модуль сопряжен с другим bluetooth устройством.

Смонтированный на плате bluetooth модуль работает в режиме Slave. Это значит, что то устройство, к которому выполняется подключение (ПК, смартфон), будет являться управляющим. Для подключения модуля к такому устройству необходимо подать питание на модуль (например, подключить его ко включенному контроллеру) и произвести на нем поиск bluetooth модуля (Рис. 3.8 а). После этого выбрать найденный модуль из списка и ввести пин-код доступа, состоящий из 4-х цифр (Рис. 3.8 б). По умолчанию это 1234 (Рис. 3.8 в). В результате устройство и модуль будут сопряжены.

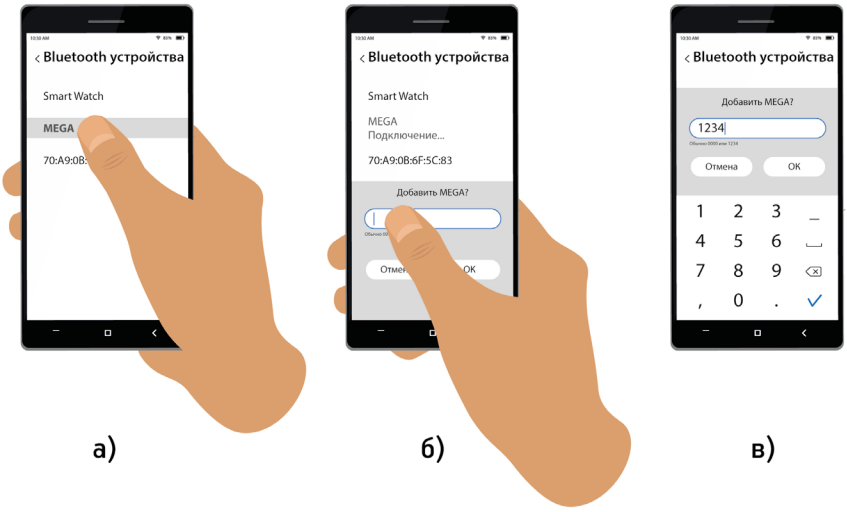

Рис. 3.8

После сопряжения устройства могут беспрепятственно обмениваться данными. Для этого в контроллер необходимо загрузить скетч, содержащий команды по чтению данных, получаемых от сопряженного с модулем устройства. Пример такого скетча приведен ниже:

```
char temp;                  // Инициализация переменной типа char с именем temp
void setup() {
Serial.begin(115200); }     // Инициализация последовательного порта Serial на скорость работы 115200 бод
void loop() {
if (Serial.available()) {   // Если по последовательному порту Serial поступили данные, то...
temp = Serial.read();       // Выполнить присвоение их переменной temp
Serial.print(temp);}}       // Вывести значение переменной temp в монитор последовательного порта
```

Здесь показано, как переменной **temp** присваиваются символьные данные, которые поступили от управляющего устройства на контроллер через bluetooth модуль.

3.4 Продвинутая настройка Bluetooth модуля на контроллере (его программирование)

Программирование модуля Bluetooth на контроллере выполняется с помощью так называемых АТ-команд. Для этого необходимо загрузить в контроллер представленный выше скетч, открыть монитор порта (Рис. 3.9) и выставить его настройки в соответствии с рисунком 3.10 (добавление в конце строки символа возврата каретки в начало строки и перевода ее на новую строку /r/n (пункт NL&CR), выбирается скорость 115200 бод). Далее необходимо вводить АТ команды в строку монитора порта, а ответы модуля будут печататься в его главном окне.

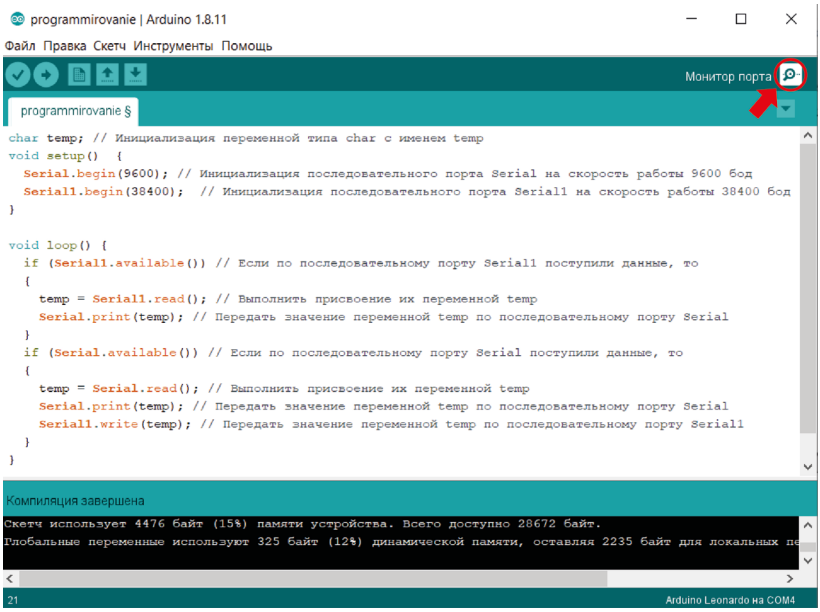

Рис. 3.9

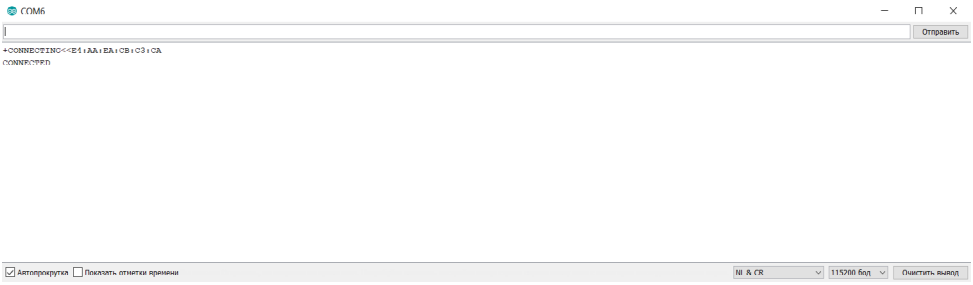

Рис. 3.10

Принцип программирования bluetooth модуля сводится к вводу AT-команд и получению ответа от модуля. Ответ - это либо запрашиваемая информация, либо подтверждение о том, что команда выполнена успешно, если она направлена на изменение какого-нибудь параметра. Ниже в таблице приведены в общем виде основные AT-команды, которые могут потребоваться для настройки модуля при выполнении образовательных проектов. Набирать все их символы надо именно так, как указано в таблице. Помимо самих команд указаны ответы модуля на них и пояснения, что они означают.

Команда	Ответ модуля	Примечание
AT	OK	Тестовая команда для проверки установки соединения с модулем
AT+RESET	OK	Команда перезагрузки модуля
AT+VERSION	+VERSION=**ВЕРСИЯ** OK	Запрос версии прошивки модуля. Модуль возвращает версию загруженной прошивки.
AT+LADDR	+LADDR=**АДРЕС** OK	Запрос адреса модуля.
AT+LADDR**АДРЕС**	OK	Установка адреса модуля.
AT+NAME	+NAME=**ИМЯ** OK	Запрос имени модуля.
AT+NAME**ИМЯ**	OK	Команда установки имени модуля
AT+PIN	+PIN=**КОД** OK	Запрос кода доступа, который необходимо ввести при сопряжении с другими устройствами
AT+PIN**КОД**	OK	Команда установки кода доступа
AT+BAUD	+BAUD=**СКОРОСТЬ**	Запрос скорости передачи/приема данных модулем (бод). 1-1200 2-2400 3-4800 4-9600 5-19200 6-38400 7-57600 8-115200 9-230400 A-460800 B-921600 C-1382400

3.5 Загрузка программ в контроллер по Bluetooth

Для загрузки скетчей в контроллер Вертор Мега через bluetooth модуль необходимо проделать следующие шаги:

1. Убедиться, что на используемом компьютере есть bluetooth передатчик. Он может быть как встроенным, так и подключаться к компьютеру в виде адаптера;
2. Выполнить сопряжение bluetooth модуля контроллера и bluetooth модуля компьютера с помощью стандартных средств настройки;
3. Определить номера двух COM портов, которые были созданы операционной системой компьютера для обмена данными с подключенным bluetooth устройством. Сделать это можно с помощью диспетчера устройств (на рисунке 3.11 приведен пример выделения COM портов 5 и 6 для обмена данными с bluetooth модулем контроллера);

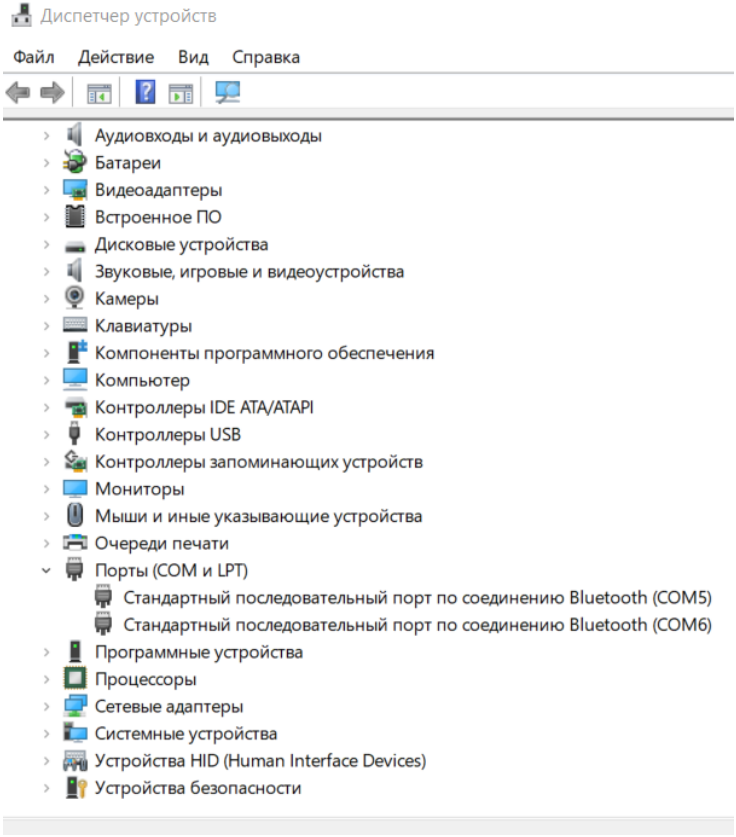

Рис. 3.11

4. В файле boards.txt, находящемся в папке с вашей Arduino IDE (например по пути C:\Program Files (x86)\Arduino\hardware\arduino\avr) найти область с настройками контроллера Вертор Мега (Рис. 3.12) и вставить строку mega.upload.disable_flushing=true (Рис. 3.13).

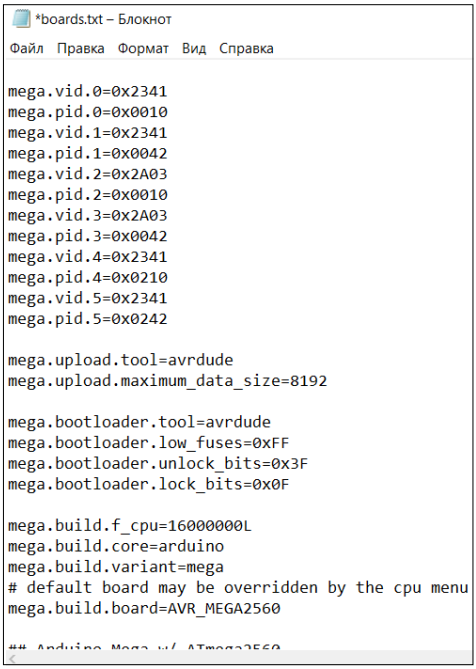

Рис. 3.12

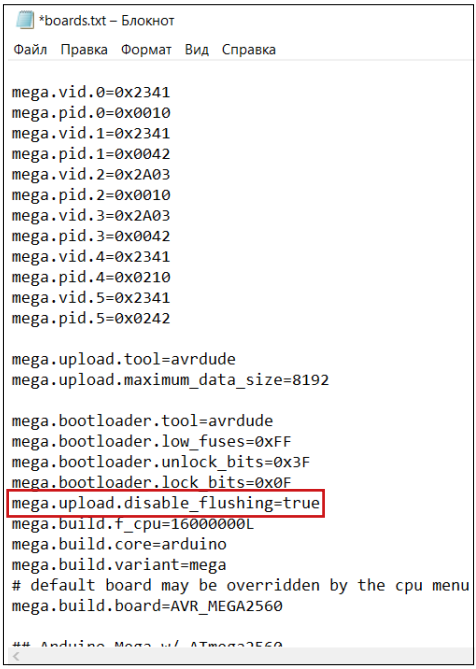

Рис. 3.13

5. При загрузке скетча в контроллер, выбрать соответствующий COM порт (подобрать из двух доступных экспериментально при необходимости) и выполнить загрузку, по завершении которой в статусной строке среды Arduino IDE должно быть указано, что загрузка завершена.

3.6 Особенности управления портами

На плате контроллера напротив каждого разъема нанесена маркировка контактов, используемых в данном разъеме. Она соответствует именам пинов микросхемы микроконтроллера, используемого на плате. Поэтому их можно указывать напрямую при подаче сигналов на контакты или чтения сигналов с пинов посредством использования стандартных операторов digitalwrite(), analogwrite(), или digitalread(), analogread().

Однако на контроллере также присутствуют разъемы, к контактам которых можно обратиться только используя специальные имена или библиотеку. Это разъемы из групп PK PJ.

Ниже приведена таблица с информацией по именам, которые необходимо использовать в Arduino IDE для работы с контактами PK0-PK7.

Маркировка вывода	Обращение к выводу в скетче
PK0	A8
PK1	A9
PK2	A10
PK3	A11
PK4	A12
PK5	A13
PK6	A14
PK7	A15

А вот для работы с разъемами порта PJ необходимо применять специально созданную для этих целей библиотеку «MEGAIO».

В качестве примера использования данной библиотеки ниже приведен скетч, в котором реализован алгоритм изменения уровня напряжения на пине PJ6 два раза в секунду и вывод значения сигнала на пине PJ4 в последовательный порт один раз в секунду. Обратите внимание, что для подачи сигналов или их чтения применяются свои библиотечные операторы с двойным двоеточием.

```
#include "MEGAIO.h"                                   // Подключение библиотеки <MEGAIO.h> для работы с пинами разъемов группы №5
void setup() {
Serial.begin(9600);                                   // Инициализация последовательного порта на скорость работы 9600 бод
MEGAIO::pinMode(MEGAIO::PJ_6, OUTPUT);                // Настройка пина PJ6 на вывод
MEGAIO::pinMode(MEGAIO::PJ_4, INPUT); }               // Настройка пина PJ4 на ввод
void loop() {
MEGAIO::digitalWrite(MEGAIO::PJ_6, HIGH);             // Установить на пине PJ6 напряжение, соответствующее уровню HIGH
delay(500);                                           // Ожидание 0,5 секунды
MEGAIO::digitalWrite(MEGAIO::PJ_6, LOW);              // Установить на пине PJ6 напряжение, соответствующее уровню LOW
delay(500);                                           // Ожидание 0,5 секунды
Serial.println(MEGAIO::digitalRead(MEGAIO::PJ_4)); }  // Вывод в монитор порта Serial значения напряжения на пине PJ4
```

3.7 Управление моторами и мобильными шасси с помощью контроллера

Функционал контроллера по управлению двигателями реализуется с помощью следующего набора библиотек:

Board

DualWheelTruck

I2CTransport

PCA9685

Servo

SmoothServo

STM8_I2C

STM8_I2C_standart

TimerOne

Tweak

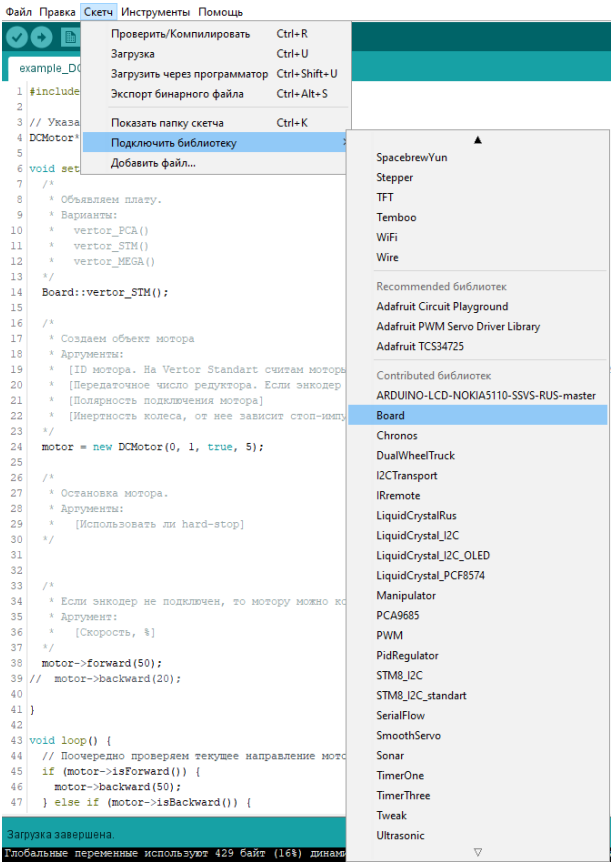

Рис. 3.14

Для возможности использования в скетчах их необходимо скачать с сайта Академии Эвольвектор и поместить в папку libraries, находящуюся в директории с установленной средой программирования Arduino IDE. Если в процессе перемещения библиотек в указанную папку будет выдано предупреждение, содержащее сообщение о том, что такие файлы и папки уже содержатся по указанному пути, то необходимо выполнить их замену.

После размещения библиотек необходимо проверить корректность их установки. Для этого следует выполнить запуск Arduino IDE, нажать левой кнопкой мыши по пункту «Скетч», после чего выбрать пункт меню «Подключить библиотеку» и если в выпавшем списке содержатся только что загруженные библио-

теки (Рис. 3.14), то библиотеки размещены верно.

Порядок использования указанных библиотек сводится к следующим действиям:

1. В начале скетча производится подключение файлов библиотек.
2. Выполняется инициализация каналов передачи данных и объектов, управление которыми будет осуществляться с помощью библиотек.
3. В «теле» скетча вызываются конкретные библиотечные функции, через которые выполняется управление подключенными к контроллеру двигателями или сервомоторами.

С командами, которые обеспечивают выполнение указанных действий, можно ознакомиться с помощью примера скетча example_DCMotor, находящегося в скачанном архиве в папке Sketches. В этом примере продемонстрировано управление ДПТ.

```
#include <DualWheelTruck.h>                 //Подключение библиотеки
DCMotor* motor;                             //Создание объекта с именем motor, которым будет
                                            //осуществляться управление
Board::vertor_MEGA();                       //Выбор типа используемой платы
```

Для инициализации мотора используется функция new DCMotor (a, b, c, d), где:

a - номер подключенного мотора (если мотор подключен к выводам мотора M1, то в качестве этого аргумента необходимо записать 1, если M2- – 2, M3 – 3, M4 – 4, M5 – 5, M6 – 6);

b - передаточное число редуктора (если энкодер установлен на выходном валу мотора или не используется в этом проекте, то передаточное число равно 1);

c - полярность подключения двигателя, которая необходима, когда требуется поменять направление вращения вала двигателя без переподключения проводов двигателя к клеммной колодке (принимает значения true или false);

d - число, которое характеризует степень инерционности вращающихся деталей двигателя или колесной модели в целом (принимает значение от 0 до 255).

```
motor = new DCMotor(1, 1, true, 5);     //Инициализация мотора, подключенного к клеммникам мото-
                                        ра M1, с передаточным числом редуктора, равным 1, с прямой
                                        полярностью подключения моторов и с инерционностью вала,
                                        равной 5
```

Для управления мотором используются функции:

```
motor->isForward();     //Функция isForward() возвращает значение true, если вал
                        //мотора вращается вперед, и значение false в ином случае
motor->isBackward();    //Функция isBackward() возвращает значение true, если вал
                        //мотора вращается назад, и значение false в ином случае
motor->isBusy();        //Функция isBusy() возвращает значение true, если вал
                        //мотора вращается, и значение false, если находится в покое
motor->forward(a);      //Вращать вал мотора вперед со скоростью a
                        //(a измеряется в % и изменяется от 0 до 100)
```

```
motor->backward(a);        //Вращать вал мотора назад со скоростью a
                           //(a измеряется в % и изменяется от 0 до 100)
motor->stop(false)         //Функция, осуществляющая остановку вращения вала мотора
motor->backward(50);       //вращать вал мотора назад со скоростью 50% от максимальной
motor->forward(50);        //вращать вал мотора вперед со скоростью 50% от максимальной
```

Если в проекте необходимо выполнять точное позиционирование вала мотора, то необходимо применять энкодер. Пример работы с мотором, на валу которого установлен энкодер, находится в папке example_DCMotor_with_Encoder. При выборе разъема, к которому будет осуществляться подключение энкодера, обратите особое внимание на то, что нельзя выбирать разъемы с линиями порта PJ, так как работа с ними (чтение и передача данных) может быть осуществлена только с помощью специальной библиотеки, описанной в пункте 3.6

Частично процесс инициализации такого мотора и используемые для его управления функции похожи на описанные выше.

```
#include <DualWheelTruck.h>      //Подключение библиотеки
DCMotor* motor;                  //Создание объекта с именем motor, которым будет
                                 //осуществляться управление
BaseTickEncoder* encoder;        //Создание объекта энкодер, посредством которого будет
                                 //осуществляться управление энкодером и получение от него данных
void motorTick() {               //Функция для установки частоты опроса мотором энкодера
motor->externalTick(1);          //Установка частоты 1 мс обращения мотора к энкодеру
}
Board::vertor_MEGA();            //Выбор типа используемой платы
```

Для инициализации двигателя используется функция **new DCMotor(a, b, c, d)**, назначение аргументов которой точно такое же, как в предыдущем примере:

```
motor = new DCMotor(1, 1, true, 5);   //Инициализация мотора, подключенного к клеммникам мотора
                                      M1, с передаточным числом, равным 1, с прямой полярностью
                                      подключения моторов и с инерционностью вала, равной 5
```

Инициализация оптического энкодера производится с помощью функции
new OpticalTickEncoder(a, b), где:
a – номер пина контроллера, к которому подключен вывод DO оптического энкодера;
b – количество секторов на стороне черно белого диска, повернутой к чувствительному элементу оптического энкодера.

```
encoder = new OpticalTickEncoder(10, 36);   //Инициализация энкодера, вывод DO
                                            //которого подключен к контакту 10 и с рабочей
                                            //стороной черно-белого диска с 36-ю секторами
```

Инициализация магнитного энкодера (датчик Холла) производится с помощью функции **new HallTickEncoder(a, b, c)**, где:
a - первый номер пина контроллера, к которому подключен вывод магнитного энкодера;
b - второй номер пина контроллера, к которому подключен вывод магнитного энкодера;
c - количество импульсов на выводе энкодера за один оборот вала мотора.

```
encoder = new HallTickEncoder(10, 8, 1),        //Инициализация энкодера, выводы которого
                                                //подключены к цифровым пинам 10 и 8 контрол-
                                                лера
                                                //и на выходе которого формируется 1 импульс за
                                                один оборот вала двигателя
motor->attachEncoder(encoder);                  //Функция для подключения мотора с именем motor
                                                //к энкодеру с именем encoder
Tweak::attachMsCallback(motorTick, 1);          //Функция для вызова функции
                                                //motorTick каждую мс
```

Для управления мотором используются функции:

```
motor->isForward();         //Функция isForward() возвращает значение true, если вал
                            //мотора вращается вперед, и значение false в ином случае
motor->isBackward();        //Функция isBackward() возвращает значение true, если вал
                            //мотора вращается назад, и значение false в ином случае
motor->isBusy()             //Функция isBusy() возвращает значение true, если вал
                            //мотора вращается, и значение false, если находится в покое
motor->forward(a,b);        //Вращать вал мотора вперед со скоростью a
                            //(измеряется в % и изменяется от 0 до 100), выполнить b оборотов
motor->backward(a,b);       //Вращать вал мотора назад со скоростью a
                            //(измеряется в % и изменяется от 0 до 100), выполнить b оборотов
motor->stop(false);         //Функция, осуществляющая остановку вращения вала мотора
```

С использованием данного набора библиотек есть возможность реализовать не только независимое управление подключенными моторами, но и шасси в целом. Типичный вид такого шасси приведен на рисунке 3.15.

Пример рассматриваемой программы находится в папке sketches/truck/**DualWheelTruck_STM**.

Для создания нового объекта - двухколесного шасси - и его настройки применяется функция **DualWheelTruck(a, b)**, где:
a - радиус колеса (половина диаметра) в мм;
b - расстояние между колесами в мм;

```
DualWheelTruck truck = DualWheelTruck(30, 230);     //Создание шасси с именем truck,
                                                    //имеющего радиус колеса 30 мм и межколесное
                                                    расстояние 230 мм
```

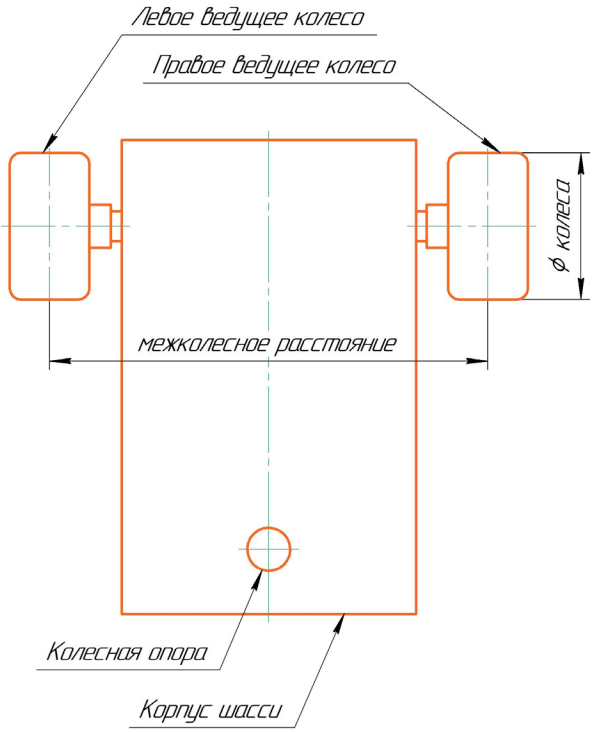

Рис. 3.15

```
Board::vertor_MEGA();      //Объявление разновидности используемой платы

truck.init();              //Инициализация шасси
```

Для настройки левого и правого моторов шасси применяются соответственно функции **truck.attMotorL(a, b, c, d)** и **truck.attMotorR(a, b, c, d)**, аргументы которых имеют следующее назначение:

a - номер подключенного мотора (если мотор подключен к выводам мотора M1, то в качестве этого аргумента необходимо записать 1, если M2 – 2, M3 – 3, M4 – 4, M5 – 5, M6 – 6);

b - передаточное число редуктора (если энкодер установлен на выходном валу мотора или не используется в этом проекте, то передаточное число равно 1);

c - полярность подключения двигателя, которая необходима, когда требуется поменять направление вращения вала двигателя без переподключения проводов двигателя к клеммной колодке (принимает значения true или false);

d - число, которое характеризует степень инерционности вращающихся деталей двигателя или колесной модели в целом (принимает значение от 0 до 255).

```
truck.attMotorL(1, 75, true, 40);
truck.attMotorR(2, 75, false, 40);
```

Далее осуществляется подключение энкодеров. При инициализации левого и правого оптических энкодеров применяются функции соответственно **truck.attOptEncL(a, b)** и **truck.attOptEncR(a, b)**, где:

a - номер пина контроллера, к которому подключен вывод DO оптического энкодера;

b - количество секторов на стороне черно-белого диска, повернутой к чувствительному элементу оптического энкодера.

При инициализации левого и правого магнитных энкодеров применяются функции соответственно **truck.attHallEncL(a, b, c)** и **truck.attHallEncR(a, b, c)**, где:

a - первый номер пина контроллера, к которому подключен вывод магнитного энкодера;

b - второй номер пина контроллера, к которому подключен вывод магнитного энкодера;

c - количество импульсов на выводе энкодера за один оборот вала мотора.

```
truck.attHallEncL(7, A2, 1);    //Инициализация левого магнитного энкодера,
                                //подключенного к выводам 7 и A2 контроллера и формирующего
                                //на выходе 1 импульс за один оборот вала мотора
truck.attHallEncR(10, 8, 1);    //Инициализация правого магнитного энкодера,
                                //подключенного к выводам 10 и 8 контроллера и формирующего
                                //на выходе 1 импульс за один оборот вала мотора
```

Для регулирования моторов с энкодерами в целях обеспечения более точного прямолинейного движения или поворота шасси, в библиотеке применяется пропорционально-интегрально-дифференциальный регулятор. Он также требует инициализации и настройки. Выполняется это с помощью нижеприведенных функций, аргументами которой выступают коэффициенты, влияющие на характер работы регулятора. У них есть базовые значение, относительно которых можно производить более тонкую настройку ПИД-регулятора.

Для прямолинейного движения модели:

truck.setPidAlign (Kp, Ki, Kd);

Здесь:

Kp - коэффициент пропорциональной составляющей (базовое значение 30);

Ki - коэффициент интегральной составляющей (базовое значение 0,1);

Kd - коэфф дифференциальной составляющей (базовое значение 10);

Для поворота модели:

truck.setPidDiff (Kp, Ki, Kd);

Здесь:

Kp - коэф пропорциональной составляющей (базовое значение 10);

Ki - коэф интегральной составляющей (базовое значение 0,1);

Kd - коэф дифференциальной составляющей (базовое значение 5);

```
truck.setPidAlign(30, 0.1, 10);     //Настройка ПИД-регулятора для прямолинепйногом
truck.setPidDiff(10, 0.1, 5);       //перемещения и поворота шасси

truck.enEncoders();                 //Включение энкодеров

truck.disEncoders();                //Отключение энкодеров
```

Для управления шасси применяются следующие функции:

```
truck.motorLFwd(speed, distance);   //вращение левого колеса вперед со скоростью speed (измеряется в процентах от 0 до 100) и на расстояние, равное distance см
truck.motorLBwd(speed, distance);   //вращение левого колеса назад со скоростью speed (измеряется в процентах от 0 до 100) и на расстояние, равное distance см
truck.motorLStop(false);            //остановить вращение левого колеса
```

Аналогичные команды для правого колеса:

```
truck.motorRFwd(speed, distance);
truck.motorRBwd(speed, distance);
truck.motorRStop(false);
```

Для управления платформой целиком применяются функции:

```
truck.moveFwd(speed, distance);     //Проехать вперед со скоростью speed (измеряется в процентах от 0 до 100) на расстояние distance см
truck.moveBwd(speed, distance);     //Проехать вперед со скоростью speed (измеряется в процентах от 0 до 100) на расстояние distance см
truck.turnL(speed, radius, angle);  //Повернуть налево по окружности радиусом radius см со скоростью speed (измеряется в процентах от 0 до 100) на центральный угол angle
truck.turnR(speed, radius, angle);  //Повернуть направо по окружности радиусом radius см со скоростью speed (измеряется в процентах от 0 до 100) на центральный угол angle
truck.isBusy();                     //Функция, возвращающая 1, если платформа еще не завершила предыдущее перемещение, и 0, если платформа стоит на месте
truck.getDistance();                //Функция, возвращающая суммарную величину пройденного платформой расстояния
truck.resetDistance();              //Функция, осуществляющая сброс отсчета суммарного расстояния, пройденного платформой
```

3.8 Управление серводвигателями с помощью контроллера

Контроллер ВЕРТОР МЕГА поддерживает работу не только с электродвигателями, но и со стандартными хобби-сервоприводами. Для управления серводвигателями предусмотрена отдельная библиотека. Пример для изучения сервопривода находится в папке Sketches и имеет название **example_smoothServo**.

По аналогии с мотор-редукторами скетч должен начинаться с подключения библиотеки и инициализации серводвигателя:

```
#include <SmoothServo.h>          //Подключение библиотеки

SmoothServo* servo;               //Создание объекта с именем servo

Board::vertor_MEGA();             //Объявление разновидности используемой платы
```

Для инициализации сервопривода применяется функция **new SmoothServo(a)**, где
a - номер разъема для подключения сервоприводов.

```
servo = new SmoothServo(0);       /Инициализация сервопривода, подключенного к 0-му разъему
```

Для настройки динамических возможностей сервопривода применяются функции:

```
servo->setMinAngularSpeed(70);        //Установка минимальной угловой скорости вала при  повороте
servo->setMaxAngularSpeed(180);       //Установка максимальной угловой скорости вала при  повороте
servo->setMaxAcceleration(200);       //Установка максимального ускорения вала при повороте

servo->begin(30);                 //Запуск сервопривода
```

Для управления сервоприводом используются функции:

servo->smoothRotate(b); //Плавный поворот вала сервопривода до угла b градусов

servo->rotate(b); //Поворот вала сервопривода до угла b градусов

servo->isBusy(); //Функция осуществляет проверку занятости сервопривода и возвращает логическую «1», если он еще не завершил поворот и логический «0», если предыдущее движение было отработано

4. Технические характеристики

Наименование характеристики	Значение
Размеры контроллера, мм	94x125
Количество разъемов XH-2.54-4P общего назначения	8
Количество разъемов XH-2.54-4P с протоколом I2C	3
Количество разъемов XH-2.54-4P с протоколом UART	3
Максимальное количество подключаемых электродвигателей	6
Максимальное количество подключаемых серводвигателей	12
Размеры разъема питания, мм	5,5x2,1
USB разъем	Тип B
Наличие выключателя питания	Да
Допустимый диапазон входного напряжения питания	12...34 В
Диапазон регулировки напряжения питания электродвигателей при входном напряжении питания от 12 до 34 В	0...11,7 В
Напряжение питания серводвигателей при входном напряжении питания от 12 до 36 В	6
Номинальное рабочее напряжение микроконтроллера и на разъемах общего назначения и разъемах I2C	5 В
Максимальный суммарный ток потребления модулей, подключенных к контроллеру	До 0,5 А
Максимальный суммарный ток потребления моторов, подключенных к контроллеру	До 5 А
Максимальный суммарный ток потребления серводвигателей, подключенных к контроллеру	До 5 А
Максимальный суммарный ток потребления модулей, подключенных к разъемам общего назначения и разъемам I2C	До 0,4 А
Тактовая частота микроконтроллера	16 МГц
Оперативная память	8 кБ
Встроенная Флеш-память	64 Кб (ATmega2560) из которых 4 Кб используются для загрузчика
Программное обеспечение для программирования контроллера	Arduino IDE

5. Условия гарантии

ООО «Эвольвектор» гарантирует работоспособность электронного модуля на протяжении всего гарантийного срока эксплуатации, который составляет 12 месяцев с момента приобретения устройства. Также гарантируется совместимость модуля с другими устройствами системы управляющей электроники ВЕРТОР. Гарантийные обязательства производителя распространяются только на ту продукцию, которая не имеет повреждений и не выведена из строя в результате неверных действий пользователя.

По вопросам гарантийного обслуживания, а также по всем техническим и информационным вопросам можно обращаться на электронную почту:
info@evolvector.ru
help@evolvector.ru
а также по телефону +7 (499) 391-01-05
Адрес для корреспонденции: 143300, Московская область, г. Наро-Фоминск, ул. Московская, д.15.