



МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

12-15
ЛЕТ



К. В. Ермишин

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

ОБРАЗОВАТЕЛЬНЫЙ
РОБОТОТЕХНИЧЕСКИЙ МОДУЛЬ

(БАЗОВЫЙ УРОВЕНЬ)

12 – 15 ЛЕТ

Учебно-методическое пособие



МОСКВА
2015

УДК 372.8:004
ББК 32.816
Е72

Ермишин К. В.

Е72 Методические рекомендации для преподавателя: образовательный робототехнический модуль (базовый уровень): 12 – 15 лет/ К. В. Ермишин. — М.: Издательство «Экзамен», 2015. — 144 с.

ISBN 978-5-377-10297-7

Данное пособие предназначено для применения совместно с образовательным робототехническим модулем «Базовый уровень», созданным на базе робототехнического конструктора VEX EDR. В пособии описываются состав и функциональные возможности робототехнического модуля и примеры его применения. Пособие содержит информацию о назначении модуля и элементов, входящих в его состав, а также о возможностях применения данного модуля в образовательном процессе в средних и старших классах. Образовательная робототехническая платформа VEX EDR представляет собой открытую платформу для создания робототехнических комплексов для образовательной, соревновательной и исследовательской деятельности. В дополнение к этому образовательный робототехнический модуль «Базовый уровень» оснащён программируемым контроллером, представляющим собой открытую программно-аппаратную платформу, преемственную с программируемыми контроллерами типа Arduino. Благодаря этому робототехнический модуль «Базовый уровень» может применяться на стыке двух направлений образовательной деятельности учащихся – реализации творческих инженерных проектов на базе программно-аппаратных платформ открытого типа, а также создания робототехнических комплексов для задач образовательного и соревновательного характера.

Данное пособие носит рекомендательный характер и содержит основную информацию, необходимую для работы с образовательным робототехническим модулем «Базовый уровень», а также типовые примеры по созданию и программированию учебных моделей роботов.

УДК 372.8:004
ББК 32.816

Подписано в печать с диапозитивов 20.08.2015.
Формат 60x90/8. Гарнитура «Calibri». Бумага офсетная.
Усл. печ. л. 18. Тираж 300 экз. Заказ №

ISBN 978-5-377-10297-7

© Ермишин К. В., 2015

© Издательство «ЭКЗАМЕН», 2015

© «ЭКЗАМЕН-ТЕХНОЛАБ», 2015

Содержание

| | |
|---|---------|
| Введение | Стр. 5 |
| Состав образовательного робототехнического модуля | Стр. 7 |
| Конструктивные элементы и комплектующие конструкторов VEX | Стр. 13 |
| Исполнительные механизмы конструкторов VEX | Стр. 17 |
| Базовые принципы проектирования роботов | Стр. 22 |
| Программируемый контроллер | Стр. 25 |
| Основы работы в Arduino IDE | Стр. 30 |
| Программирование контроллеров Arduino | Стр. 34 |
| Приложение 1. Работа с основными устройствами и комплектующими | Стр. 37 |
| Пример подключения и работы с тактильными датчиками, концевыми выключателями и кнопками | Стр. 40 |
| Пример подключения и работы с датчиком освещённости | Стр. 44 |
| Пример подключения и работы с ИК-датчиком линий | Стр. 48 |
| Пример подключения и управления моторами | Стр. 51 |
| Пример подключения и управления сервоприводом | Стр. 55 |
| Пример подключения и работы с УЗ-сонаром | Стр. 57 |
| Пример подключения и работы с оптическим энкодером | Стр. 62 |
| Пример подключения и работы с инкрементным энкодером | Стр. 67 |
| Работа со встроенным Bluetooth-модулем | Стр. 72 |
| Приложение 2. Разработка макета робота | Стр. 75 |
| Движение робота вперёд-назад и осуществление поворотов | Стр. 78 |
| Управление манипулятором робота | Стр. 81 |
| Подключение ультразвукового дальномера | Стр. 85 |

| | |
|--|----------|
| Работа с ИК-датчиками для обнаружения линии | Стр. 88 |
| Разработка комплексной системы управления робота | Стр. 92 |
| Приложение 3. Руководство по сборке Clawbot | Стр. 97 |
| Приложение 4. Руководство по сборке мобильного робота с манипулятором | Стр. 117 |
| Приложение 5. Руководство по сборке мобильного робота повышенной проходимости | Стр. 127 |
| Приложение 6. Руководство по сборке мобильного робота на базе гусениц | Стр. 135 |



Введение

Мировые тенденции развития инженерного образования свидетельствуют о глобальном внедрении информационных технологий в образовательный процесс. Робототехника является весьма перспективной областью для применения образовательных методик в процессе обучения за счёт объединения в себе различных инженерных и естественнонаучных дисциплин. В результате такого подхода наблюдается рост эффективности восприятия информации учащимися за счёт подкрепления изучаемых теоретических материалов экспериментом в междисциплинарной области.

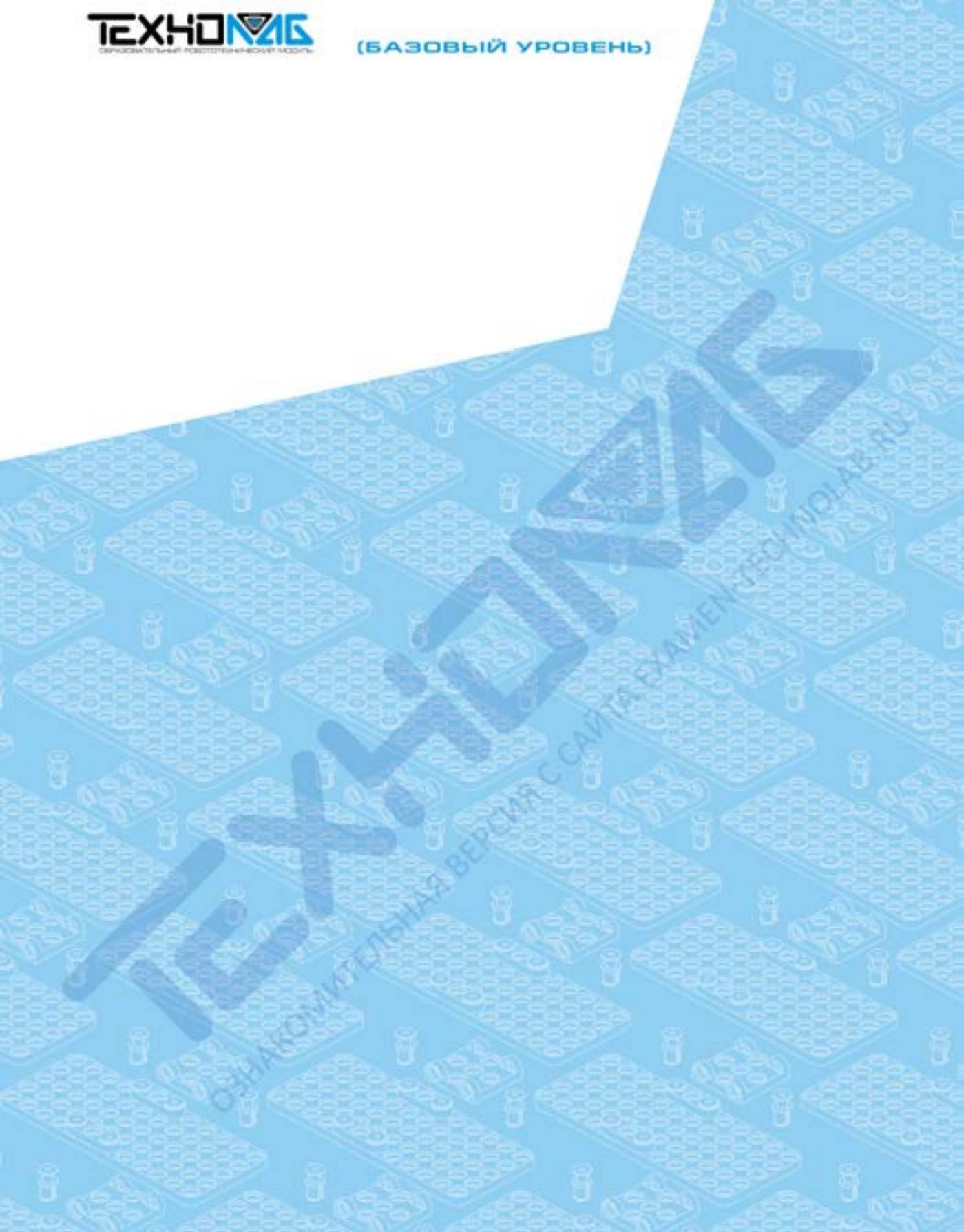
Образовательный робототехнический модуль «Базовый уровень» предназначен для изучения основ робототехники, элементов электроники и микропроцессорной техники, теоретических основ механики и деталей машин, а также программирования микропроцессорных устройств и разработки систем управления роботами.

Помимо применения в образовательных целях, данный модуль в первую очередь ориентирован для применения в робототехнических соревнованиях. Поэтому данный модуль не нацелен на проведение отдельных лабораторных работ по каким-либо направлениям, а предназначен для применения произвольным образом в рамках решения робототехнических задач различной сложности.

В состав модуля входят различные металлические детали, крепёжные элементы, зубчатые передачи и многое другое. Благодаря конструктивным возможностям модуля можно разрабатывать сложные механизмы, состоящие из различных передач и металлических конструкций. С использованием данного модуля также возможно разрабатывать роботов и робототехнические устройства, выполняющие вполне реальные задачи различной сложности, например исследование местности, манипулирование объектами, погрузка и разгрузка грузов, транспортирование объектов, патрулирование территорий и многое другое.

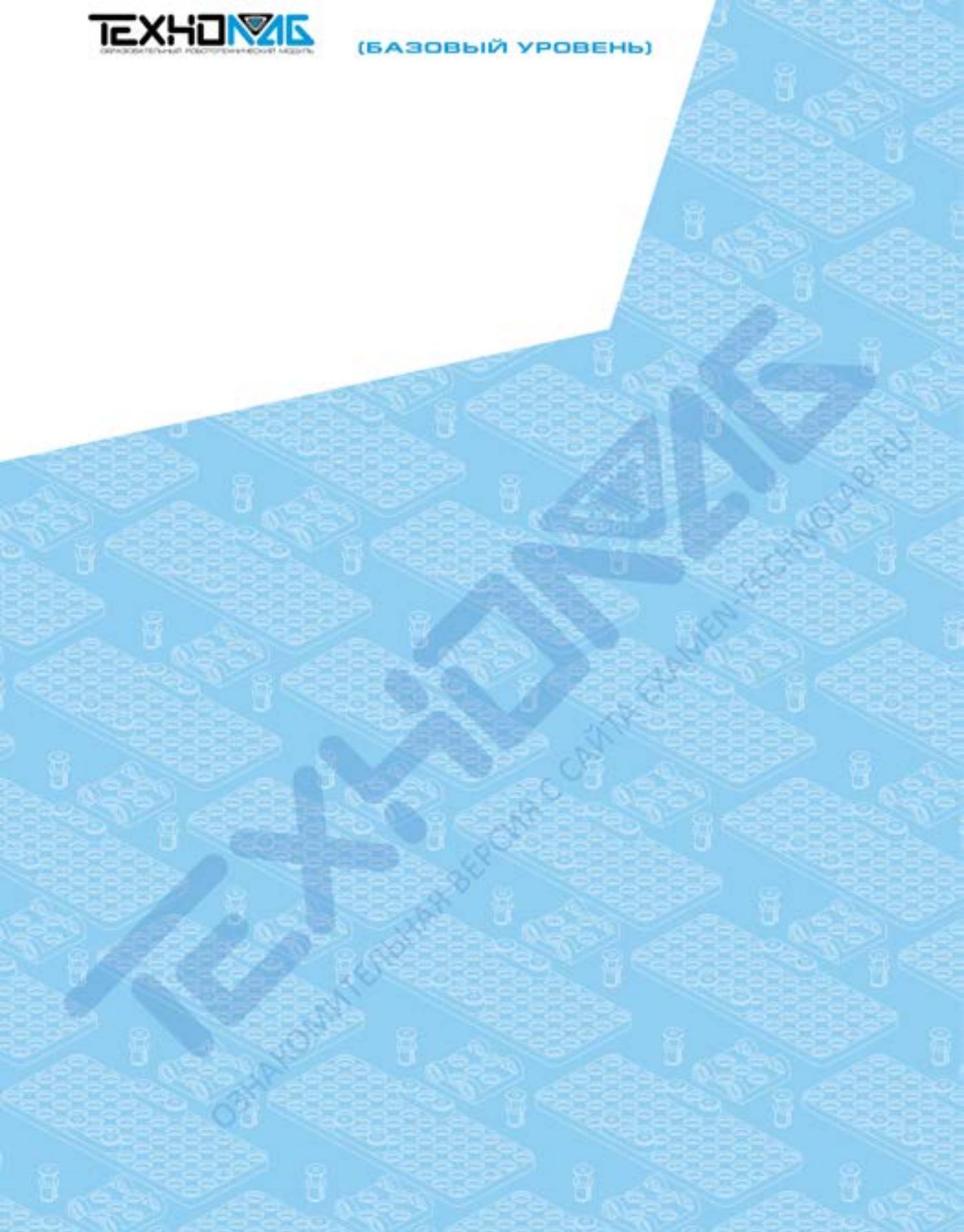
Таким образом, применение данного образовательного робототехнического модуля даёт возможность осуществить плавный переход применения образовательных технологий в области робототехники к полноценной инженерной и проектной деятельности.





Состав образовательного робототехнического модуля





Состав образовательного робототехнического модуля

Образовательный робототехнический модуль «Базовый уровень» создан на базе робототехнического конструктора VEX EDR Clawbot и программируемого контроллера, а также комплекта инкрементных энкодеров, набора УЗ-сонаров и ИК-датчиков.



В состав робототехнического модуля входят металлические детали, представляющие собой основные конструктивные элементы робота, предназначенные для сборки различных конструкций и механизмов.



Соединение комплектующих модуля осуществляется с помощью различных крепёжных элементов, а также зубчатых передач. С помощью комплекта зубчатых передач становится возможным создавать подвижные механизмы, выполняющие различные движения робота.



Стоит отметить, что все комплектующие представлены в трёхмерном формате для моделирования и разработки конструкций в специализированных конструкторских программах. Непосредственно со стороны производителя предлагается к применению программное обеспечение компании Autodesk, которое распространяется свободно для применения в учебных и образовательных целях.



С помощью профессионального программного обеспечения можно проектировать различные конструкции роботов, проверять возможность сборки различных элементов, моделировать конструкции роботов на возможность выполнения движений, производить прочностные расчёты конструкций и выполнять множество других инженерных задач. В том числе одной из важных возможностей подобных систем автоматизированного проектирования является автоматизация процесса подготовки конструкторской документации. На основе разработанных моделей роботов можно создать полный комплект чертежей как для отдельных деталей, так и для сборочных единиц.

Применение систем автоматизированного проектирования при решении инженерных задач является одним из важнейших аспектов подготовки специалистов. С их помощью помимо развития профессиональных навыков в области робототехники данные модули становятся возможным применять в рамках образовательных программ по технологиям, черчению и т.п.

Для того чтобы разрабатываемые конструкции можно было привести в движение, в состав образовательного модуля входят привода на базе двигателей постоянного тока, которые могут оснащаться инкрементными энкодерами в качестве датчиков положения.



Помимо осуществления различных передвижений зачастую перед роботами ставятся задачи манипулирования различными объектами, для этого применяется захватное устройство, которое можно использовать при конструировании манипуляторов или любых других роботов.



Поскольку роботы и робототехнические устройства большинство своих задач должны выполнять максимально автономно, в состав набора входит комплект сенсорных устройств, предназначенных для оценки состояния окружающей среды.



В состав модуля входят следующие датчики:

комплект ИК-датчиков для обнаружения линии, вдоль которой робот должен осуществлять движения;

комплект УЗ-сонаров для обнаружения объектов и определения расстояния до них.

Вышеуказанные датчики могут использоваться как по отдельности, в рамках изучения принципов их функционирования и возможностей применения в робототехнике, так и совместно – для создания сенсорных систем робота, позволяющих ему реагировать на изменение окружающей среды. Благодаря этому становится возможным разрабатывать модели роботов, предназначенные для решения практико-ориентированных задач.

Решение прикладных задач с помощью модуля «Базовый уровень» позволяет повысить качество образовательного процесса за счёт совмещения теории и практики. Применение роботов в соревновательной деятельности и при решении прикладных задач позволяет развивать навыки профессионального проектирования технических сложных систем.

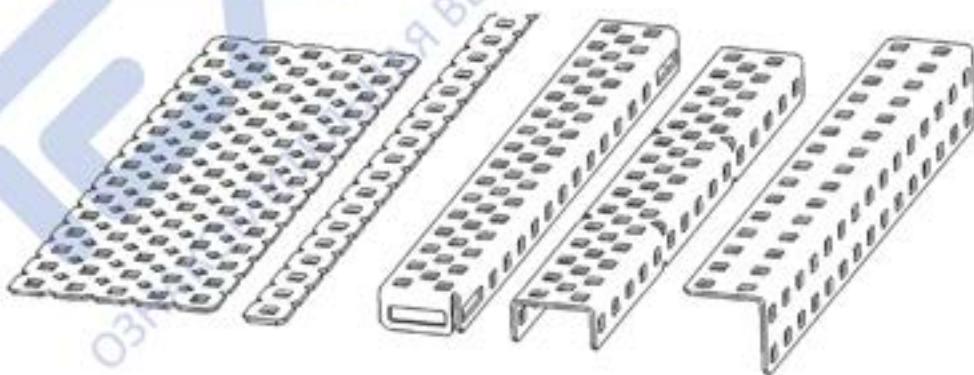


Конструктивные элементы и комплектующие конструкторов VEX

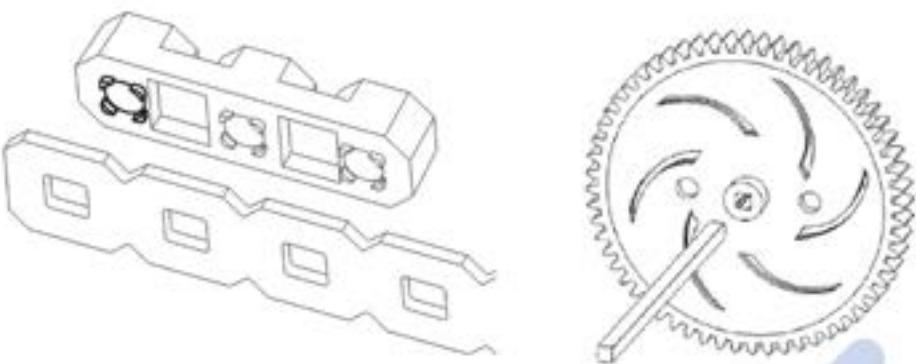
Отличительной особенностью наборов VEX является многообразие различных конструктивных элементов. Конструктивные элементы VEX представляют собой металлические детали и крепёжные элементы, позволяющие разрабатывать на их базе сложные и прочные механизмы.



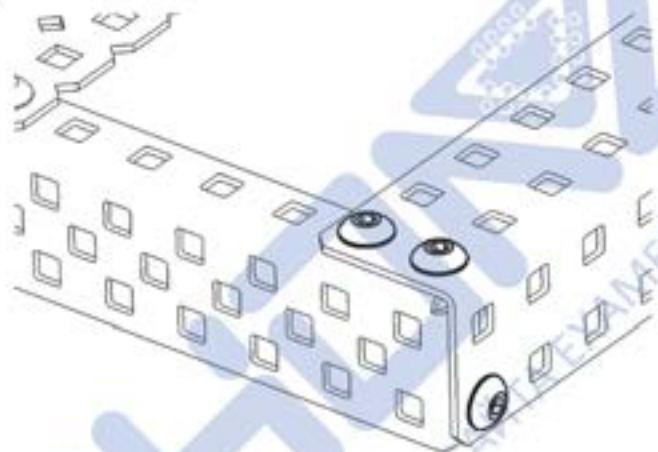
Основными комплектующими VEX являются пластины и уголки из перфорированного алюминия. Благодаря наличию множества отверстий с равным шагом данные детали могут скрепляться друг с другом произвольным образом.



Отличительной особенностью деталей VEX является то, что все отверстия в них квадратной формы. Благодаря этому становится возможным фиксировать положение различных элементов и деталей относительно друг друга.



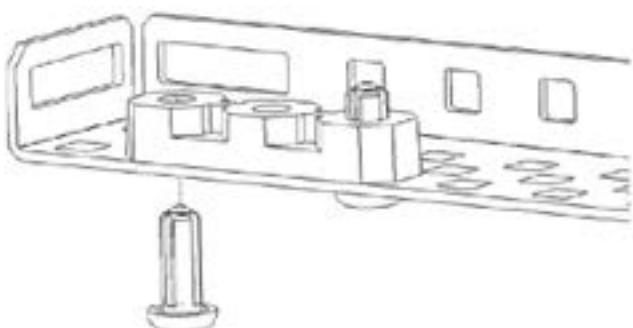
Фиксация элементов осуществляется за счёт специальных бортиков, входящих в углы квадратных отверстий, тем самым задавая ориентацию деталей. В случае закрепления валов или осей квадратные детали прочно и надёжно устанавливаются в соответствующие отверстия.



Соединение деталей между собой осуществляется с помощью резьбовых соединений на основе гаек и винтов различной длины. Все винты имеют шляпку с отверстием под шестигранный инструмент, входящий в робототехнический набор. Фиксация винтов осуществляется с помощью гаек различного типа, как обычных, так и стопорящих.



Также соединение некоторых комплектующих может осуществляться с помощью пластиковых заклепок.



Для закрепления различных конструкций или устройств могут применяться как стандартные детали и крепёжные элементы, так и специализированные стойки. В робототехнический набор VEX входит комплект стоек различной длины.

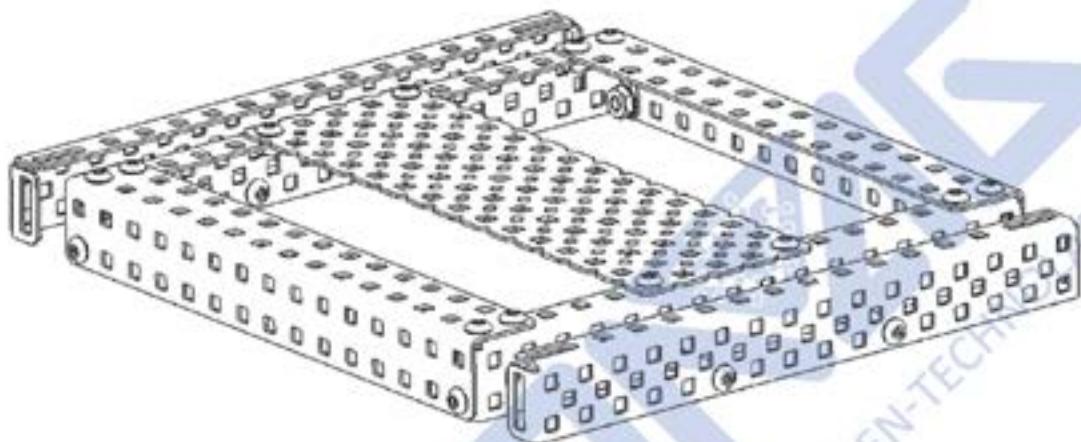


Одной из ключевых особенностей деталей и комплектующих VEX является то, что любая из плоских пластин может быть согнута или разрезана вдоль специальных направляющих отверстий. Благодаря этому можно создавать детали и конструктивные элементы специальной формы, удовлетворяющей проекту конструкции.



С помощью различных компонент и деталей можно сконструировать модели роботов различных габаритов и назначения. Соединяя детали между собой, можно создавать как статичные конструкции, так и подвижные механизмы.

Примечание: При разработке конструкций роботов и прочих механизмов не забывайте о жёсткости конструкции в целом. Для упрочнения конструкции необходимо использовать как стандартные уголки и рёбра жёсткости, так и специализированные элементы.



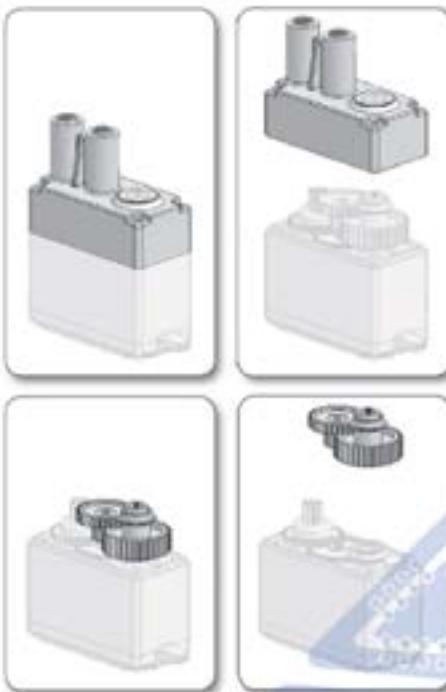
Исполнительные механизмы конструкторов VEX

Разработка сложных робототехнических устройств и подвижных механизмов помимо надёжных конструктивных элементов требует наличия специализированных исполнительных механизмов, таких как: привода, системы линейного перемещения и элементы зубчатых передач.



В базовый робототехнический набор VEX входят привода и сервопривода на базе двигателей постоянного тока. С помощью данных устройств можно разрабатывать подвижные механизмы и конструкции различного назначения.

Каждый привод представляет собой электромеханическое устройство, состоящее из двигателя постоянного тока, редуктора и системы управления.

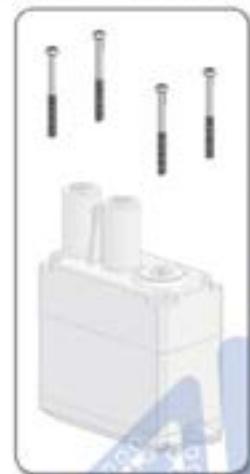
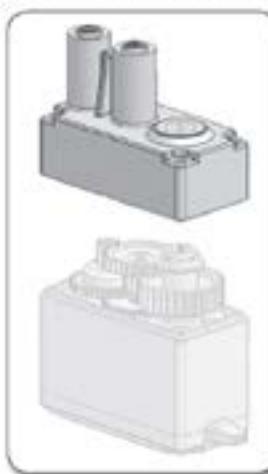


Конструкция каждого из приводов разборная, и пользователь в любой момент может внести изменения в его конструкцию, например, заменить зубчатые колёса редуктора. Замена зубчатых колёс, как правило, осуществляется для изменения передаточного отношения редуктора или с целью ремонта вследствие механической поломки.

Примечание: При сборке элементов механических зубчатых передач соблюдайте соосность валов и старайтесь не допускать перекосов зубчатых колёс.



После установки или замены зубчатых колёс необходимо установить крышку, скрывающую механические передачи привода. Крышка обладает специальной конструкцией, благодаря которой привод можно крепить к различным механическим элементам.

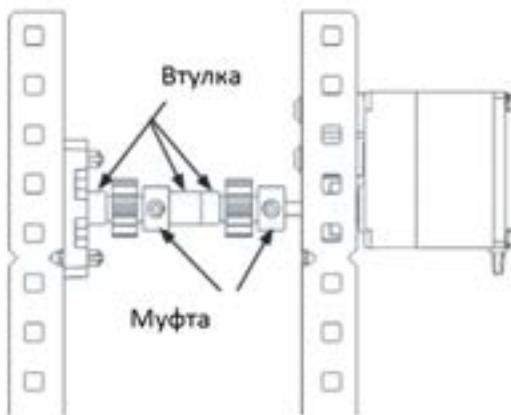


Установка привода осуществляется с помощью двух цилиндрических стоек, расположенных на его крышке. Каждая из стоек обладает посадочным фланцем, предназначенным для фиксации в квадратных отверстиях деталей и пластин конструктора VEX. Фиксирование устройства осуществляется с помощью крепёжных винтов.

В отличие от большинства аналогичных приводов и сервоприводов, применяемых в робототехнических конструкторах, устройства из базовых робототехнических наборов VEX дают возможность пользователю частично изменять их конструкцию, например, на привода можно устанавливать датчики, определяющие скорость вращения и положение вала; также можно изменять внутреннюю конструкцию редуктора, выходного вала привода и т.п.



Конструкция приводов позволяет пользователю передавать с них вращение на удалённые механизмы с помощью валов различной длины. Вал может быть установлен напрямую в привод, а также может быть закреплён с помощью специальной муфты, ограничивающей передаваемый момент и препятствующей поломке привода или механизма в случае заклинивания.



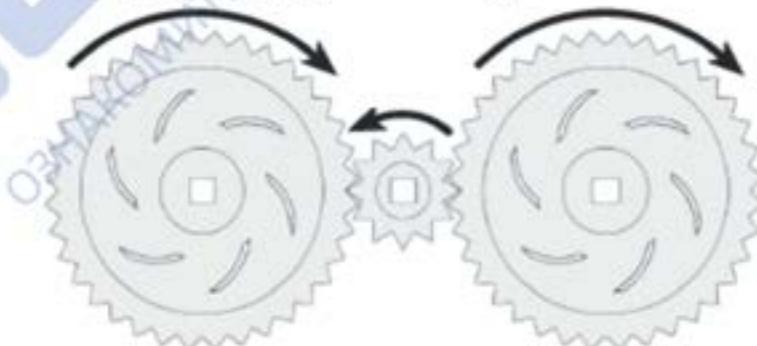
Помимо сменных валов привода могут передавать вращения на различные механизмы, устанавливаемые на валы с помощью фиксирующих втулок. Таким образом можно разрабатывать различные конструкции и механизмы, состоящие из валов и зубчатых передач.

Комбинируя зубчатые колеса различного диаметра, можно конструировать механизмы с различным передаточным отношением, тем самым изменяя скорость вращения и передаваемый ими момент.

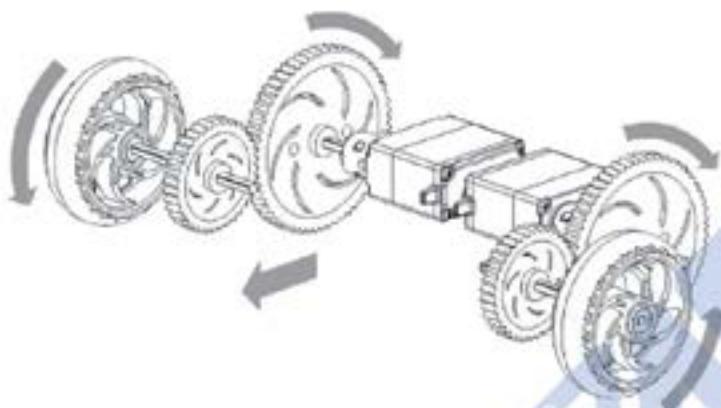
Ведущее колесо Ведомое колесо



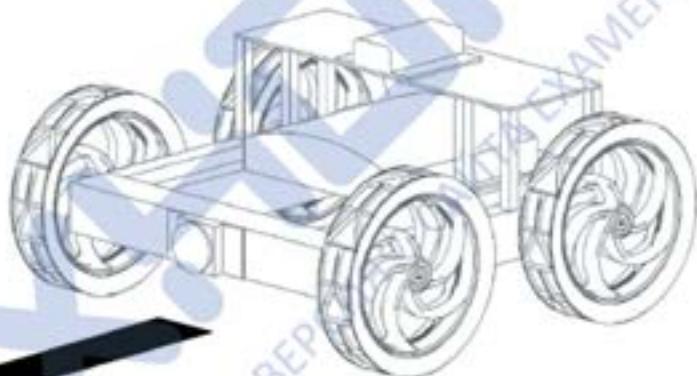
Ведущее колесо Ведомое колесо



Используя различные зубчатые передачи, можно конструировать все возможные механизмы роботов – гусеничные и колёсные шасси, поворотные основания и привода качения и т.п.



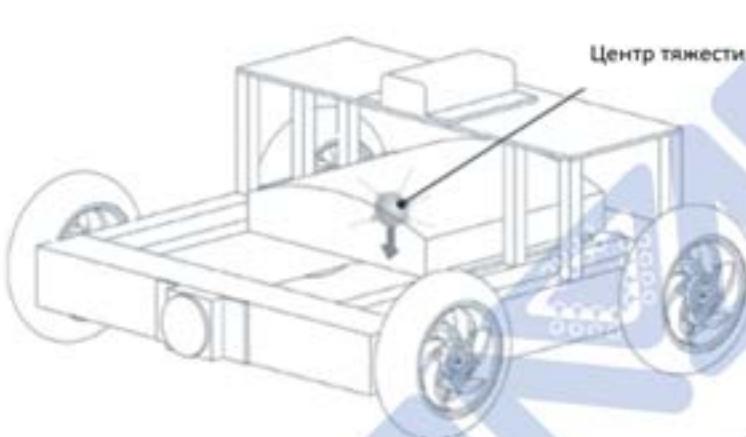
Комплектующие VEX позволяют конструировать различных роботов, решающих широкий спектр задач, начиная от образовательных, исследовательских и соревновательных, вплоть до прикладных задач, решаемых профессиональными роботами.



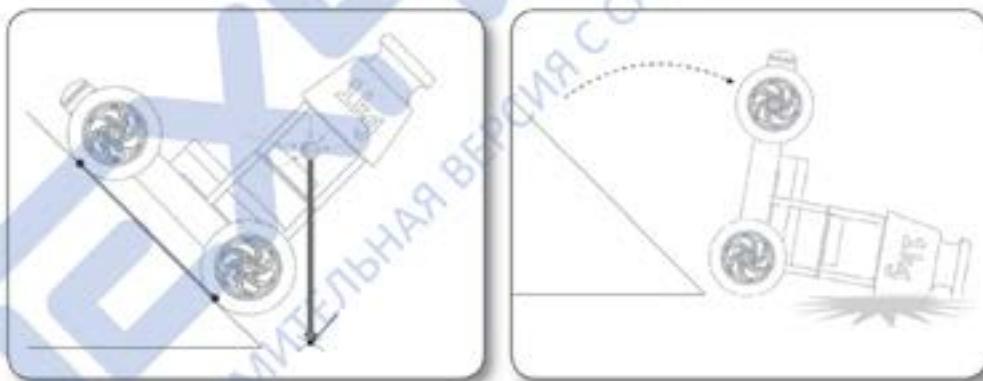
ОЗНАКОМИТЕЛЬНАЯ ВЕРСИЯ

Базовые принципы проектирования роботов

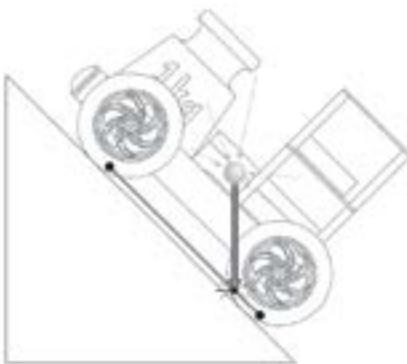
Робототехнические конструкторы VEX позволяют конструировать достаточно технически сложных роботов, предназначенных для решения сложных задач. В зависимости от сложности робота и решаемых им задач все больше внимания следует уделять надёжности роботов, прочности их конструкций и т.п.



Одно из основных требований к мобильным роботам – это сохранение их устойчивости в процессе движения или работы. В процессе проектирования следует уделять внимание балансировке механизмов робота и равномерному распределению нагрузки по всей конструкции.

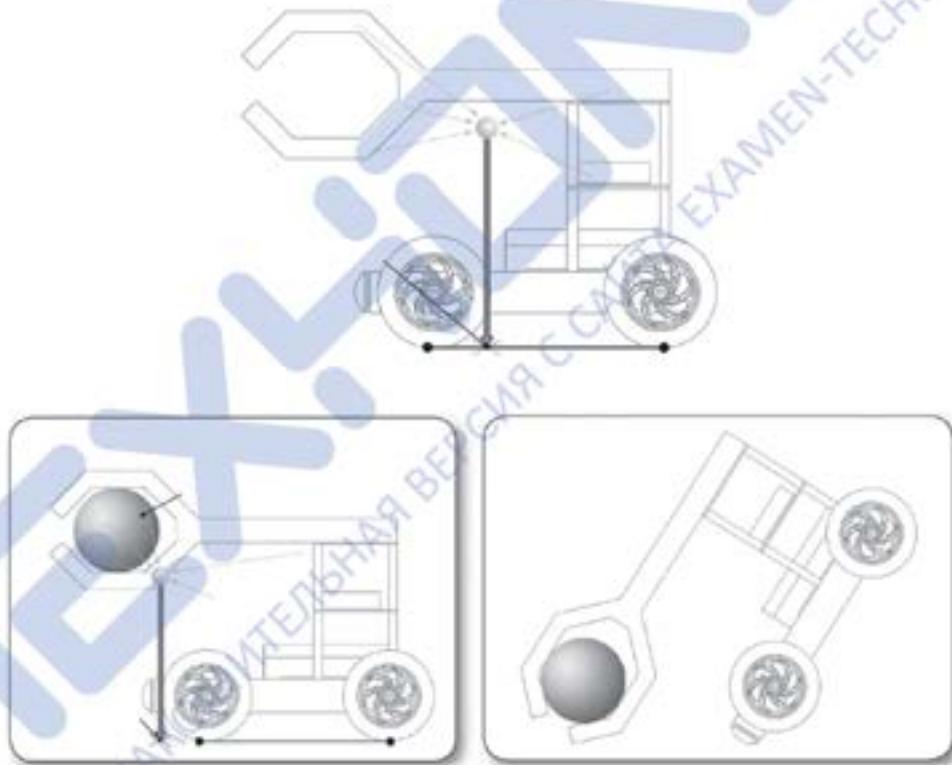


В зависимости от решаемых роботом задач необходимо проектировать конструкцию таким образом, чтобы её особенности не препятствовали выполнению основных функций робота.



Например, при перемещении робота по наклонным поверхностям необходимо смещать центр тяжести робота как можно ниже к основанию и к его центру, чтобы препятствовать возможному опрокидыванию при подъёме.

При проектировании роботов, оснащённых захватным устройством, необходимо учитывать возможное смещение центра тяжести робота при манипулировании объектами.

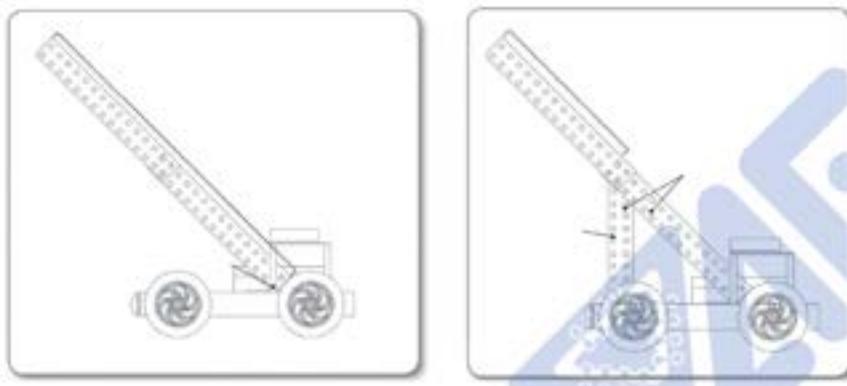


В подобной ситуации следует усовершенствовать либо конструкцию робота, либо умышленно смещать его центр тяжести с целью балансировки конструкции.

Следует уделять повышенное внимание вопросам распределения нагрузки равномерно по всему роботу. Плохо сбалансированная конструкция робота подвержена излишним нагрузкам и имеет склонность к заносам и опрокидываниям. Одним из наи-

более действенных приёмов балансировки роботов является распределение аккумуляторных батарей по шасси так, чтобы они уравновешивали конструкцию необходимым образом.

При проектировании роботов часто возникают ситуации, когда конструкция или механизмы робота обладают достаточно большими габаритами, в результате чего его собственная конструкция может быть неустойчивой или неуравновешенной.



Для того чтобы конструкция робота сохраняла жёсткость и устойчивость, необходимо устанавливать рёбра жёсткости, поддерживающие основные конструктивные элементы.

При проектировании робота необходимо достичь сохранности всех его внутренних узлов и устройств в процессе работы. Необходимо следить за тем, чтобы никакие управляющие электронные устройства не выступали за габариты робота, чтобы не допустить их повреждения. Также важно следить за тем, чтобы соединяющие кабели и шлейфы не перетирались в процессе движение робота или его механизмов.

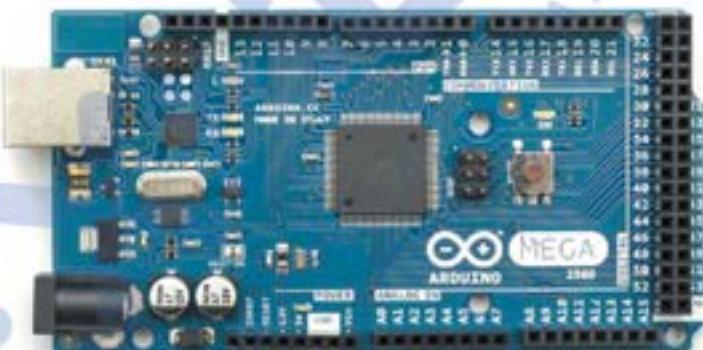
Помните, что в независимости от сложности робота и решаемых им задач, процесс проектирования робота должен быть одинаково ответственным. Однако при росте сложности и числа решаемых задач, возлагаемых на робота, необходимо учитывать как можно больше влияющих факторов и заранее прогнозировать результаты работы проектируемого робота. Подобные навыки проектировщиков развиваются исключительно с ростом их опыта работ в конкретной области. Образовательный робототехнический модуль «Базовый уровень» содержит в себе всё необходимое для развития профессиональных навыков проектирования роботов и робототехнических систем.

Программируемый контроллер

В состав робототехнического модуля «Базовый уровень» входит программируемый контроллер, преемственный контроллерам семейства Arduino. Контроллеры семейства Arduino представляют собой программно-аппаратный комплекс на базе платы ввода/вывода с использованием микроконтроллеров семейства AVR (Atmel), а также специальной среды программирования на основе языка C/C++.

Отличительная особенность контроллеров семейства Arduino заключается в том, что каждый контроллер данного семейства обладает унифицированным расположением портов и единым стандартом разъёмов, встроенной схемой электропитания, позволяющей использовать большинство распространённых аккумуляторных батарей, а также встроенными средствами для программирования. Несмотря на небольшие базовые функциональные возможности, контроллеры семейства Arduino обладают многообразием дополнительных плат расширения (Shield), позволяющих подключать к базовой плате различные привода, датчики, устройства связи и мультимедийные системы.

Благодаря этому контроллеры семейства Arduino на сегодняшний день являются одними из наиболее популярных устройств для быстрого prototyping инженерных проектов.

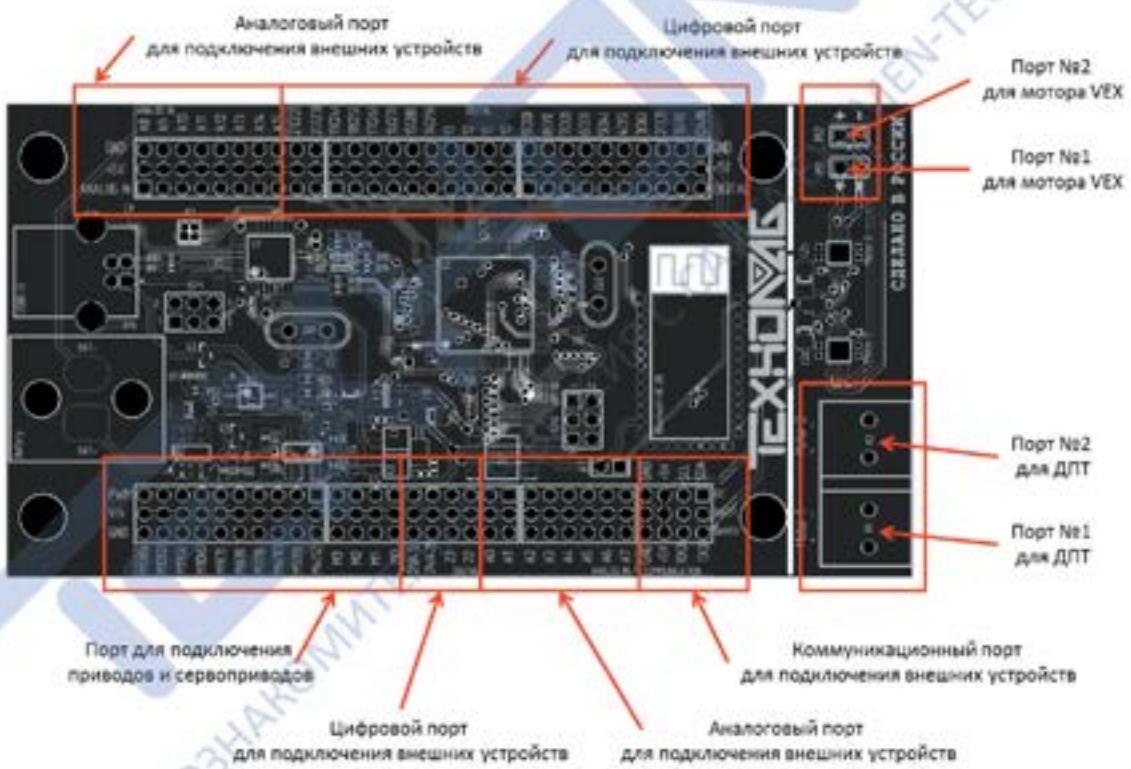


Семейство контроллеров Arduino выпускается с открытой документацией, благодаря чему любой пользователь или производитель может создать собственный контроллер или аксессуар, преемственный к оригинальным контроллерам Arduino. Единственное ограничение касается торговой марки Arduino – вновь созданные проекты имеют право указывать на собственную преемственность Arduino, но не имеют права использовать это название в своей продукции.

На сегодняшний день в мире существует множество различных Arduino-подобных контроллеров, каждый из которых копирует оригинальные платы. В силу ограниченности функциональных возможностей плат Arduino без дополнительных внешних устройств они не могут в полной мере обеспечить все потребности разработчиков роботов. Чаще всего при разработке роботов возникает необходимость подключения силовой нагрузки (чаще всего двигателя), обеспечения контроля за уровнем питания, а также обеспечения беспроводной связи с внешними элементами систем управления робота.

В состав образовательного робототехнического модуля «Базовый уровень» входит программируемый контроллер, преемственный контроллерам семейства Arduino и специально созданный для применения с робототехническими конструкторами VEX.

Программируемый контроллер является преемственным по отношению к оригинальной плате Arduino Mega 2560, но в отличие от неё обладает встроенной системой контроля заряда внешней аккумуляторной батареи, интегрированной силовой частью для подключения внешних двигателей постоянного тока, встроенным модулем беспроводной связи Bluetooth, а также полной совместимостью с комплектующими VEX.



Программируемый контроллер содержит:

- 1) 2 силовых порта (с аппаратной поддержкой PWM) для подключения двигателей постоянного тока (ДПТ) и параллельные им порты для подключение моторов, входящих в состав робототехнического конструктора VEX.

- 2) 26 цифровых портов ввода/вывода (с нумерацией от 0 до 25) для работы с внешними дискретными сигналами.
- 3) 16 аналоговых портов (с нумерацией от 0 до 15) для взаимодействия с внешними аналоговыми устройствами и сигналами.
- 4) 14 портов (10 портов с аппаратным PWM и 4 с GPIO) для подключения внешней нагрузки, приводов/моторов и сервоприводов робототехнической платформы VEX EDR.
- 5) 2 коммуникационных порта для взаимодействия по последовательному интерфейсу UART.
- 6) 1 коммуникационный порт для взаимодействия по последовательному интерфейсу I2C.
- 7) 1 встроенный контроллер интерфейса для беспроводной связи Bluetooth.
- 8) 1 встроенный USB порт для программирования контроллера и передачи данных (UART).
- 9) 1 встроенный порт для подключения внешнего питания и схему контроля состояния заряда аккумуляторной батареи.

Все порты программируемого контроллера преемственны портам оригинальной платы Arduino. Нумерация портов программируемого контроллера выполнена по мере увеличения их порядкового номера, но нумерация не всех портов совпадает с нумерацией оригинальной платы. В случае если нумерация портов совпадает, то порты маркируются идентично, например, как в случае с аналоговыми портами программируемого контроллера и оригинальной платы Arduino. Если же нумерация портов не совпадает, то при маркировке порта программируемого контроллера в скобках указывается соответствующая маркировка порта оригинальной платы Arduino.

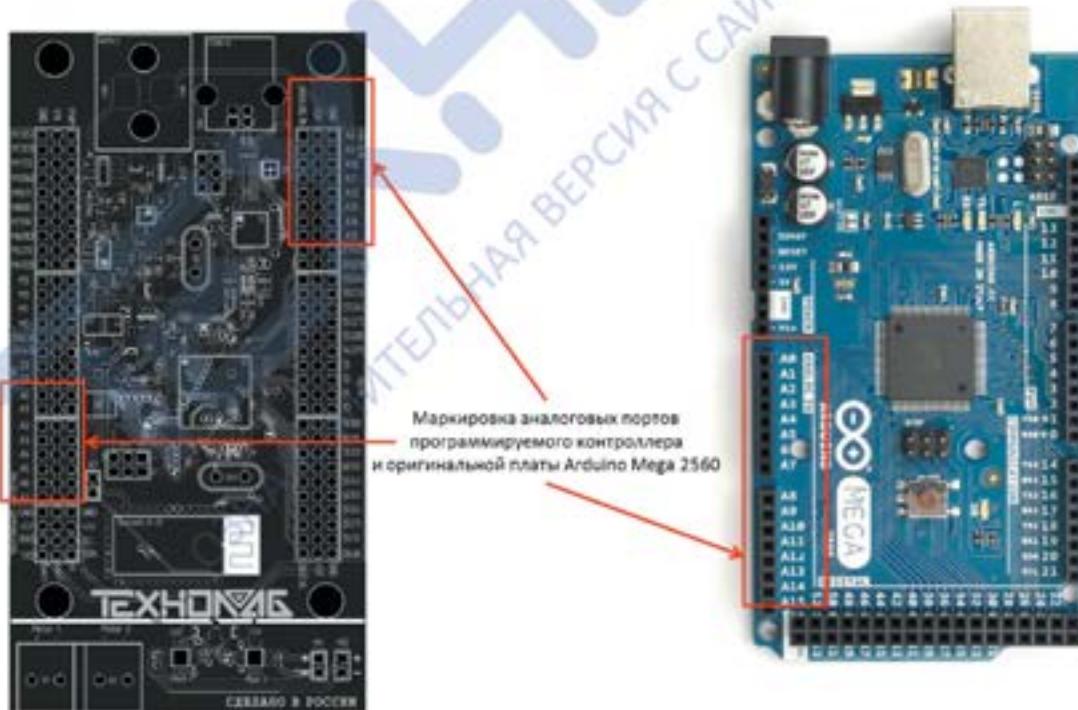


Таблица соответствия маркировки портов

| Тип порта | Маркировка портов | | Примечания |
|-------------------|----------------------------|-------------------|--|
| | Программируемый контроллер | Arduino Mega 2560 | |
| Коммуникационный | UART1 | UART1 | Четырёхпиновый разъём с питанием +5В и GND |
| | UART3 | UART3 | |
| Коммуникационный | I2C | I2C | Четырёхпиновый разъём с питанием +5В и GND |
| Аналоговый AIN | 0-15 | 0-15 | Трёхпиновый разъём с питанием +5В и GND |
| Цифровой GPIO | 0 | 48 | Трёхпиновый разъём с питанием +5В и GND |
| | 1 | 49 | |
| | 2 | 37 | |
| | 3 | 36 | |
| | 4 | 35 | |
| | 5 | 34 | |
| | 6 | 33 | |
| | 7 | 32 | |
| | 8 | 31 | |
| | 9 | 30 | |
| | 10 | резерв (PJ3) | |
| | 11 | резерв (PJ4) | |
| | 12 | резерв (PJ5) | |
| | 13 | резерв (PJ6) | |
| | 14 | 29 | |
| | 15 | 28 | |
| | 16 | 27 | |
| | 17 | 26 | |
| | 18 | 25 | |
| | 19 | 24 | |
| | 20 | 23 | |
| | 21 | 22 | |
| | 22 | резерв (ICP1) | |
| | 23 | резерв (ICP3) | |
| | 24 | 42 | |
| | 25 | 43 | |

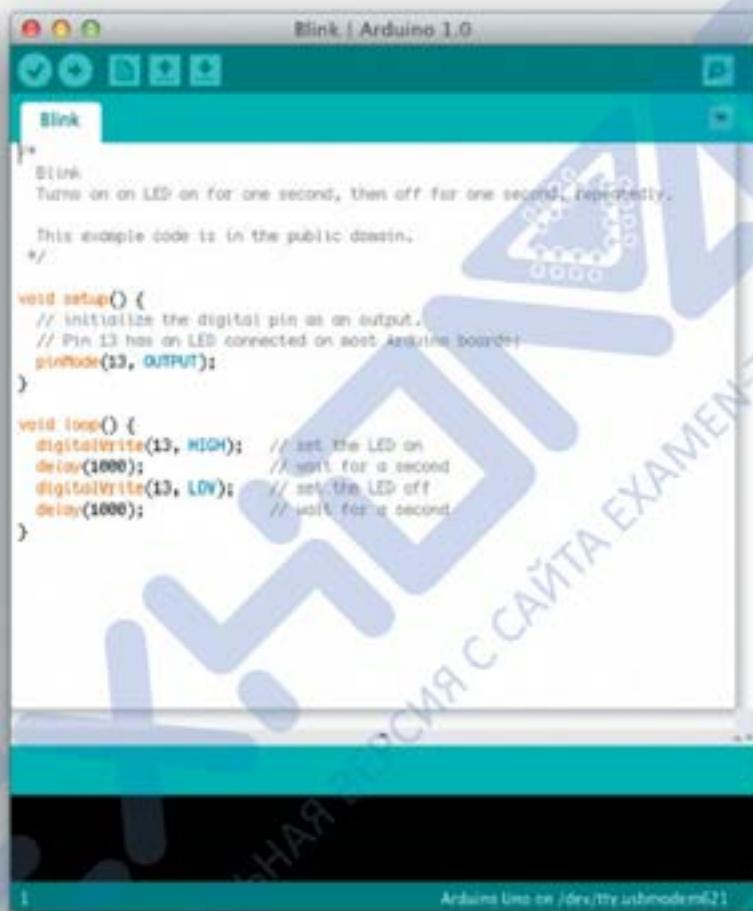
| | | | |
|-----------------------|-----------|-------|---|
| Цифровой PWM | M 0 | 38 | Трёхпиновый разъём с питанием +5В и GND |
| | M 1 | 44 | |
| | M 2 | 45 | |
| | M 3 | 46 | |
| | M 4 | 12 | |
| | M 5 | 11 | |
| | M 6 | 10 | |
| | M 7 | 9 | |
| | M 8 | 8 | |
| | M 9 | 7 | |
| | M 10 | 6 | |
| | M 11 | 3 | |
| | M 12 | 2 | |
| | M 13 | 5 | |
| Силовой порт M1 | M1 + | 6 | Двухпиновый разъём VCC и GND |
| | M1 - | 7 | |
| Силовой порт M2 | M2 + | 46 | Двухпиновый разъём VCC и GND |
| | M2 - | 45 | |
| Коммуникацион- ный | Bluetooth | UART2 | Беспроводной интер- фейс |

Для взаимодействия с дополнительной вычислительной платой часть портов зарезервированы как системные. Соответствующие цифровые порты отмечены в таблице соответствия маркировки портов.



Основы работы в Arduino IDE

Для программирования контроллеров Arduino и совместимых с ними устройств применяется специализированная среда Arduino IDE. Arduino IDE представляет собой совокупность встроенного текстового редактора программного кода, области вывода системных сообщений, области вывода системной информации в виде консоли, панели инструментов и меню.



Программы, написанные в Arduino IDE, называются скетчами (sketch) и разрабатываются в окне текстового редактора. Во время редактирования скетча, сохранения или экспортования в области сообщений появляются соответствующие текстовые уведомления, а также могут появляться возникшие ошибки. Область вывода системной информации (консоль) предоставляет пользователю полные отчеты об ошибках. С помощью кнопок панели инструментов можно выполнить базовые операции с текстом программы:



Verify/Compile

Компилирование программы и проверка программного кода на ошибки

Stop

Остановка мониторинга и прочих процессов.

New

Создание нового скетча.

Open

Открытие меню доступа ко всем скетчам в блокноте. Открывается нажатием в текущем окне.

Примечание: из-за наличия ошибки в Java данное меню не может прокручиваться; при необходимости открыть скетч из этого списка проследуйте в меню File | Sketchbook.

Save

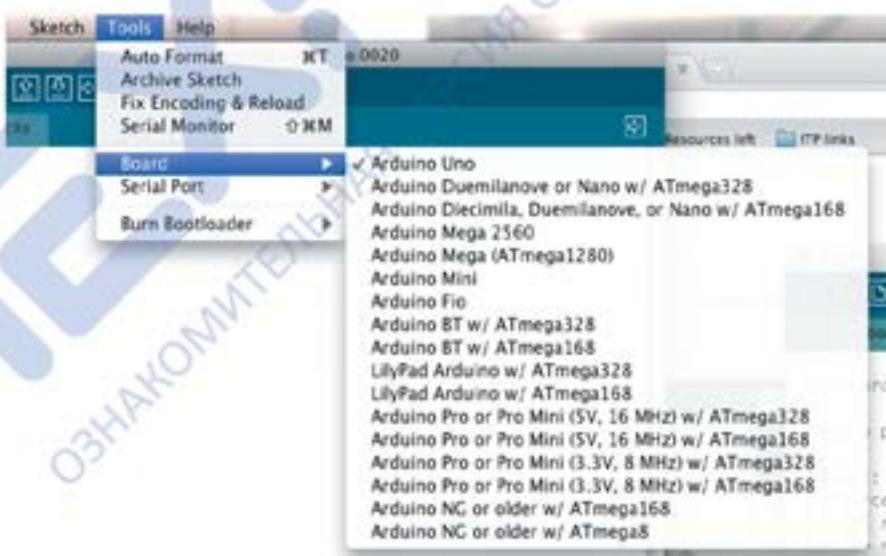
Сохранение скетча.

Upload to I/O Board

Компилирование программного кода и загрузка в программируемый контроллер.

Serial Monitor

Открытие консоли для мониторинга состояния устройства.



Дополнительные команды сгруппированы в панели: File, Edit, Sketch, Tools, Help.

Edit*Copy for Discourse*

Копирование в буфер обмена кода скетча.

Copy as HTML

Копирует код скетча в буфер обмена как HTML код, для размещения на веб-страницах.

Sketch*Verify/Compile*

Проверка скетча на ошибки.

Import Library

Добавление в скетч библиотеки.

Show Sketch Folder

Открытие папки, в которой располагается скетч.

Add File...

Добавление файла в скетч.

Tools*Auto Format*

Автоматическое форматирование кода и оптимизация отображения кода скетча.

Board

Выбор типа программируемого контроллера. В данном случае выбирается Arduino Mega 2560.

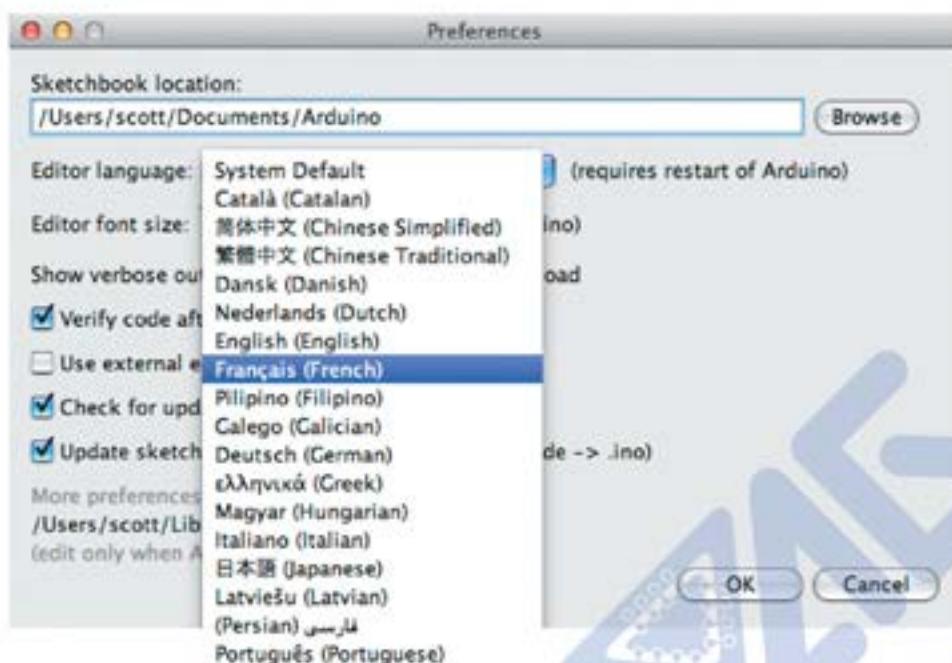
Serial Port

Список устройств, взаимодействующих по последовательному интерфейсу.

Burn Bootloader

Позволяет загрузить базовую программу ЗАГРУЗЧИК (Bootloader) в микроконтроллер платы Arduino.

Среда программирования Arduino IDE хранит скетчи в специальном разделе *Sketchbook* (блокнот). При первом запуске Arduino IDE сразу же создается блокнот для хранения будущих скетчей, расположение которого можно определить в окне *Preferences*.



Для загрузки скетча в программируемый контроллер необходимо выбрать тип программируемого контроллера и номер последовательного порта, к которому подключена плата, с помощью команд **Tools > Board** и **Tools > Serial Port**. После чего загрузка скетча осуществляется с помощью команды **File > Upload to I/O Board**. Большинство контроллеров перезагружаются перед загрузкой программы, но если этого не произошло, необходимо нажать кнопку ручной перезагрузки.

Программирование контроллеров Arduino

Программирование контроллеров осуществляется на специальном языке основанном на C/C++. Поэтому язык программирования оперирует базовыми понятиями языков C/C++.

| Операторы | Данные | Функции |
|--|---|---|
| setup() | <u>Константы</u> HIGH LOW INPUT OUTPUT true false <u>Целочисленные константы</u> Константы с плавающей запятой | <u>Цифровой ввод/вывод</u> pinMode() digitalWrite() digitalRead() |
| loop() | <u>Тип данных</u> boolean char byte int unsigned int word long unsigned long float double string - массив символов String - объект класса массив (array) void | <u>Аналоговый ввод/вывод</u> analogRead() analogReference() analogWrite() |
| <u>Управляющие операторы</u> if if...else for switch case while do... while break continue return goto | | <u>Функции ввода/вывода</u> tone() noTone() shiftOut() pulseIn() |
| <u>Синтаксис</u> ; (semicolon) { } (curly braces) // (single line comment) /* */ (multi-line comment) | | <u>Работа со временем</u> millis() micros() delay() delayMicroseconds() |

| <u>Арифметические операторы</u> | <u>Преобразование типов данных</u> | <u>Математические функции</u> |
|---------------------------------|-------------------------------------|-------------------------------|
| = (assignment) | char() | min() |
| + (addition) | byte() | max() |
| - (subtraction) | int() | abs() |
| * (multiplication) | long() | constrain() |
| / (division) | float() | map() |
| % (modulo) | | pow() |
| <u>Операторы сравнения</u> | <u>Область видимости переменных</u> | <u>sq()</u> |
| == (equal to) | static | sqrt() |
| != (not equal to) | volatile | sin() |
| < (less than) | const | cos() |
| > (greater than) | | tan() |
| <= (less than or equal to) | | randomSeed() |
| >= (greater than or equal to) | | random() |
| <u>Логические операторы</u> | | attachInterrupt() |
| && (И) | | detachInterrupt() |
| (ИЛИ) | | Serial |
| ! (Отрицание) | | |
| <u>Унарные операторы</u> | | |
| ++ (increment) | | |
| -- (decrement) | | |
| += (compound addition) | | |
| -= (compound subtraction) | | |
| *= (compound multiplication) | | |
| /= (compound division) | | |

Подробное описание конструкций языка программирования можно найти в электронной документации в меню СПРАВКА – Содержание.

Для использования дополнительных функциональных возможностей применяются библиотеки. Библиотеки добавляют дополнительную функциональность скетчам, например, при работе с аппаратной частью или при обработке данных. Для использования библиотеки необходимо выбрать меню **Sketch > Import Library**. Одна или несколько директив **#include** будут размещены в начале кода скетча с последующей компиляцией библиотек и вместе со скетчем. Загрузка библиотек требует дополнительного места в памяти Arduino. Неиспользуемые библиотеки можно удалить из скетча, убрав директиву **#include**.

На Arduino.cc имеется список библиотек. Некоторые библиотеки включены в среду разработки Arduino. Другие могут быть загружены с различных ресурсов. Для установки скачанных библиотек необходимо создать директорию **libraries** в папке блокнота и затем распаковать архив. Например, для установки библиотеки **DateTime** её файлы должны находиться в подпапке **/libraries/DateTime** папки блокнота.

Стандартные библиотеки:

EEROM – чтение и запись энергонезависимой памяти контроллера

Ethernet – для соединения по Интернет с помощью Arduino Ethernet Shield

Firmata – для взаимодействия с устройствами с помощью serial protocol

LiquidCrystal – для управления ЖК-экранами (LCDs)

SD – для чтения SD-карт

Servo – для управления сервоприводами

SPI – для взаимодействия с устройствами с помощью Serial Peripheral Interface (SPI) Bus

SoftwareSerial – для коммуникации с помощью последовательного интерфейса с помощью цифровых портов

Stepper – для управления шаговыми моторами.

Wire – Two Wire Interface (TWI/I2C) для управления устройствами и передачи данных.

Заключение

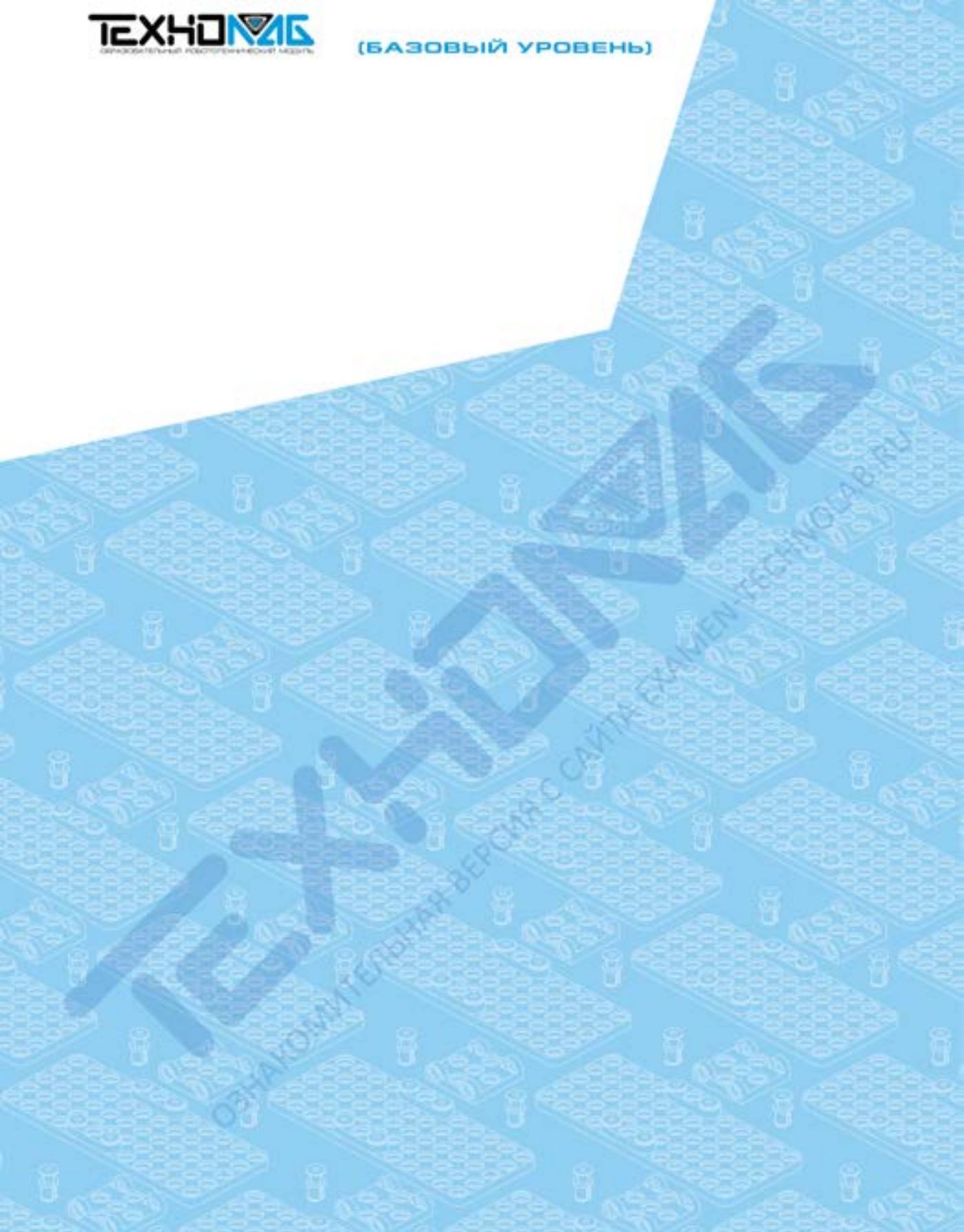
Программирование контроллеров типа Arduino является творческой задачей, требующей подготовки и базовых знаний в электронике и языках программирования. Необходимый для этого объём информации достаточно велик и не может быть приведён в инструкции по эксплуатации к базовому робототехническому набору. Поэтому в случае возникновения проблем, связанных с программированием робототехнического модуля, следует обратиться к электронной документации по контроллерам Arduino на официальном сайте производителя.



Работа с основными устройствами и комплектующими

Приложение 1





Приложение 1.

Работа с основными устройствами и комплектующими

Программируемый контроллер является Arduino-подобной платой, но обладает рядом отличительных особенностей, связанных: с расположением и нумерацией разъёмов, наличием системы контроля заряда батареи, наличием дополнительных выводов для подключения силовой нагрузки, а также наличием встроенного модуля Bluetooth. Однако, несмотря на различия, выдержаны все основные стандарты по подключению типовых интерфейсов: UART, I2C и т.д., благодаря этому сохранилась совместимость платы со средой Arduino IDE, всеми стандартными библиотеками и большинством сторонних библиотек.

Питание программируемого контроллера может осуществляться двумя путями:

1. Путём подключения платформы с помощью USB провода к персональному компьютеру;
2. Путём использования входящего в комплект аккумулятора VEX 7.2V.

В случае питания контроллера от персонального компьютера не допускается подключение к плате силовой нагрузки, а также двигателей. Для использования двигателей необходимо подключение внешнего аккумулятора.

При подключении внешнего аккумулятора индикатор показывает состояние аккумулятора:

- горит непрерывно – все в порядке;
- мигает – аккумулятор разряжен, питание платформы не происходит.

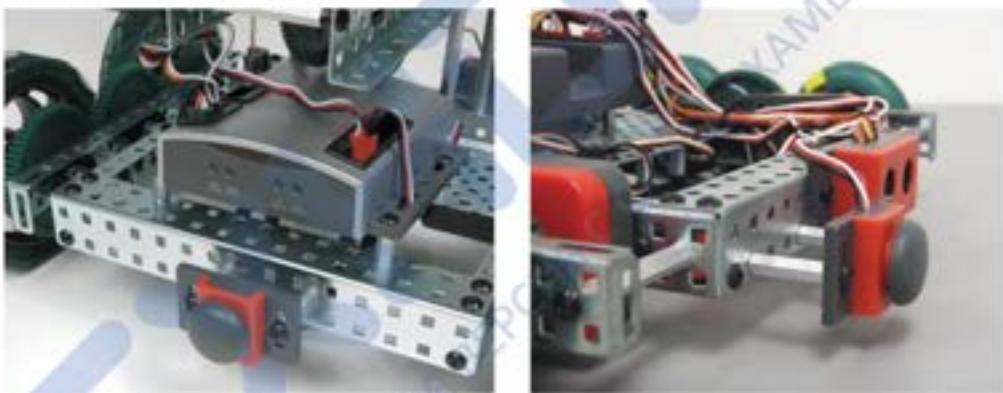


Пример подключения и работы с тактильными датчиками, концевыми выключателями и кнопками

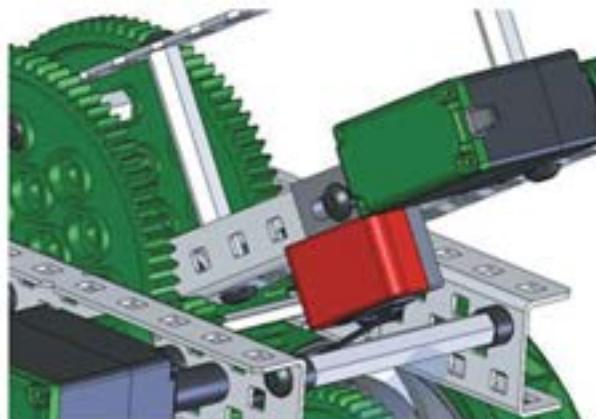
Тактильные датчики или концевые переключатели являются одним из простейших типов датчиков, представляющих собой механический переключатель, замыкающий или размыкающий цепь управления.



Такие устройства представляют собой наиболее простейшие цифровые датчики, которые генерируют дискретный сигнал в зависимости от собственного состояния. Наиболее часто встречается применение подобных датчиков в качестве кнопок управления или ограничивающих устройств.

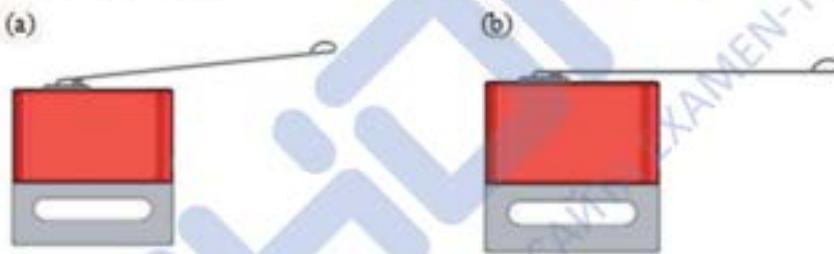


Тактильные датчики чаще всего применяются в качестве кнопок или контактных бамперов безопасности, останавливающих движение робота в запрещённом направлении в случае нажатия на них.



Концевые переключатели применяются чаще всего для срабатывания в случае со-прикосновения с различными движущимися частями или механизмами робота, например: в случае ограничения угла поворота манипулятора и т.п.

При использовании концевых выключателей особое внимание необходимо уделять их расположению вблизи подвижных частей механизма. Датчик должен быть установлен таким образом, чтобы в требуемом положении на него оказывалось необходимое давление для срабатывания.



Примечание: Для срабатывания концевого выключателя необходимо развить усилие, эквивалентное массе 0,25 г, на расстоянии 2 см от оси вращения.

Каждый из рассматриваемых датчиков подключается к цифровым входам программируемого контроллера. Оба датчика идентичны друг другу в плане применения в процессе разработки программы управления. Показания датчиков носят логический характер – «Нажат» или «Не нажат», т.е. «1» или «0».

Подключение подобных датчиков может осуществляться к любому свободному цифровому порту. При нажатии кнопки происходит замыкание контакта на землю, что приводит к падению уровня сигнала на выбранном цифровом порту до низкого логического уровня.

Пример:

Подключим простую кнопку к порту 0(48) таким образом, чтобы белый провод (сигнальный) смотрел внутрь платы.

Напишем в среде Arduino IDE следующий скетч:

```
//указываем порт, к которому подключена кнопка - 0(48)
const int buttonPort = 48;
//объявляем переменную для работы с кнопкой
int buttonState = 0;

void setup(){
    //конфигурируем выбранный порт на чтение
    pinMode(buttonPort, INPUT);
    //инициализируем соединение с ПК по последовательному порту
    //со скоростью 9600 бод
    Serial.begin(9600);
}

void loop(){
    //подаём высокий уровень сигнала в выбранный цифровой порт
    digitalWrite(buttonPort, HIGH);
    //читаем уровень сигнала на выбранном цифровом порту
    buttonState=digitalRead(buttonPort);
    //выводим в последовательный порт сообщение в зависимости от состояния
    //кнопки
    //если не нажата – замыкания на «землю» нет и на порту держится высокий уро-
    //вень
    if (buttonState == HIGH)
        Serial.println("Not pressed");
    else
        //если нажата – происходит замыкание на «землю» и уровень становится низким
        Serial.println("Pressed");
    //задаём паузу между опросами порта в 50 мс
    delay(50);
}
```



После загрузки скетча в плату в окне монитора последовательного порта можно будет увидеть результат опроса кнопки.



Пример подключения и работы с датчиком освещённости

Датчик освещённости предназначен для измерения интенсивности дневного освещения и позволяет определять интенсивность светового потока, благодаря чему можно существенно расширить функциональные возможности роботов.



Датчик освещённости даёт роботу дополнительный источник информации об окружающей среде и позволяет реализовывать алгоритмы автономной работы. Например, датчик освещённости может быть использован для перемещения робота за источником света.



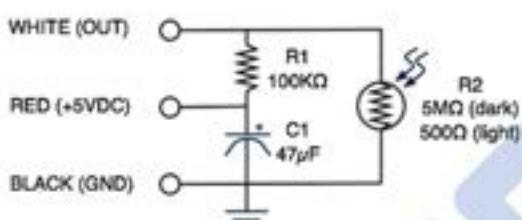
Также датчик освещённости может быть использован в качестве защиты от проникновения робота в труднодоступные места с ограниченной видимостью.

Проявив фантазию, датчик освещённости можно применять для создания алгоритмов энергосбережения заряда аккумулятора, например, для автоматического отключения робота в тёмное время суток и т.п. Если использовать внешние дополнительные световые фильтры, с помощью датчика освещённости становится возможным различать цвета, благодаря чему робот может манипулировать объектами разного цвета.

Датчик освещённости использует в качестве чувствительного элемента фоторезистор на базе материала CdS, который меняет собственное сопротивление в зависимости от интенсивности светового потока.



Vex Light Sensor – 276-2158



Фоторезистор является аналоговым датчиком, плавно изменяющим сопротивление в зависимости от интенсивности светового потока, что влечёт за собой изменения напряжения в диапазоне от 0 до 5В на выходных клеммах датчика освещённости.



Программируемый контроллер робота считывает данные от датчика и после АЦП-преобразования выдаёт результат в виде целого числа в диапазоне от 0 до 1023. Причём значение 1023 соответствует минимальному уровню освещённости (темнота), а значение 0 – максимальному уровню освещённости.



Примечание: В зависимости от разрядности АЦП-преобразования результат измерений датчика может быть: в диапазоне от 0 до 255 в случае 8-разрядного преобразования, в диапазоне от 0 до 1023 в случае 10-разрядного преобразования, в диапазоне от 0 до 4095 в случае 12-разрядного преобразования.

Датчик освещённости реагирует на видимый человеческим глазом свет, а его чувствительность позволяет определять интенсивность светового потока от объекта на расстоянии 1,5 – 2 метра.

Поскольку датчик освещённости является устройством аналогового типа, его следует подключать к аналоговым входам программируемого контроллера. Как и любой датчик, входящий в состав образовательного модуля, данный датчик имеет специальный разъём для подключения к аналоговым портам контроллера, благодаря чему его можно подключать к клеммам контроллера, не опасаясь за полярность.

Пример:

Подключение датчика освещённости выполняется к любому аналоговому порту. Датчик освещённости построен на основе фоторезистора, что позволяет фиксировать не только наличие/отсутствие освещённости, но и определять степень освещённости. Чем ниже освещённость – тем выше напряжение на выходе датчика.

Подключим датчик освещённости к порту A0 таким образом, чтобы белый (сигнальный) провод «смотрел» внутрь платы.

Напишем в среде Arduino IDE следующий скетч:

```
//указывает порт, к которому подключен датчик освещенности:  
int sensorPort = A0;  
//объявляем переменную для работы с датчиком освещенности  
int sensorValue=0;  
  
void setup() {  
//инициализируем соединение с ПК по последовательному порту  
Serial.begin(9600);  
}  
  
void loop() {  
//читываем значения с аналогового порта  
sensorValue = analogRead(sensorPort);  
//выводим в последовательный порт полученные значения  
//с конвертацией в вольты  
Serial.println(sensorValue*0.0049);  
//задаем паузу между опросами порта в 50 мс  
delay(50);  
}
```

После загрузки скетча в плату в окне монитора последовательного порта можно будет увидеть результат опроса датчика освещённости.



Пример подключения и работы с ИК-датчиком линии

Датчик определения линии представляет собой инфракрасный датчик, определяющий интенсивность отражённого от рабочей поверхности света.



Датчик предназначен для определения чёрной линии на белом фоне, благодаря чему робот может автономно перемещаться вдоль неё. Подобная технология достаточно часто встречается в транспортно-логистических системах в цехах производственных предприятий и даёт возможность минимизировать ручной труд водителей транспортных средств.

Датчик состоит из ИК-светодиода и ИК-датчика, реагирующего на интенсивность отражённого света. Принцип функционирования данного датчика основывается на различии отражающих способностей поверхностей разного цвета. Наиболее светлые поверхности отражают падающий на них свет, тёмные поверхности его полностью поглощают.

При проектировании роботов, движущихся вдоль чёрной линии, применяется как минимум два датчика, определяющих положение линии по правому и левому борту робота. Для более точной работы робота следует применять не меньше трёх датчиков, а именно: два датчика по бокам робота и один над направляющей линией.

Подключение детектора линии может быть выполнено как к цифровому, так и к аналоговому порту.

В случае подключения детектора линии к цифровому порту, детектор используется для определения чёрной линии на белом фоне: при попадании на чёрную линию на выходе детектора будет высокий уровень сигнала (есть линия), при попадании на белый фон – низкий уровень (нет линии). Однако, при необходимости можно использовать детектор для различения тёмных оттенков. В таком случае подключение и опрос детектора линии выполняется к аналоговому порту с полной аналогией подключения

и опроса датчика освещённости.

Пример подключения детектора в качестве детектора чёрной линии на белом фоне:

Подключим детектор к порту 0(48) таким образом, чтобы белый провод (сигнальный) смотрел внутрь платы.

Напишем в среде Arduino IDE следующий скетч:

```
//указываем порт, к которому подключен детектор - 0(48)
const int lineTrackerPort = 48;
//объявляем переменную для работы с детектором
int lineTrackerState = 0;

void setup(){
//конфигурируем выбранный порт на чтение
pinMode(lineTrackerPort, INPUT);
//инициализируем соединение с ПК по последовательному порту
//со скоростью 9600 бод
Serial.begin(9600);
}

void loop(){
//читаем уровень сигнала на выбранном цифровом порту
lineTrackerState=digitalRead(lineTrackerPort);
//выводим в последовательный порт сообщение в зависимости от состояния детектора
//если попал на чёрную линию, уровень становится высоким
if (lineTrackerState == HIGH)
Serial.println("White background");
else
//если попал на белый фон, на выходе держится низкий уровень
Serial.println("Black line");
//задаем паузу между опросами порта в 50 мс
delay(50);
}
```



После загрузки скетча в плату в окне монитора последовательного порта можно будет увидеть результат опроса детектора линии.

```
Black line
Black line
Black line
Black line
Black line
Black line
White background
```



Пример подключения и управления моторами

Привод 2-Wire Motor 269 w является двигателем коллекторным постоянного тока с редуктором, состоящим из металлических зубчатых передач. В комплект к двигателю входит набор винтов для его крепления и вал для передачи вращения от двигателя к исполнительному механизму.



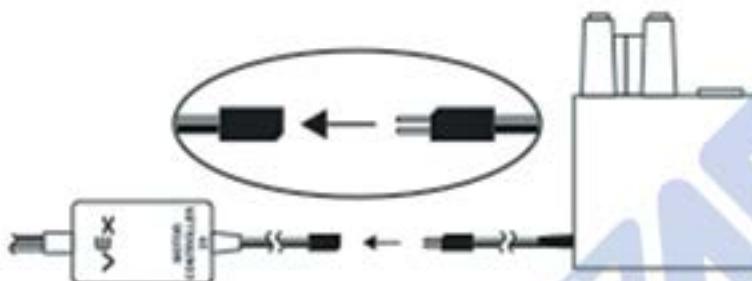
Данный привод является наиболее распространённым в роботах на базе наборов из комплектующих VEX, поскольку Motor 269 обладает наиболее оптимальным соотношением технических характеристик и габаритов, а также может быть оснащён внешним инкрементным энкодером. При стандартном для комплектующих VEX питании в 7,2В данный привод развивает момент 0,97 Нм при скорости до 100 об/мин. Поскольку в процессе работы привод потребляет ток величиной до 2,6А, для подключения его к программируемому контроллеру необходимо применять силовую схему (драйвера двигателя).



Устройство Motor Controller 29 представляет собой силовую схему для управления приводами на базе двигателей постоянного тока. С помощью Motor Controller 29 мож-

но подключать к программируемому контроллеру привода с максимальным рабочим током, не превышающим ЗА. При подключении стандартного привода к программируемому контроллеру становится возможным регулировать скорость его вращения с помощью программируемого значения выходного ШИМ сигнала контроллера.

Для подключения между собой устройства 2-Wire Motor 269 и Motor Controller 29 необходимо соединить с помощью двухпроводного разъёма.



При соединении устройств между собой следует обращать внимание на полярность каждого из них. В случае, если устройства были соединены с нарушением взаимной полярности, вращение привода будет происходить в противоположном от заданного программой направления. Для исправления данной ошибки достаточно осуществить подключение устройств заново или изменить направление вращения привода в программе.



Подключение устройства Motor Controller 29 к программируемому контроллеру осуществляется с помощью специально отведённых портов контроллера для работы с приводами.

Управление приводом осуществляется за счёт изменение мощности на выходном канале ШИМ программируемого контроллера. Изменение сигнала ШИМ осуществляется программным образом, для этого выбирается порт, к которому подключен привод, и ему присваивается значение в диапазоне от -127 до 127, что соответствует максимальным оборотам привода в прямом и обратном направлении.

Пример подключения и управления мотором через драйвер мотора:

Выполним соединение мотора и драйвера мотора таким образом, чтобы чёрный провод мотора совпал с чёрным проводом драйвера, а красный провод драйвера соединялся с красным проводом мотора. Затем подключим 3-штырьковый разъём к порту M13(5) таким образом, чтобы белый (сигнальный) провод «смотрел» внутрь платы.

Напишем в среде Arduino IDE следующий скетч:

```
//подключаем библиотеку для работы с моторами по PWM шине
#include <Servo.h>

//инициализируем наш мотор присвоением ему имени
Servo myservo;

void setup() {
  //указываем порт, к которому подключен мотор M13(5)
  myservo.attach(5);
}

void loop() {
  //задаём ширину импульсов управляющей последовательности в диапазоне
  //1000 - 2000 мс.
  //значение 1500 мс означает остановку двигателя
  myservo.writeMicroseconds(1300);
  //задаём паузу в 2 секунды
  delay (2000);
  //останавливаем мотор
  myservo.writeMicroseconds(1500);
  //задаем паузу в 2 секунды
  delay (2000);
}
```

После загрузки скетча в плату мотор начнёт вращаться в течение 2 секунд с 2-секундовыми перерывами.

Пример подключения управления мотором без использования внешнего драйвера мотора:

Выполним подключение двухпинового разъёма мотора к порту M1. При подключении необходимо соблюдать полярность.

Напишем в среде Arduino IDE следующий скетч:

```
//указываем пины, отвечающие за управление портом M1
const int motorPinOne = 6;
const int motorPinTwo = 7;

void setup() {
//конфигурируем пины порта M1 на вывод сигнала
pinMode(6, OUTPUT);
pinMode(7, OUTPUT);
}

void loop() {
//устанавливаем различные уровни на пинах порта -
//мотор начинает вращение
digitalWrite (motorPinOne, HIGH);
digitalWrite (motorPinTwo, LOW);
//ждём 2 секунды
delay(2000);
//устанавливаем одинаковые уровни на пинах порта -
//мотор останавливается
digitalWrite (motorPinOne, LOW);
digitalWrite (motorPinTwo, LOW);
//ждём 2 секунды
delay(2000);
//меняем уровни на пинах порта на противоположные -
//мотор меняет направление вращения
digitalWrite (motorPinOne, LOW);
digitalWrite (motorPinTwo, HIGH);
//ждём 2 секунды
delay(2000);
//устанавливаем одинаковые уровни на пинах порта -
//мотор останавливается
digitalWrite (motorPinOne, LOW);
digitalWrite (motorPinTwo, LOW);
//ждём 2 секунды
delay(2000);
}
```

После загрузки скетча в плату мотор начинает вращение, затем через 2 секунды останавливается на 2 секунды и затем снова начинает вращение, но в другую сторону. Затем через 2 секунды он снова останавливается и снова меняет направление своего вращения.

Пример подключения и управления сервоприводом

Сервопривод является специализированным устройством, предназначенным для осуществления точных перемещений исполнительных механизмов роботов и робототехнических устройств.

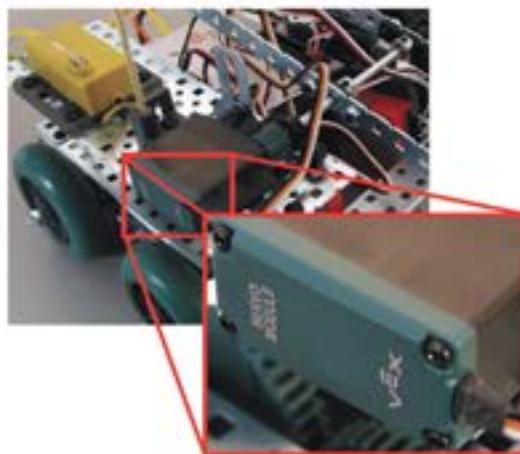


Сервопривод является устройством на базе привода постоянного тока и схемы управления, определяющей положение выходного вала с помощью датчика обратной связи. Чаще всего в качестве датчиков обратной связи применяются потенциометры, имеющие ограниченный угол поворота, в данном случае рабочий угол сервопривода лежит в пределах 100 градусов.

Управление положением выходного вала сервопривода осуществляется с помощью ШИМ-сигнала, т.е. угол поворота привода изменяется пропорционально частоте генерации ШИМ. Трёхпроводный интерфейс сервопривода содержит: чёрный провод – «земля», оранжевый провод – «питание», белый провод – ШИМ. Благодаря этому сервопривод подключается к программируемому контроллеру с помощью трёхпроводных разъёмов, как обычный привод.

В процессе работы сервопривод может развивать момент до 0,73 Нм при рабочем токе в диапазоне от 20mA-1,5A. Вращение сервопривода возможно как в прямом, так и в обратном направлении в пределах рабочего угла.

Обычно сервоприводы применяются в конструкциях и механизмах, где необходимо осуществлять точные перемещения в ограниченном диапазоне, например, захватные устройства механизмов, манипуляторы, поворотные основания и т.п.



Способ присоединения сервопривода идентичен обычному приводу, поэтому они могут применяться в одинаковых конструкциях или заменять друг друга.

Управление сервоприводом сводится к заданию его конечной координаты в пределах допустимого диапазона от -127 до 127.

Пример подключения и управления сервоприводом:

Подключим сервопривод к порту M13(5) таким образом, чтобы белый (сигнальный) провод смотрел внутрь платы.

Напишем в среде Arduino IDE следующий скетч:

```
//подключаем библиотеку для работы с сервоприводами по PWM шине
#include <Servo.h>
//инициализируем наш сервопривод присвоением ему имени
Servo myservo;
```

```
void setup() {
//указываем порт, к которому подключен сервопривод M13(5)
myservo.attach(5);
}
```

```
void loop() {
// устанавливаем сервопривод в положение 0 градусов
myservo.write(0);
//ждём 1 секунду
delay(1000);
// устанавливаем сервопривод в положение 180 градусов
myservo.write(180);
//ждём 1 секунду
delay(1000);
}
```

После загрузки в плату сервопривод начнёт с периодичностью в 1 секунду менять своё положение из 0 градусов в 180 и обратно.

Пример подключения и работы с УЗ-сонаром

Ультразвуковой датчик является дальномером, измеряющим расстояние до объектов с помощью отражённого от поверхности объекта ультразвукового сигнала.

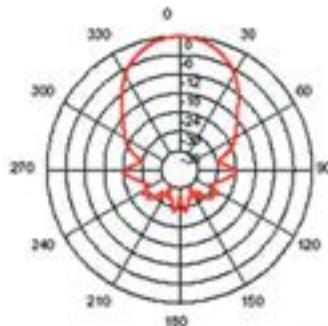
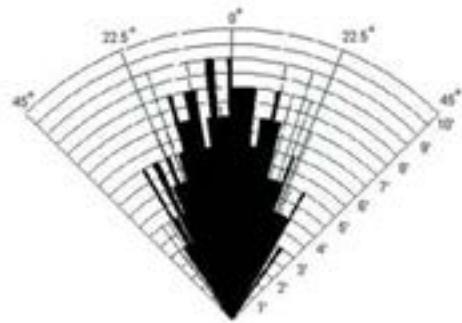


В отличие от тактильных датчиков и концевых выключателей данное устройство может заблаговременно предупредить о приближении робота к объекту, благодаря чему робот может передвигаться в среде с различными препятствиями и планировать свой маршрут.



Примечание: Следует обращать внимание на то, что отражающая способность поверхности влияет на чувствительность датчика и точность измерений. Помимо этого существенное влияние на процесс измерения расстояния до объекта оказывает его форма и взаимное расположение с роботом.

Ультразвуковой дальномер обладает диаграммой направленности, т.е. рабочей зоной, в пределах которой он может обнаруживать объекты. Если какой-либо объект находится в пределах диаграммы направленности, то он будет обнаружен данным датчиком и до него будет измерено расстояние. Точность измерения расстояния зависит от разрешающей способности датчика и его взаимного расположения с объектом.

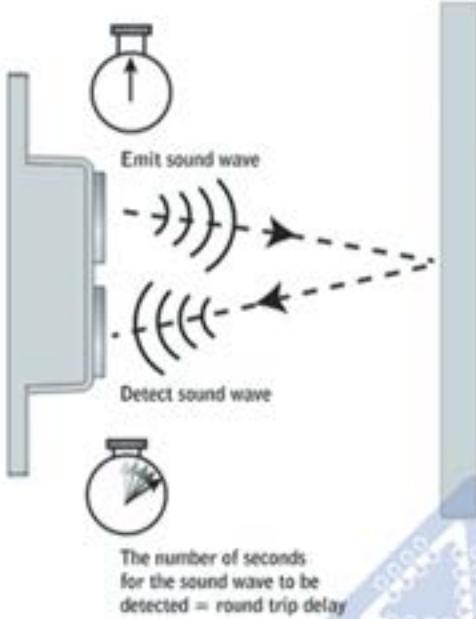


Для расширения области применения ультразвукового датчика и увеличения его зоны сканирования можно установить его на поворотное основание. В этом случае, вращая датчик, можно обозревать окружающее пространство в более широком секторе, тем самым имея возможность обнаруживать объекты по обе стороны от робота.



Ультразвуковой дальномер определяет расстояние до препятствия за счёт вычисления времени полёта звуковой волны с момента её распространения и до возвращения после отражения от объекта. Излучение звуковой волны осуществляется с частотой 40 кГц и позволяет измерять расстояние в диапазоне от 3 см до 3 м.





Алгоритмически процесс работы датчика делится на ряд отдельных этапов. На первом этапе датчик отправляет звуковую волну и начинает отсчёт времени. После того как датчик принимает отражённый сигнал, отсчёт времени прекращается. Расстояние до объекта рассчитывается как скорость распространения волны, умноженная на затраченное время.

Подключение ультразвукового датчика расстояния к программируемому контроллеру выполняется с помощью двух 2-пиновых коннекторов: INPUT и OUTPUT. Вывод INPUT отвечает за подачу управляющего импульса на датчик, вывод OUTPUT необходим для отправки полученного, результирующего импульса.

Пример подключения и опроса ультразвукового датчика расстояния:

Подключим вывод OUTPUT к порту M11(3) таким образом, чтобы оранжевый провод смотрел внутрь платы. Вывод INPUT подключим к порту M12(2) так, чтобы жёлтый провод смотрел внутрь платы.

Напишем в среде Arduino IDE следующий скетч:

```
//указываем порт, к которому подключён вход INPUT датчика расстояния M12(2)
const int Trig = 2;
//указываем порт, к которому подключён выход OUTPUT датчика расстояния
M11(3)
const int Echo = 3;

void setup() {
//настраиваем порт входа датчика расстояния на вывод сигнала
pinMode(Trig, OUTPUT);
//настраиваем порт выхода датчика расстояния на ввод сигнала
pinMode(Echo, INPUT);
//инициализируем соединение с ПК по последовательному порту
//со скоростью 9600 бод
Serial.begin(9600);
}

//объявляем и инициализируем переменную для работы с временем
unsigned int time_us=0;
//объявляем и инициализируем переменную для работы с расстоянием
unsigned int distance_sm=0;

void loop() {
// подаём сигнал на вход датчика
digitalWrite(Trig, HIGH);
// удерживаем его 10 микросекунд
delayMicroseconds(10);
// убираем сигнал со входа датчика
digitalWrite(Trig, LOW);
// замеряем длину импульса на выходе датчика
time_us=pulseIn(Echo, HIGH);
// пересчитываем полученные данные в сантиметры
distance_sm=time_us/58;
// выводим в последовательный порт полученные значения расстояния
Serial.println(distance_sm);
//пауза между опросами датчика расстояния - 100 мс
delay(100);
}
```

После загрузки скетча на плату в окне монитора последовательного порта можно будет увидеть результат опроса датчика расстояния в сантиметрах.

```
6
5
5
5
6
6
6
6
8
13
73
8
9
6
7
6
6
7
7
7
7
6
6
0000 0000 0000 0000
```



Пример подключения и работы с оптическим энкодером

Данный датчик представляет собой оптический квадратурный энкодер, предназначенный для установки на подвижные части и механизмы роботов с целью определения их положения.

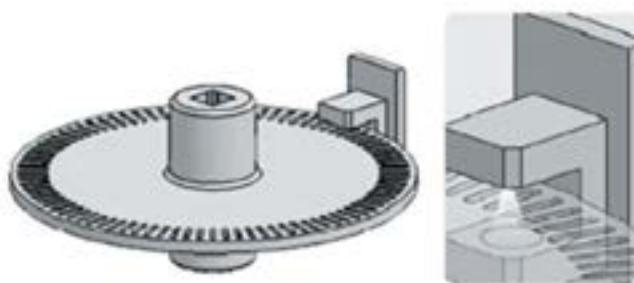


Оптический энкодер может измерять скорость вращения и количество оборотов вала привода, тем самым позволяет регулировать скорость движения роботов и дальность их перемещения.

Конструкция датчика позволяет ему принимать вращение от сменных валов и осей, благодаря чему его можно устанавливать в произвольных местах и использовать для определения параметров вращения не только приводов, но и непосредственно исполнительных механизмов робота.



Оптический энкодер представляет собой датчик, состоящий из ИК-излучателя и приёмника, расположенных по обе стороны от диска с прорезями. В процессе работы излучатель генерирует световой поток, который либо проходит сквозь прорези в диске, либо отражается от него. В случае, если световой поток проходит сквозь прорези диска, он попадает на приёмник, и датчик выдаёт свидетельствующий об этом сигнал.



Считая количество подобных прерываний и зная шаг последовательности прорезей на диске и его размеры, можно определить угол поворота вала привода. В свою очередь, проведя аналогичные расчёты за единицу времени, можно определить скорость вращения вала привода.

Примечание: Точность определения скорости вращения привода и его положения определяется количеством прорезей на диске оптического энкодера. Оптические энкодеры являются относительными датчиками, т.е. они позволяют вычислить угол поворота оси относительно какого-то начального положения.

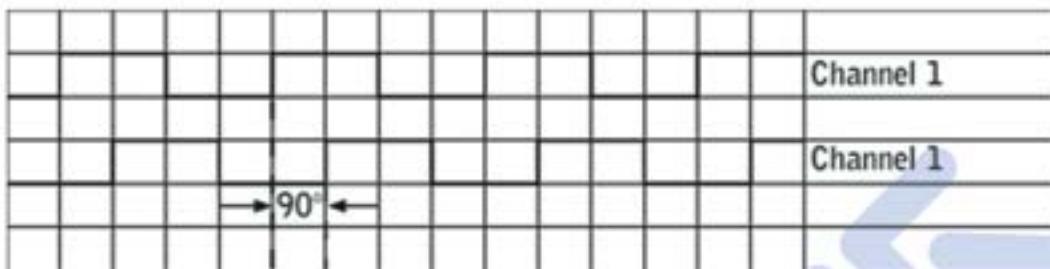
Благодаря использованию оптического энкодера можно определить перемещение робота, вычислив путь, пройденный его колёсами. Для того чтобы вычислить пройденный колесом робота путь, необходимо умножить его периметр на число совершенных им оборотов.

$$\text{circumference} = \pi \times \text{diameter of wheel}$$

Таким образом, точность перемещения робота зависит от разрешающей способности оптического энкодера и диаметра колеса робота. Очевидно, что при равной точности показаний датчика точность перемещений робота будет в случае использования колёс с минимальным диаметром.

С помощью оптического энкодера можно вести подсчёт до 1700 импульсов в секунду, что соответствует 1133 оборотам в минуту. Не рекомендуется применять данный датчик на более высоких скоростях вращения, т.к. это может привести к ошибкам измерения положения или скорости.

Оптический квадратурный инкрементный энкодер является цифровым двухканальным датчиком. Благодаря тому, что конструкция датчика содержит два приёмника отражённого света, смещённых относительно друг друга, в результате измерений датчик выдаёт два меандра – последовательности прямоугольных импульсов, смещённых по фазе относительно друг друга.



Каждый из каналов датчика содержит последовательность импульсов амплитудой 0 или 5В в зависимости от того, в каком положении находится диск с прорезями относительно приёмника отражённого света.

С помощью двухканального энкодера можно определить направление вращения вала. В случае если первый канал опережает по фазе второй канал, то вращение вала происходит по часовой стрелке. В противоположном случае вращение происходит в обратном направлении.



Поскольку диск энкодера вращается с большой скоростью и частота следования импульсов достаточно велика, для того чтобы в процессе вычислений программируемый контроллер робота не пропустил ни одного из импульсов, датчики следует подключать к специальному порту контроллера, выделенным под внешние прерывания.

Подключение энкодера выполняется с помощью двух 3-пиновых выводов, которые подключаются к цифровым портам аппаратной платформы. Результатом работы энкодера является получение меандра - сигнала, в котором содержится информация о вращении диска энкодера.

Пример подключения и опроса сдвигового энкодера:

Выполним подключение обоих выводов сдвигового энкодера к портам 0(48) и 1(49) таким образом, чтобы белые (сигнальные) провода смотрели внутрь платы.

Напишем в среде Arduino IDE следующий скетч:

```
//укажем порты, к которым подключены выводы сдвигового энкодера
enum { ENC_PORT1 = 48, ENC_PORT2 = 49 };

void setup(){
    //сконфигурируем порты на ввод сигналов
    pinMode(ENC_PORT1, INPUT);
    pinMode(ENC_PORT2, INPUT);
    //инициализируем соединение с ПК по последовательному порту
    //со скоростью 9600 бод
    Serial.begin(9600);
}

/* Функция декодирования кода Грея (см. Википедию).
 * Принимает число в коде Грея, возвращает обычное его представление.
 */
unsigned graydecode(unsigned gray){
    unsigned bin;
    for (bin = 0; gray; gray >>= 1)
        bin ^= gray;
    return bin;
}

void loop(){
    //объявим и проинициализируем переменную для предыдущего считанного
    //кода
    static uint8_t previous_code = 0;
    /* gray_code - считанное с энкодера значение
     * code - декодированное значение
     */
    //считываем значение кода Грея с выводов и декодируем его
    uint8_t gray_code = digitalRead(ENC_PORT1) | (digitalRead(ENC_PORT2) << 1),
    code = graydecode(gray_code);
```

ОЗНАКОМИТЬСЯ С ВЕРСИЕЙ САЙТА EXAMEN-TECHNOLAB.RU

```

/* Если считался нуль, значит, был произведен щелчок ручкой энкодера */
if (code == 0)
{
    /* Если переход к нулю был из состояния 3 - ручка вращалась
     * по часовой стрелке, если из 1 - против.
    */
    if (previous_code == 3)
        Serial.println("->");
    else if (previous_code == 1)
        Serial.println("<-");
    }

    /* Сохраняем код и ждём 1 мс - вполне достаточно опрашивать энкодер
     * не более 1000 раз в секунду.
    */
    previous_code = code;
    delay(1);
}

```

После загрузки скетча на плату в окне монитора последовательного порта можно будет увидеть результат опроса энкодера в виде указания направления вращения его диска.



Пример подключения и работы с инкрементным энкодером

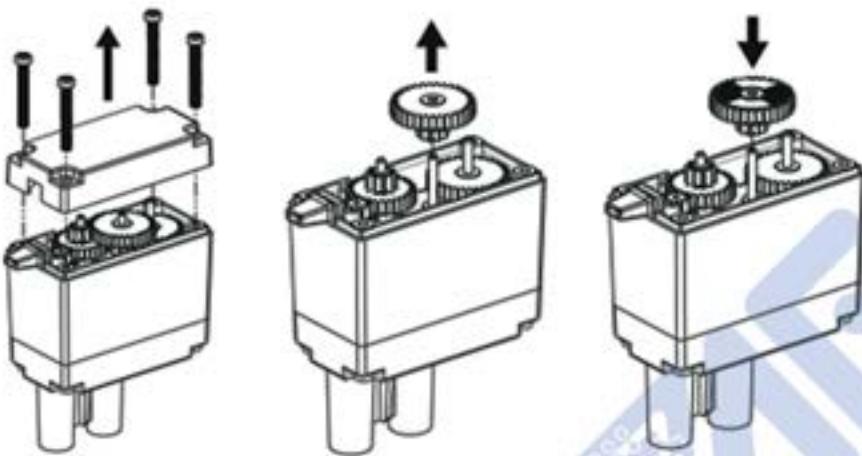
Устройство Motor 269 Integrated Encoder Module представляет собой встраиваемый в обычный привод инкрементный энкодер, позволяющий определять скорость и направление вращения привода.



Инкрементный энкодер представляет собой оптический датчик, содержащий диск с различной отражающей способностью. Данный диск располагается напротив ИК-излучателя и в процессе своего вращения прерывает поток излучения, в результате чего становится возможным определить скорость вращения двигателя по интенсивности импульсов, приходящих с датчика.



Для установки датчика необходимо открутить 4 винта крышки привода и снять её, после чего необходимо установить отражающий диск вместо шестерни, как это показано на рисунке.

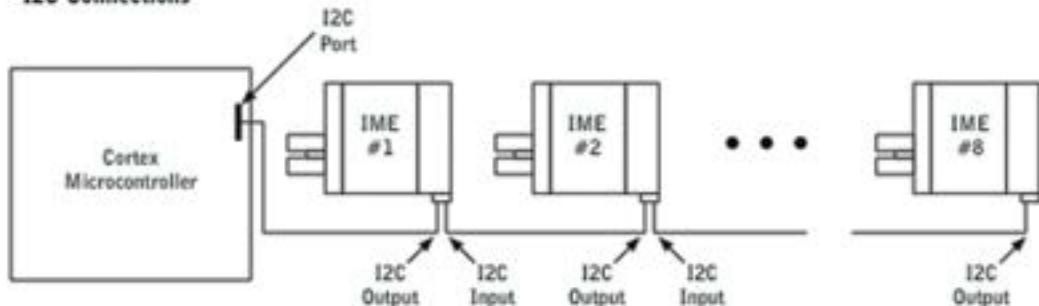


После замены шестерёнки необходимо удостовериться в том, что выходной вал привода прокручивается без заклинивания. Далее можно установить взамен старой крышки привода новую с выходными каналами для энкодера.



В завершение необходимо окончательно удостовериться в правильности сборки привода и проверить его на плавность вращения и отсутствие заклиниваний.

Устройство Motor 269 Integrated Encoder Module подключается к программируемому контроллеру с помощью интерфейса I2C. Данный интерфейс позволяет подключать к одной шине все энкодеры, применяемые в работе разом, т.е. до 8 штук в соответствии с общим числом портов управления приводами.

I2C Connections

Подключение приводов к шине I2C осуществляется последовательным образом, т.е. путём соединения выхода первого устройства со входом второго и т.д. Важно заметить, что последовательность подключения энкодеров не обязательно должна совпадать с последовательностью подключения приводов к портам программируемого контроллера. Так, например, первый привод может быть подключен к порту №1, второй же к порту №10, но энкодеры будут все равно подключены последовательно друг к другу.

Подключение интегрируемого энкодера выполняется по I2C шине одним 4-пиновым проводом.

Для работы с этим энкодером необходимо использование сторонней библиотеки для Arduino IDE - I2CEncoder.h.

Пример подключения и опроса интегрируемого энкодера:

Подключим интегрируемый энкодер к I2C порту аппаратной платформы так, чтобы вывод GND соответствовал чёрному проводу вывода энкодера, а SDA – белому.



Напишем в среде Arduino IDE следующий скетч:

```
//подключение библиотеки для работы с i2c шиной
#include <Wire.h>
//подключение библиотеки для работы с I2C энкодером
#include <I2CEncoder.h>
//объявляем энкодер путём присвоения ему имени
I2CEncoder encoder;

void setup() {
    // инициализируем подключение по i2c шине
    Wire.begin();
    //инициализируем соединение с ПК по последовательному порту
    //со скоростью 9600 бод
    Serial.begin(9600);
    //инициализируем энкодер, установленный на 269 моторе
    encoder.init(MOTOR_269_ROTATIONS, MOTOR_269_TIME_DELTA);
}

void loop() {
    //получаем значение скорости в оборотах в минуту
    //и отправляем данные в последовательный порт
    Serial.print(encoder.getSpeed());
    Serial.print(",");
    //получаем значение положения в оборотах
    Serial.println(encoder.getPosition());
    //пауза между опросами энкодера 200 мс
    delay(200);
}
```



После загрузки скетча на плату в окне монитора последовательного порта можно будет увидеть результат опроса интегрируемого энкодера в виде скорости вращения мотора и его положения.

| Position | Speed |
|----------|-------|
| 27.25 | 0.59 |
| 16.15 | 0.64 |
| 16.80 | 0.69 |
| 16.90 | 0.70 |
| 6.78 | 0.70 |
| 4.01 | 0.70 |
| 14.18 | 0.77 |
| 29.65 | 0.87 |
| 28.00 | 0.98 |
| 28.25 | 1.05 |
| 31.83 | 1.16 |
| 29.74 | 1.22 |
| 20.29 | 1.22 |
| 6.65 | 1.22 |
| 3.97 | 1.24 |
| 27.75 | 1.32 |
| 24.35 | 1.41 |
| 21.13 | 1.48 |
| 22.77 | 1.56 |
| 18.69 | 1.62 |
| 20.88 | 1.69 |
| 20.88 | 1.70 |
| 7.68 | 1.70 |
| 4.32 | 1.70 |
| 3.01 | 1.70 |
| 2.31 | 1.70 |



Работа со встроенным Bluetooth-модулем

В программируемом контроллере имеется встроенный Bluetooth-модуль, позволяющий организовывать обмен данными между платой и мобильным устройством, таким как планшет. Для использования Bluetooth-модуля необходимо снять на аппаратной платформе перемычку AT и перезапустить платформу путём отключения питания от неё. Затем на мобильном устройстве необходимо выполнить обычным поиском ближайшие Bluetooth-устройства и выполнить подключение к устройству HC-05. При запросе PIN-кода используйте «0000» или «1234».

Пример организации обмена данными между ПК и мобильным устройством:
Напишем в среде Arduino IDE следующий скетч:

```
void setup()
{//устанавливаем соединение с bluetooth-модулем на скорости 115200
Serial2.begin(115200);
//устанавливаем соединение с персональным компьютером на скорости 9600
Serial.begin(9600);
}

void loop()
{//если на модуль пришли данные со стороны мобильного устройства
if(Serial2.available())
{//читаем их
byte a=Serial2.read();
//отправляем на ПК
Serial.write(a);
}
//если пришли данные со стороны ПК
if(Serial.available())
{//читаем их
byte a=Serial.read();
//отправляем на ПК
Serial2.write(a);
}
}
```

После загрузки скетча на плату, подключившись к Bluetooth-модулю с мобильного устройства и открыв на компьютере монитор последовательного порта, можно обмениваться данными. Однако важно заметить, что Bluetooth-модуль может быть настроен с помощью AT-команд. С помощью этих AT-команд можно изменить скорость обмена данными модуля, его имя, пин-код, режим работы и т.д.

Примеры AT команд:

AT проверить статус подключения

AT+VERSION? узнать версию прошивки.

AT+UART? узнать установленную скорость.

AT+UART=38400, 0,0 установить скорость 38400.

AT+NAME? узнать имя.

AT+NAME=HC-05 BLUE установить имя HC-05 BLUE.

AT+PSWD? узнать пароль.

AT+PSWD=0000 установить пароль 0000.

AT+ORGL сброс на заводские настройки

Для перехода в режим настройки модуля необходимо загрузить скетч (обязательно необходимо изменить скорости обмена на 38400), представленный выше, установить перемыкатель AT и перезагрузить платформу. После чего открыть монитор последовательного порта, выбрать скорость 38400, режим «Оба NL & CR» вместо «Не найден конец строки» и отправить команду «AT». Если в ответ придет «OK», то модуль успешно вошёл в режим настройки AT-командами.

Пример настройки модуля в качестве ведомого (*slave*):

Сброс предыдущих настроек: AT+ORGL

Сброс спаренных устройств: AT+RMAAD

Установка пароля: AT+PSWD=1234

Включение режима ведомого: AT+ROLE=0

Дополнительно можно узнать адрес устройства

(понадобится для настройки спаренного модуля): AT+ADDR?

В ответ получим сам адрес: ADDR=xx:xx:xxxxxx

Пример настройки модуля в качестве ведущего:

Сброс предыдущих настроек: AT+ORGL

Сброс спаренных устройств: AT+RMAAD

Включение режима ведущего: AT+ROLE=1

Рестарт после смены роли: AT+RESET

Если необходимо связать ведомого и ведущего, используются команды:

Установка пароля ведомого: AT+PSWD=1234

Указываем парное устройство: AT+PAIR=<адрес>,<таймаут>

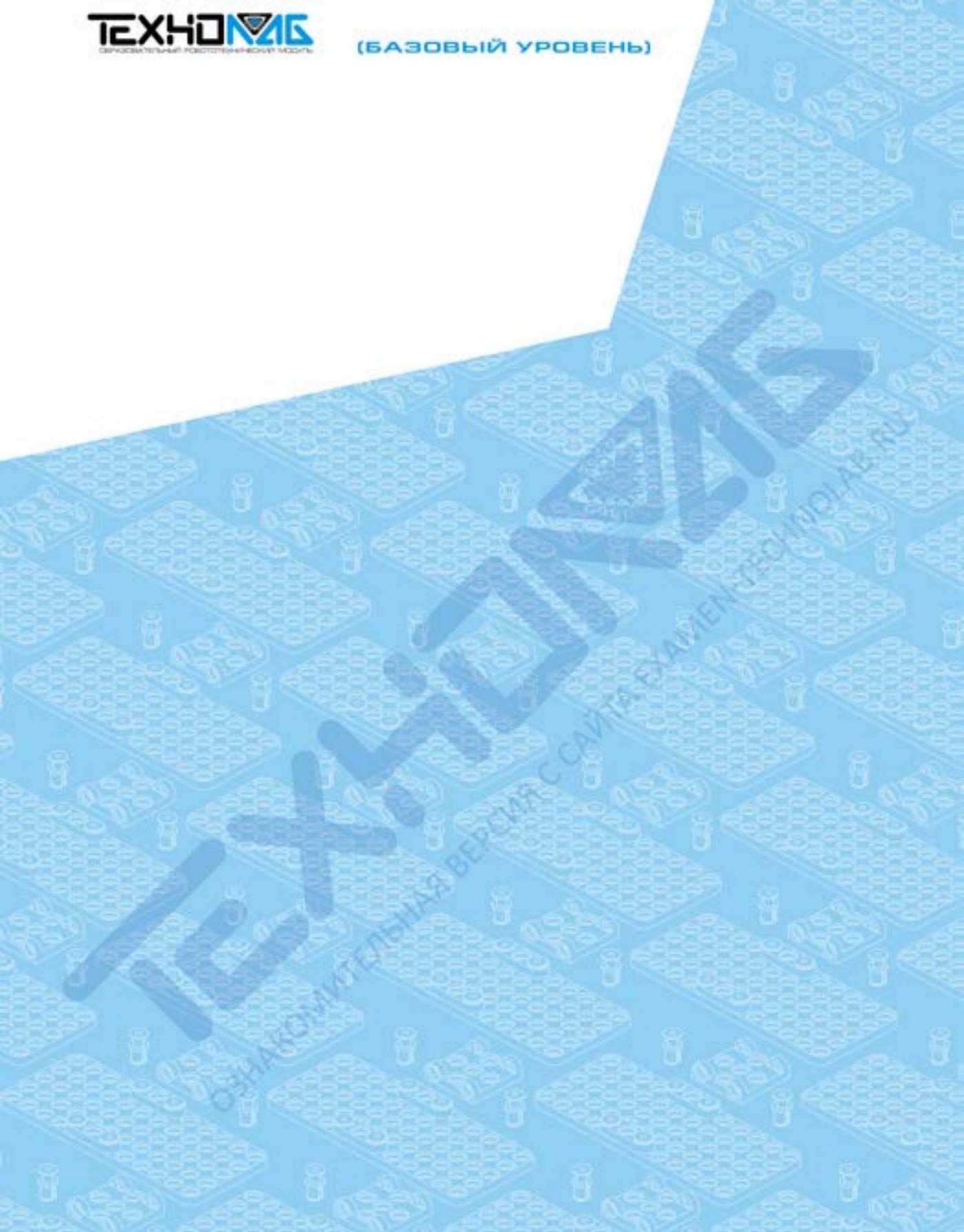
(пример: AT+PAIR=12,6,143117, 5)

Связываем с конкретным адресом: AT+BIND=<адрес>

(пример: AT+BIND=12,6,143117)

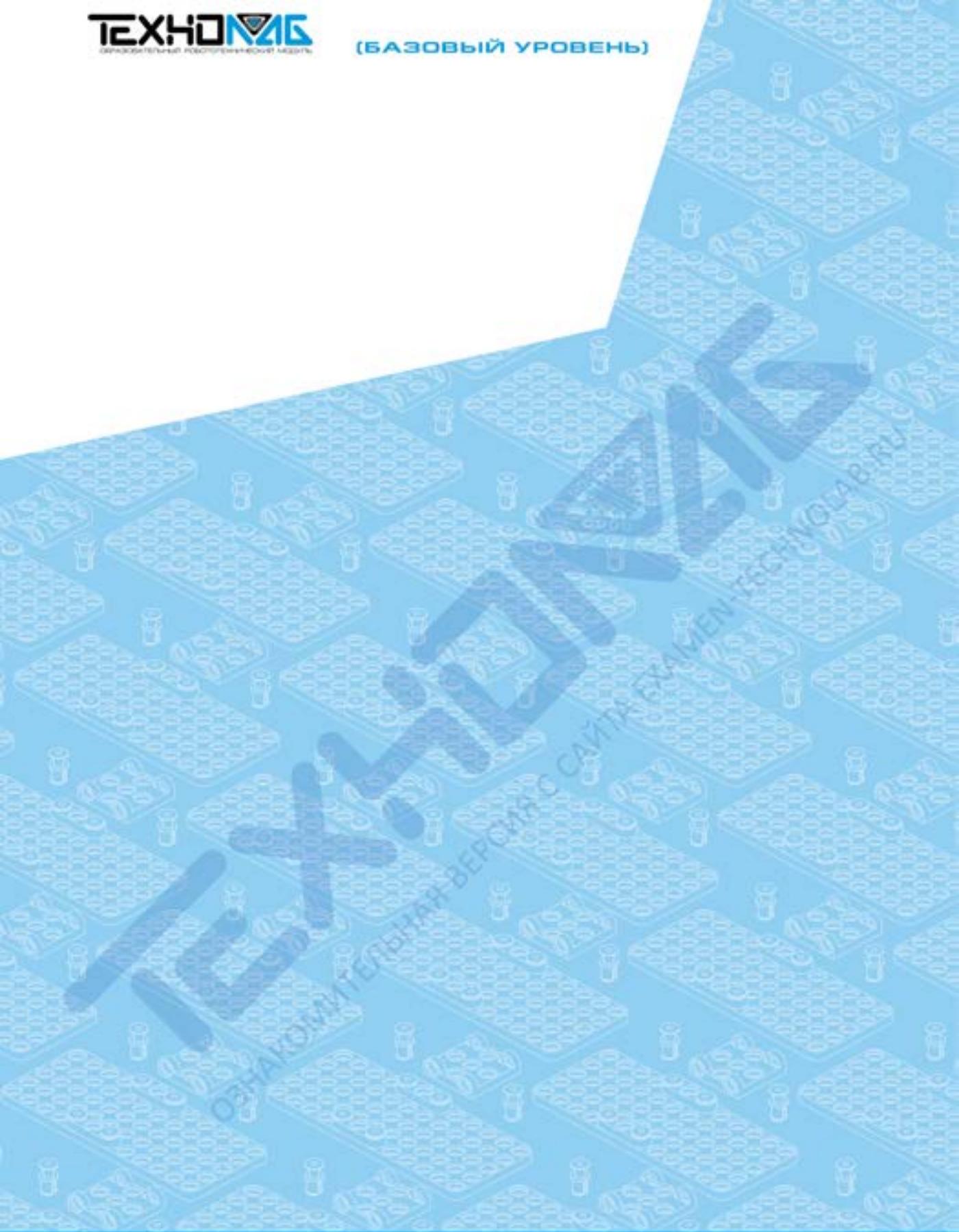
Запрещаем соединяться с другими адресами: AT+CMODE=0

После настройки отключаем перемыкатель AT и перезапускаем модуль.



Разработка макета робота





Приложение 2.

Разработка макета робота

В рамках процесса разработки действующего макета робота предлагается решить последовательно ряд следующих задач:

1. Обеспечить подвижность робота (вперёд-назад, повороты).
2. Обеспечить движение манипулятора (хватательное + выброс).
3. Обеспечить опрос ультразвукового дальномера и начать его использование совместно с работой манипулятора (убирать препятствие на пути робота).
4. Обеспечить опрос трёх детекторов линии и использовать их при движении робота.
5. Объединить полученные решения в одну программу, решающую задачу движения робота по линии и убирания препятствия на пути.

Сам робот собирается из входящих в состав набора компонентов, по входящей в набор инструкции, однако приведённые ниже примеры разработаны для слегка модифицированного робота – три детектора линии установлены на нижней части схвата, а ультразвуковой дальномер установлен непосредственно на мотор схвата. Питание аппаратной платформы, моторов и компонентов робота осуществляется от стандартного аккумулятора VEX, имеющего 7,2В на выходе. После сборки робота и установки на него программируемого контроллера можно приступить к решению задач по обеспечению подвижности робота.



Движение робота вперёд-назад и осуществление поворотов

В качестве моторов для движения робота используются два двухпроводных мотора 393, установленные на левой и правой частях платформы соответственно. Подключение и управление ими осуществляется напрямую от аппаратной платформы, без использования внешних драйверов мотора. Для подключения моторов к аппаратной платформе необходимо соединить кабель правого мотора с портом M2, а левого с портом M1 аппаратной платформы с соблюдением полярности.

Затем напишем в среде Arduino IDE следующий скетч:

```
//указываем пины, отвечающие за управление портом M1
const int motorM1PinOne = 6;
const int motorM1PinTwo = 7;

//указываем пины, отвечающие за управление портом M2
const int motorM2PinOne = 46;
const int motorM2PinTwo = 45;

void setup() {
    //конфигурируем пины порта M1 на вывод сигнала
    pinMode(motorM1PinOne, OUTPUT);
    pinMode(motorM1PinTwo, OUTPUT);
    //конфигурируем пины порта M2 на вывод сигнала
    pinMode(motorM2PinOne, OUTPUT);
    pinMode(motorM2PinTwo, OUTPUT);

    //едем вперед в течение 2x секунд
    forward();
    delay(2000);
    //поворот направо
    right();
    delay(1000);
    //поворот налево
    left();
    delay(1000);
    //едем назад в течение 2 секунд
    reverse();
    delay(2000);
    //останавливаемся
    stopMove();
}
```

```

void loop() {
}

void forward (){
//правый мотор
digitalWrite (motorM2PinOne, LOW);
digitalWrite (motorM2PinTwo, HIGH);
//левый мотор
digitalWrite (motorM1PinOne, LOW);
digitalWrite (motorM1PinTwo, HIGH);
}

void right (){
//правый мотор
digitalWrite (motorM2PinOne, HIGH);
digitalWrite (motorM2PinTwo, LOW);
//левый мотор
digitalWrite (motorM1PinOne, LOW);
digitalWrite (motorM1PinTwo, HIGH);
}

void left (){
//правый мотор
digitalWrite (motorM2PinOne, LOW);
digitalWrite (motorM2PinTwo, HIGH);
//левый мотор
digitalWrite (motorM1PinOne, HIGH);
digitalWrite (motorM1PinTwo, LOW);
}

void reverse (){
//правый мотор
digitalWrite (motorM2PinOne, HIGH);
digitalWrite (motorM2PinTwo, LOW);
//левый мотор
digitalWrite (motorM1PinOne, HIGH);
digitalWrite (motorM1PinTwo, LOW);
}

```

ОЗНАКОМЛЕНІЯ ВЕРСІЯ С САЙТА EXAMEN-TECHNOLAB.RU

```
void stopMove (){  
//правый мотор  
digitalWrite (motorM2PinOne, LOW);  
digitalWrite (motorM2PinTwo, LOW);  
//левый мотор  
digitalWrite (motorM1PinOne, LOW);  
digitalWrite (motorM1PinTwo, LOW);  
}
```

После загрузки на плату робот начнёт движение вперёд и будет двигаться в этом направлении в течение 2 секунд, затем начнёт двигаться сначала по часовой стрелке, затем против часовой стрелки, после чего поедет назад и остановится. В будущем для выполнения поворотов на определённый угол необходимо будет изменять значение времени, в течение которого робот двигается по часовой или против часовой стрелки, например:

```
//поворот направо  
right ();  
delay(1000); - в этом месте надо указать необходимое время движения в миллисекундах
```

Такое действие необходимо, поскольку в данном примере не используются энкодеры.



Управление манипулятором робота

«Хватательное» движение манипулятора может быть обеспечено либо сервоприводом, либо мотором. Это необходимо учитывать при программировании этого движения. В случае использования мотора для хватательного движения используется мотор 269, подключённый через внешний драйвер мотора. Движение манипулятора вперёд обеспечивается мотором 396, установленным на ось штанги. Подключение и управление им осуществляется через внешний драйвер мотора. Подключим мотор, отвечающий за движение штанги, к порту M13(5) через внешний драйвер мотора, а мотор, отвечающий за работу «клешни», также через драйвер мотора подключим к порту M12(2)

Напишем в среде Arduino IDE следующий скетч:

```
#include <Servo.h>
```

```
Servo grab; //объявили мотор, отвечающий за хватательное движение
Servo roll; //объявили мотор, отвечающий за выбрасывательное движение
```

```
void setup()
```

```
{
```

```
grab.attach(2); //указываем порт, к которому подключен мотор клешни
roll.attach(5); //указываем порт, к которому подключен мотор штанги
```

```
reveal(); // вызов функции раскрытия хвата
delay (2000); // ожидание 2 сек.
```

```
toclose(); // вызов функции сокращения хвата
delay (2000); // ожидание 2 сек.
```

```
reroll_up(); // поднимаем руку
delay (2000); // ожидание 2 сек.
```

```
reveal(); // вызов функции раскрытия хвата
delay (2000); // ожидание 2 сек.
```

```
toclose(); // вызов функции сокращения хвата
reroll_down(); //опускаем руку
```

```
}
```

Приложение

```
void reveal (){
    grab.writeMicroseconds(1800);
    delay (900);
    grab.writeMicroseconds(1500);
}

void toclose (){
    grab.writeMicroseconds(1200);
    delay (900);
    grab.writeMicroseconds(1500);
}

void reroll_up (){
    roll.writeMicroseconds(1800);
    delay (1000);
    roll.writeMicroseconds(1500);
}

void reroll_down (){
    roll.writeMicroseconds(1200);
    delay (1000);
    roll.writeMicroseconds(1500);
}

void loop (){
    //здесь ничего не пишем, т.к. зацикливать нечего - программа отрабатывает один
    раз
}
```

После загрузки скетча в платформу робот выполнит следующие действия:

- откроет «клешню» и, чуть подождав, закроет ее
- перекинет через себя манипулятор
- снова откроет и закроет «клешню»
- вернёт манипулятор в исходное состояние

В случае использования сервопривода вместо мотора для управления клешней схвата подключение сервопривода может быть произведено к порту M12(2), и код скетча будет выглядеть несколько иначе:

```
#include <Servo.h>

Servo grab; //объявили мотор, отвечающий за хватательное движение
Servo roll; //объявили мотор, отвечающий за выбрасывательное движение

void setup()
{
    grab.attach(2); //указываем порт, к которому подключён мотор клешни
    roll.attach(5); //указываем порт, к которому подключён мотор штанги

    reveal(); // вызов функции раскрытия хвата
    delay (2000); // ожидание 2 сек.

    toclose(); // вызов функции сокращения хвата
    delay (2000); // ожидание 2 сек.

    reroll_up(); // поднимаем руку
    delay (2000); // ожидание 2 сек.

    reveal(); // вызов функции раскрытия хвата
    delay (2000); // ожидание 2 сек.

    toclose(); // вызов функции сокращения хвата
    reroll_down(); //опускаем руку
}

void reveal (){
    grab.write(180); // раскрытие на 180 градусов
}

void toclose (){
    grab.write(50); //сокращение до 50 градусов
}

void reroll_up (){
    roll.write(1000); // движение против часовой стрелки
    delay (1700);
    roll.write(1500); // остановка
}
```

```
void reroll_down (){
roll.write(1800); // движение по часовой стрелке
delay (1600);
roll.write(1500); // остановка
}

void loop (){
//здесь ничего не пишем, т.к. зацикливать нам нечего - программа отрабатывает
один раз
}
```

Отличием данного скетча от предыдущего является то, что управление сервоприводом осуществляется путём задания его положения в градусах, тогда как в предыдущем скетче управление осуществлялось путём PWM импульсами с использованием искусственных временных задержек.



Подключение ультразвукового дальномера

В рамках данной задачи ультразвуковой дальномер будет использоваться для обнаружения какого-либо объекта на пути робота и определения расстояния до него с целью последующего его захвата манипулятором. В данном случае дальномер подключается своим выводом INPUT к порту 0(48), а выводом OUTPUT к порту 1(49).

Скетч для работы с дальномером и манипулятором выглядит следующим образом:

```
#include <Servo.h>
```

```
Servo grab; //объявили сервомодуль, отвечающий за «хватательное» движение
Servo roll; //объявили сервомодуль, отвечающий за «выбрасывательное» движение
```

```
const int input = 48; // объявили порт для управляющего сигнала
const int output = 49; // объявили порт для считанного сигнала
```

```
void setup()
{
  pinMode(input, OUTPUT); //сконфигурировали порт для управляющего сигнала на
  выход
  pinMode(output, INPUT); //сконфигурировали порт для считанного сигнала на
  вход
```

```
grab.attach(2); //указываем порт, к которому подключен мотор «клешни»
roll.attach(5); //указываем порт, к которому подключен мотор штанги
}
```

```
unsigned int time_us=0; //объявили переменную для времени
unsigned int distance_sm=0; //объявили переменную для расстояния
unsigned int distance=0; //объявили переменную для расстояния
```

```
void loop() {
  distance = ultrasonic();
```

```
if (distance <= 20) { //если дистанция до объекта менее 20 см, то захватываем его
  reveal(); // вызов функции раскрытия хвата
  delay(1100);
  toclose(); // вызов функции закрытия хвата
  delay(500);
```

```
reroll_up(); // поднимаем руку
delay (2000); // ожидание 2 сек.

reveal(); // вызов функции раскрытия хвата
delay (2000); // ожидание 2 сек.

toclose(); // вызов функции закрытия хвата
reroll_down(); // опускаем руку
}
delay(100);

}

unsigned int ultrasonic (){
digitalWrite(input, HIGH); // подаём управляющий сигнал
delayMicroseconds(10); // удерживаем 10 микросекунд
digitalWrite(input, LOW); // убираем управляющий сигнал

time_us=pulseIn(output, HIGH); // замеряем длину импульса
distance_sm=time_us/58; // пересчитываем в сантиметры
Serial.println(distance_sm); // передаём значение в порт
return distance_sm;
}

void reveal (){
grab.writeMicroseconds(1800);
delay (700);
grab.writeMicroseconds(1500);
}

void toclose (){
grab.writeMicroseconds(1200);
delay (700);
grab.writeMicroseconds(1500);
}

void reroll_up (){
roll.writeMicroseconds(1800);
delay (2000);
roll.writeMicroseconds(1500);
}
```

```
void reroll_down (){
roll.writeMicroseconds(1200);
delay (2000);
roll.writeMicroseconds(1500);
}
```

Данный скетч реализует непрерывный опрос датчика расстояния, и в случае обнаружения какого-либо объекта на расстоянии менее 20 см производит открытие хвата, захват объекта и его последующий выброс позади себя. После чего робот снова начинает опрашивать дальномер.

ОЗНАКОМИТЕЛЬНАЯ ВЕРСИЯ С САЙТА EXAMEN-TECHNOLAB.RU



Работа с ИК-датчиками для обнаружения линии

Одним из способов удержания робота на определённом маршруте является езда по линии. Линия (чёрного цвета), изображённая на контрастном фоне (белого цвета), легко фиксируется роботом с помощью детекторов линии, что позволяет роботу двигаться строго вдоль заранее заданной линии, не уходя с неё. В нашем случае используются 3 детектора линии, установленные параллельно друг другу, таким образом, чтобы расстояние от нижнего края детектора до поверхности, по которой едет робот, было минимальным. В случае когда центральный детектор фиксирует наличие линии, робот едет прямо, а если наличие линии фиксирует один из крайних детекторов, то робот поворачивает в нужную сторону до тех пор, пока центральный детектор не зафиксирует линию.

Подключение детекторов линии выполняется следующим образом: левый детектор линии подключается к порту 2(37), центральный – к порту 3(36) и правый – к 4(35).

Скетч для простого опроса всех трёх детекторов одновременно выглядит следующим образом:

```
int sensorPort1 = 37; //указываем порт, к которому подключен Line Tracker 1
int sensorPort2 = 36; //указываем порт, к которому подключен Line Tracker 2
int sensorPort3 = 35; //указываем порт, к которому подключен Line Tracker 3

int sensorValue1=0; //объявляем переменную для Line Tracker 1
int sensorValue2=0; //объявляем переменную для Line Tracker 2
int sensorValue3=0; //объявляем переменную для Line Tracker 3

void setup() {
    pinMode(sensorPort1, INPUT);
    pinMode(sensorPort2, INPUT);
    pinMode(sensorPort3, INPUT);
    //инициализация подключения по соп-порту
    Serial.begin(9600);
}
```

```

void loop() {
//читываем значения с портов
sensorValue1 = digitalRead(sensorPort1);
sensorValue2 = digitalRead(sensorPort2);
sensorValue3 = digitalRead(sensorPort3);
//выводим на последовательный порт с конвертацией значений в вольты
Serial.println(sensorValue1);
Serial.println(sensorValue2);
Serial.println(sensorValue3);
delay(100);
}
  
```

Выполнив объединение скетчей для езды и скетча для опроса детекторов, получаем скетч для езды робота по линии:

```

//указываем пины, отвечающие за управление портом M1
const int motorM1PinOne = 6;
const int motorM1PinTwo = 7;

//указываем пины, отвечающие за управление портом M2
const int motorM2PinOne = 46;
const int motorM2PinTwo = 45;

int sensorPort1 = 37;//указываем порт, к которому подключен Line Tracker 1
int sensorPort2 = 36;//указываем порт, к которому подключен Line Tracker 2
int sensorPort3 = 35;//указываем порт, к которому подключен Line Tracker 3

int sensorValue1=0; //объявляем переменную для Line Tracker 1
int sensorValue2=0; //объявляем переменную для Line Tracker 2
int sensorValue3=0; //объявляем переменную для Line Tracker 3

void setup() {
pinMode(sensorPort1, INPUT);
pinMode(sensorPort2, INPUT);
pinMode(sensorPort3, INPUT);

//конфигурируем пины порта M1 на вывод сигнала
pinMode(motorM1PinOne, OUTPUT);
pinMode(motorM1PinTwo, OUTPUT);
//конфигурируем пины порта M2 на вывод сигнала
pinMode(motorM2PinOne, OUTPUT);
pinMode(motorM2PinTwo, OUTPUT);
}
  
```

```
void loop() {
    forvard();
    //считываем значения с портов
    sensorValue1 = digitalRead(sensorPort1);
    sensorValue2 = digitalRead(sensorPort2);
    sensorValue3 = digitalRead(sensorPort3);

    if (sensorValue3){ //контролируем значения с правого line tracker
        right(); //если меньше определённого - подворачиваем
    }

    if (sensorValue1){ // контролируем значения с левого line tracker
        left(); //если меньше определённого - подворачиваем
    }

    if (sensorValue1 && sensorValue2 && sensorValue3) {//если все 3 детектора фикси-
        руют линию
        stopMove(); //остановка
    }
    delay(10);
}

void forvard (){
    //правый мотор
    digitalWrite (motorM2PinOne, LOW);
    digitalWrite (motorM2PinTwo, HIGH);
    //левый мотор
    digitalWrite (motorM1PinOne, LOW);
    digitalWrite (motorM1PinTwo, HIGH);
}

void right (){
    //правый мотор
    digitalWrite (motorM2PinOne, HIGH);
    digitalWrite (motorM2PinTwo, LOW);
    //левый мотор
    digitalWrite (motorM1PinOne, LOW);
    digitalWrite (motorM1PinTwo, HIGH);
}
```

```
void left (){
    //правый мотор
    digitalWrite (motorM2PinOne, LOW);
    digitalWrite (motorM2PinTwo, HIGH);
    //левый мотор
    digitalWrite (motorM1PinOne, HIGH);
    digitalWrite (motorM1PinTwo, LOW);
}
```

```
void stopMove (){
    //правый мотор
    digitalWrite (motorM2PinOne, LOW);
    digitalWrite (motorM2PinTwo, LOW);
    //левый мотор
    digitalWrite (motorM1PinOne, LOW);
    digitalWrite (motorM1PinTwo, LOW);
}
```



Разработка комплексной системы управления робота

Выполнив слияние всех скетчей, написанных по отдельности, можно получить готовый скетч для робота, который будет двигаться по линии, а также детектировать объекты, подхватывать их манипулятором и убирать с пути.

Полностью готовый скетч выглядит следующим образом:

```
#include <Servo.h>

Servo grab; //объявили сервомодуль, отвечающий за хватательное движение
Servo roll; //объявили сервомодуль, отвечающий за выбрасывательное движение

//указываем пины, отвечающие за управление портом M1
const int motorM1PinOne = 6;
const int motorM1PinTwo = 7;

//указываем пины, отвечающие за управление портом M2
const int motorM2PinOne = 46;
const int motorM2PinTwo = 45;

int sensorPort1 = 37;//указываем порт, к которому подключен Line Tracker 1
int sensorPort2 = 36;//указываем порт, к которому подключен Line Tracker 2
int sensorPort3 = 35;//указываем порт, к которому подключен Line Tracker 3

int sensorValue1=0; //объявляем переменную для Line Tracker 1
int sensorValue2=0; //объявляем переменную для Line Tracker 2
int sensorValue3=0; //объявляем переменную для Line Tracker 3

const int input = 48; //объявили порт для управляющего сигнала
const int output = 49; //объявили порт для считанного сигнала

unsigned int time_us=0; //объявили переменную для времени
unsigned int distance_sm=0; //объявили переменную для расстояния
unsigned int distance=0; //объявили переменную для расстояния

void setup() {
pinMode(sensorPort1, INPUT);
pinMode(sensorPort2, INPUT);
pinMode(sensorPort3, INPUT);
```

```

//конфигурируем пины порта M1 на вывод сигнала
pinMode(motorM1PinOne, OUTPUT);
pinMode(motorM1PinTwo, OUTPUT);
//конфигурируем пины порта M2 на вывод сигнала
pinMode(motorM2PinOne, OUTPUT);
pinMode(motorM2PinTwo, OUTPUT);

pinMode(input, OUTPUT); //сконфигурировали порт для управляющего сигнала на
выход
pinMode(output, INPUT); //сконфигурировали порт для считанного сигнала на
вход

grab.attach(2); //указываем порт, к которому подключён мотор «клешни»
roll.attach(5); //указываем порт, к которому подключён мотор штанги
Serial.begin(9600);
}

void loop() {
forward();
distance = ultrasonic();

Serial.println(distance);
if (distance <= 20) { //если дистанция до объекта менее 20 см, то захватываем его
reveal(); // вызов функции раскрытия хвата
delay(1100);
stopMove();
toclose(); //вызов функции сокращения хвата
delay(500);

reroll_up(); //поднимаем руку
delay (2000); //ожидание 2 сек.

reveal(); //вызов функции раскрытия хвата
delay (2000); //ожидание 2 сек.

toclose(); //вызов функции сокращения хвата
reroll_down(); //опускаем руку
}

// считываем значения с портов
sensorValue1 = digitalRead(sensorPort1);
sensorValue2 = digitalRead(sensorPort2);
sensorValue3 = digitalRead(sensorPort3);

```

```
if (sensorValue3){ //контролируем значения с правого line tracker  
right(); //если меньше определённого - подворачиваем  
}  
  
if (sensorValue1){ //контролируем значения с левого line tracker  
left(); // если меньше определённого - подворачиваем  
}  
  
if (sensorValue1 && sensorValue2 && sensorValue3) { //если все 3 детектора фикси-  
руют линию  
stopMove(); //остановка  
}  
  
delay(10);  
}  
  
void forward (){  
//правый мотор  
digitalWrite (motorM2PinOne, LOW);  
digitalWrite (motorM2PinTwo, HIGH);  
//левый мотор  
digitalWrite (motorM1PinOne, LOW);  
digitalWrite (motorM1PinTwo, HIGH);  
}  
  
void right (){  
//правый мотор  
digitalWrite (motorM2PinOne, HIGH);  
digitalWrite (motorM2PinTwo, LOW);  
//левый мотор  
digitalWrite (motorM1PinOne, LOW);  
digitalWrite (motorM1PinTwo, HIGH);  
}  
  
void left (){  
//правый мотор  
digitalWrite (motorM2PinOne, LOW);  
digitalWrite (motorM2PinTwo, HIGH);  
//левый мотор  
digitalWrite (motorM1PinOne, HIGH);  
digitalWrite (motorM1PinTwo, LOW);  
}
```

```

void stopMove (){
//правый мотор
digitalWrite (motorM2PinOne, LOW);
digitalWrite (motorM2PinTwo, LOW);
//левый мотор
digitalWrite (motorM1PinOne, LOW);
digitalWrite (motorM1PinTwo, LOW);
}

unsigned int ultrasonic (){
digitalWrite(input, HIGH); //подаём управляющий сигнал
delayMicroseconds(10); //удерживаем 10 микросекунд
digitalWrite(input, LOW); //убираем управляющий сигнал

time_us=pulseIn(output, HIGH); //замеряем длину импульса
distance_sm=time_us/58; //пересчитываем в сантиметры
Serial.println(distance_sm); //передаём значение в порт
return distance_sm;
}

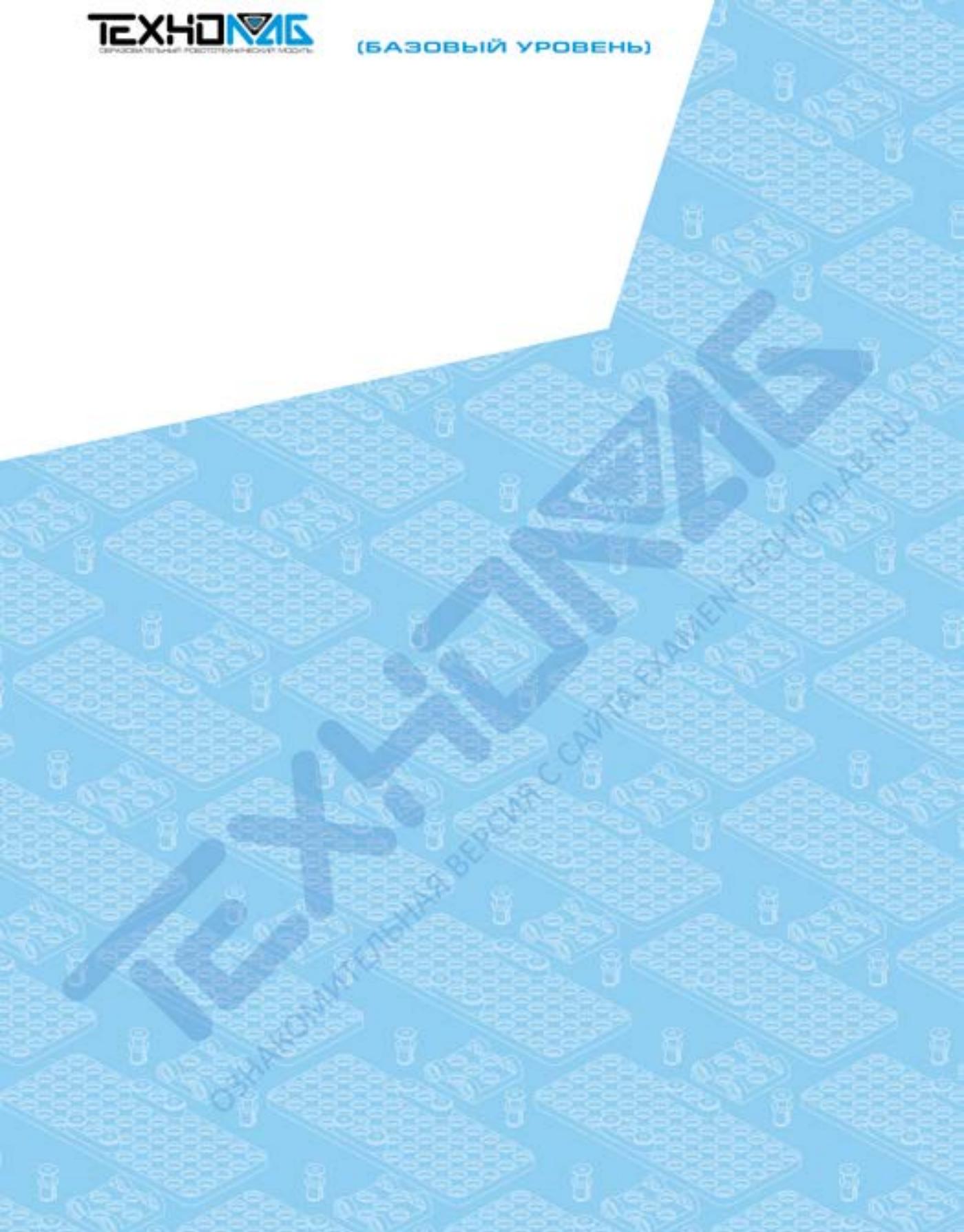
void reveal (){
grab.writeMicroseconds(1800);
delay (700);
grab.writeMicroseconds(1500);
}

void toclose (){
grab.writeMicroseconds(1200);
delay (700);
grab.writeMicroseconds(1500);
}

void reroll_up (){
roll.writeMicroseconds(1800);
delay (2000);
roll.writeMicroseconds(1500);
}

void reroll_down (){
roll.writeMicroseconds(1200);
delay (2000);
roll.writeMicroseconds(1500);
}

```

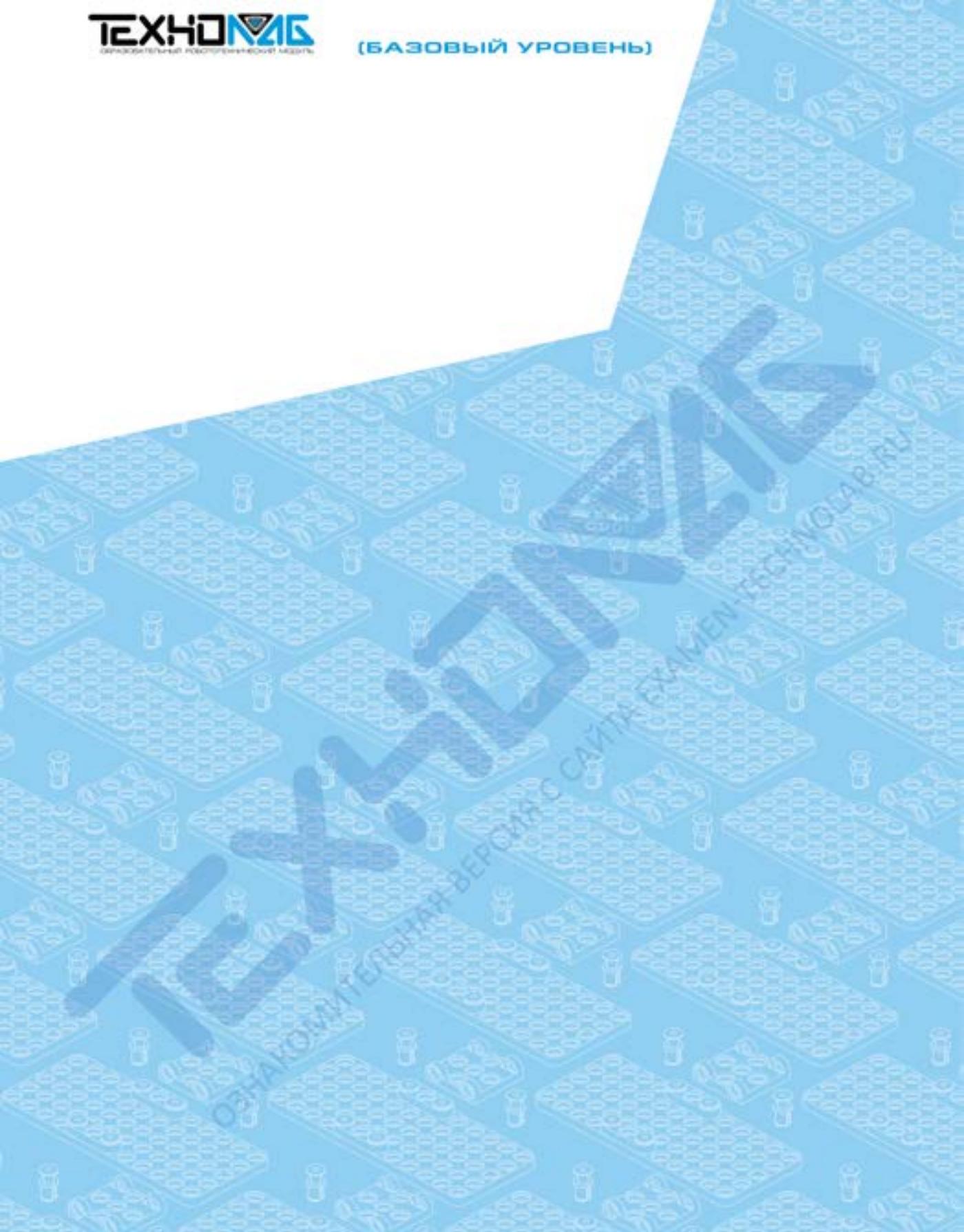


Руководство по сборке Clawbot



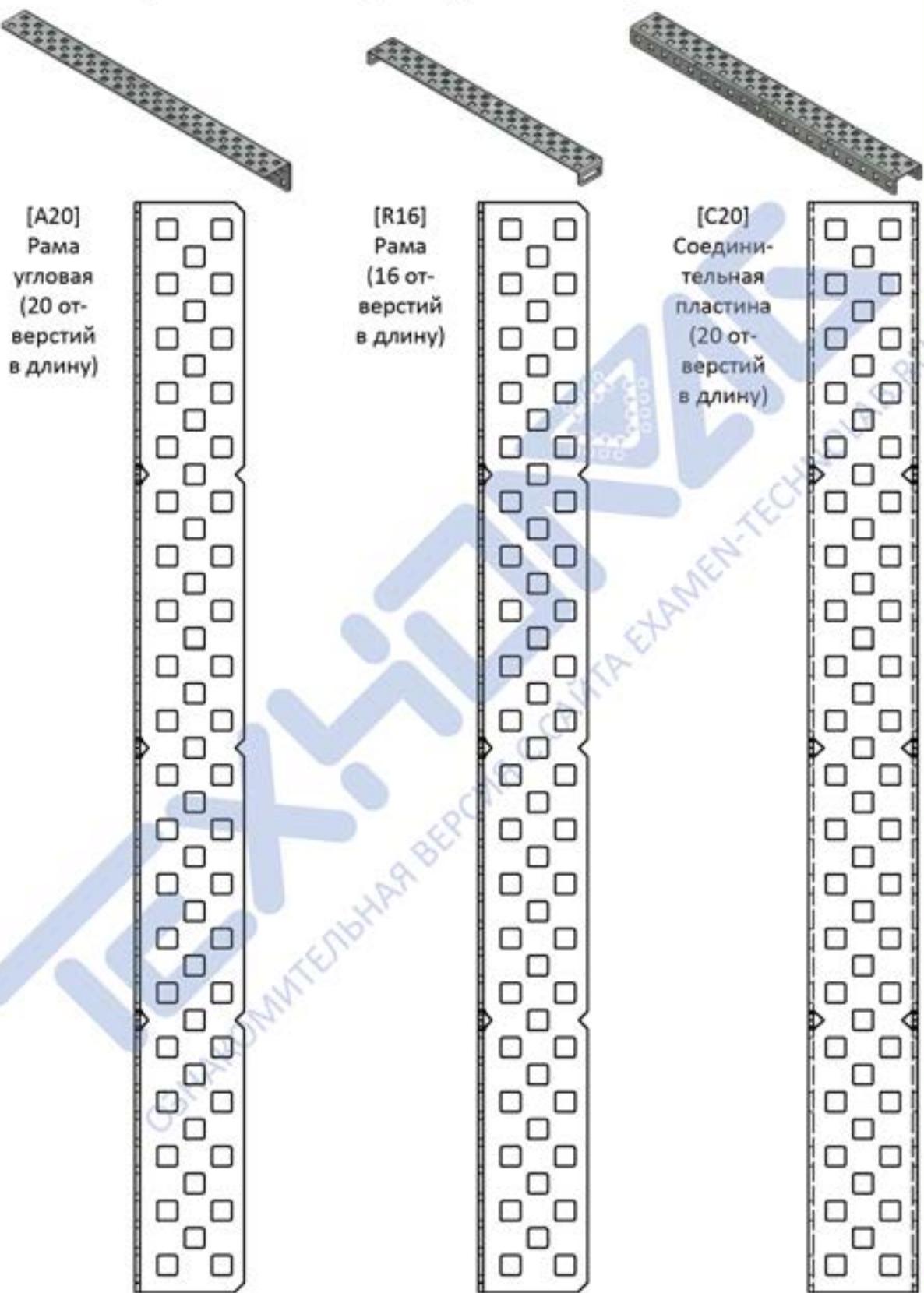
ЭКЗАМЕН
ТЕХНОЛАБ

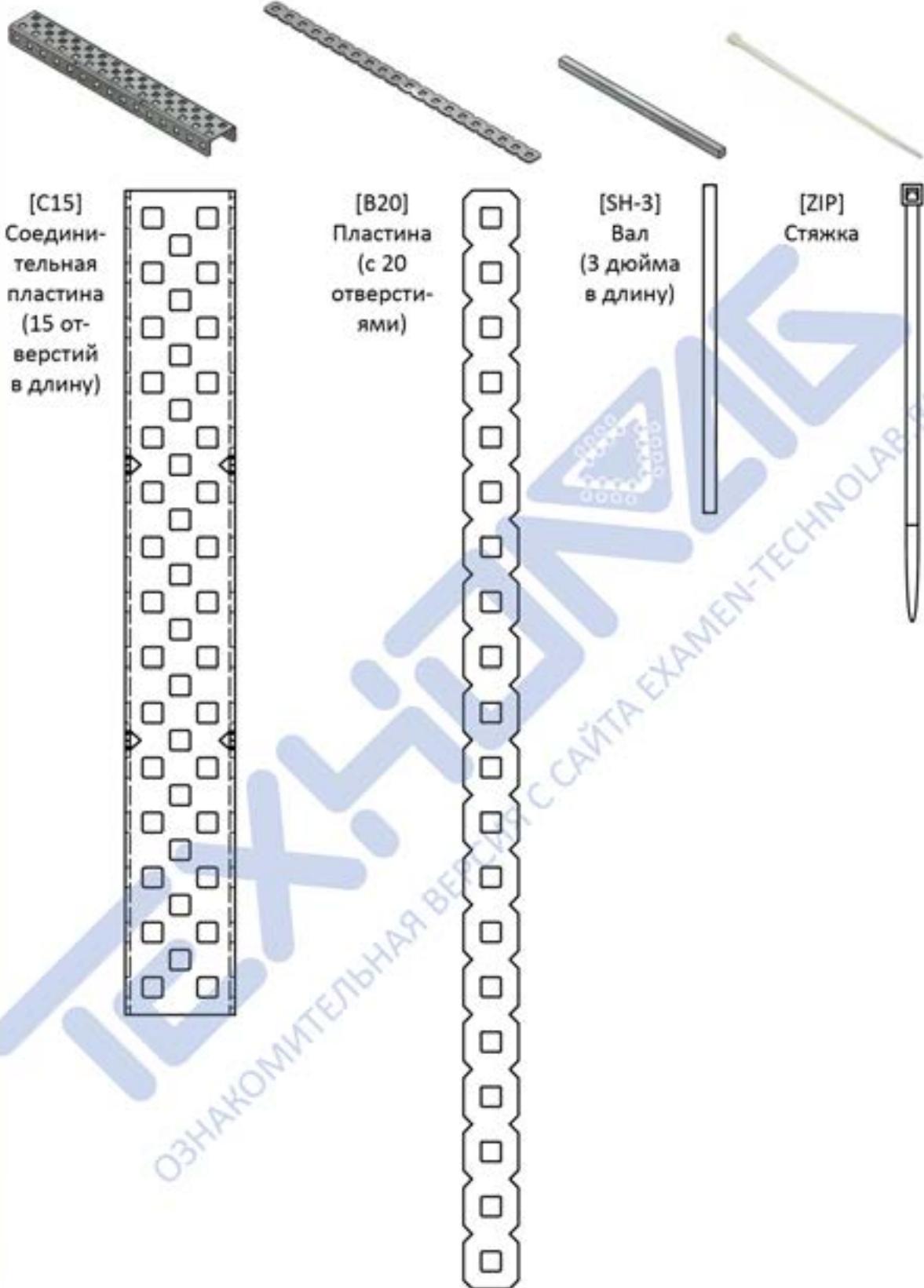




Приложение 3. Руководство по сборке Clawbot

Приложение 3







[BF] ОПОРНАЯ ПЛАНКА



[ST-1] РАЗДЕЛИТЕЛЬ



[BR-I]
ЗАКЛЕПКА (ВНУТР.)



[BR-O]
ЗАКЛЕПКА (ВНЕШН.)



[CPLR] СОЕДИНИТЕЛЬ ВАЛОВ



[CP] СОЕДИНИТЕЛЬНЫЙ ЭЛЕМЕНТ (ВНУТР.)



[SP4.8] РАЗДЕЛИТЕЛЬ (4.8 ММ)



[NK] #8-32 ГАЙКА



[NL] #8-32 ГАЙКА С НЕЙЛОННЫМ ДЕРЖАТЕЛЕМ



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



[SS-L] ДЛИННЫЙ ВИНТ МОТОРА С НЕЙЛОННЫМ ДЕРЖАТЕЛЕМ



[SS-S] КОРОТКИЙ ВИНТ МОТОРА С НЕЙЛОННЫМ ДЕРЖАТЕЛЕМ



[S4] ВИНТ 8-32x1/2 ДЮЙМА



[S2] ВИНТ 8-32x1/4 ДЮЙМА



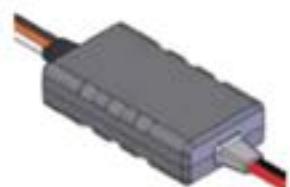
[S12] ВИНТ 8-32x1.5 ДЮЙМА



[CLAW]
УСТРОЙСТВО СХВАТА
(показано не в масштабе)



[M393]
ДВУХПРОВОДНОЙ МОТОР 393
(показано не в масштабе)



[MC29]
КОНТРОЛЛЕР МОТОРА 29
(показано не в масштабе)



[BKT]
КРЕПЛЕНИЕ БАТАРЕИ
(показано не в масштабе)



[BATT]
ПЕРЕЗАРЯЖАЕМАЯ
БАТАРЕЯ 7.2В
(показано не в масштабе)



[CTX]
МИКРОКОНТРОЛЛЕР
(показано не в масштабе)



[VNET]
КЛЮЧ VEXnet
(показано не в масштабе)



[G12]
ШЕСТЕРНЯ
(12 ЗУБЬЕВ)
(показано не в масштабе)



[G60]
ШЕСТЕРНЯ
(60 ЗУБЬЕВ)
(показано не в масштабе)



[G84]
ШЕСТЕРНЯ
(84 ЗУБЬЕВ)
(показано не в масштабе)



[W4]
КОЛЕСО 4 ДЮЙМА
(показано не в масштабе)



5/64"



3/32"



ГАЕЧНЫЙ КЛЮЧ



[5/64"]



[SS-5] 1/4" ВИНТ МОТОРА



[SS-L] 1/2" ВИНТ МОТОРА



[11/32"]



[NK] 8-32 ГАЙКА



[NL] 8-32 ГАЙКА С НЕЙЛОННЫМ ДЕРЖАТЕЛЕМ

[1/4"]



[ST-1]



[5/64"]



[3/32"]



[S2] 1/4" ВИНТ



[S4] 1/2" ВИНТ



[S12] 1.5" ВИНТ



[5/64"]



[BR] ЗАКЛЮЧКИ



[BF] ОПОРНАЯ ПЛАНКА



[3/32"]

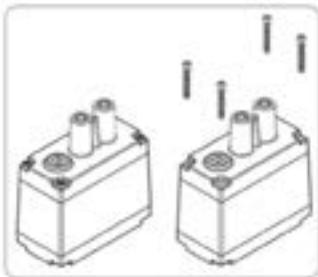


[BF] ОПОРНАЯ ПЛАНКА

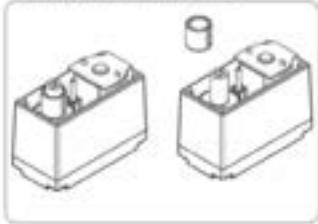
ВЫСОКАЯ СКОРОСТЬ/НИЗКИЙ КРУТИЩИЙ МОМЕНТ для моторов (необязательная опция):

Для модификации двухпроводного мотора 393 в режим повышенной скорости - просто замените шестерню мотора на одну из прилагаемых шестерней для замены, следуя инструкции

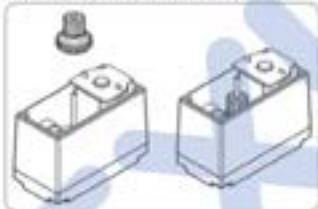
1. Открутите и удалите винты с передней части мотора.



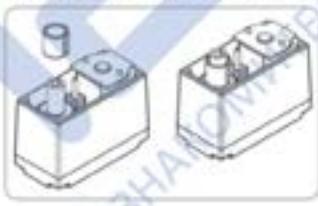
3. Снимите втулку, как показано на рисунке, и отложите в сторону. Она понадобится позже.



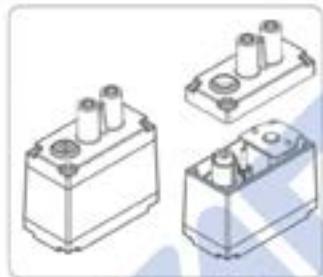
5. Установите центральную шестернию повышенной скорости.



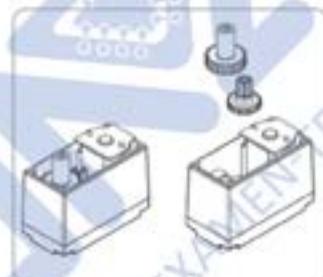
7. Установите втулку, снятую в шаге №3. Убедитесь, что она установлена как на рисунке.



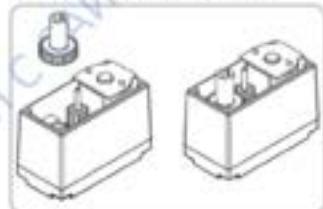
2. Снимите крышку с передней части мотора. Не трогайте шестерни внутри.



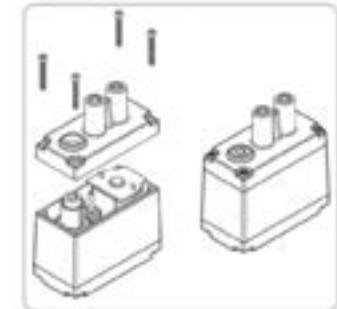
4. Снимите центральную шестернию и шестернию вала.



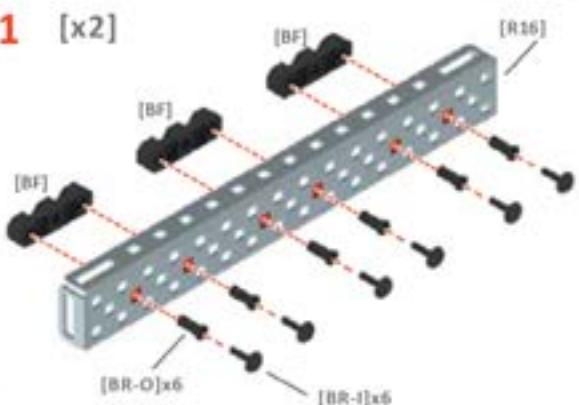
6. Установите шестернию вала повышенной скорости.



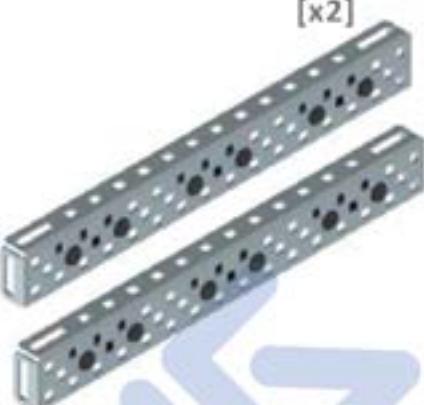
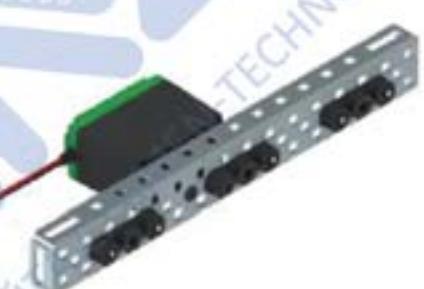
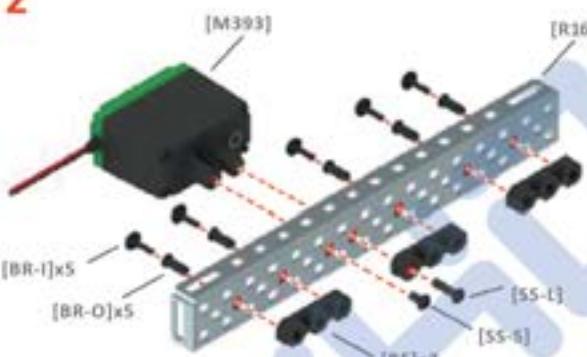
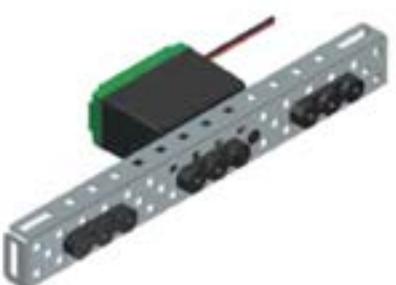
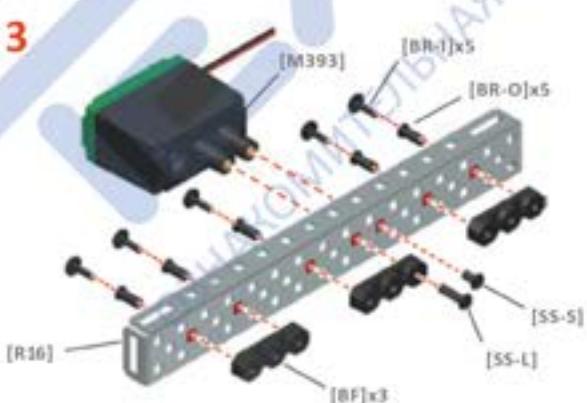
8. Установите обратно крышку и винты, снятые в шагах №1 и №2

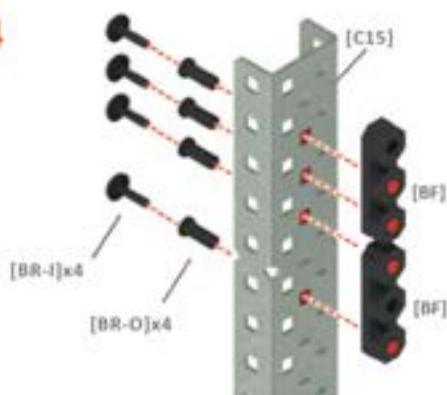
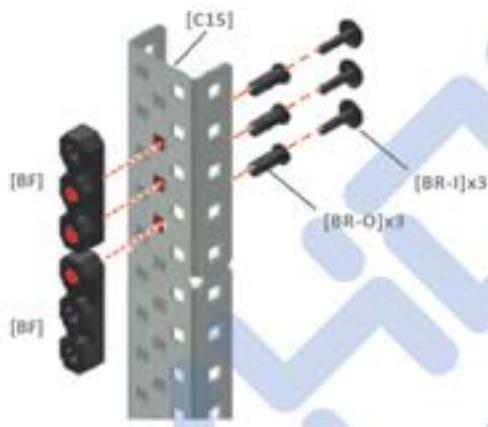
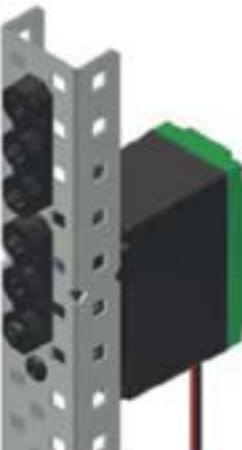
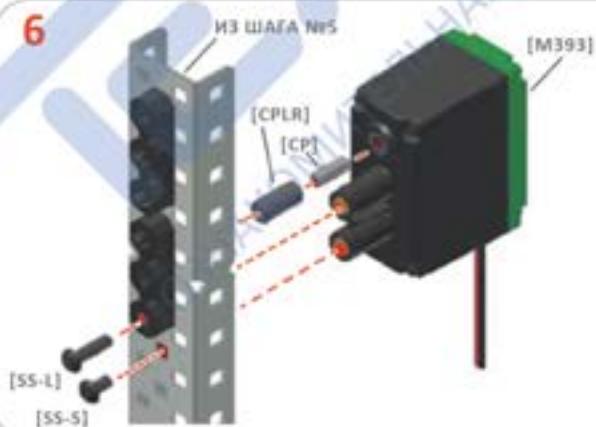


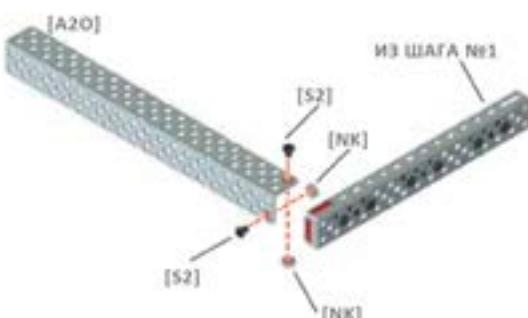
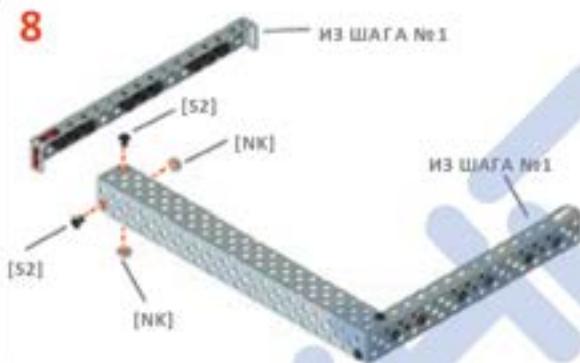
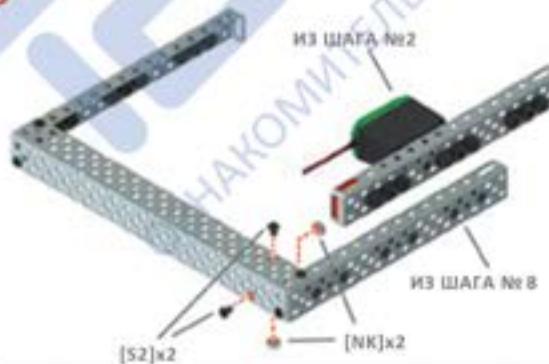
Режим повышенной скорости обеспечивает ускорение вращения вала на 60% и снижение крутящего момента на 60%.

1 [x2]

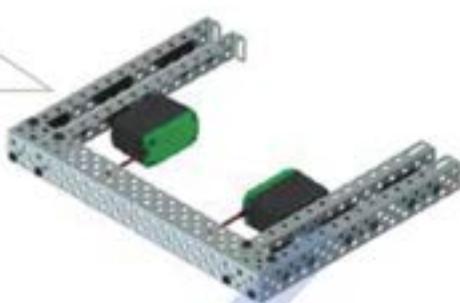
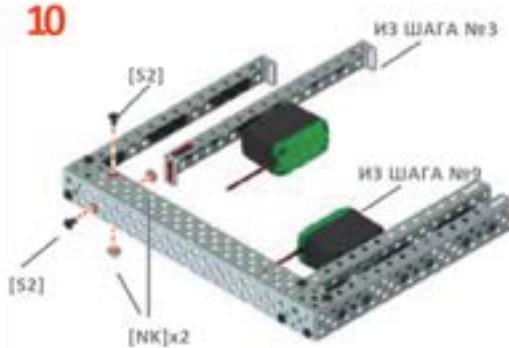
[x2]

**2****3**

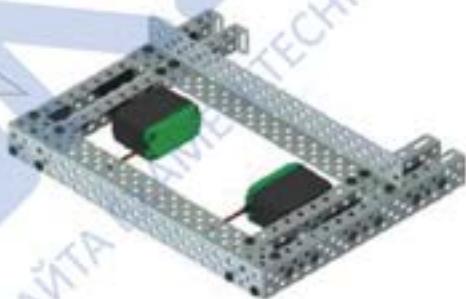
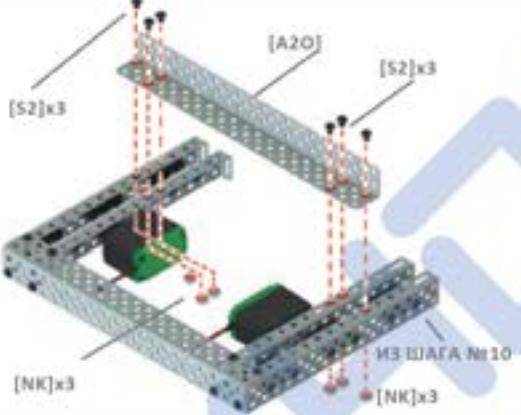
4**5****6**

7**8****9**

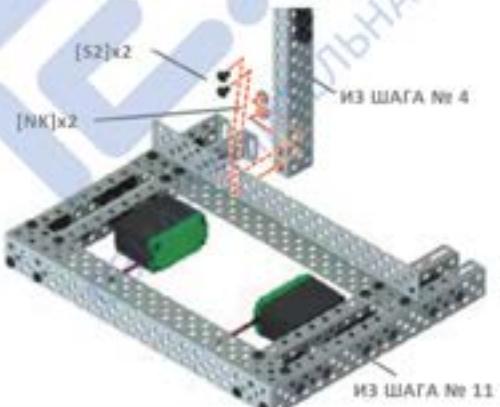
10

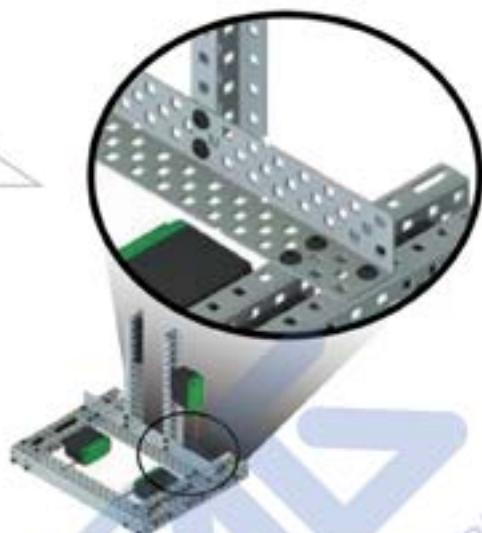
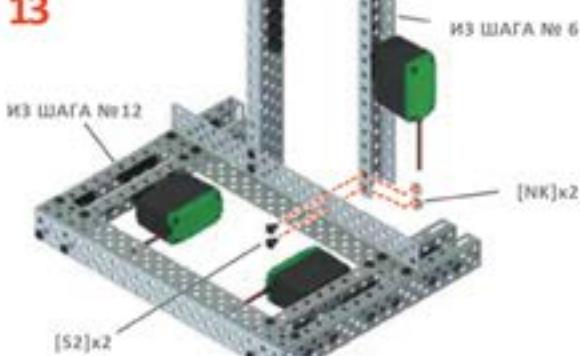
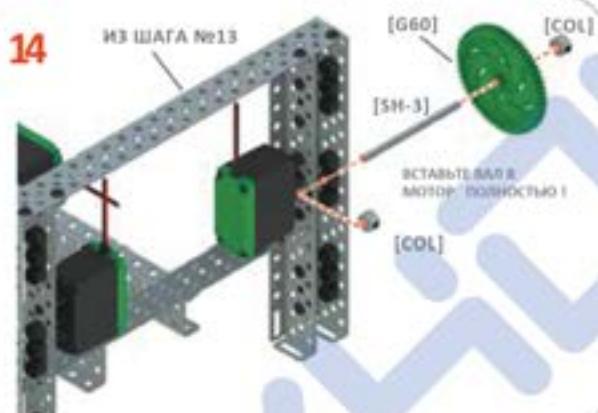
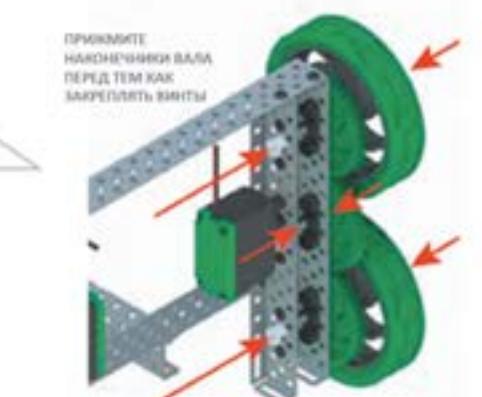
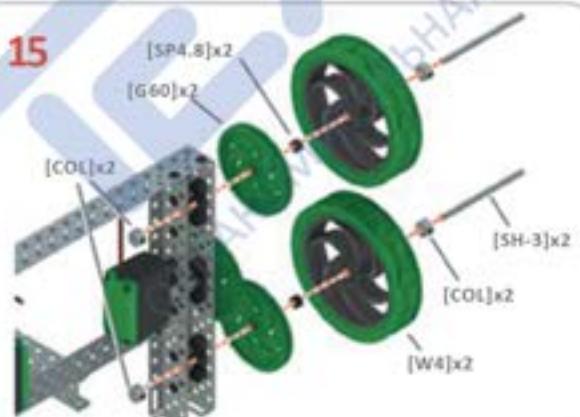


11



12



13**14****15**

16

[COL]

[SH-3]

ВСТАВЬТЕ ВАЛ В МОТОР ПОЛНОСТЬЮ!

[COL]

ПРИКЛЮЧИТЕ НАКОНЕЧНИКИ ВАЛА, ПЕРЕД ТЕМ КАК ЗАКРЫТЬ ВИНТЫ

17

[SH-3]x2

[COL]x2

[W4]x2

[SP4.8]x2

ПРИКЛЮЧИТЕ НАКОНЕЧНИКИ ВАЛА ПЕРЕД ТЕМ КАК ЗАКРЫТЬ ВИНТЫ

[G60]x2

ПРИКЛЮЧИТЕ НАКОНЕЧНИКИ ВАЛА ПЕРЕД ТЕМ КАК ЗАКРЫТЬ ВИНТЫ

18

[S2]

[ST-1]

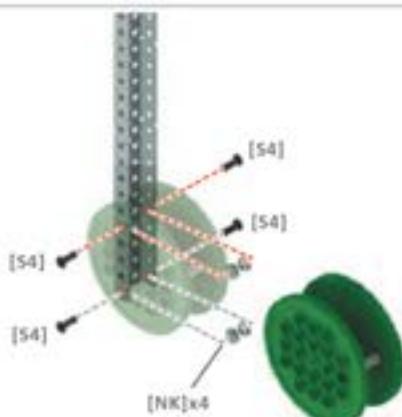
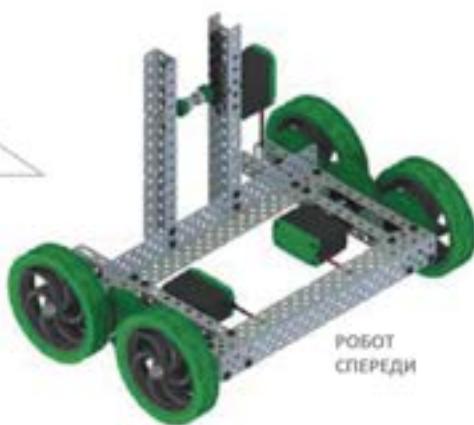
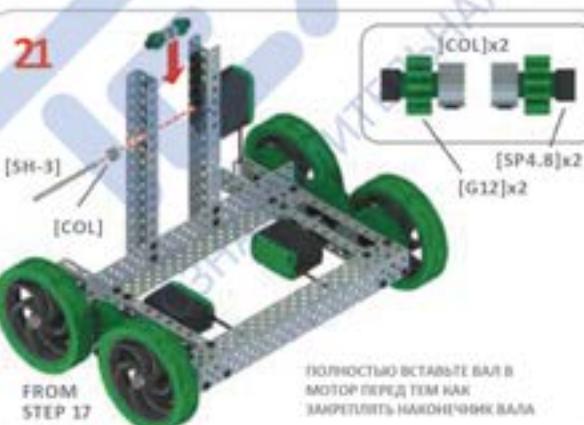
[S2]

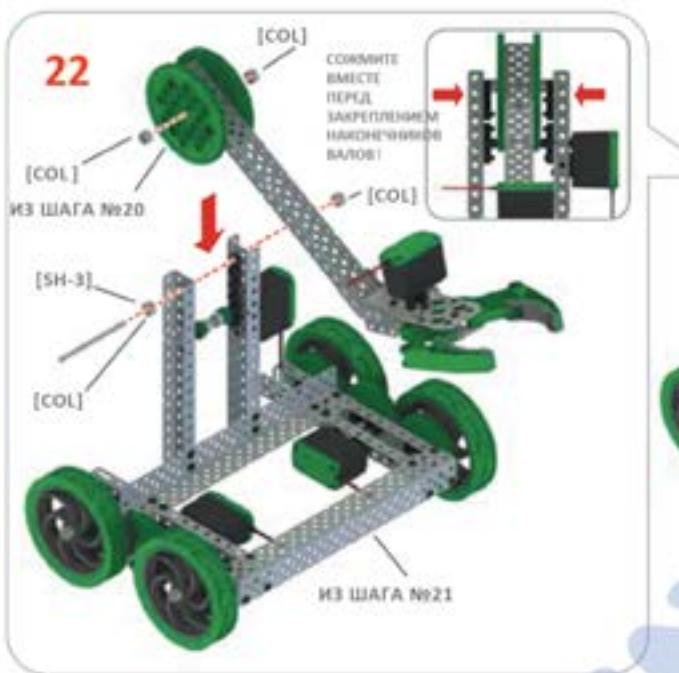
[G84]

[S2]

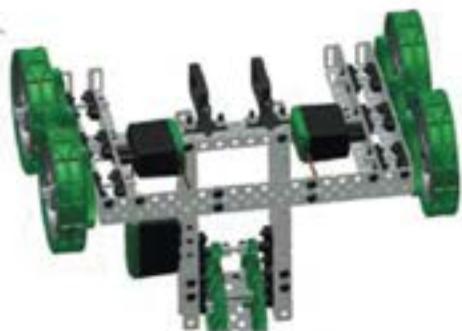
[G84]



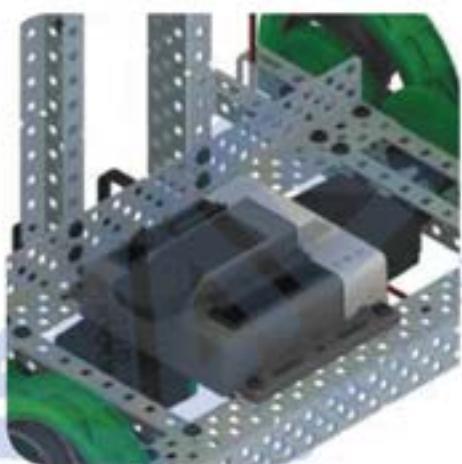
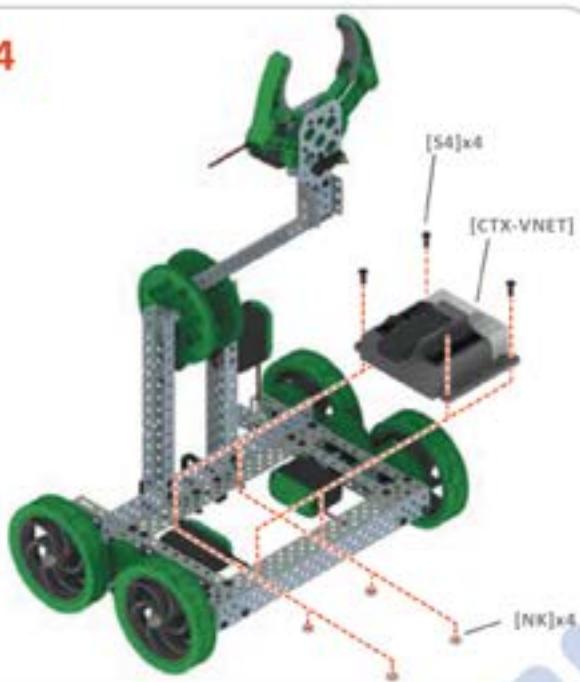
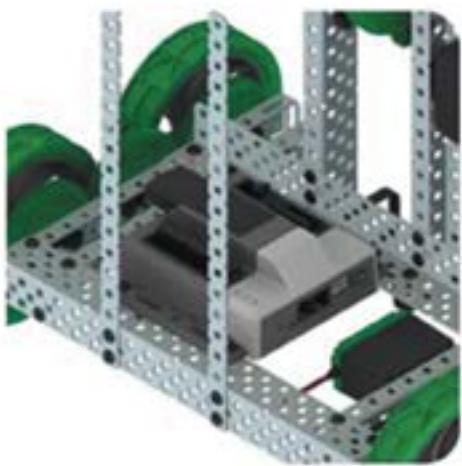
19**20****21**



- ПРОВЕРЬТЕ, ЧТО "РУКА" ПОДНИМАЕТСЯ И ОПУСКАЕТСЯ ПЛАВНО, С НЕБОЛЬШИМ СОПРОТИВЛЕНИЕМ ОТ МОТОРА



УСТАНОВИТЕ КРЕПЛЕНИЕ БАТАРЕИ ПОД РАМОЙ

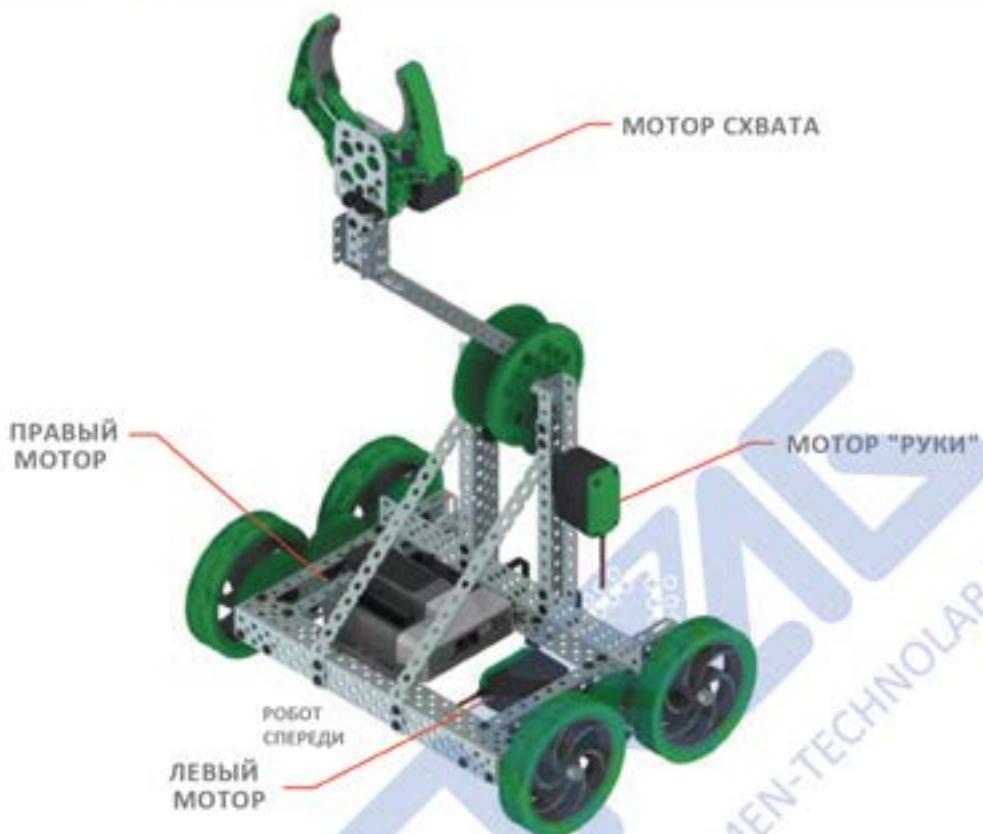
24**25**

26

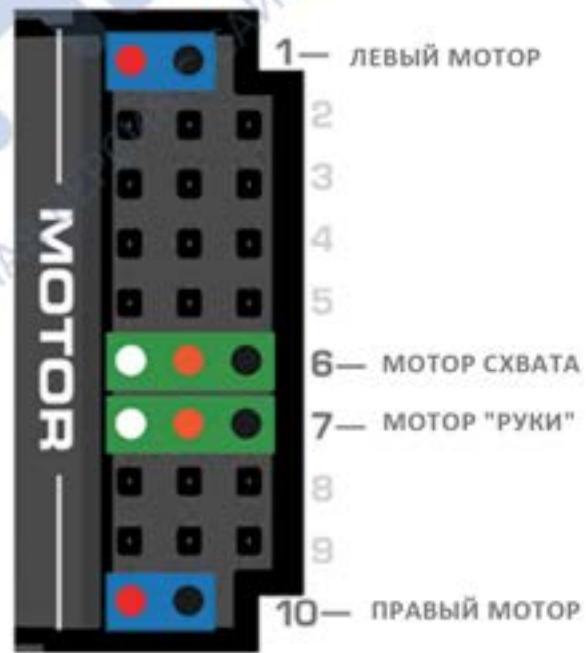


27

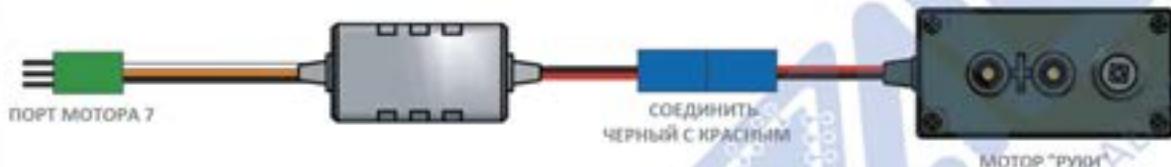
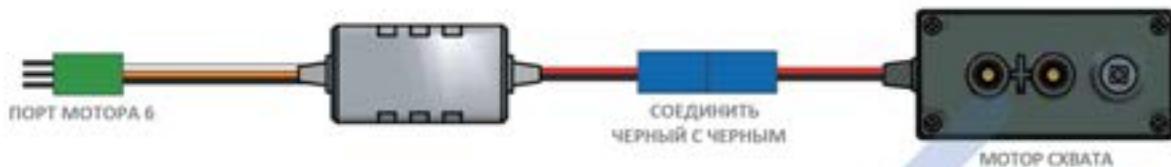




КОНФИГУРАЦИЯ
ОСНОВАНА НА БАЗОВЫХ
НАСТРОЙКАХ
МИКРОКОНТРОЛЛЕРА



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



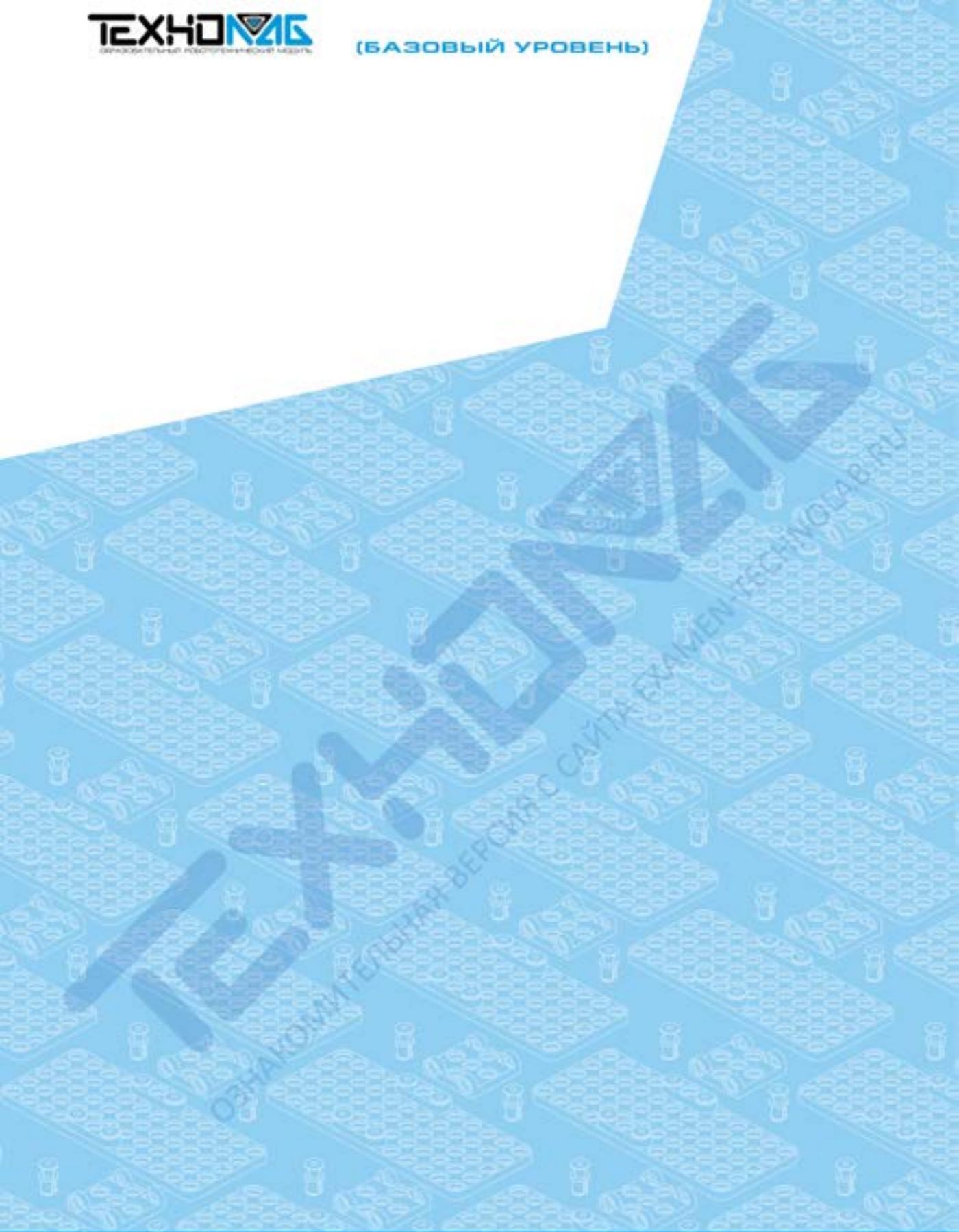
красный провод должен быть подключен ближе к внутренней стороне микроконтроллера (см. рисунок на предыдущей странице)



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

Руководство по сборке мобильного робота с манипулятором





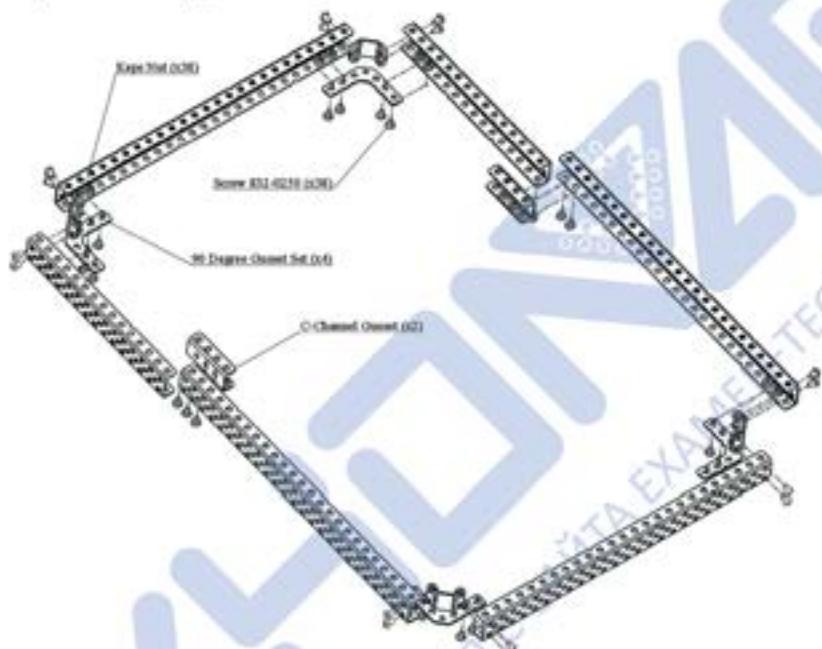
Приложение 4.

Руководство по сборке мобильного робота с манипулятором

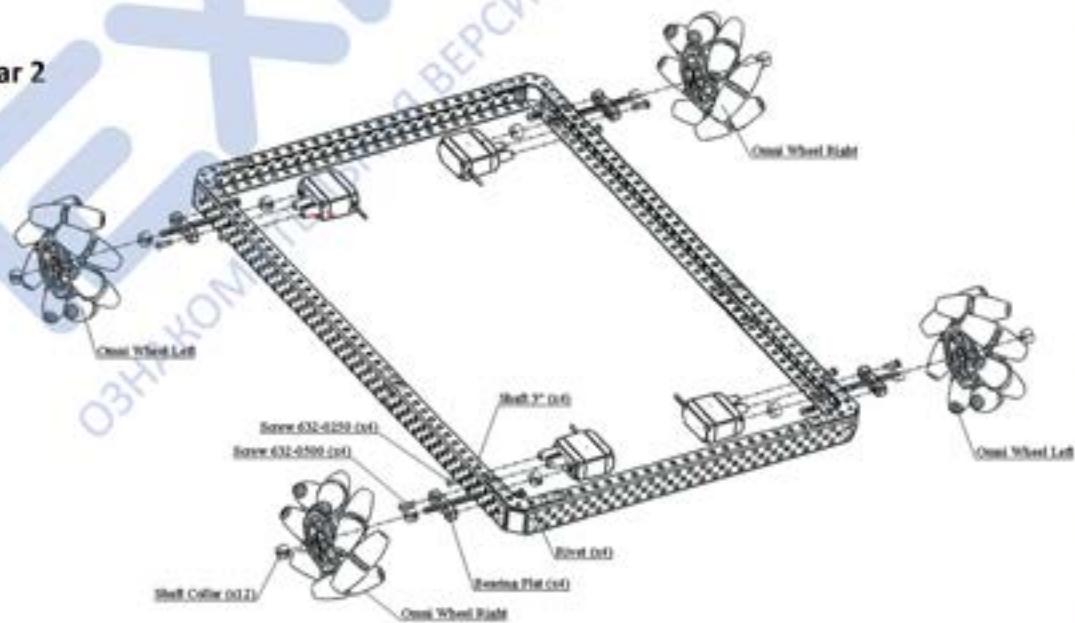
В данном руководстве используются дополнительные детали, не входящие в базовый набор.

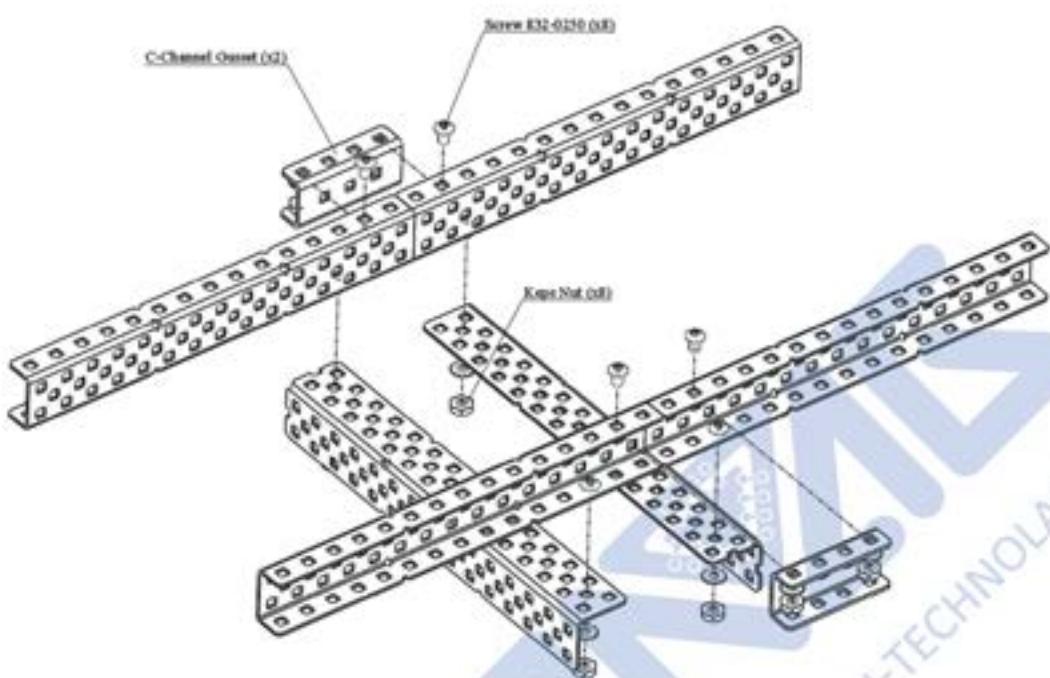
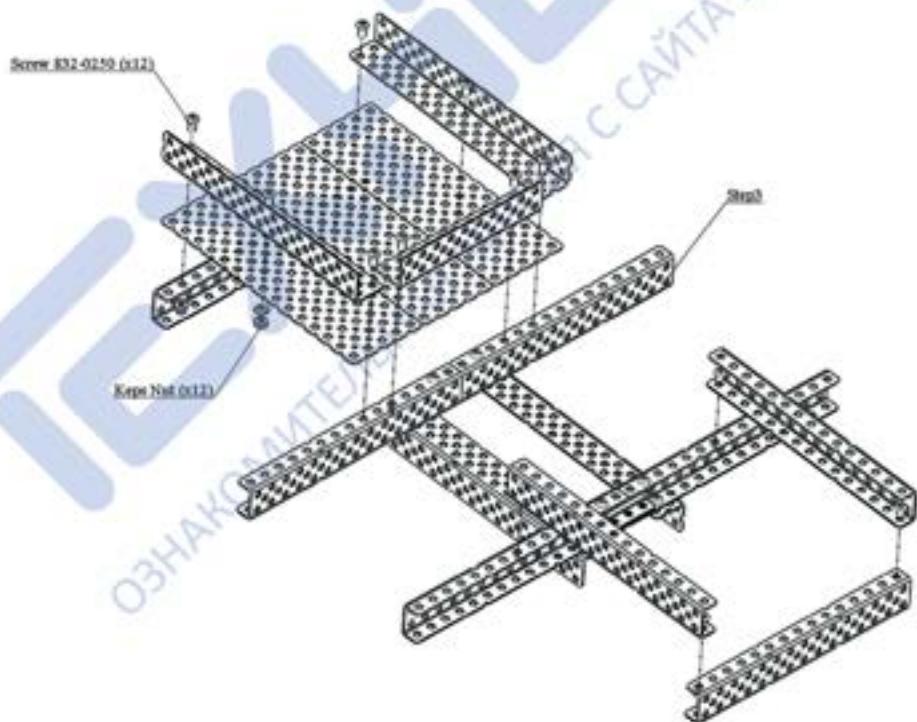
Руководство носит рекомендательный характер и может быть изменено пользователем в соответствии с собственным техническим решением. В том числе данная инструкция содержит этапы сборки конструкции, на которых применяются металлические комплектующие, подвергнутые изменению форме, а именно – сгибанию или резке на части различной длины.

Шаг 1

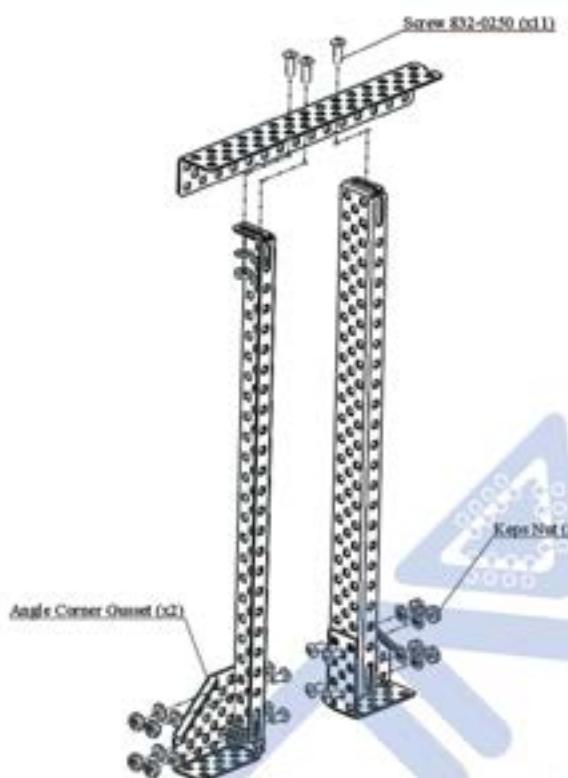


Шаг 2

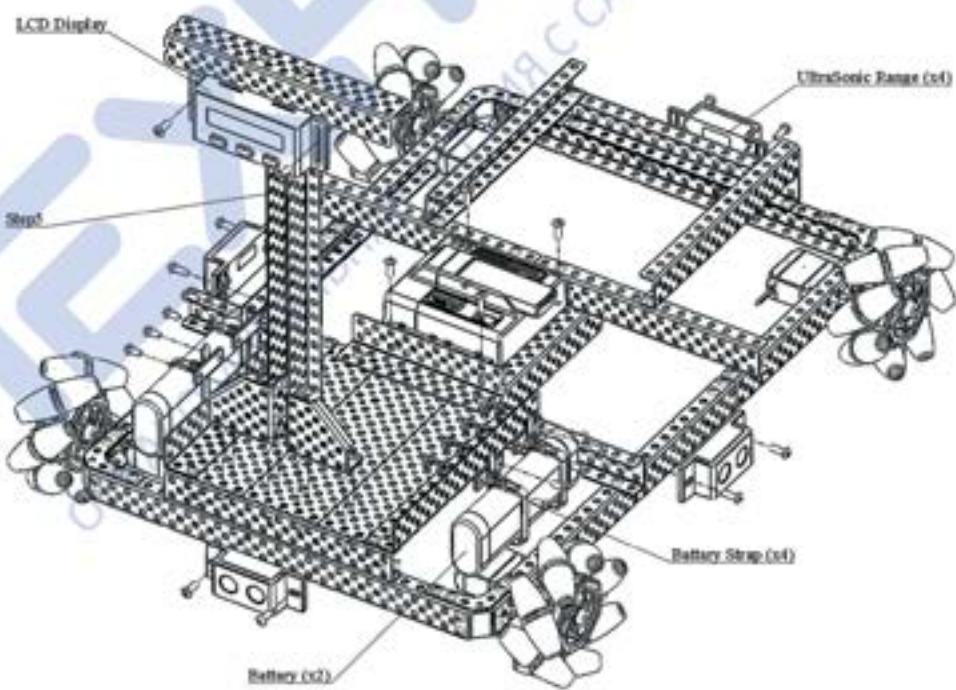


Шаг 3**Шаг 4**

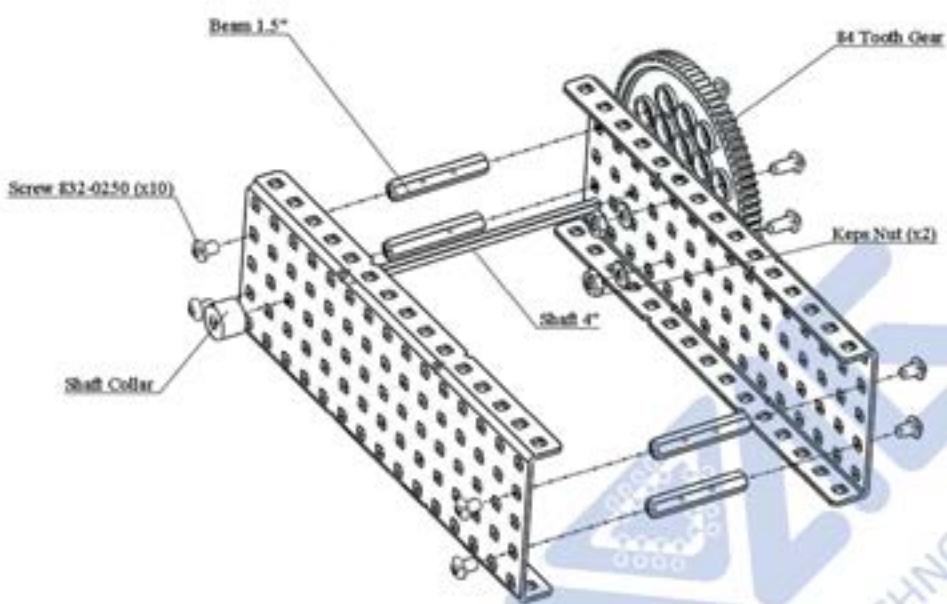
Шаг 5



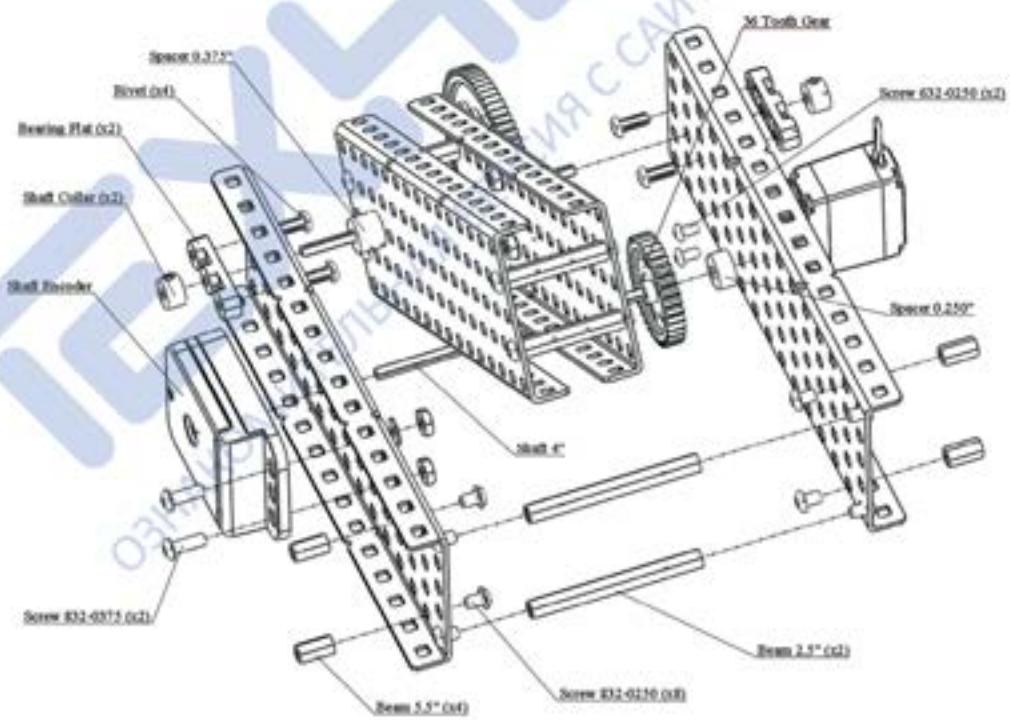
Шаг 6



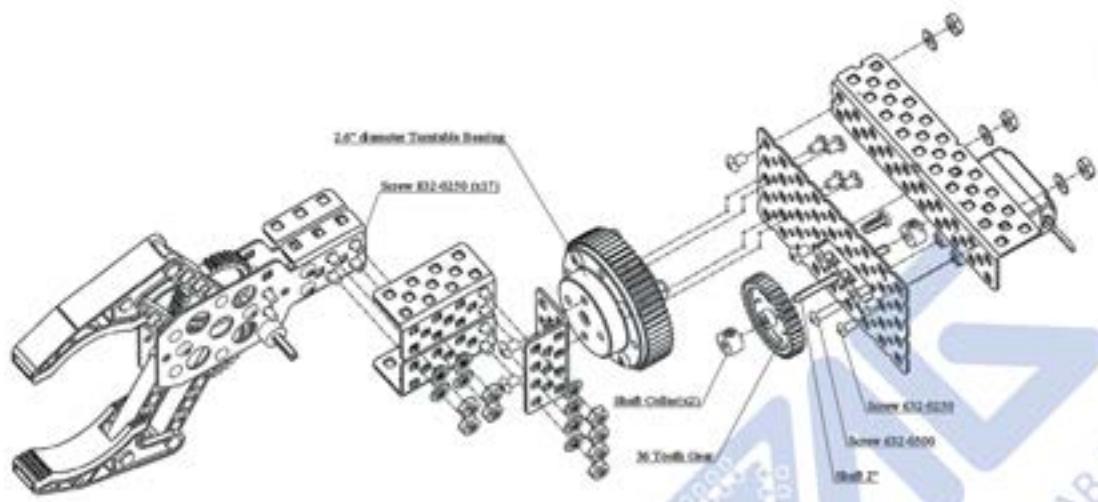
Шаг 7



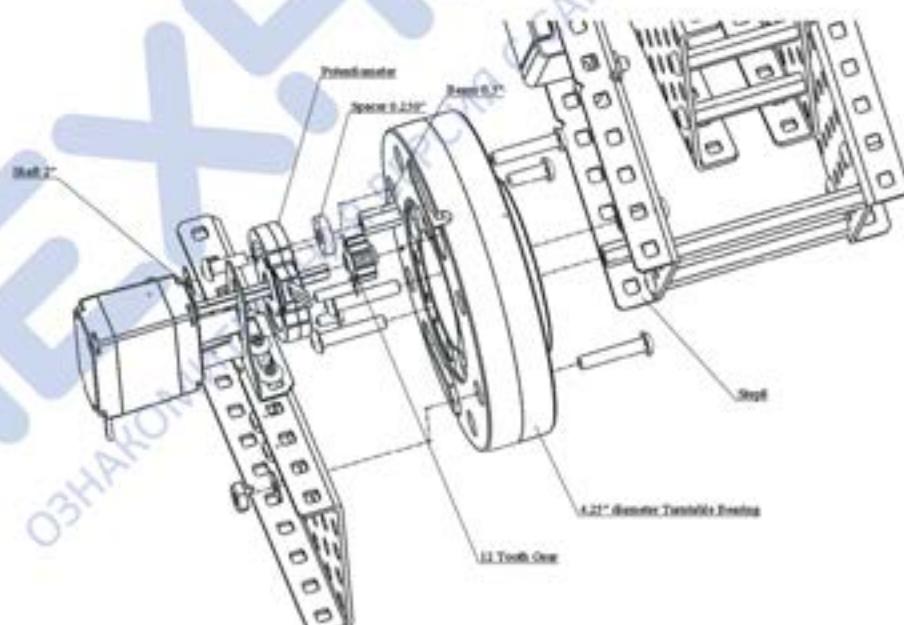
Шаг 8



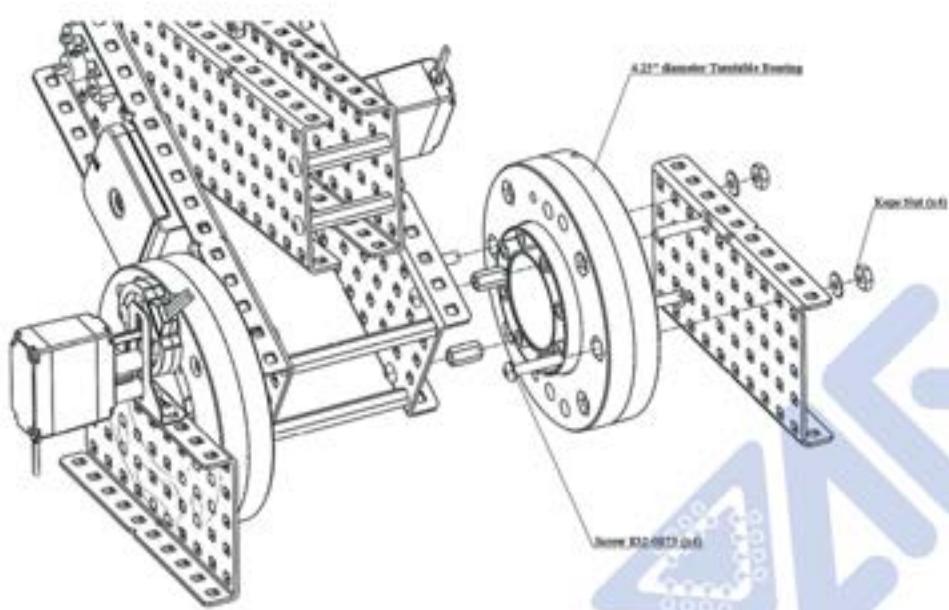
Шаг 9



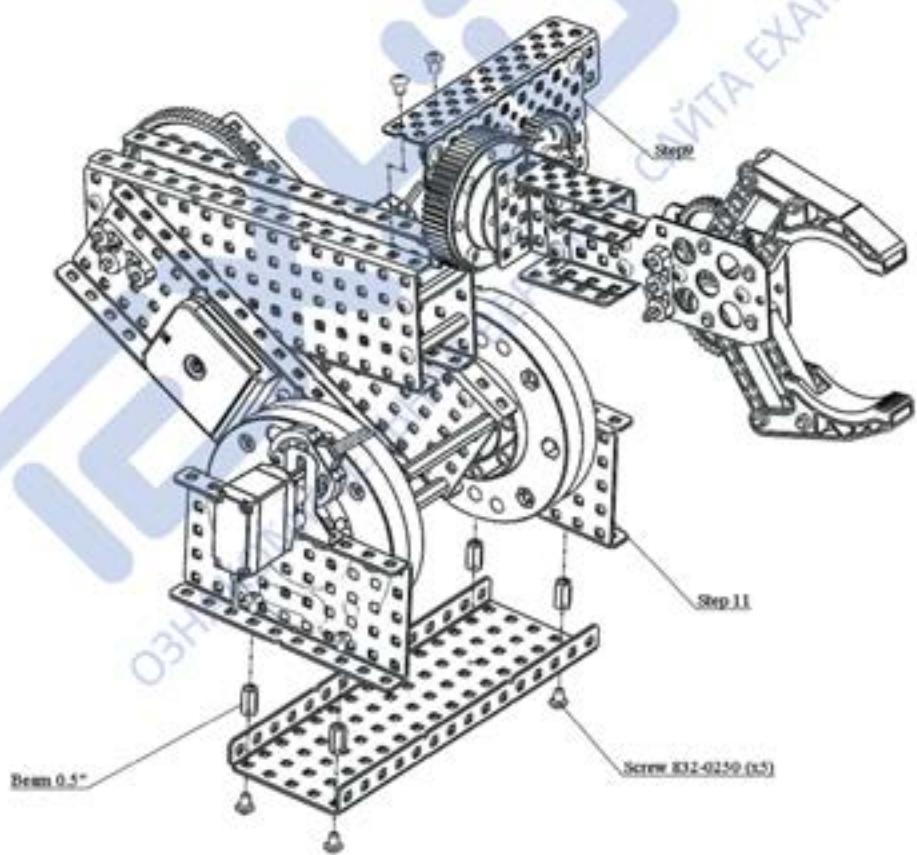
Шаг 10



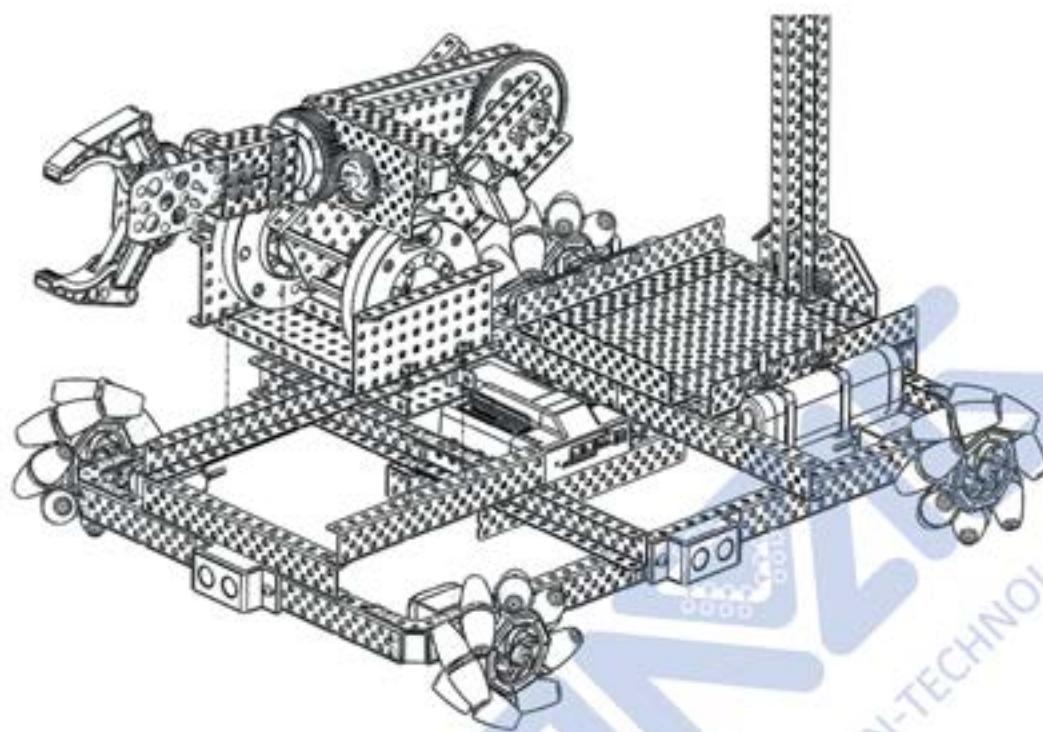
Шаг 11



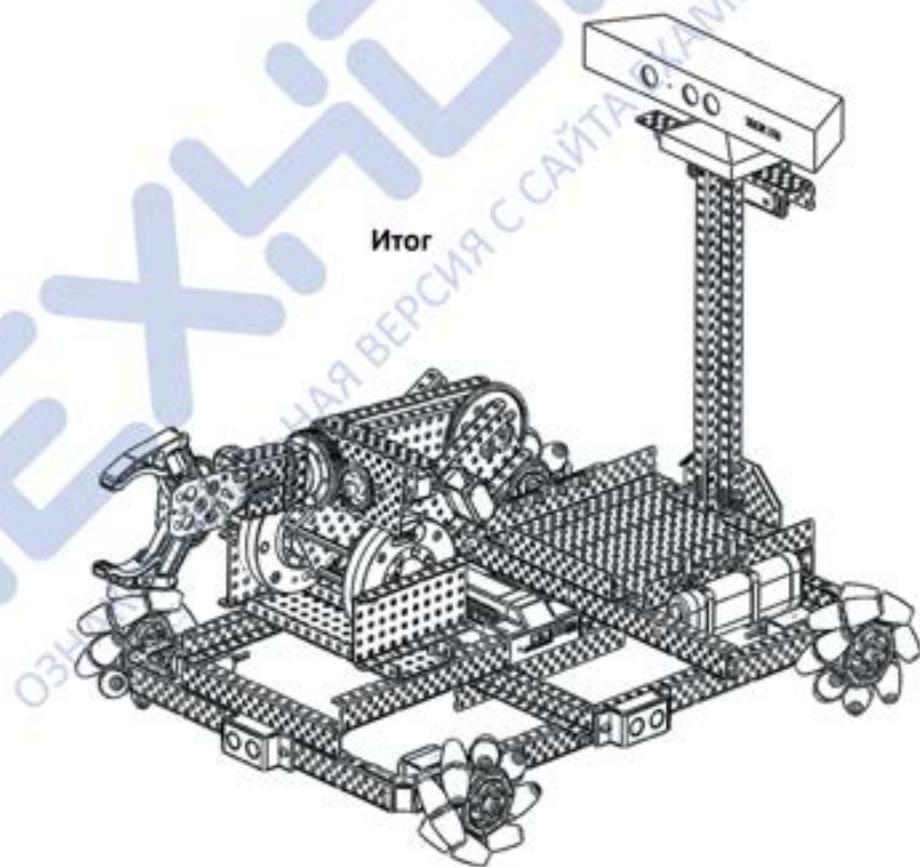
Шаг 12

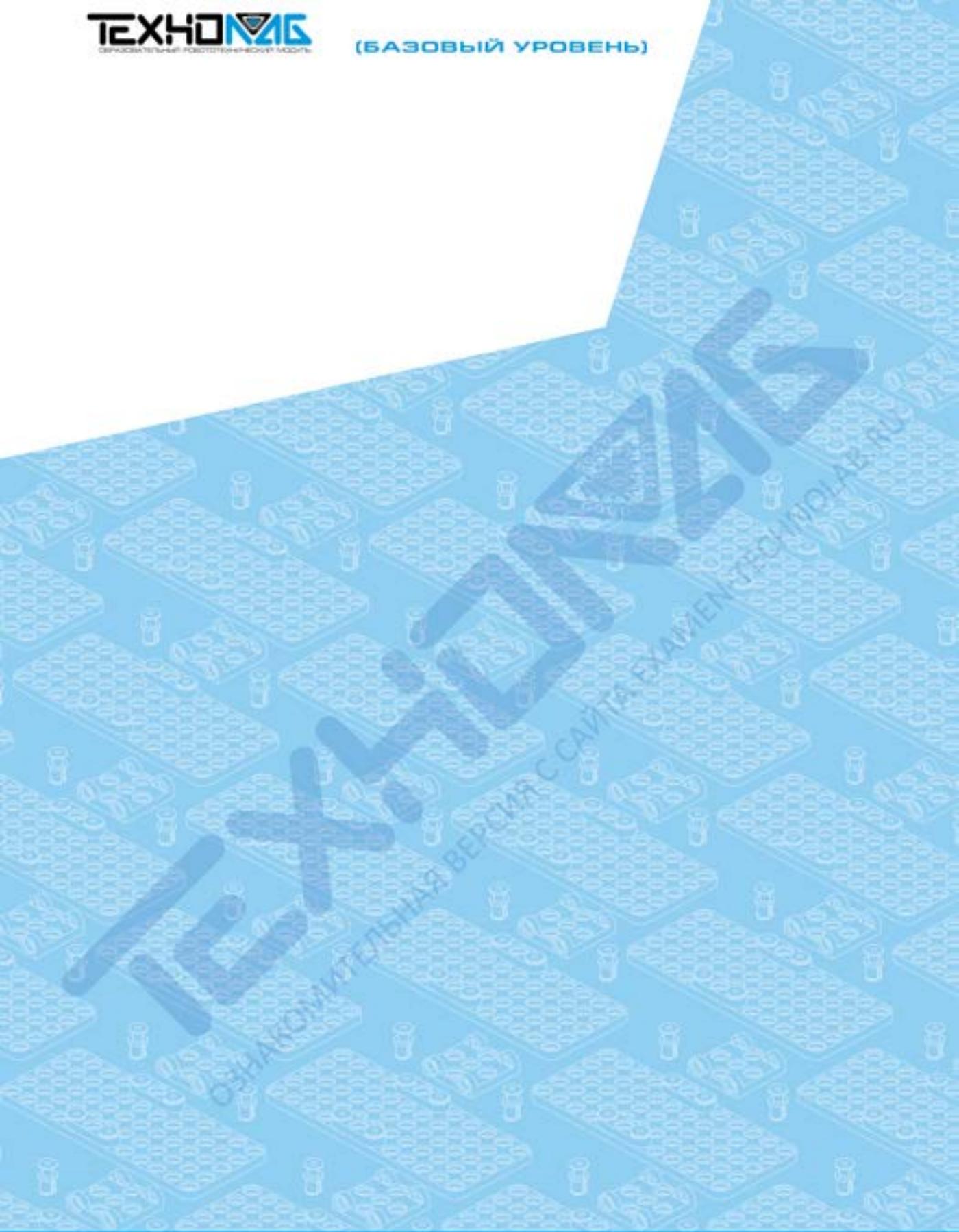


Шаг 13



Итог





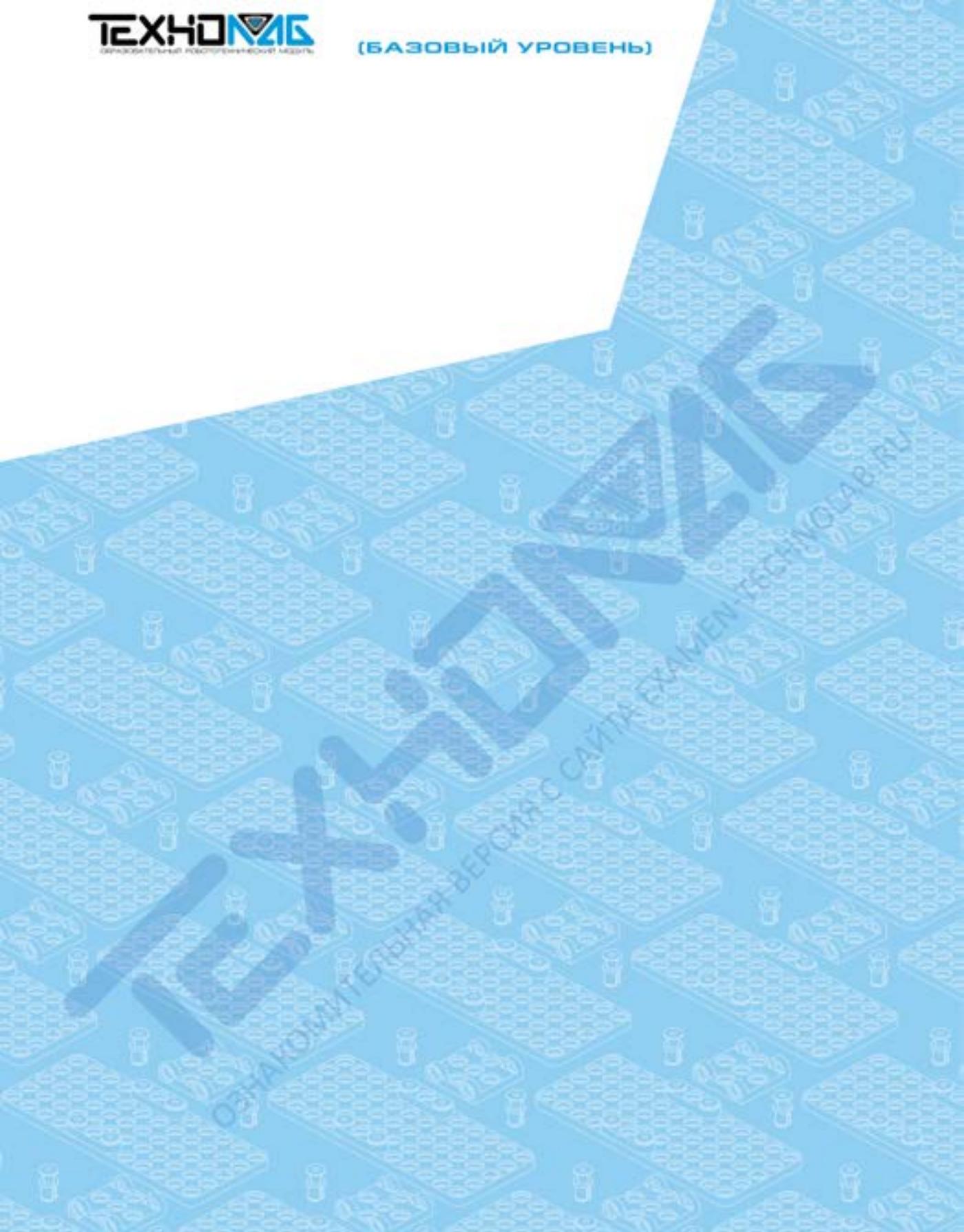
ЭКЗАМЕН
ТЕХНОЛАБ

Руководство по сборке мобильного робота повышенной проходимости

Приложение 5



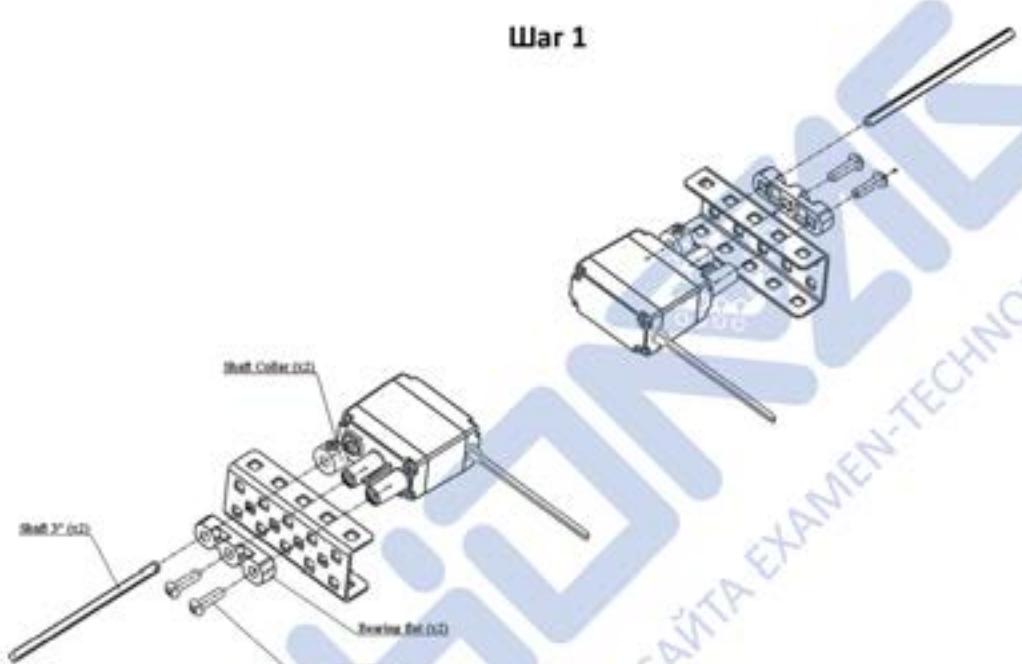
СКАЧАТЬ



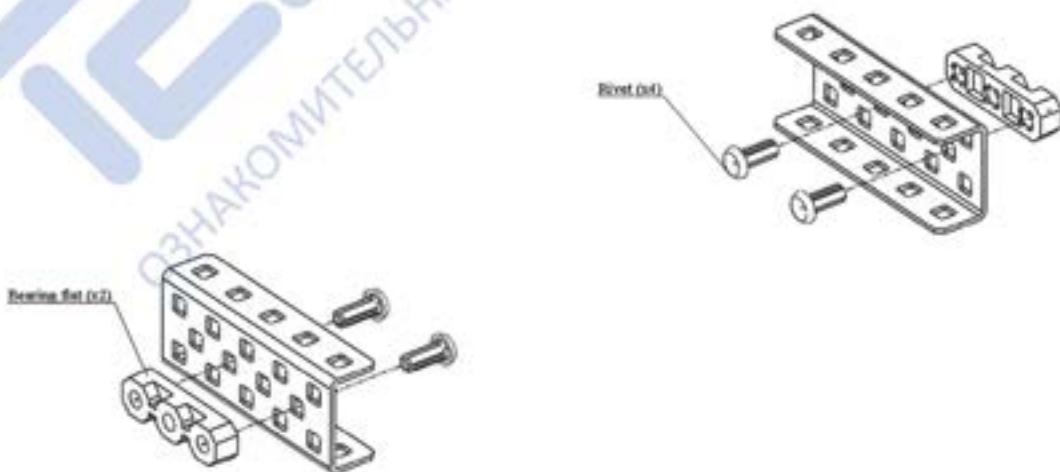
Приложение 5. Руководство по сборке мобильного робота повышенной проходимости

В данном руководстве используются дополнительные детали, не входящие в базовый набор.

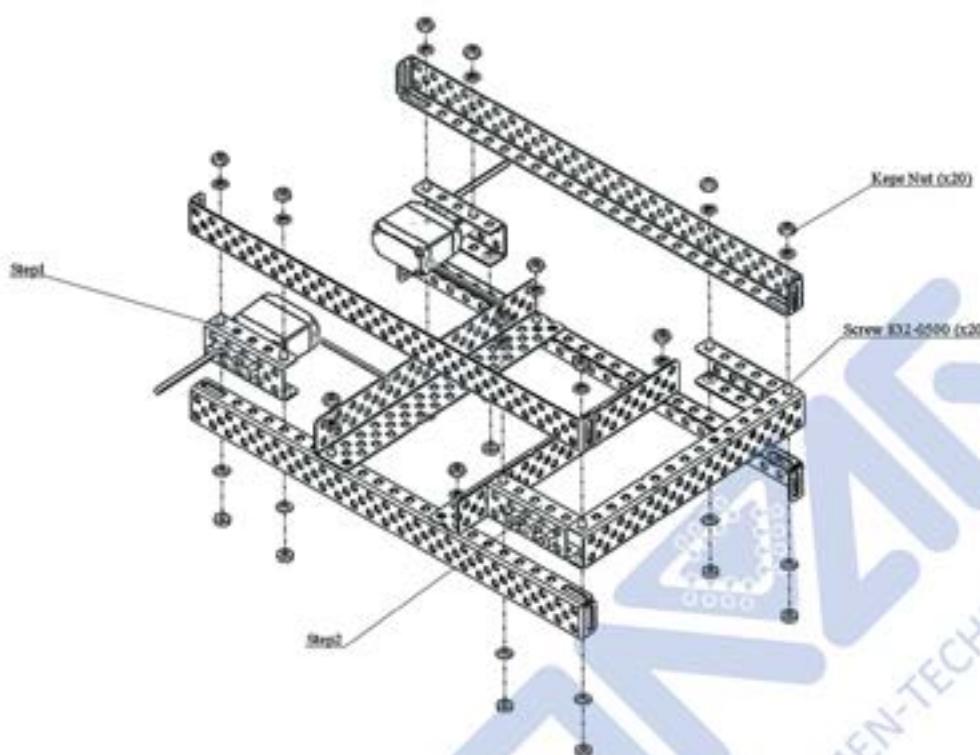
Шаг 1



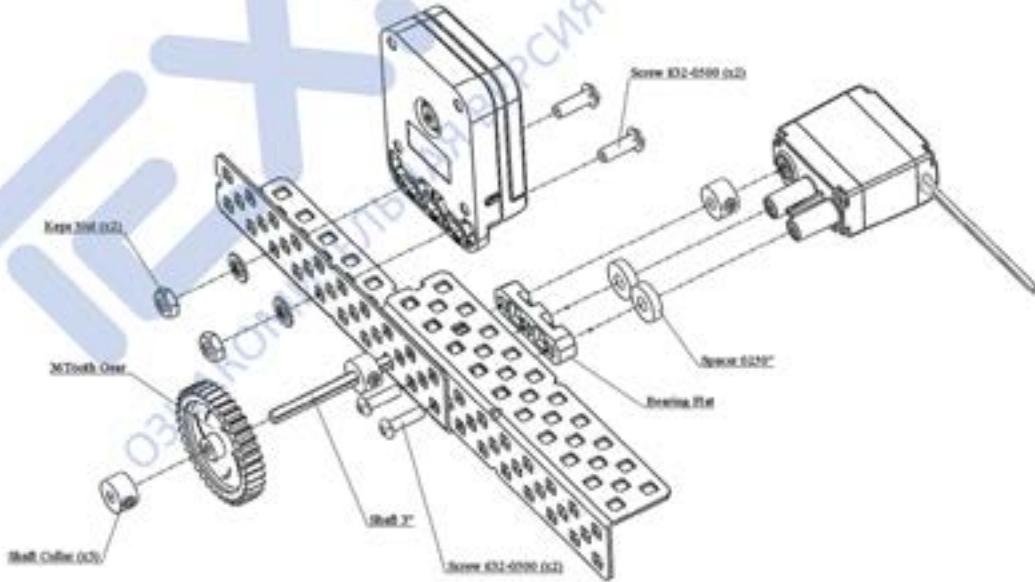
Шаг 2

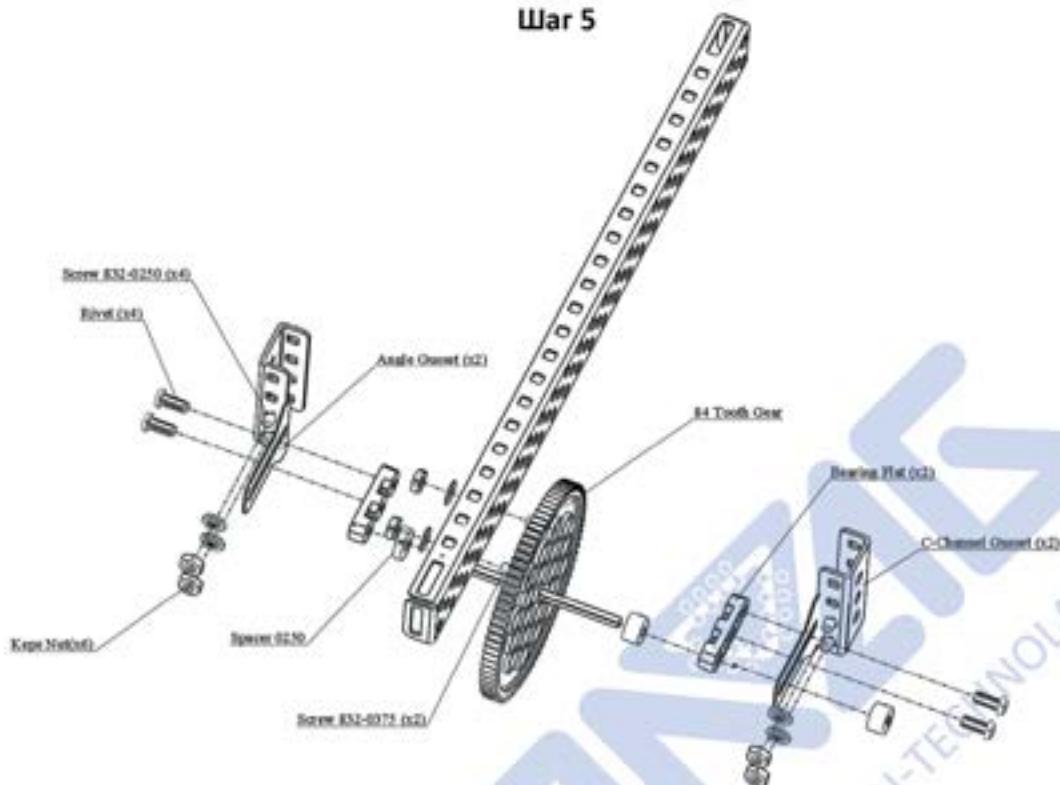


Шаг 3

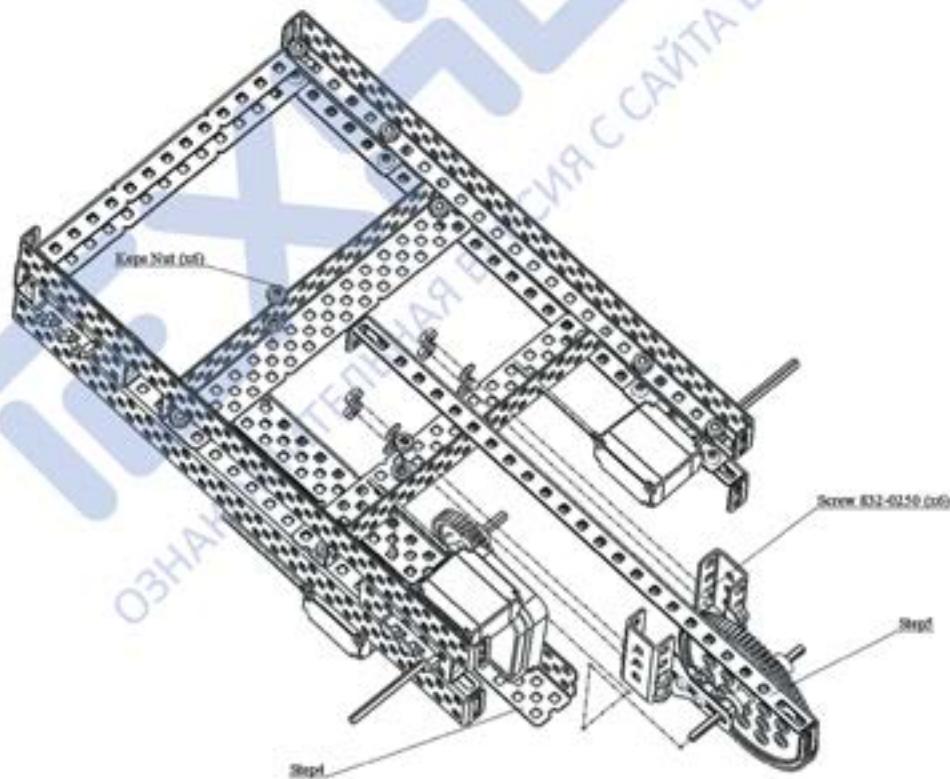


Шаг 4

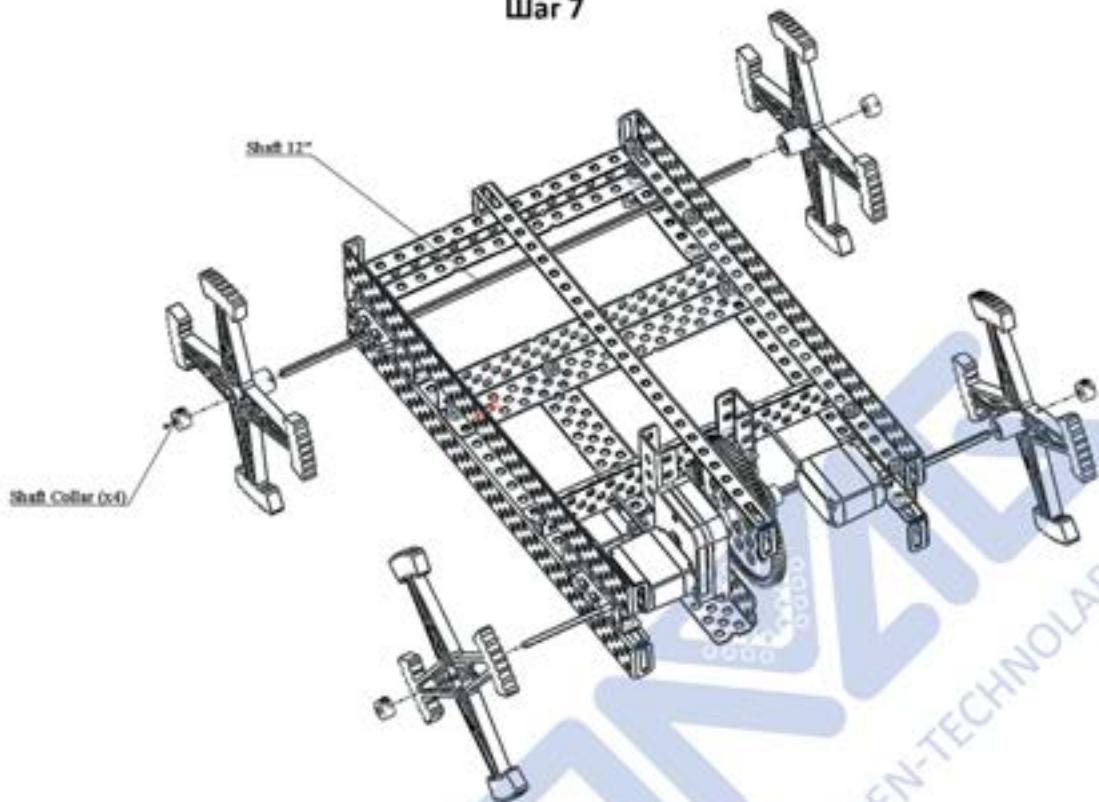


Шаг 5**Шаг 6**

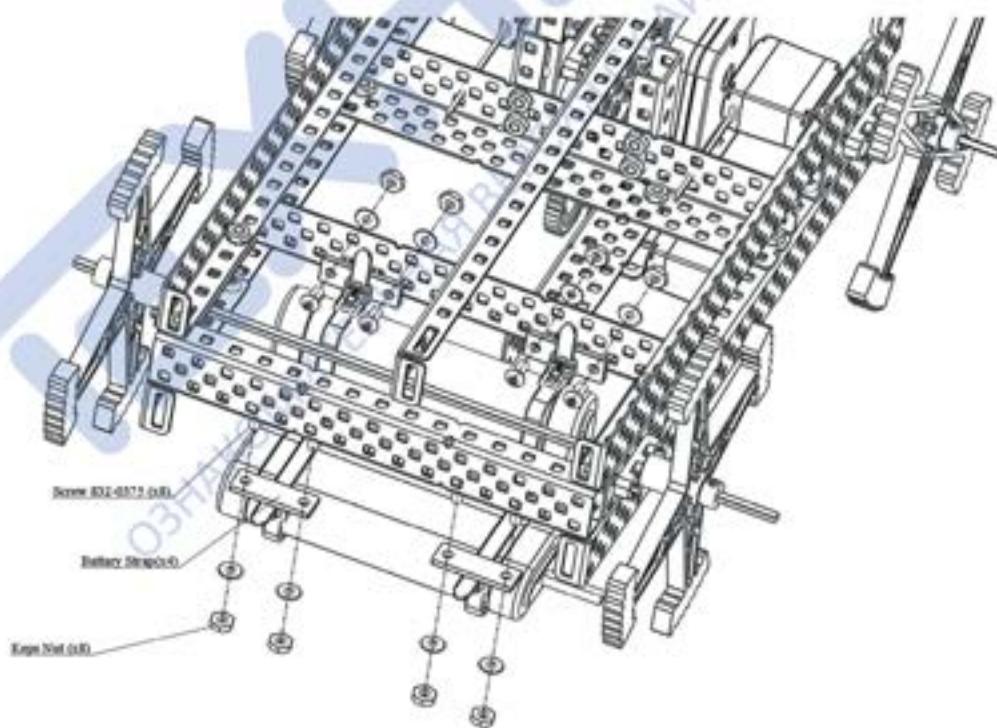
Вал из Шага 5 вставить в соответствующее отверстие в Shaft Encoder.

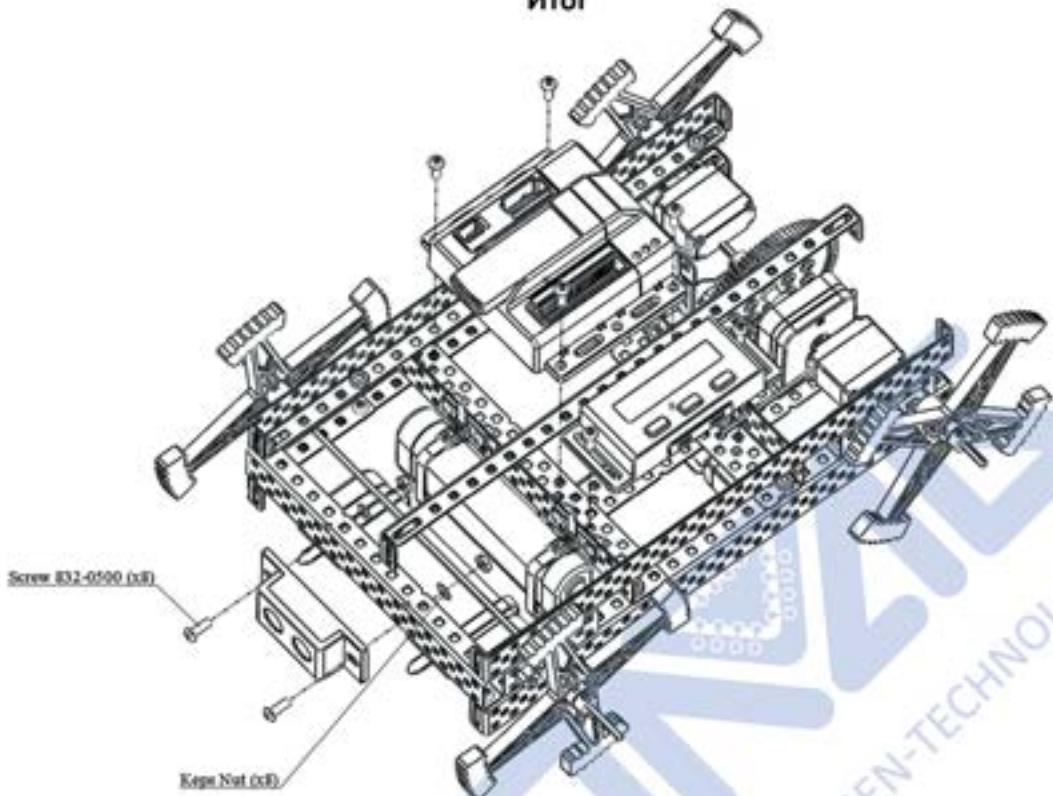


Шаг 7



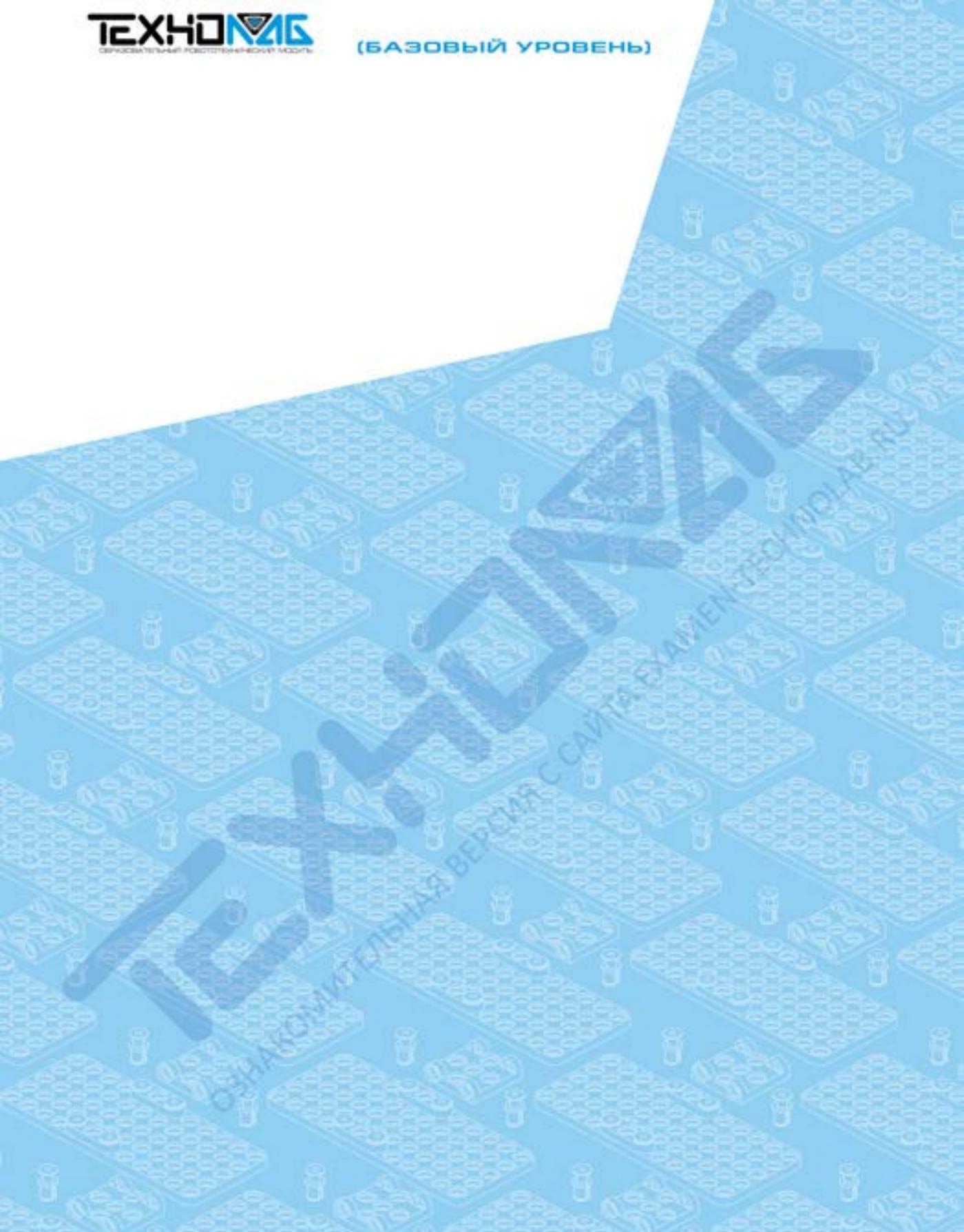
Шаг 8



Итог

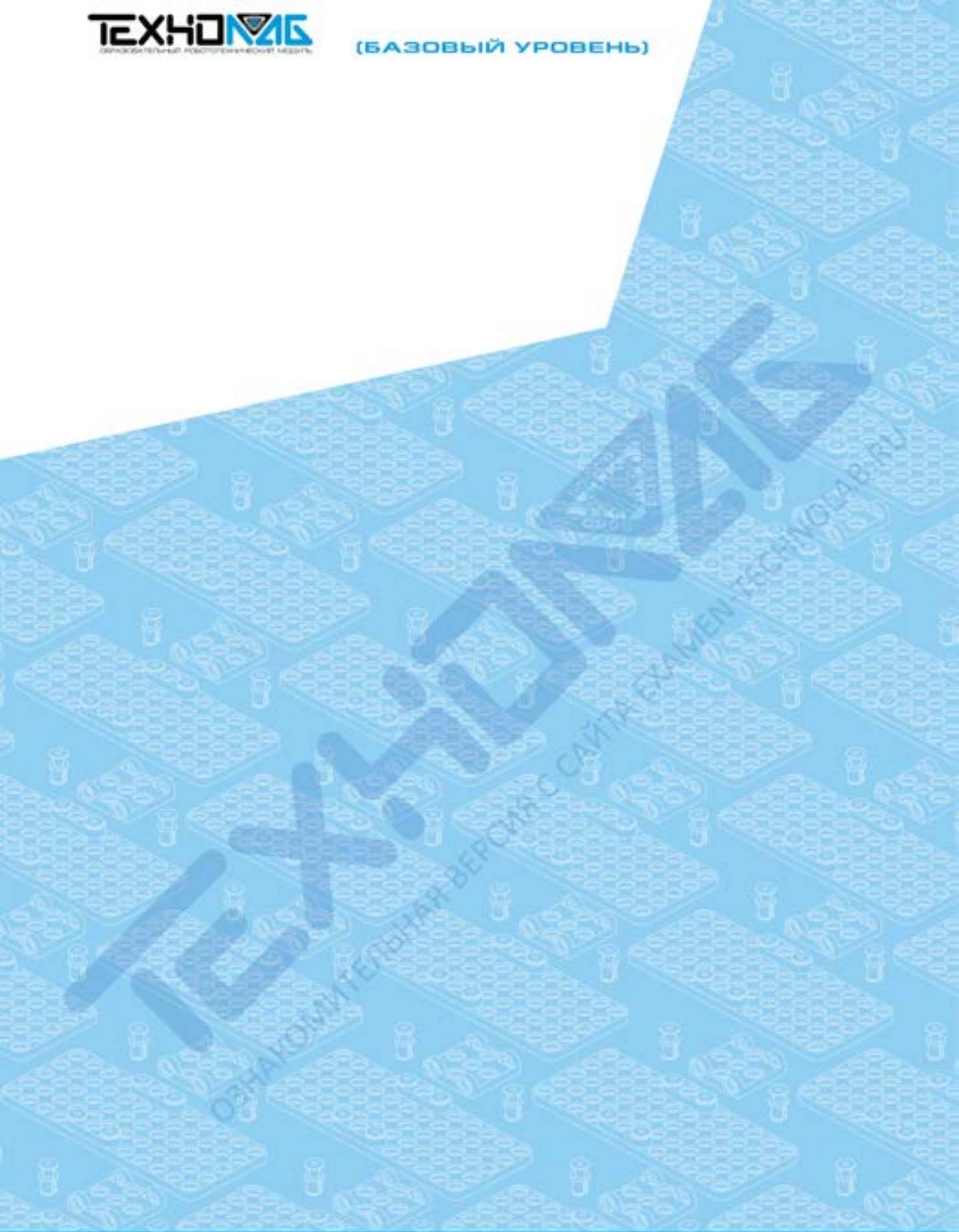
Ознакомительная
версия сайта EXAMEN-TECHNOLAB.RU





Руководство по сборке мобильного робота на базе гусениц

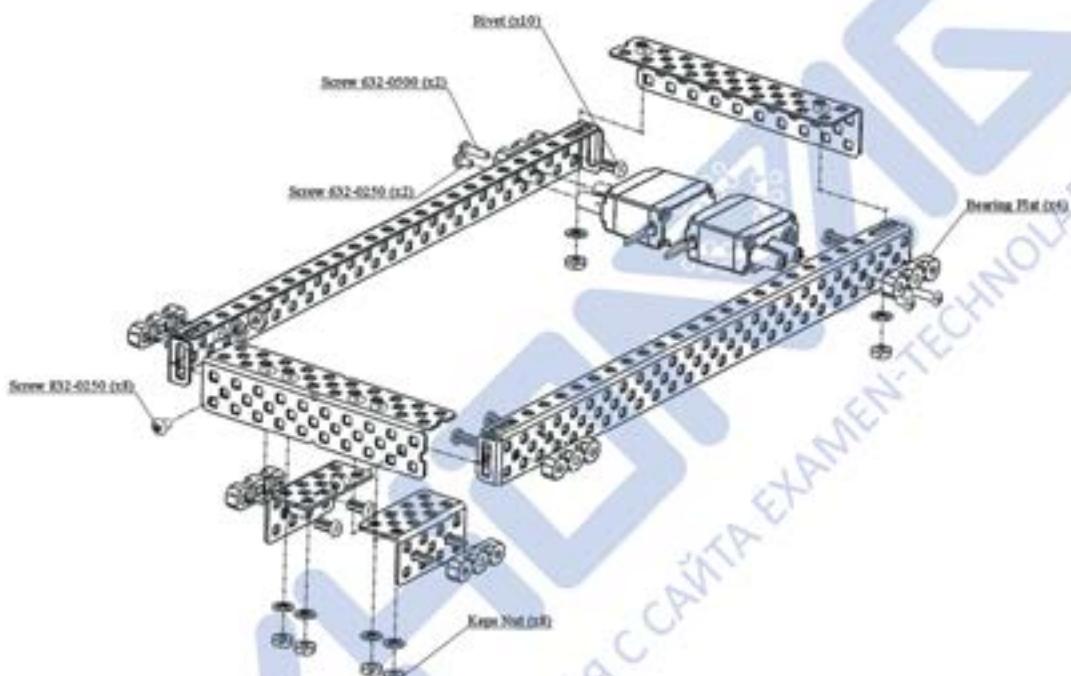
ЭКЗАМЕН
ТЕХНОЛАБ



Приложение 6. Руководство по сборке мобильного робота на базе гусениц

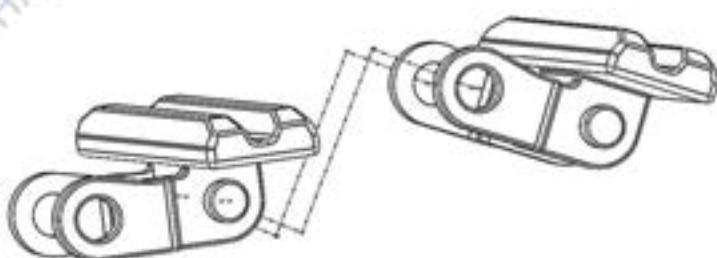
В данном руководстве используются дополнительные детали, не входящие в базовый набор.

Шаг 1

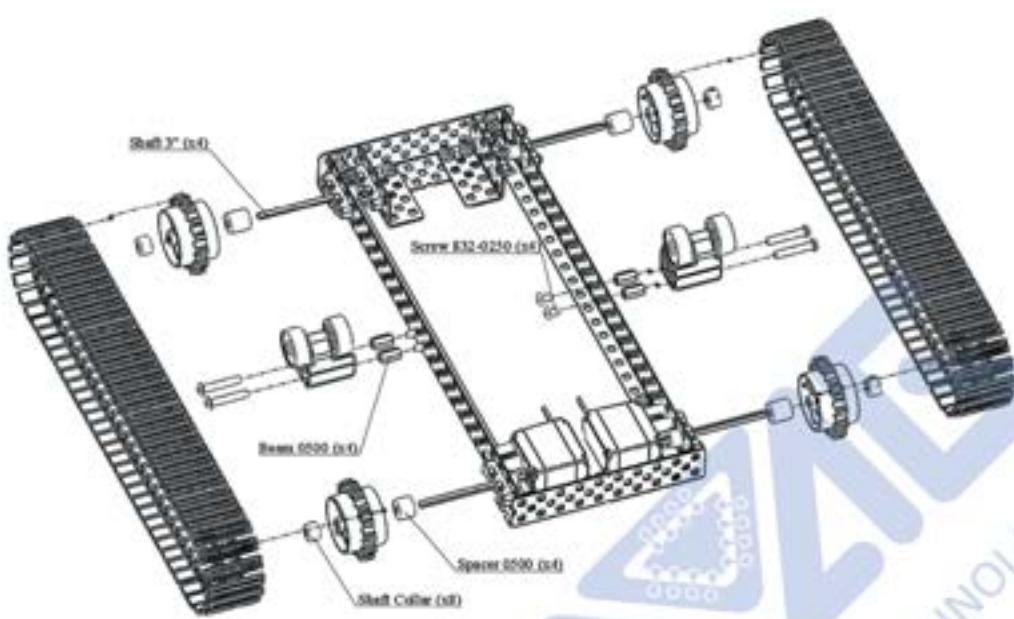


Шаг 2

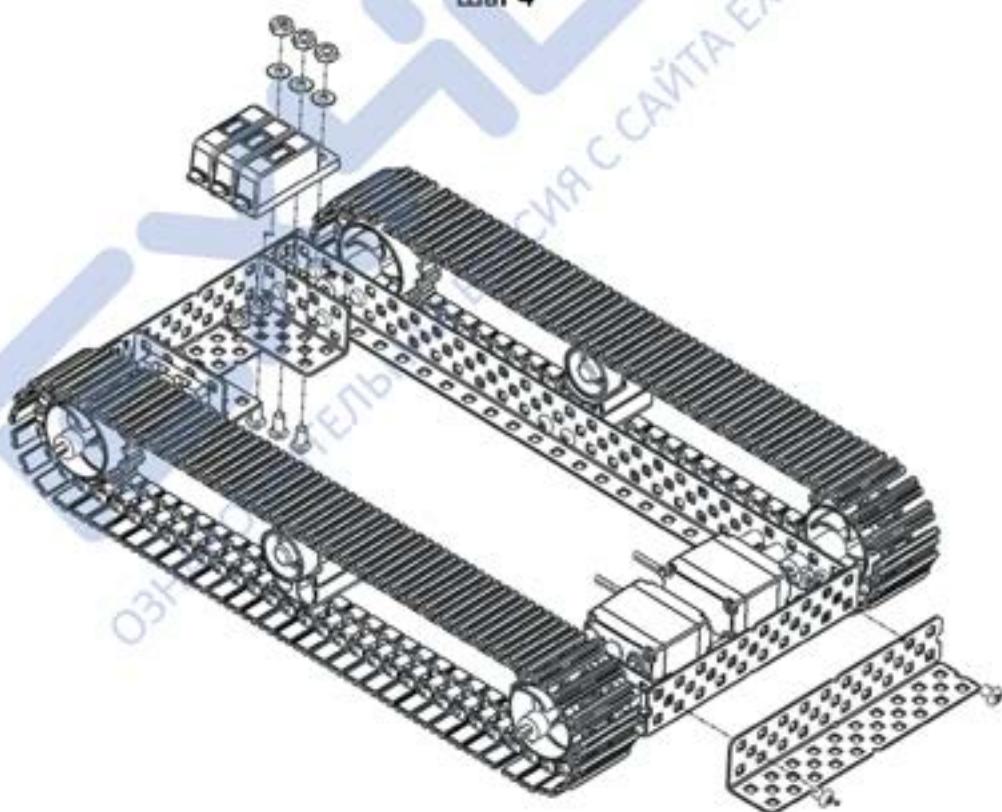
Собрать 2 гусеницы по 71 звену каждого.

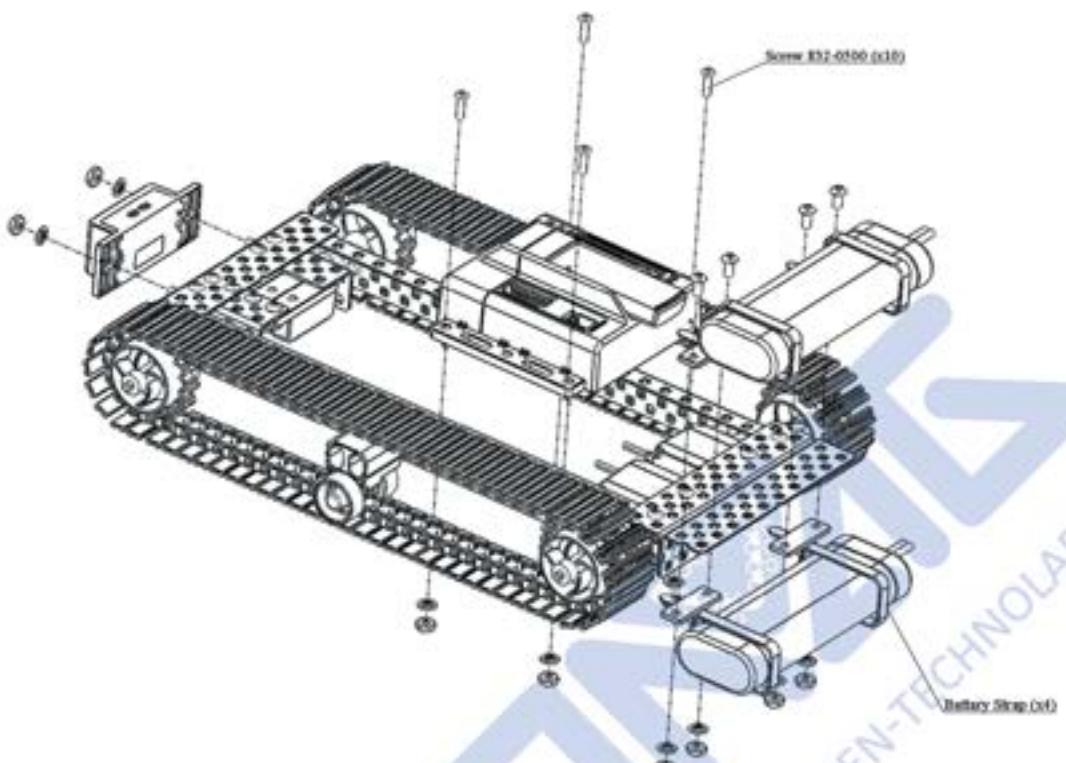
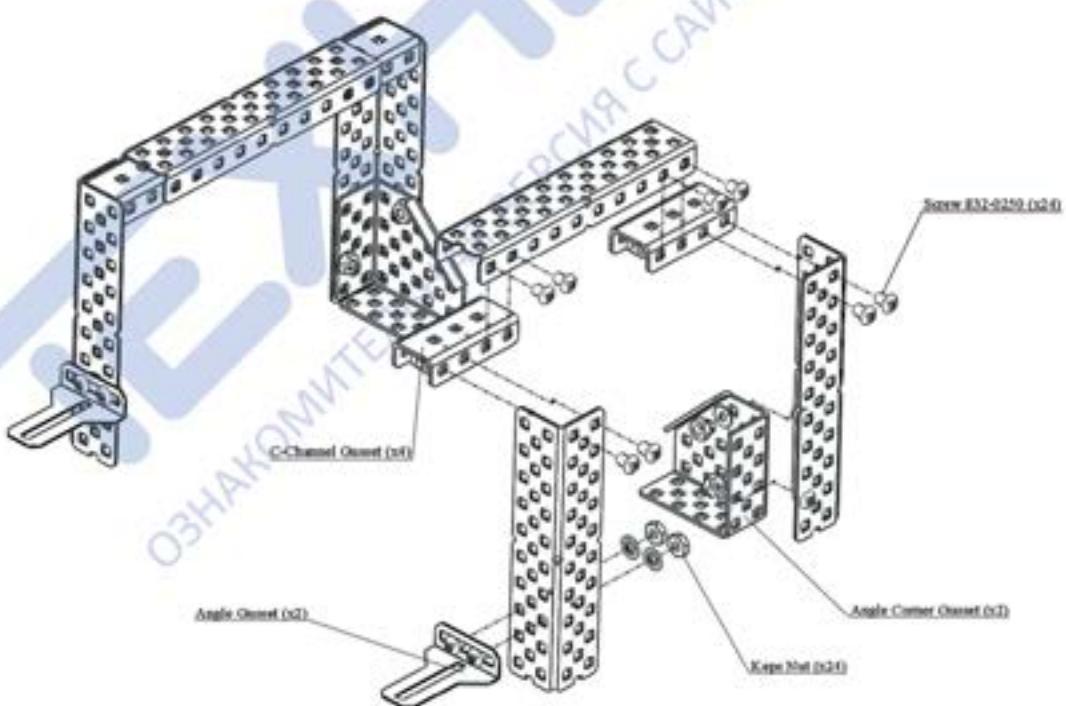


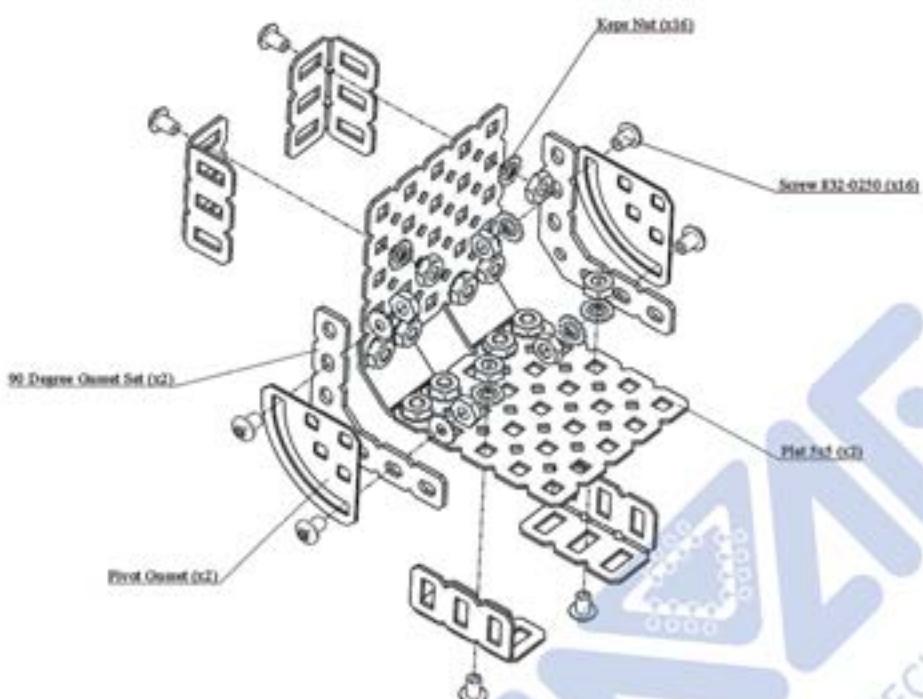
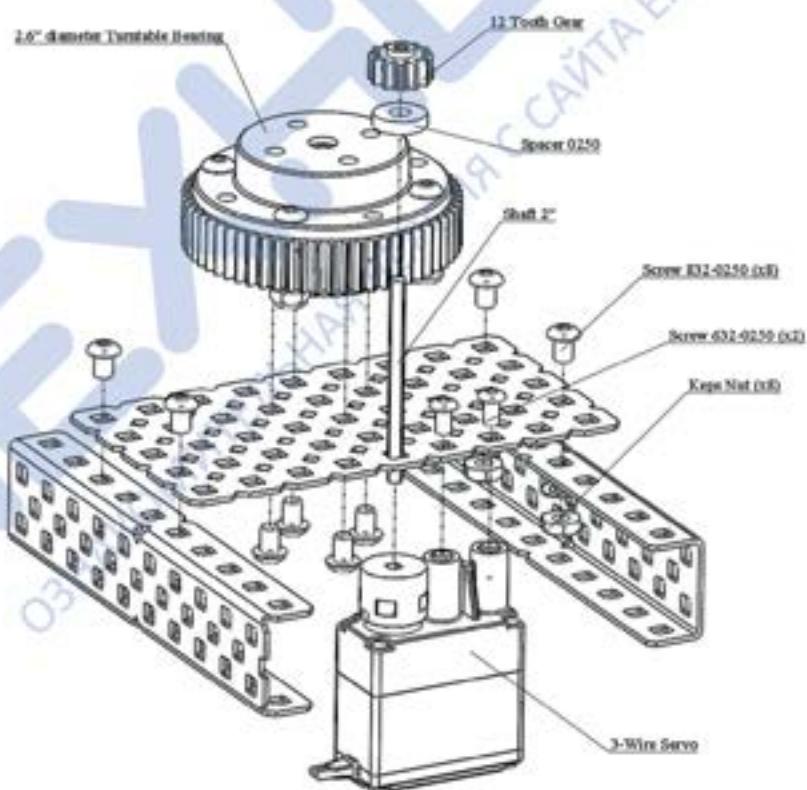
Шаг 3



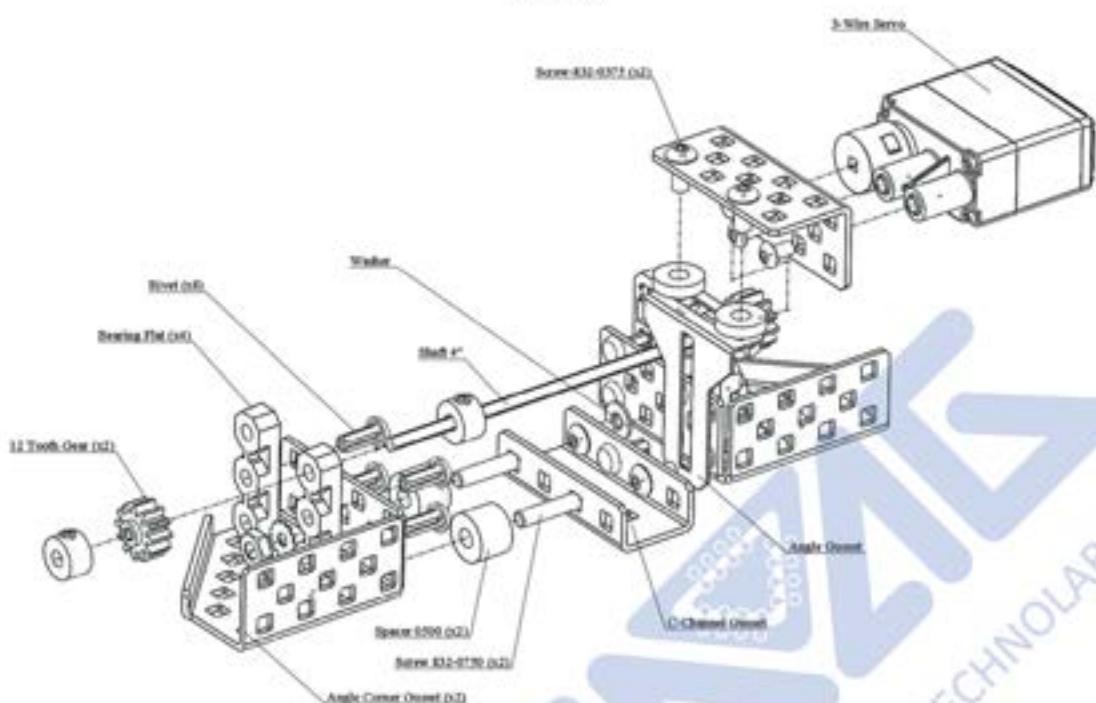
Шаг 4



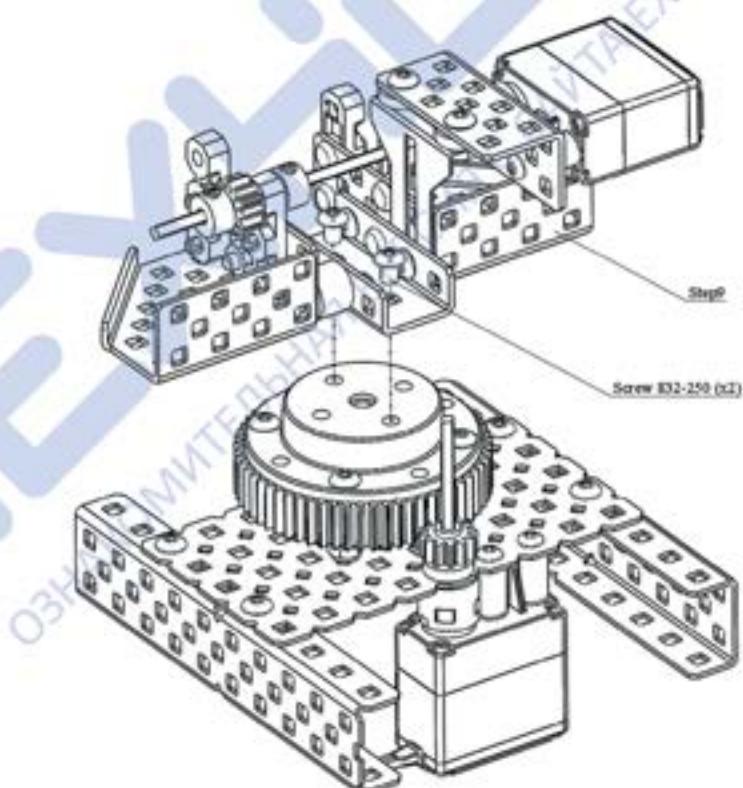
Шаг 5**Шаг 6**

Шаг 7**Шаг 8**

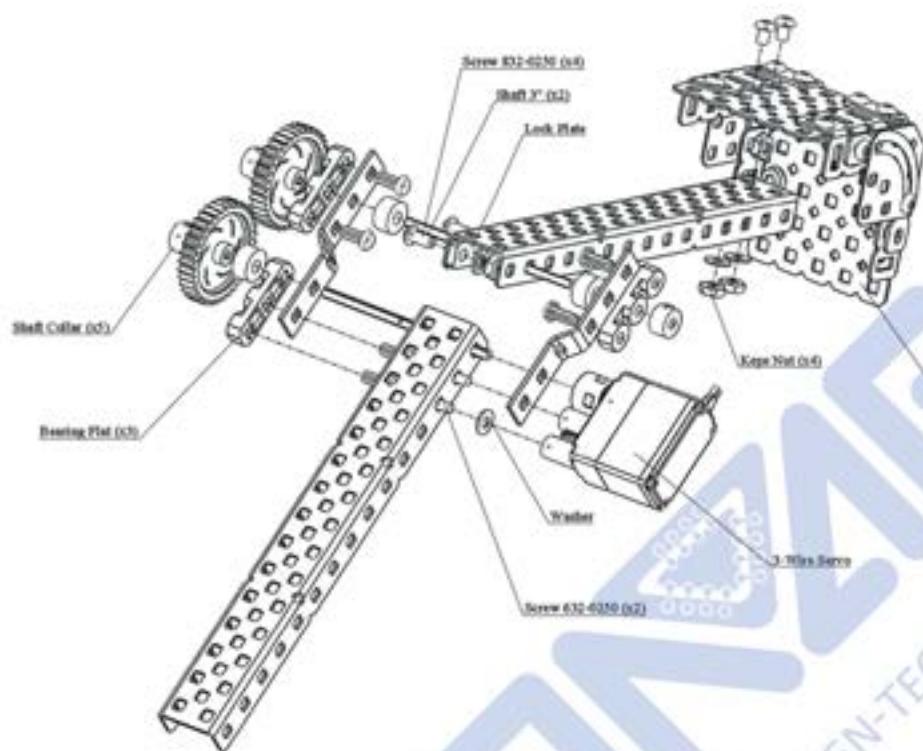
Шаг 9



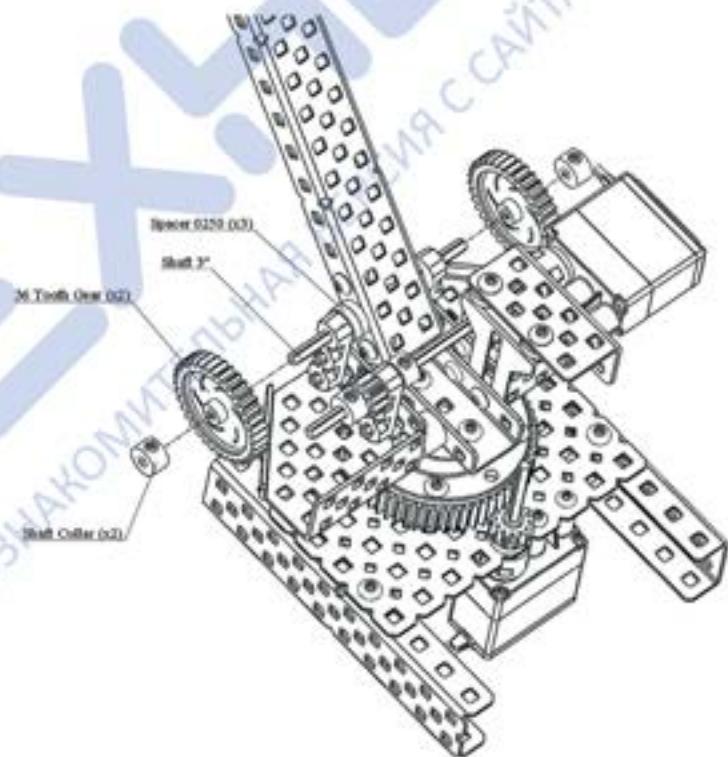
Шаг 10



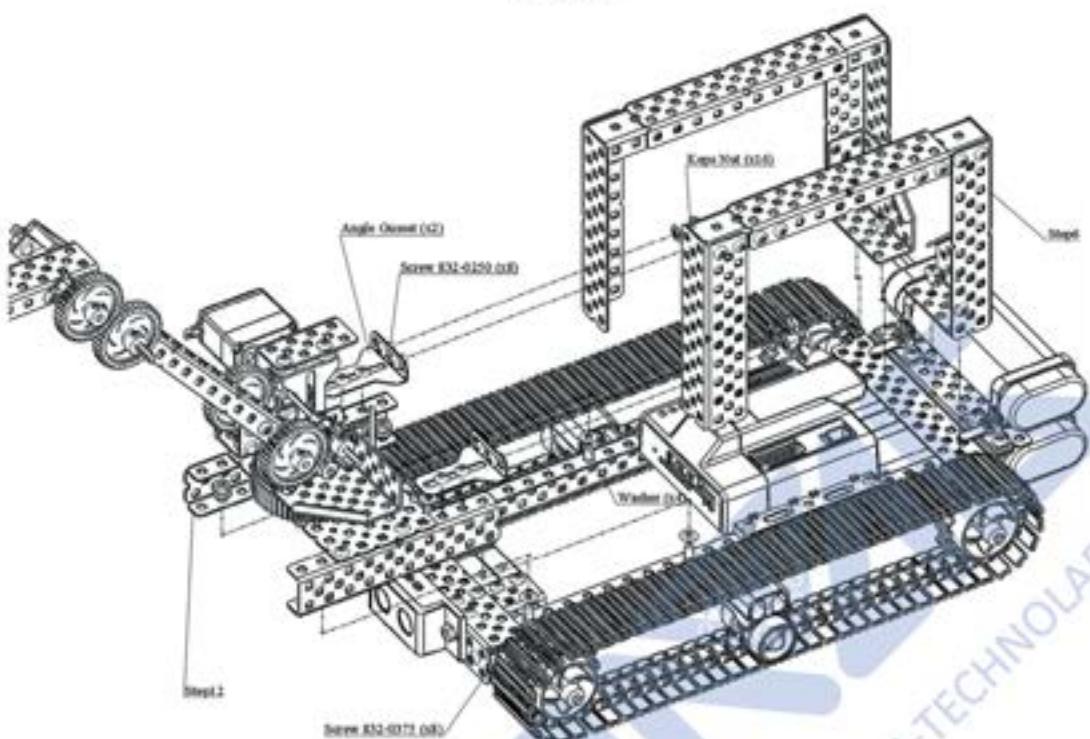
Шаг 11



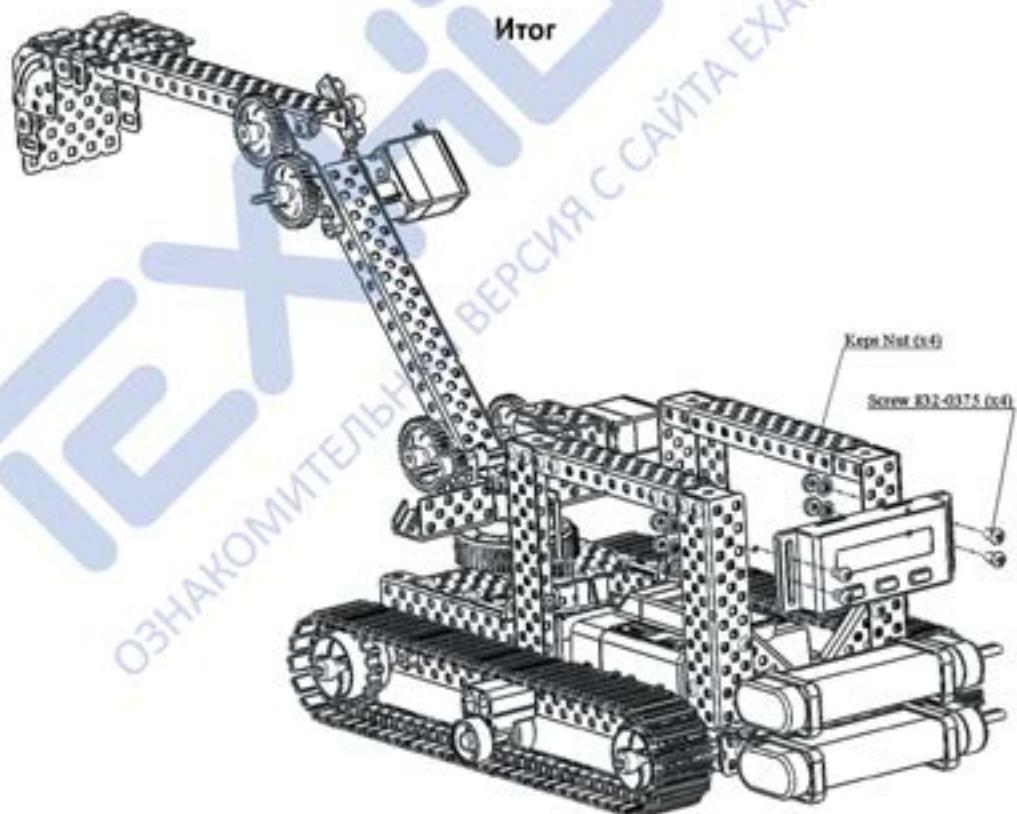
Шаг 12



Шаг 13



Итог



Учебно-методическое издание

Ермишин Константин Владимирович

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

ОБРАЗОВАТЕЛЬНЫЙ
РОБОТОТЕХНИЧЕСКИЙ МОДУЛЬ

(БАЗОВЫЙ УРОВЕНЬ)
12 – 15 ЛЕТ

Издательство «ЭКЗАМЕН»
«ЭКЗАМЕН-ТЕХНОЛАБ»

Гигиенический сертификат

№ РОСС RU. AE51. Н 16678 от 20.05.2015 г.

Главный редактор Л. Д. Лаппо

Корректор Баринская И. Д.

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)



www.examen-technolab.ru

Артикул ТВ-0441-М

ISBN 978-5-377-10297-7

9 785377 102977

12-15
ЛЕТ

