Программируемый контроллер ВЕРТОР СТАНДАРТ V1.2

Артикул ПЭМ10.3158

Технические данные и руководство пользователя.

1. Назначение устройства

Программируемый контроллер ВЕРТОР СТАНДАРТ (Рис. 1.1) является одним из ключевых элементов системы управляющей электроники «Эвольвектор ВЕРТОР» (далее ВЕРТОР). Он предназначен для создания систем управления различного назначения в рамках основного и дополнительного школьного образования. Рассчитан на применение совместно с электронными модулями, входящими в систему ВЕРТОР (подробная информация о системе представлена в соответствующем разделе сайте https://academy.evolvector.ru).

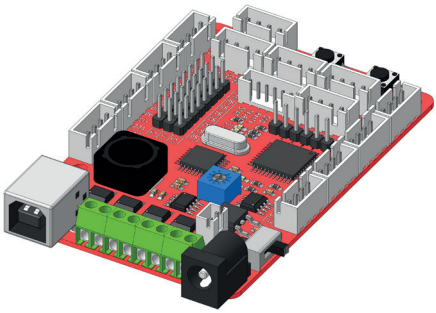

Рис. 1.1

Контроллер может использоваться для создания системы управления учебными автоматическими и робототехническими моделями или электронными устройствами. В рамках стандартных функциональных возможностей контроллер может принимать сигналы от входящих в систему датчиков, обрабатывать их и формировать управляющие сигналы для исполнительных устройств согласно заложенной программе.

2. Конструкция модуля и назначение выводов (контактов)

Контроллер выполнен в виде печатной платы, которая оснащена следующими элементами (Рис. 2.1):

- разъемами типа XH-2.54-4P для подключения совместимых электронных модулей;
- разъемами питания под штекер размером 5,0x2,1 и для подачи электроэнергии от заряжаемого модуля;
- клеммниками для подсоединения электрических двигателей, соответствующих требованиям номинального напряжения и мощности;
- штыревыми контактами для подключения стандартных RC серводвигателей;
- разъемом для подключения одноплатного компьютера Raspberry Pi;
- разъемом для подключения устройств, работающих по последовательному протоколу UART;
- регулятором напряжения для питания электродвигателей и сервоприводов;
- разъемом USB тип B для подключения контроллера к персональному компьютеру и загрузки в него программ управления (скетчей);
- тактовыми кнопками, для которых можно назначить пользовательские функции при реализации проектов и тем самым обойтись без подключения внешних тактовых кнопок;
- выключателем питания для клеммников электродвигателей и контактов сервоприводов;
- кнопкой принудительной перезагрузки контроллера в случае его сбоя или некорректной работы.

Перечисленные элементы представлены на рисунке 2.1.

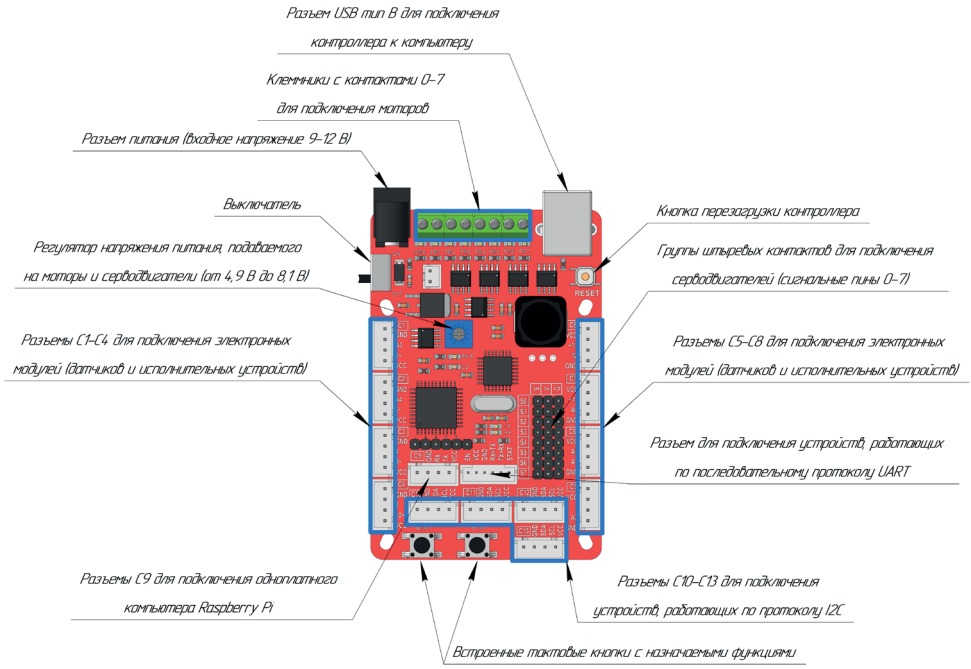

Рис. 2.1

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 одноплатном компьютере Raspberry Pi, что дает возможность крепить эти устройства с помощью стоек одно над другим. Помимо этого, по расстоянию между крепежными отверстиями (кратно 16 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 имеют разные возможности по подключению к ним периферийных устройств (электронных модулей). 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.

Разъемы C1-C8 являются разъемами общего назначения и размещены в два ряда по бокам платы контроллера. У части из них (разъемы C1, C2, C5, C6 и C7) присутствует один цифровой (промаркирован просто числовым номером) и один аналоговый (промаркирован буквой A с номером) контакты. К таким возможно подключение датчиков, выдающих аналоговый сигнал. Несколько разъемов (C3, C4, C8) имеют исключительно цифровые управляющие контакты, поэтому могут работать только с цифровыми датчиками.

Группа разъемов C10-C13 предназначена для подключения устройств, работающих по протоколу передачи данных I2C.

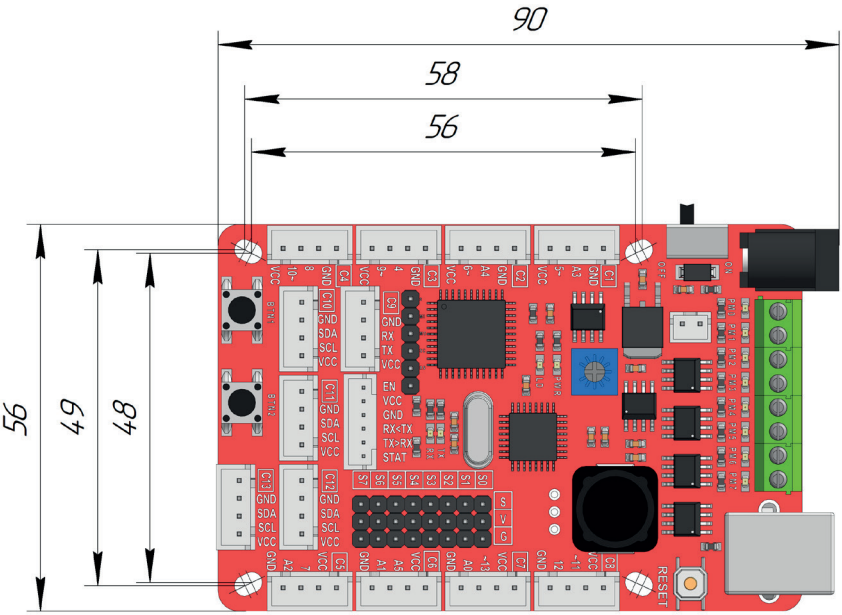

Рис. 2.2

Разъем С9 необходим для обеспечения обмена данными с одноплатным компьютером Raspberry по последовательному интерфейсу UART.

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

3. Принцип работы контроллера.

Контроллер ВЕРТОР СТАНДАРТ построен на основе платформы Arduino Leonardo. Он представляет собой плату с контактами для подключения различных внешних устройств. Контакты сгруппированы в разъемы в виде гнезд, в каждом из которых по 4 контакта: питание, «земля», и 2 контакта для приема или передачи сигналов. Через указанные контакты принимаются сигналы от датчиков и формируются управляющие сигналы для исполнительных устройств.

Наличие четырехконтактных разъемов является преимуществом по сравнению со стандартной платой Arduino Leonardo, которая оснащена колодками с отдельными контактами. Оно выражается в том, что модули могут надежно закрепляться с помощью стоек и винтов на элементах конструкций роботов или любого другого устройства, совместимого по крепежным отверстиям. Также разъемы позволяют более удобно подключать датчики и исполнительные устройства, т.к. для подключения не требуется макетная плата.

Еще одной особенностью контроллера ВЕРТОР СТАНДАРТ является поддержка заметно большего количества одновременно подключаемых устройств по сравнению с классическими платами Arduino Leonardo.

Это возможно благодаря встроенному микроконтроллеру STM8S003, выполняющего роль расширителя портов. В результате обеспечивается возможность подключения сразу 14-ти электронных модулей системы электроники ВЕРТОР, 8-ми серводвигателей небольшой мощности (контакты имеют маркировку S0-S7) и 4-х обычных коллекторных моторов (контакты промаркированы PM0-PM7).

Для организации питания перечисленных моторов и серводвигателей в контроллере предусмотрен преобразователь напряжения, допускающий суммарный ток потребления подключенными моторами до 2,5 А.

То, как принимаются входные сигналы, обрабатываются и генерируются управляющие сигналы, определяется программой (скетчем), загруженной в контроллер. Для загрузки программы необходим только персональный компьютер с разъемом USB и с установленным на него программным обеспечением Arduino IDE. Дополнительные программаторы для программирования контроллера не требуются. С инструкцией по подключению контроллера к компьютеру можно ознакомиться в соответствующем разделе сайта Академии Эвольвектор.

Полный функционал контроллера по управлению двигателями и электронными модулями реализуется с помощью следующего набора библиотек:

Board
DualWheelTruck
I2CTransport
PCA9685
Servo
SmoothServo
STM8_I2C
STM8_I2C_standart
TimerOne
Tweak

Для возможности использования в скетчах их необходимо скачать 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 легко 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 библиотеки (Рис. 3.1), то би-

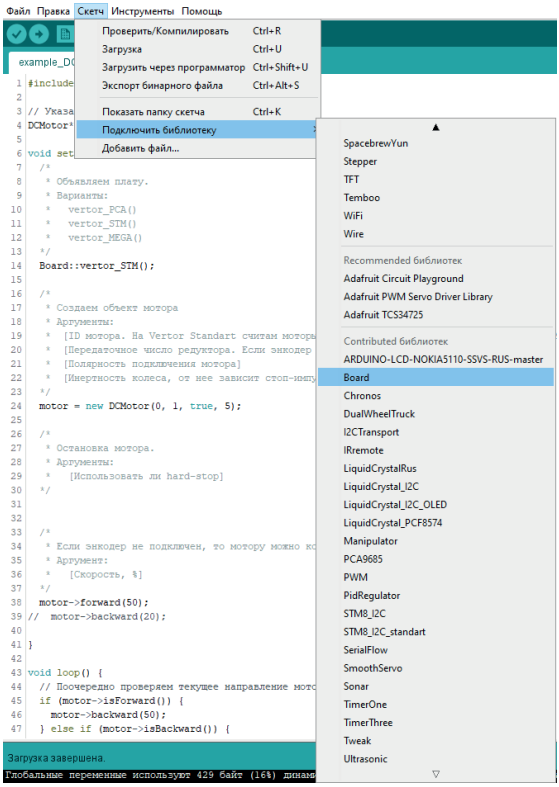

Рис. 3.1

блиотеки 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_STM();          //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 PM0-PM1, то в качестве этого аргумента необходимо записать 0, если PM2-PM3 – 1, PM4-PM5 – 2, PM6-PM7 – 3);

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(0, 1, true, 5);   //Инициализация мотора, подключенного к клеммникам  PM0-PM1,
                                      с передаточным числом, равным 1, с прямой полярностью подклю-
                                      чения моторов и с инерционностью вала, равной 30
```

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 точное позиционирование вала мотора, то необходимо применять энкодер. Пример работы с мотором, на валу которого установлен энкодер, находится в папке **example_DCMotor_with_Encoder.**

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_STM();          //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(0, 1, true, 30);   //Инициализация мотора, подключенного к клеммникам PM0-
                                       PM1, с передаточным числом, равным 1, с прямой полярностью
                                       подключения моторов и с инерционностью вала, равной 30
```

Инициализация оптического энкодера производится с помощью функции **new OpticalTickEncoder(a, b)**, где:

a – номер пина контроллера, к которому подключены 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 подключены вывод магнитного энкодера;

b – второй номер пина контроллера, к которому подключены 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 контроллера
                                         //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 3.2.

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

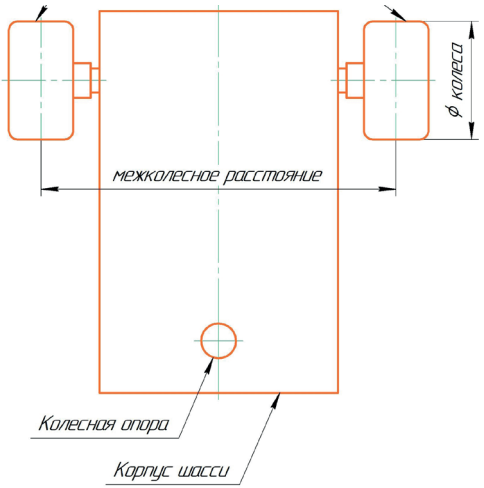

Рис. 3.2

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
Board::vertor_STM();                                //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 PM0-PM1, то он в качестве этого аргумента необходимо записать 0, если PM2-PM3 – 1, PM4-PM5 – 2, PM6-PM7 – 3,);

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(0, 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 требует также инициализации и настройки. Выполняется это с помощью нижеприведенных функций, аргументами которых 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 прямолинепйного
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
truck.motorLFwd(speed, distance);   //вращение левого колеса назад со скоростью speed (измеряется в про-
                                    центах от 0 до 100) и на расстояние, равное distance
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 про-
                                    центах от 0 до 100) на расстояние  distance

truck.moveBwd(speed, distance);     //Проехать назад со скоростью speed (измеряется в процен-
                                    тах от 0 до 100) на расстояние  distance

truck.turnL(speed, radius, angle);  //Повернуть налево по окружности радиусом radius 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 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

Контроллер ВЕРТОР СТАНДАРТ 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_STM();              //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	58x90 мм
Количество разъемов XH-2.54-4P общего назначения	8
Количество разъемов XH-2.54-4P с протоколом I2C	5
Максимальное количество подключаемых электродвигателей	4
Максимальное количество подключаемых серводвигателей	8
Размеры разъема питания	5,0x2,1 мм
USB разъем	Тип B
Наличие выключателя питания	да
Допустимый диапазон входного напряжения питания	7,5...12 В
Диапазон регулировки напряжения питания сервоприводов при входном напряжении питания 7,5 В	4,9...7,2 В
Диапазон регулировки напряжения питания сервоприводов при входном напряжении питания 9 В	4,9...8,2 В
Диапазон регулировки напряжения питания сервоприводов при входном напряжении питания 12 В	4,9...9,2 В
Диапазон регулировки напряжения питания электродвигателей при входном напряжении питания 7,5 В	4,4...6,5 В
Диапазон регулировки напряжения питания электродвигателей при входном напряжении питания 9 В	4,4...7,6 В
Диапазон регулировки напряжения питания электродвигателей при входном напряжении питания 12 В	4,4...8,7 В
Номинальное рабочее напряжение микроконтроллера и на разъемах общего назначения и разъемах I2C	5 В
Максимальный суммарный ток потребления модулей, подключенных к разъемам общего назначения и разъемам I2C	До 0,5 А
Максимальный суммарный ток потребления электроприводов, подключенных к контроллеру	До 2,5 А
Тактовая частота микроконтроллера	16 МГц
Оперативная память	2 кБ
Встроенная Флеш-память	32 Кб (ATmega32u4) 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.