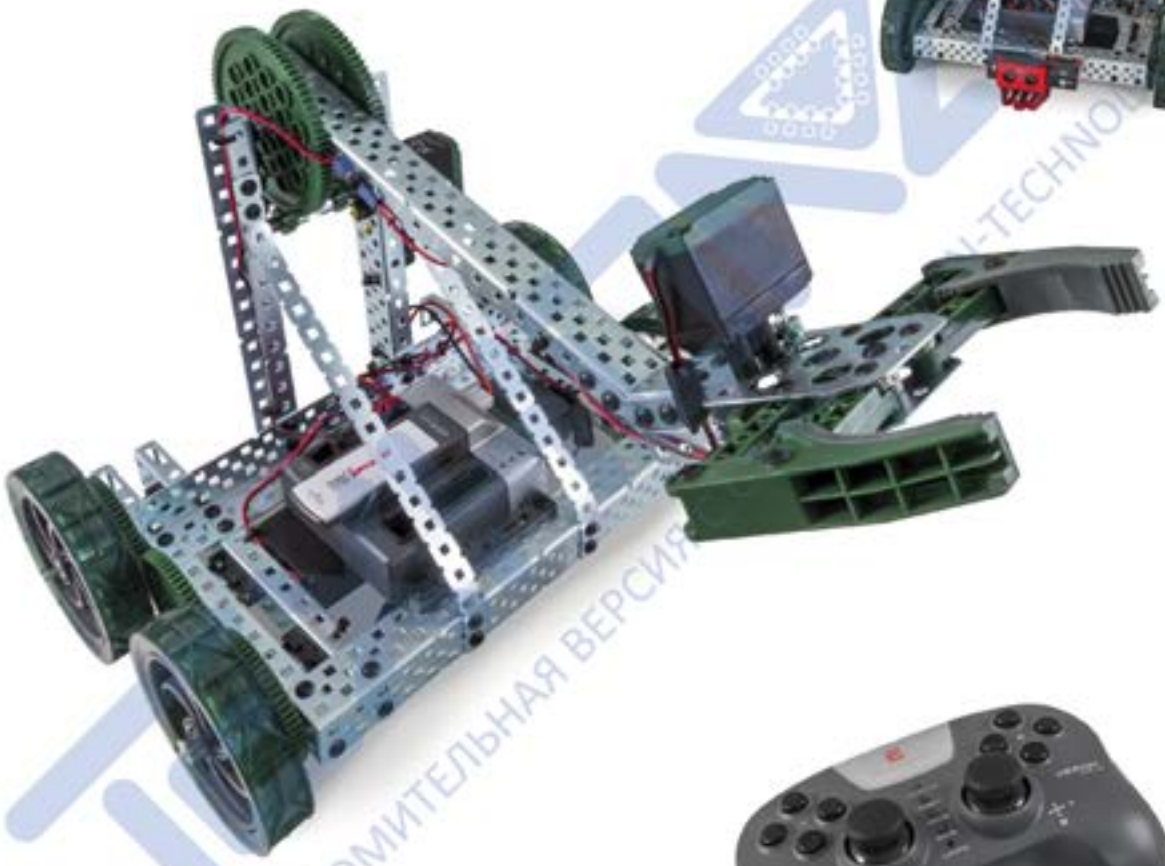
Основы робототехники и программирования с VEX EDR

Учебно-методическое пособие



VEX EDR

О. А. Горнов

Основы робототехники и программирования с VEX EDR

Учебно-методическое пособие

МОСКВА
2016

УДК 372.8:004
ББК 32.816
Г69

Горнов О. А.

Г69 Основы робототехники и программирования с VEX EDR / О. А. Горнов. — М. : Издательство «Экзамен», 2016. — 160 с.
ISBN 978-5-377-11671-4

Предлагаемое пособие содержит сведения о конструировании роботов, начальные сведения о языке С, дает разбор основных алгоритмов управления. Оно поможет организовать проектную деятельность в образовательных организациях.

Пособие предназначено педагогам, учащимся старших классов, студентам технических вузов. Оно будет интересно всем, кто интересуется и занимается робототехникой самостоятельно.

УДК 372.8:004
ББК 32.816

Подписано в печать с диапозитивов 19.09.2016.
Формат 60x90/8. Гарнитура «Calibri». Бумага офсетная.
Усл. печ. л. 20. Тираж 1000 экз. Заказ №

ISBN 978-5-377-11671-4

Содержание

Введение	стр. 5
Руководство по сборке робота	стр. 7
Робот. Функциональная схема робота	стр. 27
Конструкция робота для решения задач автоматического управления	стр. 29
Первоначальные сведения о программировании в языке С	стр. 33
Особенности программирования в RobotC	стр. 43
Тайминговый контроль перемещений робота. Простейшие передвижения робота	стр. 53
Движения с контролем оборота двигателей	стр. 59
Автономное движение робота с объездом препятствий за счет применения датчиков касания	стр. 63
Датчик освещенности. Танец в круге	стр. 69
Движение по линии на одном датчике	стр. 75
Сложные ветвления. Пульт из датчиков касания	стр. 77
Релейный регулятор. Удерживание подъемного устройства манипулятора	стр. 81
Движение по линии на одном датчике с использованием релейного регулятора	стр. 83
Движение вдоль стены по датчику расстояния с использованием релейного регулятора	стр. 85
Движение вдоль линии на двух датчиках	стр. 89
Пропорциональный регулятор. Удерживание манипулятора	стр. 93
Езда по линии на одном датчике и вдоль стены на пропорциональном регуляторе	стр. 97
Точные движения робота, основанные на использовании пропорционального регулятора и энкодеров	стр. 99

Езда по линии на двух датчиках освещенности с использованием пропорционального регулятора стр. 103

Движение по линии с использованием пропорционально-кубического регулятора стр. 107

Движение по линии с использованием пропорционально-дифференциального регулятора стр. 109

Пульт управления роботом стр. 111

Управление движениями робота на omni-колесах стр. 115

Управление роботом на omni-колесах с пульта управления стр. 117

Послесловие стр. 119

VEX Robotics Competition Starstruck - Регламент соревнований. Часть 1 - Введение стр. 121

VEX Robotics Competition Starstruck - Регламент соревнований. Часть 2 - Соревнования стр. 123

VEX Robotics Competition Starstruck - Регламент соревнований. Часть 3 - Турнир стр. 137

VEX Robotics Competition Starstruck - Регламент соревнований. Часть 4 - Робот стр. 145

Введение

Здравствуйте, уважаемые друзья!

Вашему вниманию представляется учебное пособие, посвященное программированию роботов VEX-EDR в среде RobotC. Это пособие построено как справочник, в который Вы можете заглянуть для решения какой-либо практической задачи робототехники. Поэтому для удобства использования изложение построено по модульному принципу.

Первый модуль посвящен основным принципам конструирования мобильных роботов.
Второй – основам программирования языка С.
Третий – управлению движением робота без использования обратных связей.
Четвертый – решению задач автоматического управления.
Пятый – управлению роботом с пульта управления.

В зависимости от Вашего желания возможно начинать изучение материала с любого модуля или использовать программный код из любого раздела для решения своей задачи. Рекомендуем Вам использовать модульный подход и в программировании робота для конкретной задачи. То есть уже на этапе проектирования робота следует понимать, какие из программных модулей пригодятся Вам для «оживления» робота.

В том случае если у Вас нет опыта программирования роботов, рекомендуем самостоятельно набрать все строки кода из данного пособия, изучить, как повлияет на поведение робота изменение какого-либо параметра. Постарайтесь провести как можно больше экспериментов с уже написанным автором пособия кодом.

Почти в каждом параграфе приведены задачи, которые возможно использовать для построения учебного процесса на основе уже изученного материала. Прошу Вас относиться к любому параграфу как к точке старта для Ваших фантазий и фантазий Ваших учеников. Придумывайте новые интересные задания, создавайте роботов!

Все вопросы, замечания и пожелания отправляйте по адресу vex.edr@yandex.ru

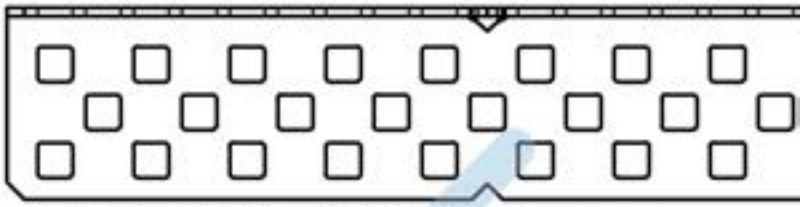
С уважением, Горнов Олег Александрович.

Для заметок

Руководство по сборке робота

5 ОТВЕРСТИЙ

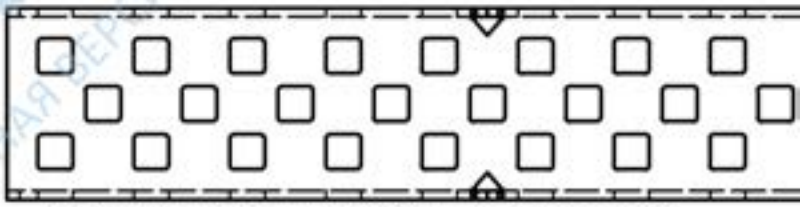
10 ОТВЕРСТИЙ



[A20] РАМА УГЛОВАЯ (20 ОТВЕРСТИЙ В ДЛИНУ)

[R16] РАМА (16 ОТВЕРСТИЙ В ДЛИНУ)

[C20] СОЕДИНИТЕЛЬНАЯ ПЛАСТИНА (20 ОТВЕРСТИЙ В ДЛИНУ)

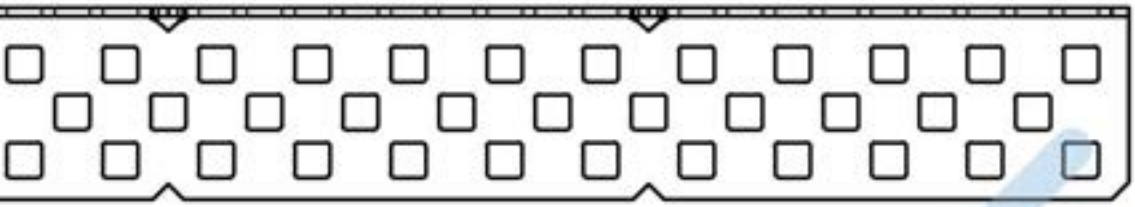


[C15] СОЕДИНИТЕЛЬНАЯ ПЛАСТИНА (15 ОТВЕРСТИЙ В ДЛИНУ)

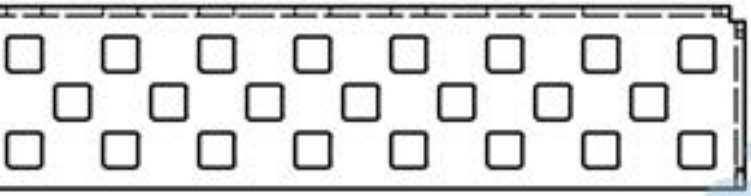
[B20] ПЛАСТИНА С 20 ОТВЕРСТИЯМИ

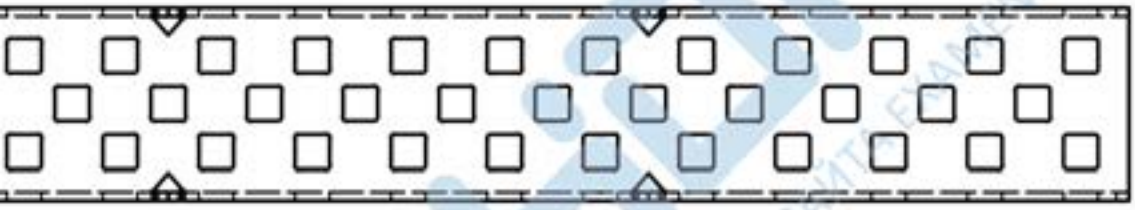
[SH-3] ВАЛ (3 ДЮЙМА В ДЛИНУ)

15 ОТВЕРСТИЙ

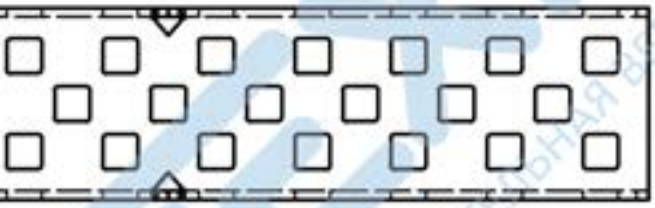


20 ОТВЕРСТИЙ





25 ОТВЕРСТИЙ



[ZIP] СТЯЖКА

[BF]
ОПОРНАЯ ПЛАНКА

[ST-1]
РАЗДЕЛИТЕЛЬ

[BR-I]
ЗАКЛЕПКА
(ВНУТР.)

[BR-O]
ЗАКЛЕПКА
(ВНЕШН.)

[CPLR]
СОЕДИНИТЕЛЬ
ВАЛОВ

[SP4.8]
РАЗДЕЛИТЕЛЬ (4.8 ММ)

[NK]
8-32 ГАЙКА

[CP]
СОЕДИНИТЕЛЬНЫЙ
ЭЛЕМЕНТ (ВНУТР.)

[COL]
НАКОНЕЧНИК ВАЛА
С ВИНТОМ
8-32 x 1/8 ДЮЙМА

[NL]
8-32 ГАЙКА
С НЕЙЛОНОВЫМ
ДЕРЖАТЕЛЕМ

[S12]
ВИНТ 8-32x1.5 ДЮЙМА

[SS-L]
ДЛИННЫЙ ВИНТ МОТОРА
С НЕЙЛОНОВЫМ ДЕРЖАТЕЛЕМ

[S4]
ВИНТ 8-32x1/2 ДЮЙМА

[SS-S]
КОРОТКИЙ ВИНТ МОТОРА
С НЕЙЛОНОВЫМ ДЕРЖАТЕЛЕМ

[S2]
ВИНТ 8-32x1/4 ДЮЙМА

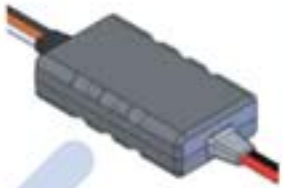
ГАЕЧНЫЙ КЛЮЧ

5/64"

3/32"

[CLAW]
УСТРОЙСТВО СХВАТА

[M393]
ДВУХПРОВОДНОЙ
МОТОР 393



[MC29]
КОНТРОЛЛЕР
МОТОРА 29

[BST]
КРЕПЛЕНИЕ БАТАРЕИ

[CTX]
МИКРОКОНТРОЛЛЕР

[VNET]
КЛЮЧ VEXnet

[BATT]
ПЕРЕЗАРЯЖАЕМАЯ
БАТАРЕЯ 7.2В

[G12]
ШЕСТЕРНЯ
(12 ЗУБЬЕВ)

[G60]
ШЕСТЕРНЯ
(60 ЗУБЬЕВ)

[G84]
ШЕСТЕРНЯ
(84 ЗУБЬЕВ)

[W4]
КОЛЕСО 4 ДЮЙМА

1

[SS-S]
1/4" ВИНТ МОТОРА

[SS-L]
1/2" ВИНТ МОТОРА

2

[NK]
8-32 ГАЙКА

[NL]
8-32 ГАЙКА С
НЕЙЛОНОВЫМ
ДЕРЖАТЕЛЕМ

[ST-1]
РАЗДЕЛИТЕЛЬ

3

[COL]
НАКОНЕЧНИК ВАЛА

4

[S2] 1/4" ВИНТ

[S4] 1/2" ВИНТ

[S12] 1.5" ВИНТ

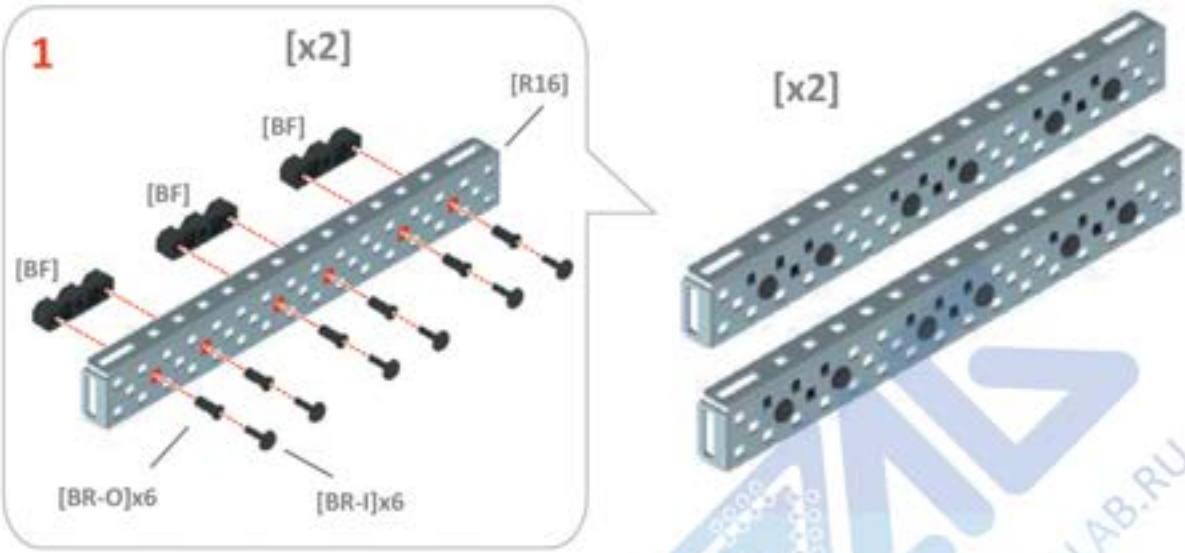
5

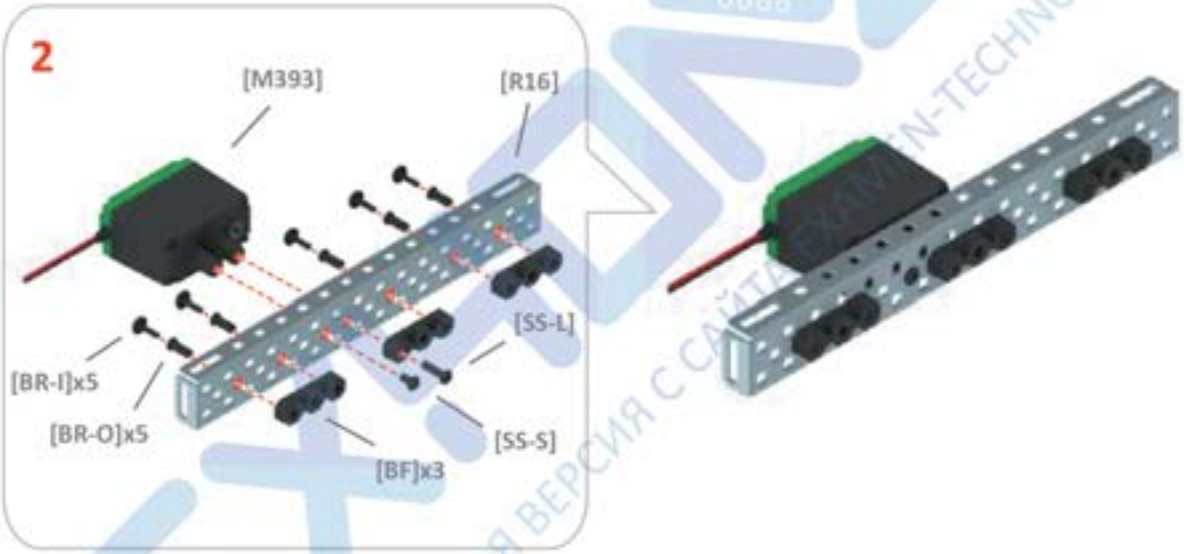
[BR] ЗАКЛЕПКИ

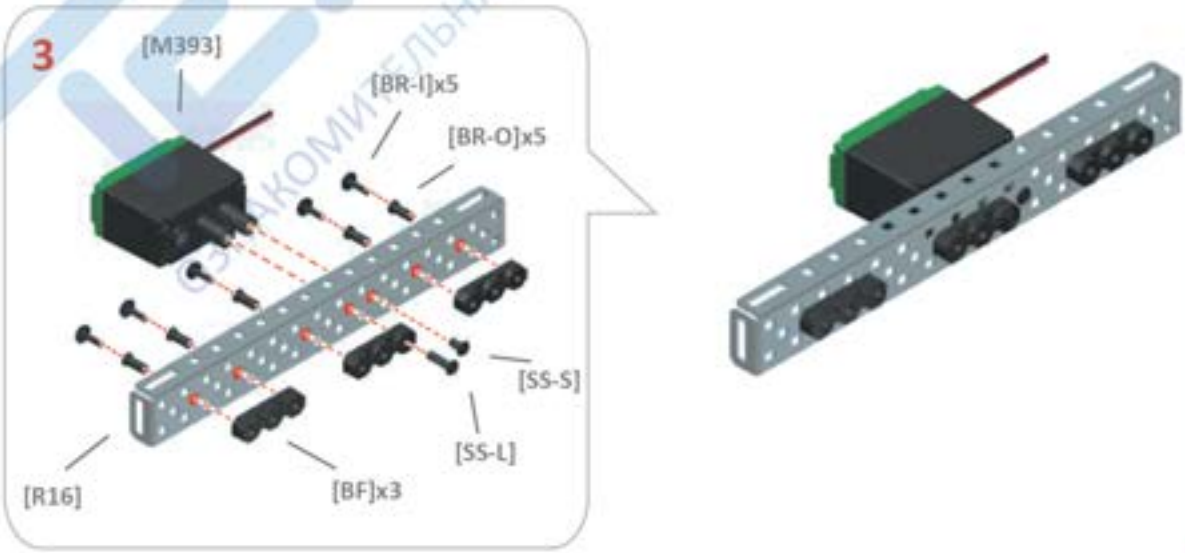
[BF] ОПОРНАЯ ПЛАНКА

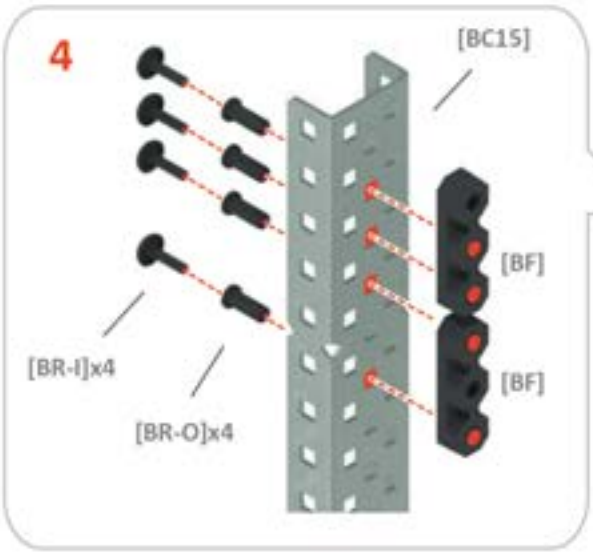
6

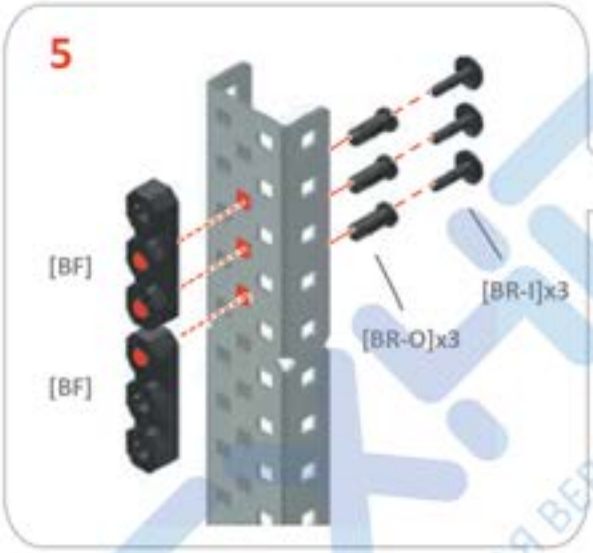
[BF] ОПОРНАЯ ПЛАНКА

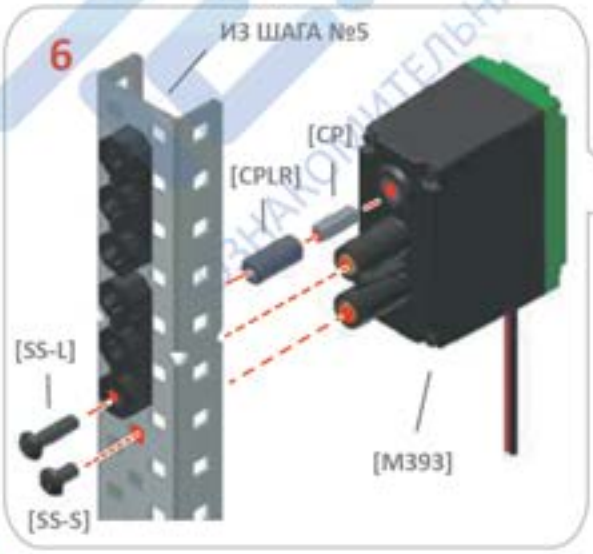




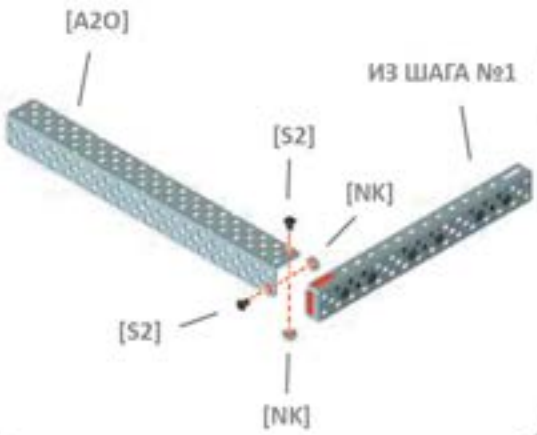




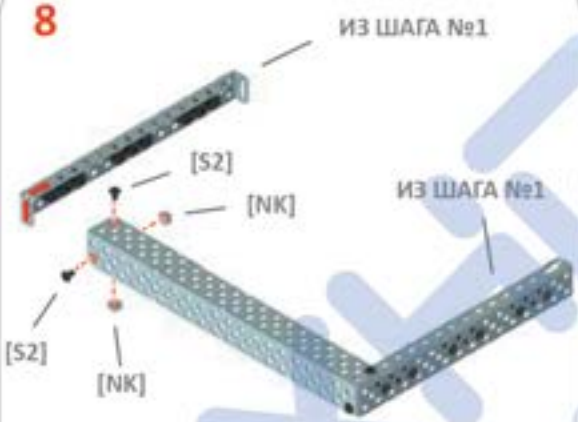




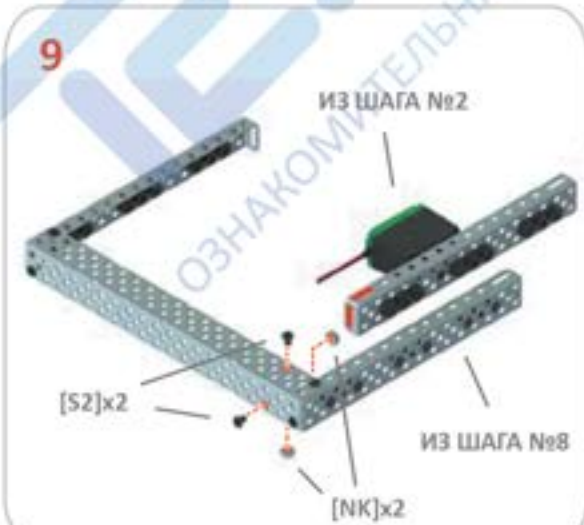
7

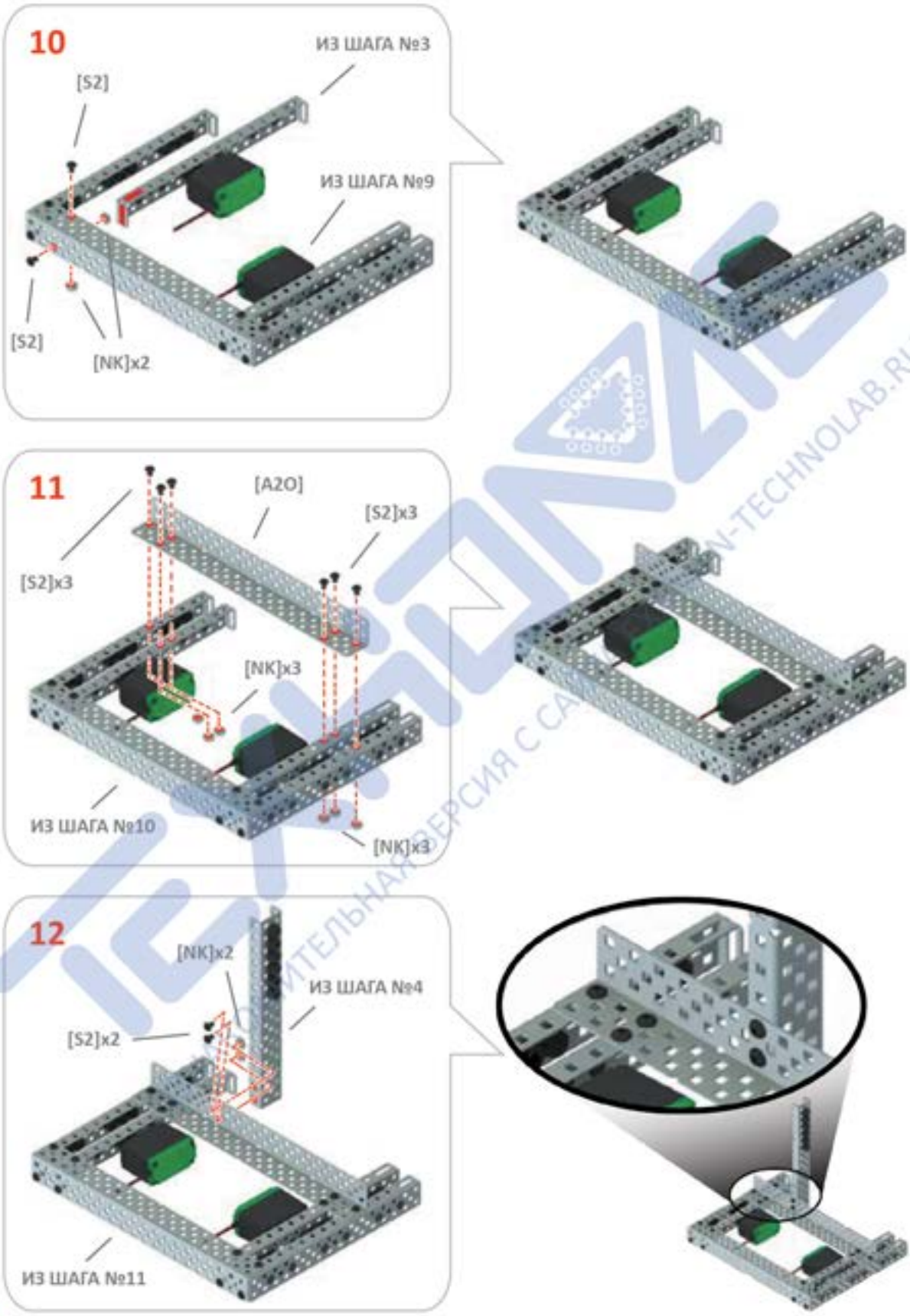


8

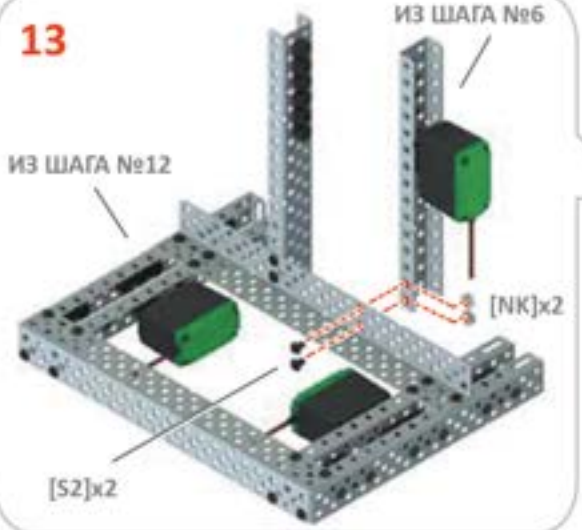


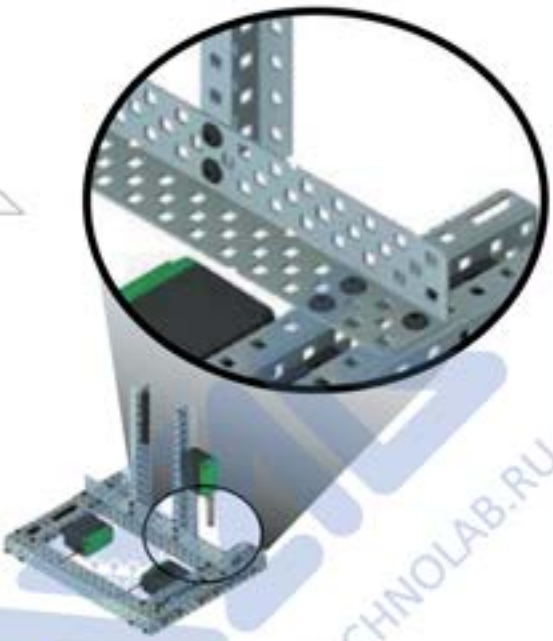
9



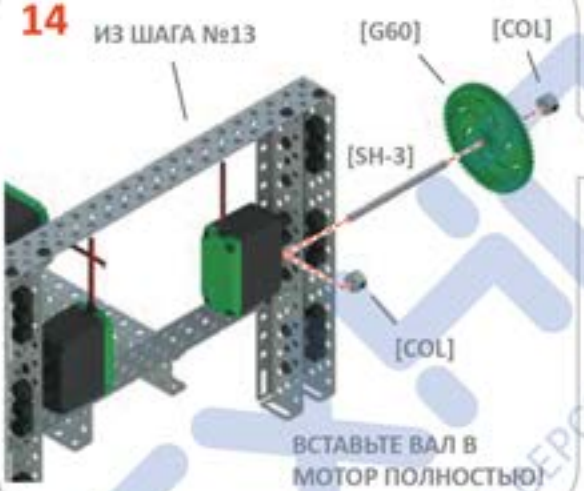

10
ИЗ ШАГА №3
[S2]
ИЗ ШАГА №9
[S2]
[NK]x2
11
[A2O]
[S2]x3
[S2]x3
[NK]x3
ИЗ ШАГА №10
[NK]x3
12
[NK]x2
ИЗ ШАГА №4
[S2]x2
ИЗ ШАГА №11

13

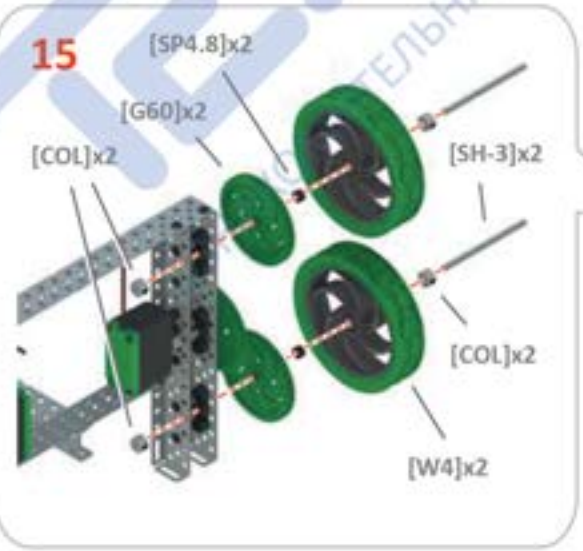


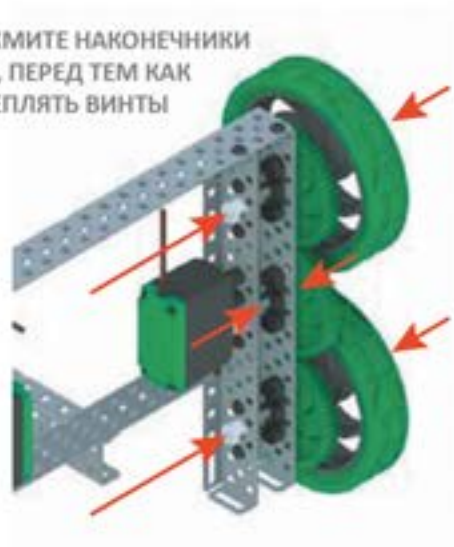


14

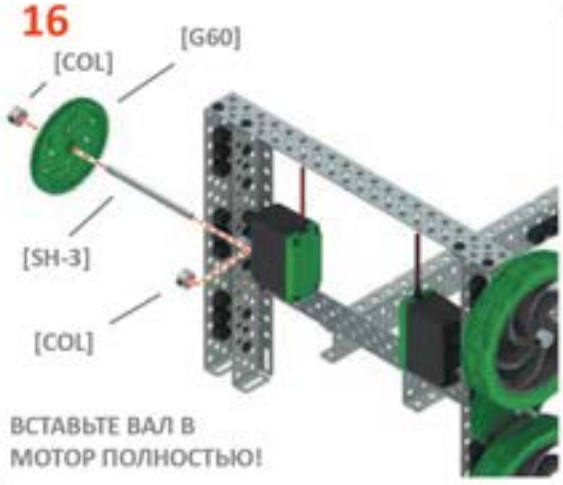


15



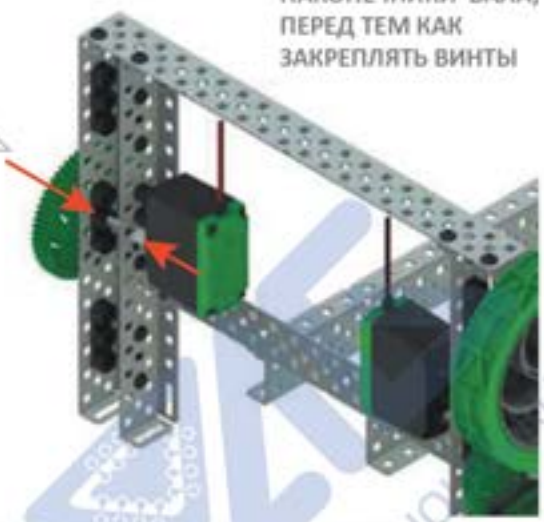


16

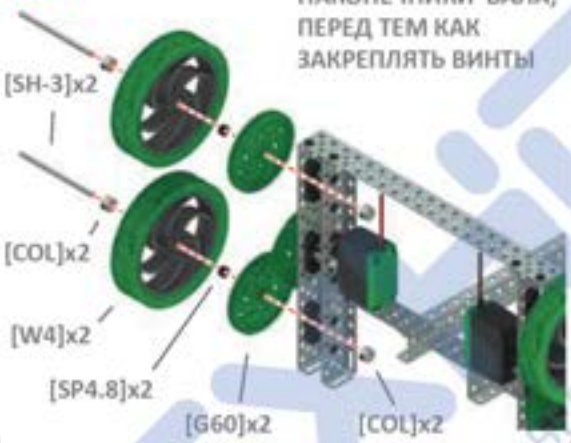


ВСТАВЬТЕ ВАЛ В МОТОР ПОЛНОСТЬЮ!

ПРИЖМИТЕ НАКОНЕЧНИКИ ВАЛА, ПЕРЕД ТЕМ КАК ЗАКРЕПЛЯТЬ ВИНТЫ

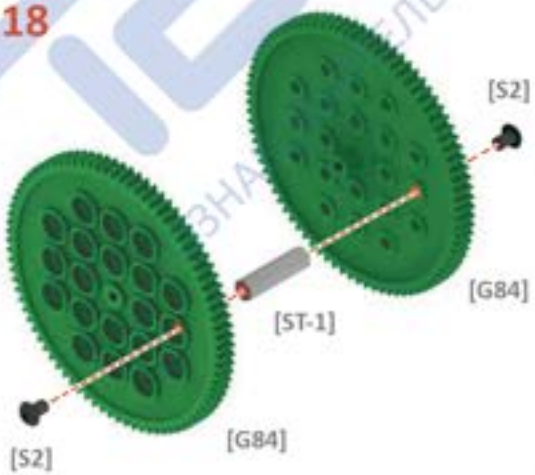


17

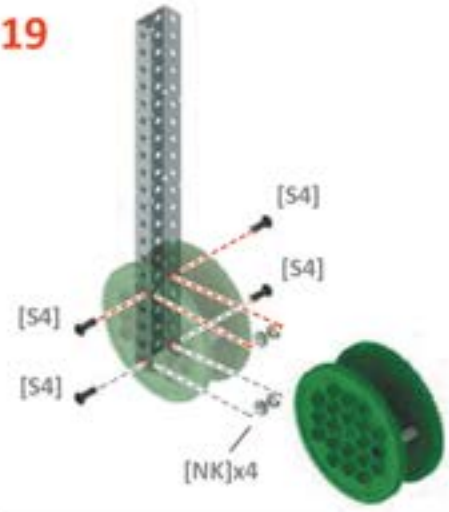


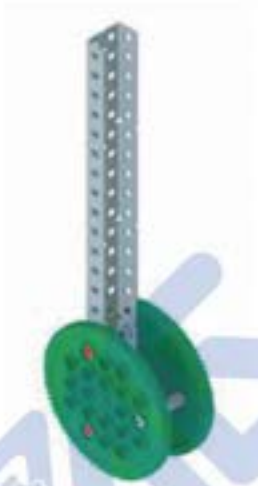
ПРИЖМИТЕ НАКОНЕЧНИКИ ВАЛА, ПЕРЕД ТЕМ КАК ЗАКРЕПЛЯТЬ ВИНТЫ

18

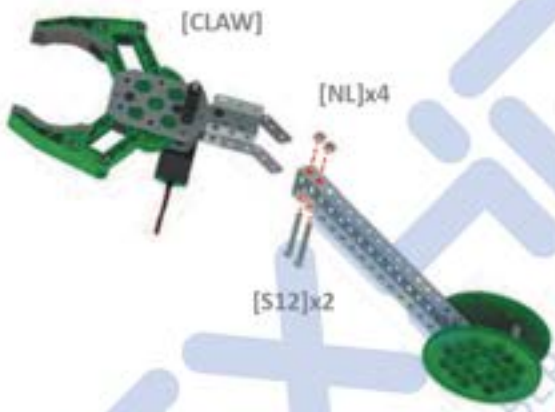


19

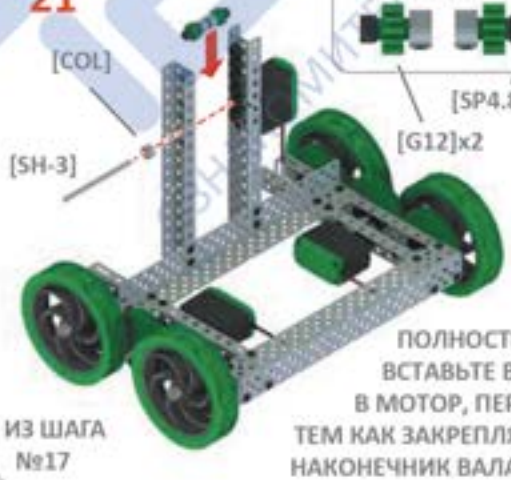




20



21

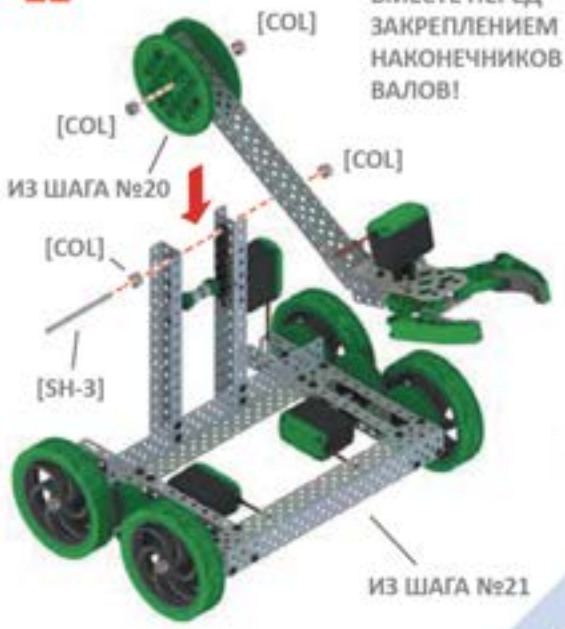


ИЗ ШАГА №17

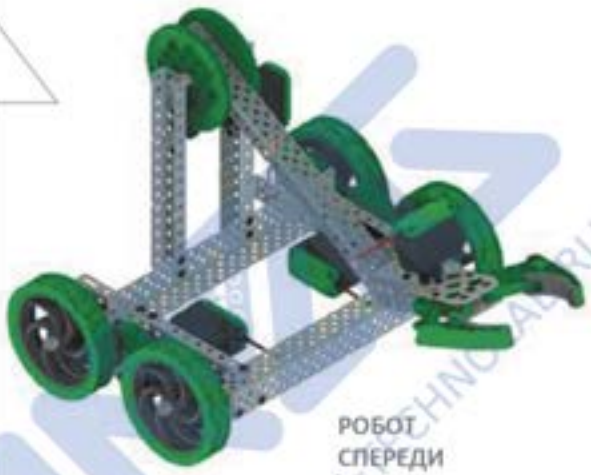
ПОЛНОСТЬЮ ВСТАВЬТЕ ВАЛ В МОТОР, ПЕРЕД ТЕМ КАК ЗАКРЕПЛЯТЬ НАКОНЕЧНИК ВАЛА

РОБОТ СПЕРЕДИ

22



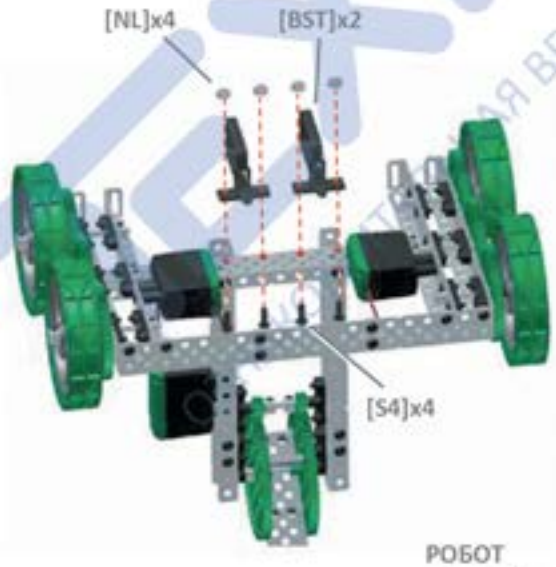
СОЖМИТЕ ВМЕСТЕ ПЕРЕД ЗАКРЕПЛЕНИЕМ НАКОНЕЧНИКОВ ВАЛОВ!



РОБОТ СПЕРЕДИ

• ПРОВЕРЬТЕ, ЧТО «РУКА» ПОДНИМАЕТСЯ И ОПУСКАЕТСЯ ПЛАВНО, С НЕБОЛЬШИМ СОПРОТИВЛЕНИЕМ ОТ МОТОРА

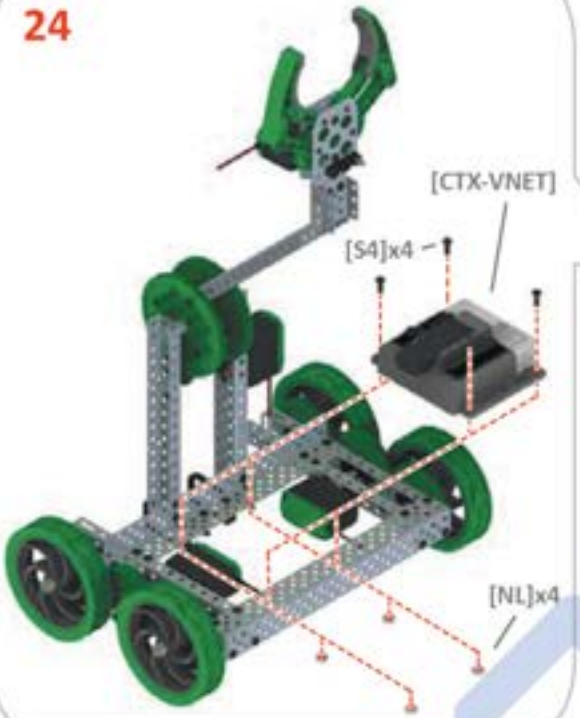
23

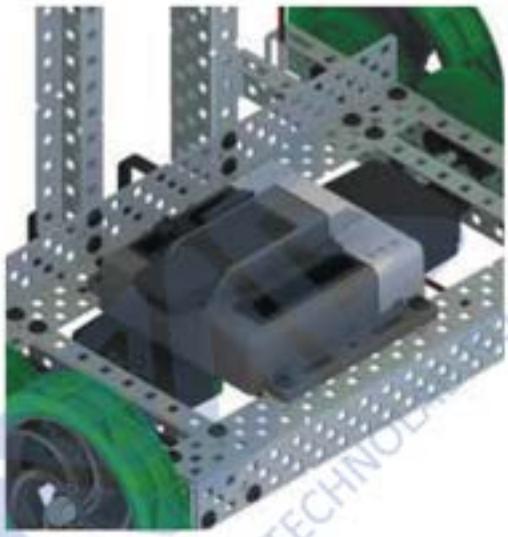


РОБОТ СПЕРЕДИ

УСТАНОВИТЕ КРЕПЛЕНИЕ БАТАРЕИ ПОД РАМОЙ

24

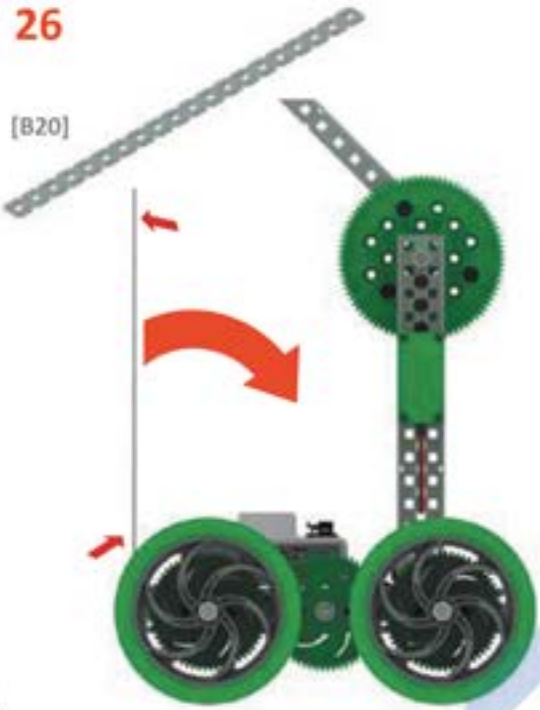




25

26

[B20]

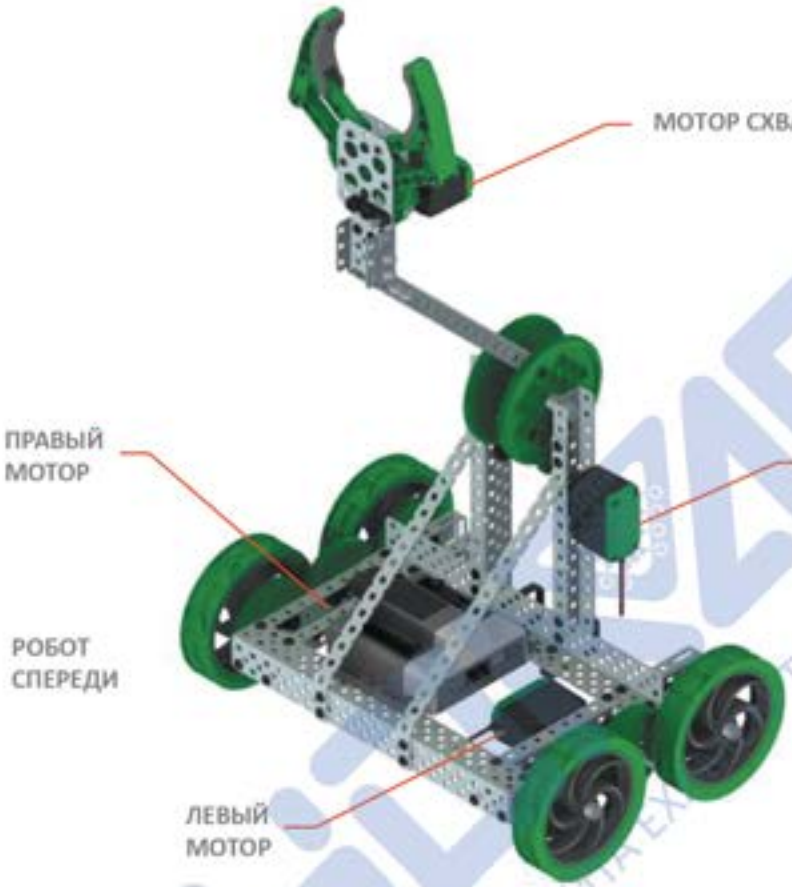


27

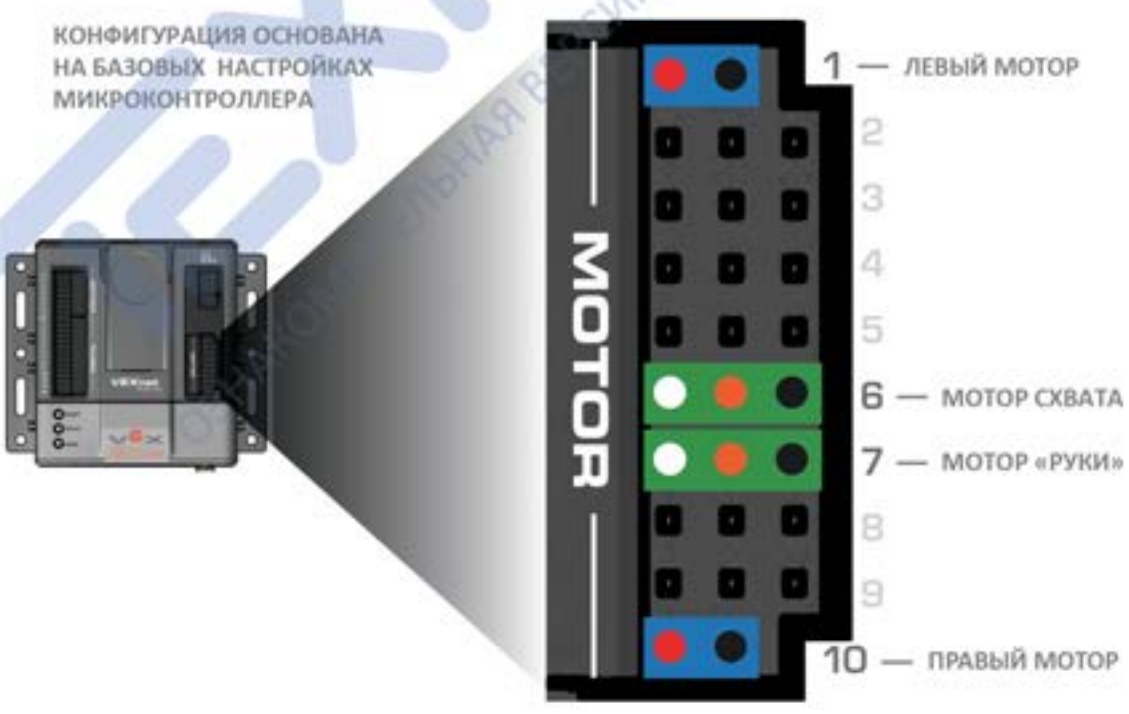
[S2]x4

[NL]x4

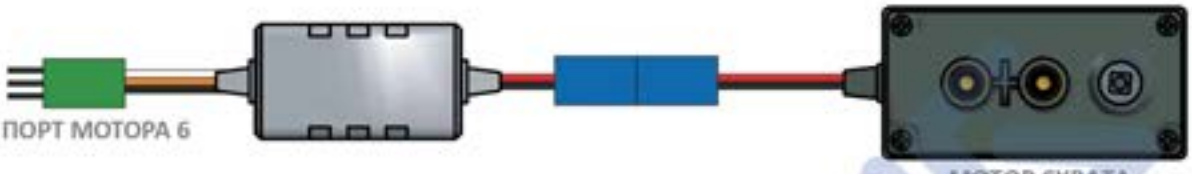
РОБОТ
СПЕРЕДИ



КОНФИГУРАЦИЯ ОСНОВАНА НА БАЗОВЫХ НАСТРОЙКАХ МИКРОКОНТРОЛЛЕРА



красный провод должен быть подключен ближе к внутренней стороне микроконтроллера (см. рисунок на предыдущей странице)

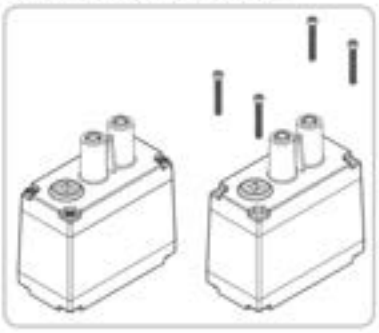


красный провод должен быть подключен ближе к внутренней стороне микроконтроллера (см. рисунок на предыдущей странице)

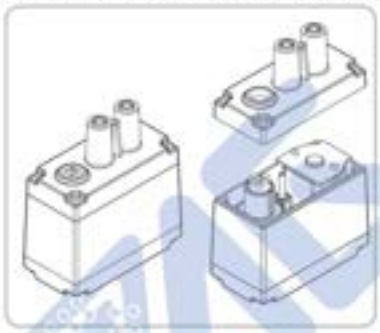
Более подробную информацию о микроконтроллере и системе управления VEXNet можно найти на сайте www.VEXRobotics.com/cortex (на английском языке)

ВЫСОКАЯ СКОРОСТЬ/НИЗКИЙ КРУТЯЩИЙ МОМЕНТ для моторов (необязательная опция):
Для модификации двухпроводного мотора **393** в режим повышенной скорости – просто замените шестерню мотора на одну из прилагаемых шестерней для замены, следуя инструкции:

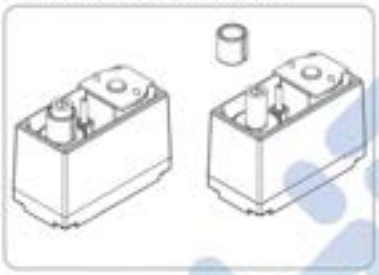
1. Открутите и удалите винты с передней части мотора.



2. Снимите крышку с передней части мотора. Не трогайте шестерни внутри.

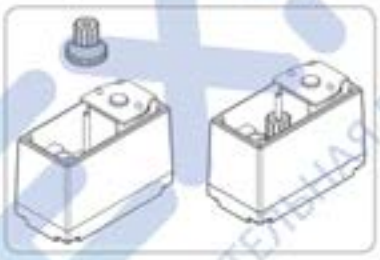


3. Снимите втулку, как показано на рисунке, и отложите в сторону. Она понадобится позже.

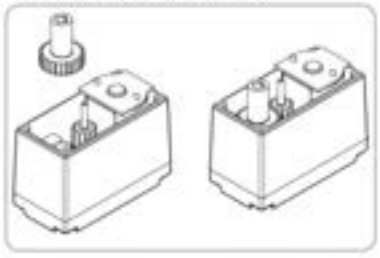


4. Снимите центральную шестерню и шестерню вала.

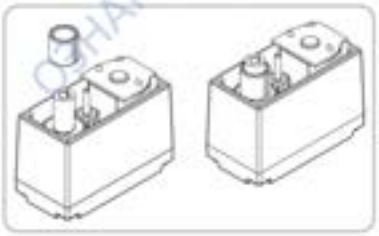
5. Установите центральную шестерню повышенной скорости.



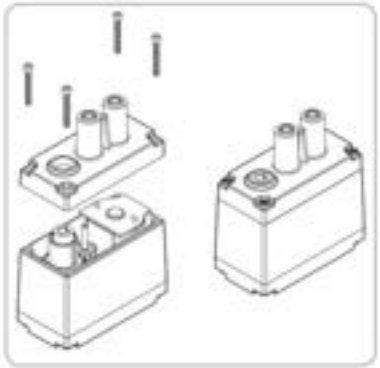
6. Установите шестерню вала повышенной скорости.



7. Установите втулку, снятую в шаге №3. Убедитесь, что она установлена, как на рисунке.



8. Установите обратно крышку и винты, снятые в шагах №1 и №2.



Режим повышенной скорости обеспечивает ускорение вращения вала на 60% и снижение крутящего момента на 60%.

Для заметок

Робот. Функциональная схема робота

Рисунок 1. Функциональная схема робота

Функционирование робота устроено так, как это показано на Рисунке 1.
Как видно из этого рисунка, робот состоит из трех основных частей:
1. Информационно-измерительная система (ИИС)
2. Информационно-управляющая система (ИУС)
3. Исполнительная система (ИС)

Как видно из Рисунка 1, функционирование робота построено циклическим образом. То есть *ИИС* считывает данные из *Внешней среды* и передает их в *ИУС*. Там, в свою очередь, эти данные обрабатываются и принимается решение об управляющем воздействии на *Внешнюю среду*. Управляющее воздействие осуществляется *ИС*, и оно заключается в изменении внешней среды. За этим изменением следит, в свою очередь, *ИИС*. Именно за счет такого построения и достигается «самостоятельное» автономное поведение робота.

Обратите внимание, что на Рисунке 1 в *ИУС* направлена еще одна стрелка из *Системы связи*. С точки зрения робота оператор – это *Внешняя среда*, а *Система связи* – ИИС.

По принципу, показанному на Рисунке 1, функционируют и устройства, называемые *Автоматы*. Дискуссия о разнице между роботами и автоматами не является предметом описания данного пособия, но может стать темой для интересных рассуждений обучающихся.

Рассмотрим робота, который функционирует по схеме на Рисунке 1.

Это Сигвей – электрическое самобалансирующееся транспортное средство с двумя колесами, расположенными по обе стороны от водителя. В качестве *ИИС* в данном устройстве используется гироскопический датчик (не путать с гироскопом!), вернее несколько гироскопических датчиков, *ИС* – это двигатели. *Внешний мир* для этого робота – это наклон робота. *ИУС* формирует такие управляющие воздействия, чтобы робот не находился в состоянии падения. То есть при заваливании вперед колеса вращаются таким образом, чтобы робот начал заваливаться назад, при этом робот движется вперед. И наоборот, при заваливании назад колеса стремятся опрокинуть робота вперед, робот начинает двигаться вперед.

С решения похожих теоретических задач имеет смысл начинать любое занятие по робототехнике. Вот несколько из них:

- Климат-контроль
- Круиз-контроль
- Квадрокоптер
- Автофокусировка фотоаппарата

Конструкция робота для решения задач автоматического управления

В данном пособии будет в основном использоваться переработанная конструкция *Clawbot*. Исходная конструкция предназначена в основном для решения задач управления роботом человеком. Однако для автономной работы, в которой все движения контролируются *ИУС VEX-EDR*, не подойдет *ИС* с полным приводом из-за невозможности выполнить точные движения.

Учебный робот сконструирован таким образом, чтобы иметь как можно более широкую *базу* (расстояние между передними и задними колесами) и *колею* (расстояние между ведущими колесами). Это необходимо для повышения устойчивости и точности движений. Центр масс необходимо располагать между передними и задними колесами, существенно смещая его к ведущим колесам, что также обеспечит устойчивость всей конструкции, а также увеличит трение между ведущими колесами и поверхностью. Робот сконструирован заднеприводным. Для уменьшения трения при осуществлении поворотов и разворотов в качестве передних колес используются *omni-колеса VEX*.

В случае если у Вас нет этих колес, воспользуйтесь классическими мебельными волокушами.

Возможно обмотать скользким скотчем и колеса *VEX*.

На Рисунке 2 и Рисунке 3 показан робот, отвечающий всем изложенным выше условиям.

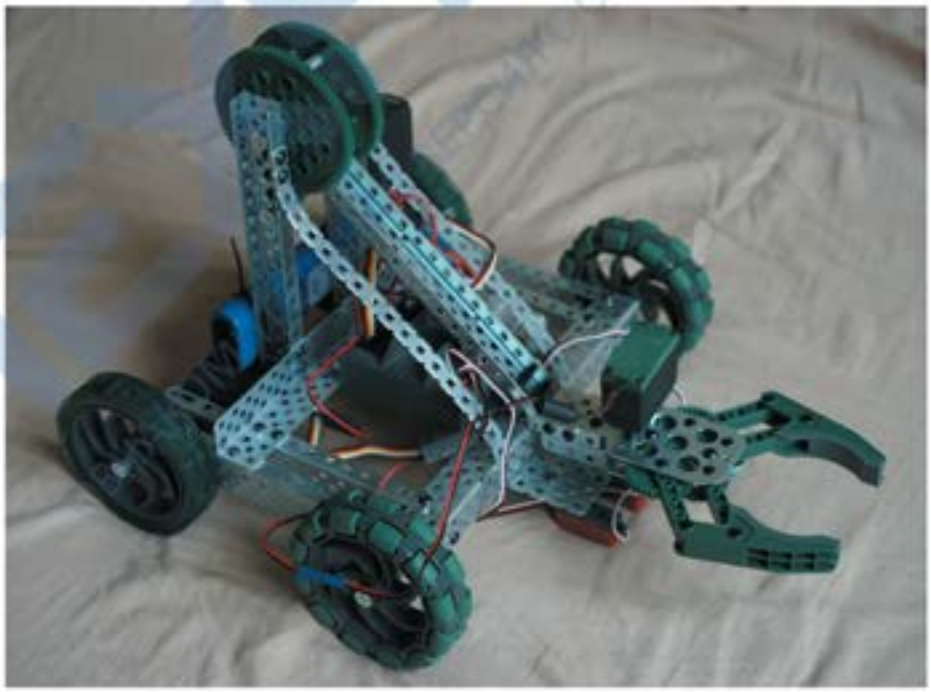


Рисунок 2. Робот для решения задач автоматического управления

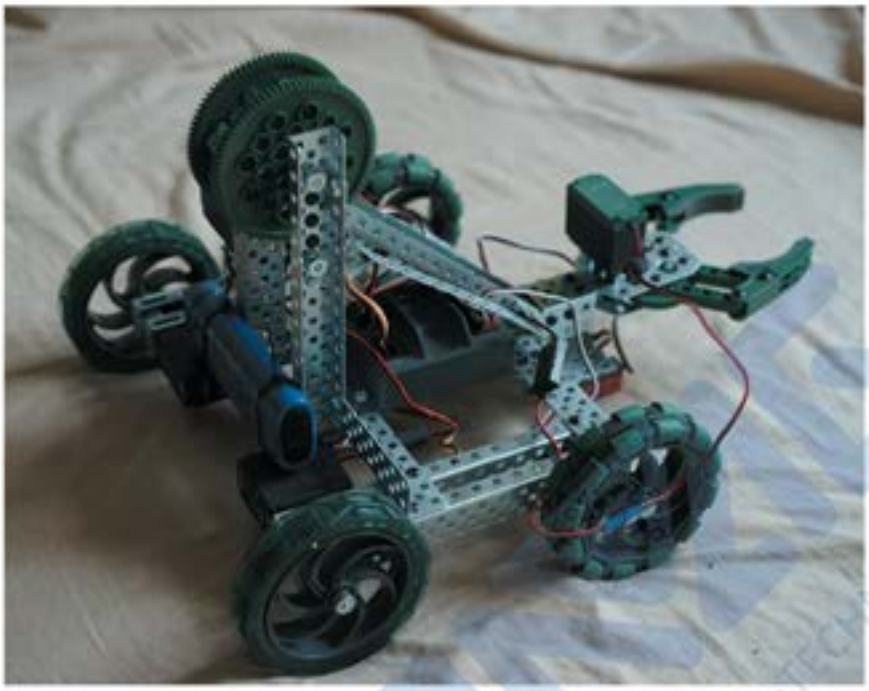


Рисунок 3. Робот для решения задач автоматического управления

Существенным достоинством конструктора *VEX-EDR* является то, что он состоит в основном из металлических деталей, надежных зубчатых колес и достаточно мощных двигателей. *Все это позволяет, даже имея базовый набор, существенно расширить возможности для творчества*, используя фанеру, шурупы и другие подручные средства.

Вы вольны сами выбирать, к каким портам подключать двигатели и датчики. Однако для того чтобы все коды из пособия работали корректно без внесения изменения в программу, необходимо осуществить подключения следующим образом:

Двигатели:

Левый двигатель – *port 1*, красный провод ближе к надписи *motor*, чем черный, к левому двигателю подключен *1 энкодер* через шину *I2C*.

Правый двигатель – *port 10*, черный провод ближе к надписи *motor*, чем красный, к правому двигателю подключен *2 энкодер* через шину *I2C*.

Подъемное устройство манипулятора подключается через драйвер двигателя в *port 7*. Красный и черные провода на драйвере и двигателе находятся в идентичном состоянии. К двигателю подключен *3 энкодер*.

Манипулятор подключен через драйвер двигателя в *port 6*. Красный и черные провода на драйвере и двигателе находятся в идентичном состоянии.

Подробнее о подключении к шине *I2C* рассказано в параграфе *Движения с контролем оборота двигателей*.

Датчики:

Левый *Line Tracker* подключен к первому аналоговому порту.

Правый *Line Tracker* подключен ко второму аналоговому порту.

Датчик *Ultrasonic Range Finder* подключается сразу к двум цифровым портам – пер-

вому и второму.

Датчики *Bamper Switch* – к цифровым портам три и четыре.

Ниже приведена еще одна конструкция, к которой мы обратимся в конце учебного пособия, – это всенаправленный робот на omni-колесах. В этой конструкции необходимо одинаково «нагрузить» все колеса.

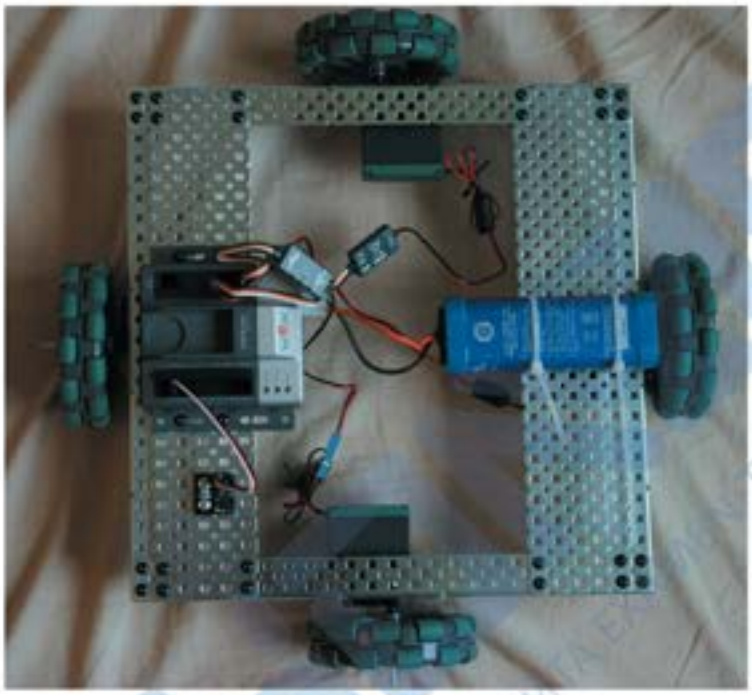


Рисунок 4. Робот на omni-колесах. Вид сверху.

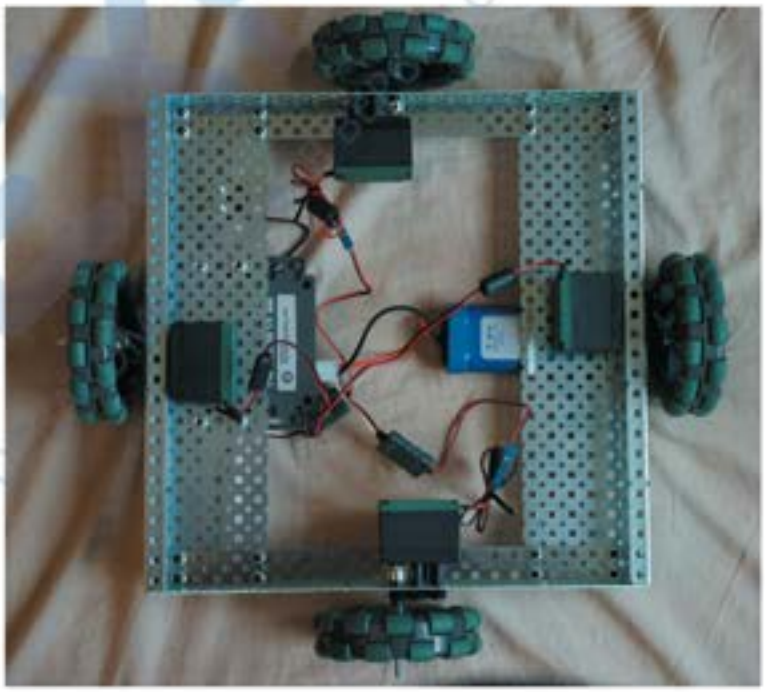


Рисунок 5. Робот на omni-колесах. Вид снизу.

Первоначальные сведения о программировании в языке С

Средством для программирования VEX-роботов является язык программирования RobotC, который, в свою очередь, является модификацией классического языка программирования *С*. Данный раздел содержит необходимые сведения об этом языке.

Основой языка *С* являются *функции*, и любая программа, созданная на этом языке программирования, является *функцией*. Иными словами, все программы *С* состоят из одной или нескольких *функций*, которые могут выполняться последовательно, параллельно, одна функция может включать в себя множество других функций.

Переменные. Типы переменных.

Назначение функции – осуществить необходимые преобразования с переменными. Под переменной будем понимать некоторое имя, с которым связано какое-либо значение. Все возможные значения переменных приведены к нескольким типам. В языке программирования *С* для использования переменной необходимо указать ее тип.

Пример 1:

```
int i=7;        /* Объявлена переменная типа int. Этой переменной присвоено значение 7 */
```

Можно выделить следующие типы переменных:

- int (целый);
- char (символьный);
- wchar_t (расширенный символьный);
- bool (логический);
- float (вещественный);
- double (вещественный с двойной точностью).

Более подробно о типах данных Вы можете узнать из пособий по программированию на языке С, а также в сети Интернет. В данном пособии будут использоваться в основном переменные типа *int*, *float* и *bool*.

Массивы

В языке программирования *С* осуществлена возможность хранить однотипные переменные в *массивах*. Любому элементу *массива* соответствует *индекс* (набор *индексов*), который однозначно указывает на этот элемент. При объявлении массива необходимо указать тип данных элементов, из которых он состоит.

Пример 2:

```
int arr [10];          /* Объявлен массив arr, в котором возможно хранить 10 элементов типа int */

arr[0]=5;              /* Элементу массива с номером 0 присвоено значение 5 */

arr[9]=6;              /* Элементу массива с номером 9 присвоено значение 6 */

arr[10]=6;             /* Эту операцию выполнить невозможно в связи с тем, что массив arr[10] состоит из 10 переменных и их нумерация начинается с 0 элемента и заканчивается 9 */

int a=8;
arr [a]=4;             /* Элементу массива с индексом a, который равен 8, присвоено значение 4 */
```

Функции

Для описания функции в языке С необходимо описать пять элементов:

Таблица 1. Описание функций в языке С

	Название	Обязательность	Назначение
1	Тип функции	Обязательно	Тип данных, возвращаемых функцией. Аналогичен типу данных переменных.
2	Название функции	Обязательно	Используя название функции, можно вызвать ее из любого места в программе.
3	Аргументы функции	Необязательно	Переменные, которые преобразуются во время выполнения функции. Если указываются аргументы, то необходимо указать их тип.
4	Тело функции	Необязательно	Преобразования, которые необходимо осуществить с аргументами функции.
5	Возвращаемые значения	Необязательно	Параметры, которые получаются в результате выполнения тела функции.

В следующем примере описана функция, которая, получив два целочисленных значения *a* и *b*, перемножает их и возвращает результат этого действия.

Пример 3:

```
int function (int a, int b)        /* Объявлена функция с именем function и типом возвращаемых данных int. В круглых скобках объявлены аргументы функции int a и int b */

{                                  /* Открывающаяся фигурная скобка указывает на начало описания тела функции */

return a*b;                        /* В теле функции указано – перемножить аргументы a и b. Оператор return указывает на возвращаемое значение */

}                                  /* Закрывающаяся фигурная скобка указывает на конец описания тела функции */
```

Обратите внимание на символы /* и */ в Примере 1, Примере 2 и Примере 3. Эти символы необходимы для оставления комментариев в программе. При выполнении программы (точнее ее компиляции) все, что попадает между /* и */, будет проигнорировано, что позволяет оставлять автору программы комментарии для упрощения работы с кодом. В случае если комментарий занимает менее одной строчки, удобнее пользоваться //, в этом случае игнорироваться будет весь текст, находящийся справа от // и до конца строки.

ВАЖНО! В Примере 3 в конце оператора *return a*b*; поставлена *точка с запятой*. В языке программирования *C* этот знак препинания называют разделителем команд. Обычно в одной строке кода находится одна команда, поэтому в конце строки ставят *точку с запятой*. Но допустима и запись программы в одну строчку с обязательным разделением команд *точкой с запятой*.

Преобразование переменных в языке *C* осуществляется последовательностью *команд (операторов)*. В дальнейшем для программирования роботов нам понадобятся следующие *операторы*.

Арифметические операторы

1. Присваивание

Пример 4:

```
b=7;                               /* Переменной b присвоено значение 7 */

a=b;                               /* Переменной a присвоено значение переменной b, в результате a=7 */
```

2. Сложение, вычитание, умножение, деление

Пример 5:

```
a=1; b=2; c=4;        /* Переменным a,b,c присвоены значения 1,2,4 соответственно */

d=a+b*c;              /* Переменной d присвоено значение 9 после выполнения операций сначала умножения b*c, а затем сложения с полученным числом с переменной a */

c=c/b;                /* Переменной c присвоено значение 2 после выполнения действия c разделить на b */

a=a+1;                /* Переменной a присвоено значение 2. То есть к прошлому значению a прибавлено 1. Результат этой операции записан в переменную a */

b++;                  /* Этот оператор эквивалентен действию двух операторов – оператора присваивания и оператора сложения. Выражения b=b+1 и b++ идентичны. В результате операции переменной b присвоено значение 3 */
```

ВАЖНО! Обратите внимание на последовательность действий в Примере 5. Сначала выполняется действие оператора, находящегося в скобках, затем операции умножения или деления и в самую последнюю очередь операции сложения или вычитания.

Логические операторы и операторы сравнения

Результатом действия логических операторов и операторов сравнения всегда является либо «*Истина*», что соответствует логической «*1*», либо «*Ложь*», что соответствует логическому «*0*» (если переменная принимает значения 0 и 1, то она соответствует типу данных *bool*). Вообще логические операторы или операторы сравнения можно представить в виде вопроса с двумя вариантами ответа. Операторы сравнения действуют на любые переменные, логические операторы же применяются только для переменных типа bool.

Операторы сравнения

Пример 6:

```
7==5;        /* Осуществляется сравнение чисел 5 и 7, результат этой операции «Ложь» или логический «0». Это действие эквивалентно вопросу: Семь равно пяти? */

7>5;         /* Осуществляется сравнение чисел 5 и 7, результат этой операции «Истина» или логическая «1». Это действие эквивалентно вопросу: Семь больше пяти? */

7<=5;        /* Осуществляется сравнение чисел 5 и 7, результат этой операции «Ложь» или логический «0». Это действие эквивалентно вопросу: Семь меньше или равно пяти? */
```

ВАЖНО! Сравнение переменных осуществляется оператором «==», обратите внимание на то, что «a=b» – это операция присваивания и в результате ее выполнения произойдет не сравнение переменных *a* и *b*, а переменной *a* будет присвоено значение переменной *b*.

Логические операторы

Пример 7:

```
a=1; b=0;    /* Переменным a и b присвоены значения 1 и 0 соответственно */

!a;          /* Осуществляется логическое отрицание a, или, другими словами, НЕ a, результат этой операции «Ложь» или логический «0» */

!b;          /* Осуществляется логическое отрицание b, или, другими словами НЕ b, результат этой операции «Истина» или логическая «1» */

a&&b         /* Осуществляется логическое умножение (конъюнкция) a и b, результат этой операции «Ложь» или логический «0» */

a||b         /* Осуществляется логическое сложение (дизъюнкция) a или b, результат этой операции «Истина» или логическая «1» */
```

ВАЖНО! Среди всех логических операторов наибольший приоритет имеет «логическое отрицание» или «!». То есть в любом сложном выражении первым будет выполнен именно этот оператор. Следующим по приоритету является «логическое И» или «&&». Самым низким по приоритету является оператор «Логическое ИЛИ» или «||».

Действие логических операторов показано в Таблице 2.

Таблица 2. Логические операторы

a	b	!a Логическое отрицание	a&&b Логическое умножение	a||b Логическая сумма
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Совместное использование логических операторов и операторов сравнения

Пример 8:

```
a=8; b=4;                  /* Переменным a и b присвоены значения 8 и
                           4 соответственно */

                           /* Первыми осуществляются действия в круглых скобках. Эти действия эквивалентны вопросам: 8 равно 4? и 8 больше или равно 4? Ответ на первый вопрос – Ложь, на второй – Истина. Затем осуществляется логическое умножение ответов Ложь&&Истина, результатом этого действия будет Ложь. Таким образом, результат всего этого выражения Ложь */
(a==b)&&(a>=b);
```

```
!(a==b)&& (a>=b);
```

/* Первыми осуществляются действия в круглых скобках. Эти действия эквивалентны вопросам: 8 равно 4? и 8 больше или равно 4? Ответ на первый вопрос – Ложь, на второй – Истина. Затем осуществляется логическое отрицание первого выражения – НЕ Ложь это Истина. Затем осуществляется логическое умножение ответов Истина&&Истина, результатом этого действия будет Истина. Таким образом, результат всего этого выражения Истина */

Реализация циклов и ветвлений в С

В языке программирования *С* выполнение программы осуществляется не только линейно. Достаточно часто возникает необходимость выполнять или не выполнять какую-либо группу команд, или выполнять однотипную группу команд множество раз. Такие алгоритмы получили название алгоритмы с *ветвлениями* и *циклические алгоритмы*. В языке *С* они описываются как функции, то есть для реализации необходимо описать все поля из Таблицы 1. Одним из аргументов для операторов ветвления и цикла обязательно должна быть переменная типа *bool*, чаще всего это результат действия операции сравнения или логической операции.

Операторы ветвления

Пример 9:

```
a=8; b=4;

if (a==2*b)

    {
    a=b;
    }
```

/* Переменным a и b присвоены значения 8 и 4 соответственно */

/* Объявляется использование условного оператора. Выражение в круглых скобках по сути является аргументом функции if. Эта строка кода эквивалентна вопросу: Равно ли a, два умножить на b? В данном случае ответ «Истина». Следовательно, будут выполнены группы действий, которые находятся в фигурных скобках, следующих сразу за закрывающейся круглой скобкой. Обратите внимание на то, что после «)» знак «;» не ставится */

/* В результате действия оператора ветвления a=4, b=4 */

```
                                    /* Результатом выражения в скобках является
if (a!=b)                           «Ложь», следовательно, последующая груп-
                                    па действий, находящихся между фигурными
                                    скобками, выполнена не будет */
    {
    a=15;                           /* В результате действия оператора ветвле-
    }                               ния a=4, b=4 */

if (a>b)                            /* В случае если не выполнено условие в кру-
    {                               глых скобках следующих за оператором if, вы-
    a=3;                            полняются действия в фигурных скобках, сле-
    }                               дующих за оператором else */
else
    {                               /* В результате действия оператора ветвле-
    a=5;                            ния a=5, b=4 */
    }
```

Операторы цикла

В *C* используется несколько видов различных циклических операторов. Каждый из них удобнее использовать в той или иной ситуации. Однако возможно всегда использовать только один из этих операторов, и в данном пособии при программировании роботов будет использоваться оператор *while*. Функционирование этого оператора основано на том, что каждый раз, когда его аргумент равен *логической 1* или «*Истине*», последовательно выполняются операторы в теле цикла от открывающейся фигурной скобки «{» до закрывающейся фигурной скобки «}».

Пример 10:

```
                                    /* Аргументом оператора while в данном
while (1)                           случае является логическая 1 или «Истина»,
                                    следовательно, тело цикла будет выполнять-
                                    ся бесконечно. Обратите внимание на то, что
                                    после «)» нет «;» */
    {                               /* Начало цикла */

                                    /* Это действие будет выполняться бесконеч-
    a=5;                            но в связи с тем, что аргумент цикла всегда
                                    логическая 1 */

                                    /* Конец цикла. После этой фигурной скобки
    }                               снова будет осуществлена проверка аргумен-
                                    та оператора while */
```

```
a=1; b=4;
while (a<5)

    {
    a++;
    }
```

/* Внутри круглых скобок осуществляется проверка, которая эквивалентна вопросу: Меньше ли а 5? Если ответ на этот вопрос «Истина», то выполняется тело цикла */

/* Данный цикл будет выполнен 4 раза. При первом выполнении переменная a=0, следовательно, выполняется тело цикла, в ходе чего переменной а присваивается значение 2, при втором выполнении значение переменной а по-прежнему меньше 5, следовательно, значение а будет увеличено на 1. При 4-м выполнении переменная а будет равна 5. И при ответе на вопрос, меньше ли а пяти, получается ответ – «Ложь», следовательно, не выполняется и тело цикла */

Для заметок

Особенности программирования в RobotC

Функции в RobotC

Как уже было сказано выше, программы в *C* состоят из функций и сами являются функциями. В RobotC функции могут выполняться последовательно и параллельно. Обязательное требование для любой программы — это наличие в ней функции *main*. Она объявляется при помощи ключевого слова *task*, что переводится на русский язык как «задача». Так как это основная функция программы, у нее не может быть никаких аргументов, а есть только тело.

Пример 11:

```
task main ()
{                                   /* С этой скобки начнется выполнение программы */

}                                   /* На этой скобке выполнение программы закончится */
```

Очень часто возникает необходимость повторять некоторые функции, изменяя только их параметры. Например, в первом случае роботу необходимо передвигаться по прямой до препятствия, находящегося на расстоянии 25 см от робота, а потом остановиться, в другом же случае остановиться надо на расстоянии 10 см от препятствия. И в том и в другом случае удобно воспользоваться одной и той же функцией с разными аргументами. Для описания такой функции используется ключевое слово *void*. Эту функцию можно вызвать из любого места функции *main*. Удобнее всего первым шагом продумать все функции, которые понадобятся для программирования робота, а затем в начале кода программы описать все эти функции.

Пример 12:

```
void one (int a, int b)


    {                          /* С этой скобки начнется выполнение функ-
                                  ции one */

                               /* На этой скобке выполнение программы за-
    }                             кончится */
void two (int c, int d)
    {
    }

task main ()
    {
    one (5, 7);                /* Вызов функции one с параметрами 5 и 7 */
    two (15, 18);
    }
```

ВАЖНО! Обратите внимание в Примере 12 на аргументы функции *a* и *b*. Они объявлены только внутри функции *one*. Как только закончится выполнение всех команд функции, будут уничтожены и эти переменные.

Параллельные задачи в RobotC

Важной особенностью RobotC является возможность параллельного выполнения задач.

Пример 13:

```
task one ()                    /* Задача, которая будет выполняться парал-
                                  лельно другим задачам */
    {
    }

task two ()
    {
    }

task main ()
    {

    StartTask(one);            /* Запуск задачи one, которая будет выпол-
                                  няться параллельно с остальными задачами*/
```

```
StartTask(two);
StopTask (two);                 /* Прекратить выполнение задачи two */
StopAllTasks();                 /* Прекратить выполнение всех задач */
}
```

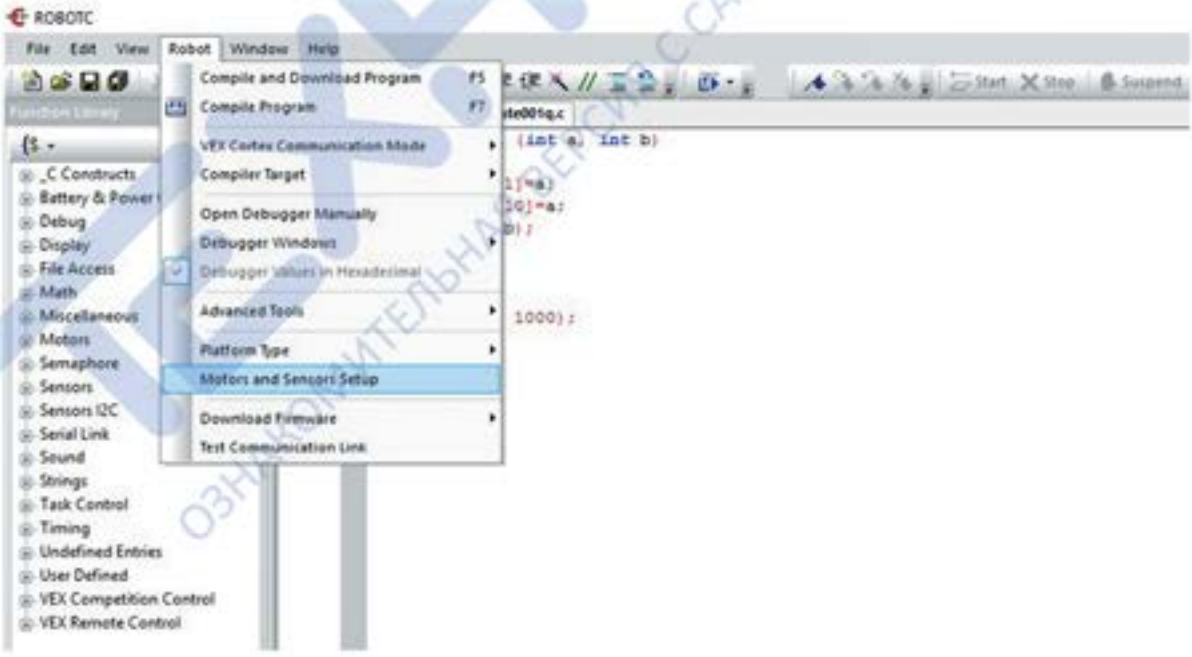
Время работы процессора распределяется между задачами таким образом, что каждой достается небольшой отрезок времени, за который она выполняет несколько инструкций. Поскольку задачам присваивается приоритет, первыми выполняются те, чей приоритет более высокий. Задачи с низшим приоритетом находятся в ожидании, пока в очереди присутствуют задачи с высшим приоритетом.

Команда *hogCPU()* передает все процессорное время текущей задаче до тех пор, пока не встретится команда *releaseCPU()*. Таким образом, на время можно полностью остановить остальные задачи.

Конфигурирование RobotC

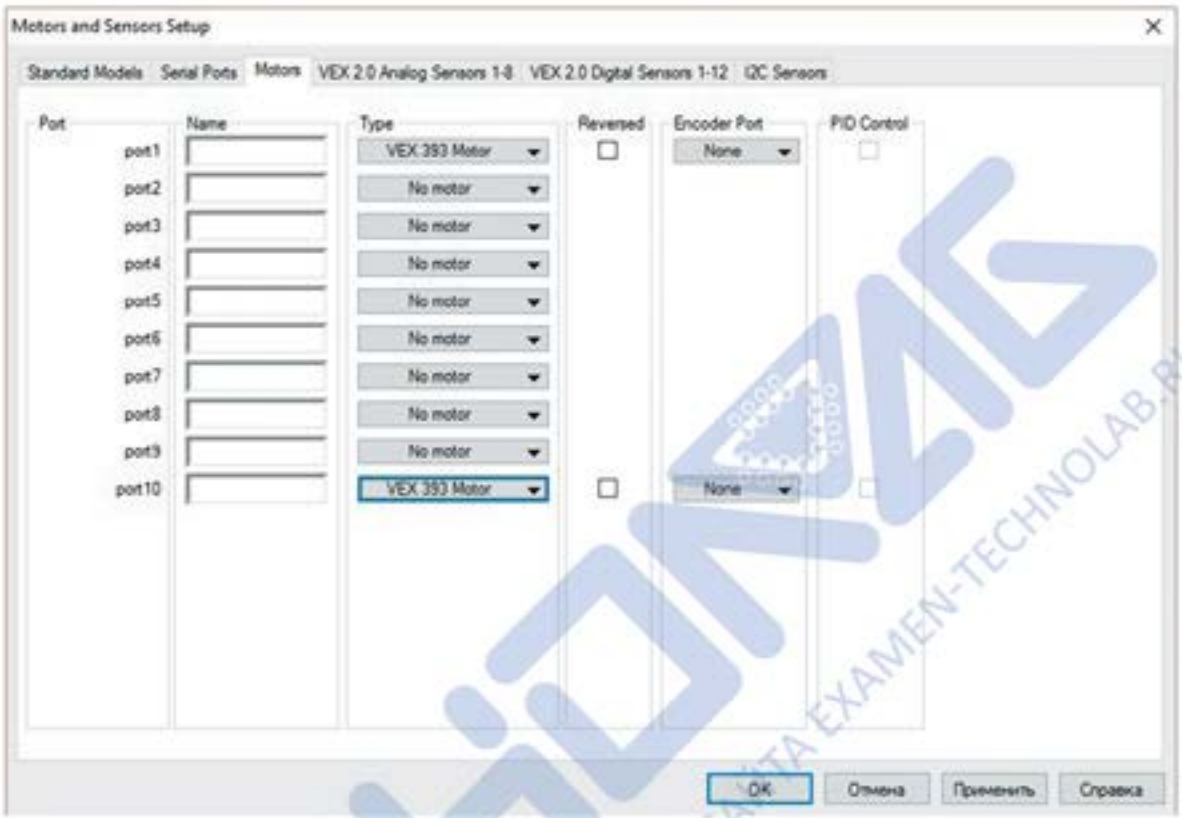
Важным отличием *RobotC* от классического языка программирования *C* является возможность считывать показания датчиков и осуществлять управление двигателями.

Для инициализации двигателей и датчиков необходимо выбрать пункт меню: *Robot→Motors and Sensors Setup*



В открывшемся окне необходимо указать то, каким образом к роботу подключены датчики и двигатели. Во вкладке *Motors* указываются используемые двигатели. Как

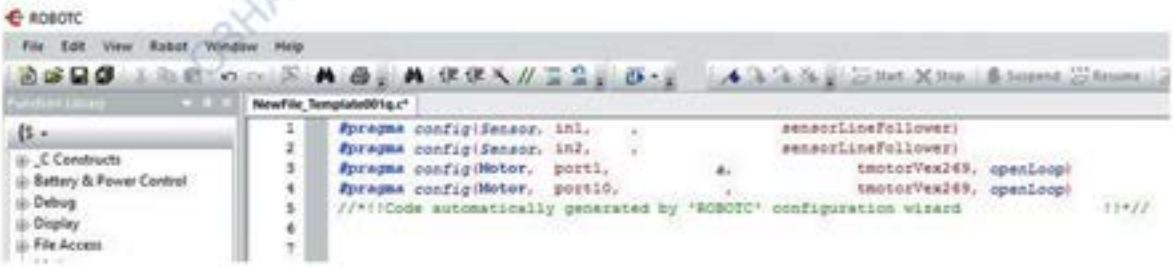
правило, для системы передвижения используются моторы, подключенные к портам *port 1* – левый двигатель *port 10* – правый.



ВАЖНО! Двигатели подключаются к *port 1* и *port 10* напрямую к микроконтроллеру VEX, ко всем остальным портам они подключаются через контроллер мотора.

Во вкладках VEX2.0 Analog Sensors 1-8 и VEX2.0 Digital Sensors 1-12 указываются подключенные к роботу датчики.

После того как будут указаны порты для подключения двигателей и датчиков, в программе автоматически сгенерируется несколько строк кода. Благодаря этому при переносе программы на другой компьютер или при подключении другого робота повторять настройку не требуется.



Управление двигателями в RobotC

После этапа конфигурирования можно составить следующую программу.

Пример 14:

```
task main ()
    {

    motor [port1]=127;          /* Включить двигатель на port1 в прямом
                                направлении с максимальной скоростью */

    motor [port10]=127;         /* Включить двигатель на port10 в прямом
                                направлении с максимальной скоростью */

    }
```

В Примере 14 используется уникальное для RobotC выражение *motor[port1]=127*. Это выражение означает, что элементу массива motor[port1] присвоено значение *127*. В дальнейшем из этого элемента массива контроллером VEX будет считано значение и в соответствии с ним будет сформировано управляющее воздействие на двигатель. Таким образом задается скорость двигателя, подключенного к *port 1*. Аналогичным образом можно управлять всеми моторами, изменяя только номер *port* (*port 2*, *port 3* и т.д.).

Широтно-импульсная модуляция

Скоростью моторов микроконтроллер VEX управляет при помощи широтно-импульсной модуляции (ШИМ). Смысл этой технологии заключается в том, что напряжение на двигатель подается равными по продолжительности импульсами, между которыми находятся равные по продолжительности паузы (Рисунок 6, Рисунок 7). Как видно из рисунков, скорость двигателя определяется суммарной шириной импульсов или, другими словами, суммарной площадью синих зон. Очевидно, что при увеличении количества пауз скорость двигателя будет снижаться.

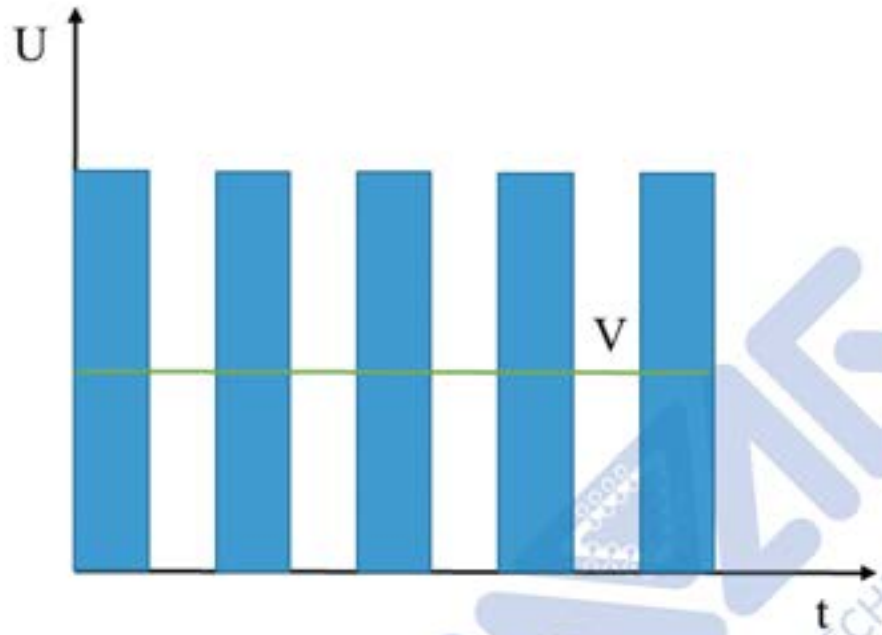


Рисунок 6. Широтно-импульсная модуляция.
Скорость вращения двигателя равна половине от максимальной.

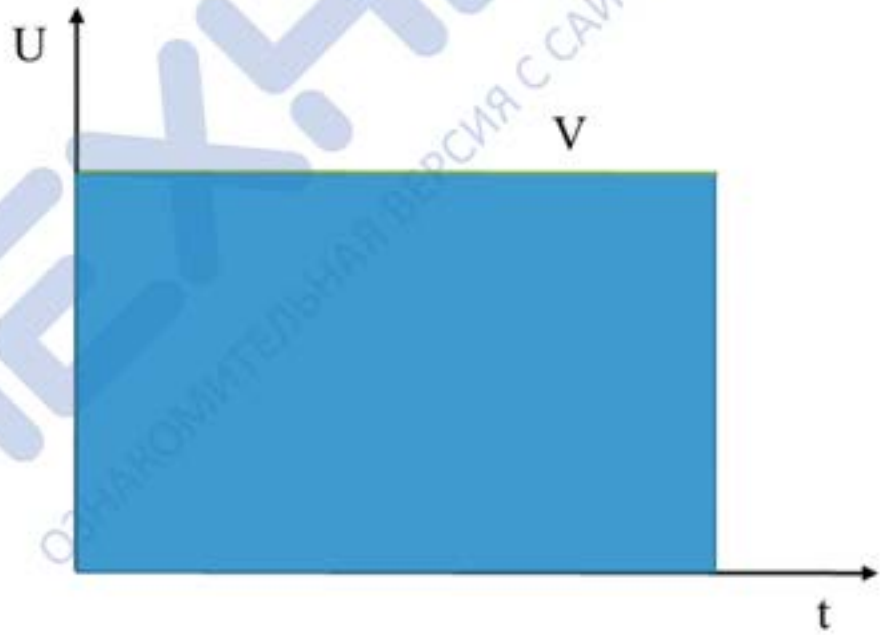


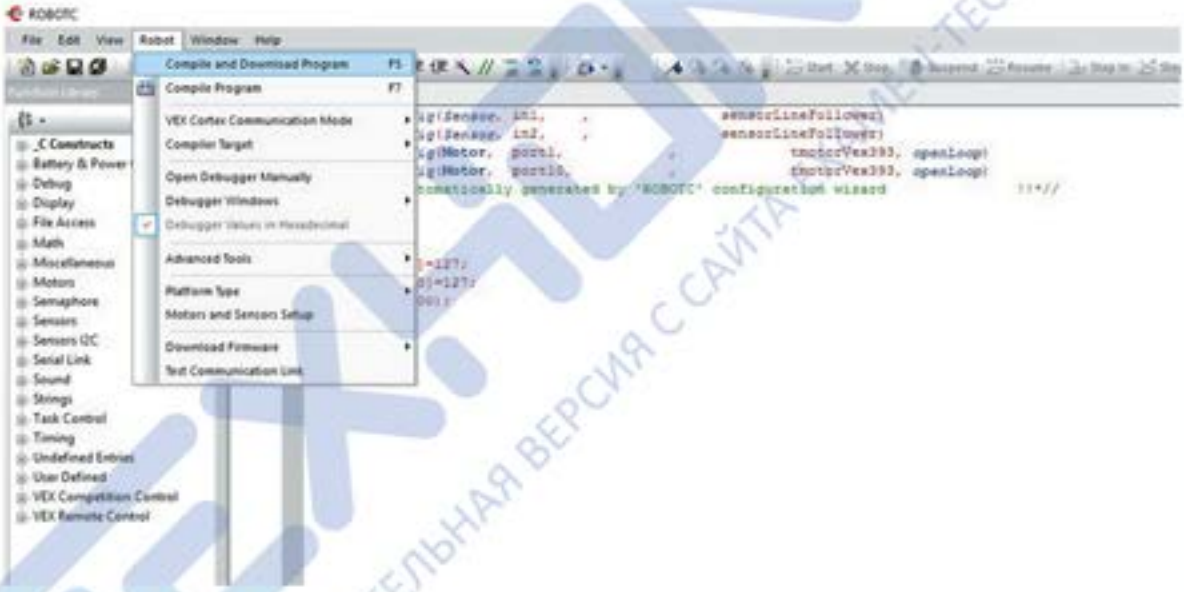
Рисунок 7. Широтно-импульсная модуляция.
Скорость вращения двигателя максимальна.

В выражении *motor[port1]=127* число *127* соответствует тому, что на двигатель будут поданы импульсы без пауз. Таким образом, двигатель начнет вращаться с максимальной скоростью. Выражение *motor[port1]=0* означает, что на двигатель будут подаваться только паузы (на двигатель вообще не будет подано никакое напряжение), то есть двигатель не будет работать. Промежуточные значения означают различные соотношения между паузами и импульсами. Для того чтобы запустить двигатель в обратном направлении с максимальной скоростью, используется выражение *motor[port1]=-127*.

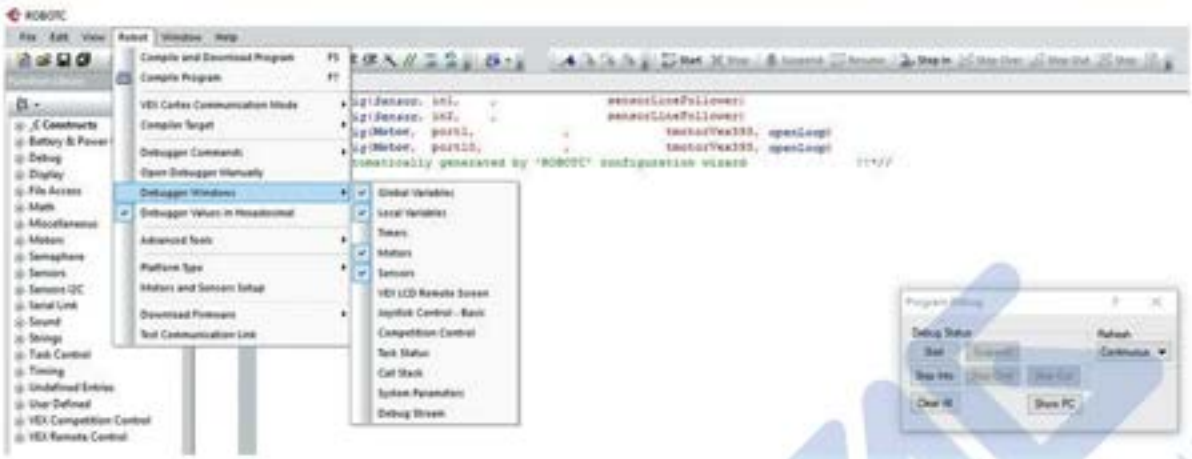
ВАЖНО! В будущем будем называть эти значения мощностью, подаваемой на двигатель, или просто мощностью, что некорректно с точки зрения физики. Было бы корректнее называть эту величину долей от максимальной мощности двигателя.

Компиляция, загрузка и запуск программы

Для запуска программы необходимо перейти в пункт меню *Robot→Compile and Download Program* или воспользоваться клавишей *F5*. После чего автоматически произойдет компиляция кода и его загрузка в микроконтроллер Vex.



На экране отобразится окно *Program Debug*, при этом станут доступны пункты вложенного меню *Debugger Windows*.



ВАЖНО! Рекомендуем активировать пункты меню – *Global Variables*, *Local Variables*, *Motors* и *Sensors*. В этом случае возможно будет наблюдать за показаниями датчиков и изменением переменных во время отладки программы.

Для выполнения команды роботом нажмите кнопку *Start*. После нажатия на эту кнопку не произойдет никаких видимых изменений. Это связано с тем, что промежуток между включением двигателей и окончанием программы очень мал.

Команды ожидания в RobotC. Задержки и таймеры

Изменим код из Примера 14.

Пример 15:

```
task main ()
    {
    motor [port1]=127;
    motor [port10]=127;

    wait1Msec (1000);          /* Команда задержки на 1000 миллисекунд
                                  или одну секунду */

    motor [port1]=0;
    motor [port10]=0;
    }
```

В *RobotC* используется две похожие команды ожидания или команды задержек – *wait1Msec ()* и *wait10Msec()*. В первом случае аргументом команды являются миллисекунды, во втором – проценты секунды. Запустим программу. Робот двигается в течение *1 секунды* вперед, а затем останавливается.

ВАЖНО! Удобно запускать робота, подключенного к компьютеру кабелем USB, так

как это позволяет отслеживать показания датчиков и изменение переменных, однако робот VEX автономен и для того чтобы запустить загруженную в микроконтроллер программу, достаточно включить робота.

Таймеры в RobotC

В языке программирования *RobotC* реализована возможность использования таймеров, то есть отслеживания времени, прошедшего от начала какого-либо события. Одновременно возможно использовать четыре таймера T1, T2, T3 и T4. Данные каждого таймера записываются в три массива, в первом хранится время в миллисекундах – time1[], во втором – время в процентах секунды time10[], в третьем – в децисекундах time100[] (в квадратных скобках указывается используемый таймер).

Пример 16:

```
task main ()
    {

    ClearTimer(T1);                  /* Обнуление и запуск таймера T1 */

                                     /* Цикл while будет выполняться, пока
                                     значение таймера T1, выраженное
                                     в децисекундах, меньше 10, иными
                                     словами, цикл будет выполняться 1
    while (time100[T1]<10)           секунду */

        {
        motor [port1]=127;
        motor [port10]=127;
        }
    motor [port1]=0;
    motor [port10]=0;
    }
```

Использование датчиков в RobotC

ВАЖНО! Сенсоры или датчики конфигурируются в среде *RobotC* так же, как и двигатели. Обратите внимание на то, что для микроконтроллера *VEX* необходимо указывать тип датчика: цифровой (VEX 2.0 Digital Sensors), аналоговый (VEX 2.0 Analog Sensors) и датчики, использующие в своей работе шину I2C.

Данные, полученные с датчиков, микроконтроллер VEX записывает в массивы.

Пример 17:

```
a=SensorValue[in4];          /* Переменной a присвоено значение, записанное микроконтроллером VEX в элемент массива SensorValue с индексом in4. Индексы in соответствуют аналоговым датчикам, а число справа от in номеру аналогового порта */

a=SensorValue[dgtl1]         /* Переменной a присвоено значение, записанное микроконтроллером VEX в элемент массива SensorValue с индексом dgtl1. Индексы dgtl соответствуют цифровым датчикам, а число справа от dgtl номеру аналогового порта */

a=SensorValue[I2C_8]         /* Переменной a присвоено значение, записанное микроконтроллером VEX в элемент массива SensorValue с индексом I2C_8 */
```

С показаниями датчиков, записанными в массивы, возможно осуществлять те же операции, что и с обычными переменными.

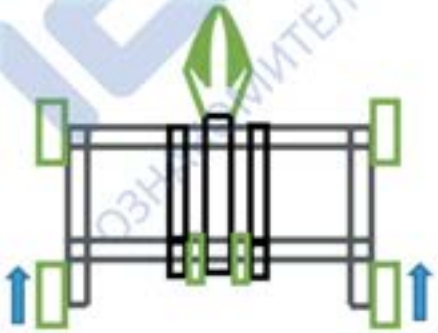
ВАЖНО! Получение данных с датчиков очень длительная для контроллера процедура, поэтому для математических преобразований этих данных лучшим решением будет запись текущего значения датчика в переменную, а затем математические преобразования переменной.

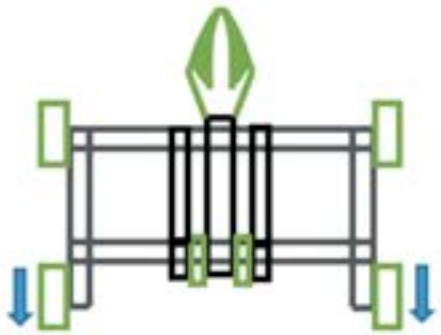
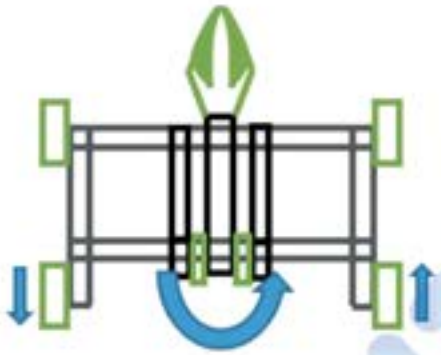
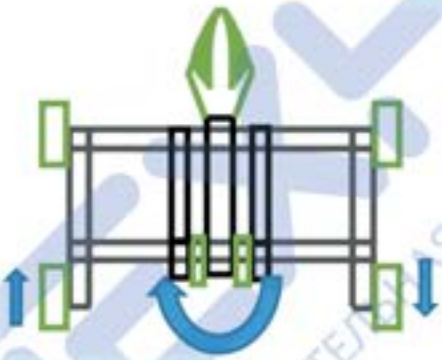
Тайминговый контроль перемещений робота. Простейшие передвижения робота

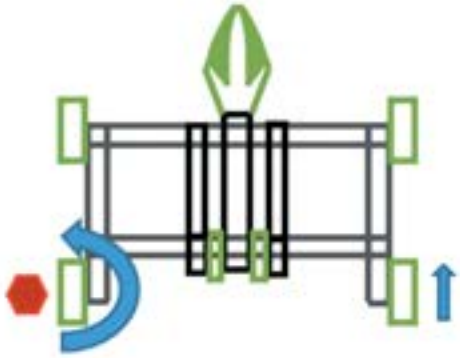
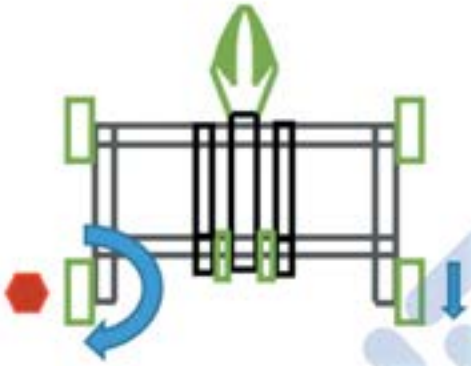
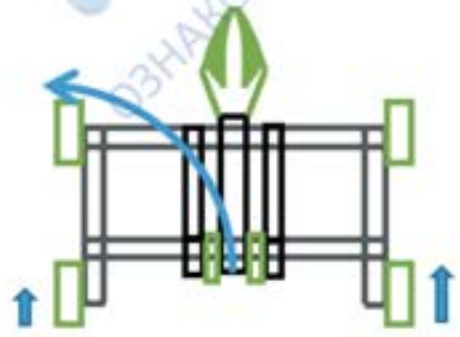
ЗАДАЧА: Создать программный комплекс, реализующий все возможные варианты движения робота с возможностью контролировать направление движения и поворотов, а также длительность этих действий.

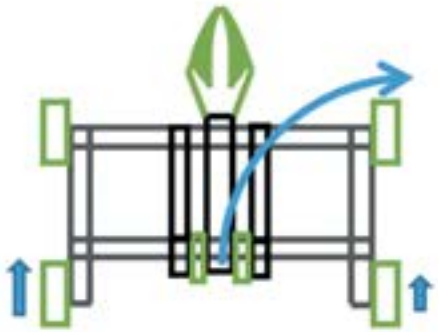
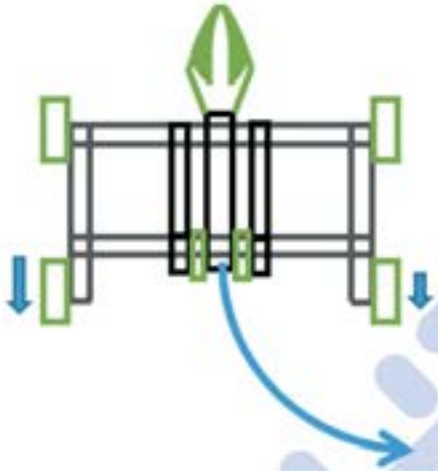
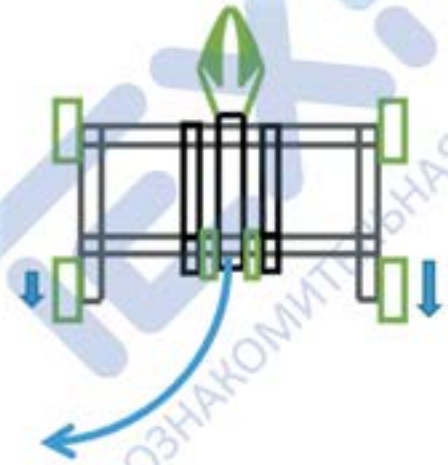
Рассмотрим все возможные варианты перемещения робота в Таблице 3. Ведущие колеса – задние. Схват в состоянии покоя находится в передней части робота. Синие стрелки – это векторы скорости верхних точек колес. Если эти векторы сонаправлены с положением робота, то считаем, что двигатель включен вперед, если противонаправлены – назад.

Таблица 3. Перемещение робота

Рисунок	Действие робота
	Оба двигателя остановлены. Робот не двигается.
	Оба двигателя включены таким образом, чтобы робот перемещался вперед.

	Оба двигателя включены таким образом, чтобы робот перемещался назад.
	Двигатели включены в разные стороны. Робот разворачивается на месте против часовой стрелки. Ось разворота пересекает оси вращения колес и находится на равном расстоянии от них.
	Двигатели включены в разные стороны. Робот разворачивается на месте по часовой стрелке. Ось разворота пересекает оси вращения колес и находится на равном расстоянии от них.
	Левый двигатель включен вперед, правый остановлен. Робот поворачивается вокруг правого колеса по часовой стрелке.

	Правый двигатель включен вперед, левый остановлен. Робот поворачивается вокруг левого колеса против часовой стрелки.
	Правый двигатель включен назад, левый остановлен. Робот поворачивается вокруг левого колеса по часовой стрелке.
	Левый двигатель включен назад, правый остановлен. Робот поворачивается вокруг правого колеса против часовой стрелки.
	Скорость левого двигателя меньше, чем скорость правого двигателя. Оба двигателя включены вперед. Робот движется по дуге вперед-налево.

	Скорость правого двигателя меньше, чем скорость левого двигателя. Оба двигателя включены вперед. Робот движется по дуге вперед-направо.
	Скорость правого двигателя меньше, чем скорость левого двигателя. Оба двигателя включены назад. Робот движется по дуге назад-направо.
	Скорость правого двигателя больше, чем скорость левого двигателя. Оба двигателя включены назад. Робот движется по дуге назад-налево.

Таким образом, соотношения скоростей и направлений движения ведущих колес определяют характер движения робота. Добавим к этому время выполнения того или иного действия и получим возможность двигаться по любым траекториям и попадать в любую точку плоскости. Продемонстрируем это на примере движения по квадрату в Примере 18.

ВАЖНО! Время, которое необходимо роботу для разворота на 90 градусов на месте, необходимо подбирать для каждой конструкции отдельно. Оно зависит от колеи, базы и расположения центра масс робота.

Пример 18:

```
void moving (int VL, int VR,
int time)                          /* Объявлена вспомогательная функция
                                   moving, ее аргументами являются скорости
                                   левого и правого мотора, а также время вы-
                                   полнения движения */
    {
    motor[port1]=VL;               /* Левому мотору присвоено значение
                                   VL */
    motor[port10]=VR;              /* Правому мотору присвоено значение
                                   VR */
    wait1Msec(time);               /* Осуществляется пауза продолжительно-
                                   стью time миллисекунд */
    }

task main()
    {
    moving (127, 127, 500);        /* Вызывается функция moving с параме-
                                   трами: скорость левого двигателя 127, пра-
                                   вого – 127, время движения 500 миллисе-
                                   кунд */
    moving (0, 0, 300);            /* Остановка на 0,3 секунды */

    moving (-127, 127, 500);       /* Осуществляется разворот на месте. 500
                                   миллисекунд соответствует приблизитель-
                                   но развороту на 90 градусов */
    moving (127, 127, 500);
    moving (0, 0, 300);
    moving (-127, 127, 500);
    moving (127, 127, 500);
    moving (0, 0, 300);
    moving (-127, 127, 500);
    moving (127, 127, 500);
    moving (0, 0, 300);
    }
```

Движение по квадрату можно осуществить и более рациональным способом.

Пример 19:

```
void moving (int VL, int VR, int time)
	{
	motor[port1]=VL;
	motor[port10]=VR;
	wait1Msec(time);
	}

task main()
	{
	int i=0;
	while (i<3)
		{
		moving (127,127, 500);
		moving (0,0, 300);
		moving (-127,127, 500);
		i++;
		}
	moving (127,127, 500);
	}
```

Используя функцию *moving*, возможно осуществлять любые перемещения робота. Классическими задачами являются объезд препятствия, движение по различным линиям, например, восьмерке, движение змейкой, перемещение в лабиринте. Робот VEX позволяет также решать задачи с манипуляциями и перемещением объектов. Для этого рекомендуем читателям добавить к имеющейся функции moving функцию *manipulator*, аргументами которой будут скорости двигателей, отвечающих за захватывание предметов и их подъем, а также время захвата и подъема.

Движения с контролем оборота двигателей

ЗАДАЧА: Создать программный комплекс, реализующий все возможные варианты движения робота с возможностью контролировать направление движения и поворотов, количество оборотов ведущих колес.

Движения робота по таймерам или задержкам чревато большими неточностями и погрешностями. Для более точных перемещений робота используются датчики, контролирующие количество оборотов, на которые повернулся двигатель. Такие датчики называются энкодерами. К двигателям VEX 393 motors энкодеры подключаются через последовательную шину данных I2C таким образом, что в порт I2C микроконтроллера VEX подключается первый энкодер, затем в первый подключается второй, во второй третий и так далее (см. Рисунок 8. Схема подключения устройств к шине данных I2C). К микроконтроллеру VEX можно подключить 8 устройств, работающих с протоколом I2C (необязательно энкодеров).

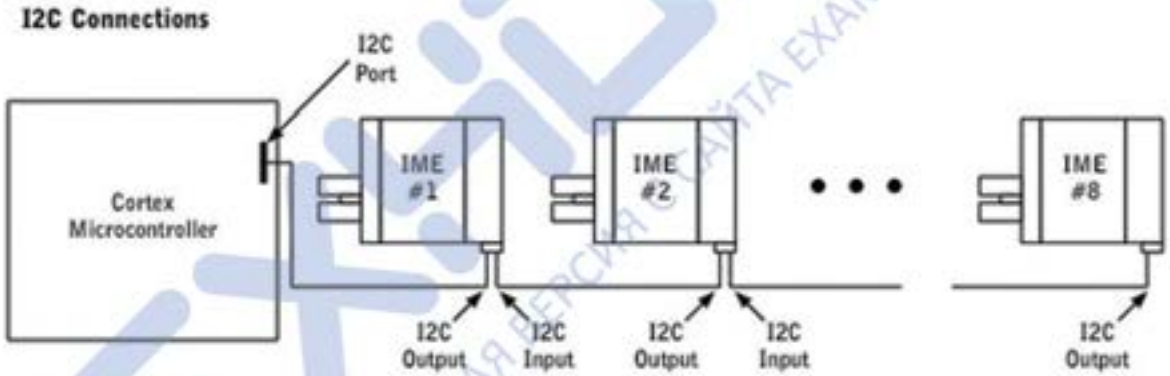
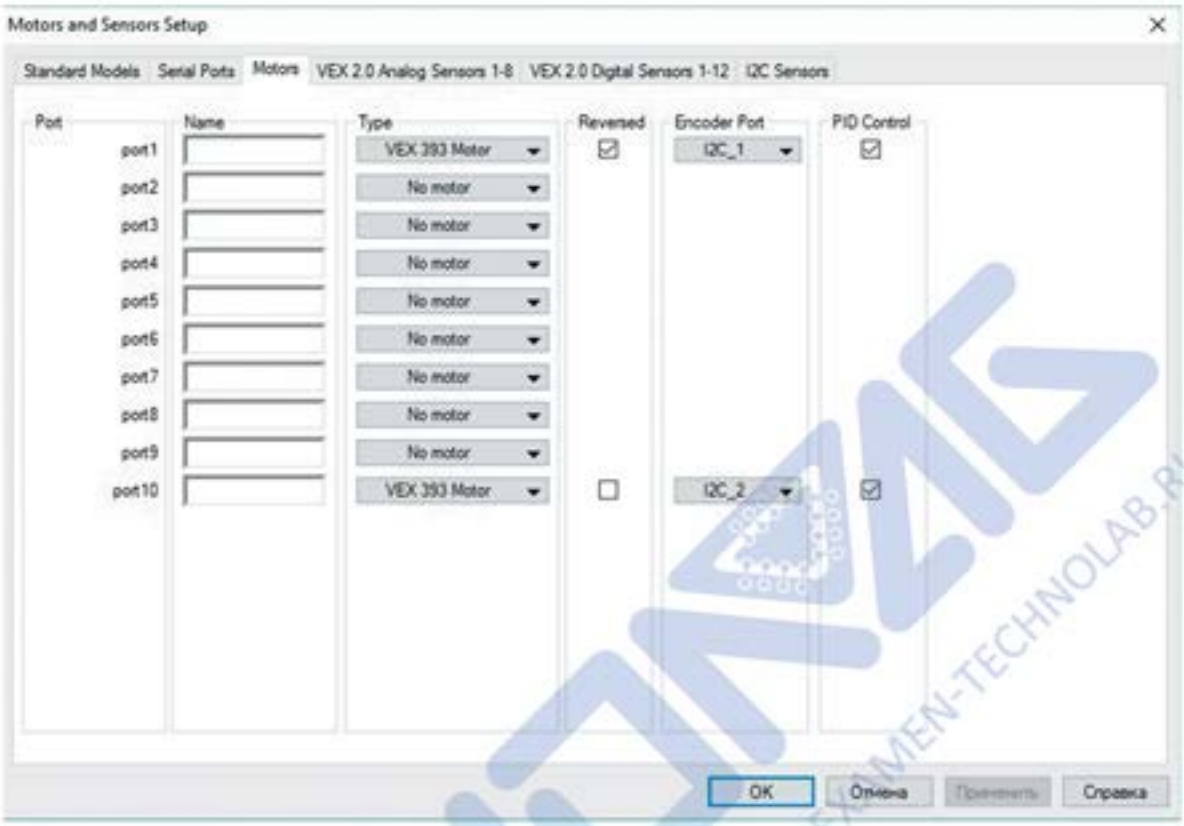
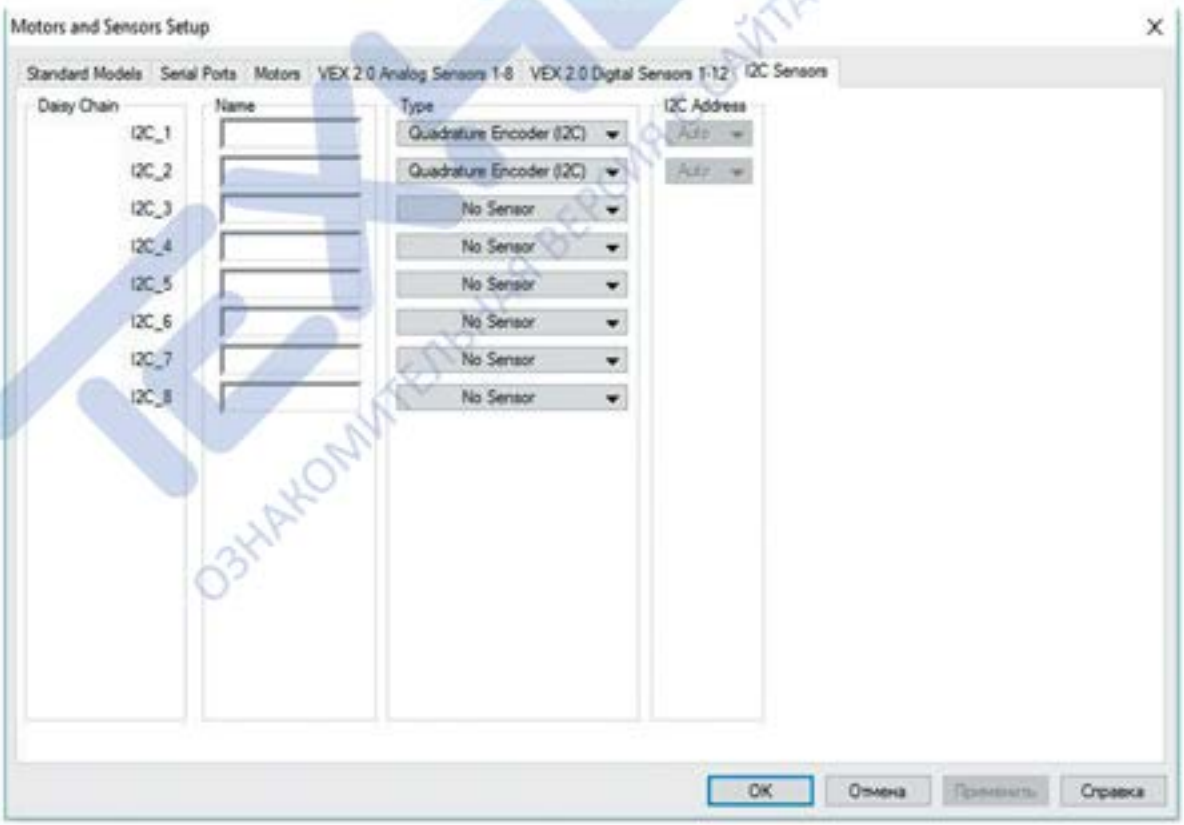


Рисунок 8. Схема подключения устройств к шине данных I2C

Необходимо указать, к какому двигателю какой энкодер относится, это делается в меню *Robot→Motors and Srensirs Setup*.





Пример 20:

```
void moving_Le (int VL, int VR, int Le)                  /* Функция движения с контролем энкодером, расположенным на правом двигателе */
    {
    nMotorEncoder[port10]=0;                             /* Обнуление энкодера, находящегося на правом двигателе [port10] */
    while (nMotorEncoder[port10]<Le)                     /* Двигатели робота будут перемещаться со скоростями VL и VR, пока значение энкодера меньше значения Le */
        {
        motor[port1]=VL;
        motor[port10]=VR;
        wait1Msec(1);
        }
    motor[port1]=0;
    motor[port10]=0;
    }

task main()
    {
    moving_Le (20,20, 100);                              /* Робот будет двигаться вперед, пока значение энкодера будет меньше 100 */
    moving_Le (-50,50, 200);                             /* Робот будет осуществлять разворот на месте по часовой стрелке, пока значение энкодера будет меньше 200 */
    }
```

Точность движений робота значительно увеличилась. Однако можно заметить, что по инерции робот всегда «проскакивает» заданное на энкодере значение. Рекомендуем Вам самостоятельно осуществить кодирование функции *moving_LR* (*int VL, int VR, int Lr*), в которой контроль скорости будет осуществляться энкодером, находящимся на правом двигателе.

Для заметок

Автономное движение робота с объездом препятствий за счет применения датчиков касания

ЗАДАЧА: Создать робота, способного самостоятельно объезжать препятствия, встреченные на пути движения.

Решим несколько задач, связанных с передвижением робота в пространстве со стенами и другими препятствиями. О существовании препятствия робот будет узнавать при помощи какого-либо из датчиков касания, в наборе VEX они представлены: *Bamper Switch* и *Limit Swich*.

Эти датчики работают одинаковым образом. Закрепим любой из датчиков касания перед роботом. Робот будет выполнять задачу по следующему циклическому алгоритму, который представлен на Рисунке 9:

1. Перемещение вперед, пока не встретится препятствие (в элемент массива, в котором хранятся данные с датчика касания, будет записана вместо *логического нуля – логическая единица*) – синяя линия;
2. Отъезд назад – оранжевая линия;
3. Разворот на месте – красная линия.

1. Перемещение вперед, пока не встретится препятствие (в элемент массива, в котором хранятся данные с датчика касания, будет записана вместо логического нуля – логическая единица).

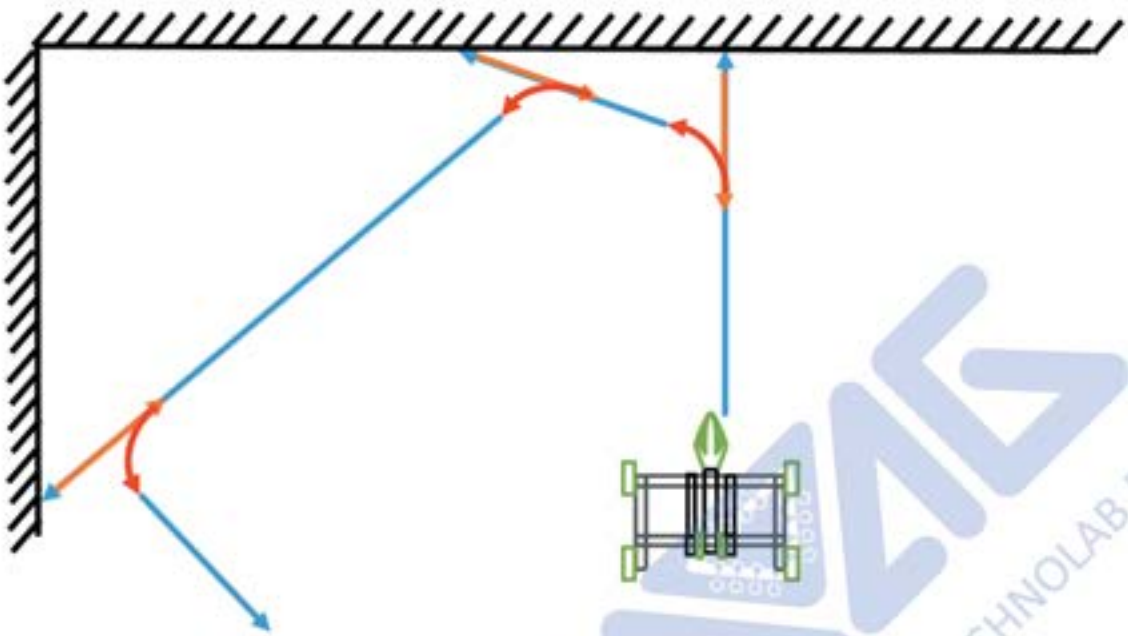


Рисунок 9. Движение робота с использованием одного датчика касания

Модернизируем базовую конструкцию таким образом, чтобы у нее впереди был бампер, на который возможно присоединить датчик касания, как это показано на Рисунке 10.

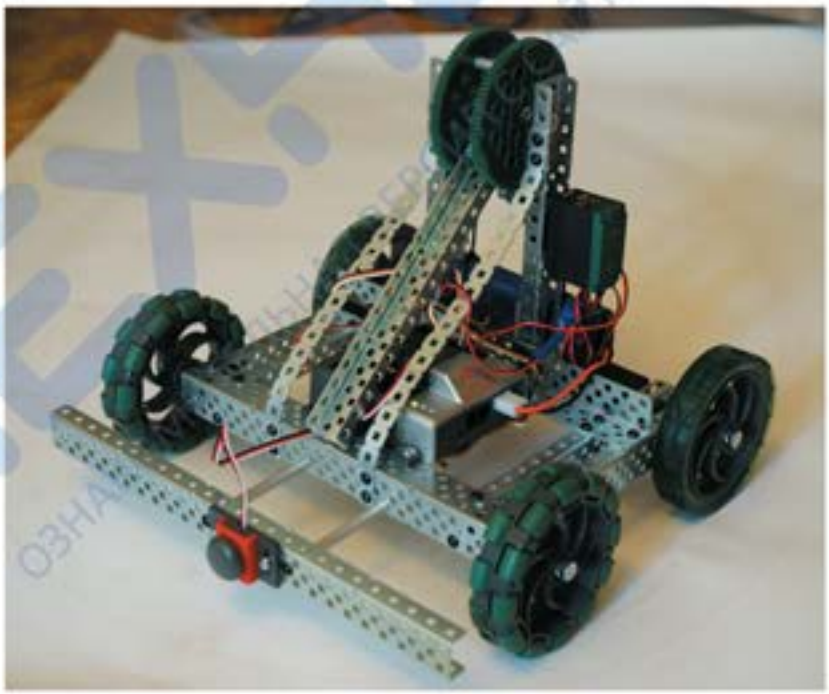


Рисунок 10. Робот с бампером и закрепленным на нем датчиком касания Bamper Switch

Необходимо также указать используемые датчики в меню *Robot→Motors and*

Sensors Setup, Touch в dgtl1. В Примерах 21, 22 и 23 поведение робота одинаково. В каждом из этих примеров иллюстрируется алгоритм, приведенный выше.

Пример 21:

```
task main()
    {
    while (1)
        {
        if (SensorValue[dgtl3]==0)
            {
            motor[port1]=60;
            motor[port10]=60;
            wait1Msec(10);
            }
        else
            {
            motor[port1]=-60;
            motor[port10]=-60;
            wait1Msec(300);
            motor[port1]=60;
            motor[port10]=-60;
            wait1Msec(500);
            }
        }
    }
```

Пример 22:

```
task main()
    {
    while (1)
        {
         if (SensorValue[dgtl3]==1)
            {
            motor[port1]=-60;
            motor[port10]=-60;
            wait1Msec(300);
            motor[port1]=60;
            motor[port10]=-60;
            wait1Msec(500);
            }
        motor[port1]=60;
        motor[port10]=60;
        wait1Msec(10);
        }
    }
```

Пример 23:

```
task main()
    {
    while (1)
        {
        while(SensorValue[dgtl3]==0)
            {
            motor[port1]=60;
            motor[port10]=60;
            wait1Msec(10);
            }
        motor[port1]=-60;
        motor[port10]=-60;
        wait1Msec(300);
        motor[port1]=60;
        motor[port10]=-60;
        wait1Msec(500);
        }
    }
```

Может возникнуть ситуация, при которой робот, упершись краем бампера в препятствие, будет продолжать движение вперед, так как датчик касания не будет нажат. В этом случае необходимо изменить конструкцию робота, введя в нее два датчика касания, находящихся по краям бампера, а также внести изменения в элементе меню *Robot→Motors and Sensors Setup*.

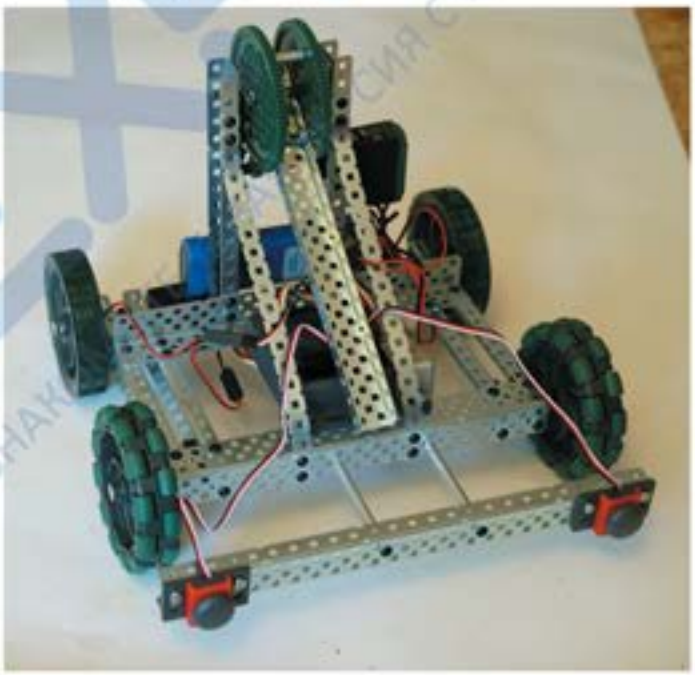


Рисунок 11. Робот с бампером и закрепленными на нем датчиками касания Bamper Switch

Пример 24:

```
task main()
    {
    while (1)
          {


          if ((SensorValue[dgtl3]==1)||(Sensor
          Value[dgtl4]==1))


                {
                motor[port1]=-60;
                motor[port10]=-60;
                wait1Msec(300);
                motor[port1]=60;
                motor[port10]=-60;
                wait1Msec(500);
                }
    motor[port1]=-60;
    motor[port1]=-60;
    }
```

```
/* Если нажать датчик dgtl3
ИЛИ датчик dgtl4, то вы-
полняются операции, нахо-
дящиеся между фигурных
скобок, следующих за опе-
ратором if() */
```

Для заметок

Датчик освещенности. Танец в круге

ЗАДАЧА: Создать подвижного робота, который не будет выезжать за границы черного круга.

Для решения задачи будем использовать датчик освещенности *Line Tracker*.

Датчик *Line Tracker* относится к аналоговым датчикам, и диапазон значений аналоговых датчиков, подключаемых к микроконтроллеру VEX, – 0..4095 (2^{12}-1). Это значение обосновывается разрядностью аналого-цифрового преобразователя VEX – 12 бит.

Line Tracker состоит из двух основных элементов: инфракрасного источника и инфракрасного приемника (оптопара). Более светлые исследуемые поверхности характеризуются меньшим значением освещенности, чем более темные поверхности. Кроме того, чем выше датчик находится над поверхностью, тем большее значение он демонстрирует. Это связано с тем, что чем выше находится ИК-источник, тем хуже он освещает поверхность, а следовательно, и ИК-приемник показывает меньшее значение. Оптимальное расстояние от поверхности до датчика 3-5 мм (см. Рисунок 12).

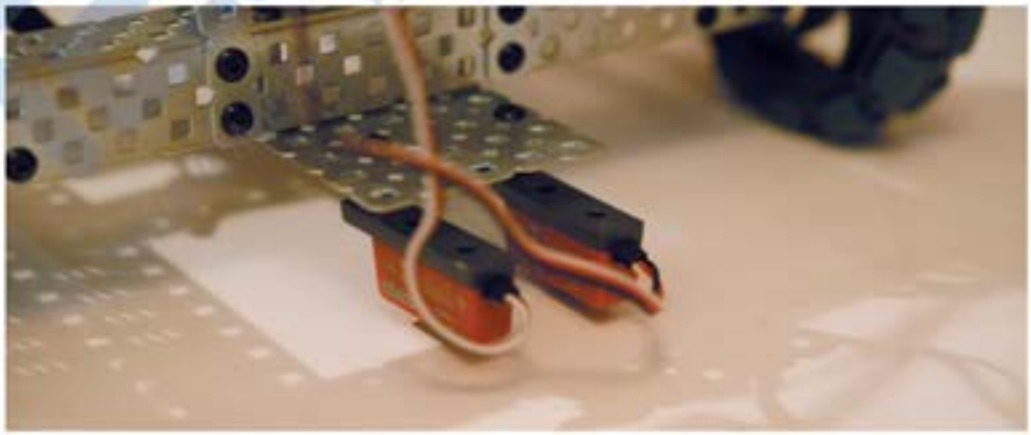


Рисунок 12. Оптимальное расстояние датчиков Line Tracker от пола

Для решения задач, связанных с движением робота на основе данных, полученных с датчиков освещенности, оптимальным расположением датчиков будет впереди робота таким образом, чтобы между ведущими колесами и сенсорами образовывался равносторонний треугольник, как это показано на Рисунке 13.

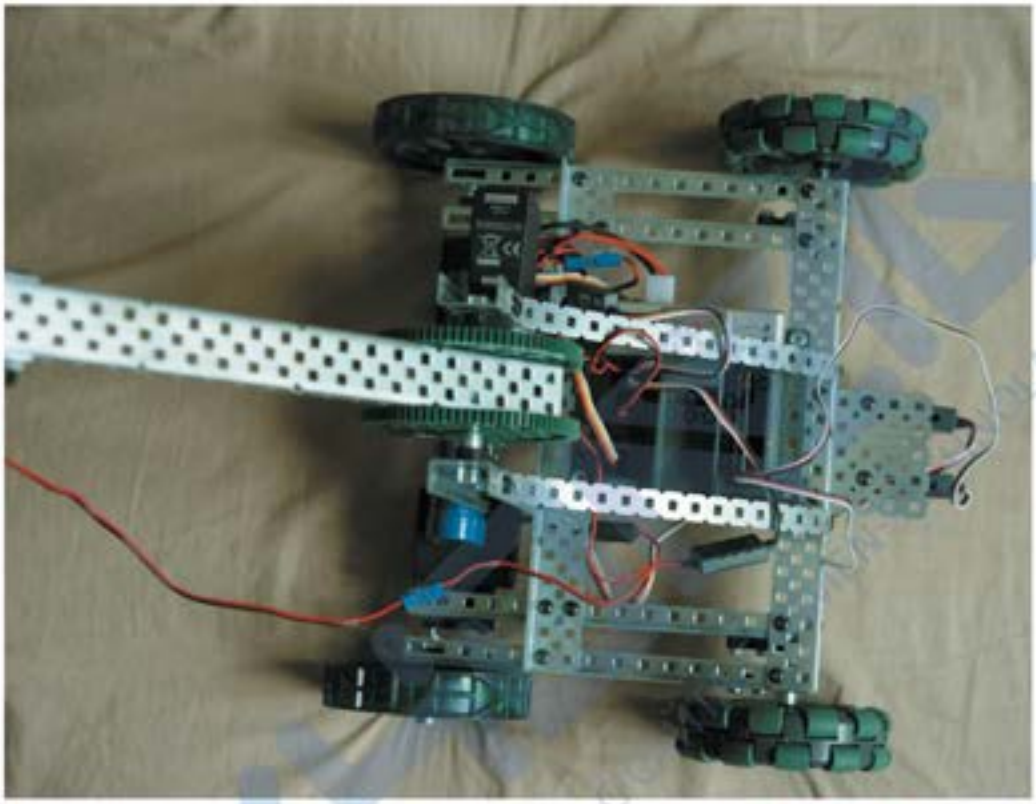


Рисунок 13. Положение датчиков Line Tracker на роботе

Важно отметить, что на показания датчика влияет и внешнее освещение (особенно солнечный свет).

ВАЖНО! Проведем эксперимент. Для его осуществления понадобятся пересекающиеся черная и белая поверхности. Поместим датчик над черной поверхностью, а затем начнем перемещать его к белой поверхности. В результате увидим, что значения датчика изменяются не скачкообразно, а плавно. Это связано с тем, что датчик измеряет освещенность не в одной точке, а на некоторой площади. На Рисунке 14 положение этой площади обозначается кругами *L1 – L5*, пусть им соответствуют значения *Line Tracker SL1 – SL5*. Очевидна следующая закономерность: *SL1<SL2<SL3<SL4<SL5*.

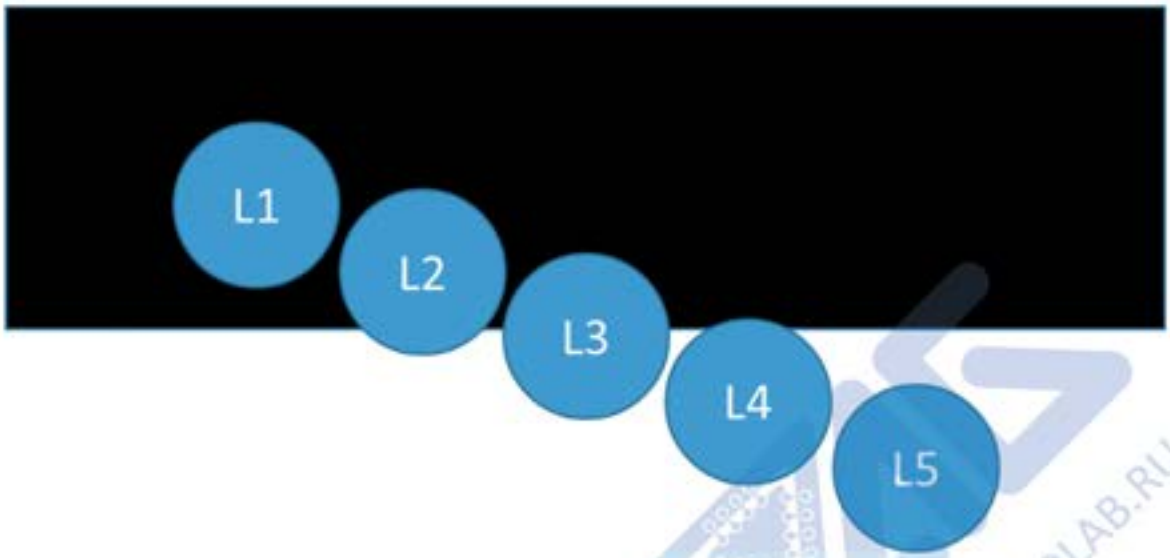


Рисунок 14. Положение датчика освещенности над границей черной и белой поверхностей

Для решения задачи движения робота в границах окружности необходимо использовать либо напечатанное на баннере поле, либо изготовить это поле самостоятельно. Для этого понадобятся белая поверхность из баннера или тыльной стороны линолеума и изолента или клейкая лента. Как показано на Рисунке 15, ширина линии должна быть не менее 5 см. Значение датчика для границы черной линии и белого фона найдем как среднее арифметическое между показаниями на белом и черном фоне. В условиях написания данного пособия это:

$$\frac{2232 + 144}{2} = 1188$$

Алгоритмы же поведения роботов подобны примерам, рассмотренным в разделе «Движение с использованием датчиков касания», см. Рисунок 15 и Пример 25.

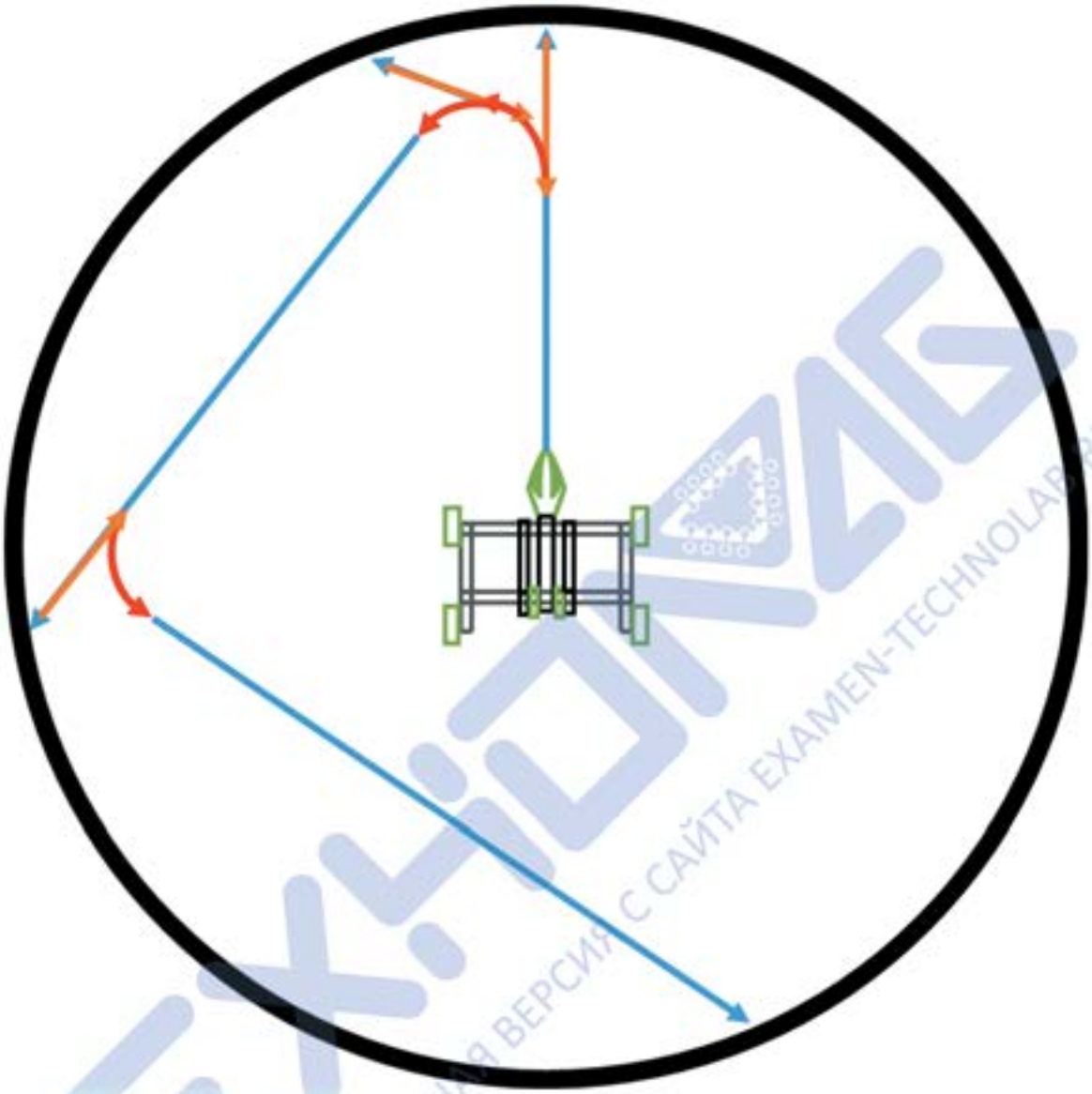


Рисунок 15. Алгоритм выполнения задачи «Танец в круге»

Пример 25:

```
task main()
    {
    while (1)
            {
            if (SensorValue[in1]>1188)
                    {
                    motor[port1]=-40;
                    motor[port10]=-40;
                    wait1Msec(500);
                    motor[port1]=40;
                    motor[port10]=-40;
                    wait1Msec(500);
                    }
            motor[port1]=40;
            motor[port10]=40;
            wait1Msec(10);
            }
    }
```

При выполнении роботом программы, описанной в Примере 25, может произойти следующий сбой: робот при развороте может оказаться за пределами окружности. Для того чтобы избежать этого, необходимо осуществить движение назад на такое же расстояние, как и движение вперед. Эту задачу можно осуществить двумя способами:

- с использованием таймера
- с использованием энкодера

Пример 26 демонстрирует решение этой задачи с использованием энкодера.

Пример 26:

```
task main()
    {

    nMotorEncoder[port1]=0;                          /* Необходимо сбросить
                                                     значение энкодера */

    while (1)
          {
          if (SensorValue[in1]>1188)
                {

                                                     /* Отъезд назад, пока не
                while (nMotorEncoder[port1]<=0)      вернемся в начальную точ-
                                                     ку */

                      {
                      motor[port1]=-40;
                      motor[port10]=-40;
                      }
                motor[port1]=40;
                motor[port10]=-40;
                wait1Msec(500);

                                                     /* Перед движением впе-
                nMotorEncoder[port1]=0;              ред сбрасываем значение
                                                     энкодера */

                }

                                                     /* Пока не доехали до ли-
          else                                       нии, передвигаемся впе-
                                                     ред */

                {
                motor[port1]=40;
                motor[port10]=40;
                wait1Msec(2);
                }
          }
    }
```

На основе движений робота в круге можно построить большое количество задач. Одна из самых распространенных – это «Кегильринг». Суть ее сводится к выталкиванию роботом определенного количества цилиндров из круга. Наличие манипулятора в базовой версии набора VEX EDR позволяет значительно расширить количество таких заданий, внеся, например, необходимость переставить какой-либо объект в круге.

Движение по линии на одном датчике

Задача: Создать робота, который будет перемещаться по черной линии на белом фоне.

Программу из Примера 25 возможно модернизировать таким образом, чтобы робот совершал движения по черной линии. Этот алгоритм очень прост:

1. Робот перемещается вперед до черной линии – синяя стрелка;

2. Робот поворачивает внутрь замкнутого контура – красная стрелка.

1. Робот перемещается вперед до черной линии – синяя стрелка.

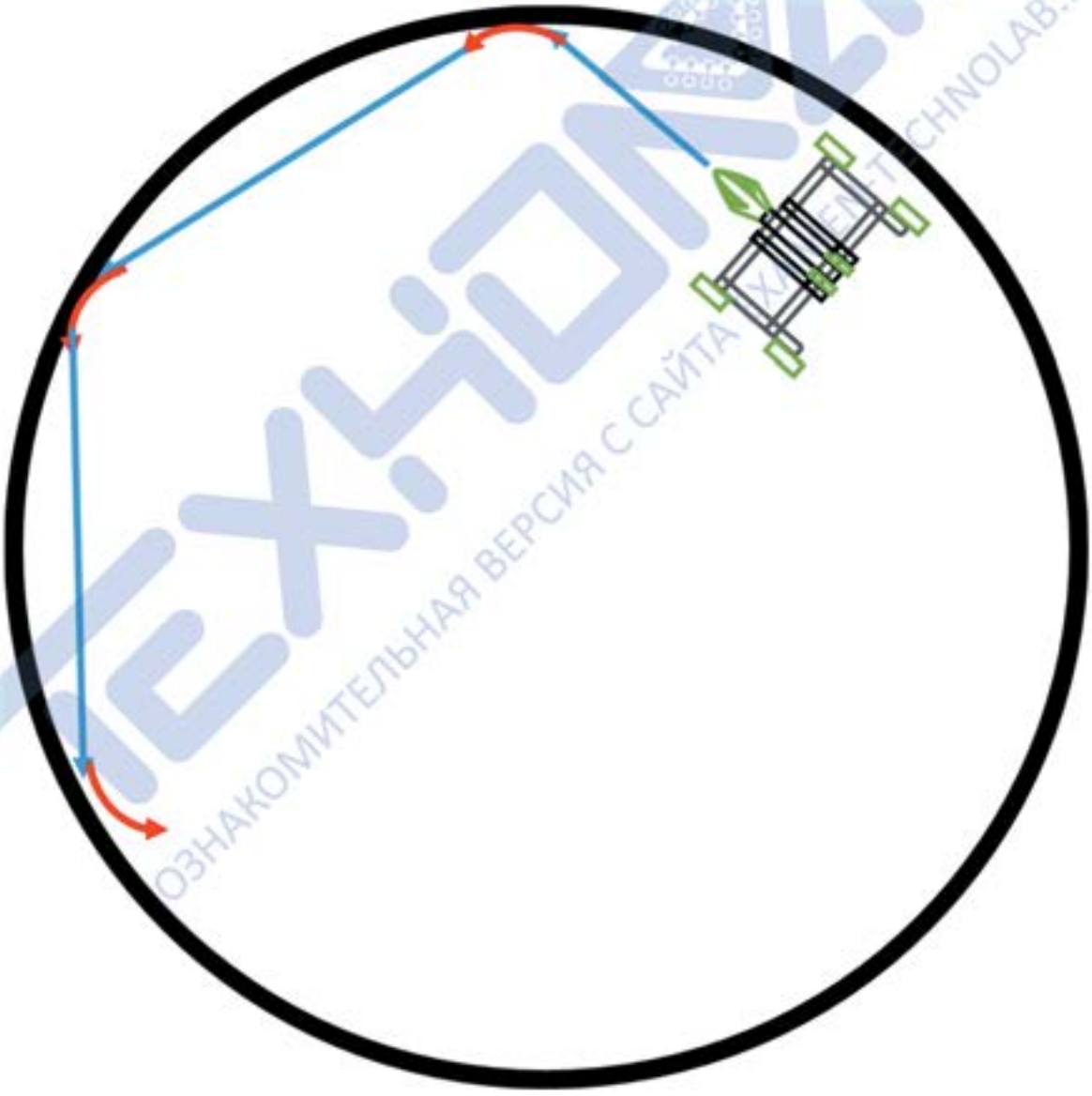


Рисунок 16. Алгоритм движения по линии на одном датчике Line Tracker

Пример 27:

```
task main()
    {
    while (1)
        {

        if (SensorValue[in1]>1188)

            {
            motor[port1]=40;
            motor[port10]=-40;
            }

        else

            {
            motor[port1]=40;
            motor[port10]=40;
            }
        wait1Msec(2);
        }
    }
```

/* Если значение Line Tracker больше 1188, то робот осуществляет разворот на месте */

/* Если значение Line Tracker меньше 1188, то робот перемещается вперед */

Сложные ветвления. Пульт из датчиков касания

ЗАДАЧА: Осуществить управление при помощи 4-позиционного пульта управления.

В случае использования датчика касания из параграфа «*Автономное движение робота с объездом препятствий за счет применения датчиков касания*» возможно закодировать 2 события – первое будет соответствовать *нажатию* (логическая *единица* или «*Ложь*»), второе состоянию, когда кнопка *отпущена* (логический ноль или «*Истина*»). Сколько датчиков касания необходимо, чтобы закодировать большее количество событий? На этот вопрос легко ответить, воспользовавшись формулой Хартли:

$$k = log_2^N,$$

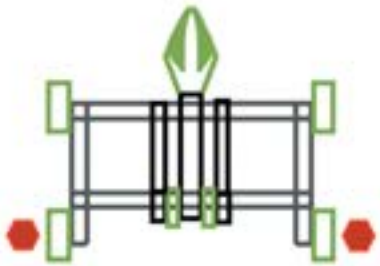
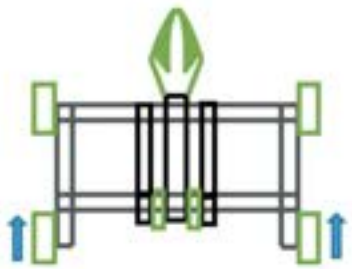
где N – количество состояний;

k – количество датчиков касания.

Следовательно, имея один датчик касания, возможно осуществить команды только для перемещения вперед-назад. Имея же два датчика касания, возможно осуществить управление четырьмя разными процессами.

Сняв бампер с робота, представленного на Рисунке 11, получим двухкнопочный пульт управления роботом, который позволяет запрограммировать четыре различных движения. Эти движения и связанные с ними состояния датчиков показаны в Таблице 4.

Таблица 4. Перемещение робота, управляемого с двухкнопочного пульта

Рисунок	Левый датчик	Правый датчик
	Отпущен	Отпущен
	Нажат	Нажат
	Нажат	Отпущен
	Отпущен	Нажат

В Примере 21 приведен код для управления роботом, используется один датчик касания *dgtl 3*.

Пример 28:

```
task main()
    {
    while (1)
        {
        if (SensorValue[dgtl3]==1)
        {
    motor[port1]=60;
    motor[port10]=60;
        }
        else
        {
    motor[port1]=0;
    motor[port10]=0;
        }
        wait1Msec(2);
        }
    }
```

Усложним задачу: теперь робот будет работать по схеме, представленной в Таблице 4.

Пример 29:

```
task main()
    {
    while (1)
        {
        if (SensorValue[dgtl3]==1)
            {
            if (SensorValue[dgtl4]==1)
                {                                   /* Оба датчика нажаты */
                motor[port1]=60;
                motor[port10]=60;
                }
            else
                                                    /* первый – нажат, второй –
                {                                   нет */

                motor[port1]=60;
                motor[port10]=0;
                }
            else
```

```
                {
                if (SensorValue[dgtl4]==1)
                        {                       /* Второй – нажат, первый –
                                                отпущен */
                        motor[port1]=0;
                        motor[port10]=60;
                        }
                else
                        {                       /* Оба датчика отпущены */
                        motor[port1]=0;
                        motor[port10]=0;
                        }
                }
                wait1Msec(2);
        }
}
```

Релейный регулятор. Удерживание подъемного устройства манипулятора

ЗАДАЧА: Создать робота, который способен удерживать подъемное устройство манипулятора на определенной высоте.

Подавляющее количество задач в робототехнике связано с использованием *регуляторов*. Регулирование – это процесс приведения какой-либо системы к заданным параметрам.

Таким образом, для регулирования необходимо ответить на следующие *4 вопроса регулирования*:

1. Какой из параметров системы необходимо изменять?
2. Как осуществляется количественное измерение этого параметра?
3. Чему равно необходимое значение выбранного параметра?
4. Каким способом осуществить изменение параметра?

Самый простой пример регуляторов – это *релейный регулятор*. В случае использования *релейного регулятора* на систему осуществляется воздействие фиксированным количеством различных *способов*. Например, в предыдущем параграфе в Примере 28 на положение робота можно было повлиять двумя способами – вращать оба двигателя так, чтобы он передвигался вперед, или вращать их в обратную сторону. В Примере 29 реализовано уже 4 способа.

ВАЖНО! В Примерах 28 и 29 некорректно говорить о регулировании, так как не заданы какие-либо параметры системы.

Для решения *Задачи* этого параграфа ответим на *4 вопроса регулирования*:

1. Угол наклона подъемного устройства манипулятора.
2. При помощи энкодера.
3. Необходимое значение энкодера – -500 единиц.
4. Поднимая вверх, если значение энкодера меньше 500, опуская вниз, если значение больше.

Таким образом, видно, что в данной задаче реализовано два состояния (способа воздействия на систему):

1. Подъем подъемного устройства.
2. Опускание подъемного устройства.

Следовательно, в Примере 30 реализован релейный регулятор с двумя состояниями (способами воздействия).

Пример 30:

```
task main()
    {
    nMotorEncoder[port7]=0;
    while (1)
        {
        if (nMotorEncoder[port7]>-500)
            {
            motor[port7]=-127;
            }
        else
            {
            motor[port7]=127;
            }
    wait1Msec(2);
        }
    }
```

ВАЖНО! Обратите внимание на то, что робот находится в нестабильном состоянии. Можно уменьшить эту нестабильность, снижая скорость двигателя. Но полностью избавиться от колебания подъемного устройства невозможно. Это общая черта всех регуляторов, работающих по принципу реле. Реализуя только два состояния, устройство гарантированно не сможет остановиться при достижении необходимого значения.

Движение по линии на одном датчике с использованием релейного регулятора

ЗАДАЧА: Осуществить движение робота по черной линии на основе релейного регулятора.

Ответим на *4 вопроса регулирования*:

1. Положение датчика относительно границы черной линии с учетом того, что робот перемещается.

2. При помощи датчика *Line Tracker*.

3. Датчик должен находиться над границей черного и белого, что соответствует среднему значению показаний датчика между черным и белым полем (подробнее смотри тему «Датчик освещенности. Танец в круге»), что соответствует 1188 (в момент написания пособия).

4. Изменяя разницу в скоростях вращения двигателей.

Алгоритм движения робота по линии на одном датчике с использованием релейного регулятора следующий:

```
if (робот находится на белом)
    {
    Поворачивать в сторону черной линии;
    }
else
    {
    Поворачивать в сторону белого фона;
    }
```

Пример 31:

```
task main()
    {
    while (1)
        {
        if (sensorValue[in1]>1188)
            {
            motor[port1]=127;
            motor[port10]=0;
            }
        else
            {
            motor[port1]=0;
            motor[port10]=127;
            }
        wait1Msec(2);
        }
    }
```

Для того чтобы снизить амплитуду движений робота, а также увеличить его скорость, необходимо подобрать значения скоростей левого и правого двигателей таким образом, чтобы в любой момент были включены оба двигателя, но они работали с различными скоростями.

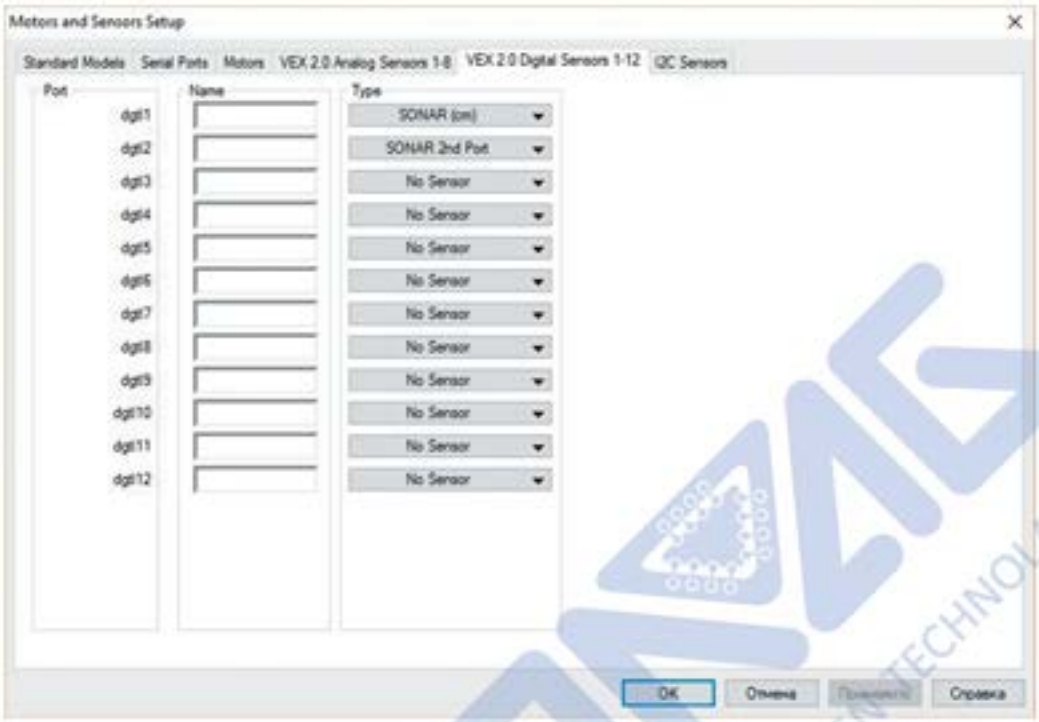
Движение вдоль стены по датчику расстояния с использованием релейного регулятора

ЗАДАЧА: Осуществить движение робота вдоль стены на основе релейного регулятора.

Данная задача подобна той, что была решена в предыдущем параграфе. Датчик расстояния (*Ultrasonic Range Finder*) необходимо установить под углом к направлению движения робота.

ВАЖНО! Показания *Ultrasonic Range Finder* сильно зависят не только от расстояния до объекта, но и от угла, под которым этот объект обращен к датчику.

Датчик Ultrasonic Range Finder конфигурируется в *Robot→Motors and Sensors Setup*, в разделе VEX 2.0 Digital Sensors 1-12. Удобнее всего указать единицы измерения – *см.*



Датчик необходимо разместить под углом к направлению движения робота, например так, как показано на Рисунке 17.

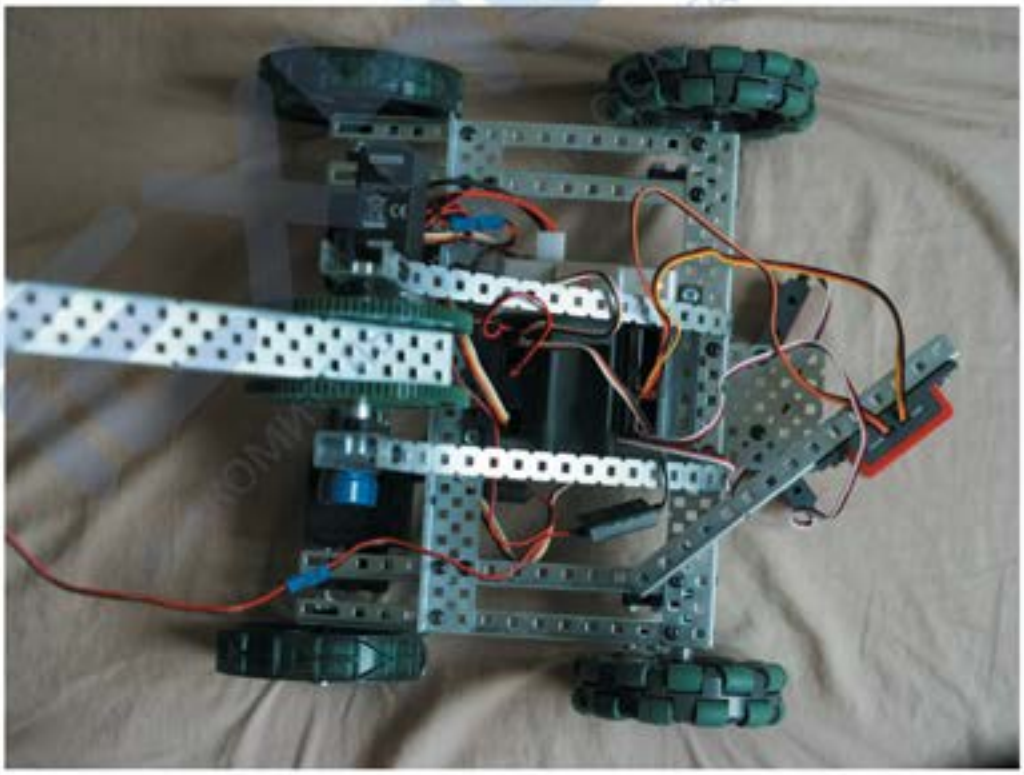


Рисунок 17. Расположение датчика Ultrasonic Range Finder на роботе

Вновь ответим на *4 вопроса регулирования*:

1. Расстояние датчика до стены, с учетом того что робот перемещается.

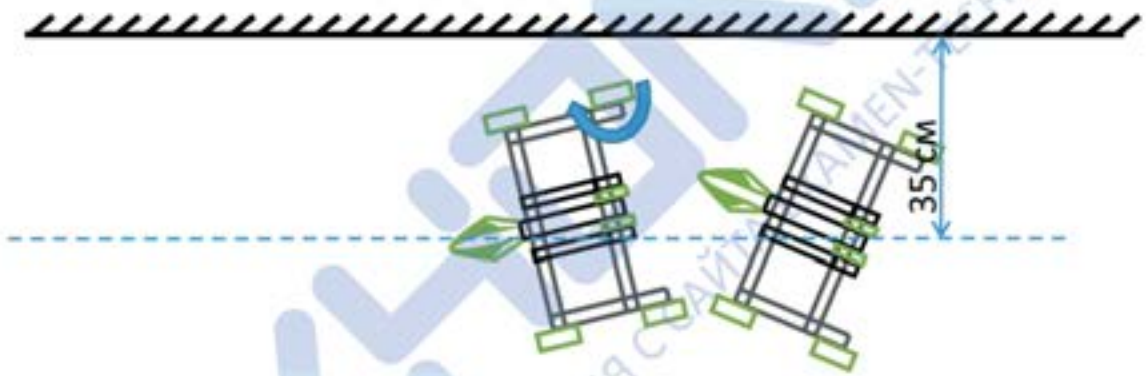
2. При помощи датчика *Ultrasonic Range Finder*.

3. Датчик должен находиться на расстоянии *35 см* от стены.

4. Изменяя разницу в скоростях вращения двигателей.

Алгоритм движения робота по линии на одном датчике с использованием релейного регулятора следующий:

if (расстояние робота от стены больше *35 см*)
{
Поворачивать к стене;
}
else
{
Поворачивать от стены;
}



Пример 32:

```
task main()
    {
    while (1)
          {
          if (sensorValue[dgtl1]>35)
                {
                motor[port1]=60;
                motor[port10]=0;
                }
          else
                {
                motor[port1]=0;
                motor[port10]=60;
                }
          wait1Msec(2);
          }
    }
```

Для заметок

Движение вдоль линии на двух датчиках

ЗАДАЧА: Осуществить движение робота вдоль черной линии с использованием двух датчиков освещенности.

При выполнении роботом задания *Движение по линии на одном датчике с использованием релейного регулятора* очевиден следующий недостаток – «теряя» линию, робот не может вернуться на нее обратно.

Для преодоления вышеуказанного недостатка необходимо использовать два датчика *Line Tracker*. Напомним, что для решения задач такого типа датчики освещенности и ведущие колеса должны образовывать равносторонний треугольник, смотри Рисунок 13.

Ответим на *4 вопроса регулирования*:

1. Положение датчиков относительно черной линии.

2. При помощи двух датчиков *Line Tracker*.

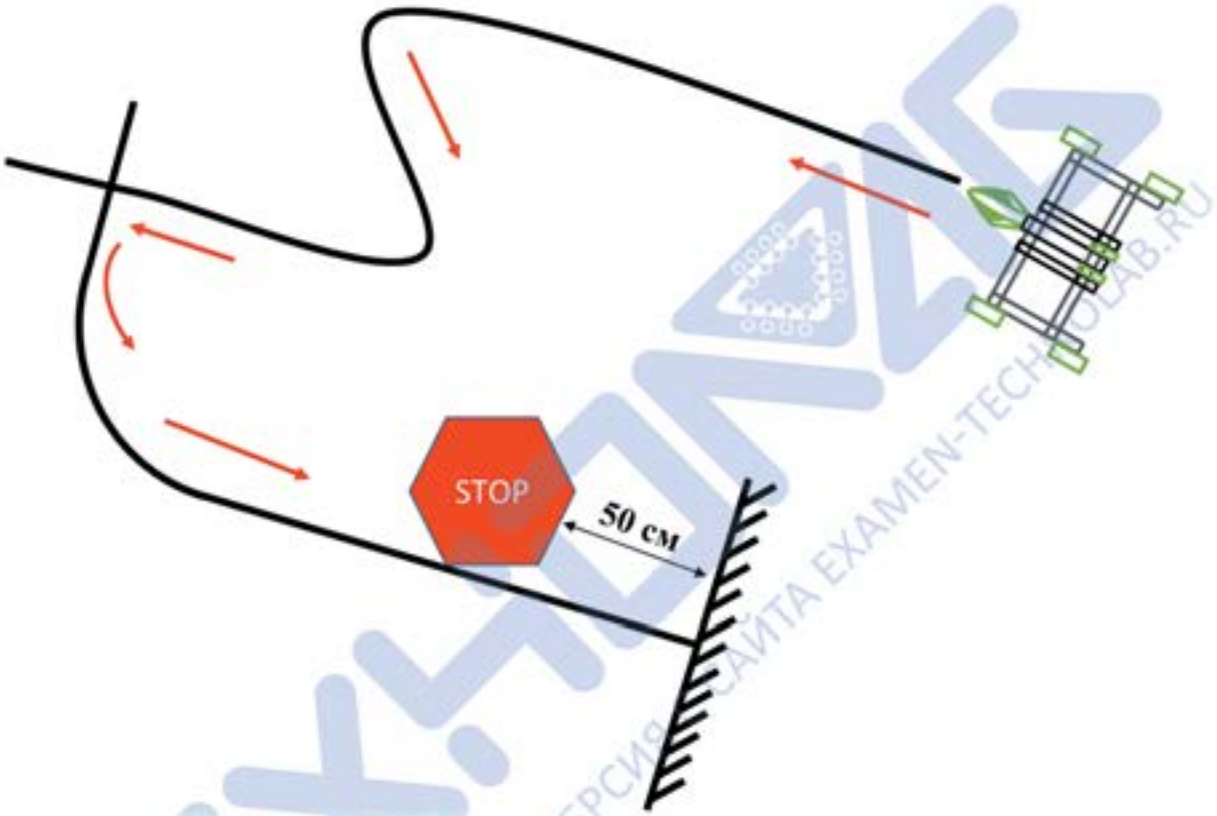
3. Черная линия должна находиться между датчиками.

4. Изменяя разницу в скоростях вращения двигателей.

Алгоритм движения робота по линии на одном датчике с использованием релейного регулятора следующий:

```
if (левый датчик на белом)
        {
        if (правый датчик на белом)
                {
                Робот двигается вперед;
                }
        else
                {
                Робот поворачивает направо;
                }
        }
else
        {
        if (правый датчик на белом)
                {
                Робот поворачивает налево;
                }
        else
                {
                Робот двигается вперед;
                }
        }
```

Вообще, используя данный алгоритм, возможно придумывать для робота и решать огромное количество задач. Например, роботу необходимо двигаться по линии до перекрестка, на перекрестке развернуться на месте против часовой стрелки и далее продолжать движение по линии, пока до впереди стоящей стены не останется 50 см. Для данной задачи датчик *Ultrasonic Range Finder* необходимо установить впереди по ходу движения робота.



Пример 33:

```
void line (VMax, Vmin)
    {
    if ( sensorValue[in1]>1188)
            {
                if (sensorValue[in2]>1188)
                    {
                    motor[port1]=VMax;
                    motor[port10]=VMax;
                    }
                else
                    {
                    motor[port1]=VMin;
                    motor[port10]=VMax;
                    }
            }
    else
            {
            if ( sensorValue[in2]>1188)
                {
                motor[port1]=VMax;
                motor[port10]=VMin;
                }
            else
                {
                motor[port1]=VMin;
                motor[port10]=VMin;
                }
            }
    }

task main ()
    {
    while ((sensorValue[in2]>1188)||
    (sensorValue[in1]>1188)))

            {

            line (80, 20);

            }
    motor[port1]=-100;
    motor[port10]=100;
```

/* Двигаться по линии, пока оба датчика не на черном */

/* Вызов функции line с параметрами VMax=80, VMin=20 */

```
                                        /* Продолжать разворачиваться,
while ( sensorValue[in1]<1188)          пока левый датчик на окажется
                                        на белом */

        {
        }

                                        /* Продолжать разворачиваться,
while ( sensorValue[in1]>1188)          пока левый датчик на окажется
                                        на черном */

        {
        }

                                        /* Продолжать разворачиваться,
while ( sensorValue[in1]<1188)          пока левый датчик на окажется
                                        на белом */

        {
        }
while (sensorValue[dgtl1]>50)           /* Двигаться по линии, пока до
                                        стены не будет меньше 50 см */

        {
        line (80, 20);
        }
}
```

Пропорциональный регулятор. Удерживание манипулятора

ЗАДАЧА: Создать робота, который способен удерживать подъемное устройство манипулятора на определенной высоте с использованием пропорционального регулятора.

В предыдущих параграфах управление действиями робота осуществлялось на основе *релейного регулятора*, главным недостатком которого является то, что нет возможности достичь необходимого значения выбранного параметра, и, как следствие, низкая точность действий, что приводит, в свою очередь, к невозможности достигнуть плавности движений.

Для преодоления вышеуказанных недостатков используется так называемый *Пропорционально-интегрально-дифференциальный (ПИД) регулятор*. Этот регулятор состоит из трех составляющих: пропорциональной, интегральной и дифференциальной. Все эти составляющие тем или иным образом зависят от величины, называемой *ошибкой*. *Ошибка* – это величина, которая характеризует отклонение регулируемой величины от заданного значения. В этом параграфе нам необходимо удерживать подъемное устройство под углом, которому соответствуют показания энкодера -500 единиц. Любое отклонение от этого значения является *ошибкой*. Будем использовать пропорциональный регулятор. В этом случае управляющее воздействие (мощность двигателя) должно быть пропорционально ошибке. Так как в дальнейшем может понадобиться осуществлять перемещения робота с поднятым манипулятором, в Примере 34 пропорциональный регулятор для подъемного устройства реализован в параллельной задаче с использованием двух глобальных переменных.

Пример 34:

```
int k=0;                                    /* Коэффициент усиления */

int alph=0;                                 /* Угол, на который необхо-
                                            димо будет поднять манипу-
                                            лятор */

task up ()
    {
    nMotorEncoder[port7]=0;
    while (1)
        {

                                            /* В круглых скобках вычис-
                                            ляется ошибка, затем она
        motor[port7]=                       умножается на коэффициент
        k*(alph-nMotorEncoder[port7]);      k. Результат – подаваемая на
                                            двигатель мощность */

        wait1Msec (1);
        }
    }

task main()
    {

                                            /* Запускается задача, кото-
    StartTask (up);                         рая будет выполняться парал-
                                            лельно с основной */

    k=-2;
    alph=-500;                              /* Подъемник поднимается */
    wait1Msec (6000);
    k=-2;
    alph=0;                                 /* Подъемник опускается */
    wait1Msec (6000);
    }
```

Разберем Пример 34 подробнее. В параллельной задаче *up* осуществляется управление подъемным устройством. В основной задаче осуществляется изменение переменных, которые используются и в задаче *up*. В момент начала выполнения программы переменная *alph*, которая характеризует угол, на который должен подняться манипулятор, равна *0*. Значение энкодера *nMotorEncoder[port7]* тоже равно *0*. Следовательно, мощность, подаваемая на двигатель *motor[port7]*, равна *0*. В момент, когда переменной *alph* присвоено значение 500, ошибка становится равной *(alph-*

nMotorEncoder[port7])=-500, после умножения ее на коэффициент *k=-2* получим число, равное 1000. Именно такая мощность и подается на двигатель, однако, как было сказано выше, диапазон мощностей для управления двигателем от -127 до 127. Внутренние алгоритмы контроллера *VEX* превращают любую мощность выше 127 в 127, в свою очередь, если мощность, подаваемая на двигатель, менее -127 единиц, она приравнивается к -127. Таким образом, в начальный момент времени на двигатель подается максимальная мощность. Манипулятор начинает подъем, при этом ошибка *(alph-nMotorEncoder[port7])* уменьшается по модулю. Рассмотрим момент времени, когда показания энкодера *nMotorEncoder[port7]=-450*. В этом случае ошибка равна -50, умножив ее на коэффициент -2, получим мощность, подаваемую на двигатель в 100 единиц. Чем ближе к заданному значению в *-500 единиц* будет находиться манипулятор, тем меньше будет скорость двигателя. В случае же если двигатель «проскочит» необходимое значение в *-500*, направление вращения двигателя сменится. Например, при значении *-510* мощность, подаваемая на двигатель, будет равна -20. Как только показания *энкодера* будут равны значению *-500*, мощность, подаваемая на двигатель, тоже будет равна *0*.

Таким образом, действия робота направлены на то, чтобы уменьшить ошибку. По-другому можно сказать, что в данном случае реализована *отрицательная обратная связь*.

Обратите внимание на то, что робот, работая по алгоритму, представленному в Примере 34, рывком поднимает манипулятор, совершает несколько осцилляций, а затем удерживает манипулятор в одной точке.

Проведите эксперименты, увеличивая коэффициент *k* по модулю. Обратите внимание на то, что с увеличением *k* возрастает «резкость» движений робота, ему требуется большее время для фиксирования в необходимой точке. При дальнейшем увеличении *k* возникает так называемое «перерегулирование» – поведение робота подобно тому, как будто используется релейный регулятор. При уменьшении же *k* у робота может не хватить мощности, чтобы достичь заданного значения. Этот коэффициент называется *коэффициент усиления пропорционального регулятора*.

Таким образом, видно, что в пропорциональном регуляторе управляющее воздействие пропорционально ошибке. Следовательно, для его реализации необходимо к четырем вопросам управления добавить пятый, для удобства он будет сформулирован внутри третьего вопроса:

1. Какой из параметров системы необходимо изменять?

2. Как осуществляется количественное измерение этого параметра?

3. Чему равно необходимое значение выбранного параметра? Как рассчитать ошибку между текущим значением и необходимым?

4. Каким способом осуществить изменение параметра?

Вообще, используя только пропорциональный регулятор, возможно решать подавляющее количество задач робототехники. В следующих параграфах будет рассмотрена часть из них.

Рекомендуем Вам самостоятельно дополнить код в Примере 34 таким образом, чтобы, используя функцию *up*, было возможно управлять не только высотой подъема манипулятора, но и самим манипулятором.

Езда по линии на одном датчике и вдоль стены на пропорциональном регуляторе

ЗАДАЧИ:

1. Создать робота, который способен передвигаться по черной линии, используя один датчик *Line Tracker*, на пропорциональном регуляторе.

2. Создать робота, который способен передвигаться вдоль стены, используя один датчик *Ultrasonic Range Finder* на пропорциональном регуляторе.

В данном параграфе рассматриваются две подобные друг другу задачи. Разница в коде – это используемый датчик и коэффициент усиления.

Ответим на 4 вопроса для движения по линии.

1. Положение датчика *Line Tracker*, установленного на робота, относительно границы черной линии с учетом того, что робот перемещается.

2. При помощи датчика *Line Tracker*. Если робот находится на белом фоне, то датчик показывает меньшее значение, чем значение границы черного и белого, если на черном – большее.

3. Датчик должен находиться над границей черного и белого, что соответствует среднему значению показаний датчика между черным и белым полем (подробнее смотри тему «Датчик освещенности. Танец в круге»), что соответствует 1188 (в момент написания пособия). В случае достижения этого параметра *ошибка* становится равной нулю и робот двигается прямолинейно. Ошибка вычисляется как разница между необходимым значением 1088 и текущим значением *Line Tracker*.

4. Изменяя разницу в скоростях вращения двигателей.

Программа для движения по линии представлена в Примере 35

Пример 35:

```
task main()
    {
    int kp=4;
    int upp;
    int V=40;
    int grey =1188;
    while (1)
        {

        upp=kp*(grey-SensorValue[in1])/100;   /* По сути коэффициент
                                                 усиления равен 0,04 */

        motor[port1]=V-upp;
        motor[port10]=V+upp;
        wait1Msec (1);
        }
    }
```

ВАЖНО! В Примере 35 используется переменная *upp*, введение которой в общем является избыточным. В этом случае код выглядел бы так:

```
motor[port1]=V- kp*(grey-SensorValue[in1])/100;
```

однако в этом случае считывание с датчика SensorValue[in1] пришлось бы делать два раза для левого и правого двигателя. А это более длительная задача, чем запись в переменную, что связано с необходимостью считать сигнал, перевести его в цифровой вид из аналогового.

Точные движения робота, основанные на использовании пропорционального регулятора и энкодеров

ЗАДАЧА: Создать подпрограммы для реализации точных движений робота с использованием энкодеров на основе алгоритма пропорционального регулирования.

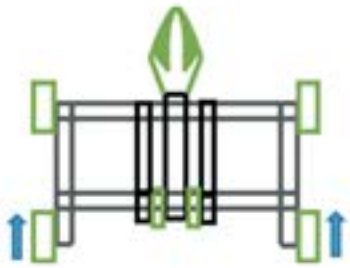
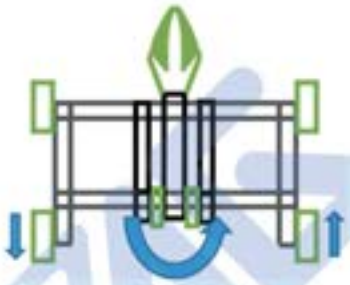
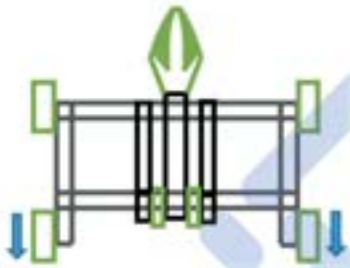
В параграфе *Движения с контролем оборота двигателей* мы уже осуществляли контроль движения робота при помощи энкодеров. Тогда даже при существенном уменьшении скорости двигателей робот «проскакивал» необходимые значения по инерции. В данном параграфе при помощи пропорционального регулятора добьемся того, чтобы робот прекращал свои движения в заданном положениии. В Примере 36 робот может совершать точные движения при перемещении вперед-назад, разворачиваться на месте по и против часовой стрелки с учетом показаний энкодеров. В примере учтено и то, что при движении робота вперед энкодер на правом двигателе *motor[port10]* увеличивает свои показания, а на левом двигателе *motor[port1]* – уменьшает. Для удобства за все точные движения робота отвечает функция *moving (int V, int l, int n)*, ее можно применять в любом месте программы, если есть необходимость совершить точный маневр или серию точных маневров.

Аргументы функции это:

- скорость совершения маневра *V*, чем она меньше, тем точнее будет выполнено движение;
- *l* – величина, которая пропорциональна углу поворота двигателя, в случае прямолинейного движения она по сути характеризует расстояние, на которое будет перемещаться робот, в случае разворота – угол разворота;
- *n* – при значении *n=1* робот двигается поступательно, при *n=-1* робот совершает разворот на месте.

То же в табличной форме.

Таблица 5. Зависимость поведения робота от аргументов функции moving

	l>0	l<0
n=1		
n=-1		

Функция *moving* состоит из двух циклов. В первом двигатели совершают 80% от необходимой работы. При этом работа двигателей синхронизирована за счет использования пропорционального регулятора. *Ошибка* в данном случае это разница между модулями показаний энкодеров. Левый двигатель *motor[port1]* стремится к тому, чтобы показания его энкодера были равны показаниям правого двигателя *motor[port10]*. В этой части точность движения не очень высока и может быть улучшена в следующей части программы.

За остальную часть (20%) отвечает второй цикл. Он выполняется, пока показания таймера менее 200 миллисекунд. Внутри этого цикла *ошибка* рассчитывается для каждого двигателя отдельно и равна разнице между текущим значением и заданным.

Выражение *l/abs(l)* может быть равно либо *1*, либо *-1*, в зависимости от того, положительное число или нет (функция *abs (l)* – возвращает модуль аргумента).

Пример 36:

```
void moving (int V, int l, int n)
    {
    nMotorEncoder[port1]=0;
    nMotorEncoder[port10]=0;
    while (nMotorEncoder[port1]*(l/abs(l))< 0.8*l*(l/abs(l)))
        {
        motor [port1]=abs (V)*(l/abs(l));
        motor [port10]=3*(n*nMotorEncoder[port1]+nMotorEncoder[port10]);
        wait1Msec(1);
        }
    ClearTimer(T1);
    while (time1[T1]<400)
        {
        motor [port10]=3*(l*n+nMotorEncoder[port10]);
        motor [port1]=3*(l-nMotorEncoder[port1]);
        wait1Msec(1);
        }
    motor [port1]=0;
    motor [port10]=0;
    }

task main()
    {
    moving (50,-500, -1);
    }
```

На занятиях по робототехнике имеет смысл сначала создать 4 программы для езды вперед-назад, разворота влево-вправо, а затем разобрать с учащимися способ объединить все эти программы в одну, как это сделано в Примере 36. Имеет смысл также по отдельности рассмотреть работу первого и второго циклов.

Для заметок

Езда по линии на двух датчиках освещенности с использованием пропорционального регулятора

ЗАДАЧА: Создать робота, который будет передвигаться по черной линии, применяя два датчика *Line Tracker*, используя пропорциональный регулятор.

В параграфе *Езда по линии на одном датчике и вдоль стены на пропорциональном регуляторе* уже было осуществлено использование пропорционального регулятора для езды по черной линии. Однако такое решение имело множество недостатков. Робот неадекватно вел себя на перекрестках. В случае если робот покидал линию, он просто начинал вращаться. Для повышения стабильности и скорости движения необходимо использовать два датчика *Line Tracker*. *Ошибка* в данном случае это разница в показаниях датчиков, если она равна *0*, робот движется прямолинейно.

Пример 37:

```
void line2p (int V, int kp)
    {
    int upp=kp*(SensorValue[in2]-SensorValue[in1])/100;
    motor[port1]=V+upp;
    motor[port10]=V-upp;
    wait1Msec(2);
    }
task main()
while (1)
    {
    line2p(40, 2);
    }
}
```

Используя код из Примера 37, робот будет бесконечно двигаться по линии. Однако, меняя условия выполнения цикла, возможно выполнять серии сложных действий. Например, роботу необходимо передвигаться так, как это показано на Рисунке 18.

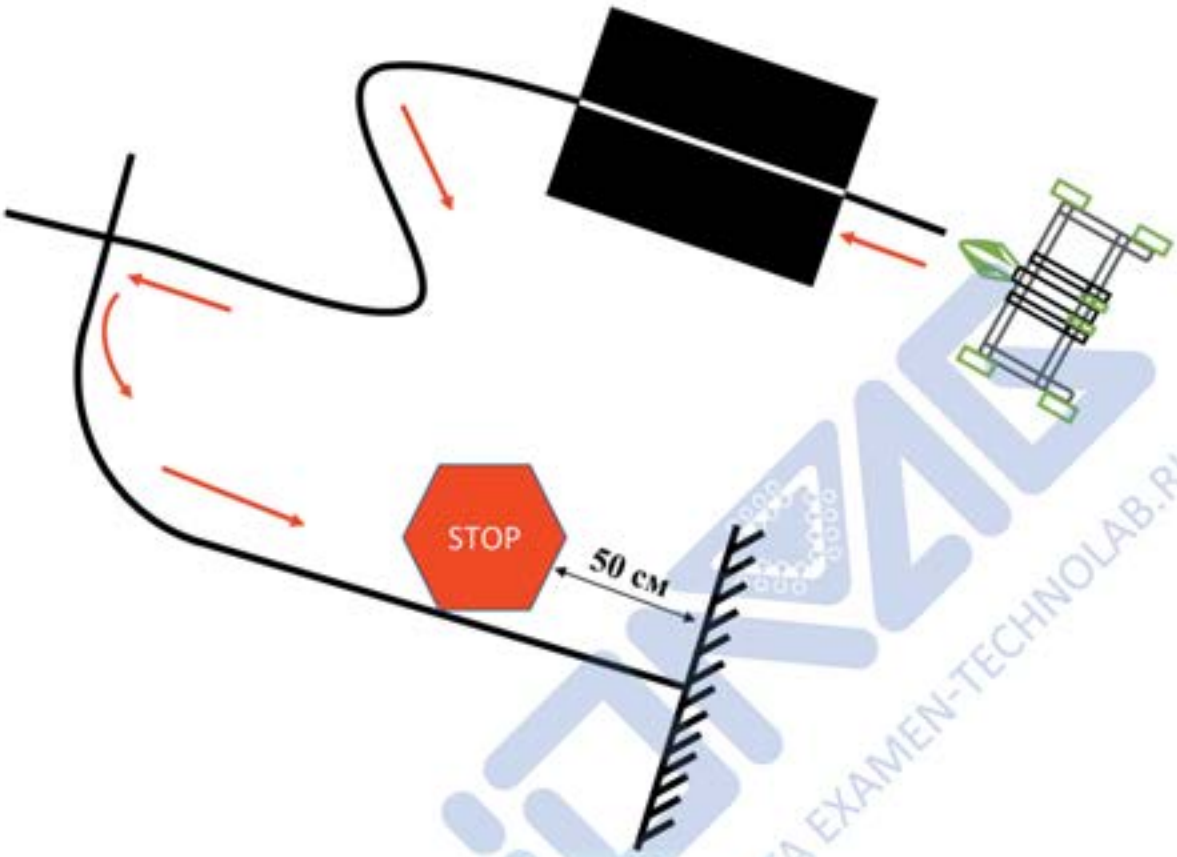


Рисунок 18

Разобьем эту задачу на части:

1. Движение по черной линии на белом фоне;
2. Движение по белой линии на черном фоне;
3. Движение по участку с крутыми поворотами до перекрестка;
4. Разворот на перекрестке против часовой стрелки;
5. Езда до стены.

Проиллюстрируем решение этой задачи с использованием функции *line2p (int V, int kp)*:

1.

```
while ((SensorValue[in2]<1188)||( SensorValue[in1]<1188))
      {
      line2p (60, 2);
      }
```

2.

```
while ((SensorValue[in2]>1188)||( SensorValue[in1]>1188))
    {
    line2p (60, -2);
    }
```

Обратите внимание на то, что для езды на черной линии по белому фону необходимо только изменить знак у коэффициента *kpp*.

3.

```
while ((SensorValue[in2]<1188)||( SensorValue[in1]<1188))
    {
    line2p (60, 4);
    }
```

Для того чтобы преодолеть крутые повороты, необходимо увеличить коэффициент усиления *kpp*.

4. В Примере 33 подробно описано, как осуществить этот маневр.

5.

```
while (SensorValue[dgtl]<50)
    {
    line2p (60, 2);
    }
```

Для заметок

Движение по линии с использованием пропорционально-кубического регулятора

В прошлом параграфе для того чтобы преодолеть крутые повороты, пришлось увеличивать усиливающий коэффициент *kpp*. В этом и следующем параграфе будет показано, как сделать более универсальный регулятор, который не позволил бы роботу «соскочить» с линии при крутых поворотах и при этом не вводил бы его в состояние перерегулирования на относительно прямых участках. В пропорционально-кубическом регуляторе к *пропорциональной* составляющей *ошибки* добавляется еще и *кубическая* составляющая.

```
err=(sensorValue[in2]- sensorValue[in1])
upk=kp*err+kk*err*err*err
```

Коэффициент при кубической составляющей *kk* подбирается таким образом, чтобы эта составляющая минимально влияла на поведение робота на прямых участках и существенно влияла на крутых. Для этого пропорциональный коэффициент должен быть значительно выше кубического коэффициента. При этом есть возможность уменьшить коэффициент *kp*, что позволяет снизить осцилляции робота на прямых участках.

Таблица 6. Зависимость кубической и пропорциональной составляющих от разницы показаний датчиков Line Tracker

Левый датчик	Правый датчик	Ошибка err	err^3	$kk*err^3$	kp*err	upk
2480	143	2337	12763686753	255,2737351	233,70	488,97
1500	143	1357	2498846293	49,97692586	135,70	185,68
1000	143	857	629422793	12,58845586	85,70	98,29
500	143	357	45499293	0,90998586	35,70	36,61
250	143	107	1225043	0,02450086	10,70	10,72
220	143	77	456533	0,00913066	7,70	7,71
200	143	57	185193	0,00370386	5,70	5,70
180	143	37	50653	0,00101306	3,70	3,70
160	143	17	4913	0,00009826	1,70	1,70
150	143	7	343	0,00000686	0,70	0,70

В Таблице 6 видно, что при небольшой ошибке доминирующей является пропорциональная составляющая, тогда как при экстремальных значениях *кубическая* составляющая значительно превосходит пропорциональную. Таблица построена для коэффициентов из Примера 38 kp=.

Пример 38:

```
void line2p (int V, float kp, float kk)
    {
    int err= (SensorValue[in2]-SensorValue[in1]);
    int upk=kp*err+kk*err*err*err;
    motor[port1]=V+upk;
    motor[port10]=V-upk;
    //wait1Msec(2);
    }
task main()
    {
    while (1)
        {
        line2p (40, 0.1, 0.000002);
        }
```

Движение по линии с использованием пропорционально-дифференциального регулятора

Пропорционально-дифференциальный регулятор основан на принципиально другом подходе, чем регулятор в предыдущем параграфе. Основная идея его в том, что коэффициент усиления *kpp* при пропорциональной составляющей подбирается таким образом, чтобы гарантированно пройти все участки трассы. Очевидно, при этом на прямых участках трассы будут возникать излишние осцилляции робота. Чтобы этого избежать, необходимо контролировать не саму *ошибку*, а *изменение ошибки* (дифференциал по времени). Если это изменение будет слишком резким, то *дифференциальная составляющая регулятора* (та составляющая, которая зависит от изменения ошибки) должна противодействовать такому изменению. Следовательно, пропорциональная и дифференциальная составляющие имеют разные знаки.

Пример 39:

```
task main()
    {
    int V=40;
    float kp=0.14;
    float kd=0.2;
    int err;
    int errold=0;
    while (1)
        {
        err=(SensorValue[in2]-SensorValue[in1]);

        motor[port1]=V+(kp*err+kd*(err-errold));   /* err-errold – это изменение ошибки */

        motor[port10]=V-(kp*err+kd*(err-errold));
        errold=err;
        }
    }
```

На этом заканчивается изложение основных принципов теории автоматического управления. Используя релейный, пропорциональный, пропорционально-кубический и пропорционально-дифференциальный регуляторы, а также комбинации этих регуляторов, возможно решать практически все задачи, связанные с автономным поведением робота.

Для заметок

Пульт управления роботом

ЗАДАЧА: Создать робота, который будет управляться с пульта дистанционного управления.

Для осуществления управления с пульта робота VEX необходимо сконфигурировать его для связи с *Joystick*. Для этого необходимо перейти во вложенное меню *Robot→VEX Cortex Communication Mode* и выбрать пункт *VEXNet or USB*.

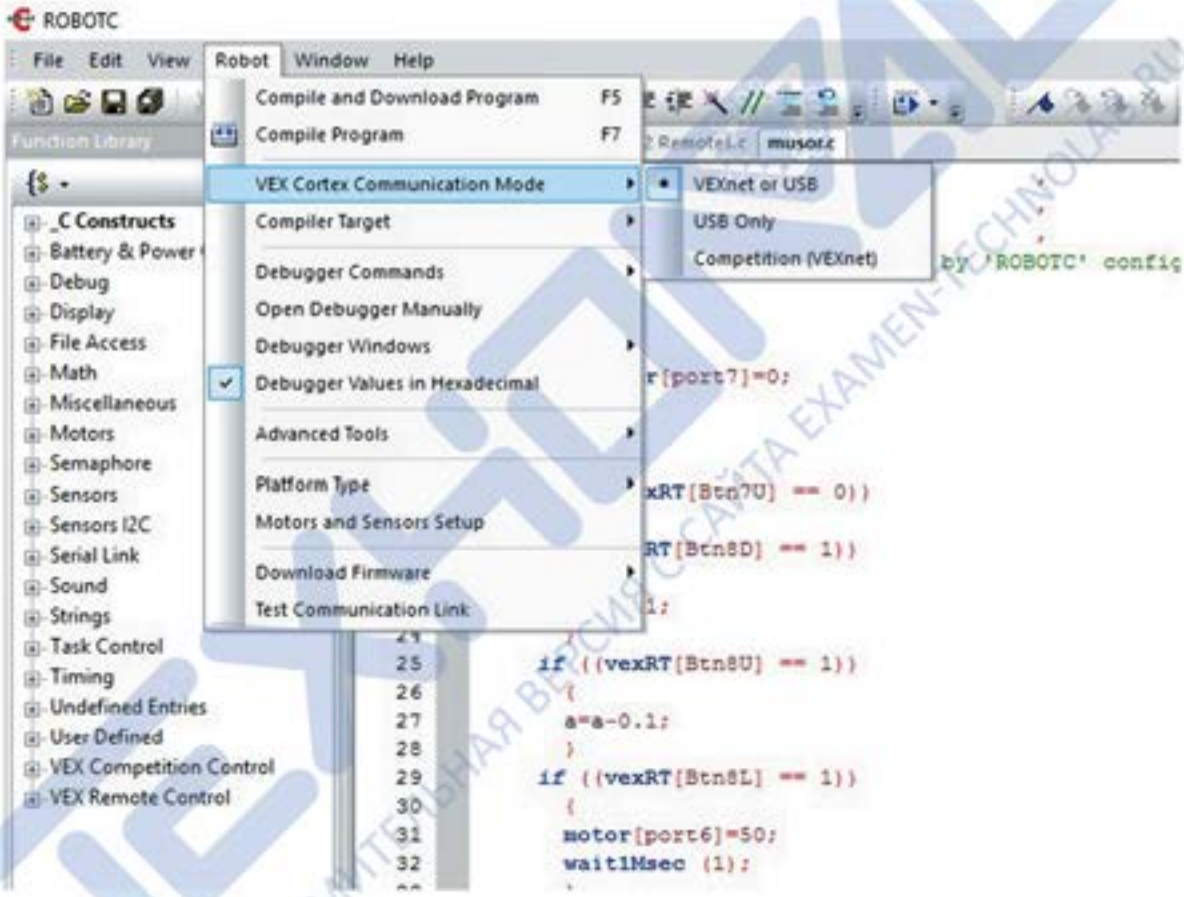


Рисунок 19. Конфигурирование робота Vex для связи с Joystick Vex

Данные, полученные с *Joystick*, записываются в массив *vexRT[]*.

Рисунок 20. Joystick VEX

Рассмотрим один из элементов этого массива – *vexRT[Btn8U]*. *Btn* означает, что в данный элемент записано значение кнопки (*0* или *1*), 8 означает восьмую крестовину, U – кнопка на 8-й крестовине. Эта кнопка на Рисунке 20 обведена розовой окружностью. Данные со стиков могут принимать значения от -127 до 127. Например, для характеристики вертикального перемещения правого стика на Рисунке 20 (зеленая полоса) используется следующая запись – vexRT[Ch2].

В Примере 40 осуществляется управление роботом с пульта дистанционного управления. За все перемещения робота отвечает левый стик. Его вертикальные отклонения – за скорость, горизонтальные – за разворот. В случае нажатия на верхнюю кнопку седьмой крестовины робот прекращает все действия (это элемент безопасности). Подъемное устройство работает на пропорциональном регуляторе. Верхняя и нижняя кнопки на восьмой крестовине задают угол, на который должно быть отклонено подъемное устройство манипулятора. Левая и правая кнопки восьмой крестовины отвечают за манипулятор.

Пример 38:

```
task main()
    {
    nMotorEncoder[port7]=0;
    float a=0;
    while (1)
        {
        while ((vexRT[Btn7U] == 0))
            {
            if ((vexRT[Btn8D] == 1))
                {
                a=a+0.1;
                }
            if ((vexRT[Btn8U] == 1))
                {
                a=a-0.1;
                }
            if ((vexRT[Btn8L] == 1))
                {
                motor[port6]=50;
                wait1Msec (1);
                }
            if ((vexRT[Btn8R] == 1))
                {
                motor[port6]=-50;
                wait1Msec (1);
                }
            motor[port7]=1*(a-nMotorEncoder[port7]);
            motor[port1]=vexRT[Ch3]+vexRT[Ch4];
            motor[port10]=vexRT[Ch3]-vexRT[Ch4];
            motor[port6]=0;
            }
        wait1Msec(2);
        motor[port1]=0;
        motor[port10]=0;
        }
    }
```

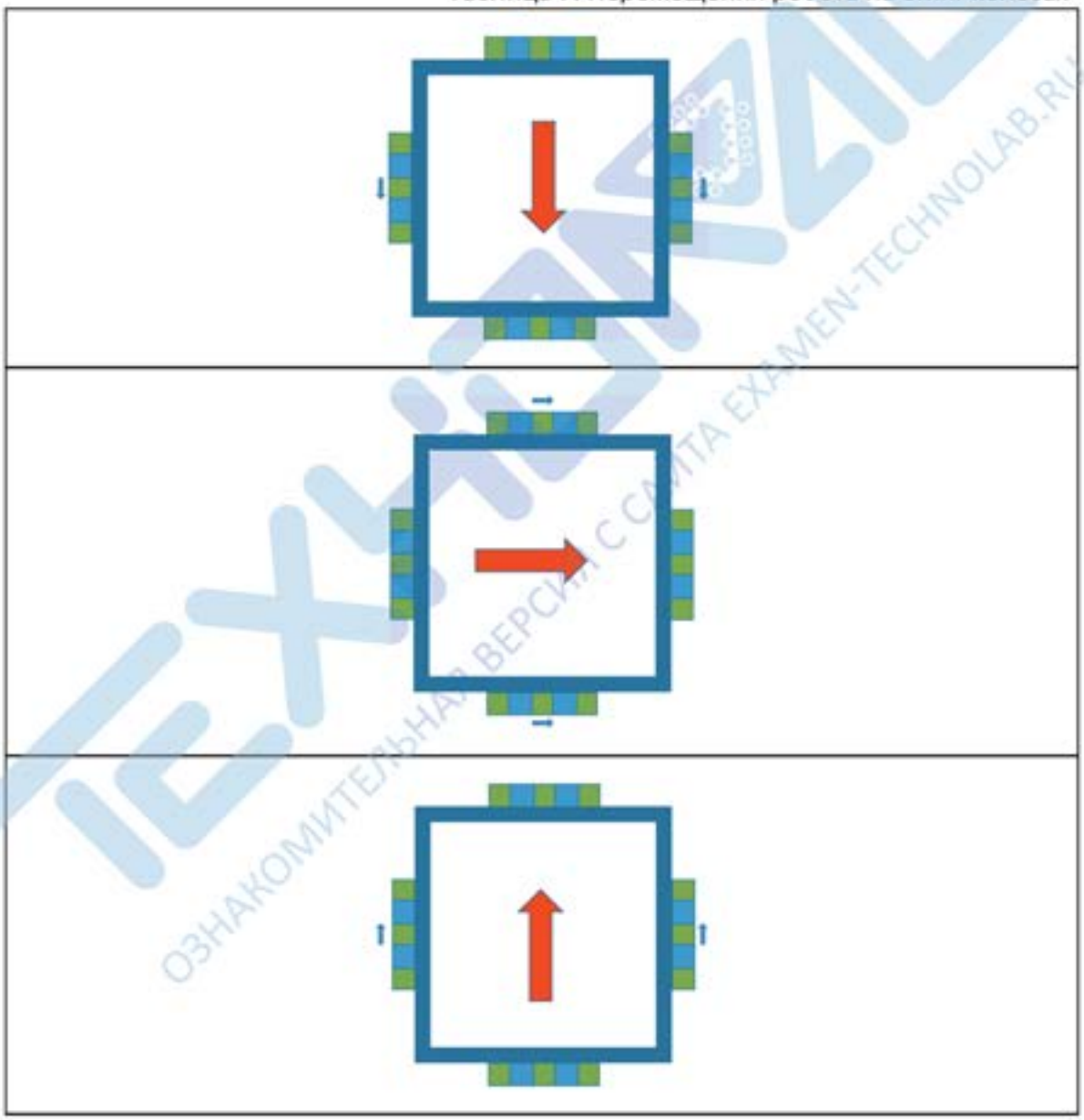
Для заметок

Управление движениями робота на omni-колесах

Конструкция робота на *omni-колесах* представлена на Рисунке 4 и Рисунке 5.

Возможные варианты движений робота представлены в Таблице 7. Синими стрелками показаны направления движения колес, красными – всего робота.

Таблица 7. Перемещения робота на omni-колесах



Управление роботом на omni-колесах с пульта управления

Осуществим управление роботом на *omni-колесах* с пульта управления. В Примере 41 робот будет выполнять команды с пульта, пока не будет нажата кнопка *7U*. За поступательное движение отвечает левый джойстик, за вращательное – правый.

Пример 41:

```
task main()
    {
    nMotorEncoder[port7]=0;
    float a=0;
    while (1)
        {
        while ((vexRT[Btn7U] == 0))
            {
            motor[port1]=vexRT[Ch4]-vexRT[Ch1];
            motor[port10]=vexRT[Ch4]+vexRT[Ch1];
            motor[port2]=vexRT[Ch3]+vexRT[Ch1];
            motor[port9]=vexRT[Ch3]-vexRT[Ch1];
            }
        wait1Msec(1);
        }
    }
```

Для заметок

Послесловие

Дорогие друзья! Я надеюсь, что примеры, разобранные в учебном пособии, станут опорой для Ваших занятий и творческих проектов Ваших подопечных. Прошу Вас относиться к пособию как к теоретической основе для создания новых роботов. За рамками пособия осталась интегральная составляющая ПИД-регулятора. А также работа с несколькими датчиками – акселерометром, компасом и пр. Принципы, положенные в основу работы с ними, те же, что и с датчиками, описанными выше.

Удачи Вам и успехов в робототехнике!

Для заметок

VEX Robotics Competition Starstruck - Регламент соревнований

Часть 1 – Введение

Обзор

В этой части представлена обзорная информация по робототехническим соревнованиям VEX Robotics Competition и VRC Starstruck, которые проходят в 2016/2017 учебном году.

Соревнования VEX Robotics Competition

Сегодня мир нуждается в студентах, которые станут учеными и инженерами будущего, решающими глобальные проблемы. Непрекращающиеся открытия в области химии, медицины, новых материалов и физики ставят новые задачи и создают еще больше возможностей для решения проблем с помощью новых технологий. Именно с помощью новых технологий будут решены наиболее существенные проблемы человечества.

Сегодня недостаточное количество выпускников школы выбирает инженерно-технические специальности при поступлении в ВУЗ. Это происходит не из-за недостатка свободных учебных мест, а из-за недостатка заинтересованных и подготовленных абитуриентов. Иными словами, если мы не восполним этот недостаток сегодня, то у следующего поколения просто не будет людей, способных решать проблемы будущего. Кто выведет мир из следующего глобального кризиса?

Осознавая эту проблему, многие организации создают программы, направленные на привлечение и вовлечение учащихся в процесс изучения естественнонаучных и инженерно-технических предметов. Многие из них обнаружили, что робототехника является отличной платформой для привлечения и удержания внимания современной молодежи. Робототехника способна заинтересовать это поколение, живущее в условиях сильной конкуренции, и представляет собой идеальную комбинацию прикладной физики, математики, программирования, проектирования и прототипирования, командной работы и лидерства. Количество учащихся, проявивших интерес к STEM-дисциплинам, увеличивается благодаря усилиям школ, волонтерских организаций, кампаний и правительств по всему миру.

Соревнования VEX Robotics Competition, организуемые Фондом Робототехнического Образования и Соревнований (REC Foundation), воодушевляют тысячи учащихся по всему миру на выбор естественнонаучных и инженерно-технических специальностей. Несмотря на то, что существует много качественных робототехнических соревнований, сообщество пользователей VEX Robotics предлагает наиболее простые и доступные решения для организации соревнований.

Система VEX EDR переводит вдохновение, полученное на соревнованиях, на новый

уровень. Система используется как робототехническая платформа для класса, спроектированная для развития инженерно-технического творчества и STEM-дисциплин. VEX предоставляет учителям и студентам доступную, надежную и современную робототехническую платформу, подходящую для использования в классе, а также игровое поле. VEX использует стальные структурные компоненты, интуитивно понятные механические элементы в комбинации с набором программируемых микроконтроллеров – все это дает безграничные конструкторские возможности.

Для получения дополнительной информации по робототехнической платформе VEX Robotics посетите сайт vex.examen-technolab.ru или http://www.vexrobotics.com/.

Присоединяйтесь к нам в Twitter @VEXRobotics. Жмите «нравится» на Facebook www.facebook.com/vexrobotics.

Для получения дополнительной информации о Фонде Робототехнического Образования и Соревнований посетите сайт www.roboticseducation.org.

Присоединяйтесь к нам в Twitter @REC_Foundation. Жмите «нравится» на Facebook www.facebook.com/RECFoundation.

Для получения дополнительной информации о соревнованиях VEX Robotics Competition, включая информацию о регистрации команд, списки событий и результаты соревнований, посетите сайт www.robotevents.com.

VEX Robotics Competition Starstruck: Введение

Соревнования *VEX Robotics Competition Starstruck* проводятся на специальном сборном поле, состоящем из мягкого основания размером 3,7 x 3,7 м с ограждением по периметру из листового металла и прозрачного пластика повышенной прочности. На поле располагается 32 *звезды* и *куба*, за перемещение которых в *зачетные зоны* присуждаются очки; также очки можно получить за *зацепление* на зачетном столбе на разной высоте.

Более подробно характеристики игрового поля и правила игры описаны в части 2 - Соревнования.

В процессе участия в соревнованиях *VEX Robotics Competition Starstruck* команды приобретут множество навыков, выполняя задания и преодолевая препятствия на своем пути. Одни задачи будут решены участниками индивидуально, другие же с помощью товарищей по команде и взрослых наставников. Команды будут работать вместе с целью конструирования робота для участия в одном из многих турниров, в ходе которого они разделят радость своих достижений вместе с другими командами, семьей и друзьями. После окончания соревнований участники возвращаются домой не только с мыслями о том, что им удалось создать собственного соревновательного робота, но и с осознанием того, как они могут использовать науку и технологии для улучшения окружающего мира. Более того, они приобретают такие жизненно важные навыки, как планирование, творчество, сотрудничество, командная работа и лидерство, а также навыки исследования и конструирования.

Часть 2 - Соревнования

Обзор

В этой части описываются соревнования VEX Robotics Competition, которые называются *VEX Robotics Competition Starstruck*. Здесь также перечисляются соревновательные термины и правила.

Описание соревнований

Матчи проводятся на поле, установленном, как показано на картинке ниже. В каждом *матче* соревнуется два *альянса* – один красный, другой синий, каждый из которых состоит из двух команд. В соревновании выигрывает *альянс*, набравший наибольшее количество *очков*, полученных за перемещение *звезд* и *кубов* в *зачетные зоны*, а также за *зацепление* робота на *зачетном столбе*.

Бонусные баллы присуждаются альянсу, набравшему наибольшее количество баллов за *период автономного управления*.

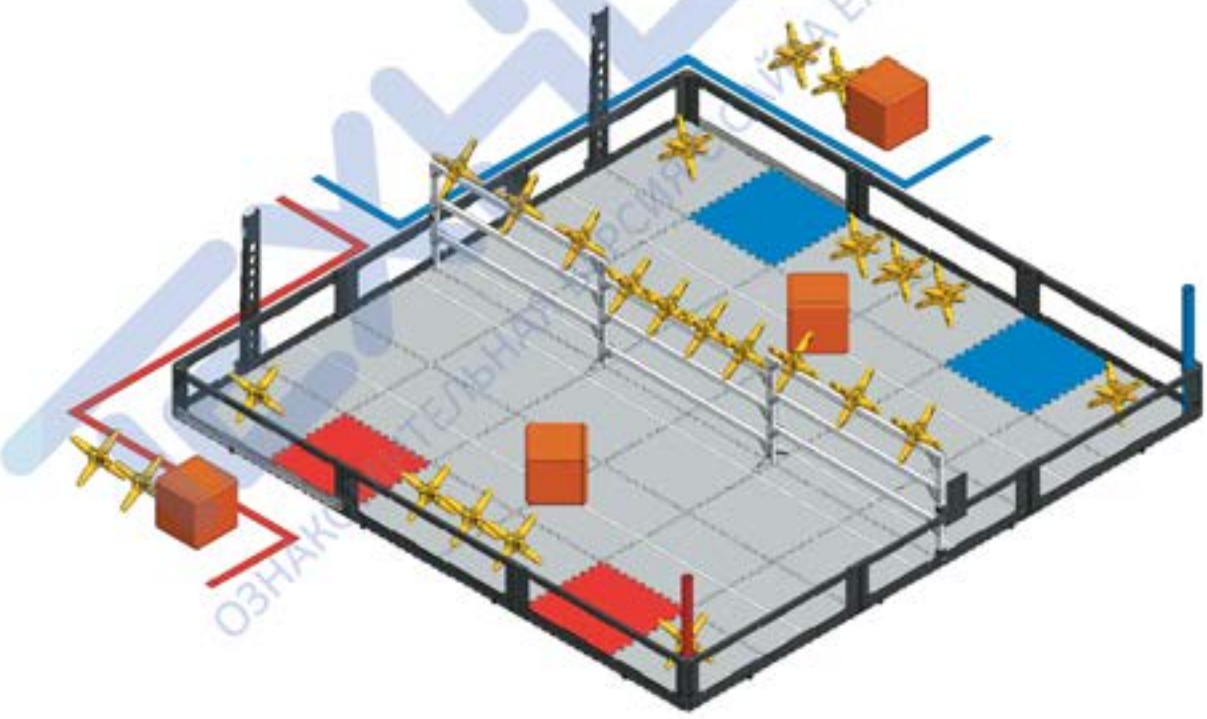
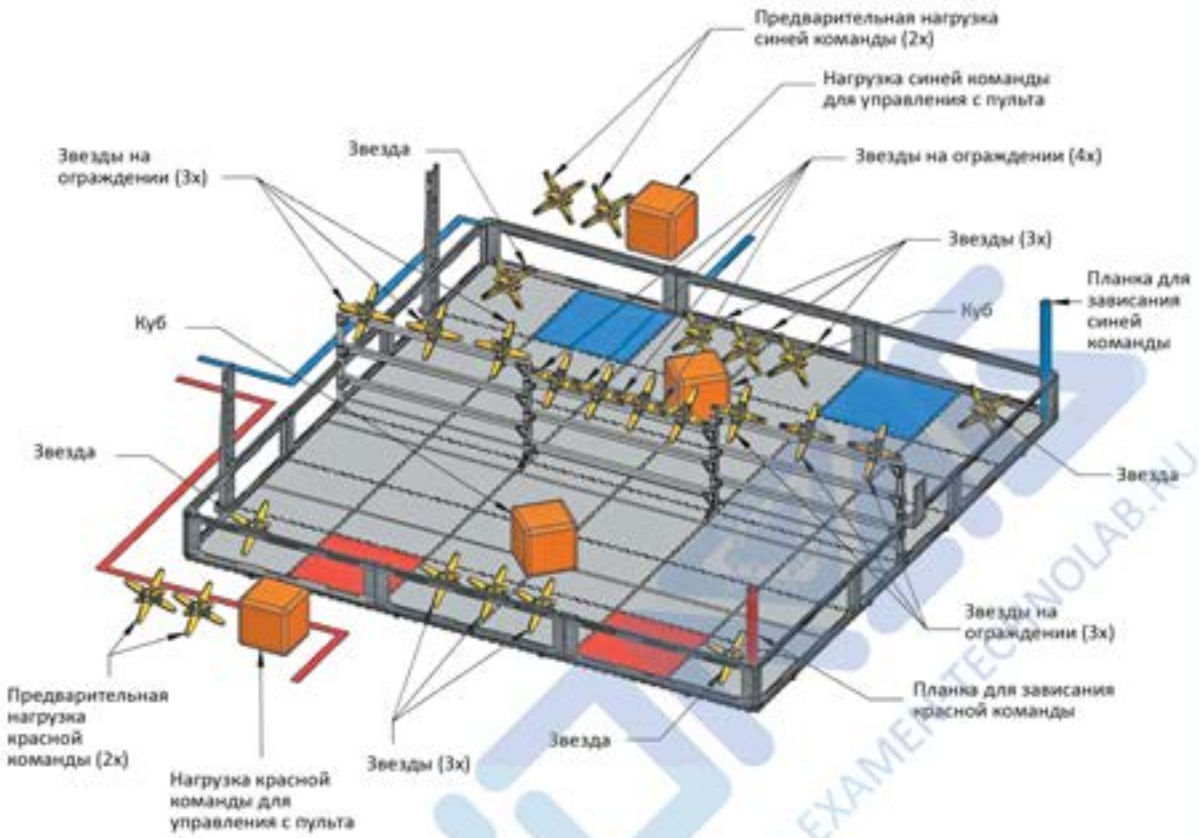


Рис. 1: *Изометрическая проекция поля*

Примечание: *данная иллюстрация лишь дает представление об общем виде поля и игровых объектов. Перед соревнованиями участникам необходимо ознакомиться с официальными размерами поля и игровых объектов.*

В каждом матче соревнований *VEX Robotics Competition Starstruck* используется двадцать восемь (28) *зачетных элементов*: двадцать четыре (24) *звезды* и четыре (4) *куба*. Перед началом матча на каждого *робота* устанавливается одна (1) *звезда*. В течение последних тридцати (30) секунд *матча* каждый *альянс* сможет загрузить на *робота* один (1) *куб*. Двадцать (20) *звезд* и два (2) *куба* устанавливаются в определенных местах на поле. Каждый *альянс* имеет доступ к одному (1) *зачетному столбу*, на котором может *зацепиться* один (1) *робот*.

Обозначения в международном формате:
Blue Alliance Preloads (2X) – загружаются перед началом матча синим альянсом, 2 шт
Blue Alliance Driver Control Load – загружается в период ручного управления синим альянсом
Stars on Fence (4X) – звезды на перегородке, 4 шт
Stars (3X) – звезды, 3 шт
Cube – куб
Blue Alliance Hanging Bar – зачетный столб синего альянса
Star – звезда
Stars on Fence (3X) – звезды на перегородке, 3 шт
Red Alliance Hanging Bar – зачетный столб красного альянса
Red Alliance Preloads (2X) – загружаются перед началом матча красным альянсом, 2 шт
Red Alliance Driver Control Load – загружается в период ручного управления красным альянсом
Red Alliance Station – станция красного альянса
Blue Alliance Station – станция синего альянса
Blue Alliance Starting Tiles – стартовые позиции синего альянса
Red Alliance Far Zone – дальняя зачетная зона красного альянса
Red Alliance Near Zone – ближняя зачетная зона красного альянса
Neutral Zone – нейтральная зона
Blue Alliance Far Zone – дальняя зачетная зона синего альянса
Blue Alliance Near Zone – ближняя зачетная зона синего альянса
Red Alliance Starting Tiles – стартовые позиции красного альянса



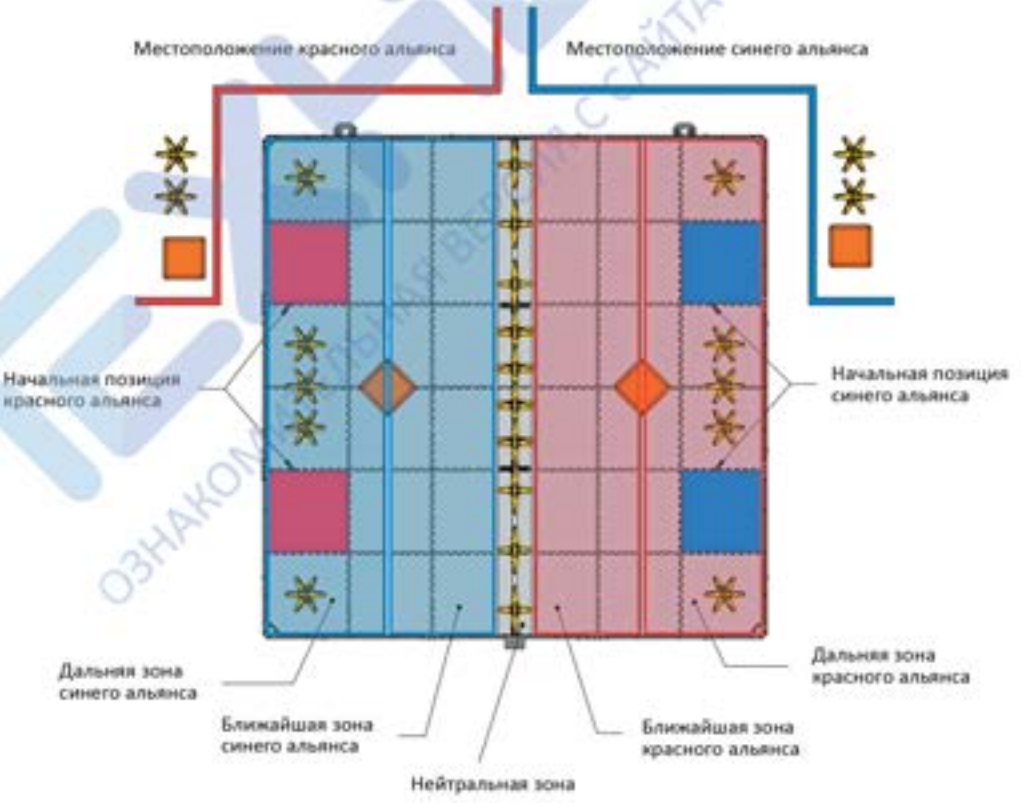


Рис. 2 и 3: *Проекция поля*

Соревновательные термины

Взрослый участник – любой человек, не подпадающий под определение *«студент»*.

Альянс – предварительно назначенная группа из двух команд, сотрудничающих друг с другом в ходе конкретного *матча*.

Стартовая панель альянса – цветная напольная панель (красная или синяя), определяющая расположение *роботов* на момент начала матча.

Станция альянса – выделенная область, где *участники команды операторов* должны находиться в ходе *матча*.

Период автономного управления – период времени длительностью 15 секунд (0:15) в начале матча, в течение которого осуществляется автономное управление *роботами* на основании данных, получаемых от датчиков, а также команд, предварительно загруженных в систему управления *робота*.

Автономный бонус – дополнительные очки, присужденные альянсу, набравшему наибольшее количество очков в ходе периода *автономного управления*.

Куб – зачетный элемент кубической формы, изготовленный из ткани с мягким наполнителем. Длина стороны примерно 31,75см ± 2,54см, вес каждого куба 0,76кг ± 15%.

Выведение из игры – штраф, назначаемый команде за нарушение правил. Команда, *выведенная из игры* в ходе *матча*, в дальнейшем не может быть допущена к управлению своим роботом и должна положить устройства управления на землю.

Дисквалификация – штраф, назначаемый команде за грубое нарушение правил. Команда, получившая *дисквалификацию* в ходе *квалификационного матча*, получает (0) *WP* и *SP*. Если команда была *дисквалифицирована* в *ходе матчей на вылет*, весь *альянс* получает *дисквалификацию*, и ему присуждается поражение в *матче*. Повторные грубые нарушения правил и *дисквалификации* одной команды могут привести к *дисквалификации* всего турнира по решению главного судьи. Для получения более подробной информации и ознакомления с сопряженными понятиями, пожалуйста, изучите часть 3.

Участник команды операторов – любой из трех (3) членов каждой команды, которые могут находиться на *станции альянса* в ходе *матча*. Только *студенты – участники команды операторов* могут прикасаться к устройству управления в ходе матча, осуществлять взаимодействие с роботом согласно <SG3> и *зачетными объектами* согласно <SG4>. *Взрослый участник* не может входить в состав *команды операторов*.

Нагрузка для периода ручного управления – два (2) куба, один (1) для каждого *альянса*, которые *участники команды операторов* каждого *альянса* могут установить на *стартовую панель альянса* или на *робота*, касающегося *стартовой панели альянса*. *Нагрузка для периода ручного управления* должна быть установлена не ранее чем за тридцать (0:30) секунд до окончания *матча*.

Период ручного управления – период длительностью 1 минута 45 секунд (1:45), когда *студенты – участники команды операторов* осуществляют управление *роботами*.

Зацепление – робот зацепился с роботом-оппонентом, если он схватил или зацепил последнего.

Дальняя зачетная зона - одна из двух (2) зачетных зон, одна (1) для каждого альянса, в которых команды могут получить очки за *зачетные элементы*. *Дальние зачетные зоны* ограничены внутренними гранями ограждения поля и полосой на напольных панелях. *Дальняя зачетная зона альянса* располагается на другой стороне от перегородки относительно *станции альянса*.

Примечание: *зачетные столбы* не являются частью *дальней зачетной зоны*.

Перегородка - конструкция высотой 61 см, изготовленная из ПВХ и разделяющая поле пополам. Также обозначает границу между двумя *ближними зачетными зонами*.

Элемент поля – мягкие панели поля, периметр поля, *перегородка*, а также все вспомогательные конструкции.

Зачетный столб - красный или синий пустотелый цилиндрический столб высотой 76 см, изготовленный из ПВХ и установленный в двух углах поля со стороны зрителей.

Высокое зацепление - робот находится в состоянии *высокого зацепления*, если он касается *зачетного столба* своего цвета и все его части располагаются выше плоскости, находящейся на высоте периметра поля. *Робот* не считается находящимся в состоянии *высокого зацепления*, если он касается периметра поля.

Примечание: *робот* в состоянии *высокого зацепления* не считается *роботом*, находящимся в состоянии *низкого зацепления*. Только один (1) *робот* из *альянса* может получить очки за *зацепление* на столбе (*высокое* или *низкое*) в ходе матча.

Низкое зацепление - *робот* находится в состоянии *низкого зацепления*, если он касается *зачетного столба* своего цвета и ни одна из его частей не касается мягких панелей поля.

Примечание: только один (1) *робот* из *альянса* может получить очки за *зацепление* на столбе (*высокое* или *низкое*) в ходе матча.

Матч – матч общей продолжительностью две минуты (2:00), состоит из *периода автономного управления,* за которым следует *период ручного управления.*

Ближняя зачетная зона - одна (1) из двух (2) зачетных зон, одна (1) для каждого альянса, в которой команды могут получить очки за *зачетные элементы. Ближние зачетные зоны* ограничены внутренними гранями периметра поля и полосой на напольных панелях. *Ближние зачетные зоны альянсов* находятся на другой стороне *перегородки* относительно *станций альянсов.*

Примечание 1: *перегородка* не является частью ни одной *ближней зачетной зоны.*
Примечание 2: полоса на напольных панелях, отделяющая *ближнюю зачетную зону* от *дальней*, считается частью обеих *зачетных зон.* Полоса на напольных панелях, ограничивающая *ближнюю зачетную зону* вдоль *перегородки*, не является частью ни одной *зачетной зоны.*

Предварительная нагрузка - четыре (4) звезды, по одной (1) для каждой команды, установленные на поле перед началом каждого *матча* таким образом, чтобы они касались *робота*, при этом не касаясь ни одной напольной панели и полностью находясь в пределах периметра поля.

Робот – любое электромеханическое устройство, прошедшее контроль и установленное командой на поле до начала *матча.*

Засчитано - зачетный элемент считается *засчитанным* в зоне, если он удовлетворяет одному из следующих требований:

1. *Зачетный элемент касается зачетной зоны*
   a. если *зачетный элемент* касается нескольких *зачетных зон*, он *засчитывается* в *зоне* с наибольшим количеством очков.

2. *Зачетный элемент* не касается ни одной *зачетной зоны* и *поддерживается* роботом и/или *зачетным элементом*, такой *зачетный элемент засчитывается* в *зоне*, которой касается поддерживающий *робот* или *зачетный элемент*
   a. если *поддерживающий робот* и/или *зачетный элемент* касается нескольких *зачетных зон*, то он *засчитывается* в *зоне* с наибольшим количеством очков
   b. если *зачетный элемент поддерживается висящим роботом*, то такой *зачетный элемент засчитывается* в *дальней зачетной зоне,* прилегающей к *зачетному столбу,* на котором *висит робот*

Примечание 1: если *зачетный элемент* поддерживается исключительно *перего-*

родкой, то он не *засчитывается* ни в одной из *зачетных зон*

Примечание 2: *если зачетный элемент поддерживается* одновременно двумя соперничающими *роботами*, то он не *засчитывается* ни в одной из *зачетных зон*

Примечание 3: если *зачетный элемент* касается двух соперничающих *зачетных зон*, то он не *засчитывается* ни в одной из *зачетных зон*

Зачетный элемент – звезда или *куб*

Звезда – желтый *зачетный элемент*, изготовленный из пенополиуретана, представляющий собой шесть (6) лучей, исходящих из одного центра, общим диаметром 35,6 см. Каждая звезда весит 0,272кг ± 15%.

Студент – любой учащийся средней школы основного или заочного отделения. Пригодность может быть также подтверждена потерей трудоспособности с отсрочкой обучения на срок более одного года.

Поддерживается – зачетный элемент считается *поддерживаемым*, если он не сохраняет свое положение без поддерживающего объекта. Судьи определяют факт *поддерживания зачетного элемента* путем аккуратного перемещения поддерживающего объекта, если такое перемещение возможно.

Зачетная зона – ближняя или *дальняя зачетная зона*.

Правила соревнований

Начисление очков

- За *звезду, засчитанную* в *ближней зачетной зоне*, присуждается одно (1) очко *альянсу*, чей цвет совпадает с цветом *ближней зачетной зоны*.
- За *звезду, засчитанную* в *дальней зачетной зоне*, присуждается два (2) очка *альянсу*, чей цвет совпадает с цветом *дальней зачетной зоны*.
- За *куб, засчитанный* в *ближней зачетной зоне*, присуждается два (2) очка *альянсу*, чей цвет совпадает с цветом *ближней зачетной зоны*.
- За *куб, засчитанный* в *дальней зачетной зоне*, присуждается четыре (4) очка *альянсу*, чей цвет совпадает с цветом *дальней зачетной зоны*.
- За *робота*, находящегося в состоянии *низкого зацепления*, *альянсу* присуждается четыре (4) очка.
- За *робота*, находящегося в состоянии *высокого зацепления*, *альянсу* присуждается двенадцать (12) очков.
- В конце *периода автономного управления альянсу*, набравшему наибольшее количество очков, присуждается четыре (4) бонусных очка.

Правила безопасности

<S1> Если в любой момент времени функционирование *робота*, либо действия команды признаны представляющими опасность, либо спровоцировавшими причи-

нение ущерба *элементам поля* или *зачетным элементам*, по решению судьи команда-нарушитель может быть *выведена из игры* или *дисквалифицирована*. При этом *робот-нарушитель* будет подвергнут повторной экспертизе, по результатам которой будет принято решение о его допуске на поле.

a. Команды должны проявлять предельную осторожность при взаимодействии с *зачетными элементами*. Любой ущерб, например, царапины или проколы, расценивается как нарушение <S1>.

<S2> В случае если *робот* полностью выходит за пределы игрового поля, он *выводится из игры* до конца *матча*.

Примечание: При нормальном ходе игры не принято штрафовать *роботов* за наличие на них внешних механизмов, непреднамеренно пересекающих пределы поля при нормальном ходе игры.

Общие правила соревнований

<G1> В процессе изучения и использования различных правил, содержащихся в настоящем документе, необходимо учитывать, что эти правила созданы для игровых соревнований по робототехнике VEX Robotics Competition и должны применяться в разумных пределах.

<G2> На момент начала *матча* габариты каждого *робота* не могут превышать установленные значения Ш x Д x В 18 x 18 x 18 (в дюймах). *Робот-нарушитель* будет удален с поля по решению главного судьи.

<G3> Каждая команда может иметь в составе трех *участников команды операторов*. В рамках любого из мероприятий ни один *участник команды операторов* не может входить в состав более чем одной команды.

<G4> В любой момент *матча* только *студенты – участники команды операторов* могут прикасаться к устройствам управления, *роботу* и *игровым элементам*, и только *участники команды операторов* могут взаимодействовать с *роботом* в соответствии с <SG3>. *Взрослые участники команды операторов* не могут прикасаться к устройствам управления и взаимодействовать с роботом или *зачетными элементами*. За незначительные нарушения этого правила, не влияющие на ход матча, выносится предупреждение. Значительные нарушения (влияющие на ход матча) ведут к *дисквалификации*. Команды, получившие несколько предупреждений, также могут быть *дисквалифицированы* по решению главного судьи.

<G5> В ходе *матча участники команды операторов* должны оставаться на *станции своего альянса*.

<G6> В ходе квалификационных матчей красный *альянс* имеет право последним размещать своих роботов на поле. В ходе матчей на вылет более сильный *альянс* име-

ет право последним размещать своих роботов на поле. Как только команда разместила своего *робота* на поле, до момента начала матча его положение не может быть изменено. *Роботы* должны быть размещены на поле должным образом. Роботы команды, нарушившей данное правило, будут перемещены в пределах поля на другие позиции по усмотрению судей.

<G7> *Участники команды операторов* не могут преднамеренно вступать в контакт с любым из *зачетных элементов*, *элементов поля* или *роботов* в ходе *матча*, за исключением видов контактов согласно <SG3> и <SG4>. За незначительные нарушения этого правила, не влияющие на ход матча, выносится предупреждение. Значительные нарушения (влияющие на ход матча) ведут к *дисквалификации*. Команды, получившие несколько предупреждений, также могут быть *дисквалифицированы* по решению главного судьи.

a. *Участникам команды операторов* запрещено пересекать периметр поля в любой момент в ходе матча, за исключением действий, предусмотренных <SG3> и <SG4>.

<G8> В ходе *матча* управление *роботами* может осуществляться только *студентами – участниками команды операторов* и/или путем использования программного обеспечения, загруженного в интегрированные системы управления. В течение *периода автономного управления участники команды операторов* не могут взаимодействовать с *роботами* или устройствами управления на своих джойстиках VEXnet ни непосредственно, ни опосредованно (напр., бесконтактный запуск датчиков *робота* запрещен). Незначительные нарушения данного правила, не влияющие на исход матча, приводят к получению предупреждения. Значительные нарушения (влияющие на ход матча) ведут к *дисквалификации*. Команды, получившие несколько предупреждений, также могут быть *дисквалифицированы* по решению главного судьи.

<G9> Предполагается, что *зачетные элементы* могут непреднамеренно покидать пределы поля в ходе матча. *Зачетные элементы*, покинувшие игровое поле, не могут быть возвращены на поле. Командам запрещается преднамеренно удалять *зачетные элементы* с поля. Предполагается, что *зачетные элементы* могут непреднамеренно покидать пределы поля при попытках роботов забить их в *ворота*. Тем не менее преднамеренное либо повторное их удаление будет считаться нарушением данного правила. За незначительные нарушения этого правила, не влияющие на ход матча, выносится предупреждение. Значительные нарушения (влияющие на ход матча) ведут к *дисквалификации*. Команды, получившие несколько предупреждений, также могут быть *дисквалифицированы* по решению главного судьи.

<G10> Счет *матча* рассчитывается сразу после его завершения и после того, как все объекты и *роботы* на поле перестали перемещаться.

<G11> В ходе матча никакие детали и механизмы не могут быть отделены от *робота* либо оставлены им на поле. За незначительные нарушения этого правила, не влияющие на ход матча, выносится предупреждение. Значительные нарушения (влияющие на ход матча) ведут к *дисквалификации*. Команды, получившие несколько

предупреждений, также могут быть *дисквалифицированы* по решению главного судьи. Множественные преднамеренные нарушения правил игры могут привести к *дисквалификации* на весь период соревнований.

<G12> Стратегии, направленные исключительно на разрушение, повреждение, опрокидывание либо *зацепление роботов*, противоречат духу робототехнических соревнований VEX Robotics Competition и являются запрещенными к использованию. Тем не менее соревнования VRC Starstruck являются интерактивными. Непреднамеренное опрокидывание, *зацепление* и повреждение может иметь место при нормальном ходе игры. В случае когда опрокидывание, *зацепление* или повреждение признано преднамеренным либо грубым, команда-нарушитель может получить дисквалификацию до конца *матча*. Повторные нарушения могут привести к *дисквалификации* команды до конца соревнований.

Соревнования *VRC Starstruck* являются соревнованиями с элементами нападения. Команды, использующие в ходе соревнований только защитные стратегии, будут подвержены более внимательному контролю в соответствии с <G12>. В этом случае право судить спорные моменты взаимодействия нападающего и защищающегося роботов принадлежит судьям, причем решение будет вынесено в пользу первого.

Команда несет ответственность за действия своего *робота* в любой момент времени, включая *период автономного управления*. Это относится к командам, чье управление роботами может быть потенциально опасным или экстремальным, а также к командам, чьи роботы имеют маленькую колесную базу. Команда должна проектировать своего *робота* таким образом, чтобы его нельзя было легко перевернуть либо повредить путем незначительного контакта.

<G13> *Роботы* должны быть спроектированы таким образом, чтобы *зачетные элементы* могли быть легко удалены из их захватывающих механизмов по окончании матча без необходимости подачи питания к роботу.

<G14> Допуски для габаритов поля находятся в пределах ±2,5 см, кроме случаев, когда предусмотрено иное, что должно быть учтено командами при проектировании *роботов*. Пожалуйста, убедитесь, что вы изучили Приложение А, содержащее информацию обо всех допусках.

Примечание: напольные панели полностью находятся в пределах периметра поля. Периметр поля не должен быть установлен сверху на напольных панелях.

<G15> Решение о повторе игры находится в компетенции партнера мероприятия и главного судьи и может быть принято только в особых случаях.

<G16> Все команды должны следовать правилам робототехнических соревнований VEX Robotics Competition в том виде, в котором они изложены в этом регламенте, а

также подчиняться намерениям, заявленным в правилах. Каждая команда имеет возможность запросить официальную интерпретацию правил на форуме вопросов и ответов робототехнических соревнований VEX Robotics Competition Question & Answer Forum. Все ответы, полученные на этом форуме, являются официальной версией правил, представленной комитетом по игровым проектам для робототехнических соревнований VEX Robotics Competition, и представляют собой точные интерпретации правил робототехнических соревнований VEX Robotics.

Периодические «Новости команд» также публикуются на странице соревнований VRC Starstruck в разделе соревнований на сайтах www.vexrobotics.com и www.roboticseducation.org. Эти обновления также являются официальной частью правил соревнований VRC Starstruck.

Форум вопросов и ответов робототехнических соревнований VEX Robotics Competition доступен по адресу www.vexforum.com либо по прямой ссылке http://www.vexrobotics.com/Starstruck.

<G17> Предполагается, что в ходе мероприятий, проводимых в рамках соревнований VEX Robotics Competition, все команды будут следовать принципам взаимоуважения и профессионализма. В случае если команда, либо любой участник команды проявляет неуважение, либо грубость по отношению к сотрудникам мероприятия, волонтерам, либо участникам других команд, нарушители могут быть *дисквалифицированы* на период текущего и предстоящего *матчей*. Важно помнить, что нас судят по тому, как мы ведем себя в момент бедствия. В сложных ситуациях, возникающих как на робототехнических соревнованиях VEX Robotics Competition, так и в повседневной жизни, важно проявлять зрелость и профессионализм.

<G18> Все правила, изложенные в настоящем регламенте, подлежат изменениям и не считаются официальными до 17 августа 2016 года. Предполагается, что значительные изменения в них вноситься не будут, однако мы оставляем за собой право вносить в них любые поправки до 17 августа 2016 года. Для настоящего регламента также запланированы два обновления: 15 июня 2016 года и 3 апреля 2017 года.

Особые правила соревнований VRC Starstruck

<SG1> На момент начала *матча* каждый *робот* должен быть размещен таким образом, чтобы он касался одной из цветных *стартовых панелей альянса* и не касался ни одного *зачетного объекта*, если иное не предусмотрено <SG2>, а также других панелей поля или роботов. На одной (1) *стартовой панели альянса* на момент начала матча может находиться только один (1) *робот*.

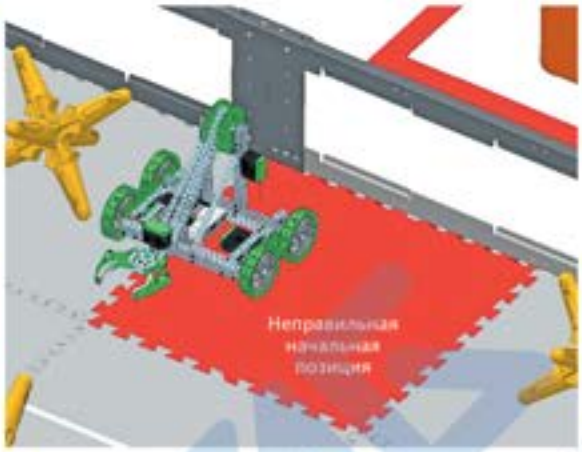
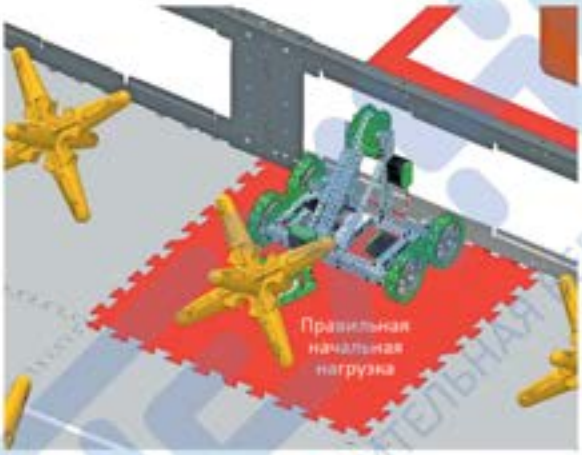


***Рис. 4 и 5:** Правильная и неправильная начальные позиции*

<SG2> Перед началом каждого *матча* на каждого *робота* должна быть установлена одна (1) *звезда* в качестве *предварительной нагрузки*. *Звезда* считается установленной правильно, если она касается *робота*, но при этом не касается ни одной напольной панели и полностью находится внутри периметра поля (см. рис. 8 и 9). Если *робот* по какой-либо причине не участвует в *матче*, то предназначавшаяся для него *звезда* устанавливается в любом месте на *стартовой панели альянса*.



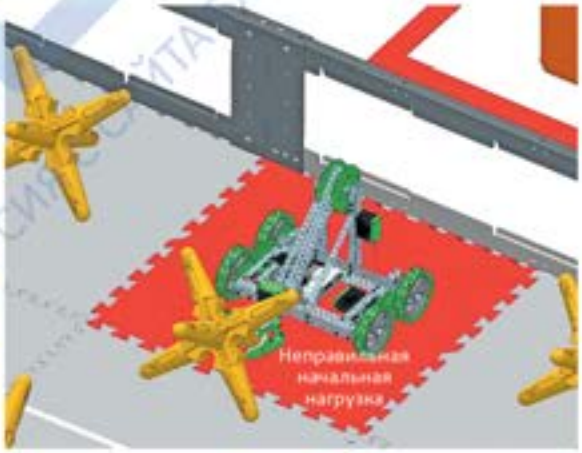


***Рис. 6 и 7:** Правильная и неправильная начальные нагрузки*

<SG3> В течение *периода ручного управления студенты – участники команды операторов* могут производить манипуляции со своим *роботом* в том случае, если робот *не сдвинулся* с места. Следующие ремонтные работы входят в число разрешенных к применению:

a. включение и выключение робота;
b. подключение батареи и/или блока Power Expander;
c. подключение адаптера VEXnet;
d. включение и выключение блока Power Expander.

За незначительные нарушения этого правила, не влияющие на ход матча, выносится предупреждение. Значительные нарушения (влияющие на ход матча) ведут к *дисквалификации*. Команды, получившие несколько предупреждений, также могут быть *дисквалифицированы* по решению главного судьи.

<SG4> Каждый *альянс* должен использовать *нагрузку периода ручного управления* в течение *матча*. *Нагрузка периода ручного управления* должна появиться в пределах поля в период между тридцатью (0:30) и нулем (0:00) секунд до конца *матча*. *Нагрузка периода ручного управления* должна быть либо аккуратно установлена на робота соответствующего цвета, находящегося на *стартовой панели альянса* в пределах *станции альянса*, либо аккуратно установлена непосредственно на *стартовую панель альянса* в пределах *станции альянса студентом – участником команды операторов*. Данное правило предназначено для ввода в соревнования кубов, но не для придания *кубам* импульса, способного переместить их за пределы *стартовой панели альянса*. Существует вероятность повреждения периметра поля во время установки *нагрузки периода ручного управления*, в связи с этим команды должны помнить о пункте <S1>.

Если альянс не использует *нагрузку периода ручного управления* согласно правилам, она автоматически засчитывается в *дальней зачетной зоне* в пользу противника.

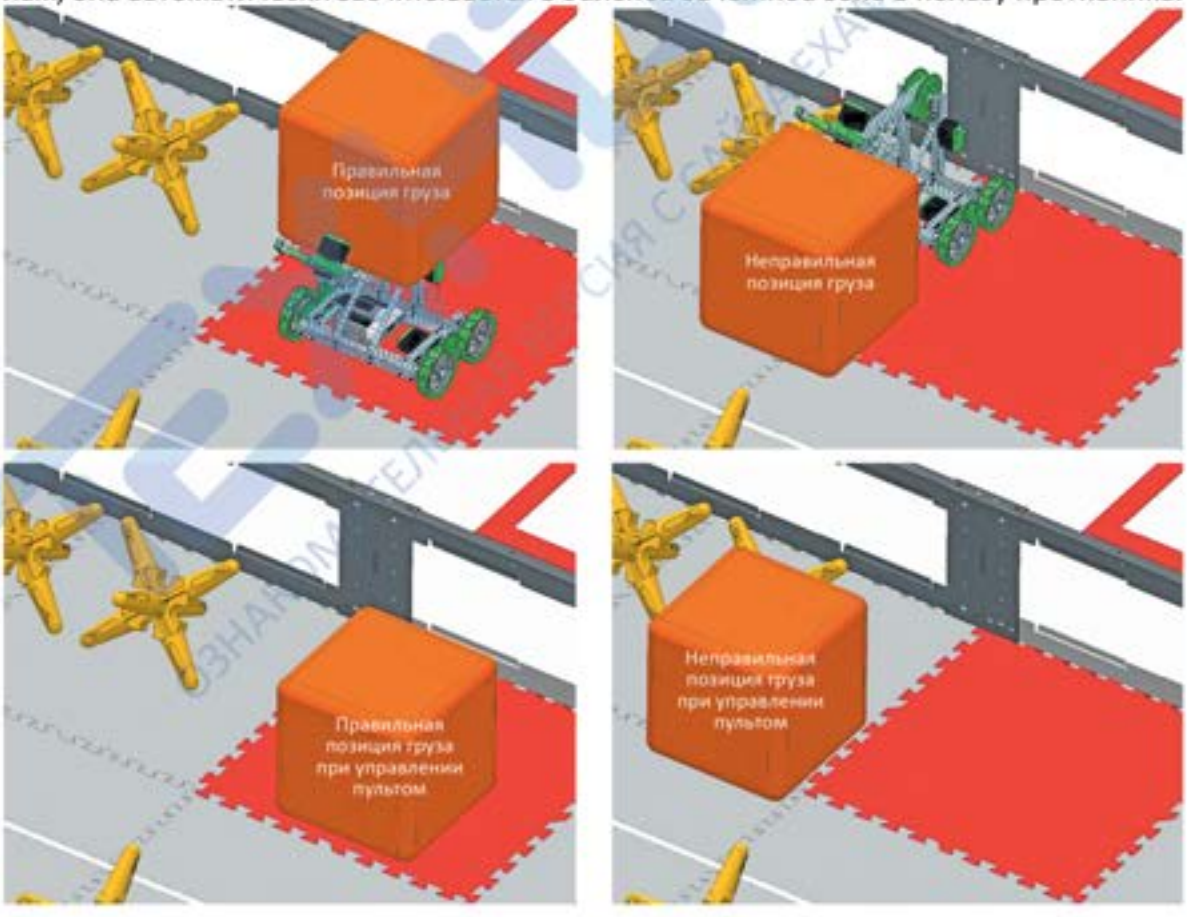


***Рис. 8-11:** Правильные и неправильные положения грузов*

<SG5> *Участникам команды операторов* не разрешается сжимать или разрушать *нагрузки периода ручного управления или предварительные нагрузки*.

<SG6> *Роботам* запрещается касаться напольных панелей в *зачетных зонах* своего цвета (т.е. *роботам* запрещается касаться *зачетных зон*, расположенных по другую сторону перегородки относительно *станции альянса*). За незначительные нарушения этого правила, не влияющие на ход матча, выносится предупреждение. Значительные нарушения (влияющие на ход матча) ведут к *дисквалификации*. Команды, получившие несколько предупреждений, также могут быть *дисквалифицированы* по решению главного судьи.

a. Учитывая специфику соревнований VRC Starstruck, возможны контакты между *роботами*, находящимися в конкурирующих *зачетных зонах*. Однако намеренное ограничение движения конкурирующего *робота* путем контакта с ним на противоположной стороне *перегородки* запрещается.

<SG7> Стратегии, приводящие к нарушению правил конкурирующим роботом, запрещаются, при этом нарушения, вызванные таким путем, не засчитываются конкурирующему альянсу. За незначительные нарушения этого правила, не влияющие на ход матча, выносится предупреждение. Значительные нарушения (влияющие на ход матча) ведут к *дисквалификации*. Команды, получившие несколько предупреждений, также могут быть *дисквалифицированы* по решению главного судьи.

<SG8> *Роботам* запрещается намеренно хвататься или сцепляться с любыми *элементами поля*, включая *зачетный столб* конкурирующего альянса. Запрещается использование механизмов, воздействующих на несколько сторон *элемента поля* с целью сцепления с ним. Цель данного правила предотвратить непреднамеренные повреждения поля и непреднамеренные сцепления с *элементами поля*. За незначительные нарушения этого правила, не влияющие на ход матча, выносится предупреждение. Значительные нарушения (влияющие на ход матча) ведут к *дисквалификации*. Команды, получившие несколько предупреждений, также могут быть *дисквалифицированы* по решению главного судьи.

a. *Роботам* разрешается контактировать с *зачетным столбом* их цвета с целью *зацепления* на нем.

<SG9> Любые нарушения правил, совершенные в течение *периода автономного управления* и не влияющие на *исход* матча, но влияющие на присуждение *автономного бонуса*, приводят к автоматическому присуждению *автономного бонуса* конкурирующему *альянсу*.

<SG10> *Зачетные элементы*, разделившиеся на несколько частей, больше не *зачитываются*.

Часть 3 – Турнир

Обзор

Главная часть робототехнических соревнований VEX Robotics Competition проходит в формате турнира. Каждый турнир включает тренировочные, квалификационные матчи и матчи на вылет. По завершении этапа квалификационных матчей команды получают место в турнирной таблице в соответствии с результатами выступлений. Команды, получившие первые места в рейтинге, примут участие в *матчах на вылет*, по результатам которых будут выявлены чемпионы турнира.

Термины, используемые в ходе турниров

Капитан альянса – представитель команды, получившей наивысшее место в рейтинге, который получает право приглашать свободные команды в свой альянс.

Формирование альянсов – процесс формирования постоянного состава альянсов, которые примут участие в *матчах на вылет*.

Автономные очки (АР) – второе основание для определения рейтинга команды. *Автономные очки* присуждаются согласно количеству автономных очков, заработанных *альянсом* в *квалификационном матче*.

Дисквалификация – штраф, назначаемый команде за грубое нарушение правил. Если команда получила дисквалификацию в ходе *квалификационного матча*, она получает (0) *WP* и *SP*. Если команда получила дисквалификацию в ходе *матча на вылет*, весь альянс получает дисквалификацию и зарабатывает поражение в матче.

Матч на вылет – матч, по результатам которого выявляется альянс-чемпион. Альянсы сталкиваются лицом к лицу в серии «две (2) победы из трех (3)», в каждом матче принимают участие две команды. Первый альянс, победивший в двух (2) матчах, переходит в следующий тур.

Тренировочный матч – матч, который не оценивается и проводится с целью предоставления командам возможности ознакомления с официальным игровым полем.

Квалификационный матч – матч, в ходе которого определяется место команды в турнирной таблице в целях последующего формирования альянсов. В ходе соревнований альянсы зарабатывают *очки за победу* и *очки за участие*.

Очки за участие (SP) – третье основание для определения рейтинга команды. *Очки*

за участие присуждаются в объеме баллов, заработанных альянсом, проигравшим в квалификационном матче.

Представитель команды – студент, избранный представителем своей команды в ходе *формирования альянсов* для участия в финальных *матчах на вылет*.

Очки за победу (WP) – первое основание для определения рейтинга команды. *Очки за победу* присуждаются за победу (два очка) и ничью (одно очко) в *квалификационном матче*.

Тренировочные матчи

Тренировочные матчи могут быть запланированы на утро дня, в который проводится мероприятие, и проводиться во время регистрации команд. Здесь необходимо приложить все усилия, чтобы предоставить командам одинаковое количество времени для практики, тем не менее система живой очереди также применима. Матчи не оцениваются, и их исход не влияет на место в турнирной таблице.

Квалификационные матчи

Расписание

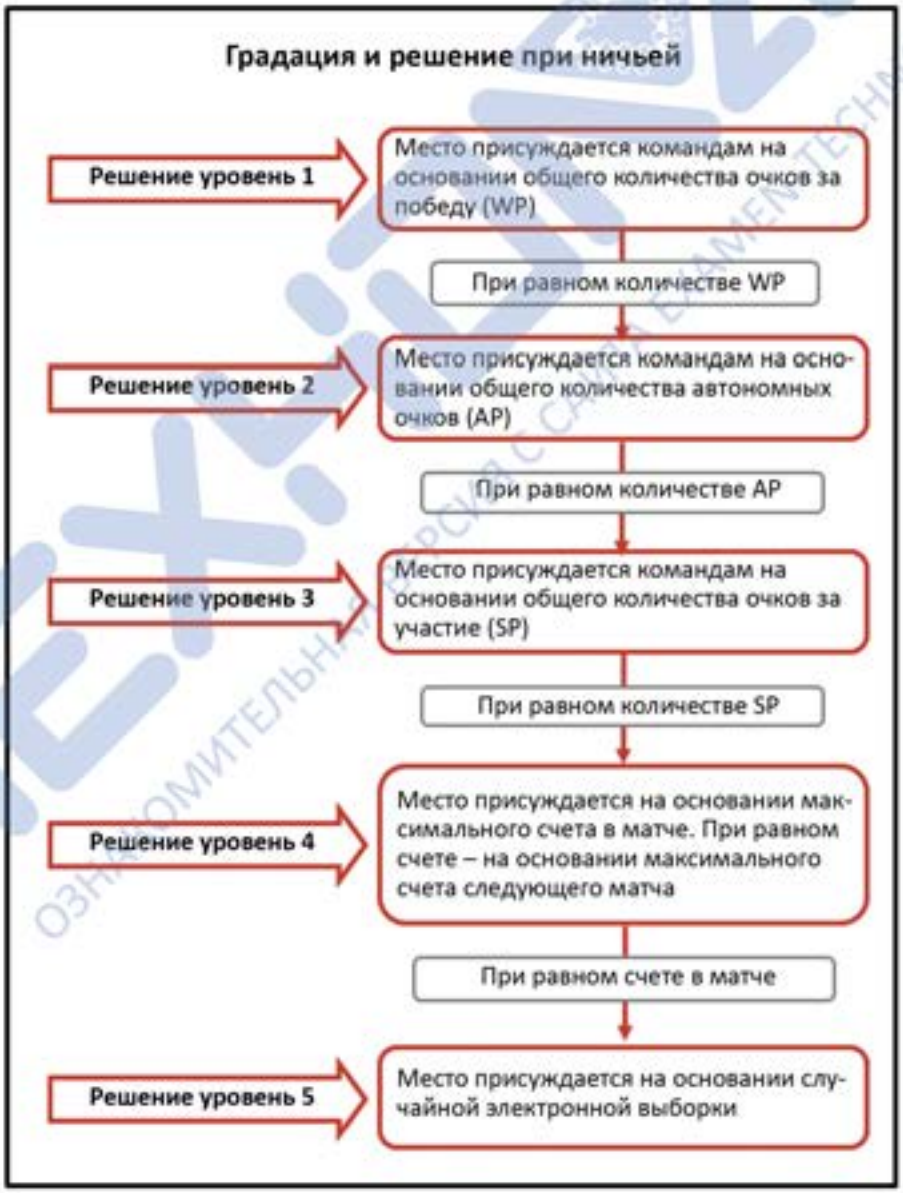
- Расписание *квалификационных матчей* должно быть объявлено до начала церемонии открытия соревнований. Расписание заключает в себе информацию о партнерах по альянсам и игровым парам для матчей. Также в него включено распределение цветов по альянсам - красный или синий. В расписание турнира, в рамках которого игры проводятся сразу на нескольких площадках, входит информация о поле, назначенном для каждого конкретного матча.
- *Квалификационные матчи* назначаются сразу по завершении церемонии открытия и проводятся в соответствии с расписанием квалификационных матчей.
- Для каждого *квалификационного матча* команды получают назначенного путем случайной выборки партнера альянса и состязаются с двумя случайно выбранными оппонентами.
- Все команды участвуют в одинаковом числе *квалификационных матчей*.
- В некоторых случаях команде может быть предложено принять участие в дополнительном *квалификационном матче*, который тем не менее не будет включен в счет этой команды.

Рейтинги

- По завершении каждого матча назначаются *очки* за победу (WP):
  - команды, победившие в *квалификационном матче*, получают два (2) *WP*;
  - команды, проигравшие в *квалификационном матче*, получают ноль (0) *WP*;
  - если *квалификационный матч* заканчивается ничьей, все четыре команды получают по одному (1) *WP*;
  - если команда получает *дисквалификацию*, ей присуждается ноль (0) *WP*;
- по итогам *квалификационного матча* всем командам также присуждаются *очки за участие (SP)*:

◦ количество *SP*, назначенных в каждом матче, соответствует счету проигравшего альянса;
◦ если матч заканчивается ничьей, оба альянса получают одинаковое количество *SP* (в соответствии с ничьей);
◦ если команда получает дисквалификацию, ей присуждается ноль (0) *SP*;
◦ если обе команды альянса получают *дисквалификацию*, командам победившего альянса присуждаются *SP* в соответствии с собственным счетом в мачте.

• В ходе *квалификационного матча*, если ни один участник команды не присутствует в зоне операторов в момент начала матча, то такая команда объявляется «отсутствующей» и получает ноль (0) *очков за победу (WP)*, ноль (0) *автономных очков (AP)* и ноль (0) *очков за участие (SP)*. Объявление команды «отсутствующей» приравнивается к *дисквалификации*.



Решение уровень 2 -> Место присуждается командам на основании общего количества автономных очков (AP) -> При равном количестве AP

Примечание: в новой версии регламента добавлен один уровень – уровень 2, вставить перевод на место старого уровня 2, остальное сместить вниз.

Матчи на вылет

• Процесс *формирования альянсов* проходит в два этапа, в ходе которых восемь капитанов формируют альянсы для участия в играх на вылет, по три команды в каждом.

• Эти восемь альянсов примут участие в турнире, где будут выявлены победители чемпионата.

• Если команда получает *дисквалификацию* в ходе *матча на вылет*, весь альянс будет *дисквалифицирован* и ему будет присуждено поражение в матче.

Процесс формирования альянсов

• Каждая команда избирает одного студента в качестве своего представителя.
  ◦ Студенты-представители примут участие в процедуре *формирования альянсов*, проводимой на поле в назначенное время.

• Всего в процессе *формирования альянсов* будет сформировано восемь альянсов.

• *Капитаном альянса* становится *студент – представитель команды*, занимающей наивысшее место в турнирной таблице и не состоящей ни в одном альянсе. Капитан имеет право приглашать другие команды для участия в своем альянсе.

• Команда может быть включена в альянс, если она еще не состоит ни в одном альянсе и до настоящего момента не отклонила ни одного приглашения о вступлении в альянс.
  ◦ Если команда приняла приглашение, она занимает место в альянсе.
  ◦ Если команда отклонила приглашение, она не может быть приглашена другим альянсом, но все же может создать собственный альянс, при наличии соответствующей возможности.
  ◦ Если команда отклонила приглашение, капитан приглашающего альянса должен озвучить другое приглашение.

• Этот процесс продолжается до тех пор, пока не будут определены все восемь *капитанов альянсов*, а также сформированы сами альянсы.

• *Тот же метод применяется при выборе капитаном альянса третьей команды-участницы. Отбор команд производится в порядке, установленном для первого раунда.* Команда, не вошедшая в состав ни одного альянса в процессе осуществления выбора первого и второго партнеров по альянсу, не будет допущена к участию в *матчах на вылет*.

• Для некоторых менее масштабных мероприятий может быть выбран другой формат формирования альянсов, наиболее подходящий для количества команд-участниц (для получения более подробной информации см. раздел «Варианты формирования» настоящего документа).

Схема проведения матчей

Матчи на выбывание проводятся в многоступенчатом формате, как показано ниже.

Присуждение очков в матчах на вылет

В ходе туров на выбывание команды не получают *очки* за победу, им присуждается победа, проигрыш или ничья. Каждая ячейка схемы соответствует матчу, в ходе которого будет выявлен альянс-победитель, прошедший в следующий тур, а именно:

- Первый альянс, победивший в двух матчах;
- При ничьей матчи будут проводиться до тех пор, пока один из альянсов не заработает две победы и не пройдет в следующий тур;

Правила проведения турнира

<T01> Судья имеет право принятия окончательного решения в ходе соревнований. *Решение судьи является окончательным.*

a. Судьи не просматривают записи.

b. Любые вопросы к судьям могут быть озвучены студентами – участниками команды операторов в период времени, равный двум (2) квалификационным матчам, либо непосредственно после оглашения счета матча на выбывание.

<T02> Единственные участники команд, которым разрешается находиться около игрового поля, это трое участников команд операторов, имеющих соответствующие знаки отличия. Эти знаки не могут быть переданы другим участникам в ходе матча.

<T03> В ходе матчей две команды альянса встречаются на поле. Любая команда, не принявшая участие в первом матче серии на выбывание, должна принять участие во втором матче, без исключений. В третьем и последующих матчах могут принимать участие любые две из трех команд. До начала каждого *матча на выбывание капитан альянса* предоставляет судьям информацию о том, какие две команды альянса примут участие в предстоящем матче.

<T04> Тайм-ауты в ходе квалификационных раундов отсутствуют; в ходе раундов на выбывание каждый альянс имеет право на один тайм-аут длительностью до трех минут, по решению главного судьи. Матчи должны проводиться в соответствии с расписанием.

a. В случае когда робот не может принять участие в матче, минимум один участник команды должен явиться на стартовую площадку матча.

<T05> В ходе матчей все участники команд, включая тренеров, находящиеся в боксе или на станции альянса, должны использовать защитные очки либо очки с боковыми щитками. Применение защитных очков рекомендовано всем участникам команд при нахождении в зоне боксов.

Изменения в мероприятии

Малые соревнования (турниры 1-го уровня):

Если в турнире принимает участие менее 24 команд (количество, необходимое для формирования восьми полных альянсов), турнир может быть организован в следующем порядке:

- если в турнире участвует от 18 до 23 команд
  - альянсы, как и прежде, формируются из трех команд
  - количество альянсов равно количеству команд, разделенному на три, не учитывая остаток (например, число команд равно 19, 19/3 = 6.33 -> 6 альянсов)
- если число команд равно 17 или менее
  - альянсы формируются из двух команд
  - количество альянсов равно количеству команд, разделенному на два, не учитывая остаток (например, число команд равно 13, 13/2 = 6.5 -> 6 альянсов)
  - Некоторые турниры данного формата могут включать альянсы различных типов, например, один альянс из трех команд. Это позволит всем командам принять участие в турах на выбывание (например, если участвуют 17 команд, 7 альянсов состоят из двух команд и один альянс - из трех). Альянсы из трех команд должны тем не менее следовать <T03>, в том числе в случае участия в состязании против альянсов, состоящих из двух команд.
    - Если для турнира принят данный формат организации, формирование альянсов производится в стандартном порядке, пока каждый альянс включает более двух команд. Оставшаяся команда затем добавляется к альянсу, занимающему последнее место в турнирной таблице (напр., 7-й находится ниже 6-го в рейтинге)
- Многоступенчатая схема матчей позволяет применить к подобному турниру тот же формат, что и к полноценному мероприятию. При этом альянс, у которого отсутствует оппонент, проходит в следующий тур (напр., при наличии семи альянсов 8-й альянс будет отсутствовать, поэтому 1-й альянс переходит в следующий тур в четвертьфинале)

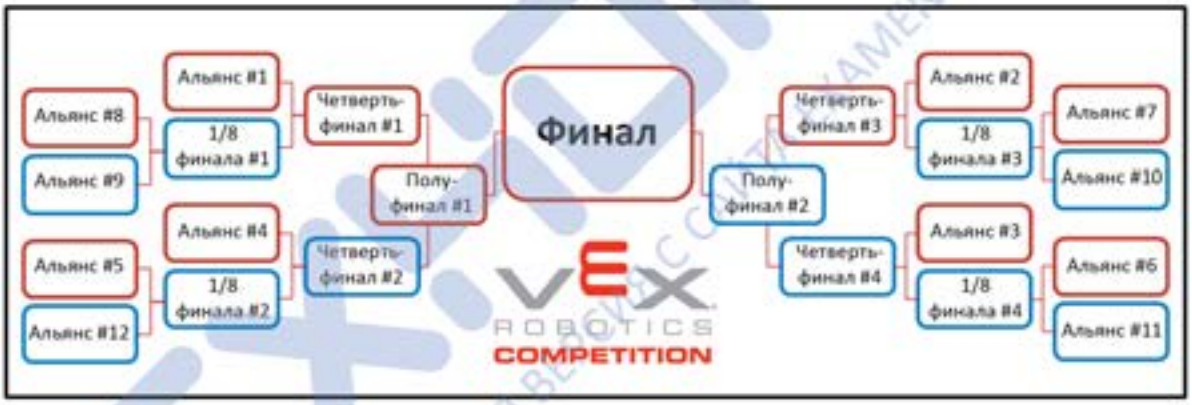
Средние соревнования (турниры 2-го уровня и выше):

Ко всем турнирам с участием менее 24 команд могут быть применены следующие правила:

- Стандартный формат (8 альянсов по 3 команды)
- 12 альянсов по 2 команды
  - Такая система рекомендована для турниров, где недостаточно квалификационных зон для проведения квалификации всего альянса (трех команд альянса) для участия в мировом чемпионате
  - Площадка для туров на выбывание при наличии 12 альянсов может быть реализована следующим образом:

Высота поля:

Игровое поле для любого турнира должно быть расположено на полу. Организаторы некоторых турниров могут принять решение о подъеме игрового поля на высоту от 61 см до 91 см. Платформы для проведения мирового чемпионата – 2017 по робототехнике VEX имеют высоту 61 см. В целях безопасности участники команды операторов могут стоять на каком-либо предмете в ходе матча, вне зависимости от наличия подъема поля.



Для заметок

Часть 4 – Робот

Обзор

В этом разделе представлены нормы и требования к проектированию и конструированию роботов. Робот, принимающий участие в соревнованиях VEX Robotics Competition, – это управляемое дистанционно и/или автономное самоходное устройство, спроектированное и собранное зарегистрированной в качестве участника соревнований командой студентов, для выполнения определенных задач в ходе соревнований VEX Robotics Competition Starstruck. Перед началом каждого мероприятия все роботы должны пройти контроль.

Требования к конструкции робота

Существуют особые нормы и ограничения, которые применяются к проектированию и конструированию робота. Пожалуйста, убедитесь, что вы ознакомились с каждым из них на момент начала работы по проектированию робота.

<R1> Каждая команда может выставить для участия в соревнованиях VEX Robotics Competition только одного (1) робота. Хотя внесение изменений в робота в ходе соревнований допускается, возможности команды ограничены только одним (1) роботом. В связи с этим робот VEX, созданный для участия в робототехнических соревнованиях VEX Robotics Competition, оснащен следующими подсистемами:

Подсистема 1: Мобильная база робота, включающая в себя колеса, гусеницы, ходули или любые другие механизмы, обеспечивающие перемещение робота по большинству плоских игровых поверхностей поля. Для стационарного робота база робота без колес оценивается как подсистема 1.
Подсистема 2: Система питания и управления, включающая батарею VEX, систему управления VEX, а также необходимые моторы для мобильной базы робота.
Подсистема 3: Дополнительные механизмы (и необходимые моторы), позволяющие манипулировать игровыми объектами и преодолевать препятствия на поле.

Перечисленное выше является описанием минимальной комплектации робота – участника мероприятия VEX Robotics Competition, в которую обязательно должны быть включены подсистемы 1 и 2. Если вы вынесли целую подсистему, описанную в пунктах 1 или 2, за пределы робота, это означает, что вы создали второго робота и нарушили правила.

a. Команда не может принимать участие с одним роботом при наличии собранного либо модифицированного второго.

b. Команды не могут использовать разных роботов в ходе соревнований.

<R2> Каждый робот должен пройти полную экспертизу и получить разрешение на участие в мероприятии. Экспертиза подтверждает, что конструкция робота удовлетворяет всем условиям и требованиям. Начальные проверки назначаются на период регистрации команд и тренировочных матчей.

a. В случае если робот подвергся значительным изменениям, он должен пройти повторную экспертизу прежде, чем будет допущен к дальнейшему участию в мероприятиях.
b. Любые конфигурации робота подлежат экспертизе и получению допуска к состязаниям.
c. Командам может быть предложено пройти случайные экспертизы, осуществляемые персоналом мероприятия. Отказ от прохождения экспертизы наказывается дисквалификацией.
d. Право принятия решения о нарушении роботом правил мероприятия принадлежит судьям и инспекторам. В рамках данного мероприятия команда-нарушитель будет дисквалифицирована и ее робот будет удален с игрового поля до момента прохождения им повторного осмотра.

<R3> Следующие типы механизмов и элементов запрещены к применению:

a. способные повредить элементы игрового поля;
b. способные повредить других соревнующихся роботов;
c. увеличивающие риск зацепления.

<R4> Габариты всех роботов на момент начала матча не должны превышать допустимых значений Ш x Д x В 18 x 18 x 18 (в дюймах).

a. В ходе проверок роботы будут измерены одним из двух способов:
i. Робот помещается в «размерный ящик», внутренние размеры которого соответствуют указанным выше ограничениям. Чтобы пройти проверку, робот должен поместиться в ящике, не касаясь при этом ни одной из стенок ящика или его потолка.
ii. Робота измеряют при помощи измерительного инструмента VEX Robotics Competition Robot Sizing Tool. Робот, размещенный на плоской поверхности, не должен касаться границ измерительной линейки в момент ее прохождения над поверхностью. Для получения визуального представления см. http://www.vexrobotics.com/vexedr/products/competition-products/276-2086.html
b. После начала матча робот может превысить указанные стартовые габариты.
c. Любые ограничители, обеспечивающие поддержание роботом установленных стартовых габаритов (стяжки, резиновые кольца и пр.), ДОЛЖНЫ находиться на роботе и быть присоединены к нему в течение всего матча.

<R5> Роботы могут быть собраны исключительно из официальных элементов, входящих в систему проектирования VEX Robotics Design System, если только они не учтены особо в настоящих правилах.

a. В ходе проверки при возникновении вопросов относительно соответствия компонентов перечню официальных компонентов VEX инспектор может попросить команду предъявить соответствующий документ, подтверждающий источник приобретения того или иного компонента. В число подобных документов входят чеки, инвентарные номера элементов либо другая печатная документация.

b. Допускается применение только элементов системы проектирования VEX Robotics Design System, предназначенных для использования при конструировании роботов. Использование дополнительных элементов, не входящих в стандартный комплект, противоречит правилам (пожалуйста, не используйте элементы оформления VEX, сопутствующие материалы, упаковку и прочие продукты, не предназначенные для сборки соревновательного робота VEX).

c. Продукты VEXpro, VEXIQ и VEX Robotics из линейки Hexbugs не могут быть использованы при конструировании робота, если только иное не установлено в <R7>. Продукты VEXpro, VEXIQ и VEX Robotics из линейки Hexbugs, одновременно входящие в линейку продуктов VEX, разрешены к использованию.

d. Официальные элементы системы проектирования VEX Robotics Design System, снятые с производства, могут быть использованы при проектировании. Тем не менее команды должны быть ознакомлены с <R5a>.

<R6> Официальные продукты VEX могут быть приобретены ИСКЛЮЧИТЕЛЬНО у VEX или у официальных дилеров VEX. На сайте www.vexrobotics.com представлена информация относительно подтверждения подлинности продукции.

<R7> Допускается использование в проекте робота следующих элементов, не входящих в линейку VEX:

a. Материал, используемый исключительно в качестве цветного фильтра для светового сенсора VEX;

b. Любые части, *идентичные* подлинным частям VEX. Под это правило подпадают продукты, идентичные официальным продуктам VEX во всем, кроме цвета.

Примечание: право принятия решения об идентичности элемента официальному элементу VEX принадлежит инспектору.

c. Любые доступные для приобретения винты #4, #6, #8, M2, M2.5, M3 или M4 до 2 дюймов в длину, а также любые доступные для приобретения гайки к этим винтам.

d. Командам разрешается добавлять нефункциональные декорации, не способные оказывать существенное влияние на функционирование робота или общий исход матча. Эти декорации должны соответствовать духу мероприятия. Право принятия решения относительно «нефункциональности» декорации принадлежит инспектору.

i. Анодирование или окраска частей считается нефункциональной декорацией.

ii. Любые щитки или наклейки должны быть закреплены на разрешенных к использованию материалах, выполняющих ту же функцию. Иными словами, если огромная наклейка, закрепленная на роботе, используется для удерживания зачетных объектов на роботе, предотвращая их падение, данная наклейка должна быть закреплена на разрешенном к использованию материале VEX, используемом в конструкции робота с той же целью.

iii. При использовании динамика VEX (276-1504) звуковая дорожка не должна служить помехой остальным участникам. Главный инспектор и главный судья принимают решение о соответствии аудиодорожки духу мероприятия.

e. Любые неаэрозольные масла и смазочные элементы, используемые в исключительно ограниченных количествах на поверхностях и в местах, не предназначенных для контакта со стенками игрового поля, мягкой поверхностью поля, игровыми объектами или другими роботами.

f. Элементы из небьющегося пластика (поликарбонат, ацетил-монополимер (делрин), ацетил-сополимер (ацетрон GP), POM (ацеталь), ABS, PEEK, PET, HDPE, LDPE, нейлон (все марки), полипропилен, FEP), вырезанные из цельного листа 12 x 24 (в дюймах) толщиной до 0,070 дюйма.

i. Пластик может быть модифицирован механическим путем - методом обрезки, сверления, сгибания и пр., но не может быть обработан химическим путем, расплавлен или отлит. Команды могут нагревать поликарбонат при сгибании.

g. Небольшой объем клейкой ленты может быть использован для следующих целей:

i. Исключительно для скрепления любого соединения между концами двух (2) кабелей VEX.

ii. Для присвоения меток проводам и моторам.

iii. Тефлоновая лента может быть использована исключительно для предотвращения возможности протечек на резьбовых участках пневматических соединений.

iv. Для закрепления и удерживания адаптера VEXnet на микроконтроллере VEX на базе ARM® Cortex®. Рекомендуется использование ленты аналогичным путем для скрепления соединений робота.

h. Горячий клей для фиксации кабельных соединений.

j. Удлинительный кабель USB может быть использован только с целью дистанционной установки адаптера VEXnet. Адаптер должен быть установлен следующим способом.

i. Адаптер VEXnet должен быть установлен таким образом, чтобы ни один металлический элемент не касался адаптера выше логотипа VEXnet.

ii. Не рекомендуется устанавливать любые металлические элементы на высоте до 2 дюймов над поверхностью адаптера VEXnet.

k. Объем применяемой плетеной нейлоновой веревки 1/8 дюймов не ограничен.

l. Разрешается использование доступных для приобретения элементов в целях защиты или обмотки 2-, 3- и 4-проводных кабелей, а также использование пневматической трубки для защиты, организации или управления. Сюда входит неограни-

ченный объем изоляционной ленты, держателей и направляющих для кабелей и пр. Обратите внимание: право принятия решения об использовании материалов в допустимых пределах (указано выше) принадлежит инспектору.
m. Шпильки VEX IQ, используемые исключительно для присоединения пластинки с идентификационным номером команды VEX.

<R8> Дополнительные элементы системы проектирования VEX Robotics Design System, выпущенные в период проведения чемпионата, могут быть использованы при проектировании роботов.

При выпуске некоторые «новые» элементы могут сопровождаться инструкциями относительно ограничений по применению. Эти ограничения документируются в «Новостях команд». Новости команд публикуются на домашней странице соревнований *VEX Starstruck* в разделе Competition на сайте www.vexrobotics.com

<R9> Робот может быть оснащен только одним (1) микроконтроллером VEX EDR.

a. В число микроконтроллеров VEX входят микроконтроллер VEX v.5 PIC и микроконтроллер VEX ARM® Cortex®-based.
b. Использование микроконтроллеров, не вошедших в линейку VEX, таких как VEXpro, VEX RCR, VEX IQ и VEX Robotics на базе Hexbugs, запрещено.

<R10> Для передачи данных роботы могут использовать только систему коммуникации VEXnet.

a. Использование радиопередатчиков VEX 75 мГц запрещено. (На некоторых мероприятиях допускается использование радиопередатчиков 75 мГц; см. «Специальные изменения правил мероприятия» ниже в настоящем разделе.)
b. Электронная продукция линеек VEXpro, VEX-RCR, VEX IQ, и VEX Robotics на базе Hexbug запрещена к применению, включая ВСЮ электронику VEXplorer.
c. Джойстик VEXnet может быть использован только вместе с микроконтроллером VEX на базе ARM® Cortex®. Модернизированный передатчик VEXnet 75 мГц может быть использован только в совокупности с микроконтроллером PIC. Изменение приведенных выше пар запрещено.

<R11> В конструкции робота допускается использование следующих компонентов:

Вариант 1: До десяти (10) двигателей VEX EDR или серводвигателей VEX Servos (любая комбинация, до 10 шт.) и оригинальная пневматическая система VRC. (см. <R18>)

Вариант 2: До двенадцати (12) двигателей VEX EDR или серводвигателей VEX Servos (любая комбинация, до 12 шт.) и ни один из компонентов пневматической системы, за исключением пневматических трубок.

a. 2-проводные двигатели должны управляться через 2-контактные порты, напрямую через микроконтроллер VEX ARM® Cortex®-based (арт. 276-2194) или через контроллер двигателей VEX Motor Controller 29 (арт. 276-2193).
b. Не допускается подключение одного двигателя к нескольким 2-контактным портам двигателей, 3-контактным ШИМ-портам двигателей или контроллерам двигателей VEX Motor Controller 29.

<R12> К одному порту двигателя микроконтроллера или блока Power Expander (арт. 276-2271) может быть присоединено не более одного (1) Y-кабеля VEX. (Запрещается подключать Y-кабель к Y-кабелю с целью подключения двух и более моторов к одному порту двигателя.)

a. Команды, использующие в системе управления микроконтроллер VEX ARM® Cortex®-based, имеют право подключать только один (1) 2-проводной двигатель к каждому из 2-проводных портов двигателя. Запрещается подключать Y-кабель к 2-проводному порту.
b. Командам запрещено подключать Y-кабель к контроллеру двигателей VEX Motor Controller 29.

<R13> В качестве источника питания для робота, разрешенного к применению в рамках робототехнических соревнований VEX Robotics Competition, может быть использован один (1) комплект батарей VEX 7.2V любого типа (за исключением случая, когда в конструкцию робота входит блок VEX Power Expander), а также один (1) комплект резервных батарей 9V. Роботы, в конструкцию которых входит блок VEX Power Expander, могут также использовать вторую (2) батарею для робота VEX 7.2V.

a. Запрещается размещать любые другие дополнительные батареи на роботе (даже если они не подключены к роботу).
b. В систему питания робота может входить только один (1) блок VEX Power Expander.
c. В целях обеспечения бесперебойного беспроводного обмена данными к системе VEXnet каждой команды должна быть присоединена полностью заряженная резервная батарея 9V (для присоединения используется держатель резервной батареи VEXnet (арт. 276-2243).
d. Разрешается применение любого комплекта батарей VEX 7.2V в количествах, указанных выше.
e. Для осуществления зарядки комплекта батарей VEX 7.2V могут быть использованы следующие зарядные устройства VEX: Умное зарядное устройство (арт. 276-1445); Умное зарядное устройство v2 (арт. 276-2519); (арт. 276-2221 (снято с производства)), (арт. 276-2235 (снято с производства)). Запрещается использование любых других зарядных устройств.
f. К джойстику VEXnet могут быть подключены только батареи AAA.

i. В рамках некоторых мероприятий допускается подключение периферийных источников питания к джойстику VEXnet. При наличии подобных предусмотренных источников питания их применение командами в целях энергоснабжения джойстиков VEXnet считается допустимым.

<R14> Для управления одним роботом в рамках турнира может использоваться не более двух ручных передатчиков VEX. Любые модификации передатчиков запрещены.

a. Любые другие методы управления роботами (световые, звуковые и пр.) запрещены.

<R15> Запрещено модифицировать компоненты VEX следующими способами:

a. не допускается внесение любых изменений в оригинальную конструкцию электромоторов (включая внутренние PTC), удлинительные провода, датчики, контроллеры, блоки батарей, резервуары, электромагнитные элементы, поршни и другие электрические или пневматические элементы системы проектирования VEX Robotics Design System;

i. допускается внутренний или внешний механический ремонт ограничительного или бамперного переключателя VEX; использование элементов указанных устройств в других конструкциях запрещено.

ii. Ремонт внешних проводов на электрических элементах VEX может быть осуществлен методом пайки с использованием поворотных или обжимных соединителей, изоляционной ленты или сжимающейся трубки, с учетом оригинальной длины/функциональности проводов. Провод, использованный для ремонта, должен быть идентичен оригинальному проводу VEX.

Ответственность за проведение ремонта ложится на команды. Некорректный ремонт может привести к нежелательным последствиям.

iii. Команды имеют право заменять шестерни 2-проводного мотора 393 и 2-проводного мотора 269 соответствующими сменными шестернями VEX.

iv. Команды не могут обрезать пневматические трубки по желаемой длине.

b. Запрещается сварка, пайка мягким и твердым припоем, склеивание и соединение элементов любым другим методом, не входящим в систему проектирования VEX Robotics Design System.

i. Разрешается фиксация механических крепежных средств с помощью локтайта либо другого аналогичного жидкого фиксатора; применение жидких фиксаторов разрешено исключительно в соединении с крепежными средствами.

ii. Командам разрешается обжигать/оплавлять концы нейлоновой веревки 1/8 в целях предотвращения их изнашивания.

iii. Склеивание, разрешенное по <R7h>, является исключением из настоящего правила.

<R16> Переключатель питания робота должен находиться в прямом доступе (не требующем поднятия или перемещения робота). Световые индикаторы на микроконтроллере должны находиться в прямой видимости, чтобы персонал мероприятия имел возможность по ним контролировать состояние робота.

<R17> Команды должны доставить своих роботов на поле в состоянии полной готовности к игре. Команды, использующие пневматику VEX, должны полностью зарядить системы перед предъявлением своего робота на игровое поле.

<R18> Максимальный заряд пневматических устройств не может превышать значения в 690 кПа. Командам разрешается использовать в конфигурации робота до двух (2) включительно официальных пневматических воздушных баллонов.

Цель этого правила заключается в ограничении допустимого значения давления воздуха в пневматических баллонах, а также нормального рабочего давления воздуха в пневматических цилиндрах и трубках робота. Никакие другие элементы (например, медицинские жгуты) не могут быть использованы для создания воздушного давления. Использование командами цилиндров или дополнительных пневматических трубок, не несущих никакой функциональной нагрузки, кроме создания дополнительных резервуаров для хранения воздуха, считается нарушением данного правила и приводит к дисквалификации.

<R19> Каждая команда, пожелавшая принять участие в робототехническом турнире VEX Robotics Competition Tournament, должна пройти регистрацию на robotevents.com. При регистрации команды получают идентификационные номера команд VEX (ID команды VEX) и приветственный набор, куда входят идентификационные таблички команд VEX. Каждый робот должен иметь таблички с ID создавшей его команды VEX как минимум с двух противоположных сторон корпуса.

a. Идентификационные пластинки считаются декорацией, не несущей функциональной нагрузки, и не могут быть использованы в качестве функциональной части робота.
b. Эти таблички должны полностью соответствовать правилам, применяемым к роботам (то есть они должны удовлетворять требованию стартовых габаритов согласно <R4> и не могут служить для зацепления, и т.д.).
c. Для каждого матча пластинки роботов должны быть окрашены в цвета своего альянса (то есть роботы красного альянса должны перед началом матча быть укомплектованы пластинками красного цвета).

<R20> В течение периода автономного управления операторы не могут использовать ручные устройства управления. В связи с этим команды несут ответственность за загрузку в своих роботов пользовательского программного обеспечения, посредством которого будет осуществляться управление роботом в течение автономного периода.

Для получения более подробной информации командам необходимо проконсультироваться с руководствами, выпускаемыми изготовителями выбранного программного обеспечения.

<R21> Любое нарушение правил, применяемых к роботам, ведет к тому, что команда не будет допущена к состязаниям до прохождения инспекции (по <R2d>). Более того, команды, которые пытаются намеренно обойти или нарушить правила с целью получения преимущества перед оппонентами, рассматриваются как нарушители соревновательного духа мероприятия. В связи с этим при выявлении факта нарушения правил одним из указанных способов нарушитель может быть дисквалифицирован на предстоящие матчи, мероприятие и даже последующие мероприятия по решению игрового проектного комитета робототехнических соревнований VEX Robotics Competition.

Особые изменения правил мероприятия

Правила, перечисленные в данном разделе, определяют порядок проведения игр в рамках ВСЕХ мероприятий чемпионатов по робототехнике VEX Robotics Competition. Тем не менее иногда возникает необходимость внесения незначительных модификаций в правила с целью приведения их в соответствие с условиями локальных мероприятий. В частности, следующие исключения допускаются для ряда мероприятий:

a. Использование радиоустройств приема-передачи VEX 75 мГц вместо либо в комбинации с системой VEXnet.
b. Допускается использование батарей AA для питания робота вместо комплекта батарей VEX 7.2V.

При внесении любых изменений в данные правила организаторы мероприятия должны незамедлительно проинформировать об этом участвующие команды. Исключительно важно, чтобы команды – участницы мероприятий, для которых разрешается использование устройств приема-передачи 75 мГц, использовали правильный канал связи.

Для заметок

Для заметок

Для заметок

Для заметок

Для заметок

Для заметок

Учебно-методическое издание

Горнов Олег Александрович

Основы робототехники и программирования с VEX EDR

Издательство ***«ЭКЗАМЕН»***
«ЭКЗАМЕН-ТЕХНОЛАБ»

Гигиенический сертификат
№ РОСС RU. ПЩ01.Н00199 от 19.05.2016 г.

Главный редактор *Л. Д. Лаппо*
Корректор *О. Ю. Казанаева*
Дизайн обложки
и компьютерная верстка *А. А. Винокуров*

107045, Москва, Луков пер., д. 8.
E-mail: по общим вопросам: robo@examen-technolab.ru;
www.examen-technolab.ru
по вопросам реализации: sale@examen-technolab.ru
тел./факс +7 (495) 641-00-19 (многоканальный)

Отпечатано в соответствии с предоставленными материалами
в ООО «ИПК Парето-Принт», г. Тверь, www.pareto-print.ru

Основы робототехники и программирования с VEX EDR

Учебно-методическое пособие

www.examen-technolab.ru

Артикул ТЭ-276-М

ISBN 978-5-377-11671-4