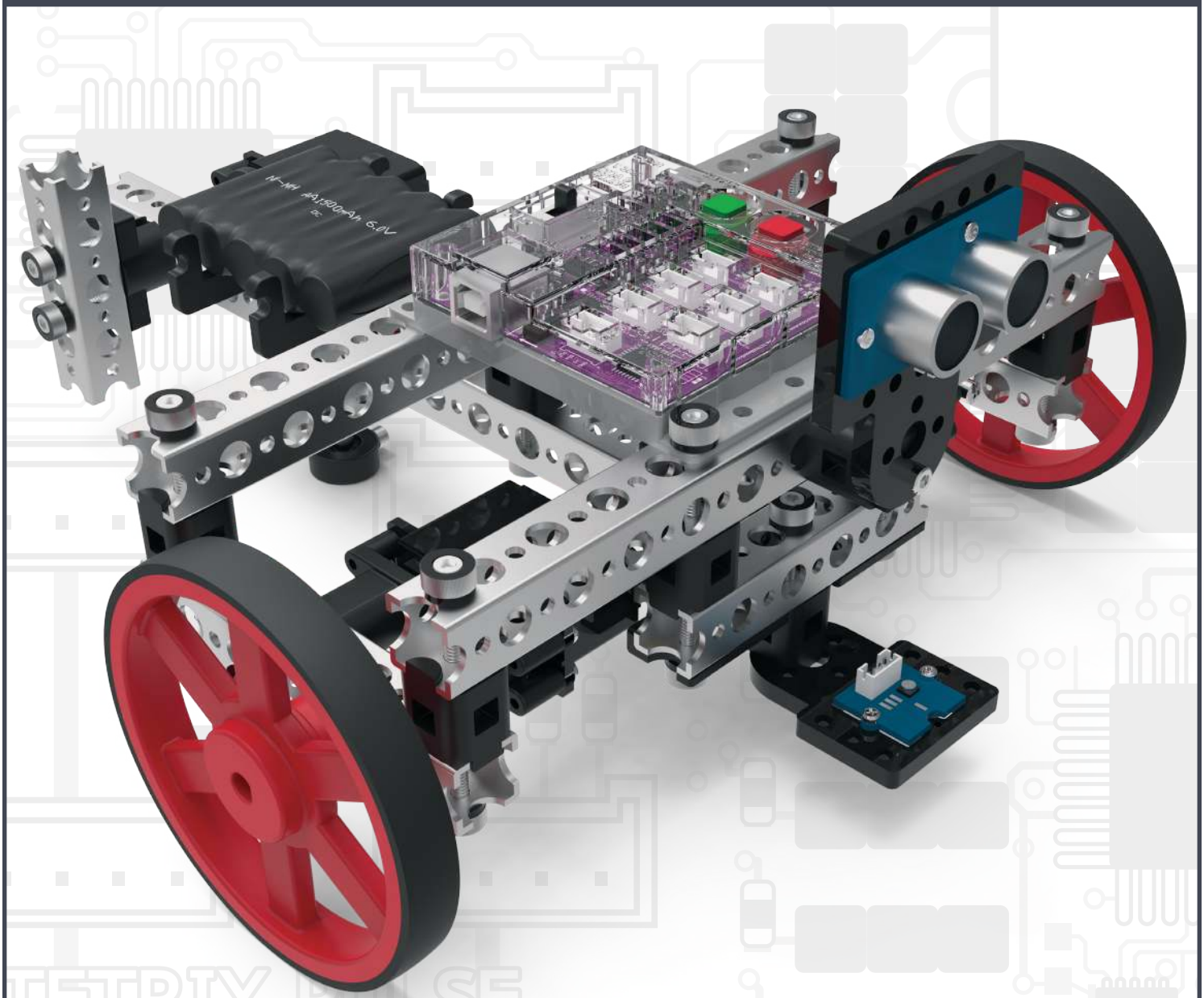
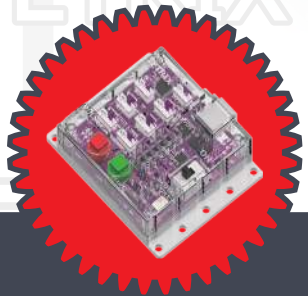
PITSCO

TETRIX® PRIME





Робототехнический контроллер TETRIX® PULSE™ Советы по программированию

PITSCO EDUCATION

44301

Консультанты по содержанию: Тэмми Панки, Пол Аттли, Тим Лэнкфорд

Авторы моделей и рендеров *SolidWorks® Composer™* и *KeyShot®*: Тим Лэнкфорд, Брайан Эккелберри и Джейсон Редд. Компьютерная вёрстка: Тодд Макджордж.

V1.0
8/17

Обновления этого руководства в виде PDF см. на сайте TETRIXrobotics.com.

Содержание

Предисловие

Настоящее руководство по программированию составлено таким образом, чтобы учащиеся получили положительный опыт в робототехническом конструировании и программировании. С помощью руководства они освоят работу с деталями конструктора TETRIX® PRIME при сборке различных роботов, а также получат навыки программирования контроллера PULSE. По окончании курса они должны научиться собирать роботов собственной конструкции из деталей TETRIX PRIME .

Возрастная категория

Включённые в руководство упражнения предназначены для учащихся основной школы. Предлагаемые упражнения, при оказании небольшой помощи со стороны, посильны и для учащихся старших классов начальной школы. Детали конструктора можно использовать для технического творчества в старших классах основной школы.

Как пользоваться руководством

Упражнения в руководстве построены по принципу нарастания трудности. Упражнения следует выполнять в предложенном порядке. Понятия, изучаемые в одном упражнении, могут не повториться в последующих, но для достижения наибольшего успеха учащимся потребуется усвоить эти понятия.

Сведения о безопасности

Механическая часть

- Пальцы, волосы, а также свободные элементы одежды должны находиться на безопасном расстоянии от зубчатых и движущихся деталей.
- Категорически запрещается подбирать робота, пока он находится в движении или пока не остановлены сервоприводы.

Электрическая часть

- Если робот не используется, проследите за тем, чтобы он был обесточен.
- Запрещается эксплуатировать робота во влажной среде.
- Перед любыми изменениями обязательно обесточьте робота.
- Работая с неизолированными проводами, действуйте осмотрительно во избежание короткого замыкания.
- Монтируя провода, будьте внимательны; при необходимости закрепите их во избежание повреждения самого провода или его изоляции.
- Надежно закрепляйте аккумуляторную батарею.

Увлекательное программирование с PULSE™ и конструирование с TETRIX PRIME!

Первое знакомство с контроллером PULSE

Pitsco Education представляет *Руководство по программированию робототехнического контроллера TETRIX® PULSE™*. Руководство включает в себя цикл увлекательных, постепенно усложняющихся упражнений, которые позволят овладеть основами программирования моделей, создаваемых из конструктора TETRIX PRIME, используя контроллер PULSE и графическую среду разработки *TETRIX Ardublockly*.

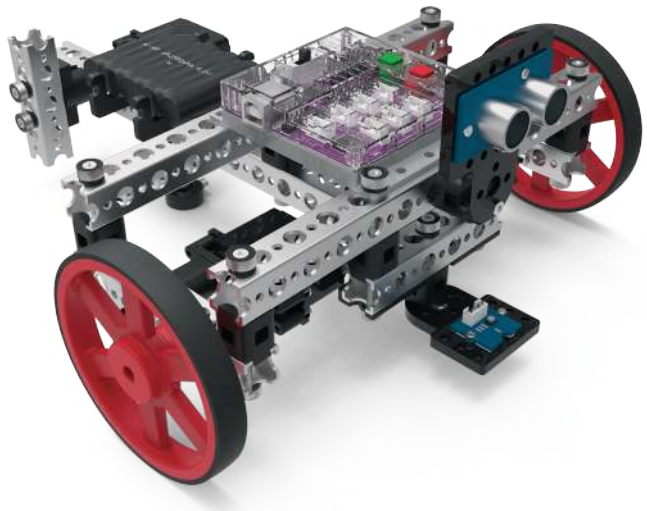
Руководство по программированию является эффективным средством обучения как учащихся, так и учителей работе с контроллером PULSE ("мозгом") и системой TETRIX PRIME при сборке и программировании интеллектуальных точных роботов, максимально приближенных к настоящей жизни. В руководство входят пять вводных упражнений, пошаговые инструкции по сборке базового робота, 10 полных уроков на основе работы с базовым роботом и дополнительные упражнения. Читатели познакомятся с функциями контроллера PULSE, элементами конструктора и программным обеспечением TETRIX. Учащиеся получат отличные базовые знания и навыки, на основе которых они смогут развиваться дальше.

Данное решение, объединяющее в себе контроллер PULSE с интуитивно понятным конструктором PRIME и лёгким в использовании графически ориентированным программным обеспечением, позволит сделать первые шаги в преподавании и изучении робототехники. Благодаря постепенному усложнению упражнений создаваемых роботов можно быстро и легко "оживить", то есть у обучающихся сразу же возникает ощущение успеха, а также остаётся больше времени для решения задач и применения знаний в области естественных и точных наук на занятии.

Контроллер PULSE — не только отличное средство для обучения программированию. Он сделает увлекательными уроки, посвященные датчикам, электричеству, передаточному числу и многим другим темам. Даже прозрачный поликарбонатный корпус контроллера ориентирован на максимальную образовательную эффективность: он позволяет пользователям видеть внутреннюю архитектуру устройства.

Мы также предусмотрели связь с естественными и точными науками (с понятиями, выходящими за рамки настоящего руководства), которую можно реализовать на каждом из уроков по желанию педагога. Реализация связей с естественными и точными науками возможна, если у вас есть соответствующие знания или возможность сотрудничества с другими учителями, которые могут ввести указанные понятия.

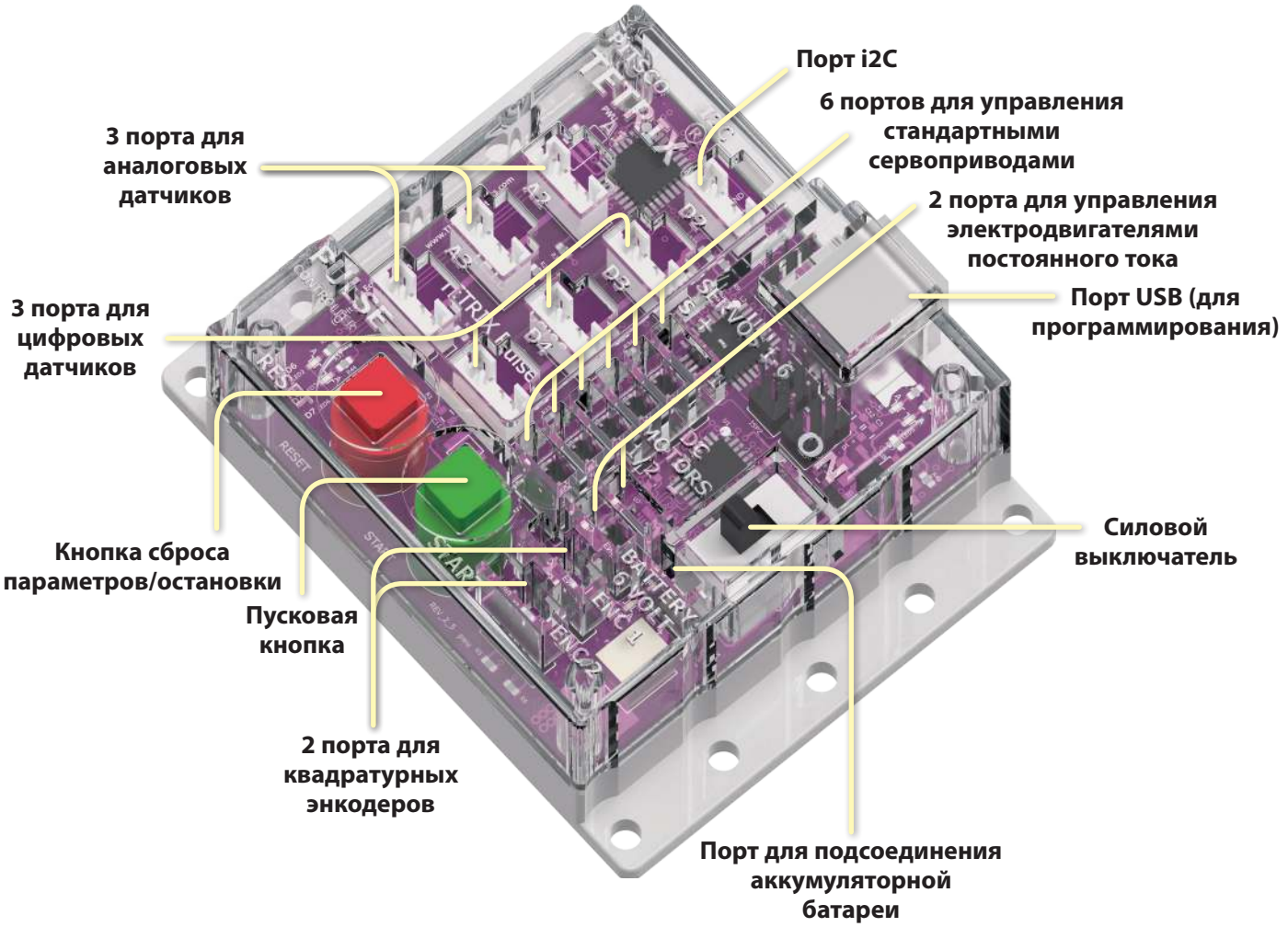
Надеемся, что настоящее руководство послужит прекрасным стартом к обучению с PULSE. С нетерпением ждем инновационных проектов и робототехнических решений, созданных вашими учениками!



Меня зовут Базовый робот! Я выполню все ваши команды.

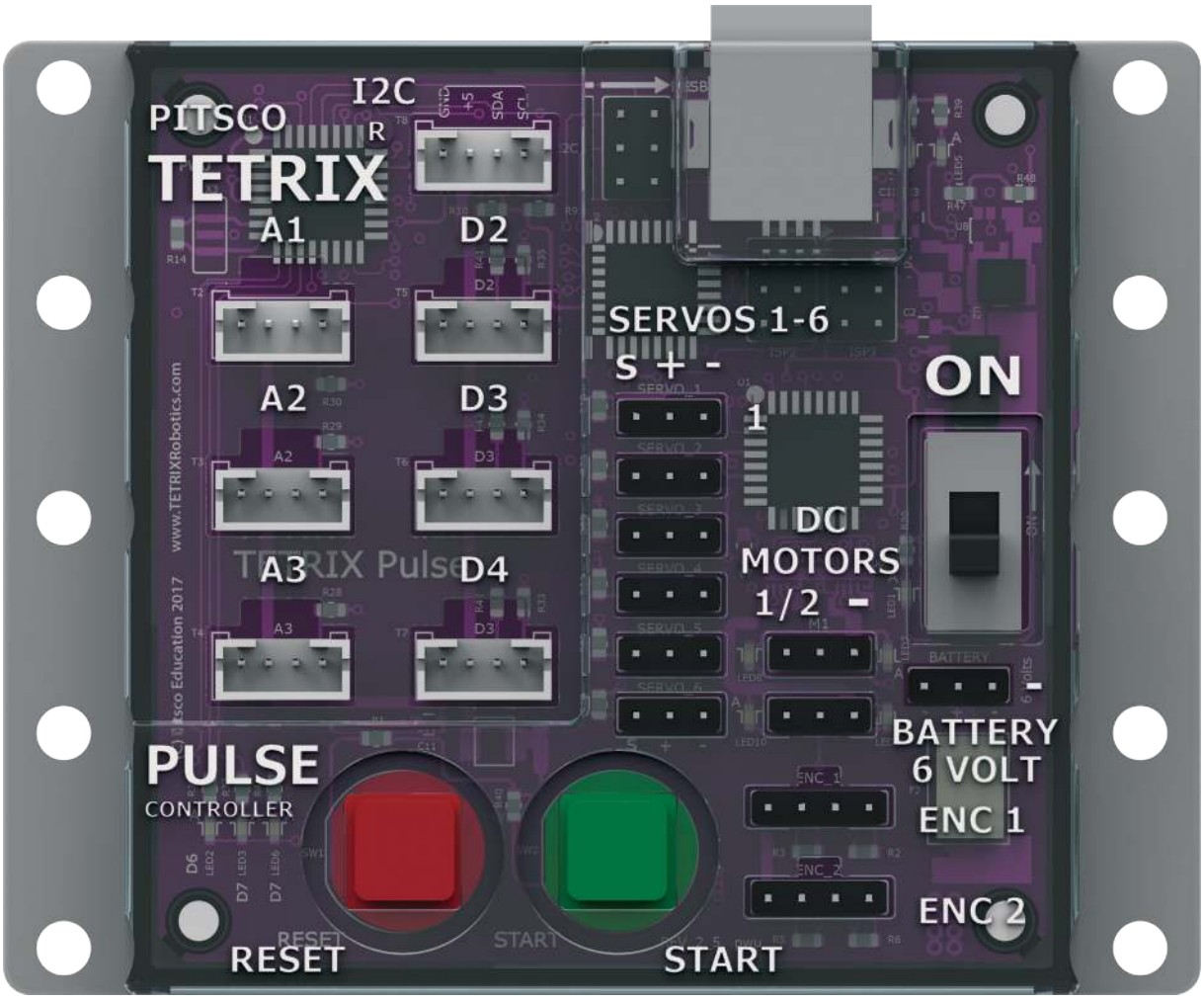
Технический обзор контроллера PULSE

Робототехнический контроллер PULSE:
Программируемое устройство, "командный центр" робота TETRIX PRIME



Совет: Полные и подробные технические характеристики представлены в разделе "Технические характеристики робототехнического контроллера TETRIX PULSE" в приложении на странице 131.

Технический обзор контроллера PULSE



Датчик:

Устройство, считывающее параметры окружающей среды и отправляющее эти данные в контроллер

Ультразвуковой датчик:
Позволяет роботу измерять расстояние до предмета и реагировать на движение

Датчик линии:
Позволяет роботу следовать по чёрной линии на белом фоне или наоборот

Электродвигатель:

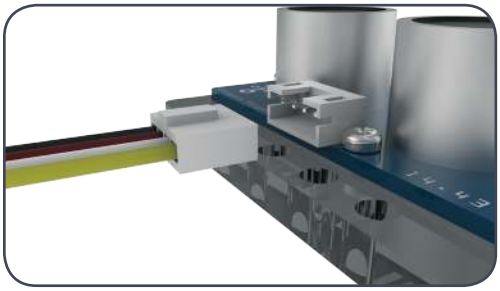
Механизм, который осуществляет движение или производит энергию для выполнения работы

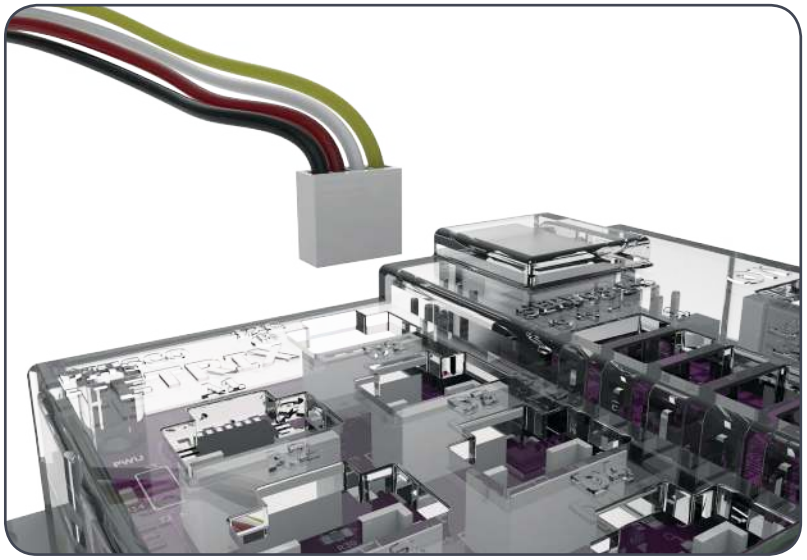
Электродвигатель постоянного тока:
Производит движение или усилие

Стандартный сервопривод:
Позволяет выбрать точное положение в пределах угла перемещения, равного 180 градусам

Подключение оборудования к контроллеру PULSE

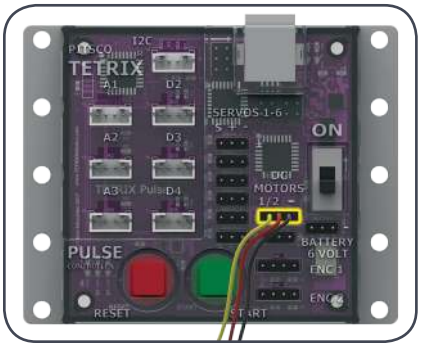
Подключение датчиков:
Для подключения датчика к контроллеру PULSE подключите провода датчика к портам, обозначенным D2-D4 (цифровые датчики), A1-A3 (аналоговые датчики) или I2C (элементы, подключаемые через шину I2C).

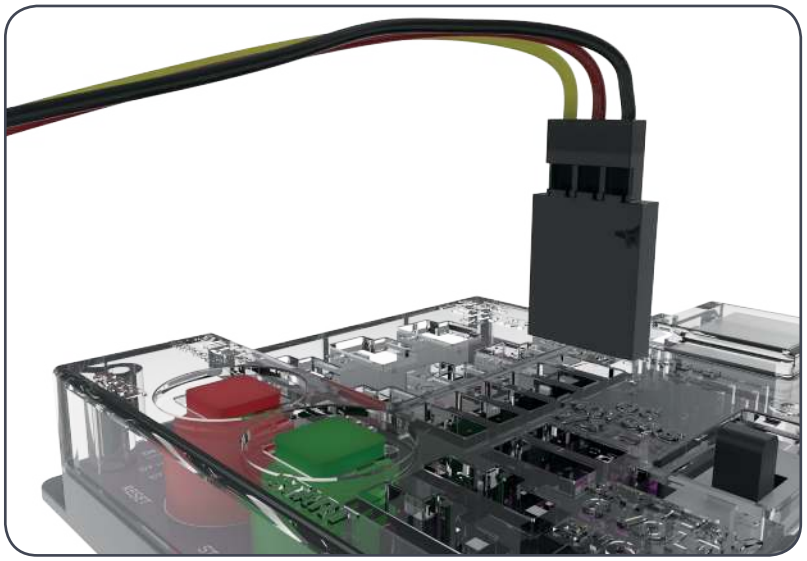




Подключение электродвигателей постоянного тока:
Для подключения электродвигателя постоянного тока к контроллеру PULSE вставьте его провода в один из двух портов для управления электродвигателями постоянного тока.

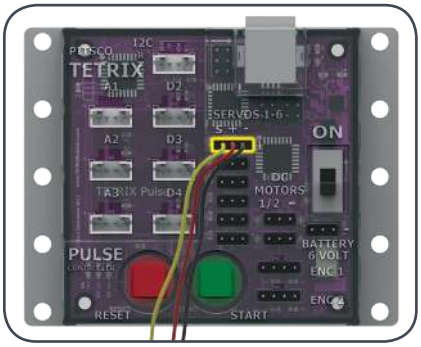
Важно: Конец чёрного провода должен находиться рядом со знаком минуса на корпусе контроллера.

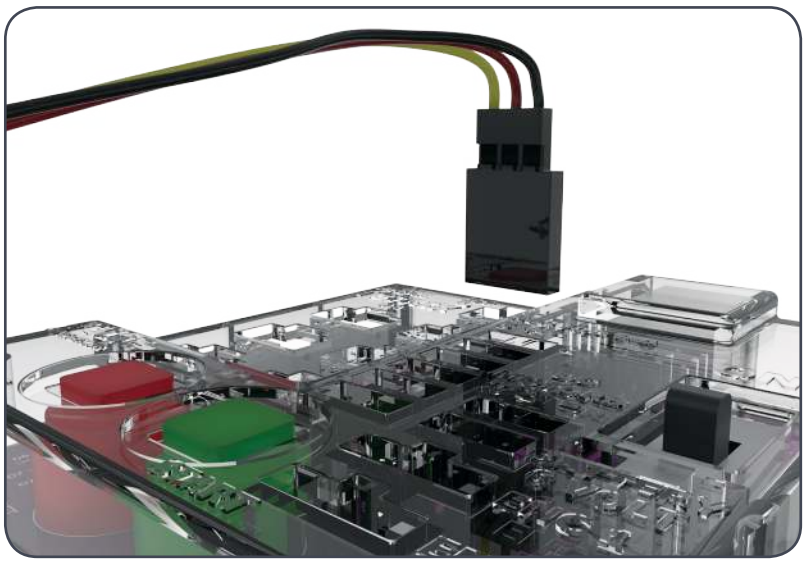




Подсоединение сервоприводов:
Для подсоединения стандартного сервопривода к контроллеру PULSE вставьте его провода в один из портов для управления сервоприводами.

Важно: Конец чёрного провода должен находиться рядом со знаком минуса на корпусе контроллера.

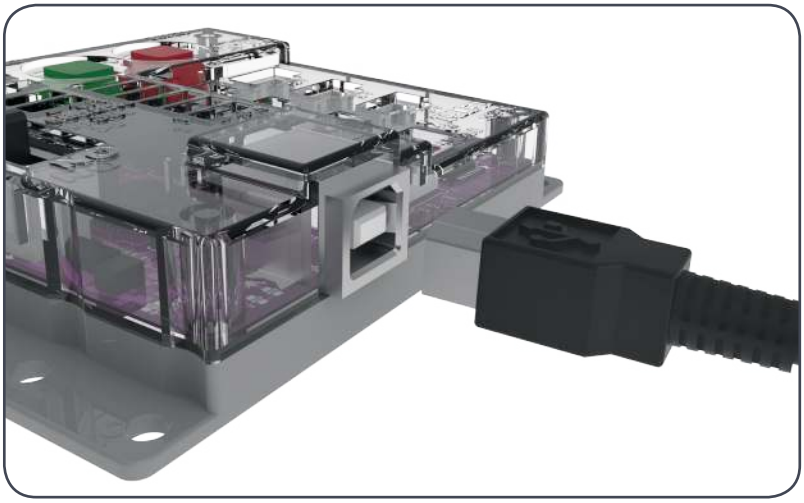




Двунаправленная загрузка данных: Порт USB контроллера PULSE используется для обмена данными между контроллером PULSE и устройством Windows или Macintosh.

Через этот порт пользователи могут загружать данные из компьютера в контроллер PULSE и наоборот.

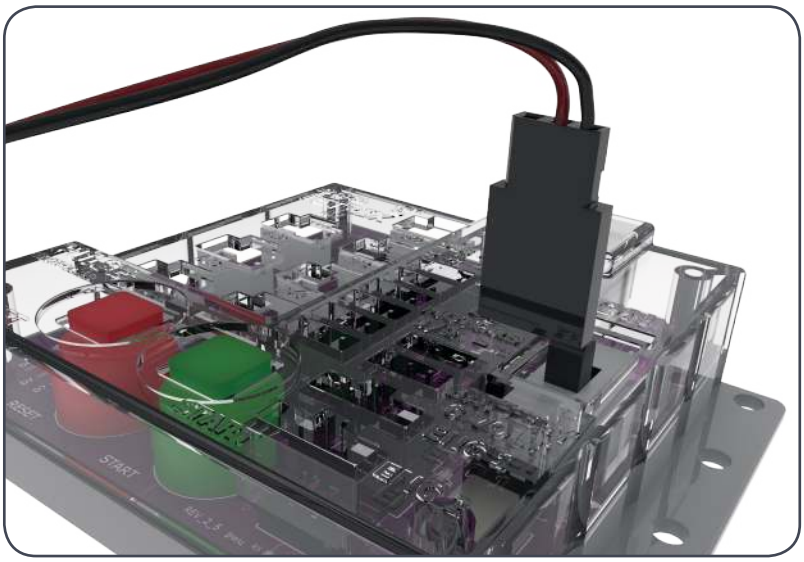
Для загрузки программы в контроллер PULSE вставьте один разъём кабеля USB в порт USB контроллера, а второй разъём — в порт USB вашего устройства.

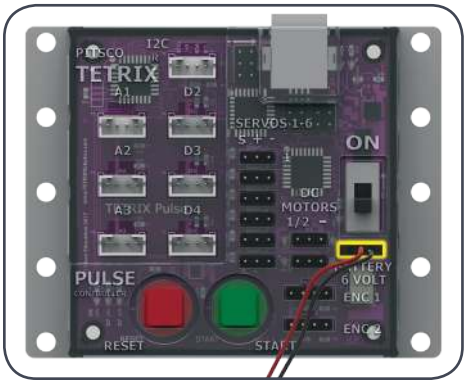


Подключение аккумуляторной батареи к контроллеру PULSE: Питание контроллера PULSE осуществляется от перезаряжаемой никель-металлгидридной аккумуляторной батареи TETRIX напряжением 6 вольт.

Для подключения аккумуляторной батареи к контроллеру PULSE вставьте разъём её провода в гнездо для подключения аккумулятора на корпусе контроллера.

Важно: Конец чёрного провода должен находиться рядом со знаком минуса на корпусе контроллера.





Предупреждение: Запрещается подключать к контроллеру PULSE аккумуляторные батареи сторонних производителей. Аккумуляторные батареи TETRIX защищены плавким предохранителем и являются единственными батареями аккумуляторов, пригодными для данной системы. При повреждении продукта вследствие нарушения данного требования гарантия аннулируется.

Обзор программного обеспечения

- Программное обеспечение *TETRIX Ardublockly* — это специальный интерфейс программирования, созданный компанией Pitsco исключительно для использования с контроллером PULSE. Программное обеспечение *TETRIX Ardublockly* было разработано на основе интерфейса Google под названием *Blockly*.
- Его можно устанавливать на различных устройствах Windows и Macintosh. Программное обеспечение устанавливается на жесткий диск устройства. Используемое устройство должно иметь порт USB для подключения к контроллеру PULSE. Данное программное обеспечение не поддерживается планшетами и ноутбуками Chromebook.
- Для связи с контроллером PULSE в фоновом режиме предусмотрено программное обеспечение *Arduino (IDE)*. Для написания программ ПО *Arduino (IDE)* открывать не требуется. Оно просто должно быть установлено на компьютере. Контроллер PULSE можно программировать с помощью ПО *Arduino (IDE)*, однако в рамках этого руководства все программирование будет выполняться с помощью ПО *TETRIX Ardublockly*.
- В системе ПО *TETRIX Ardublockly* программа называется скетчем. Каждое упражнение руководства предусматривает создание скетча, который будет давать команды роботу.
- В руководстве рассматриваются основы применения ПО *TETRIX Ardublockly* для работы с контроллером PULSE. Лёгкость применения ПО *TETRIX Ardublockly* с контроллером PULSE станет очевидна в процессе работы с примерами и практического применения программ на основе базового робота, собранного из робототехнического конструктора TETRIX PRIME.

Примечание:
Настоящее руководство не предназначено для обучения программированию на языке Arduino, разновидности языка C. В сети существует множество отличных ресурсов, позволяющих усовершенствовать навыки программирования. Если такие ресурсы представляют для вас интерес, можно начать с сайта Arduino по адресу: **www.arduino.cc**.

Установка и настройка программного обеспечения

Arduino (IDE)

Для начала работы необходимо установить ПО *Arduino (IDE)*. Данное программное обеспечение для операционных систем Windows и Macintosh представлено на сайте Arduino (**www.arduino.cc**). На главной странице Arduino нажмите на вкладку Software. На странице Software выберите загрузку для вашей операционной системы и выполните все последующие инструкции.

Примечание: Все инструкции и снимки экранов, представленные в настоящем руководстве, относятся к версии 1.8.2 ПО *Arduino (IDE)*. Инструкции и вид экранов могут незначительно отличаться в зависимости от используемой платформы и версии.

Библиотека контроллера PULSE

Добавляя пользовательские библиотеки, можно расширить применение ПО *Arduino (IDE)*. Библиотеки — это наборы базовых функций, облегчающие написание программ или, в терминологии Arduino, скетчей. Добавление библиотеки Arduino для контроллера PULSE осуществляется после установки ПО *Arduino (IDE)*. Библиотека включает в себя специальные программы, написанные для контроллера TETRIX PULSE и

распространяется в формате .zip: TETRIX_PULSE.zip. Сначала необходимо загрузить библиотеку PULSE с сайта TETRIX по адресу: **www.TETRIXrobotics.com/PULSEdownloads**.

После загрузки библиотеки её можно включить в ПО *Arduino Software (IDE)* двумя способами.

Примечания для учителя:

- С учётом условий в классе и наличия электронного оборудования можно загрузить и установить ПО *Arduino (IDE)* и библиотеку Arduino для TETRIX PULSE на компьютеры, за которыми вы будете работать с учениками.
- Рекомендуется разбить класс на пары. Также педагогу желательно выполнить все действия самостоятельно, прежде чем предлагать упражнения классу. Это позволит предугадать возможные вопросы учеников и подготовить ответы.

Импорт библиотеки в формате .zip

Первый вариант — применение пункта меню Add .ZIP Library программы *Arduino Software (IDE)*.

В программе *Arduino Software (IDE)* перейдите в пункты меню**Sketch > Include Library**. В выпадающем меню выберите **Add .ZIP Library** (рис. 1).

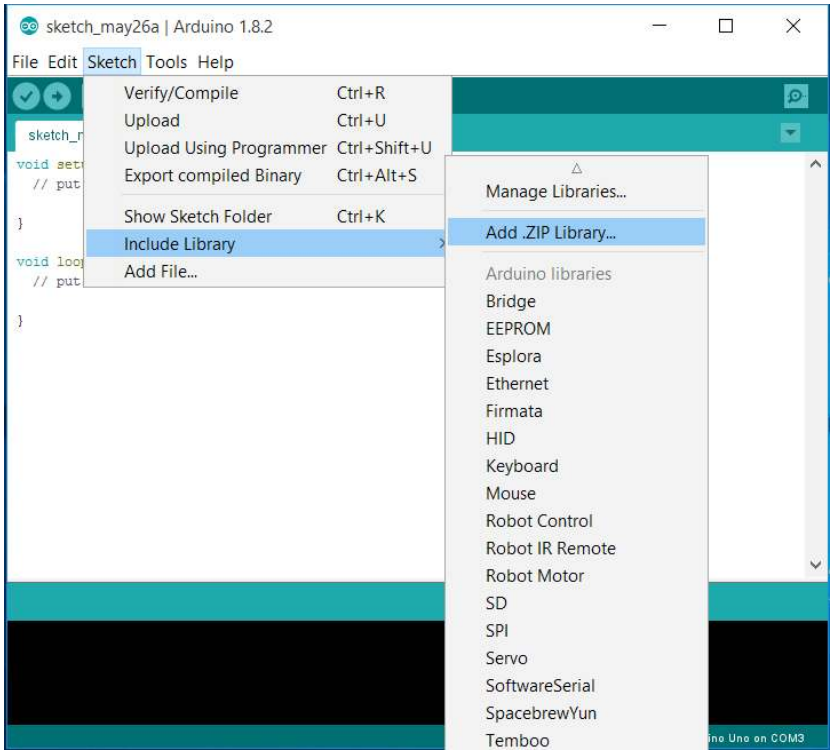


Рисунок 1

Совет: Иллюстрации в настоящем разделе показывают стандартный процесс установки в ОС Windows. Вид экранов и расположение файлов в ОС Mac могут отличаться.

Появится окно выбора библиотеки для добавления. Перейдите по адресу, где вы сохранили файл библиотеки TETRIX_PULSE.zip, выберите и откройте его (рис. 2).

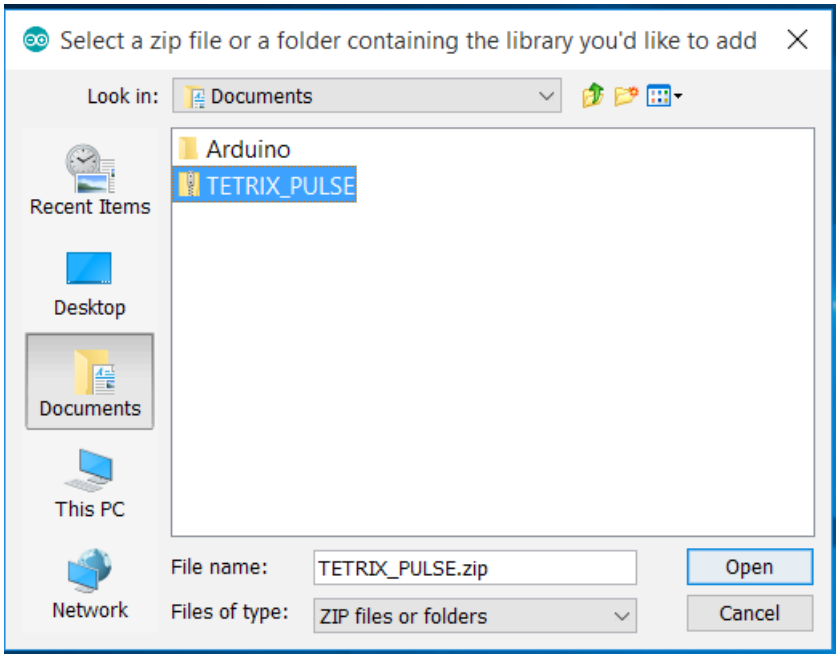


Рисунок 2

Вернитесь в меню **Sketch > Include Library**. Теперь в нижней части выпадающего меню должна отображаться библиотека. Она готова к использованию в скетчах. Однако примеры скетчей для библиотеки появятся в меню **File > Examples** только после перезапуска ПО *Arduino (IDE)*.

Установка вручную

Чтобы установить библиотеку PULSE вручную сначала закройте приложение ПО *Arduino (IDE)*. Затем извлеките папку с библиотекой из архива TETRIX_PULSE.zip. После извлечения перетащите или копируйте папку TETRIX_PULSE в папку с библиотеками Arduino.

В ОС Windows её вероятное расположение:**Documents\Arduino\libraries**.

В ОС Mac её вероятное расположение: **Documents\Arduino\libraries**.

Перезапустите ПО *Arduino (IDE)*. Проверьте, появилась ли библиотека TETRIX_PULSE в меню программы **Sketch > Include Library**.

Также в выпадающем меню **File > Examples > TETRIX_PULSE** теперь появятся несколько примеров скетчей для контроллера PULSE.

Библиотека Arduino для контроллера PULSE успешно установлена!

Настройка связи через USB

Связь между контроллером PULSE и ПО *Arduino (IDE)* осуществляется через порт USB компьютера.

Таким образом, прежде чем приступать к программированию, необходимо убедиться, что контроллер PULSE имеет надлежащие настройки в ПО*Arduino (IDE)* для связи через порт USB.

Самый лёгкий способ проверки: запустить ПО *Arduino (IDE)*, перейти в меню **Tools > Board** и выбрать**Arduino/Genuino Uno** (рис. 3). В контроллере PULSE используется та же процессорная микросхема, что и на подлинной плате Arduino UNO, соответственно, нужно выбрать именно эту плату.

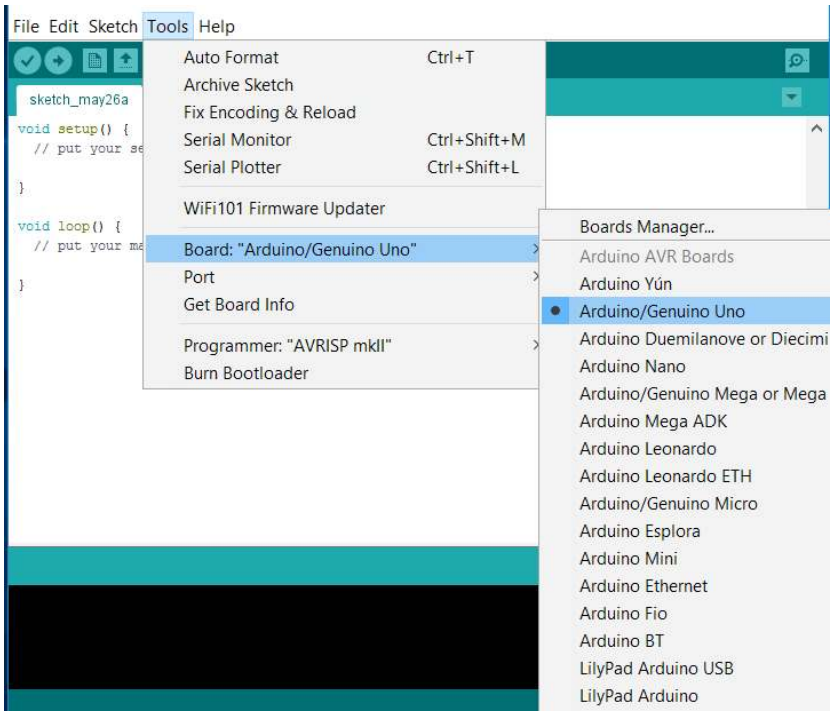
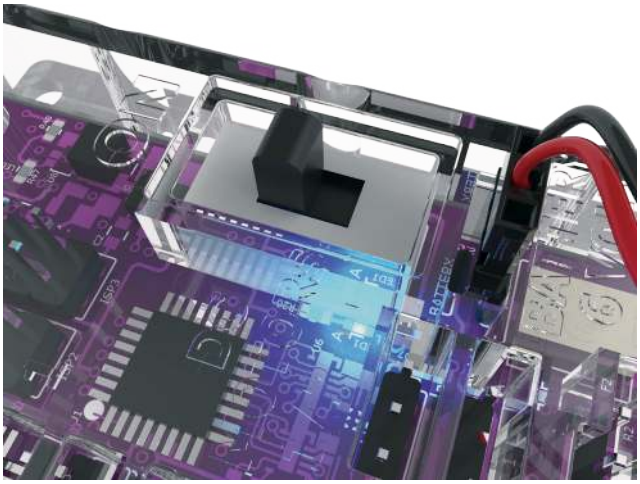


Рисунок 3

Далее, **отсоединив контроллер PULSE**, перейдите в **Tools > Port** и проверьте текущие подключения. При отсутствии текущих подключений пункт меню *Port* будет неактивным. При наличии подключений обратите внимание на перечень COM-портов.

Затем подсоедините контроллер PULSE к порту USB и подайте на контроллер питание: соедините его с аккумуляторной батареей TETRIX и переведите выключатель питания в положение "включено".

При включении питания загорится синий светодиодный индикатор. Дайте контроллеру PULSE достаточно времени для выполнения автоматической установки при первом подключении. Это займет 5-10 секунд. По завершении подключения и установки контроллера PULSE система компьютера присвоит ему COM-порт.



Перейдите в **Tools > Port** и выберите недавно установленный COM-порт. Новый COM-порт и будет контроллером PULSE. Выбрав новый COM-порт, вы подтверждаете, что ПО *Arduino (IDE)* должно использовать его для связи с контроллером. Используемый вами COM-порт может отличаться от показанного на рис. 4.

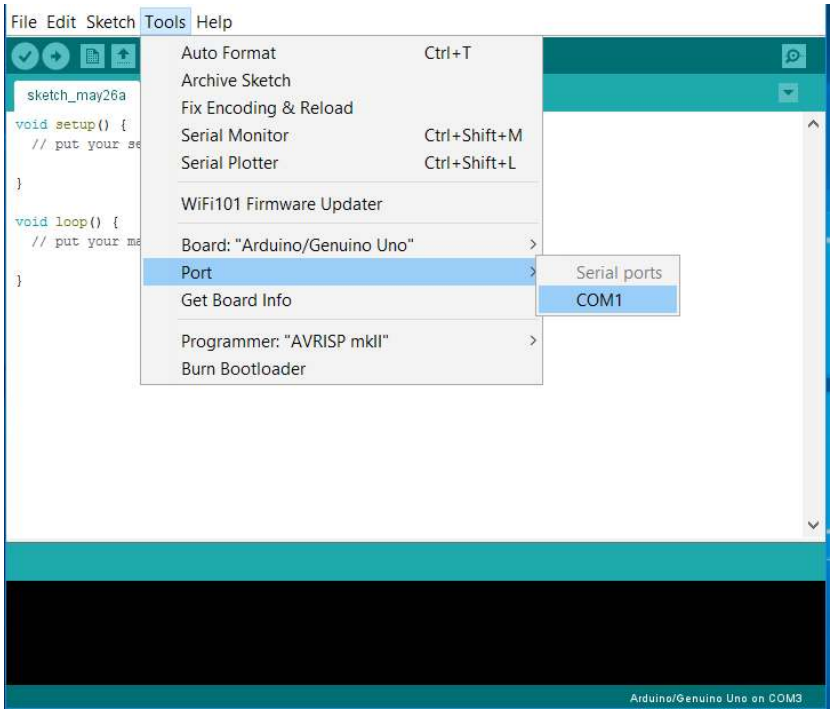


Рисунок 4: выпадающее меню порта, если контроллер PULSE отсоединён. Перечни могут различаться.

Новый порт в данном примере обозначен как COM1. Выберите этот новый порт, чтобы ПО *Arduino (IDE)* использовало его для связи с контроллером. Скорее всего, при реальной установке у вас будет другой порт, и это нормально. По завершении настройки последовательного порта связь с контроллером PULSE установлена.

После выполнения описанного шага система компьютера будет автоматически использовать выбранный порт при каждом подключении контроллера PULSE и запуске ПО *Arduino (IDE)*.

Совет: Каждое устройство PULSE будет использовать отдельный COM-порт на общем компьютере. При подключении каждого нового контроллера PULSE необходимо выполнить шаги по присвоению номеров контроллерам и регистрации в системе компьютера, как описано выше. Вы можете присвоить номер каждому из контроллеров PULSE и закрепить его за соответствующим компьютером. Это позволит компьютеру выбирать верный порт для контроллера PULSE при каждом подключении контроллера и включении питания.

Примечание: Другие устройства, подсоединённые к компьютеру, например, сотовые телефоны, могут также отображаться в виде COM-порта. При одновременном подсоединении другого устройства может возникнуть необходимость возврата в программу *Arduino (IDE)*, чтобы проконтролировать выбор правильного COM-порта.

Установка и настройка программного обеспечения *TETRIX Ardublockly*

Программное обеспечение *TETRIX Ardublockly* распространяется в формате .zip: TETRIX_Ardublockly.zip. Существует версия для ОС Windows и версия для ОС Mac. Загрузите версию, подходящую для вашего устройства, с сайта TETRIX. Библиотека находится по адресу: **www.TETRIXrobotics.com/PULSEdownloads**.

Примечание: Данное программное обеспечение не работает на компьютерах с 32-разрядной версией Windows.

Сохраните загруженный файл на жестком диске. Выберите в меню извлечение заархивированной папки (рис. 5).

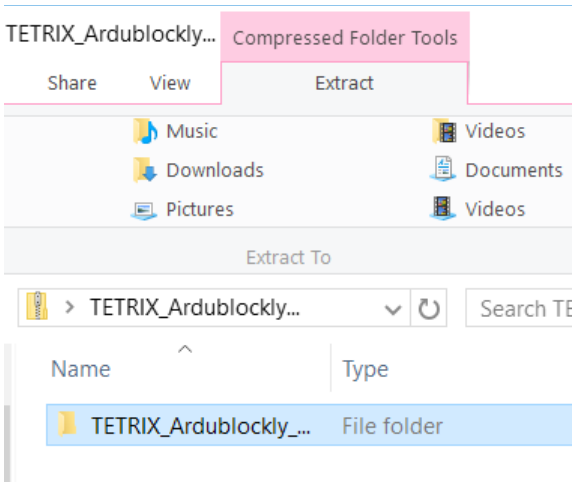


Рисунок 5

Найдите файл ardublockly_run (рис. 6).

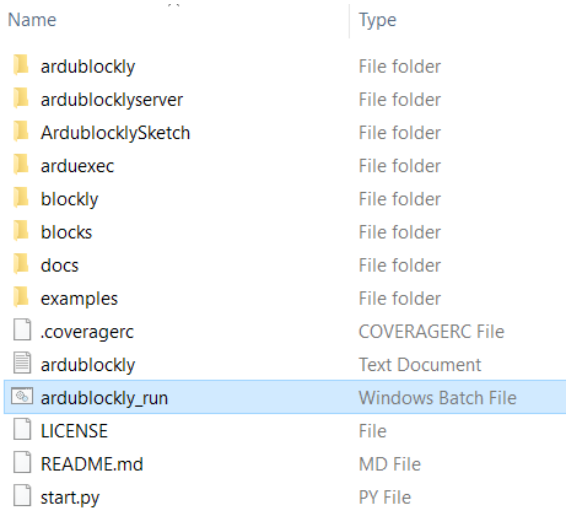


Рисунок 6

Создайте ярлык и разместите его на рабочем столе (рис. 7). Не удаляйте извлеченную папку, она необходима для работы программного обеспечения.

Рисунок 7

В процессе загрузки программного обеспечения на рабочем столе появится значок *TETRIX Ardublockly* (рис. 8).

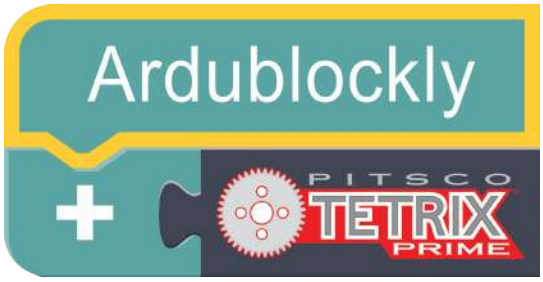


Рисунок 8

После завершения загрузки должен появиться этот экран (рис. 9).

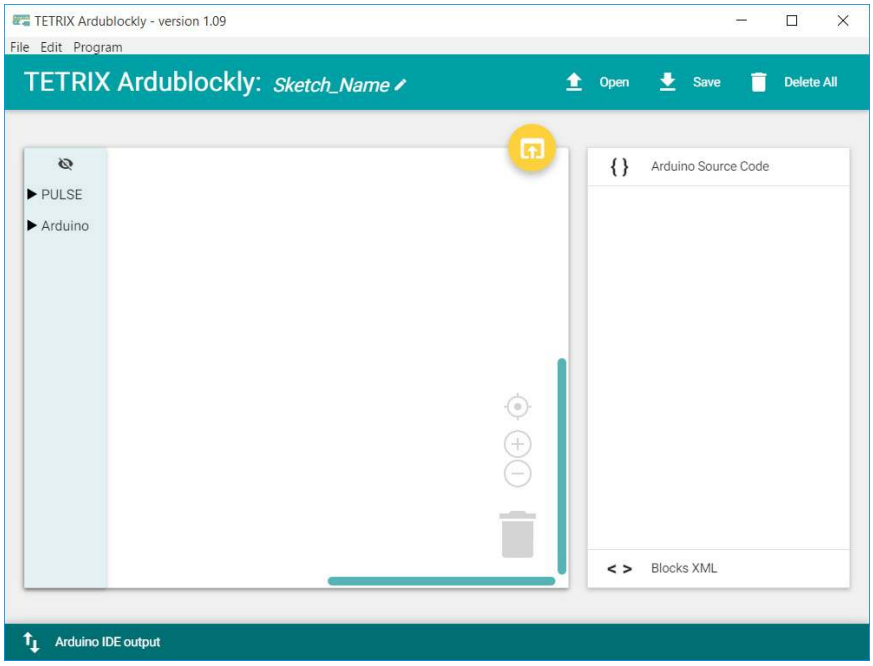


Рисунок 9

Перейдите в **Edit > Preferences** (рис. 10).

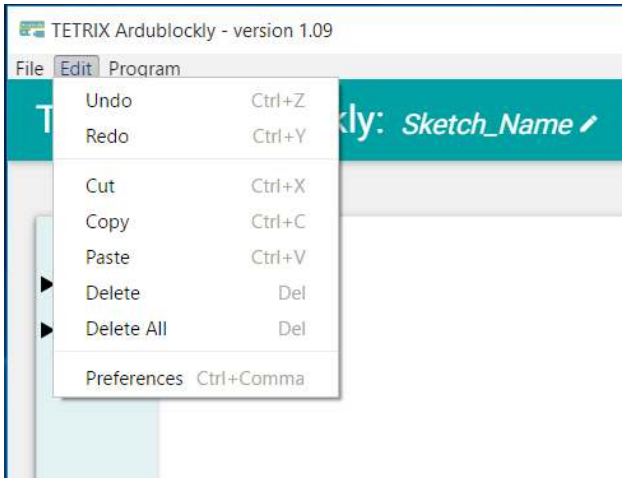


Рисунок 10

Вам необходимо скорректировать настройки в программном обеспечении. Убедитесь, что в качестве адреса компилятора указан путь к месту хранения ПО *Arduino (IDE)* на вашем устройстве (рис. 11).

Нажмите на поле под заголовком **Compiler Location**.

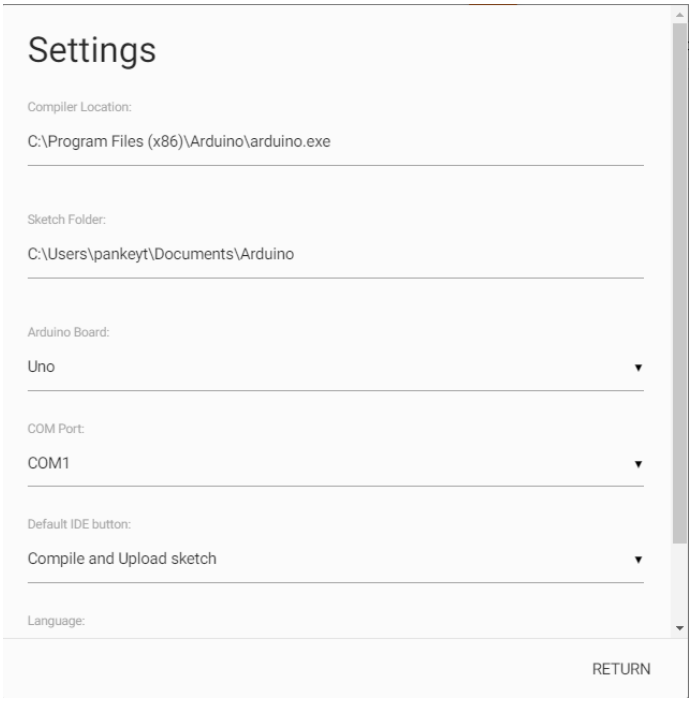


Рисунок 11

Выберите папку вашей операционной системы (рис.12).

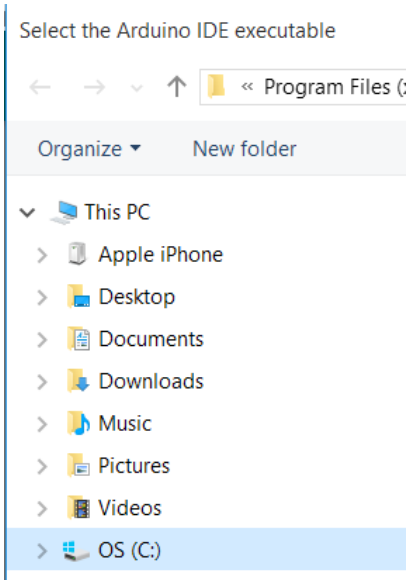


Рисунок 12

Найдите файлы программы Arduino. Выберите приложение Arduino (рис. 13).

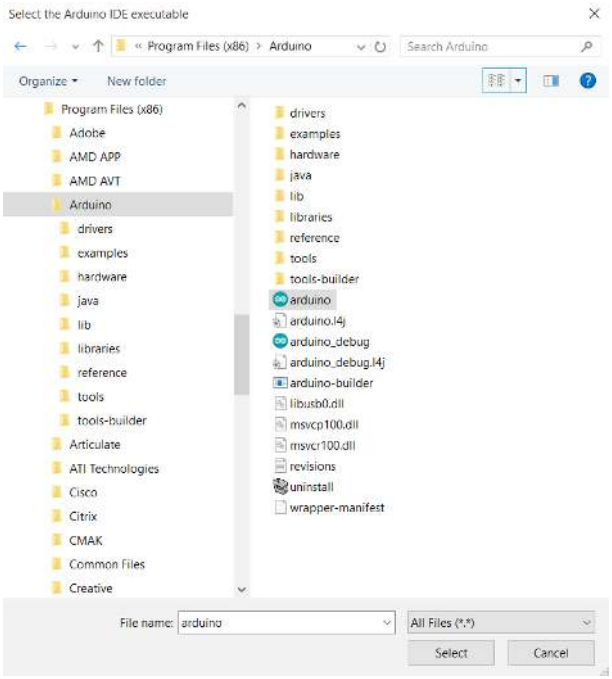


Рисунок 13

Нажмите на поле под заголовком **Sketch Folder**. Выберите место для сохранения папки скетча. Можно сохранить её в папке "Документы" или по иному адресу на вашем устройстве (рис. 14).

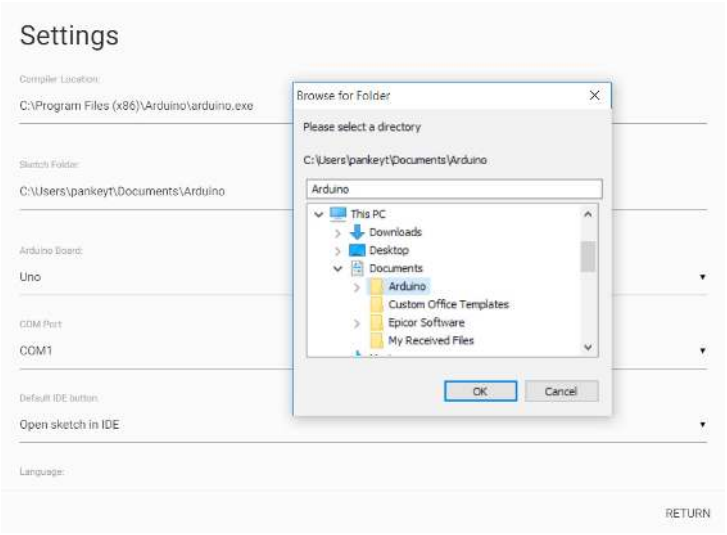


Рисунок 14

Используемая плата Arduino — Uno (рис. 15).

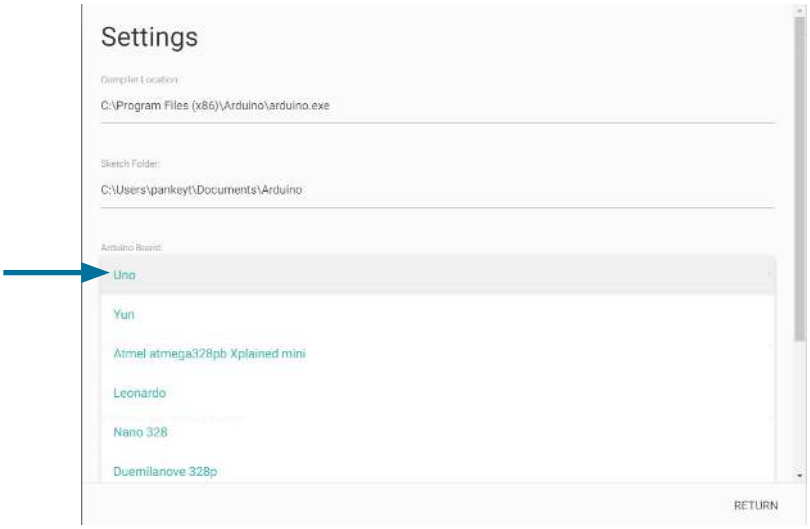


Рисунок 15

Укажите тот COM-порт, с которым связан контроллер PULSE (рис. 16).

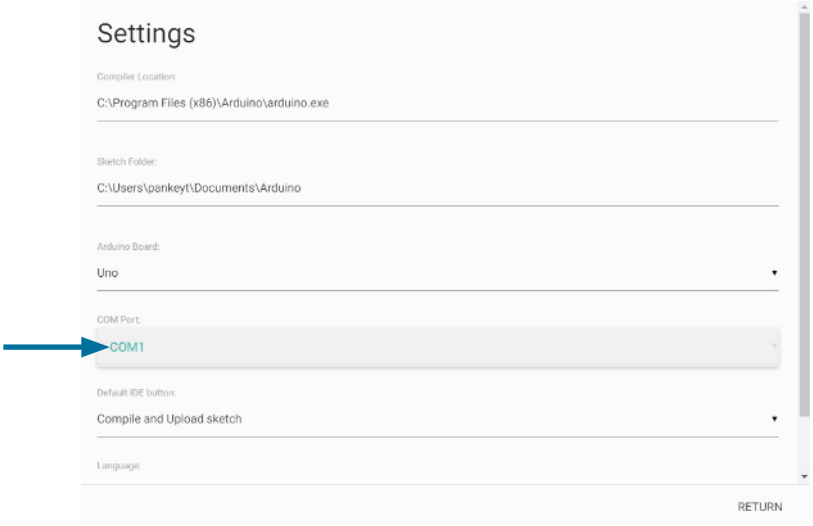


Рисунок 16

По завершении описанной процедуры контроллер PULSE будет подключён к ПО *TETRIX Ardublockly* и можно приступить к написанию кода.

Первое знакомство *с ПО TETRIX Ardublockly*

Основные сведения о программном обеспечении:

Это основной интерфейс, с которым вы будете работать при открытии программного обеспечения. В крайнем левом поле находится панель инструментов. Чтобы скрыть панель инструментов, нажмите на значок глаза вверху столбца. Панель инструментов включает в себя два вида блоков: блоки, специально предназначенные для работы с контроллером PULSE, и блоки Arduino для дополнительных функций и программирования.

В центре находится область программирования. В него переносят блоки в процессе составления программы. В нижнем правом углу области программирования находится корзина, куда можно перемещать для удаления ненужные блоки. Также с помощью кнопок со знаками плюса и минуса можно увеличивать и уменьшать текст в области программирования. Кнопка с изображением мишени позволяет центрировать область программирования относительно расположения блоков.

В правом верхнем углу области программирования находятся три кнопки. При помощи первой кнопки, "Открыть скетч", можно открыть скетч в ПО *Arduino (IDE)*. Программа откроется в синтаксическом формате. При необходимости синтаксический код можно редактировать в ПО *Arduino (IDE)*. Эта процедура в настоящем руководстве не рассматривается. Кнопка "Проверить" позволяет проверить скетч на отсутствие ошибок. Последняя кнопка — "Загрузить". Нажатием на эту кнопку можно передать написанную программу в контроллер PULSE (рис. 17).

Примечание: Эти кнопки меняют порядок, в зависимости от того, какая была нажата последней.

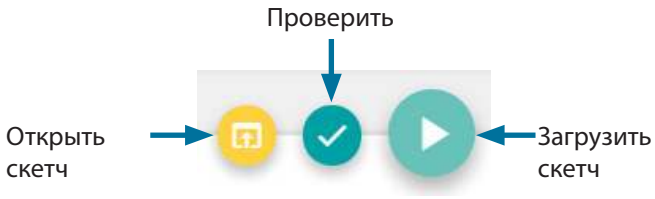


Рисунок 17

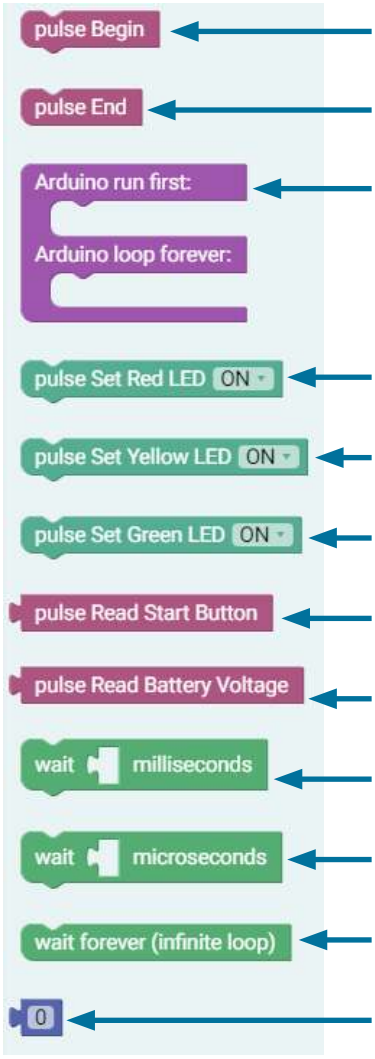
В крайнем правом поле показан текстовый исходный код Arduino. Вы не можете редактировать исходный код. При необходимости можно скрыть колонку с исходным кодом.

В самой нижней части экрана находится панель команд вывода Arduino IDE, которая скрыта по умолчанию. Если нажать на любой участок нижней панели, откроется панель состояния. Она даёт информацию в режиме реального времени о состоянии загрузки.

Чтобы присвоить написанной программе название, нажмите на **Sketch_ Name** и введите имя файла. Для сохранения копии программы необходимо нажать на **Save**. Для открытия существующего файла нажмите на **Open**.

Блоки PULSE:

Управляющие



Запускает робототехнический контроллер PULSE

Немедленно останавливает выполнение программы

Задаёт функции setup() (настройки) и loop() (цикла) в Arduino

Задаёт состояние красного светодиода PULSE

Задаёт состояние жёлтого светодиода PULSE

Задаёт состояние зелёного светодиода PULSE

Считывает состояние пусковой кнопки контроллера PULSE

Считывает напряжение аккумуляторной батареи

Ожидание определённого периода времени в миллисекундах

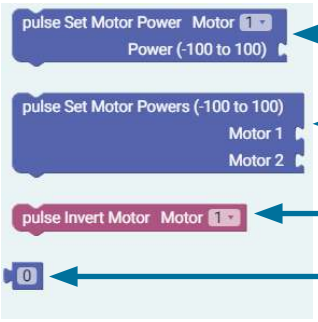
Ожидание определённого периода времени в микросекундах

Ожидание неограниченного времени, бесконечный цикл

Элемент времени (мини-блок, который можно присоединять к другим блокам и редактировать)

Рисунок 18

Электродвигатели



Задаёт мощность и направление вращения электродвигателя постоянного тока; диапазон мощности: от -100 до 100

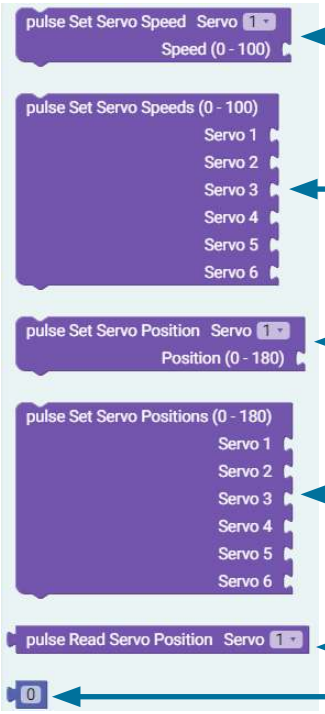
Задаёт мощность и направление вращения электродвигателей постоянного тока 1 и 2

Меняет направление вращения по выбранному каналу электродвигателя

Элемент времени

Рисунок 19

Сервоприводы



Задаёт частоту вращения по выбранному каналу сервопривода

Задаёт частоту вращения по всем шести каналам сервоприводов

Задаёт положение выбранного сервопривода в диапазоне от 0 до 180 градусов

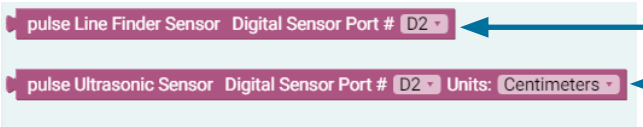
Задаёт положение всех шести сервоприводов

Считывает положение выбранных сервоприводов и сохраняет его как значение переменной

Элемент времени

Рисунок 20

Датчики



Считывает выходное значение датчика линии

Считывает выходное значение ультразвукового датчика в сантиметрах или дюймах

Рисунок 21

Совет: В выпадающем меню Arduino имеется ряд дополнительных блоков, выполняющих специальные функции.

Советы по устранению неисправностей:

- Скетч всегда должен начинаться с блока "pulse Begin". Для завершения программы можно использовать блок "pulse End" или нажать на красную кнопку сброса параметров/остановки на корпусе контроллера.
- Если ваш скетч не загружается на контроллер PULSE, попробуйте отсоединить и заново подключить кабель USB. Также можно попытаться закрыть и заново открыть программу.
- Откройте скетч в окне вывода Arduino IDE и проверьте, нет ли ошибок в коде.
- Если вам требуется дополнительная информация, можно пользоваться материалами в приложении.
- Хотите увидеть всё в действии? В этом вам поможет серия видеороликов RoboBench, снятых специально для *Руководства по программированию контроллера PULSE*. Вся серия представлена по адресу **video.tetrixrobotics.com**, а также на канале TETRIXrobotics на сайте YouTube.

Вводные упражнения

Теперь можно приступить к выполнению упражнений. Каждое из пяти вводных упражнений знакомит с ПО *TETRIX* и принципами его взаимодействия с контроллером PULSE, а также помогает выбрать основное аппаратное обеспечение. Успешно выполнив эти первые пять упражнений, вы не только убедитесь на собственном опыте в лёгкости работы с ПО *TETRIX*, но и подготовитесь к программированию базового робота PULSE.

Скетч *TETRIX Ardublockly*

Перед написанием первой программы важно понять несколько основных правил, касающихся скетчей. Лучше всего воспринимать скетч как перечень инструкций, которые будут выполняться в том порядке, в котором они записаны. Каждый скетч включает в себя несколько инструкций и, как правило, каждая инструкция равна одному блоку в скетче.

Каждый блок содержит строки текста, называемые кодом (поэтому программирование иногда называют кодированием). Текстовое программирование также известно как синтаксическое программирование. В *TETRIX Ardublockly* применяются визуальные элементы программирования, что представляет собой графическое программирование.

Во многих случаях, лучший способ научиться программировать — это следовать примеру. В рамках предложенных упражнений вы рассмотрите несколько примеров написания программы, которые помогут понять, как создавать скетчи и загружать их в контроллер PULSE.

Примечание: Если вы уже освоили написание скетчей в Arduino и хотите перейти на следующий уровень, вы можете начать с рассмотрения всех функций библиотеки. Мы составили и включили в руководство описание функций с примерами того, как каждая из них может действовать в скетче Arduino. Эта информация представлена в приложении к руководству на стр. 132-138. По мере обновления библиотеки для новых версий, в будущем в неё могут быть включены дополнительные функции.

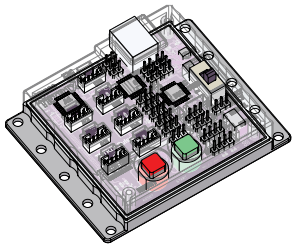
Примечание: В дополнение к библиотеке PULSE существует целая коллекция команд на языке Arduino, которые необходимо понять до создания действующих программ. Ссылки на справочные материалы по языку Arduino, а также множество руководств по его изучению можно найти на главной странице сайта Arduino **www.arduino.cc**.

Упражнение 1: Привет, мир!

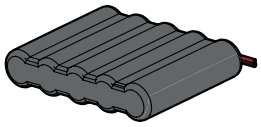
Введение

Нужно написать простую программу, или скетч, которая заставит мигать красный светодиод контроллера PULSE. Представьте, что контроллер вам подмигивает! Упражнение сводится к составлению программы "Привет, мир!" и обычно служит вводным упражнением для любого начинающего программиста. Скетч, который вам предстоит создать, представляет собой простой базовый код PULSE. Всё, что нужно для этого — контроллер PULSE, источник электропитания и подключение к компьютеру через USB.

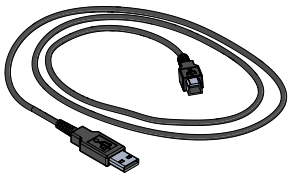
Необходимое оборудование



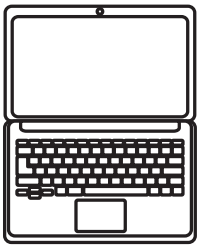
1x
контроллер TETRIX® PULSE™ 44268



1x
Аккумуляторная батарея NiMH 6 В 40235



1x
Кабель USB с разъёмами типа A-B, 40967



1x
компьютер

Открытие программы

Начнём с рассмотрения первого примера скетча. Откройте скетч, последовательно выбрав в меню пункты **Examples > GS_Activity_1**. Откроется окно нового скетча под заголовком GS_Activity_1 (рис. 22).

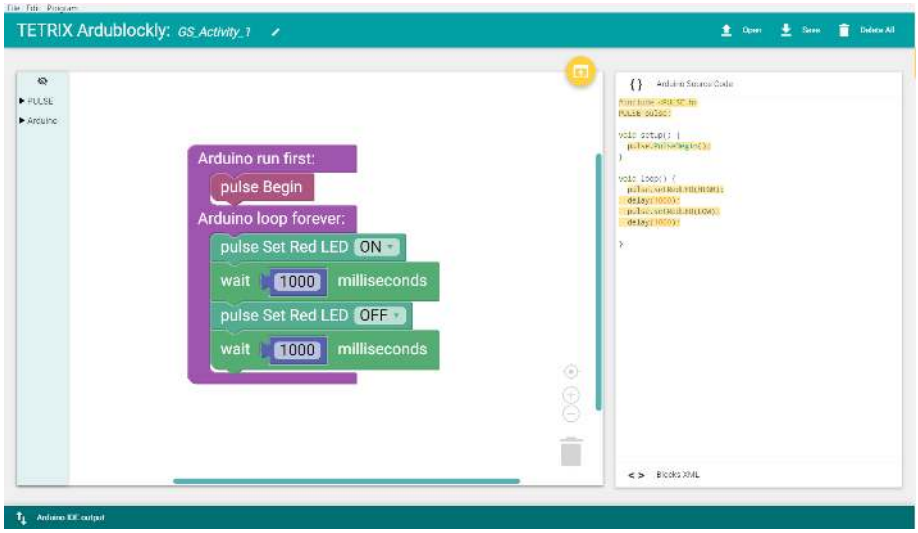


Рисунок 22

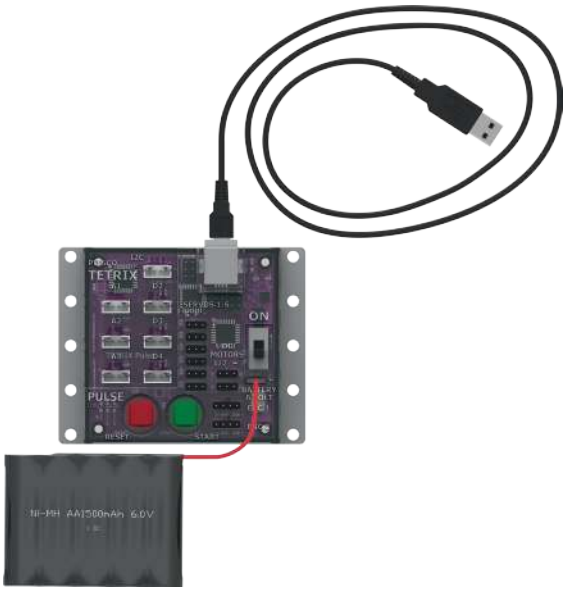
Совет: Не можете найти программу? Сверьтесь с разделом «Установка и настройка программного обеспечения *Arduino Software (IDE)*» на стр. 10-11.

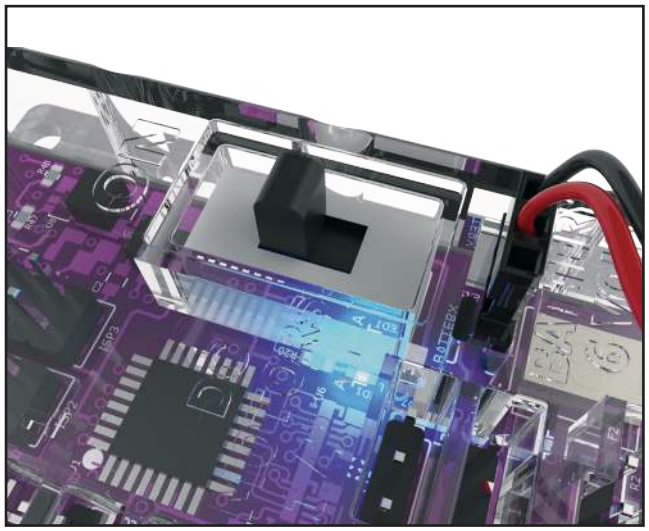
Описание

Перед загрузкой скетча в контроллер PULSE необходимо подключить контроллер к электропитанию, установить связь с компьютером и проверить, обнаружен ли контроллер системой компьютера.

После подключения контроллера PULSE, как показано на иллюстрации, переведите выключатель электропитания в положение "ON". Работающий синий световой индикатор указывает на то, что питание включено.

Совет: Чтобы убедиться, определяется ли контроллер PULSE системой компьютера, проверьте порт в том же порядке, что описан в разделе "Настройка связи через USB" на стр. 12-14.





Выполнение управляющего кода

Чтобы загрузить скетч в контроллер PULSE, щёлкните по кнопке "Загрузить скетч" [**Upload Sketch**] (рис. 23). Чтобы проверить состояние загрузки, нажмите на **Arduino IDE output** внизу окна программы (рис. 24).

Рисунок 23

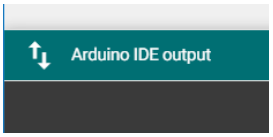


Рисунок 24

Рисунок 25

Наблюдайте за кнопкой загрузки скетча [Upload Sketch]. Ход загрузки отображается движением цветной линии по окружности кнопки (рис. 25). Дождитесь окончания загрузки.

При загрузке данных жёлтые светодиоды на корпусе контроллера PULSE будут мигать. После её завершения рядом с красной кнопкой сброса параметров/остановки будет непрерывно гореть зелёный светодиод. Свечение зелёного светодиода означает, что код готов к выполнению. Для его запуска нажмите на зелёную кнопку "Пуск". Красный светодиод рядом с кнопкой сброса параметров/остановки будет мигать с интервалом одна секунда. Чтобы остановить программу, нажмите на кнопку сброса параметров/остановки.

Поздравляем! Вы успешно загрузили свой первый скетч в контроллер PULSE и продемонстрировали всем его действие.

Совет: Почему время измеряется в миллисекундах? Программисты используют миллисекунды для более точного отсчёта времени. 1000 миллисекунд равно 1 секунде.

Дальнейшее изучение

Рассмотрите правое поле окна программы. Вы увидите текст с различной пунктуацией. Это называется синтаксическое или текстовое программирование. Каждый блок в скетче обычно представлен одной строкой текста. Строки текста в составе скетча также называют кодом, поэтому программирование иногда называют кодированием.

Совет: Блок времени находится в ПО в категории "Электродвигатели".

Можно изменить некоторые параметры скетча и посмотреть, как это повлияет на работу красного светодиода. Блок ожидания определяет длительность включения и выключения светодиода (рис. 26). Это изменяемый параметр скетча. Поэкспериментируйте с его значениями таким образом, чтобы светодиод мигал в разных режимах. Попытайтесь сделать так, чтобы он мигал быстрее или медленнее.

Рисунок 26

Каждая программа начинается с блока настройки (setup) и цикла (loop). В поле блока, предназначенном для настройки, всегда первым будет блок pulse Begin. Сюда помещают все прочие блоки, требующие лишь разовой настройки. Блоки, которые в программе будут зациклены постоянно, помещают в блок цикла (loop).

В рассматриваемом примере красный светодиод включается и выключается многократно, поскольку эти действия помещены в цикл. Выполнение цикла прекратится только при остановке выполнения программы (рис. 27).

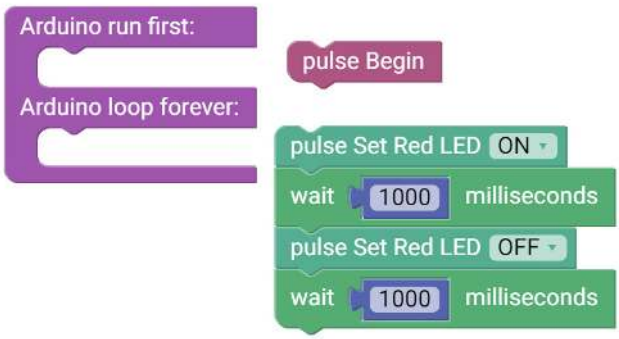


Рисунок 27

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч, управляющий миганием светодиода. Сделайте так, чтобы мигал не только красный, но и зелёный светодиод.

Вспыхивающий или мигающий свет можно использовать в качестве сигнализации или для связи на большом расстоянии. Попробуйте создать последовательность мигающих световых сигналов наподобие стоп-сигнала.

Чтобы начать новый скетч, выберите **File > New**. Вставьте соответствующие блоки в скетч. По завершении работы над скетчем вы можете проверить его на наличие ошибок при помощи встроенного программного инструмента. Для проверки написанной программы нажмите на кнопку **"Проверить скетч"** (рис. 28).

При этом произойдет компилирование кода без его загрузки в контроллер. Если в коде есть ошибки, они будут отображаться в окне ошибок компилятора в нижней части окна скетча (рис. 29). Ошибки необходимо устранить до загрузки кода в контроллер PULSE.

Рисунок 28

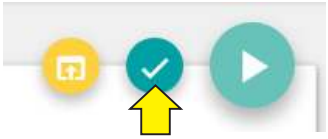


Рисунок 29

Если ошибки отсутствуют, компилятор завершит работу с выводом сообщения, что компилирование выполнено и можно загружать код в контроллер. В скетче щёлкните по кнопке "Загрузить скетч" [**Upload the Sketch**] и посмотрите, удалось ли вам создать модель светофора.

Совет: При выполнении дополнительного упражнения можно пользоваться образцом скетча в приложении под заголовком **GS_Activity_1_Extension_Example**.

Связь с реальным миром

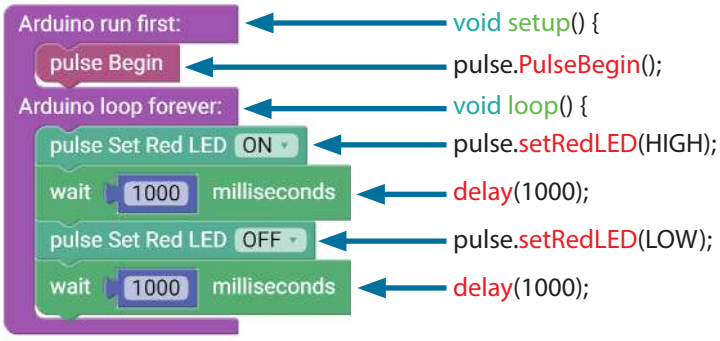
Подумайте о том, что вас окружает. В каких устройствах используется вспыхивающий или мигающий свет? Например, в их числе можно назвать светофоры, праздничные огни, полицейские мигалки. Световые сигналы могут служить для предупреждения или привлечения внимания.

Профессии: дорожный мастер, инженер службы эксплуатации светофоров, инженер-электронщик

Связь с точными и естественными науками

- Физика
  - o Электричество
  - o Светоизлучающие диоды
- Технология
  - o Проектирование электронных систем
  - o Программирование светофоров
- Техническое конструирование
  - o Виды освещения
  - o Оптика
- Математика
  - o Частота
  - o Режим

Связь блоков с текстом



Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();
}

void loop() {
pulse.setRedLED(HIGH);
delay(1000);
pulse.setRedLED(LOW);
delay(1000);

}
```

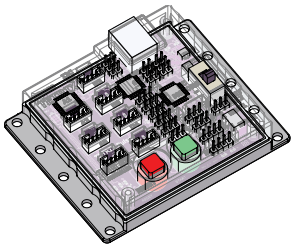
Примечание: Обратите внимание, что ON ("включено") означает "высокий", а OFF ("выключено") — "низкий".

Упражнение 2: Вращение электродвигателей постоянного тока

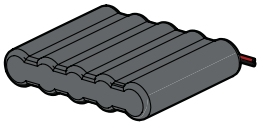
Введение

Для второго упражнения добавляется элемент механизма движения. Вы должны написать скетч, который будет приводить в движение электродвигатель постоянного тока.

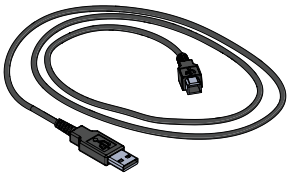
Необходимое оборудование



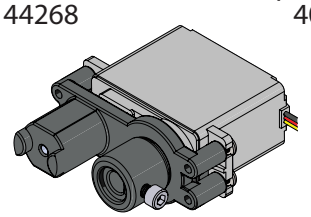
1x
контроллер TETRIX® PULSE™
44268



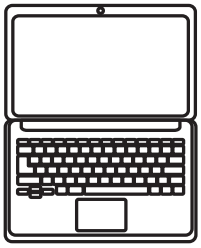
1x
Аккумуляторная батарея NiMH 6 В
40235



1x
Кабель USB с разъёмами типа A-B,
40967



1x
Электродвигатель постоянного тока TETRIX 44298 со скобой крепления сервоприводов 40232
(указания по сборке на стр. 54-55)



1x
компьютер

Открытие программы

Прежде чем открыть следующий пример скетча, необходимо сохранить все скетчи, к которым вы будете обращаться позже. Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты **Examples > GS_Activity_2.** Откроется окно нового скетча под заголовком GS_Activity_2 (рис. 30).

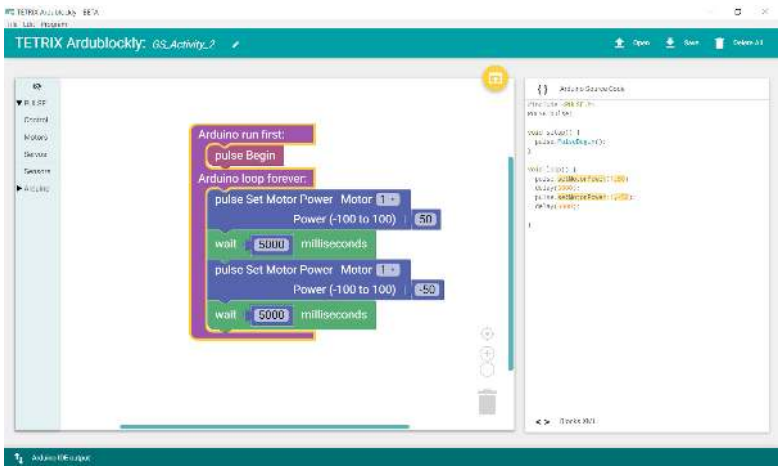


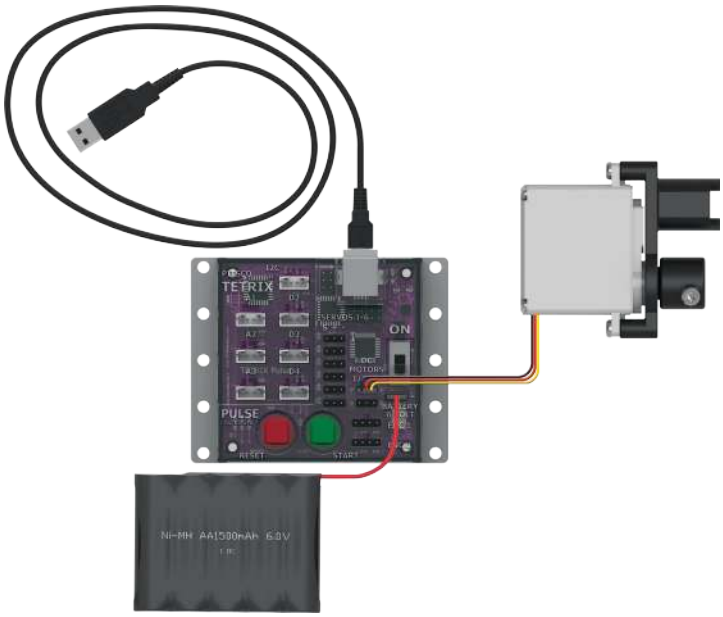
Рисунок 30

Описание

Под управлением этого скетча электродвигатель постоянного тока будет вращаться в течение пяти секунд и затем остановится. Далее он будет вращаться в противоположном направлении пять секунд. Электродвигатель будет работать на половину своей мощности. Работа в описанном режиме будет продолжаться до нажатия на кнопку сброса параметров/остановки.

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Не забывайте, что подсоединив электродвигатель, вы добавили новое подключение. Подключите провод электродвигателя к порту электродвигателя постоянного тока 1.



Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После включения светодиодного индикатора нажмите на зелёную кнопку "Пуск" на корпусе контроллера PULSE.

Проследите за направлением и продолжительностью вращения электродвигателя. Соответствует ли работа электродвигателя запланированному действию программы? Чтобы остановить электродвигатель, нажмите на кнопку сброса параметров/остановки.

Дальнейшее изучение

С помощью блоков установки мощности (pulse Set Motor Power) можно задать мощность электродвигателя в процентах, в диапазоне от 0 (нулевая мощность) до 100 (полная мощность). Если требуется половина мощности, установите её значение на 50 (рис. 31).

Рисунок 31

Совет: Чем электродвигатель постоянного тока отличается от сервопривода? Электродвигатель постоянного тока имеет два провода и может вращаться непрерывно. Сервопривод TETRIX имеет три провода и может устанавливаться в различных положениях, но он вращается только в пределах 180.

Во время выполнения программы, при вращении электродвигателя вперёд или по часовой стрелке рядом со шнуром электродвигателя будет светиться красный индикатор. В скетче это направление вращения представлено положительной величиной (рис. 32).

Рисунок 32

Во время выполнения программы, при вращении электродвигателя назад или против часовой стрелки рядом со шнуром электродвигателя будет светиться зелёный индикатор. В скетче это направление вращения представлено отрицательной величиной (рис. 33).

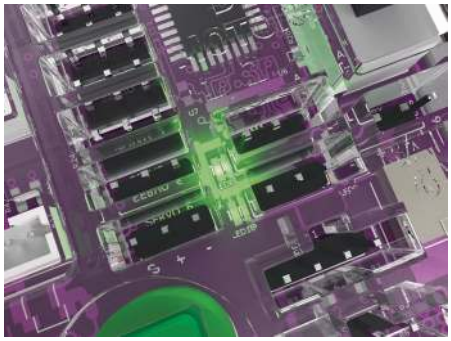


Рисунок 33

Потренируйтесь менять параметры в скетче. Можно изменять мощность электродвигателя, направление вращения электродвигателя, условия останова и задержку между действиями. Понаблюдайте за тем, как такие изменения влияют на работу электродвигателя.

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч с использованием электродвигателя постоянного тока и светодиодных индикаторов контроллера. Вспомните, чему научились, выполняя первое упражнение, и придумайте необычный способ использовать мигающие светодиоды с вращающимся электродвигателем.

Совет: При выполнении дополнительного упражнения можно пользоваться образцом скетча в приложении под заголовком **GS_Activity_2_Extension_Example**.

Связь с реальным миром

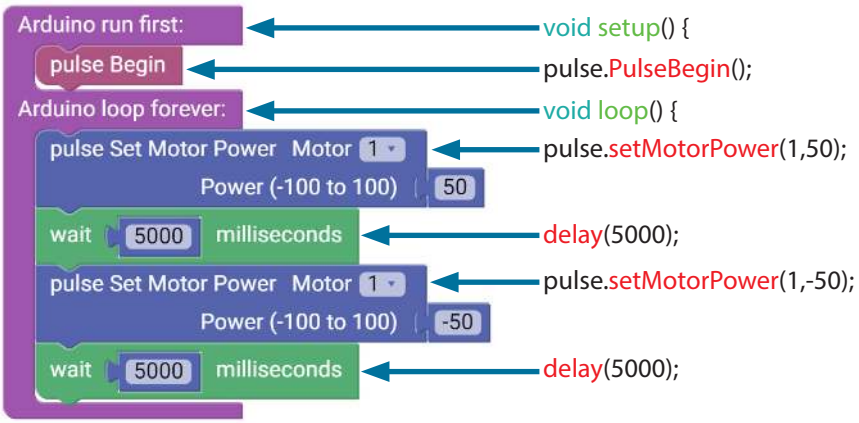
Электродвигателями постоянного тока оснащены многие устройства и оборудование, например, лифты, поезда, машинное оборудование, электроинструменты, автомобили, вентиляторы. Зачастую они приводят в действие разные электронные устройства.

Профессии: механик малых двигателей, инженер-механик, специалист по техническому обслуживанию машинного оборудования

Связь с точными и естественными науками

- Физика
  - Постоянный ток
  - Мощность
- Технология
  - Применение электродвигателей постоянного тока
  - Крутящий момент
- Техническое конструирование
  - Проводные соединения электродвигателей
  - Заземление
- Математика
  - Обороты в минуту (об/мин)
  - Градусы

Связь блоков с текстом



Примечание: Указанные в круглых скобках параметры (1,50) означают, что электродвигатель № 1 будет вращаться по часовой стрелке с мощностью 50 %.

Исходный код Arduino

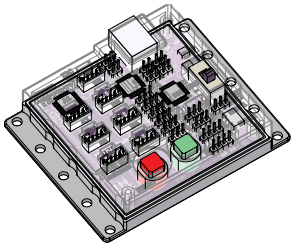
```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();
}

void loop() {
pulse.setMotorPower(1,50);
delay(5000);
pulse.
setMotorPower(1,-50);
delay(5000);

}
```

Упражнение 3: Вращение сервоприводов

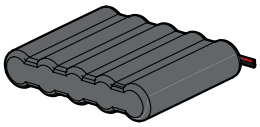
Введение

Третье упражнение предусматривает создание скетча, который будет управлять вращением сервопривода. Сервоприводы перемещаются в то положение, которое было задано, независимо от их исходного положения. Сервоприводы TETRIX могут вращаться только в диапазоне от 0 до 180°. Например, можно дать сервоприводу команду переместиться в положение 45° независимо от того, откуда он начал движение. Если исходная позиция 0°, сервопривод повернётся по часовой стрелке на 45°. Если вращение начинается от 120°, он повернётся против часовой стрелки на 45°.

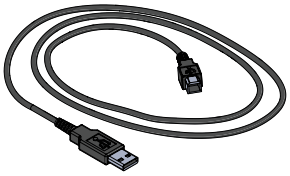
Необходимое оборудование



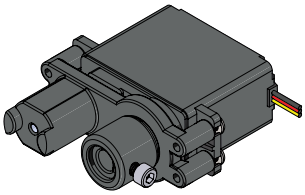
1x
контроллер TETRIX® PULSE™
44268



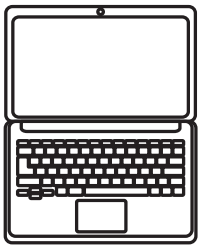
1x
Аккумуляторная батарея NiMH 6 В
40235



1x
Кабель USB с разъёмами типа A-B,
40967



1x
Сервопривод HiTec HS322-HD 40538 со скобой крепления сервоприводов 40232 (указания по сборке на стр. 54-55)



1x
компьютер

Открытие программы

Прежде чем открыть следующий пример скетча, необходимо сохранить все скетчи, к которым вы будете обращаться позже. Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты **Examples > GS_Activity_3.** Откроется окно нового скетча под заголовком GS_Activity_3 (рис. 34).

Рисунок 34

Описание

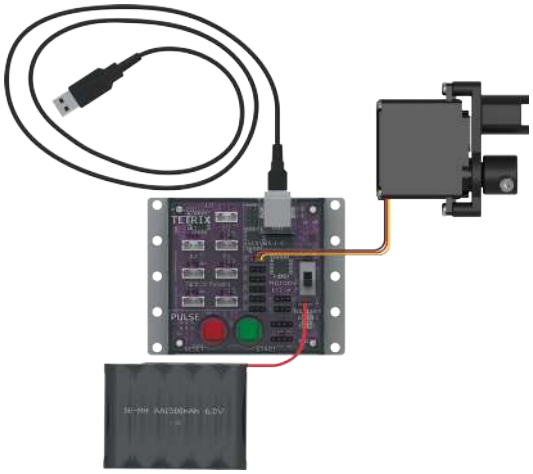
Под управлением третьего скетча сервопривод должен вращаться вперёд и назад между двумя разными положениями с установленной частотой вращения. Привод должен работать со скоростью 25 %. Работа в описанном режиме будет продолжаться до нажатия на кнопку сброса параметров/остановки.

Сервоприводы отличаются намного более высокой точностью движения по сравнению с электродвигателями постоянного тока.

Совет: Чем электродвигатель постоянного тока отличается от сервопривода? Электродвигатель постоянного тока имеет два провода и может вращаться непрерывно. Сервопривод TETRIX имеет три провода и может устанавливаться в различных положениях, но он вращается только в пределах 180.

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Не забывайте, что подсоединив сервопривод, вы добавили новое подключение. Подключите провод сервопривода к порту сервопривода 1.



Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После включения светодиодного индикатора нажмите на зелёную кнопку "Пуск" на корпусе контроллера PULSE.

Проследите за направлением и продолжительностью вращения сервопривода. Соответствует ли работа сервопривода запланированному действию программы? Чтобы остановить электродвигатель, нажмите на кнопку сброса параметров/остановки.

Дальнейшее изучение

В этом скетче вы используете два новых блока PULSE: pulse Set Servo Speed (установка частоты вращения сервопривода) и pulse Set Servo Position (установка позиции сервопривода). В оба блока входит по два параметра, но они различаются.

В блоке pulse Set Servo Speed задаются такие параметры, как канал сервопривода и частота вращения сервопривода.

В примере сервопривод 1 будет вращаться с мощностью 25 % до тех пор, пока не достигнет положения, заданного блоком pulse Set Servo Position. Этот блок находится в блоке настройки (setup), потому что должен быть однократно включён в перечень начальных блоков программы (рис. 35).

Рисунок 35

В блок pulse Set Servo Position включено два следующих параметра: канал сервопривода и заданное конечное положение. В примере параметры блока pulse Set Servo Position заданы таким образом, что сервопривод 1 повернётся в заданное конечное положение 180°. Затем он перейдет в положение 0°, двигаясь с той же скоростью.

Так как программа заключена в цикл, сервопривод будет перемещаться в указанные положения до тех пор, пока выполнение программы не будет прекращено (рис. 36).

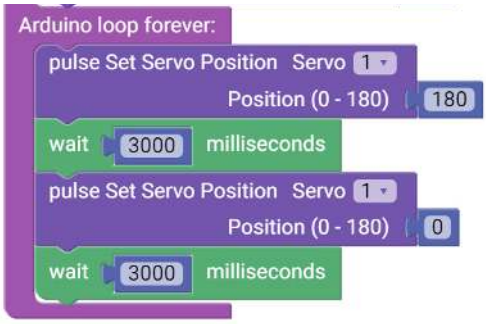


Рисунок 36

В этом скетче оба блока действуют совместно, задавая сервоприводу не только заданное конечное положение, но и скорость, с которой он должен к нему двигаться.

Вы можете менять положение и частоту вращения сервопривода, изменяя значение обеих функций. Потренируйтесь менять параметры в скетче. Понаблюдайте за тем, как такие изменения влияют на работу сервопривода.

Совет: При выполнении дополнительного упражнения можно пользоваться образцом скетча в приложении под заголовком **GS_Activity_3_Extension_Example**.

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч, управляющий движением сервопривода. Также можно задействовать электродвигатель постоянного тока и светодиодные индикаторы контроллера. Вспомните, чему научились в ходе предыдущих упражнений, и придумайте необычные способы объединить пройденные функции.

Связь с реальным миром

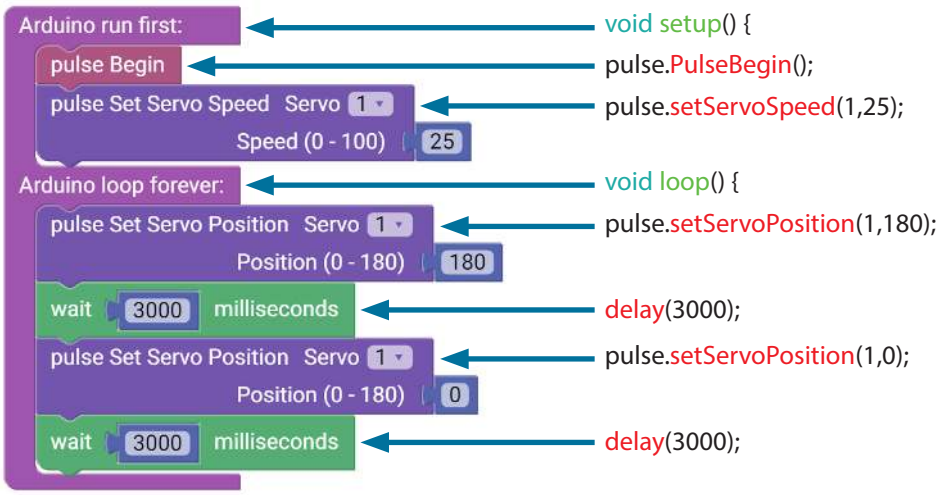
Сервоприводы присутствуют в радиоуправляемых моделях автомобилей, обеспечивая управление их движением, и в радиоуправляемых авиамоделях, где приводят в действие закрылки и руль направления. Сервоприводы применяются там, где требуется точность движения, в частности, в манипуляторах роботов, механизмах захвата, поворотных устройствах камер.

Профессии: рабочий производства, инженер промышленного производства, слесарь

Связь с точными и естественными науками

- Физика
  - Ток
  - Правила Флеминга
- Технология
  - Сборка двигателей
  - Система обратной связи
- Техническое конструирование
  - Превращение энергии
  - Зубчатая передача двигателей
- Математика
  - Правильность
  - Точность

Связь блоков с текстом



Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();
pulse.setServoSpeed(1,25);
}

void loop() {
pulse.
setServoPosition(1,180);
delay(3000);
pulse.setServoPosition(1,0);
delay(3000);

}
```

Примечание: Сервопривод вращается в диапазоне от 0 до 180 градусов.

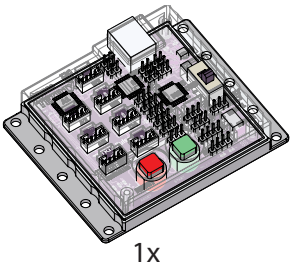
Упражнение 4: Первое знакомство с датчиком линии

Введение

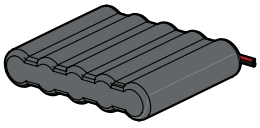
Для четвёртого упражнения понадобится датчик линии. С помощью примера вы подключите датчик линии к порту цифрового датчика D2. Вы должны написать скетч для чтения цифровых данных, поступающих от датчика линии.

Датчики позволяют нам получать информацию об окружающих условиях. Вид информации зависит от вида датчика. Датчик линии отличает светлую поверхность от тёмной с помощью отражённого инфракрасного света.

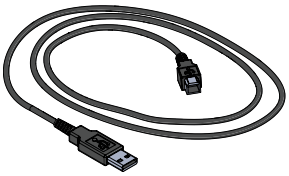
Необходимое оборудование



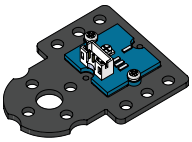
1x
контроллер TETRIX® PULSE™ 44268



1x
Аккумуляторная батарея NiMH 6 В 40235

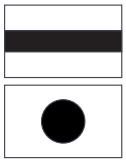


1x
Кабель USB с разъёмами типа A-B, 40967



1x
комплект датчика линии 43056

1x
компьютер



1x
поверхность с контрастными светлыми и тёмными областями

Открытие программы

Прежде чем открыть следующий пример скетча, необходимо сохранить все скетчи, к которым вы будете обращаться позже. Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты **Examples > GS_Activity_4.** Откроется окно нового скетча под заголовком GS_Activity_4 (рис. 37).

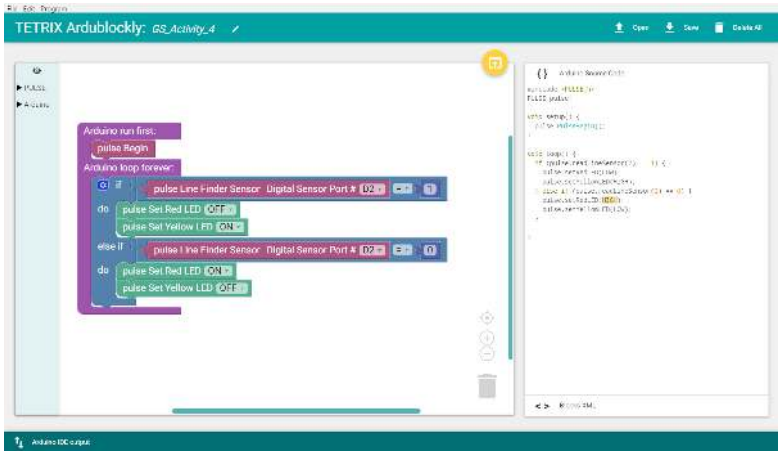


Рисунок 37

Описание

Работа над четвертым скетчем позволит вам более подробно ознакомиться с программированием датчиков и применением логических блоков. Вам предстоит при помощи датчика линии определить, светлая поверхность или тёмная.

Логический блок "если.., то" (if-do) действует в соответствии со своим названием. Все действия, включённые в цикл, будут последовательно исполняться до тех пор, пока скетч не остановят с помощью команды или кнопки сброса параметров/остановки (рис. 38).

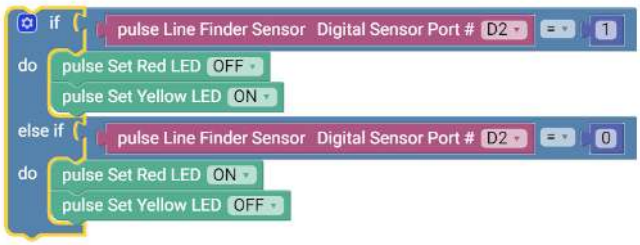
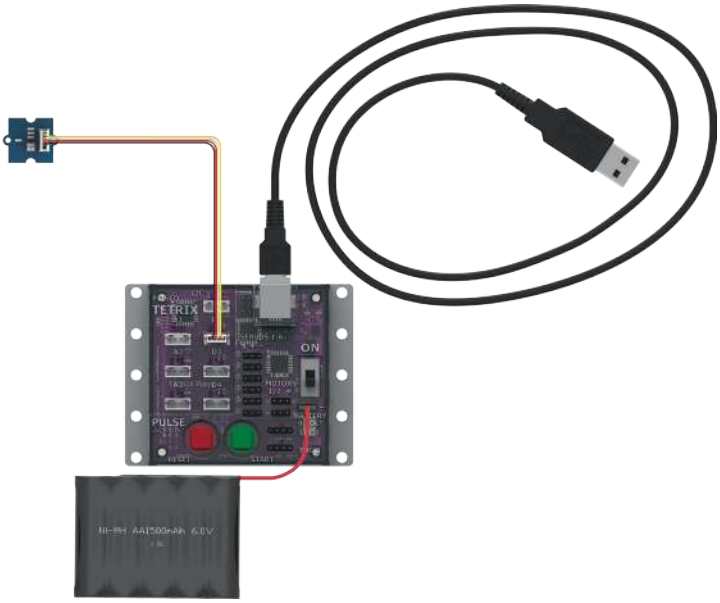


Рисунок 38

На корпусе контроллера PULSE будет загораться красный или жёлтый светодиод, в зависимости от того, на какой поверхности находится датчик.

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Не забывайте, что подсоединив датчик линии, вы добавили новое подключение к порту цифрового датчика D2.



Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После включения светодиодного индикатора нажмите на зелёную кнопку "Пуск" на корпусе контроллера PULSE.

Держите датчик над контрастной поверхностью. Перемещая датчик со светлой области на тёмную, наблюдайте за красным светодиодом на контроллере PULSE. Когда датчик окажется над неотражающей или тёмной поверхностью, включится жёлтый светодиод. Когда датчик окажется над белой или отражающей поверхностью, включится красный светодиод.

Чтобы остановить работу датчика, нажмите на кнопку сброса параметров/остановки.

Дальнейшее изучение

Этот скетч познакомит вас со структурой программы, новыми блоками и функцией сравнения. Рассматриваемая структура программы включает в себя оператор "если" (if), показания датчика и функцию сравнения "=" ("равно"). Оператор сравнения "=" (равно) определяет вид проверки. В этом скетче сигнал, поступающий от датчика линии, будет включать или выключать разные светодиоды.

Базовый оператор "если" позволяет проверять, определённые условия. Если условие соблюдено, тогда программа может выполнить некое действие. Если переменная в части "if" (если) блока цикла истинна, тогда блоки в части "do" (выполнить) выполняются. В данном случае выходной сигнал датчика линии должен быть равен 1, или HIGH, чтобы раздел "то" выполнялся (рис. 39).

Рисунок 39

Если часть "if" (если) цикла loop () ложна, тогда часть "else if" (иначе другое условие) позволяет вам проверить другое условие. Если это условие соблюдено, тогда блоки в части "do" (выполнить) под частью "else if" (иначе другое условие) будут выполнены. В разделе else if выходной сигнал датчика линии должен быть равен 0, или LOW (рис. 40).

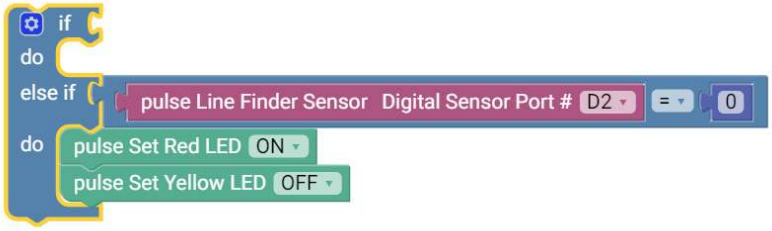


Рисунок 40

Датчик линии возвращает значение "1" (HIGH) или "0" (LOW). Значение "1" возвращается в том случае, если датчик линии обнаруживает тёмную линию или неотражающую поверхность; значение "0" возвращается, если датчик линии обнаружил белую или отражающую поверхность.

Если датчик линии обнаружил линию или неотражающую поверхность, по его сигналу включится красный светодиод. Представьте, что это предмет на пути движения. На датчике линии также загорится красный светодиод. Программа выполняет такое действие, когда значение переменной линии задано как "низкое" (LOW).

Если датчик линии обнаружил белую или неотражающую поверхность, по его сигналу включится жёлтый светодиод. Программа выполняет такое действие, когда значение переменной линии задано как "высокое" (HIGH) (рис. 41).

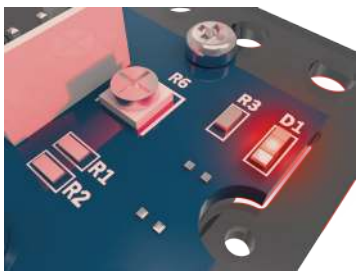


Рисунок 41

Поэкспериментируйте с датчиком линии на разных поверхностях и на разной высоте, наблюдая за тем, как он реагирует.

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч с использованием датчика линии. Вспомните, чему научились в ходе предыдущих упражнений, и придумайте новые необычные действия, выполняемые с соблюдением условия для датчика линии.

Связь с реальным миром

Роботы, способные отслеживать линию, могут применяться самыми разными способами. В настоящее время ведутся разработки по созданию роботов, которые, следуя по линиям, будут развозить материалы в больницах. В будущем они также смогут перемещать материалы по территории складов и перевозить грузы.

Профессии: инженер-материаловед, слесарь по электромеханическому оборудованию, разработчик программного обеспечения

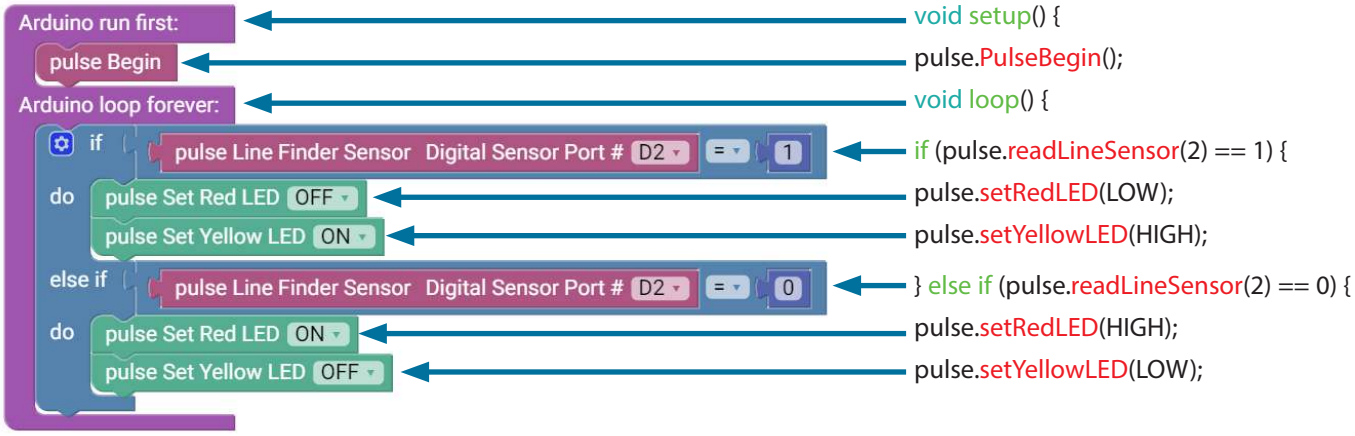
Связь с точными и естественными науками

- Физика
  - Свет
  - Спектр электромагнитного излучения
- Технология
  - Система наведения
  - Автоматизация
- Техническое конструирование
  - Микроконтроллер
  - Встроенная система
- Математика
  - Углы
  - Линии

Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

}

void loop() {
if (pulse.readLineSensor(2) == 1) {
pulse.setRedLED(LOW);
pulse.setYellowLED(HIGH);
} else if (pulse.readLineSensor(2) == 0) {
pulse.setRedLED(HIGH);
pulse.setYellowLED(LOW);
}
}
```

Связь блоков с текстом



Примечание: Все действия, выполняемые в циклах if () или if/else (), заключаются в скобки.

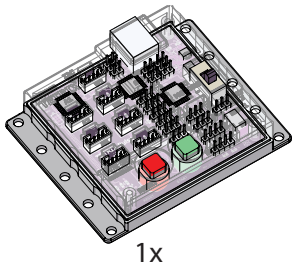
Упражнение 5: Первое знакомство с ультразвуковым датчиком

Введение

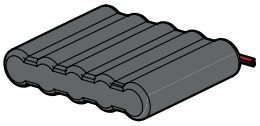
Последнее вводное упражнение позволит вам завершить изучение датчиков посредством создания скетча с применением ультразвукового датчика. В ходе этого упражнения вам предстоит подсоединить ультразвуковой датчик к порту D3 для цифровых датчиков и отобразить в окне монитора порта расстояние до помещённого перед ним объекта.

Как и все прочие датчики, ультразвуковой датчик помогает нам собирать информацию. Он собирает данные для определения расстояния. Датчик испускает короткую серию звуковых импульсов и ждёт, когда они вернутся, отразившись от некоего объекта в досягаемом пространстве. Для определения расстояния до предмета измеряется время возвращения отражённых волн. Датчик способен выполнять измерения в диапазоне около 3-400 сантиметров.

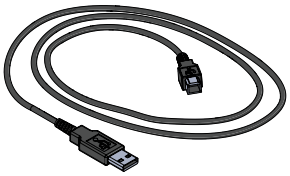
Необходимое оборудование



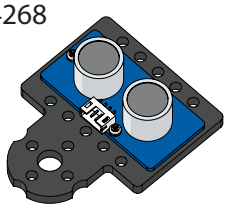
1x
контроллер TETRIX® PULSE™
44268



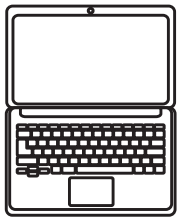
1x
Аккумуляторная батарея NiMH 6 В
40235



1x
Кабель USB с разъёмами типа A-B,
40967



1x
комплект ультразвукового датчика 43055



1x
компьютер

Открытие программы

Прежде чем открыть следующий пример скетча, необходимо сохранить все скетчи, к которым вы будете обращаться позже. Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты **Examples > GS_Activity_5.** Откроется окно нового скетча под заголовком GS_Activity_5 (рис. 42).

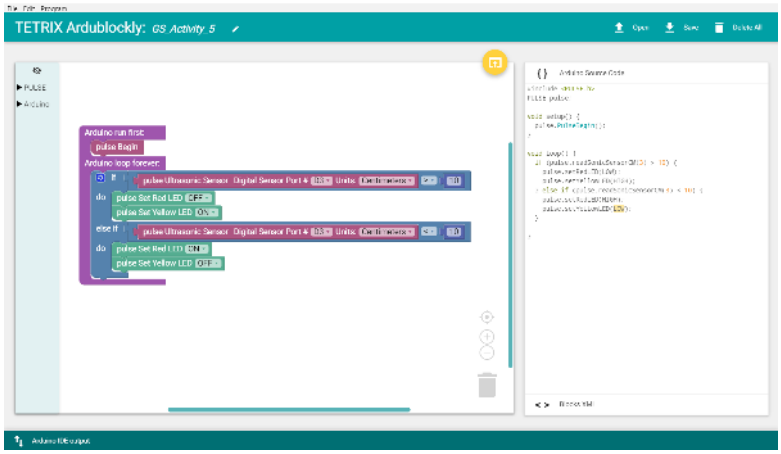


Рисунок 42

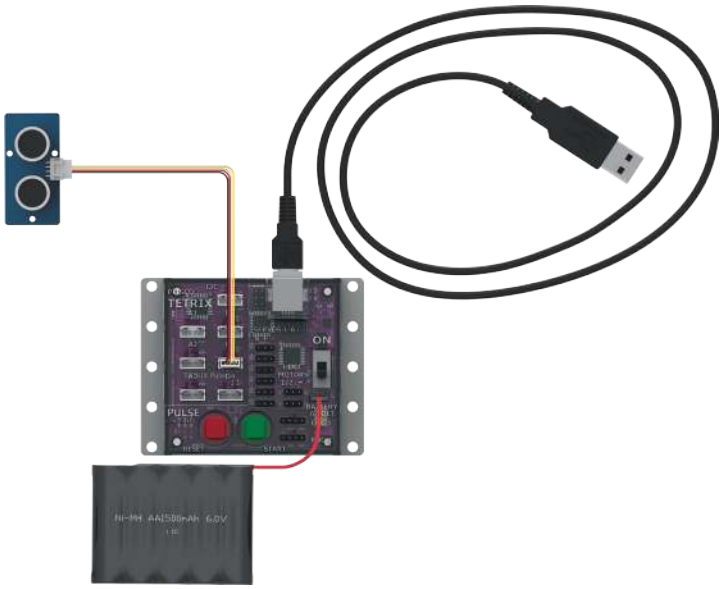
Описание

Для пятого скетча вам потребуется ультразвуковой датчик, испускающий звуковые волны, с помощью которых можно измерить расстояние от датчика до предмета. Понадобится тот же логический блок, что и в предыдущем упражнении.

только вместо измерения интенсивности света этот датчик будет определять расстояние. На корпусе контроллера PULSE будет загораться красный или жёлтый светодиод, в зависимости от того, на каком расстоянии от датчика находится предмет.

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Не забывайте, что подсоединив ультразвуковой датчик, вы добавили новое подключение к порту цифрового датчика D3. Загрузите скетч в контроллер.



Поместите датчик на ровную поверхность стола чувствительной стороной вверх и нажмите на зелёную кнопку "Пуск", чтобы инициировать выполнение кода.

Держите руку над датчиком. Двигайте рукой вверх-вниз и смотрите, что происходит со светодиодными индикаторами на корпусе контроллера PULSE. Чтобы остановить работу датчика, нажмите на кнопку сброса параметров/ остановки.

Дальнейшее изучение

При выполнении скетча различные светодиоды будут включаться и выключаться по сигналу ультразвукового датчика.

Если переменная в части "if" цикла loop () истинна, тогда блоки в части "do" цикла loop () выполняются. В данном случае, если перед датчиком на расстоянии 10 см нет никаких предметов, на корпусе контроллера PULSE будет гореть жёлтый светодиод (рис. 43).

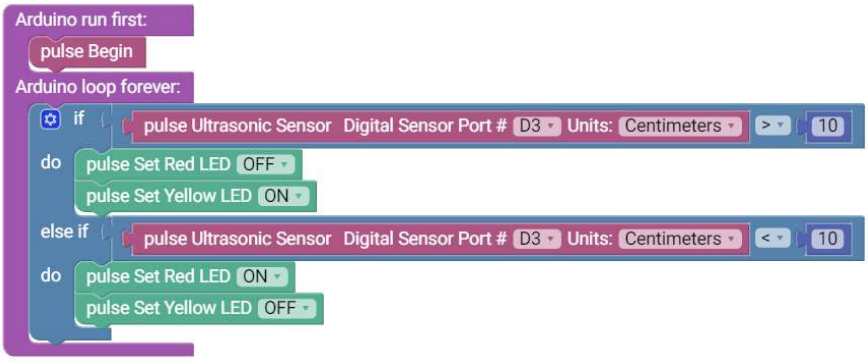


Рисунок 43

Если условие в разделе if ("если") в блоке цикла ложно, раздел else if ("иначе если") позволит выполнить проверку на наличие другого условия. Если это условие соблюдено, тогда блоки в части "do" (выполнить) под частью "else if" (иначе другое условие) будут выполнены. В данном случае, если перед датчиком на расстоянии 10 см находится предмет, на корпусе контроллера PULSE будет гореть красный светодиод (рис. 43).

Поэкспериментируйте с ультразвуковым датчиком, используя разные предметы и разную высоту, понаблюдайте за тем, как реагирует датчик.

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч с использованием ультразвукового датчика. Вспомните, чему научились в ходе предыдущих упражнений, и попробуйте поместить разные предметы перед ультразвуковым датчиком, чтобы проверить, обнаруживает ли он их. Также можно запрограммировать остановку двигателей при достижении определённого расстояния между предметом и ультразвуковым датчиком.

Попробуйте изменить единицу измерения расстояние в блоке ультразвукового датчика с сантиметра на дюйм. Умение применять ультразвуковой датчик позволит сделать робота "зрячим", он сможет объезжать предметы и препятствия на своём пути (рис. 44).

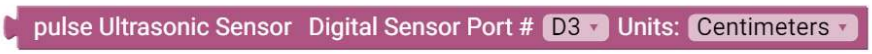


Рисунок 44

Совет: При выполнении дополнительного упражнения можно пользоваться образцом скетча в приложении под заголовком **GS_Activity_5_Extension_Example**.

Связь с реальным миром

В современных автомобилях используются интеллектуальные технологии. В автомобилях есть видеокамеры заднего вида и система продольной парковки, а если некая помеха слишком приблизится к автомобилю, то он издаст предупредительный сигнал. Все эти системы действуют благодаря ультразвуковому датчику.

Профессии: проектировщик автомобилей, техник звукорежиссуры, звукотехник

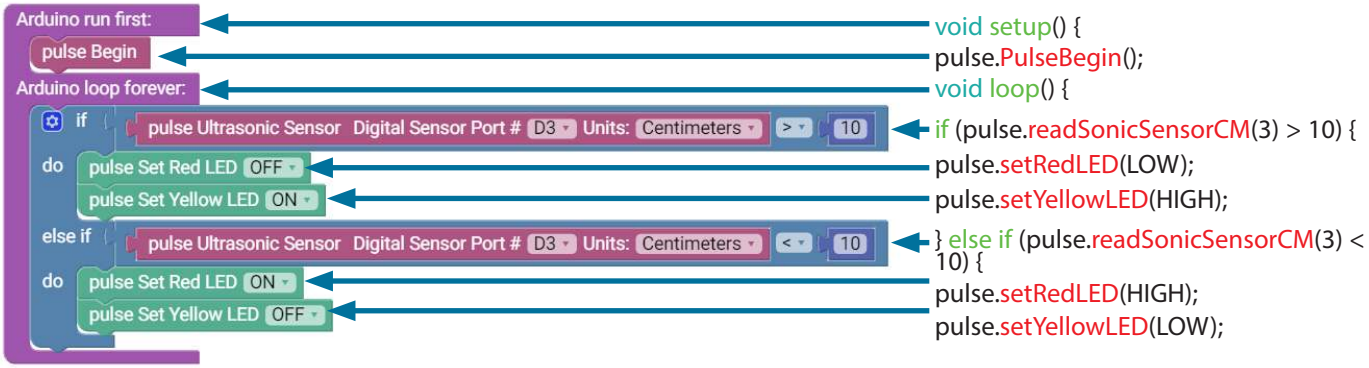
Связь с точными и естественными науками

- Физика
  - o Звуковые волны
  - o Отражение звуковых волн
- Технология
  - o Частота
  - o Оцифровка звука
- Техническое конструирование
  - o Акустические измерения
  - o Сонар
- Математика
  - o Измеритель
  - o Символы сравнения

Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

}

void loop() {
if (pulse.readSonicSensorCM(3) > 10) {
pulse.setRedLED(LOW);
pulse.setYellowLED(HIGH);
} else if (pulse.readSonicSensorCM(3) < 10) {
pulse.setRedLED(HIGH);
pulse.setYellowLED(LOW);
}
}
```

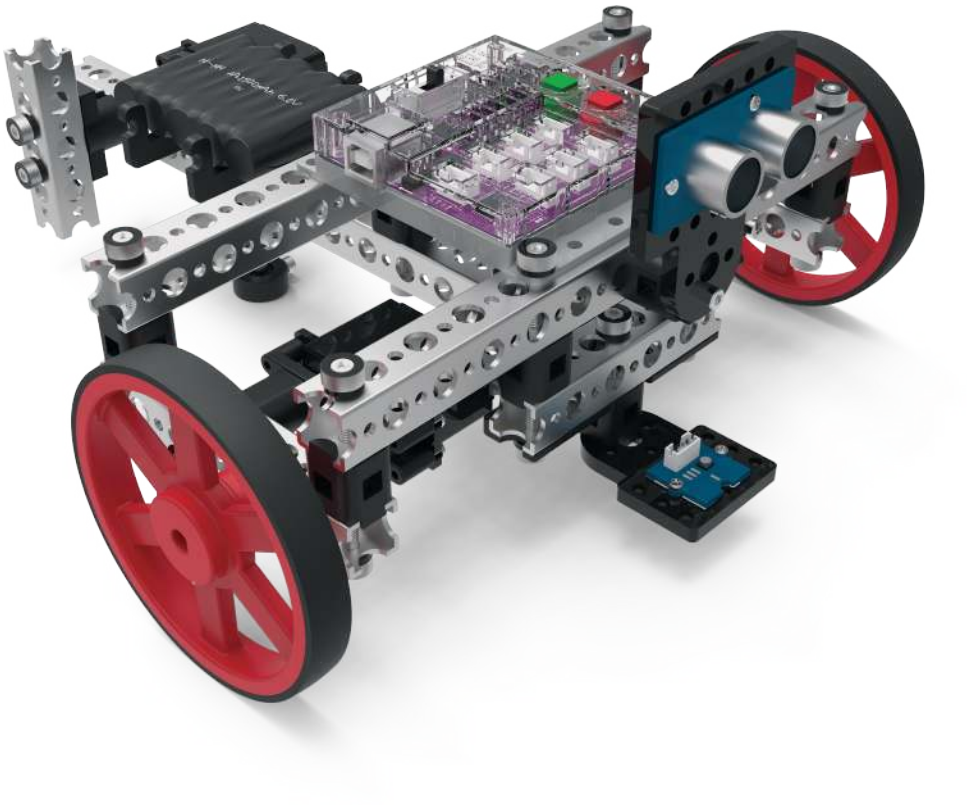
Связь блоков с текстом



Примечание: В этой программе используются математические операторы "больше" (>) и "меньше" (<). Программа управляет светодиодными индикаторами в зависимости от результата сравнения выходного значения с числом 10.

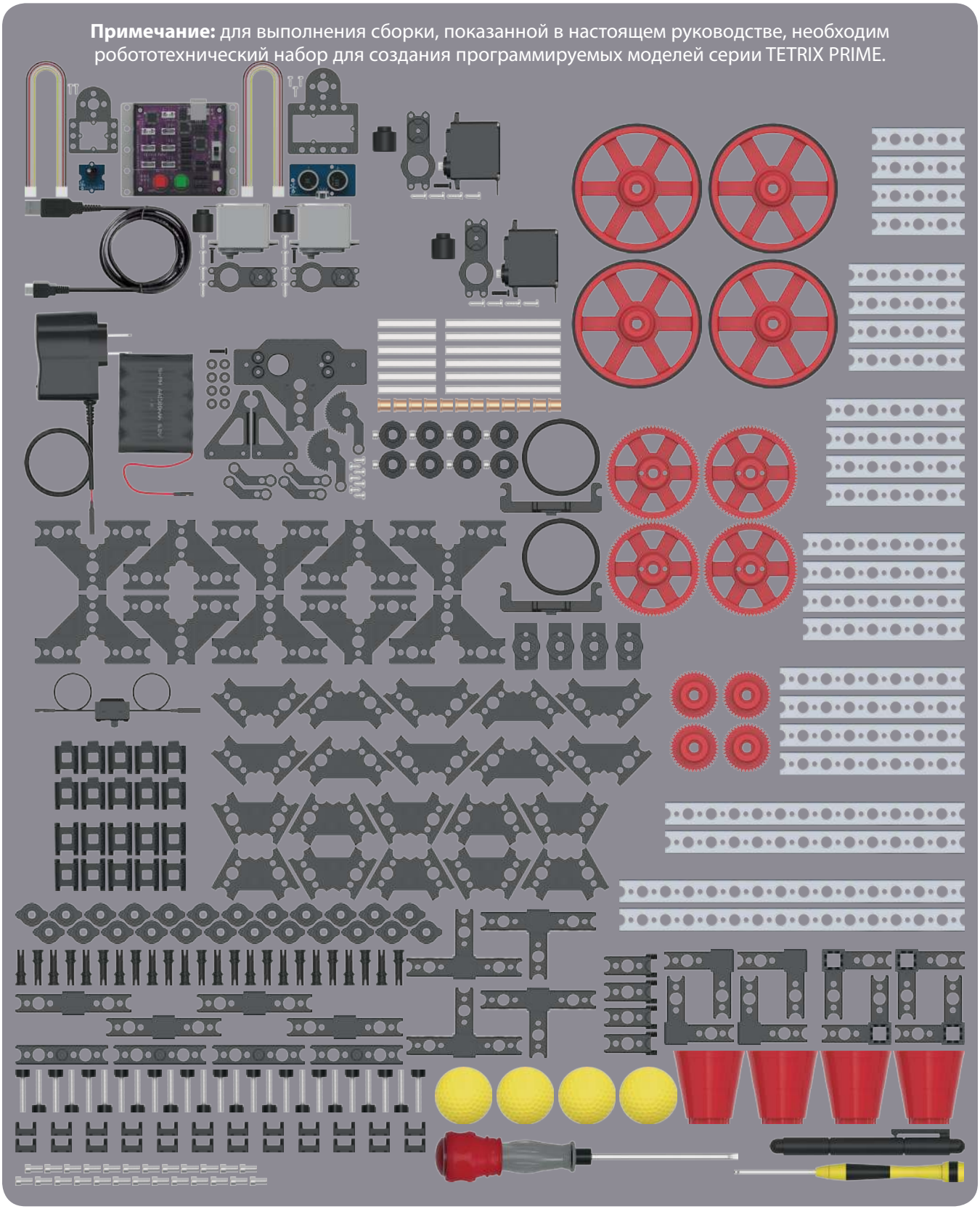
Сборка и программирование базового робота PULSE

Итак, вы готовы к переходу на следующий уровень. Приступаем к тому, чего вы так долго ждали! Вы старательно осваивали основы, и теперь настало время применить их для создания настоящего робота. В ходе следующих 10 упражнений вы научитесь собирать робота, приводить его в движение, управлять его перемещениями и устанавливать на нём датчики. И в завершение выполните упражнение, где потребуются все эти умения. Будет интересно!

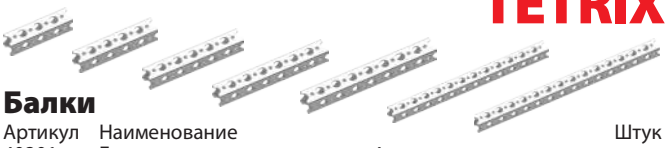


Робототехнический набор для создания программируемых моделей

Примечание: для выполнения сборки, показанной в настоящем руководстве, необходим робототехнический набор для создания программируемых моделей серии TETRIX PRIME.



Указатель деталей робототехнического набора для создания программируемых моделей серии TETRIX® PRIME



Балки

Артикул	Наименование	Штук
40201	Балка квадратного сечения с 4 отверстиями серии TETRIX PRIME	4
40202	Балка квадратного сечения с 5 отверстиями серии TETRIX PRIME	4
40203	Балка квадратного сечения с 6 отверстиями серии TETRIX PRIME	4
40204	Балка квадратного сечения с 7 отверстиями серии TETRIX PRIME	4
40205	Балка квадратного сечения с 8 отверстиями серии TETRIX PRIME	4
40206	Балка квадратного сечения с 13 отверстиями серии TETRIX PRIME	2
40207	Балка квадратного сечения с 15 отверстиями серии TETRIX PRIME	2

Вставные соединители

Артикул	Наименование	Штук
40212	Балочный соединитель, трёхнаправленный, серии TETRIX PRIME	4
40213	Т-образный балочный соединитель серии TETRIX PRIME	4
40211	Балочный соединитель для скрепления деталей под углом 90 градусов серии TETRIX PRIME	4
40214	Торцевой балочный соединитель серии TETRIX PRIME	4
40322	Соединитель для удлинения балок серии TETRIX PRIME	4
40215	Прямой соединитель для балок серии TETRIX PRIME	4

Наружные соединители

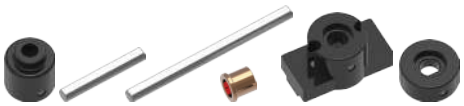
Артикул	Наименование	Штук
40208	Скоба для соединения деталей под углом 90 градусов серии TETRIX PRIME	10
40209	Скоба для соединения деталей под углом 60 градусов серии TETRIX PRIME	10
40210	Скоба для Т-образного соединения деталей серии TETRIX PRIME	10
40216	Блочный соединитель для параллельного крепления балок серии TETRIX PRIME	10
40217	Блочный соединитель для крестообразного крепления под углом 90 градусов серии TETRIX PRIME	10

Скобы и крепеж

Артикул	Наименование	Штук
40219	Втулка быстросменной заклепки серии TETRIX PRIME	24
40220	Штифт быстросменной заклепки серии TETRIX PRIME	24
40221	Барашковая гайка серии TETRIX PRIME	24
40323	Винт с рифлёной головкой серии TETRIX PRIME	24
40516	Винт с углублением под ключ	25

Колеса, шестерни, сервоприводы

Артикул	Наименование	Штук
40222	Колесо с шиной серии TETRIX PRIME	4
40223	Пластмассовая шестерня с 40 зубьями серии TETRIX PRIME	4
40224	Пластмассовая шестерня с 80 зубьями серии TETRIX PRIME	4
40538	Сервопривод HiTec HS322-HD	2
44298	Электродвигатель постоянного тока серии TETRIX PRIME	2
40232	Скоба крепления сервопривода серии TETRIX PRIME	6



Комплектующие колёс, шестерён и сервоприводов

Артикул	Наименование	Штук
40230	Ступица вала сервопривода серии TETRIX® PRIME	6
40225	Стальная ось 80 мм серии TETRIX® PRIME	6
40226	Стальная ось 40 мм серии TETRIX® PRIME	6
40227	Бронзовая втулка 8 x 6 мм серии TETRIX PRIME	12
40228	Ступица для крепления к балке серии TETRIX® PRIME	4
40229	Установочное кольцо вала с лыской серии TETRIX® PRIME	8

Захватное устройство в сборе

Артикул	Наименование	Штук
40234	Захватное устройство в комплекте серии TETRIX PRIME	1

Электронное оборудование и средства управления

Артикул	Наименование	Штук
44268	Робототехнический контроллер с USB-кабелем серии TETRIX PULSE	1
43055	Комплект ультразвукового датчика	1
43056	Комплект датчика линии	1

Аккумуляторы и комплектующие

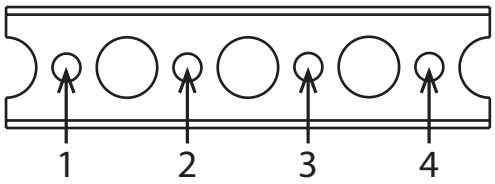
Артикул	Наименование	Штук
40236	Скоба крепления аккумуляторной батареи серии TETRIX PRIME	2
40235	Аккумуляторная батарея NiMH 6 В серии TETRIX® PRIME	1
40378	Зарядное устройство для аккумуляторной батареи NiMH из 5 элементов серии TETRIX® PRIME	1
36404	Отвертка 4-в-1	1
40341	Миниатюрная отвертка с шарообразным шестигранным наконечником	1
42991	Отвертка 2-в-1	1
41769	Пластиковые стаканчики на 56 мл	4
14041	Мячики для гольфа	4
44301	Руководство по программированию робототехнического контроллера серии TETRIX PULSE	1

Элементы конструктора серии TETRIX PRIME

Балки

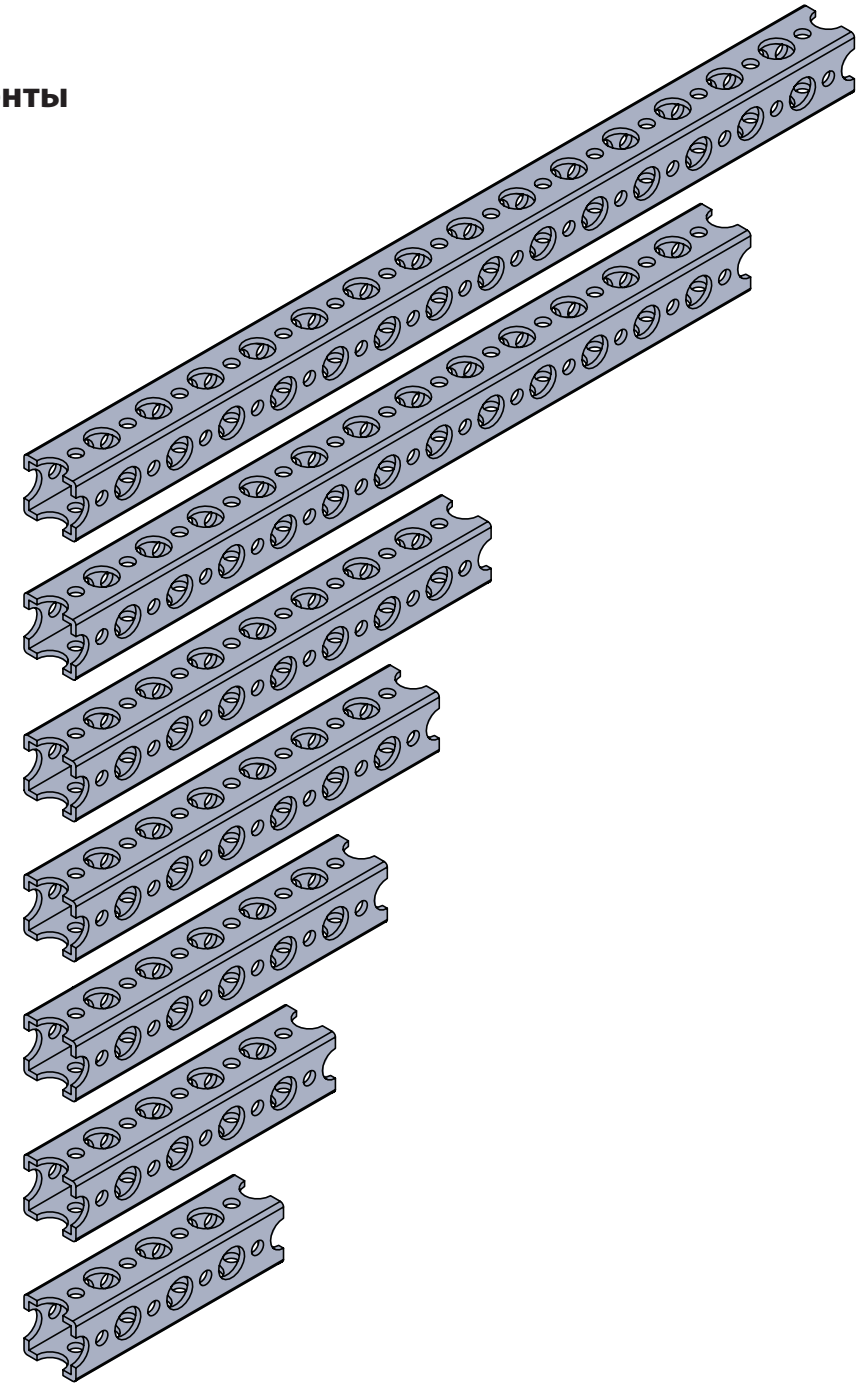
Каждая балка получает обозначение по числу малых отверстий на одной из её сторон.
Не выбирайте балки по числу больших отверстий (см. схему справа).

Масштаб 1:1



Чтобы узнать наименование балок квадратного сечения TETRIX PRIME, подсчитайте в них малые отверстия. Выше дан пример балки с четырьмя отверстиями.

Конструктивные элементы



Балка квадратного сечения с 15 отверстиями 40207

Балка квадратного сечения с 13 отверстиями 40206

Балка квадратного сечения с 8 отверстиями 40205

Балка квадратного сечения с 7 отверстиями 40204

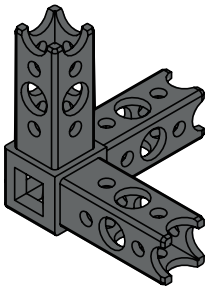
Балка квадратного сечения с 6 отверстиями 40203

Балка квадратного сечения с 5 отверстиями 40202

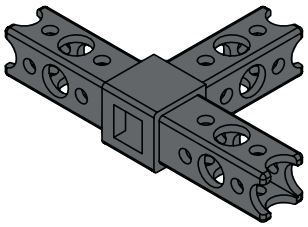
Балка квадратного сечения с 4 отверстиями 40201

Конструктивные элементы

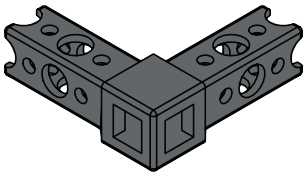
Балочный соединитель, трёхнаправленный 40212



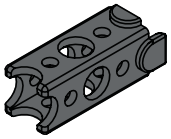
Т-образный балочный соединитель 40213



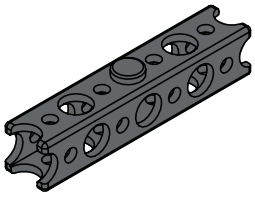
Балочный соединитель для скрепления деталей под углом 90 градусов 40211



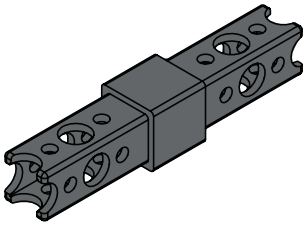
Торцевой балочный соединитель 40214



Соединитель для удлинения балок 40322

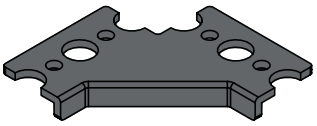


Прямой соединитель для балок 40215

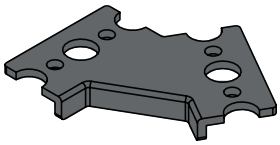


Конструктивные элементы

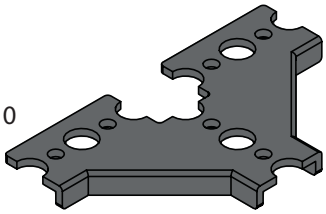
Скоба для соединения деталей под углом 90 градусов 40208



Скоба для соединения деталей под углом 60 градусов 40209



Скоба для Т-образного соединения деталей 40210



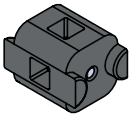
Втулка быстросменной заклепки 40219

Штифт быстросменной заклепки 40220

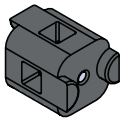
Барашковая гайка 40221

Винт с рифлёной головкой 40323

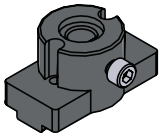
Блочный соединитель для параллельного крепления балок 40216



Блочный соединитель для крестообразного крепления под углом 90 градусов 40217



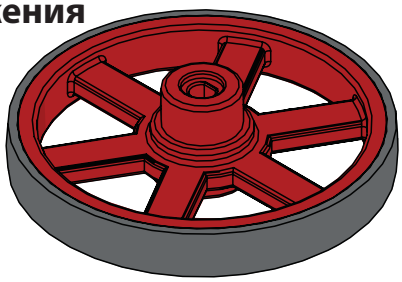
Ступица для крепления к балке 40228



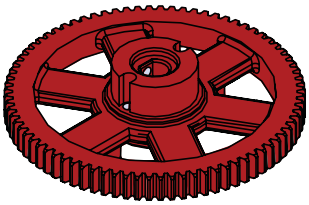
СОВЕТ: Эти две детали, блочный соединитель для параллельного крепления балок и балочный соединитель для крестообразного крепления под углом 90 градусов, очень похожи по внешнему виду и их легко перепутать. Рекомендуется тщательно проверить, та ли это деталь, которая указана в инструкции.

Детали механизмов движения

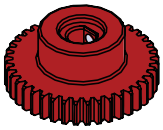
Колесо с шиной 40222



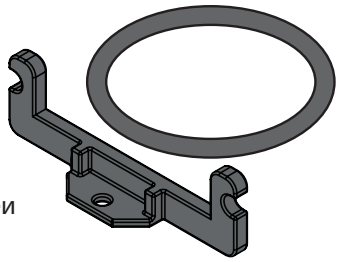
Пластмассовая шестерня с 80 зубьями 40224



Пластмассовая шестерня с 40 зубьями 40223

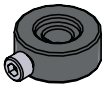


Скоба крепления аккумуляторной батареи 40236



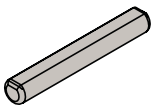
Бронзовая втулка 8 x 6 мм 40227

Установочное кольцо вала с лыской 40229



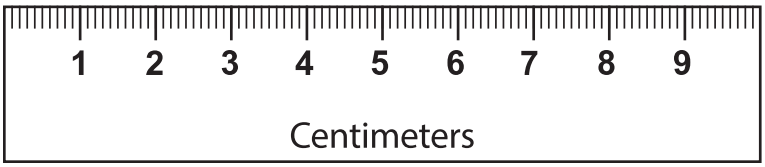
Стальная ось 80 мм 40225

Стальная ось 40 мм 40226



Винт с углублением под ключ 40516

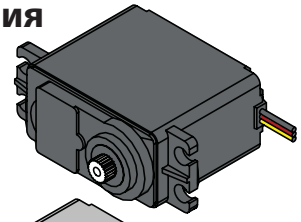
Длину стальных осей 80 мм и 40 мм измерить здесь.



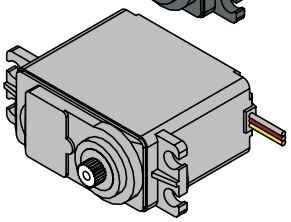
Примечание:
10 мм = 1 см

Детали механизмов движения

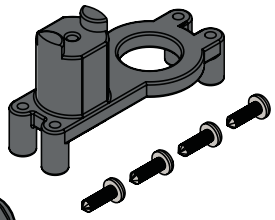
Сервопривод
HiTec HS322-HD 40538



Электродвигатель
постоянного тока серии
TETRIX PRIME 44298

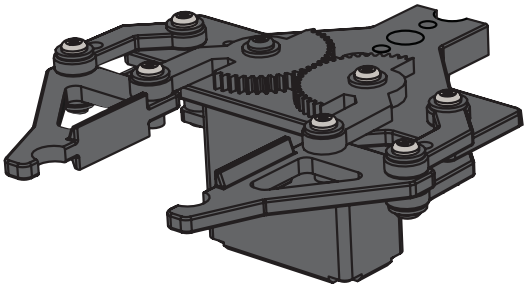


Скоба крепления
сервоприводов 40232



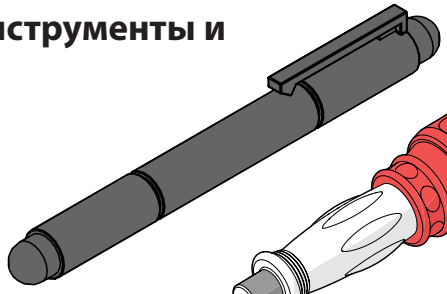
Ступица вала
сервопривода 40230

Захватное устройство в
комплекте 40234



Элементы питания, инструменты и комплектующие

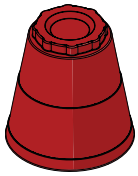
Отвертка 4-в-1 36404



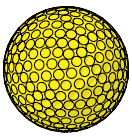
Отвертка 2-в-1 42991

Миниатюрная отвертка с шарообразным шестигранным наконечником 40341

Пластиковые стаканчики на 56 мл 41769

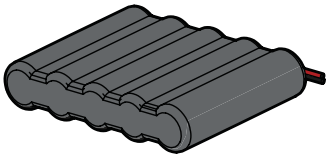


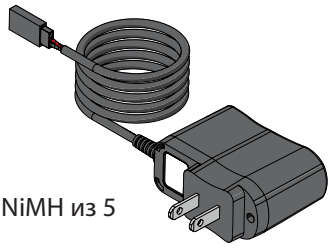
Мячики для гольфа 14041



Элементы питания, инструменты и комплектующие

Аккумуляторная батарея NiMH 6 В 40235

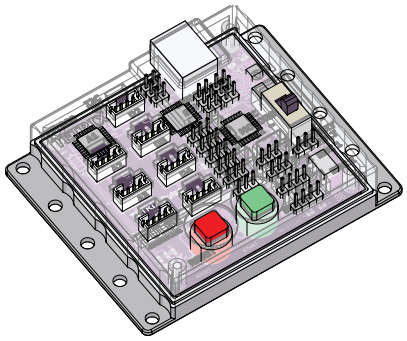




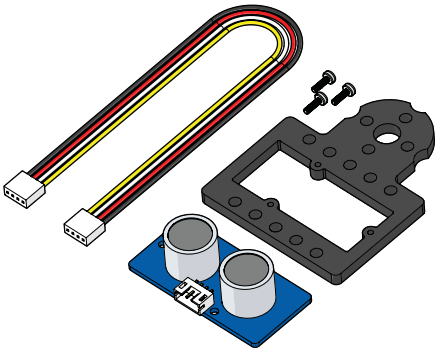
Зарядное устройство для аккумуляторной батареи NiMH из 5 элементов 40378

Элементы управления

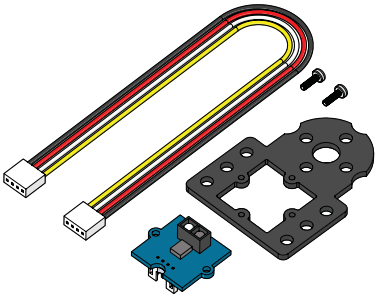
Робототехнический контроллер TETRIX PULSE 44268



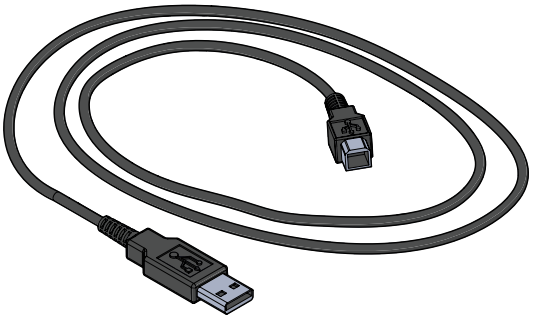
Комплект ультразвукового датчика 43055



Комплект датчика линии 43056



Кабель USB с разъёмами типа A-B, 40967

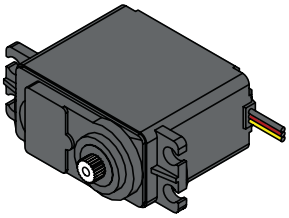


Сборка стандартного сервопривода

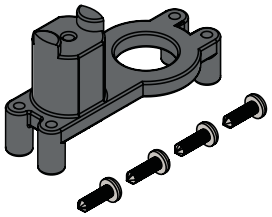
Вам потребуется сервопривод HiTec HS322-HD с винтом, скоба крепления сервоприводов с винтами, ступица вала сервопривода и винт с углублением под ключ. Также понадобится отвертка 4-в-1 и миниатюрная отвертка с шарообразным шестигранным наконечником. Снимите белую пластмассовую накладку, прикреплённую к сервоприводу. Винт сохраните — он ещё пригодится, а белая пластмассовая накладка больше не понадобится. Наклейте ярлык с обозначением стандартного сервопривода на сервопривод. Соберите один из стандартных сервоприводов, как показано на рисунке. Второй стандартный сервопривод будет использоваться для захвата.

Стандартные сервоприводы используются для пропорционального вращения, а также для захватов, рулевого управления и позиционирования.

Необходимое оборудование



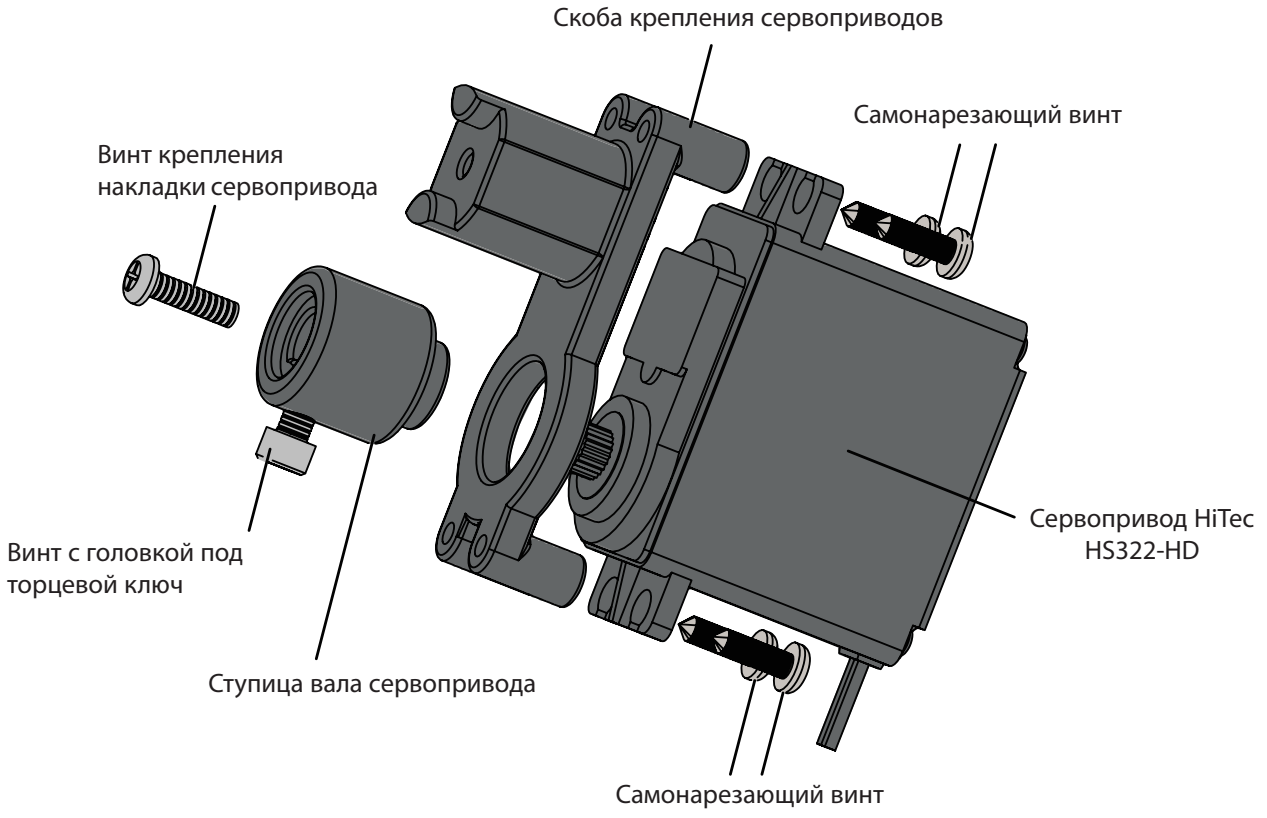
1x
сервопривод HiTec
HS322-HD 40538



1x
скоба крепления
сервоприводов
40232

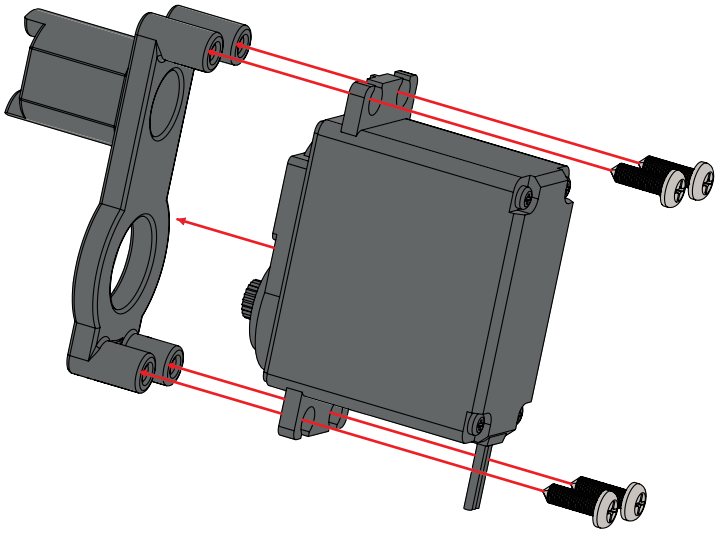
1x
ступица вала
сервопривода
40230

1x
винт с углублением
под ключ 40516

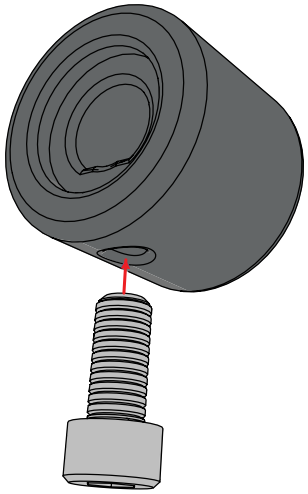


СОВЕТ: Сервопривод продолжительного вращения устанавливается на скобе крепления сервопривода таким же образом, как и стандартный сервопривод, однако без необходимости центрировать накладку сервопривода.

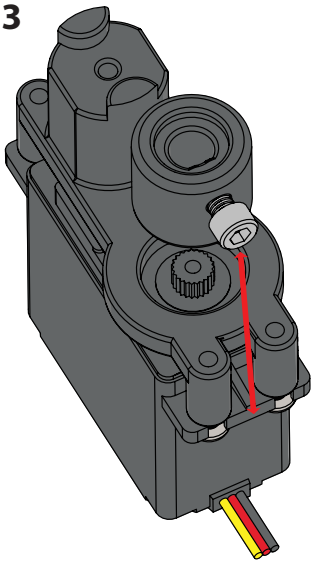
Шаг 1



Шаг 2

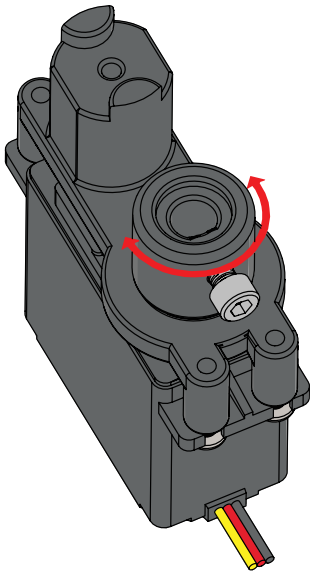


Шаг 3

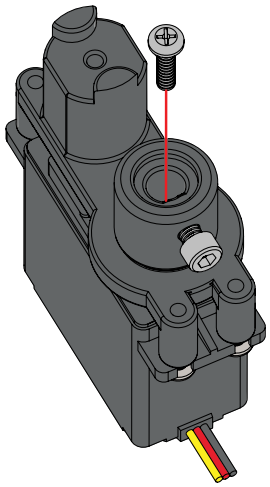


Шаг 4

Перед установкой ступицы вала сервопривода необходимо убедиться, что сервопривод находится в нейтральном положении. Для этого подсоедините сервопривод и аккумуляторную батарею к приёмнику и включите пульт управления. Переведите оба джойстика и регуляторы точной настройки в центральное (нейтральное) положение, при этом сервопривод переместится в нейтральное положение.



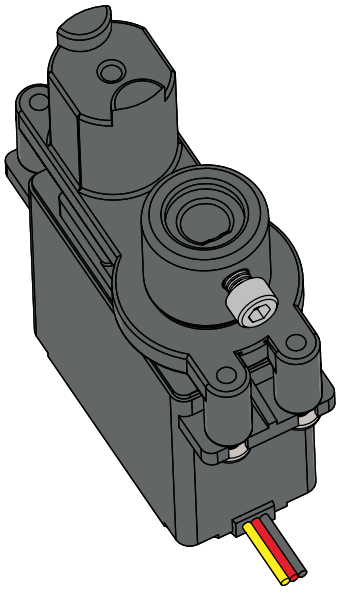
Шаг 5



Совместите шлицы на ступице со шлицами на валу сервопривода и нажатием соедините детали. Установочный винт должен максимально точно совпасть с центральной осью корпуса сервопривода. Занятие винт, удерживая ступицу в правильном положении.

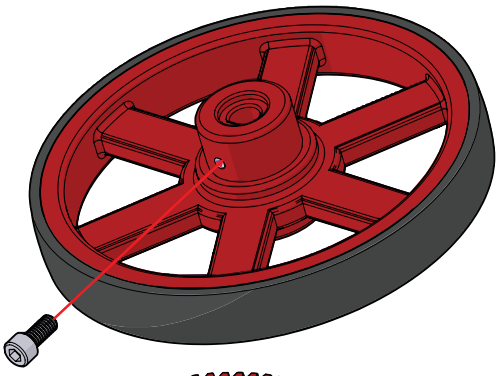
При помощи передатчика ПДУ проверьте работу сервопривода. Если сервопривод функционирует надлежащим образом, отсоедините аккумуляторную батарею от приёмника и сервопривода. Теперь сервоприводы готовы к работе.

Готовая сборка

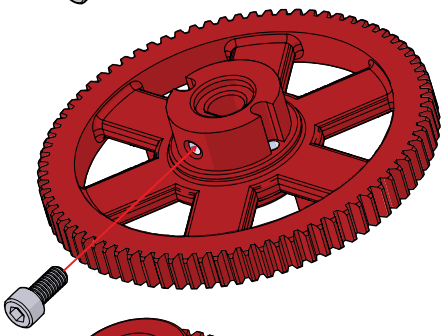


Применение установочного винта

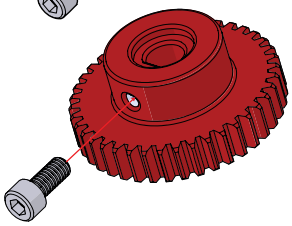
Колесо с шиной 40222



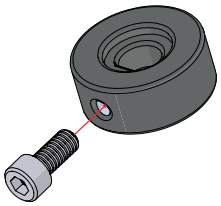
Пластмассовая шестерня с 80 зубьями 40224



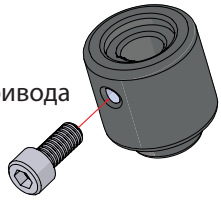
Пластмассовая шестерня с 40 зубьями 40223



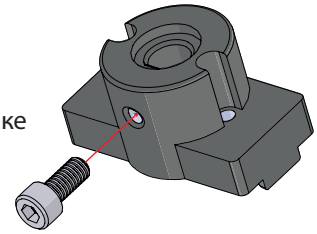
Установочное кольцо вала с лыской 40229



Ступица вала сервопривода 40230



Ступица для крепления к балке 40228



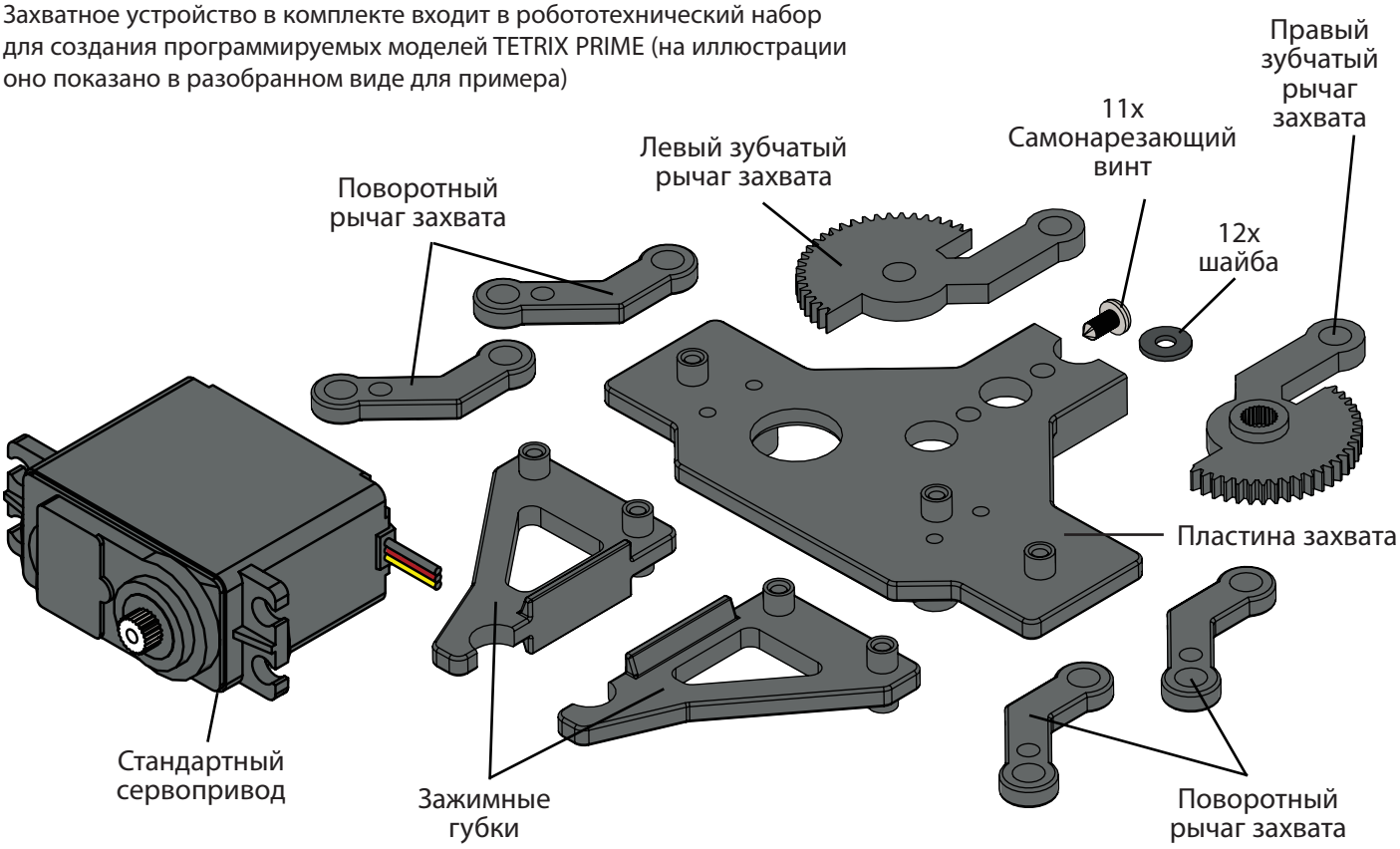
СОВЕТ: Если винты с головкой под торцевой ключ уже установлены на детали, можно пропустить инструкции по их установке и перейти к следующему шагу.

СОВЕТ: В инструкциях детали данного вида показаны в нужном положении. После установки деталей не забудьте затянуть винты с головкой под торцевой ключ перед выполнением следующего шага.

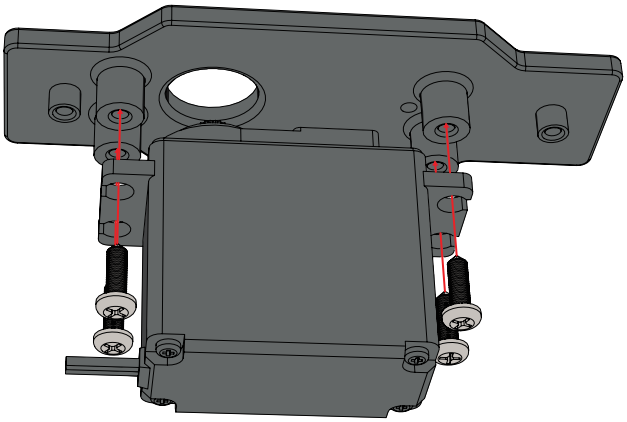
СОВЕТ: Винты с головкой под торцевой ключ должны быть затянуты плотно, но не перетянуты. Избыточная затяжка винтов может привести к повреждению деталей.

Захватное устройство в сборе

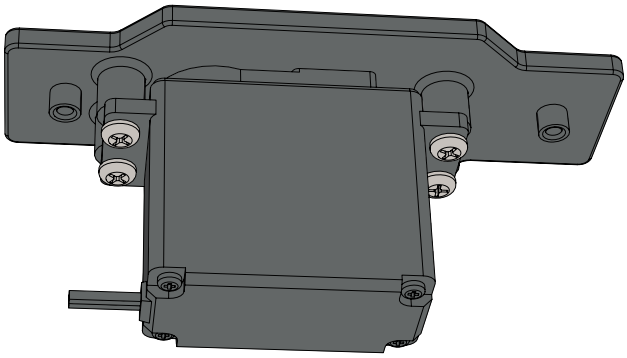
Захватное устройство в комплекте входит в робототехнический набор для создания программируемых моделей TETRIX PRIME (на иллюстрации оно показано в разобранном виде для примера)



Шаг 1

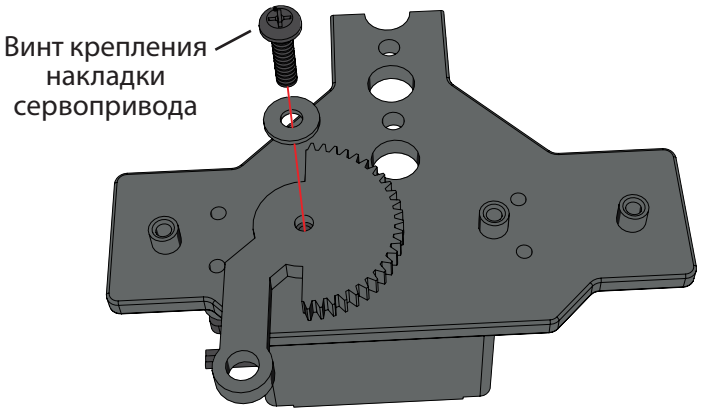


Шаг 2

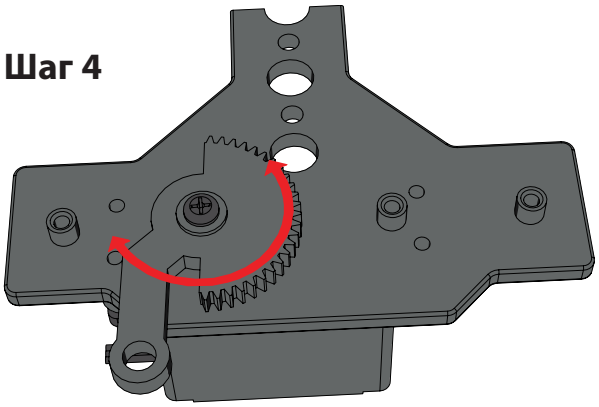


ВАЖНО: Перед установкой на стандартный сервопривод правого зубчатого рычага необходимо прикрепить стандартный сервопривод к пластине захватного устройства. Далее подключите стандартный сервопривод и приведите его в нейтральное положение, чтобы расположить зубчатые секторы, как показано на иллюстрациях к следующим шагам. Указания по установке деталей конструктора TETRIX PRIME также представлены по адресу TETRIXrobotics.com.

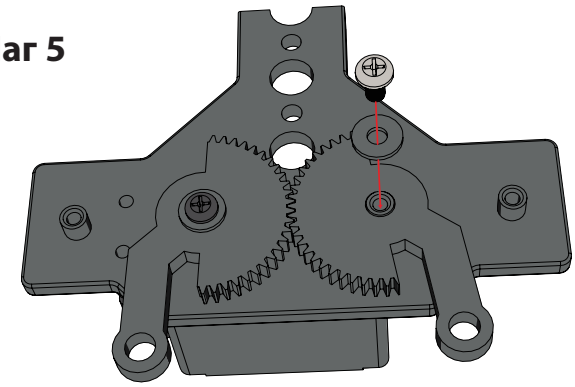
Шаг 3



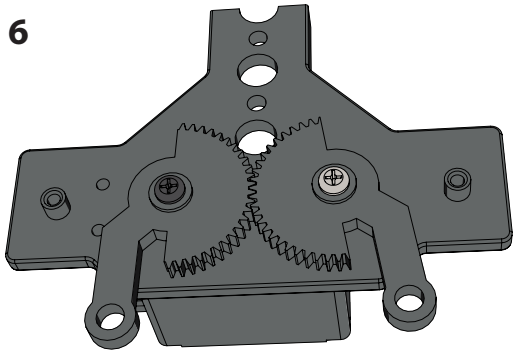
Шаг 4



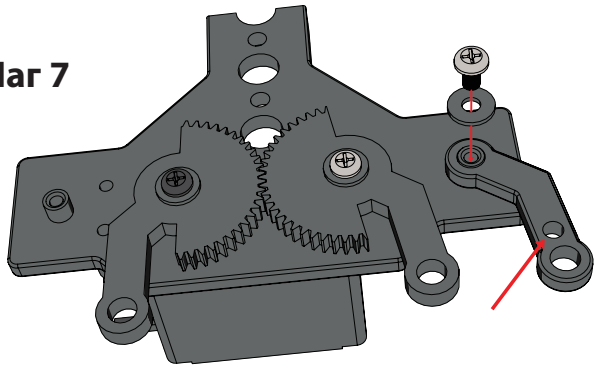
Шаг 5



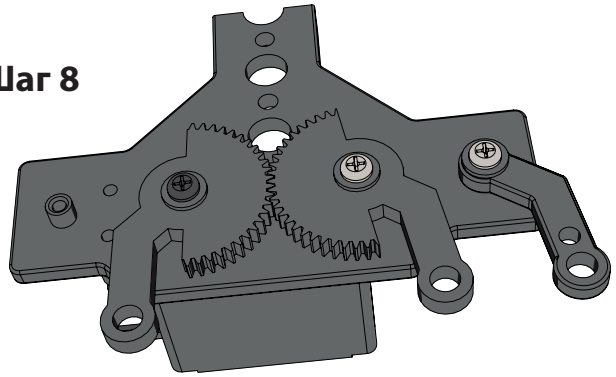
Шаг 6



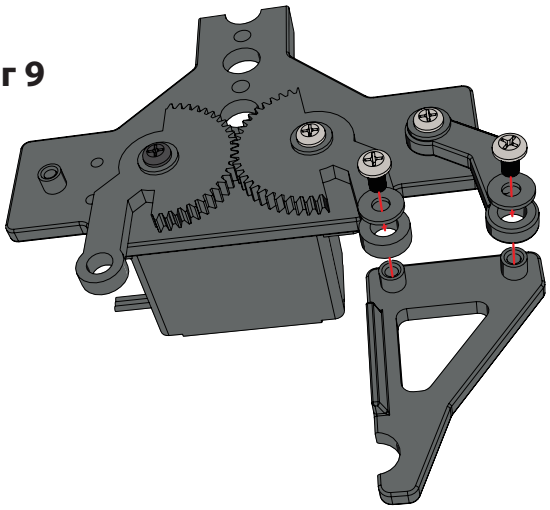
Шаг 7



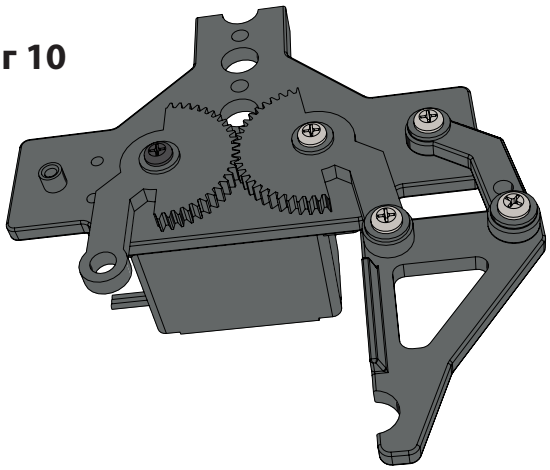
Шаг 8



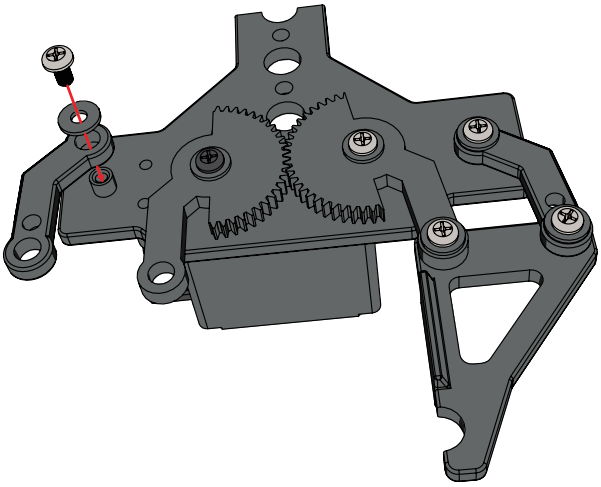
Шаг 9



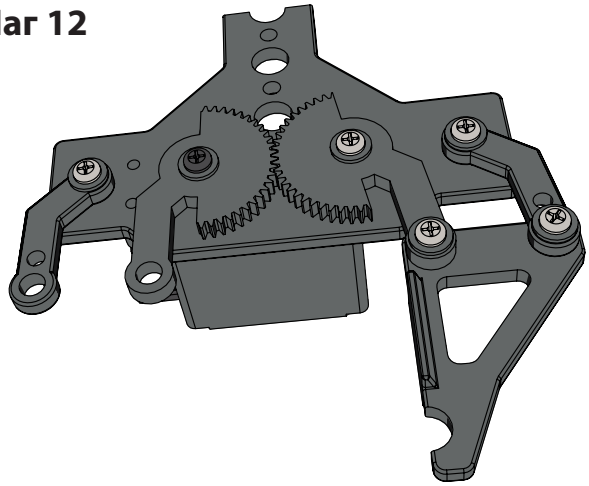
Шаг 10



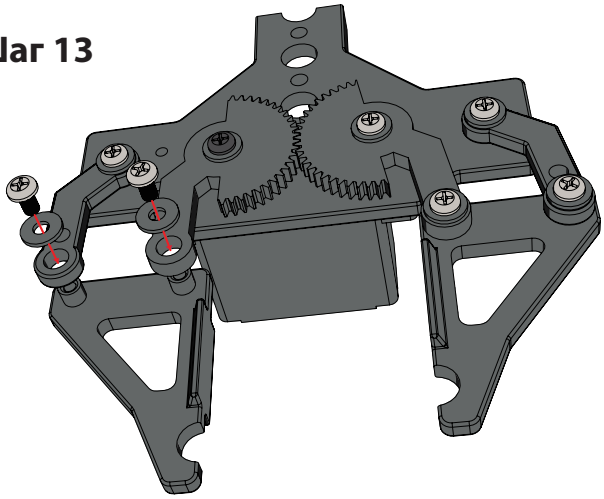
Шаг 11



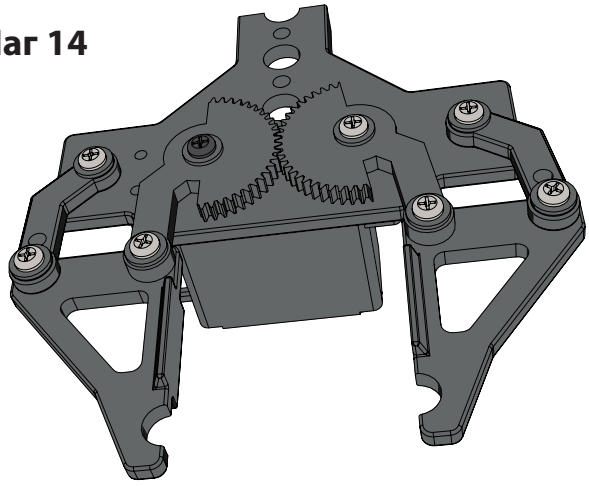
Шаг 12



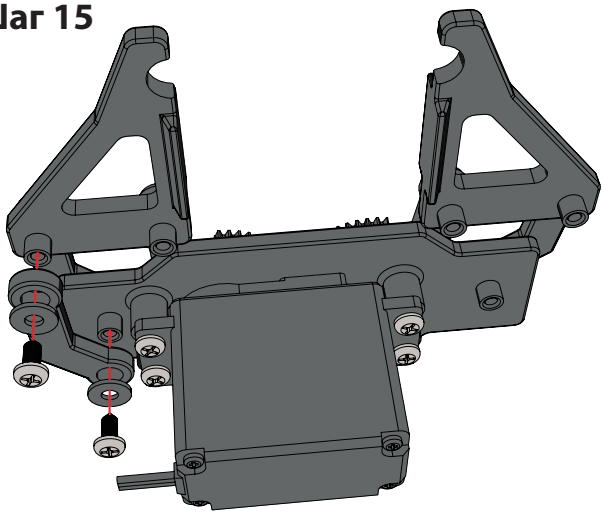
Шаг 13



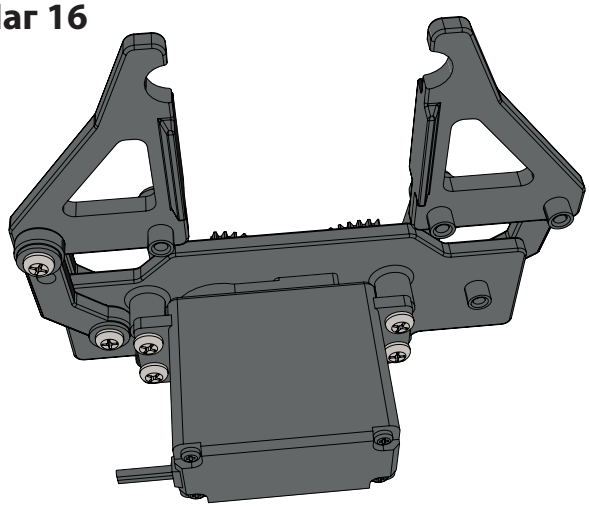
Шаг 14



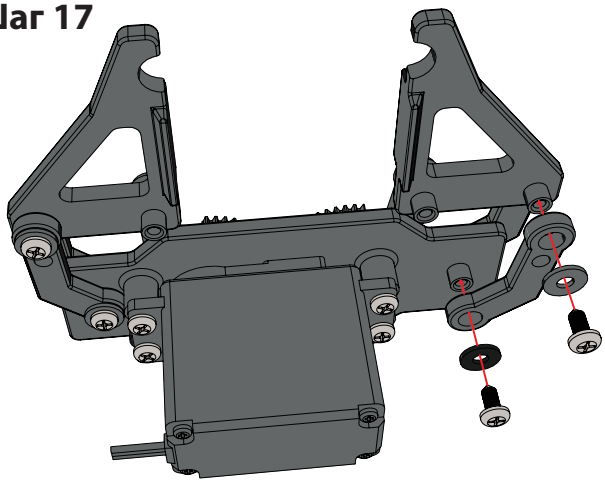
Шаг 15



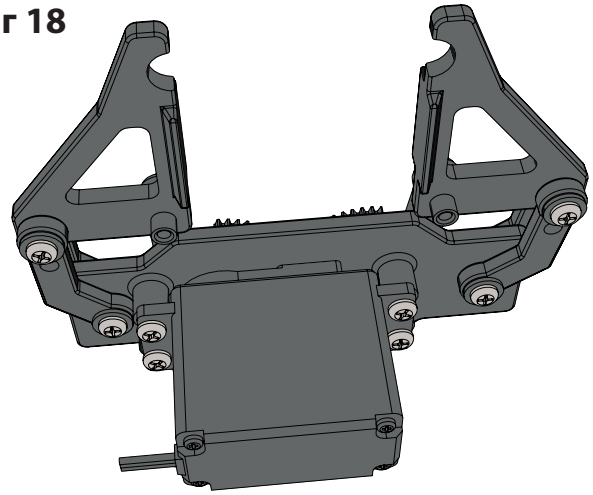
Шаг 16



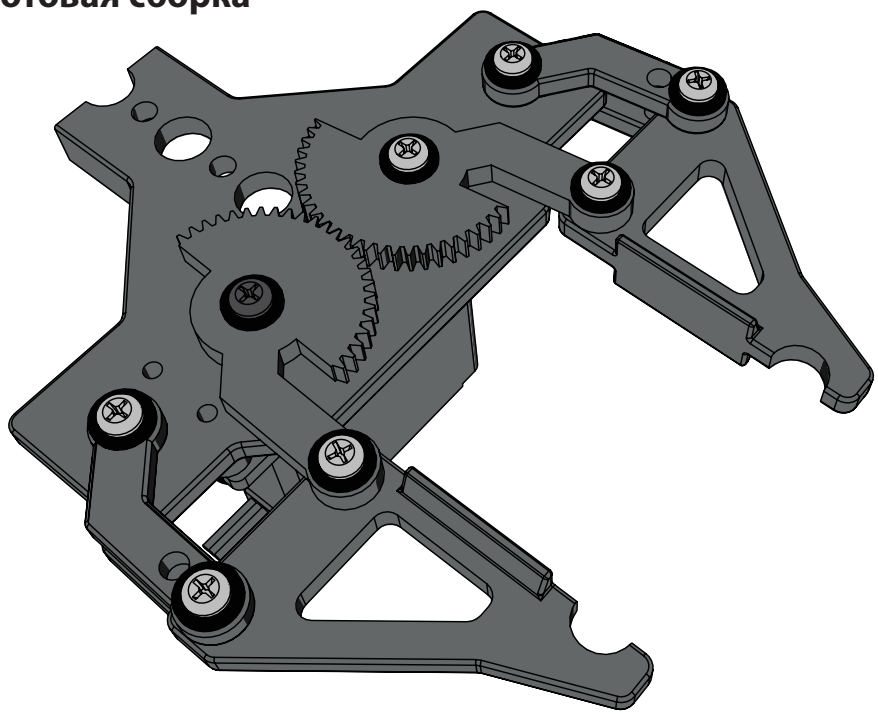
Шаг 17



Шаг 18

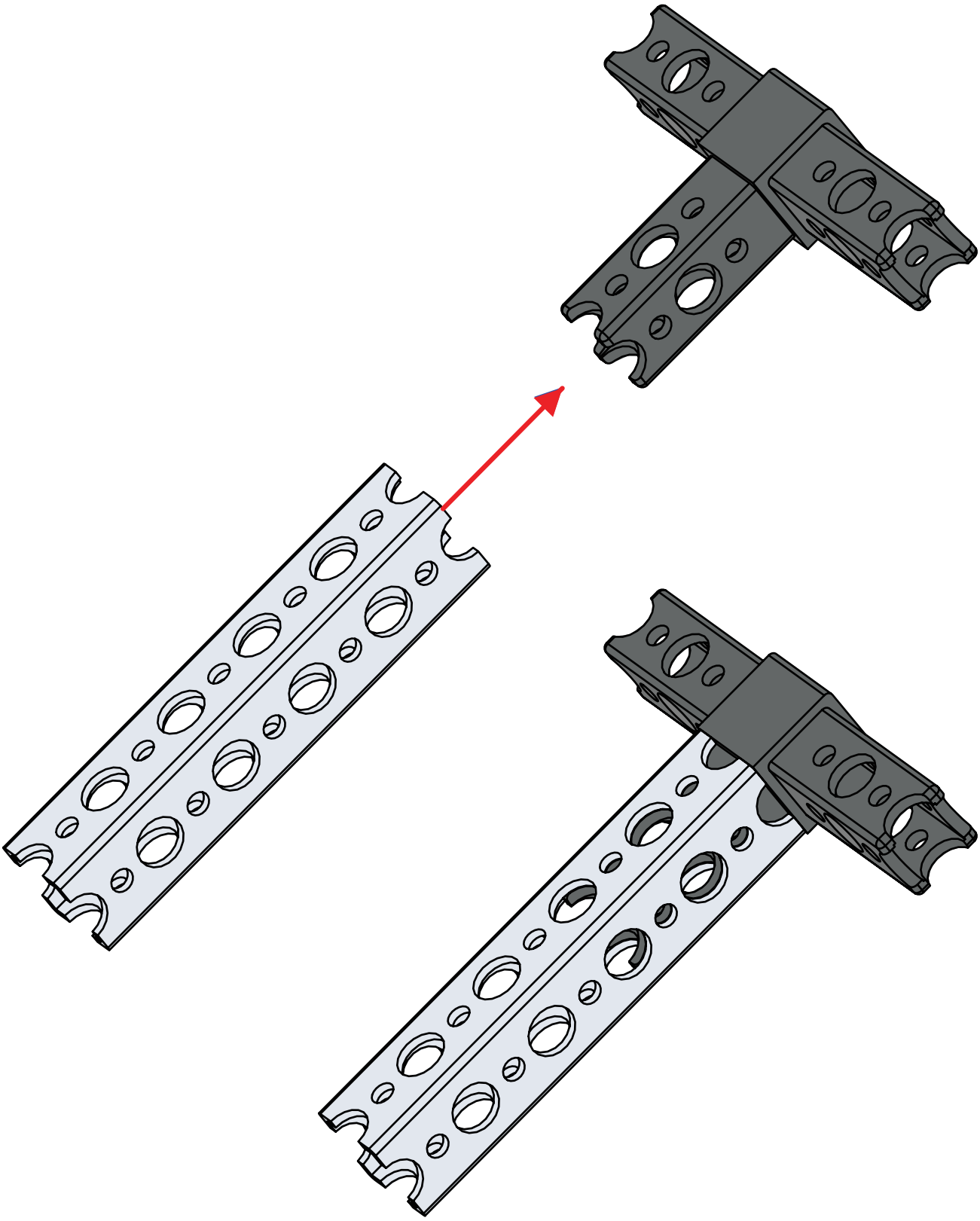


Готовая сборка

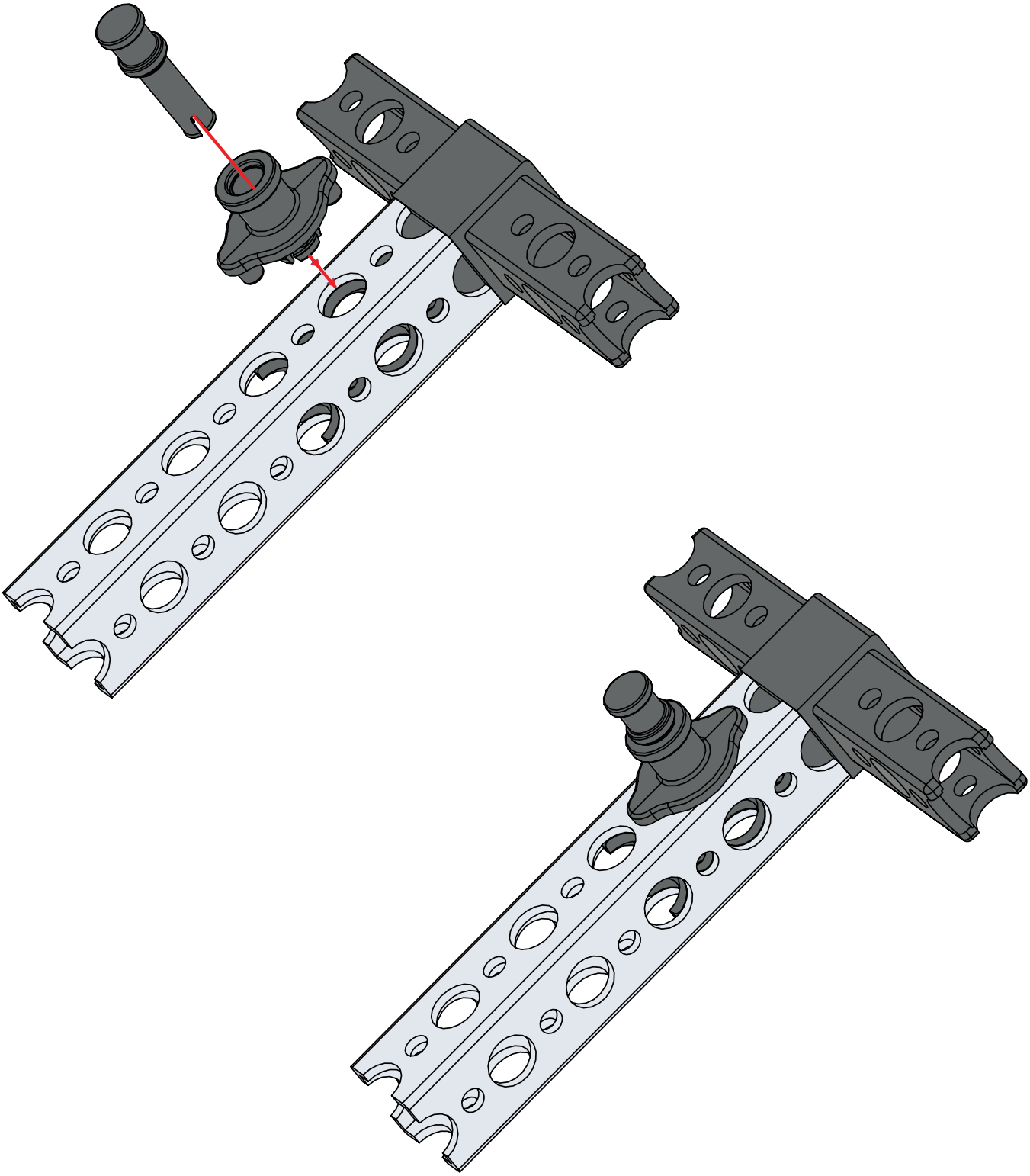


Советы по сборке конструкций

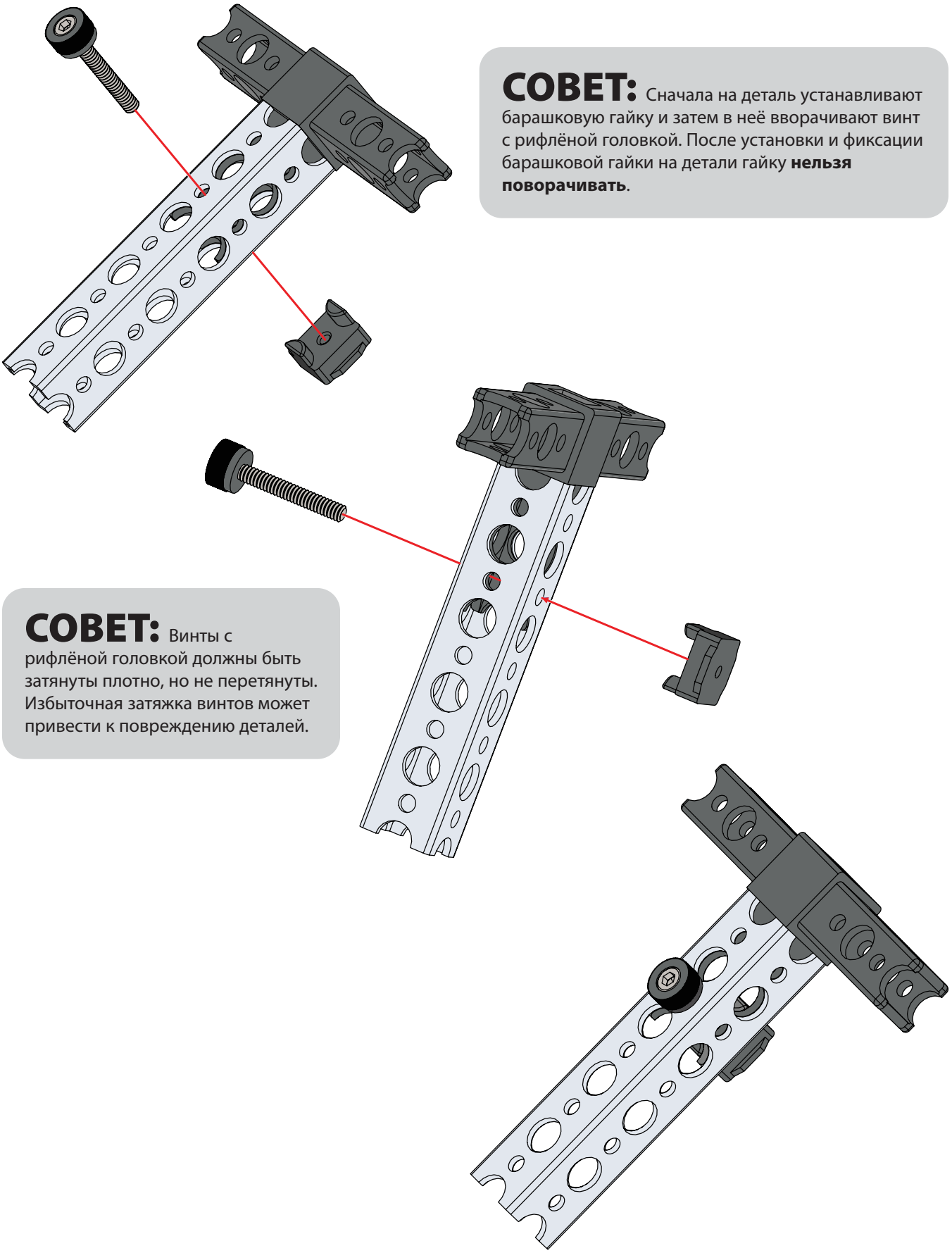
Соединители вставляются в балки и позволяют скреплять детали по трем осям, Т-образно, под углом 90 градусов, торцами, с удлинением и параллельно.



Для быстрого закрепления соединителей можно использовать втулки и штифты быстросменных заклепок. С нажимом вставьте втулку заклепки в нужное место на балке и с помощью штифта раздвиньте заклепку в отверстии, чтобы закрепить место соединения. Установка быстросменных заклепок с обеих сторон позволит повысить прочность соединения.



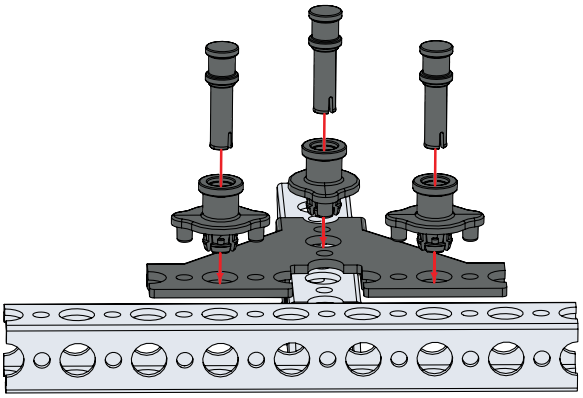
Еще прочнее места скрепления балок и соединителей можно сделать при помощи винта с рифлёной головкой и барашковой гайки.

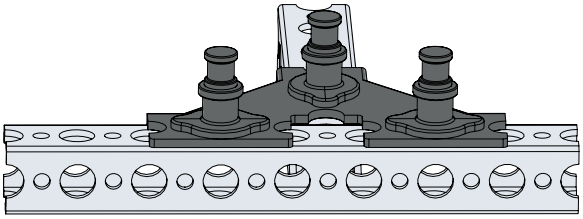


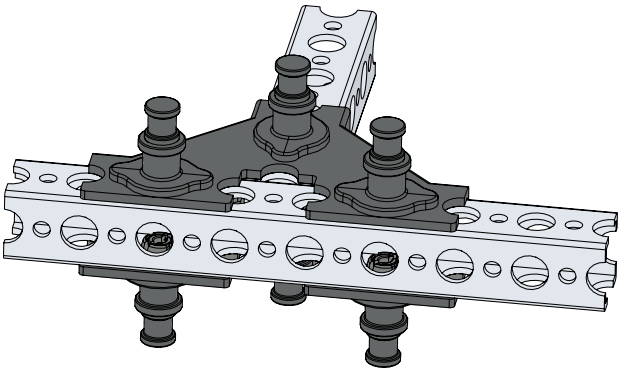
СОВЕТ: Сначала на деталь устанавливают барашковую гайку и затем в неё вворачивают винт с рифлёной головкой. После установки и фиксации барашковой гайки на детали гайку **нельзя поворачивать**.

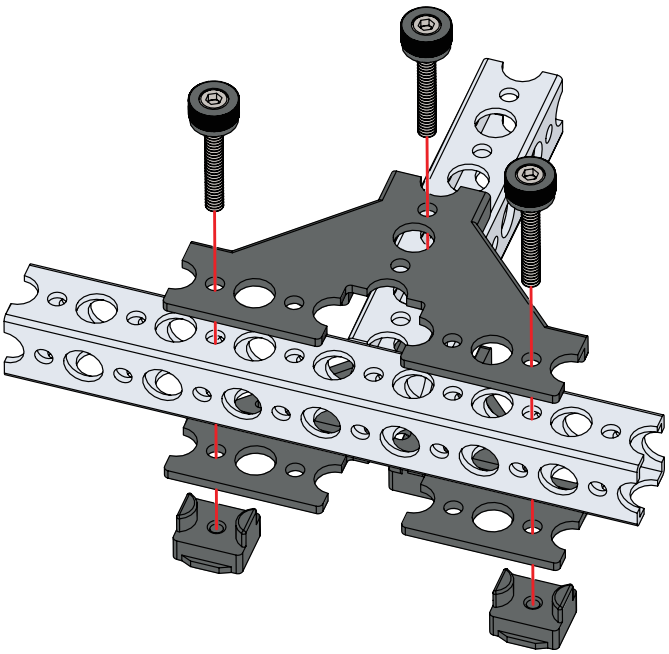
СОВЕТ: Винты с рифлёной головкой должны быть затянуты плотно, но не перетянуты. Избыточная затяжка винтов может привести к повреждению деталей.

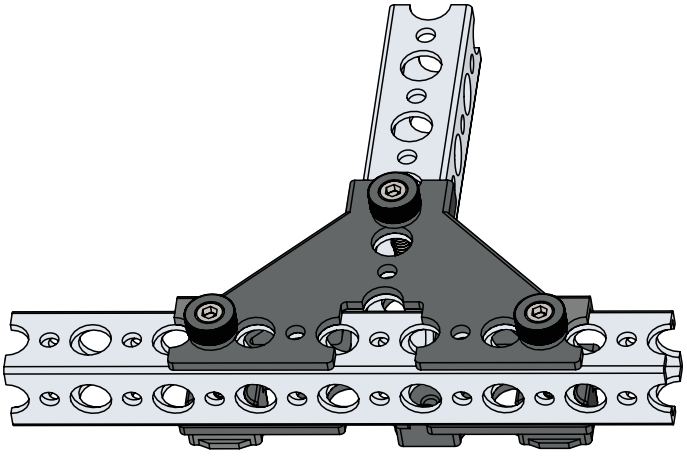
Для соединения балок можно использовать и скобы. Имеющиеся скобы позволяют соединять детали Т-образно, под углом 60 градусов или под углом 90 градусов. Скобы следует использовать попарно, устанавливая по две скобы на противоположных сторонах балки. Скобы крепятся при помощи быстросменных заклепок со штифтами, либо при помощи винтов с рифлёной головкой и барашковых гаек.



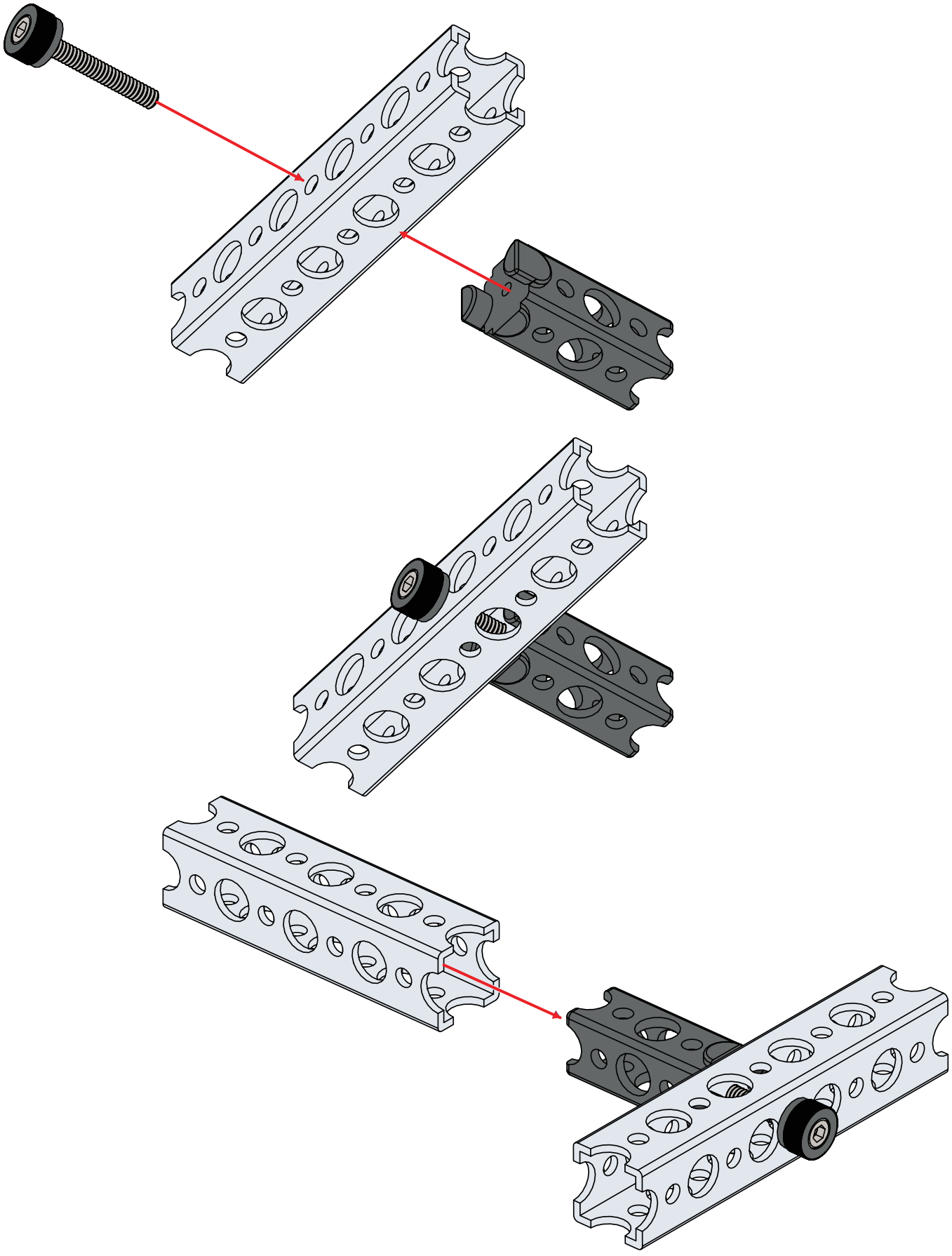




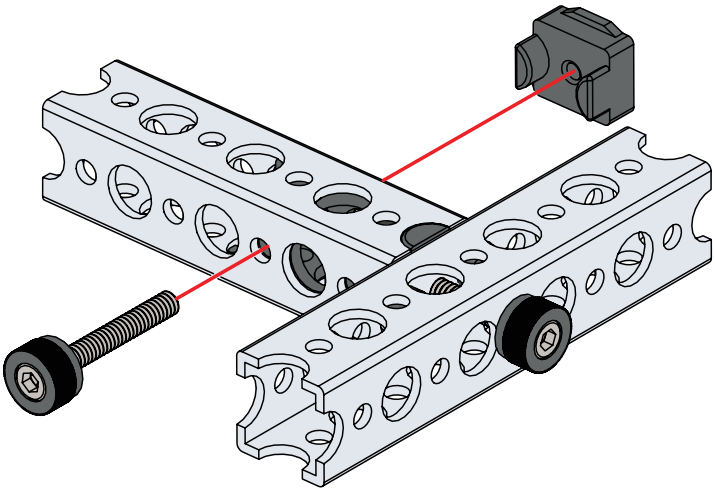


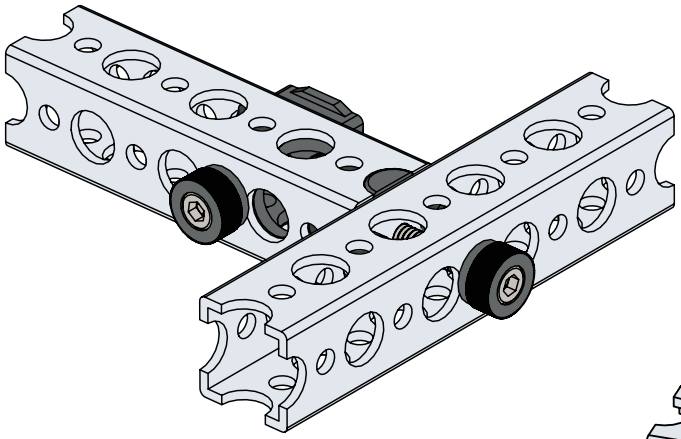


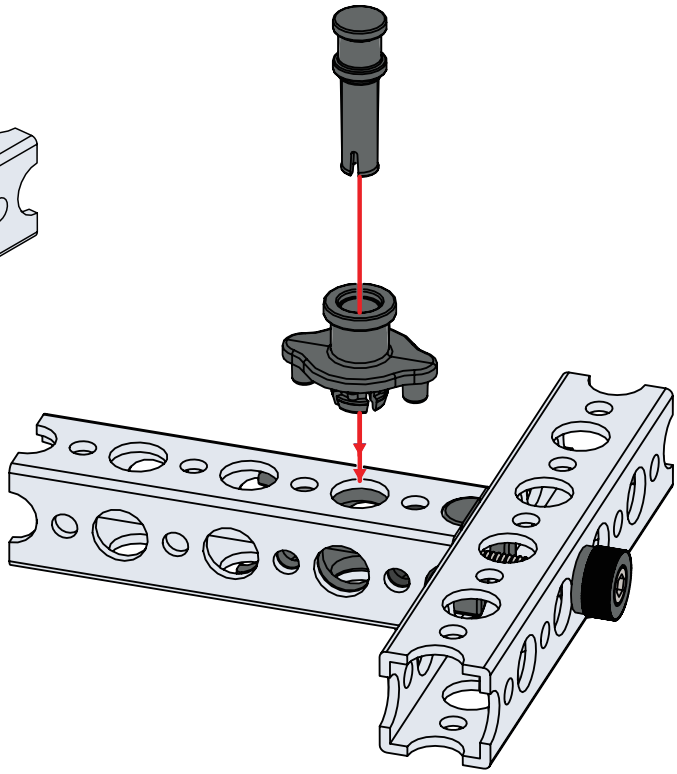
Торцевые балочные соединители, блочные соединители для параллельного крепления, а также блочные соединители для крестообразного крепления под углом 90 градусов закрепляют при помощи винта с рифлёной головкой, который необходимо пропустить через балку и ввернуть в соединитель.

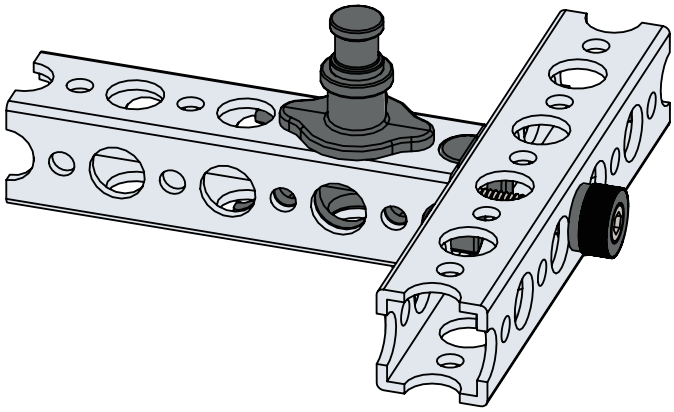


Закрепив торец соединителя винтом с рифлёной головкой, поперечную балку крепят быстросменной заклепкой со штифтом, либо винтом с рифлёной головкой и барашковой гайкой.

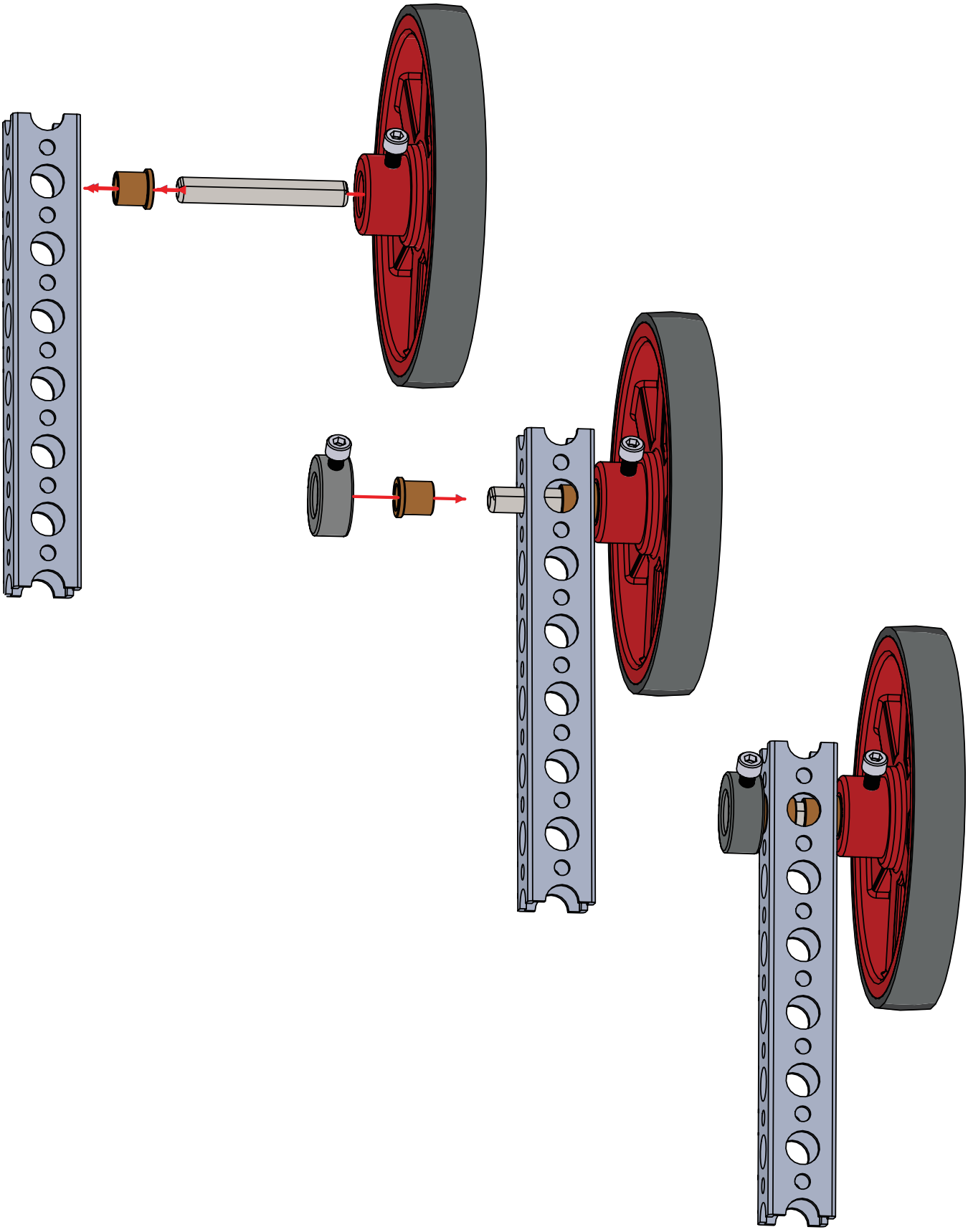




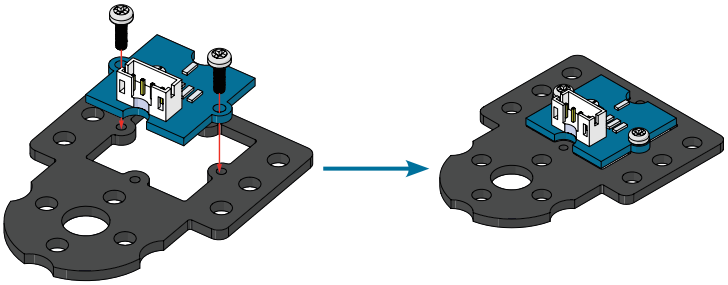




При использовании ось всегда должна иметь две точки опоры. Установите по бронзовой втулке на противоположных сторонах балки и пропустите ось через втулки. Закрепите ось на установочном кольце вала с лыской, колесе, шестерне или ступице.

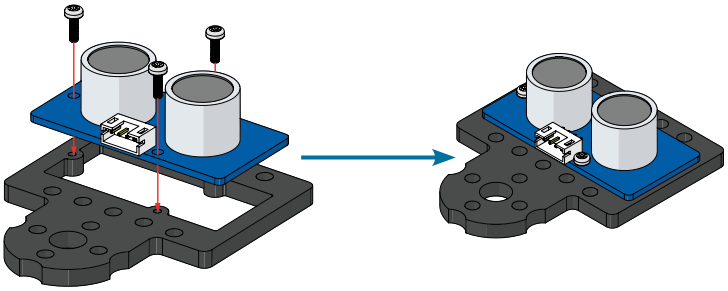


Сборка датчика линии



Винты в этом комплекте имеют самонарезающую резьбу, благодаря которой они закрепляются в гнёздах пластмассовой установочной пластины. Для их установки можно использовать отвертку 4-в-1, которая входит в робототехнический набор TETRIX. Следите за тем, чтобы не перетянуть винты при затяжке, так как это может привести к повреждению пластмассовой установочной пластины.

Сборка ультразвукового датчика



Винты в этом комплекте имеют самонарезающую резьбу, благодаря которой они закрепляются в гнёздах пластмассовой установочной пластины. Для их установки можно использовать отвертку 4-в-1, которая входит в робототехнический набор TETRIX. Следите за тем, чтобы не перетянуть винты при затяжке, так как это может привести к повреждению пластмассовой установочной пластины.

Упражнение 6: Сборка базового робота PULSE

Примечания для учителя: На продолжительность сборки могут повлиять различные факторы, включая такие, как упорядоченность набора и самостоятельная либо парная форма работы. Время указано приблизительно. Фактически затраченное время может отличаться.

Введение

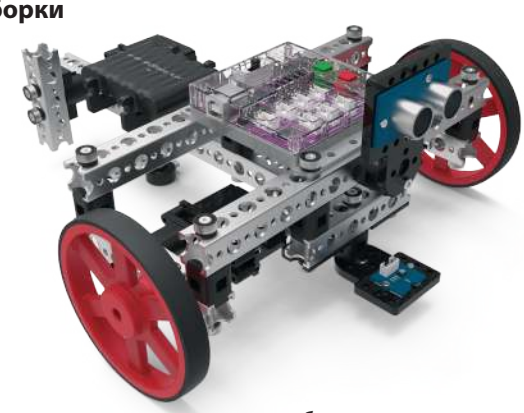
Теперь нам нужен робот. Поскольку настоящее руководство преимущественно ориентировано на работу с контроллером PULSE и программным обеспечением *TETRIX Ardublockly*, робот сложной конструкции не потребуется. С учётом этого вы и соберете базового робота PULSE. Базовый робот должен быть простым, лёгким в сборке и точно соответствовать задачам настоящего руководства, не иметь лишних деталей.

Однако всё, чему вы научитесь на примере этого простого робота, можно применять к более сложным моделям при дальнейшем освоении робототехники. Базовый робот с двумя электродвигателями постоянного тока и колёсами из конструктора PRIME с каждой стороны станет отличной испытательной машиной для изучения возможностей контроллера PULSE.

Вы начнёте работу с простейшей приводной рамы и будете постепенно дополнять её новыми деталями в ходе последующих упражнений. Такую конструкцию можно собрать даже при небольшом или нулевом опыте сборки металлических конструкторов.

Ожидаемая продолжительность сборки

30 минут



Связь с реальным миром

При работе на производственно-сборочных предприятиях необходимо точно соблюдать инструкции. Например, если попробовать изготовить некий предмет мебели, игнорируя инструкцию, скорее всего его придётся переделывать. Умение следовать определённому порядку действий — важный навык, применимый к самым разным жизненным ситуациям.

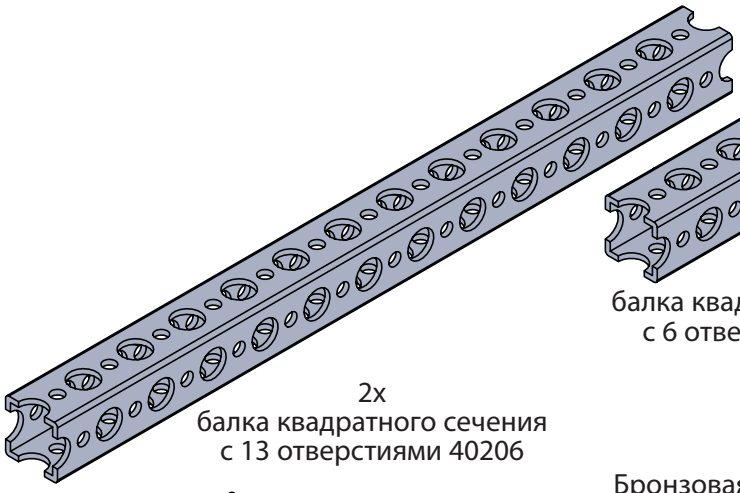
Профессии: оператор машинного оборудования, специалист по робототехнике, инженер

Связь с точными и естественными науками

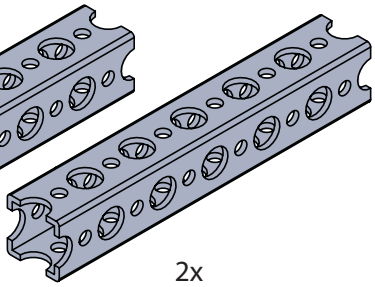
- Физика
  - Колесо и ось
  - Винт
- Технология
  - Материалы
  - Крепления
- Техническое конструирование
  - Конструирование машин
  - Строительство
- Математика
  - Длина
  - Диаметр

Шаг 1

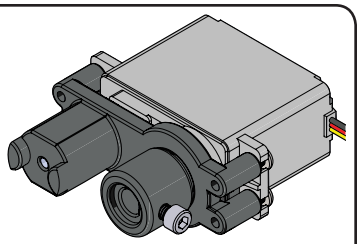
Необходимое оборудование



2x
балка квадратного сечения
с 13 отверстиями 40206

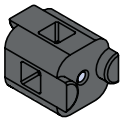


2x
балка квадратного сечения
с 6 отверстиями 40203

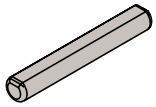


2x
электродвигатель
постоянного тока
TETRIX 44298 со
скобой крепления
сервоприводов 40232
(указания по сборке на
стр. 54-55)

4x
Бронзовая втулка 8 x 6 мм 40227



5x
блочный соединитель для
крестообразного крепления под
углом 90 градусов 40217

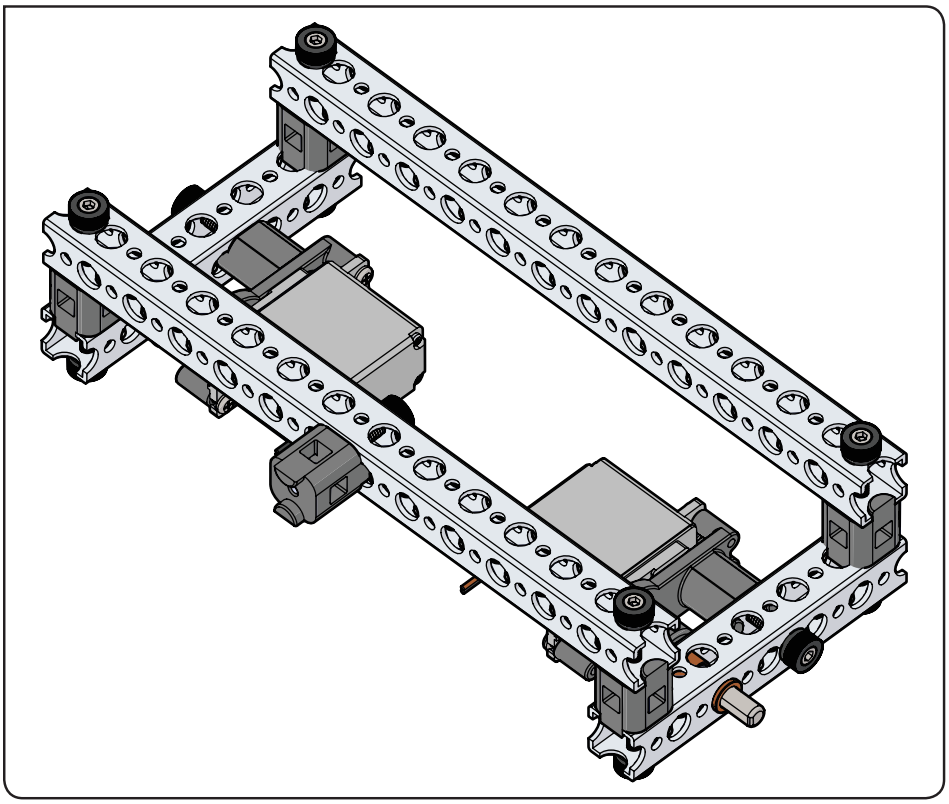


2x
стальная ось 40 мм
40226

11x
Винт с рифлёной
головкой 40323

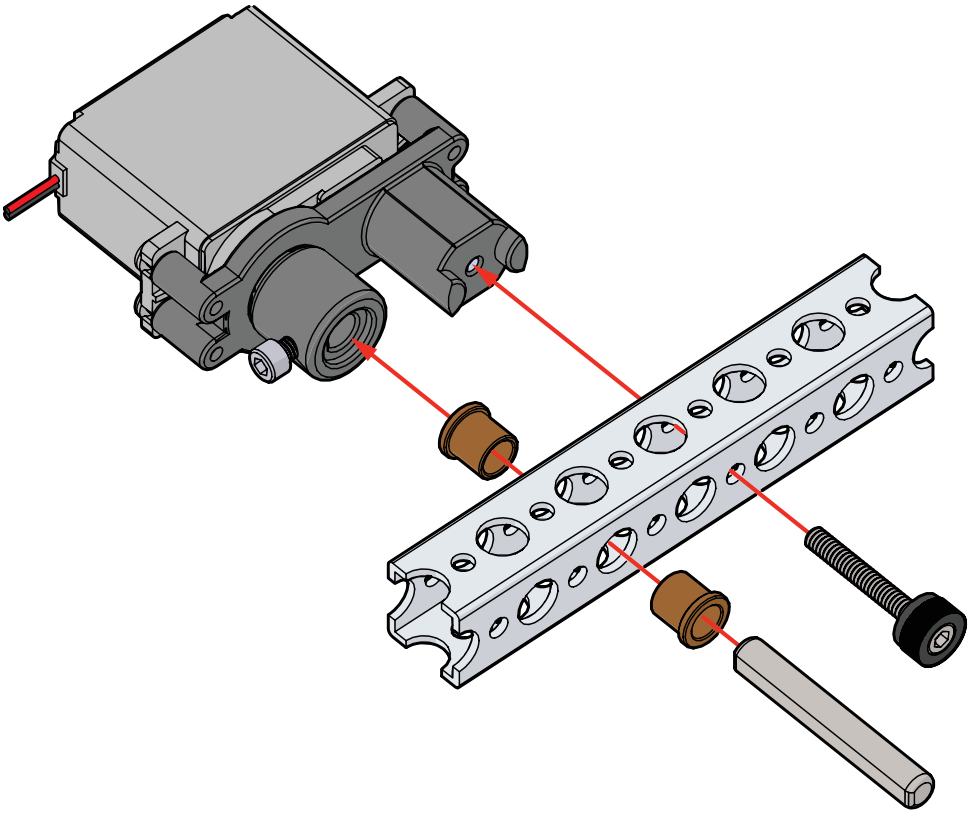
Совет: Указания по определению вида балок см. на стр. 48. Чтобы узнать наименование балок квадратного сечения TETRIX PRIME, подсчитайте в них малые отверстия.

На промежуточном этапе конструкция должна выглядеть так.

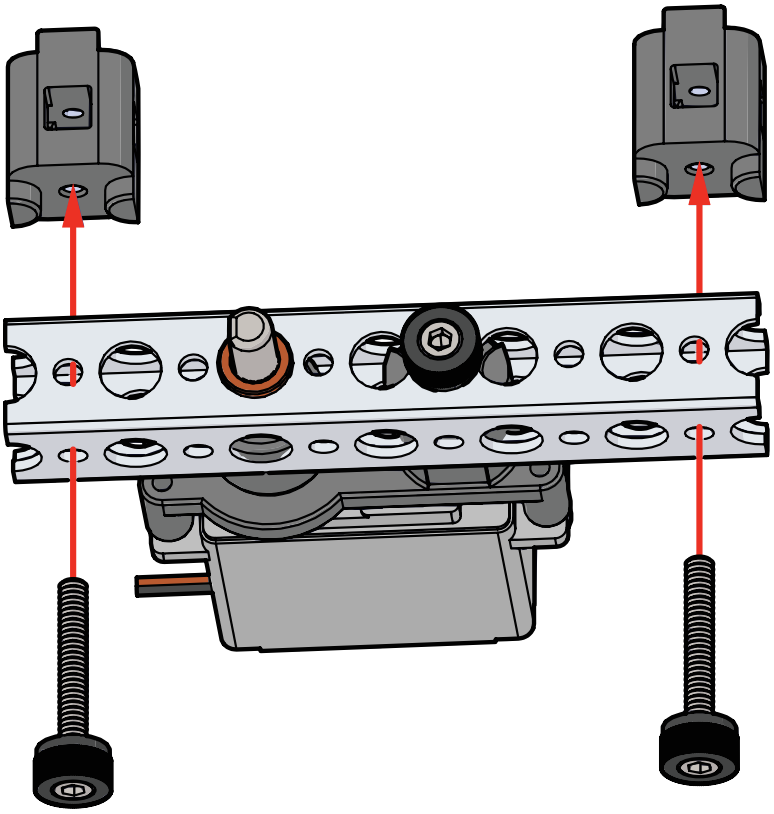


Шаг 1.0

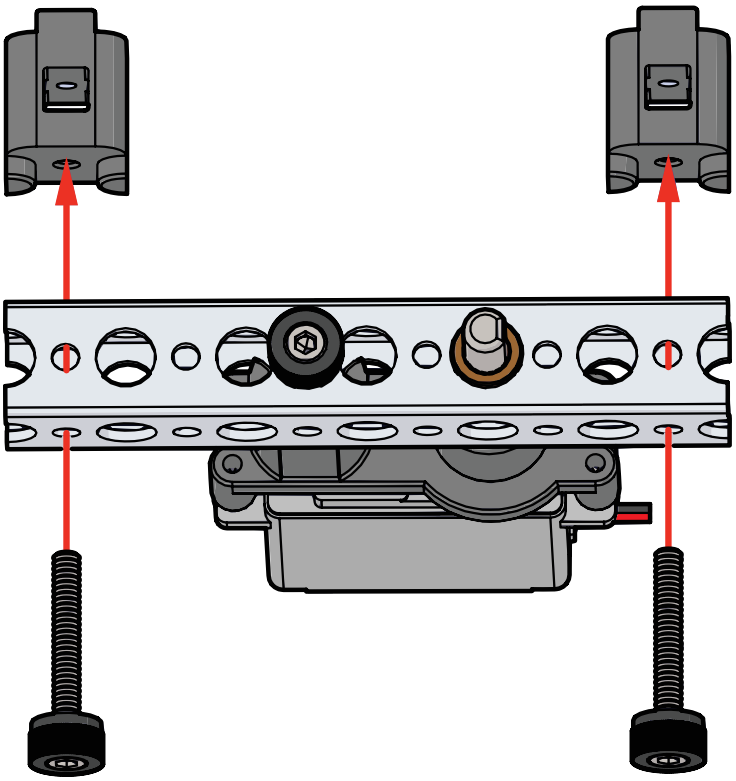
Соберите два таких узла.



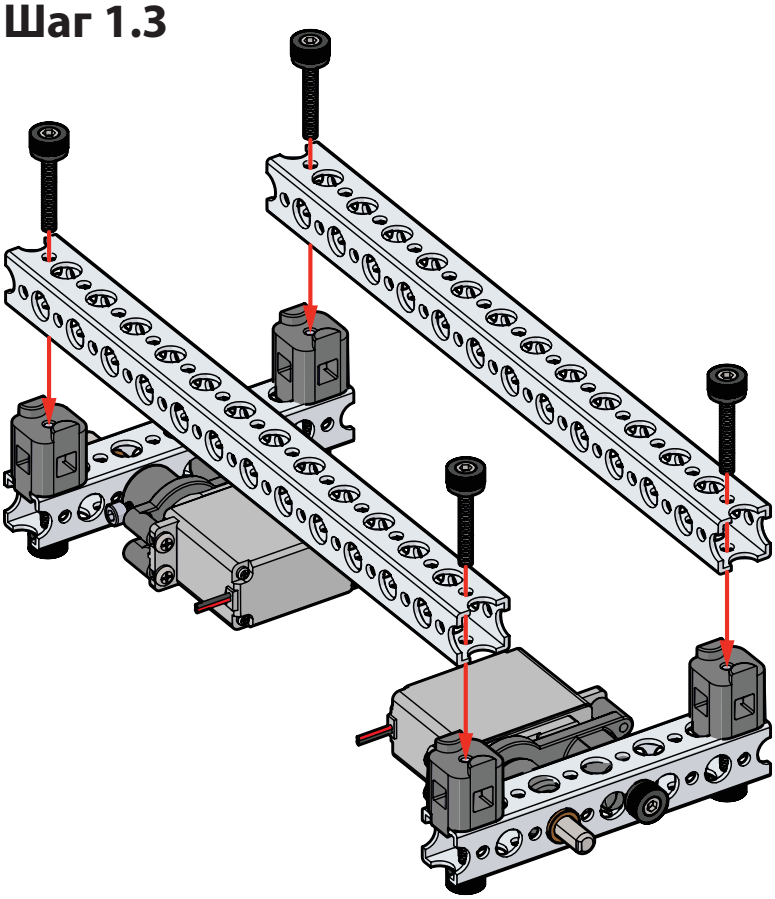
Шаг 1.1



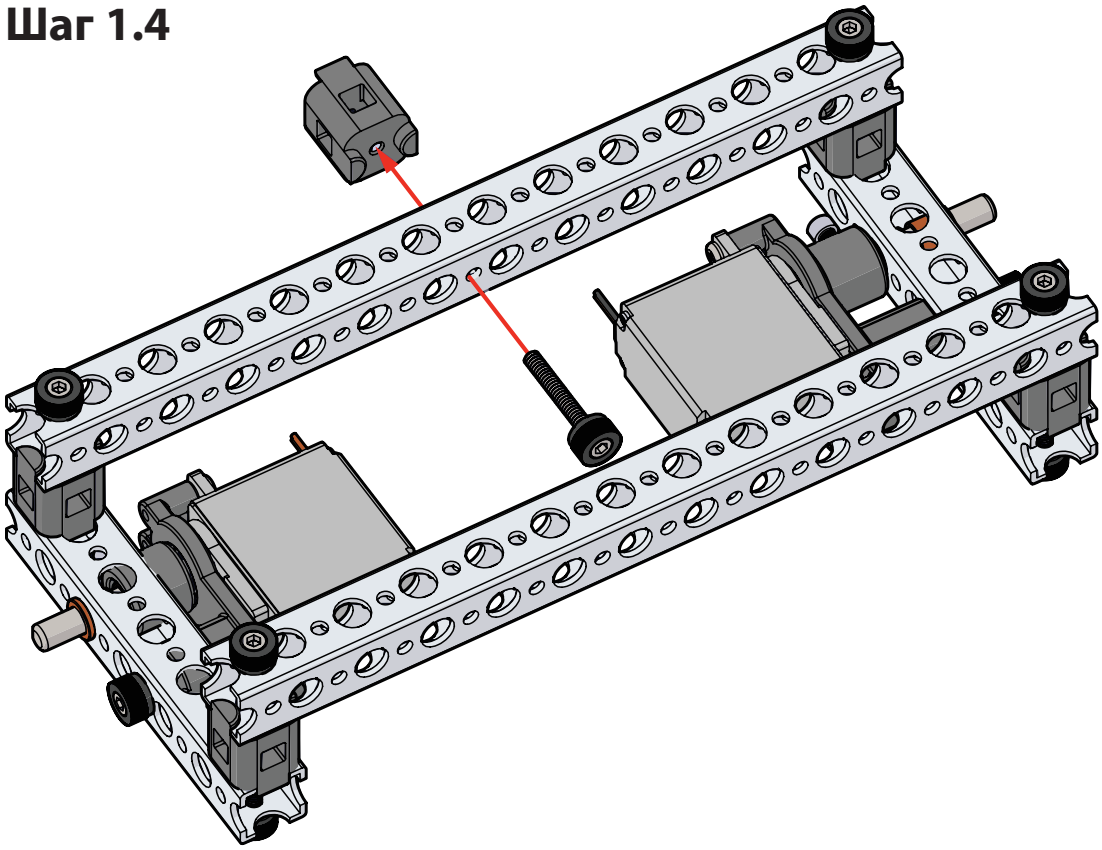
Шаг 1.2



Шаг 1.3

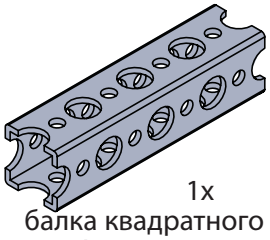


Шаг 1.4

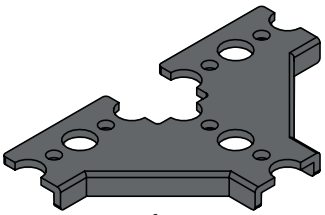


Шаг 2

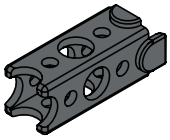
Необходимое оборудование



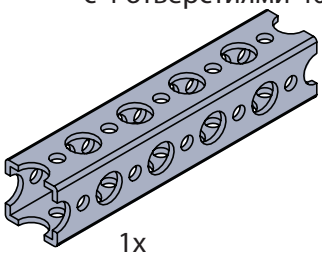
1x
балка квадратного сечения с 4 отверстиями 40201



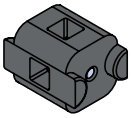
1x
скоба для Т-образного соединения деталей 40210



1x
Торцевой балочный соединитель 40214

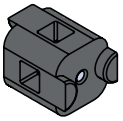


1x
балка квадратного сечения с 5 отверстиями 40202



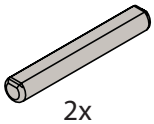
2x
блочный соединитель для параллельного крепления балок 40216

1x
балка квадратного сечения с 15 отверстиями 40207

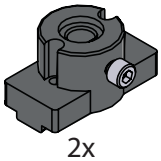


2x
блочный соединитель для крестообразного крепления под углом 90 градусов 40217

10x
Винт с рифлёной головкой 40323

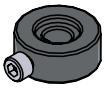


2x
стальная ось 40 мм 40226

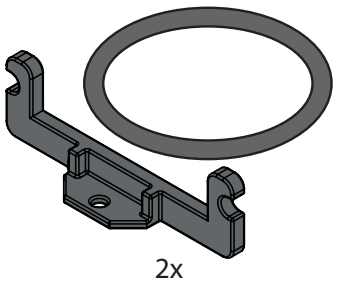


2x
ступица для крепления к балке 40228

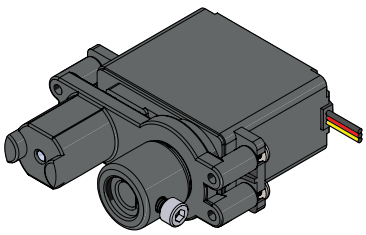
1x
барашковая гайка 40221



1x
установочное кольцо вала с лыской 40229



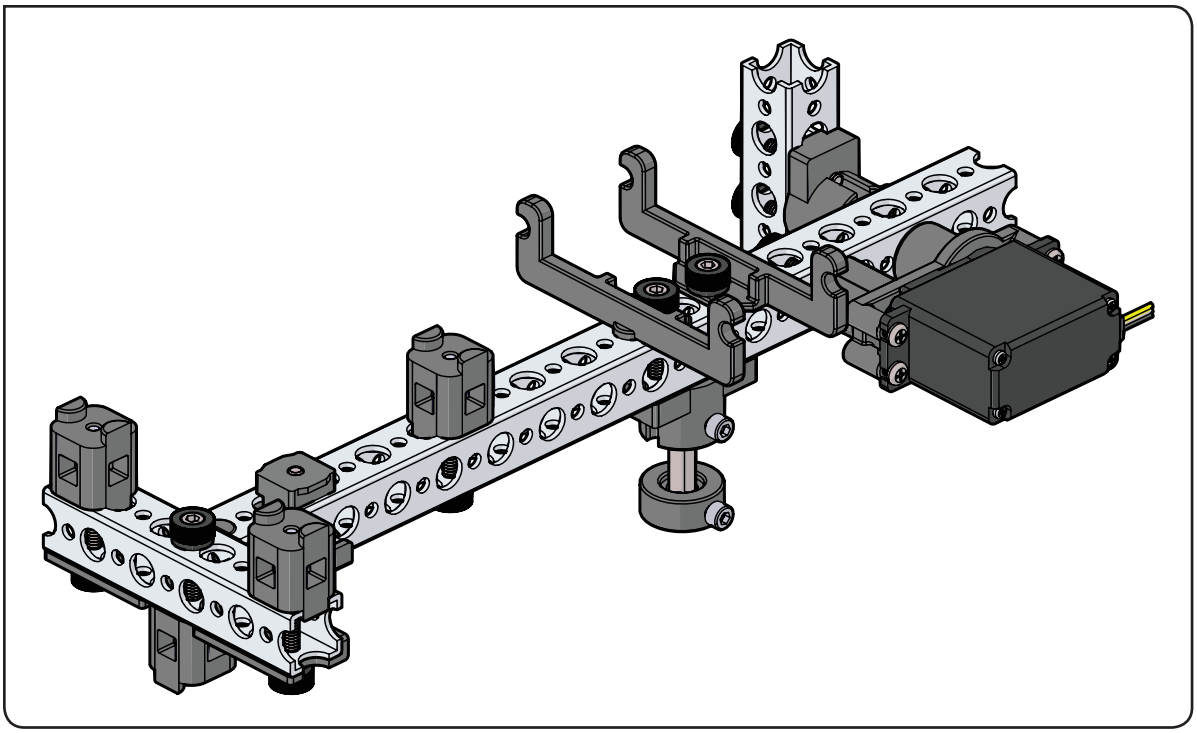
2x
Скоба крепления аккумуляторной батареи 40236



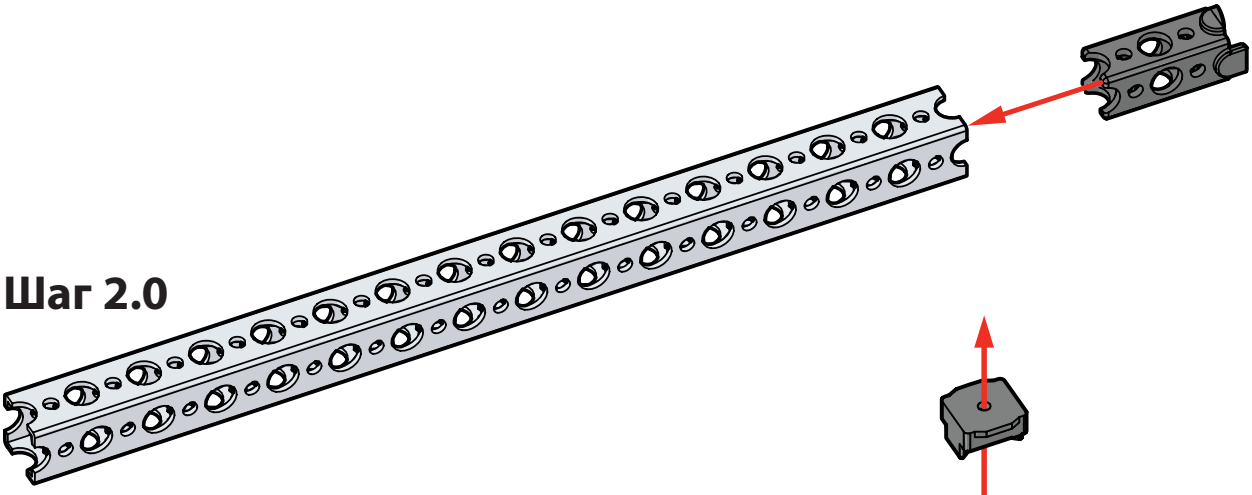
1x
сервопривод HiTec HS322-HD 40538 со скобой крепления сервоприводов 40232 (указания по сборке на стр. 54-55)

2x
Бронзовая втулка 8 x 6 мм 40227

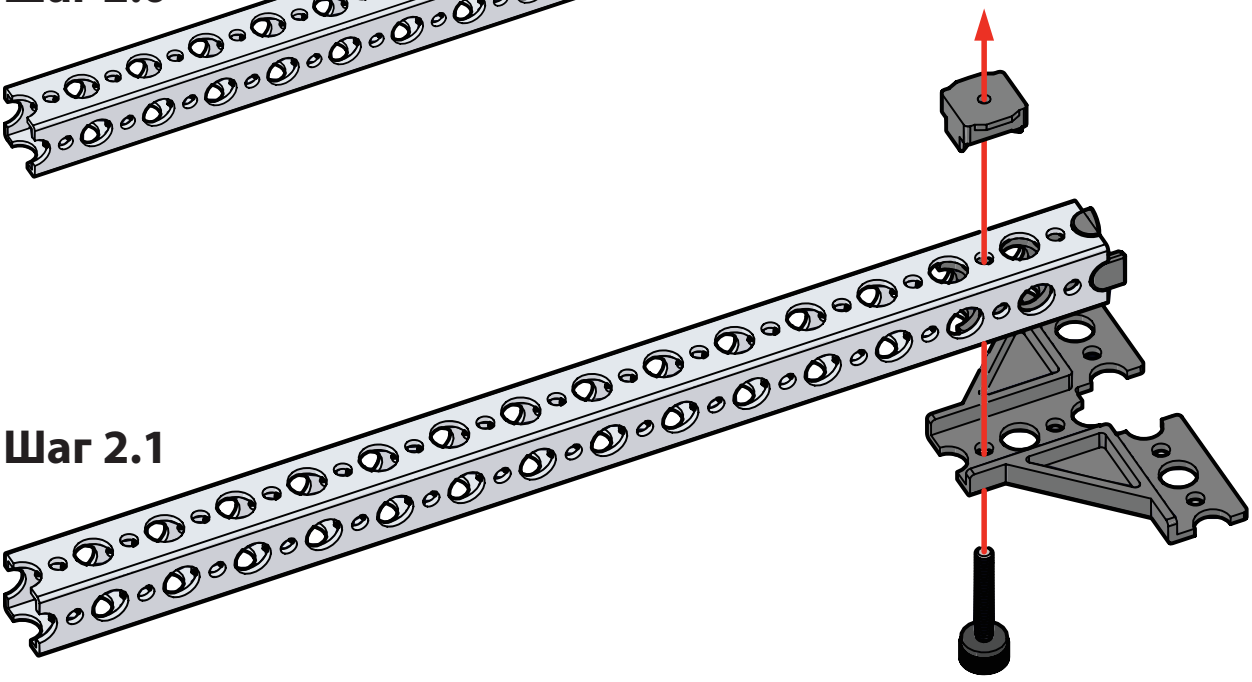
На промежуточном этапе конструкция должна выглядеть так.



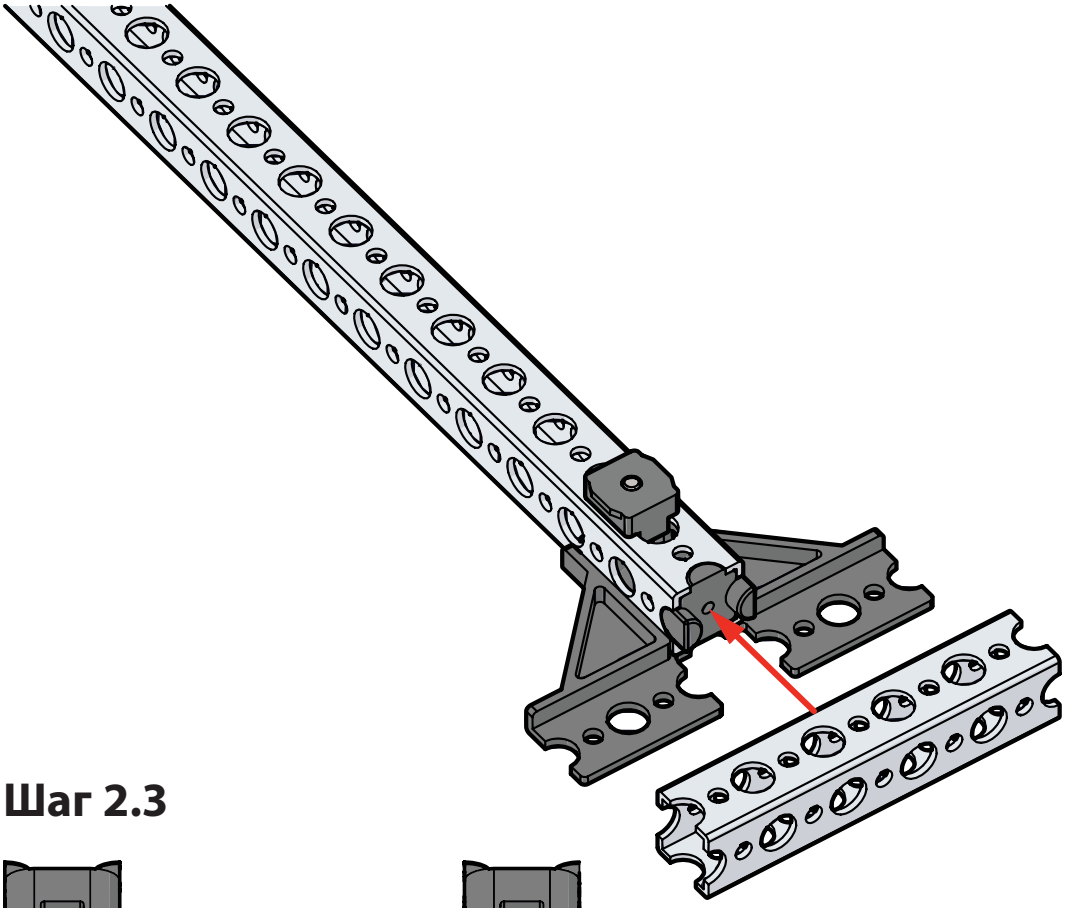
Шаг 2.0



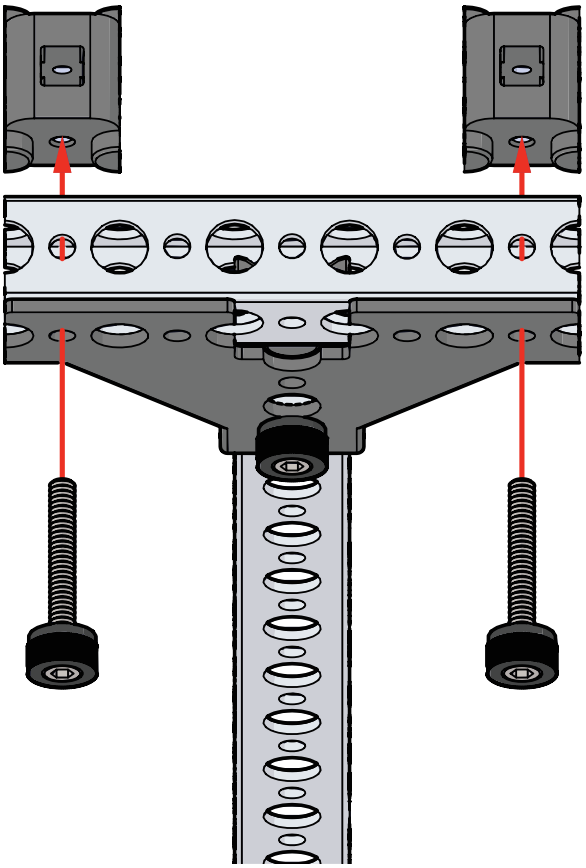
Шаг 2.1



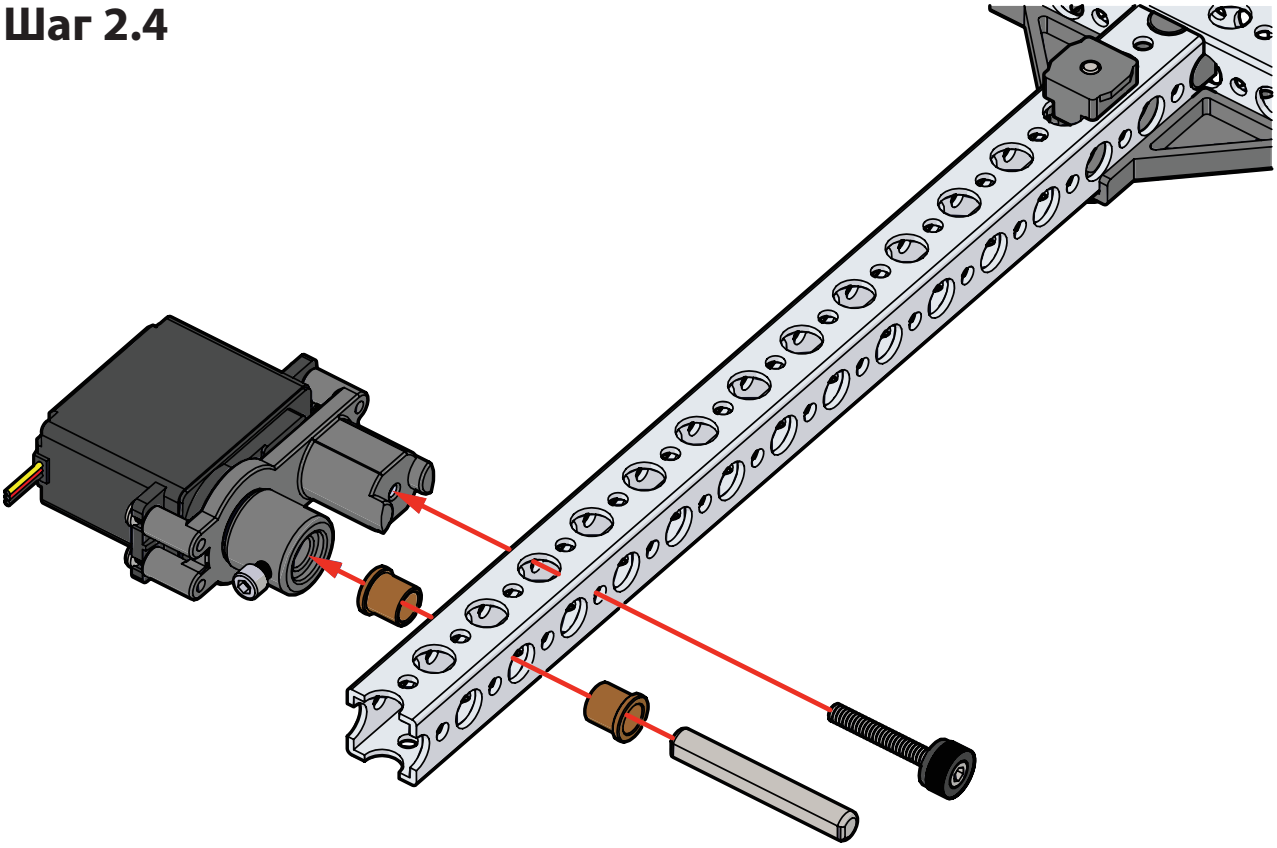
Шаг 2.2



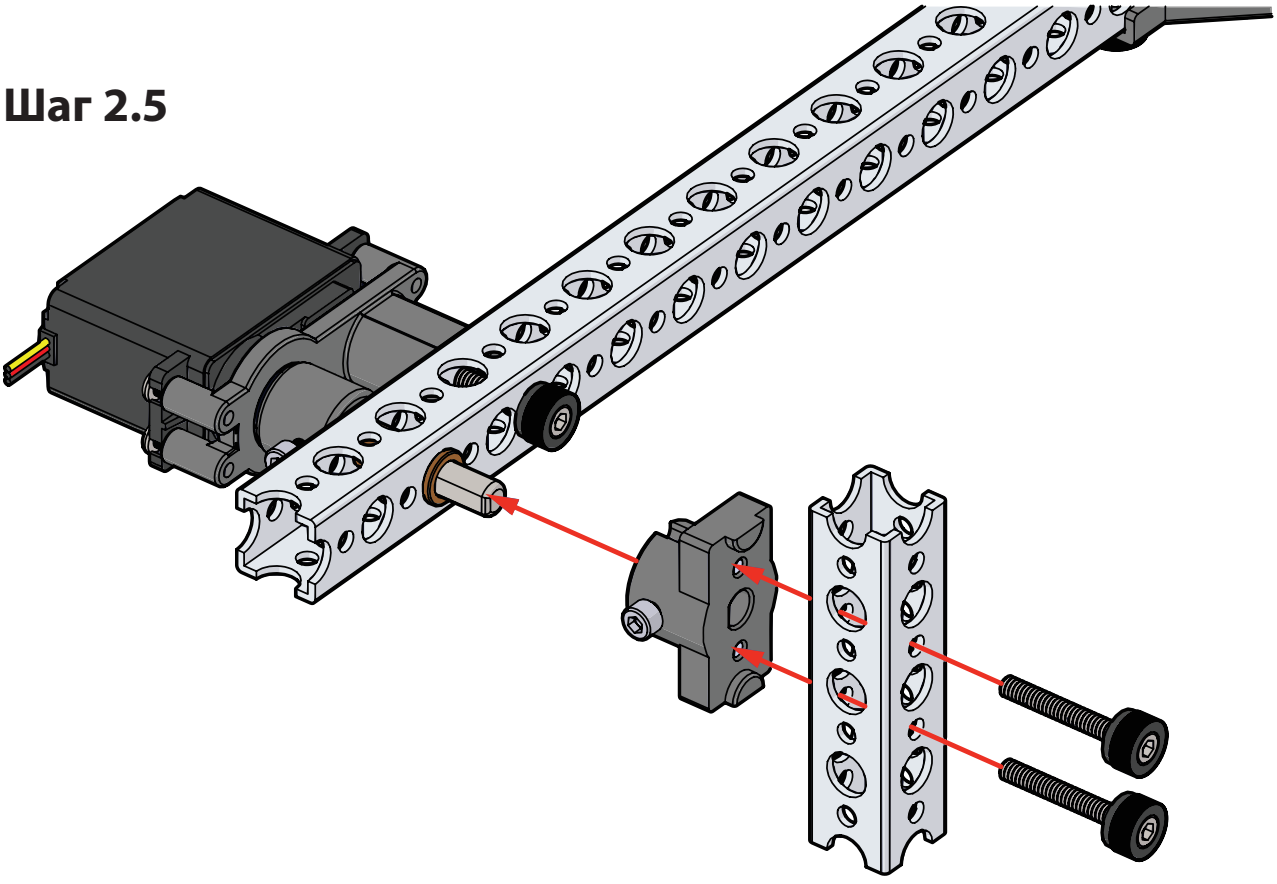
Шаг 2.3



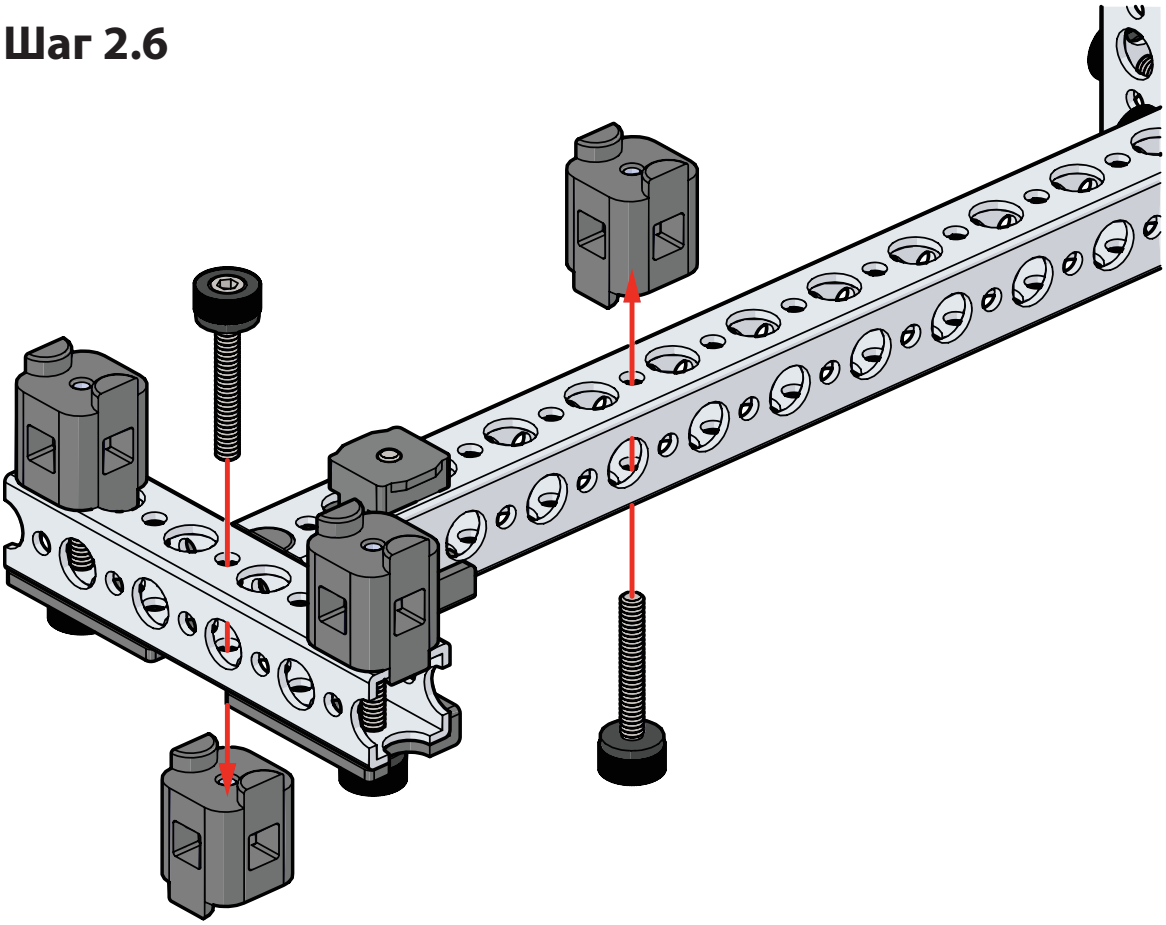
Шаг 2.4



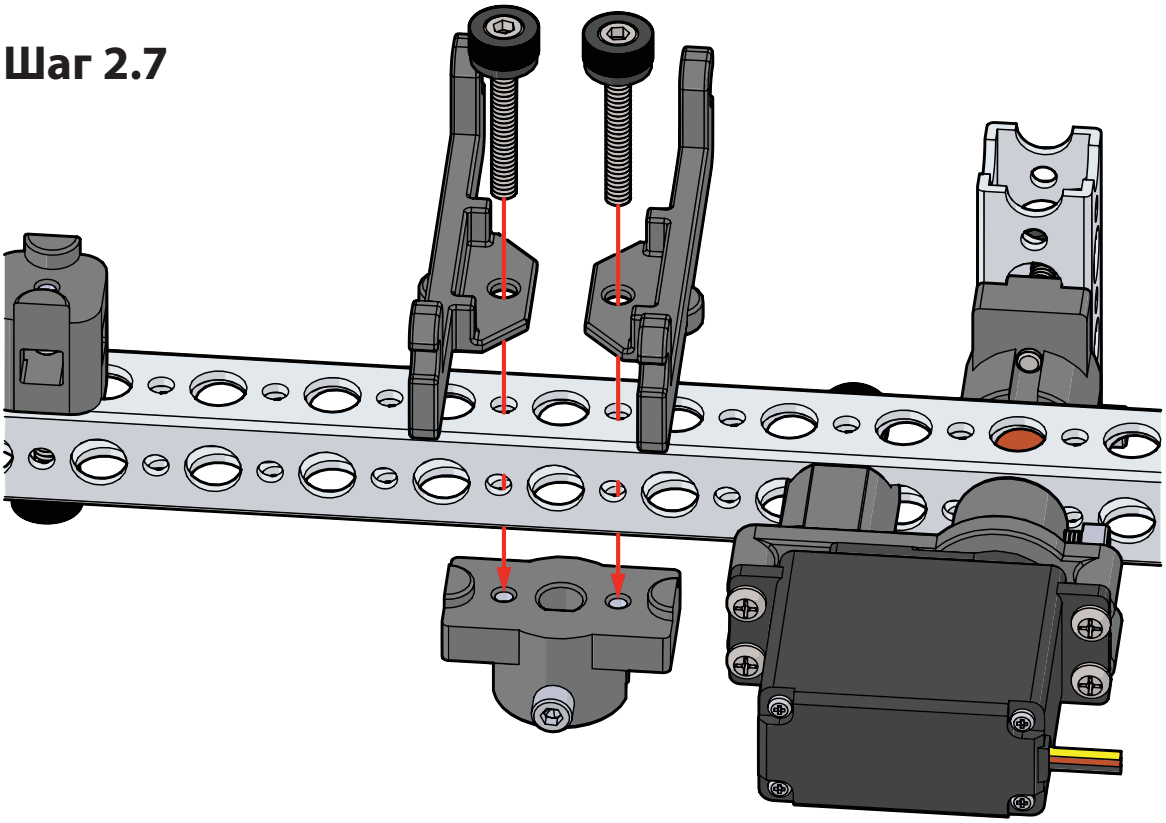
Шаг 2.5



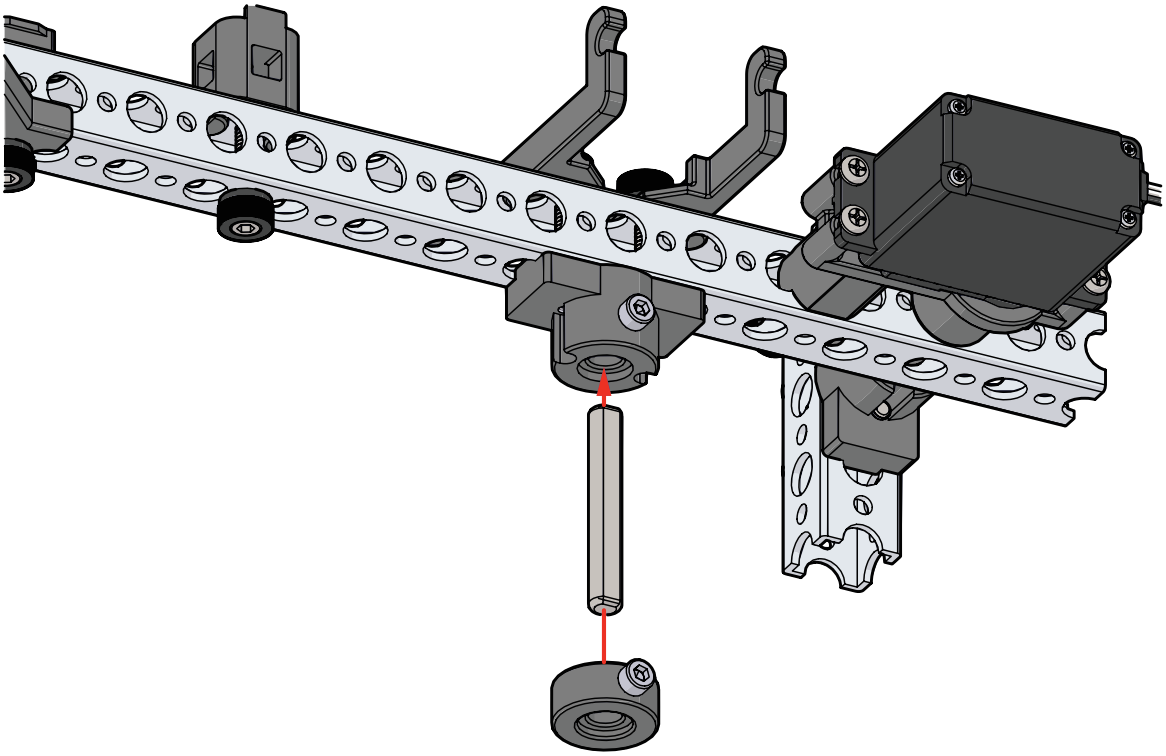
Шаг 2.6



Шаг 2.7

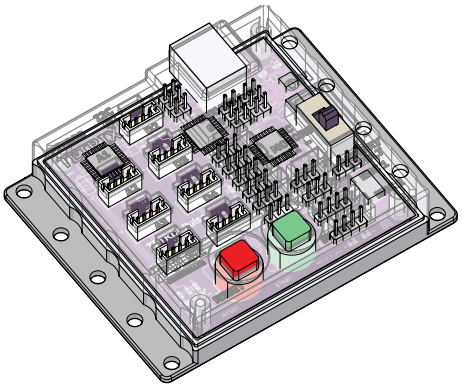


Шаг 2.8

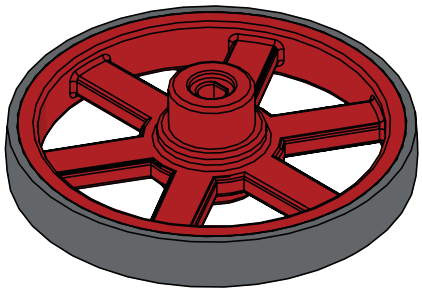


Шаг 3

Необходимое оборудование



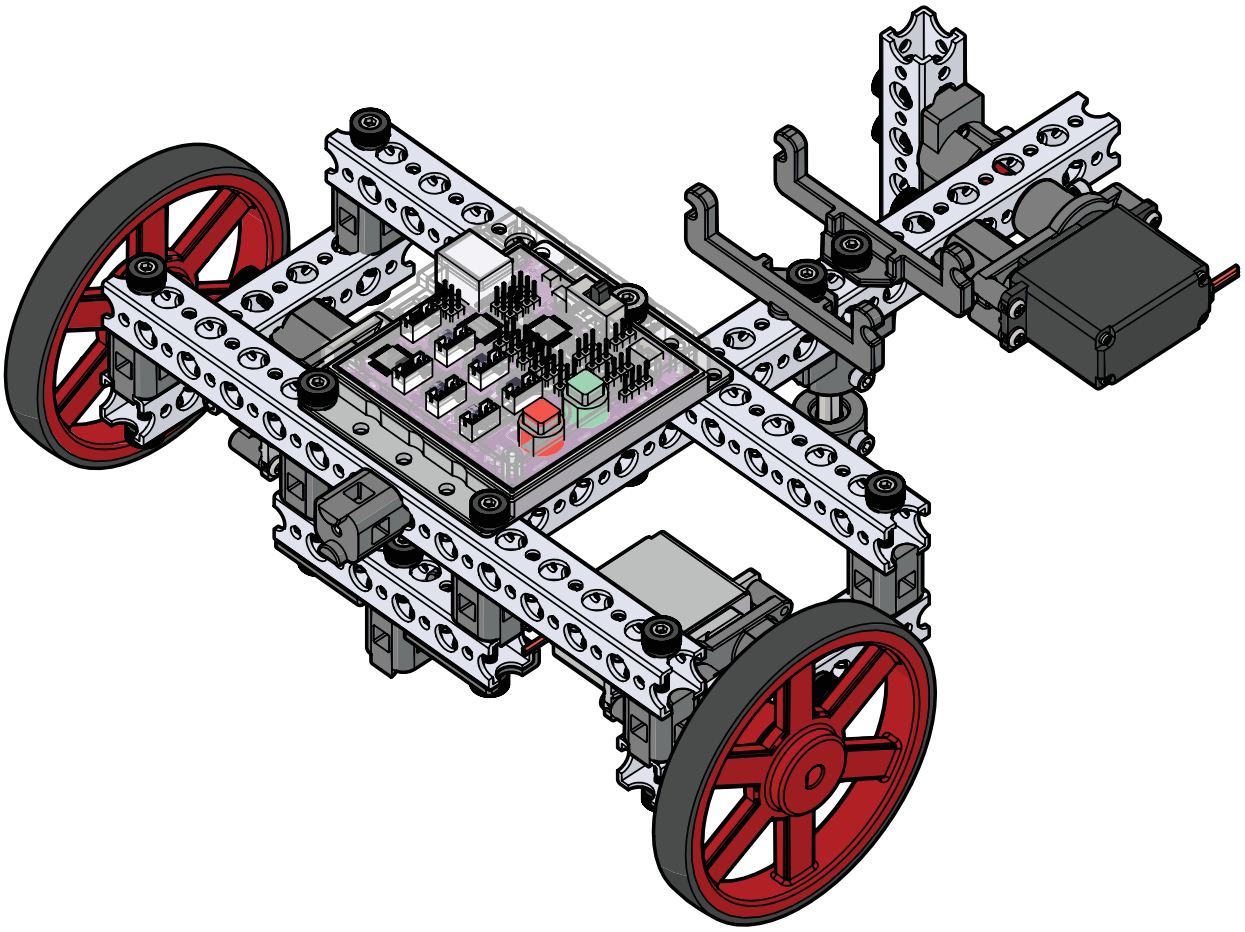
1x
контроллер TETRIX® PULSE™
44268



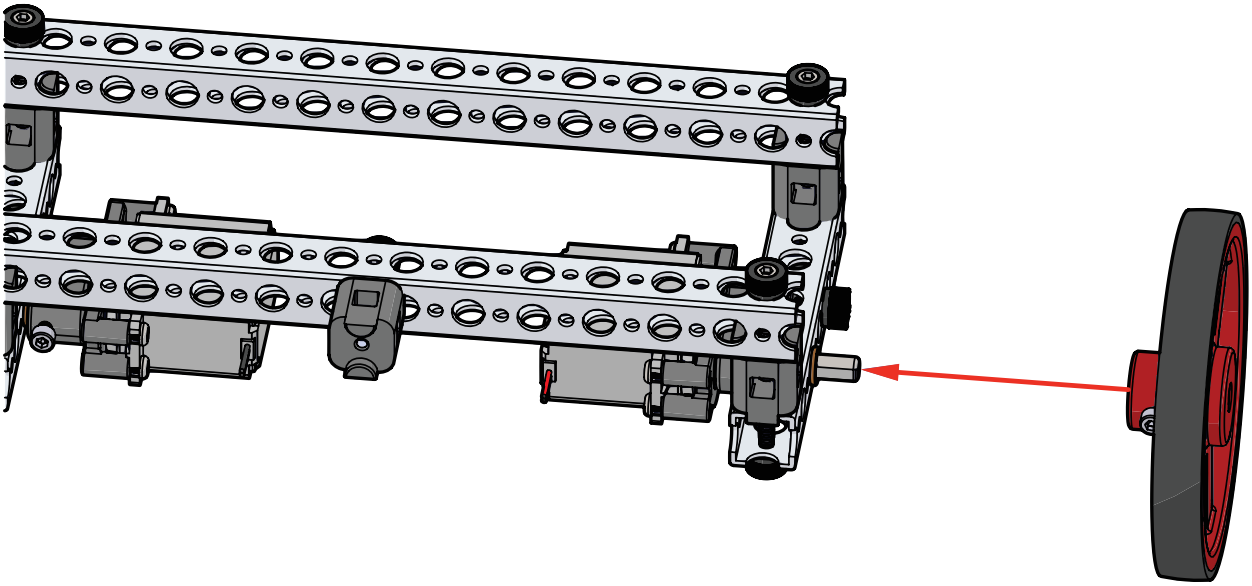
2x
Колесо с шиной 40222

3x
Винт с рифлёной головкой 40323

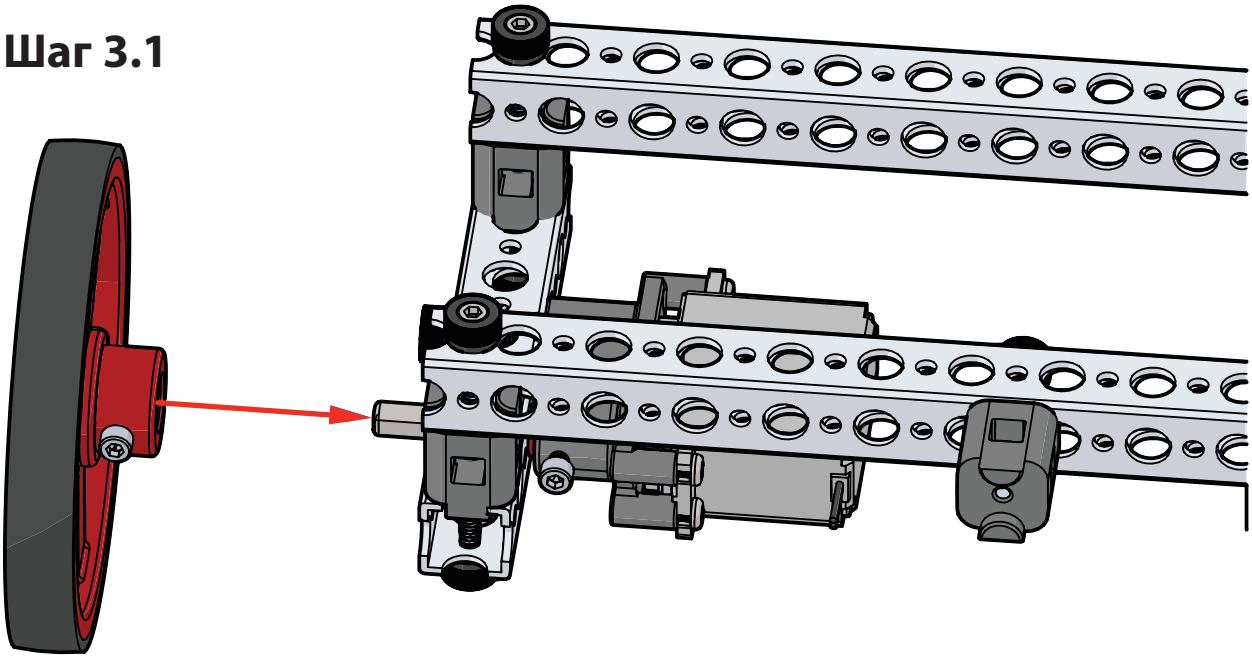
На промежуточном этапе конструкция должна выглядеть так.



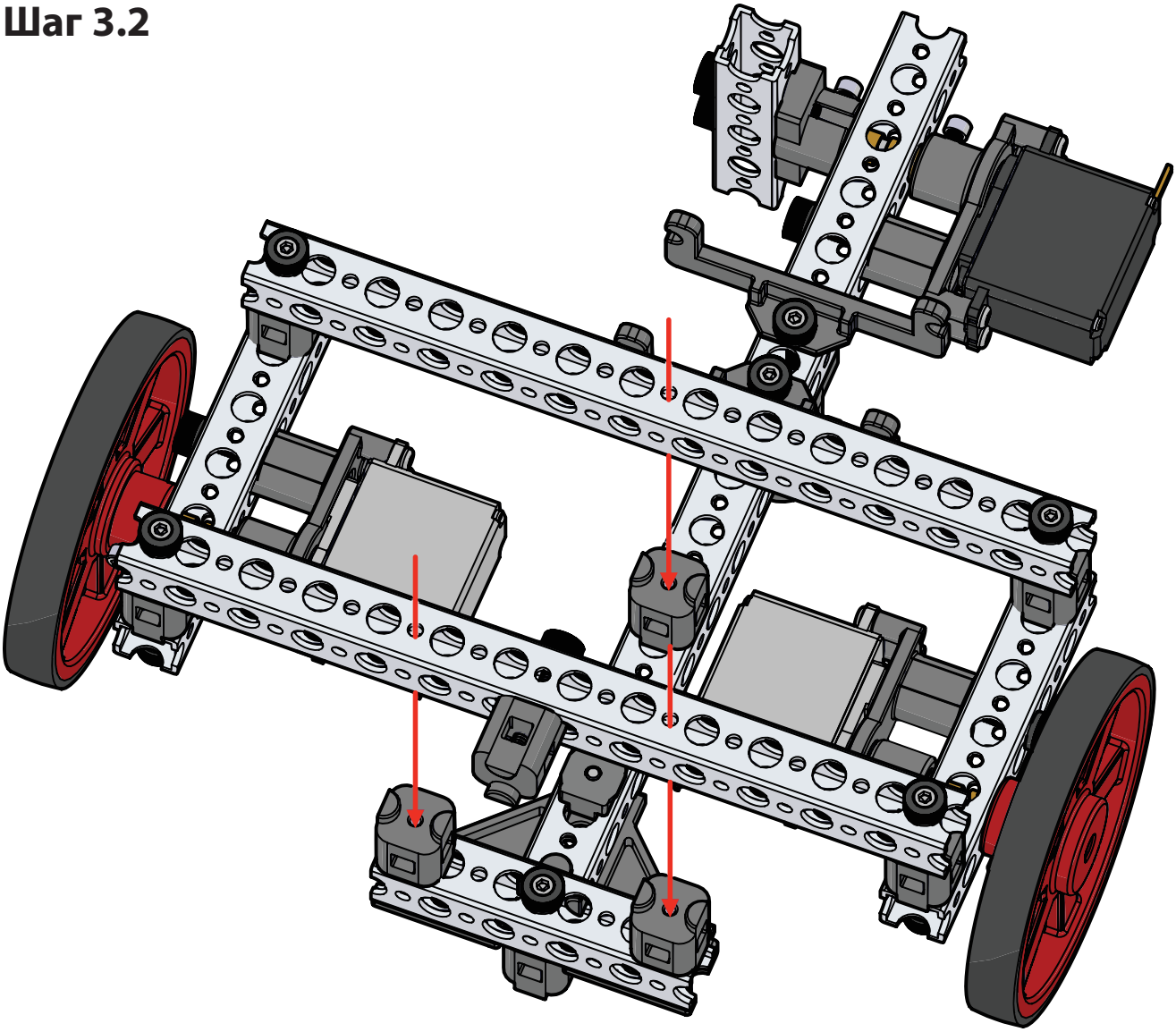
Шаг 3.0



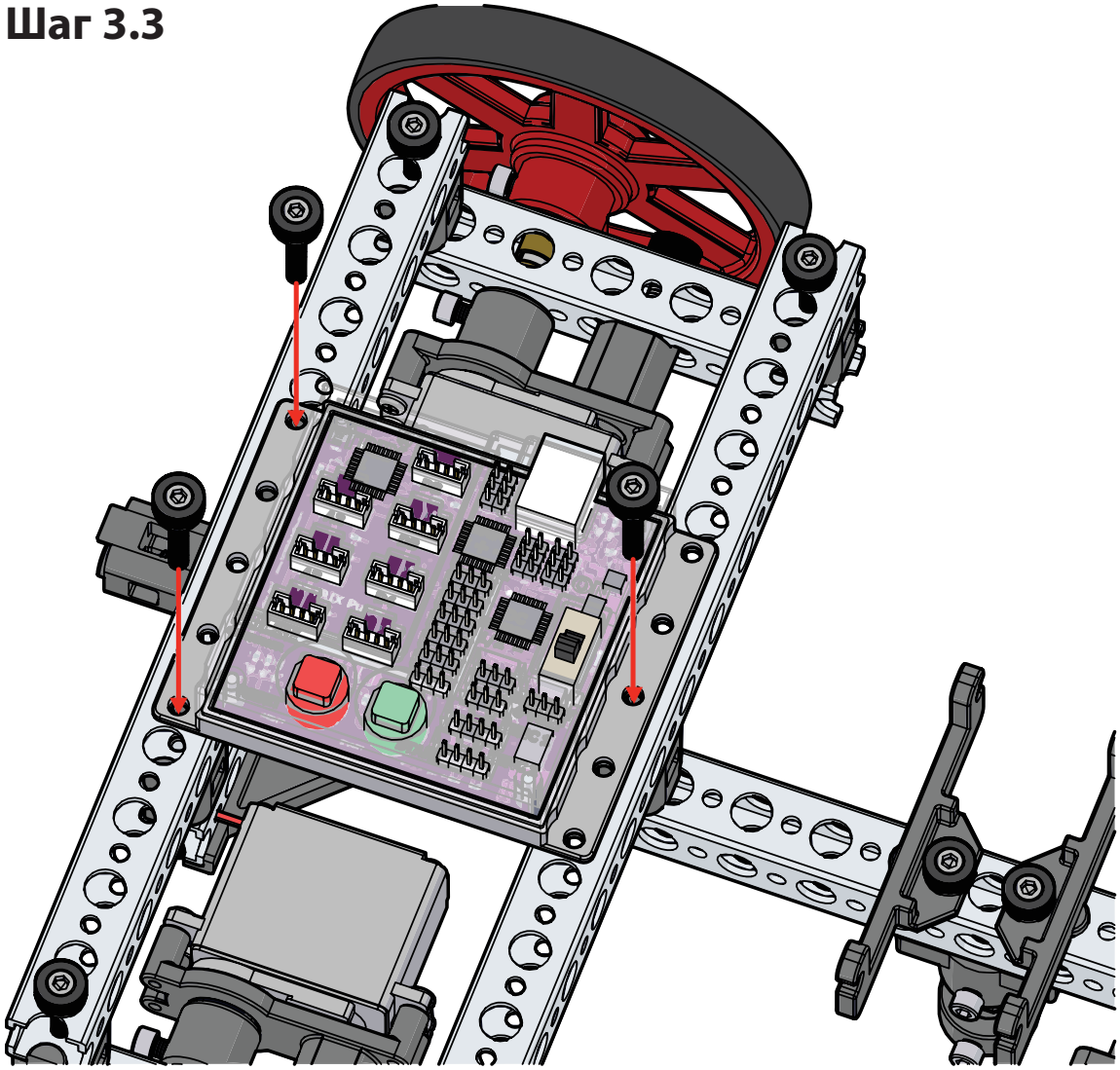
Шаг 3.1



Шаг 3.2



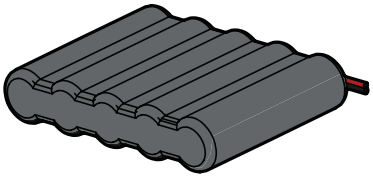
Шаг 3.3



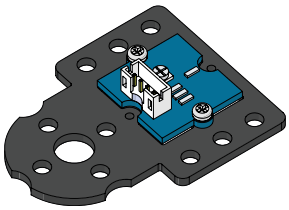
Шаг 4

Необходимое оборудование

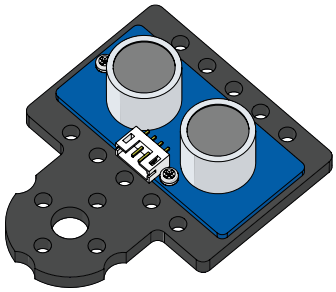
2x
винт с углублением под ключ
40516



1x
Аккумуляторная батарея NiMH
6 В 40235

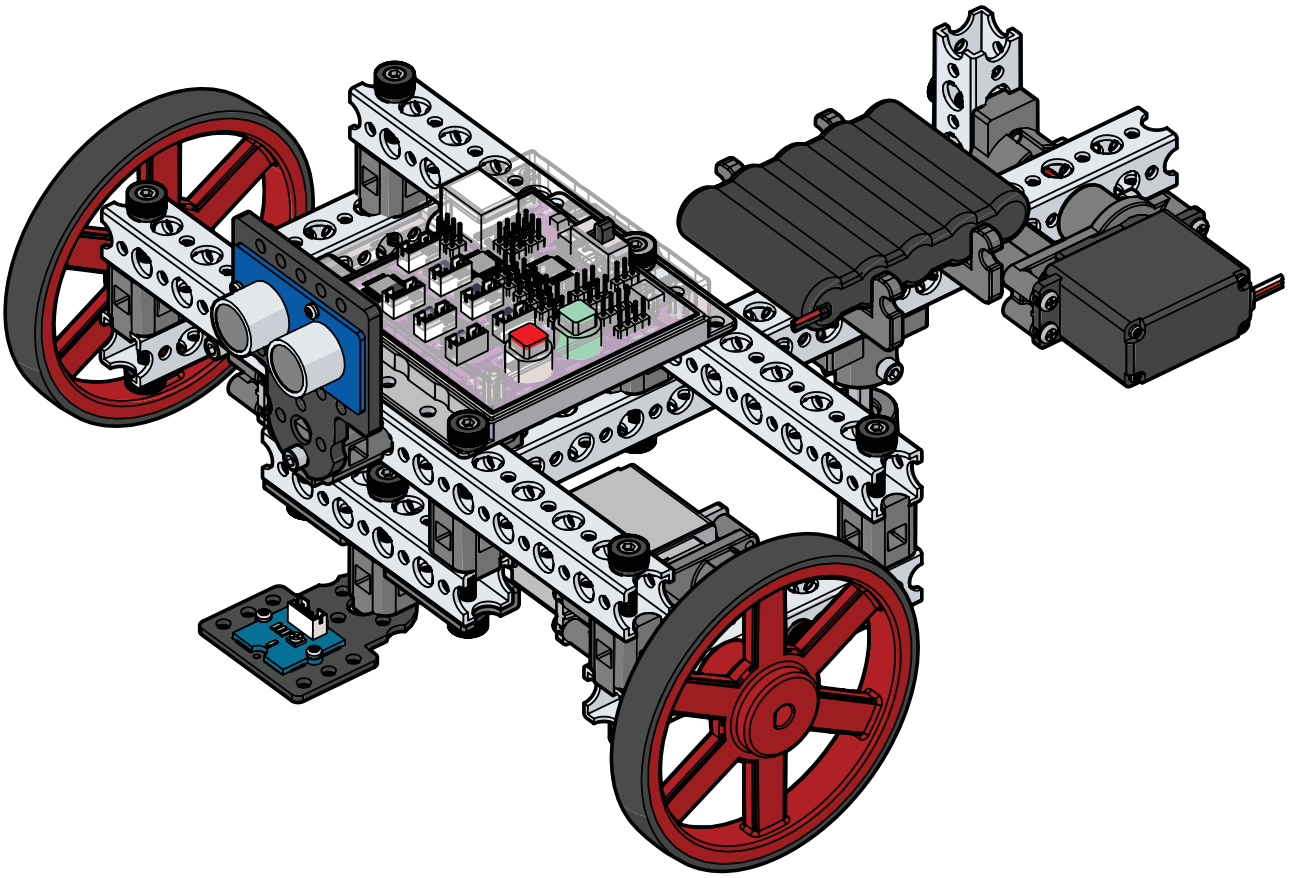


1x
комплект датчика линии 43056

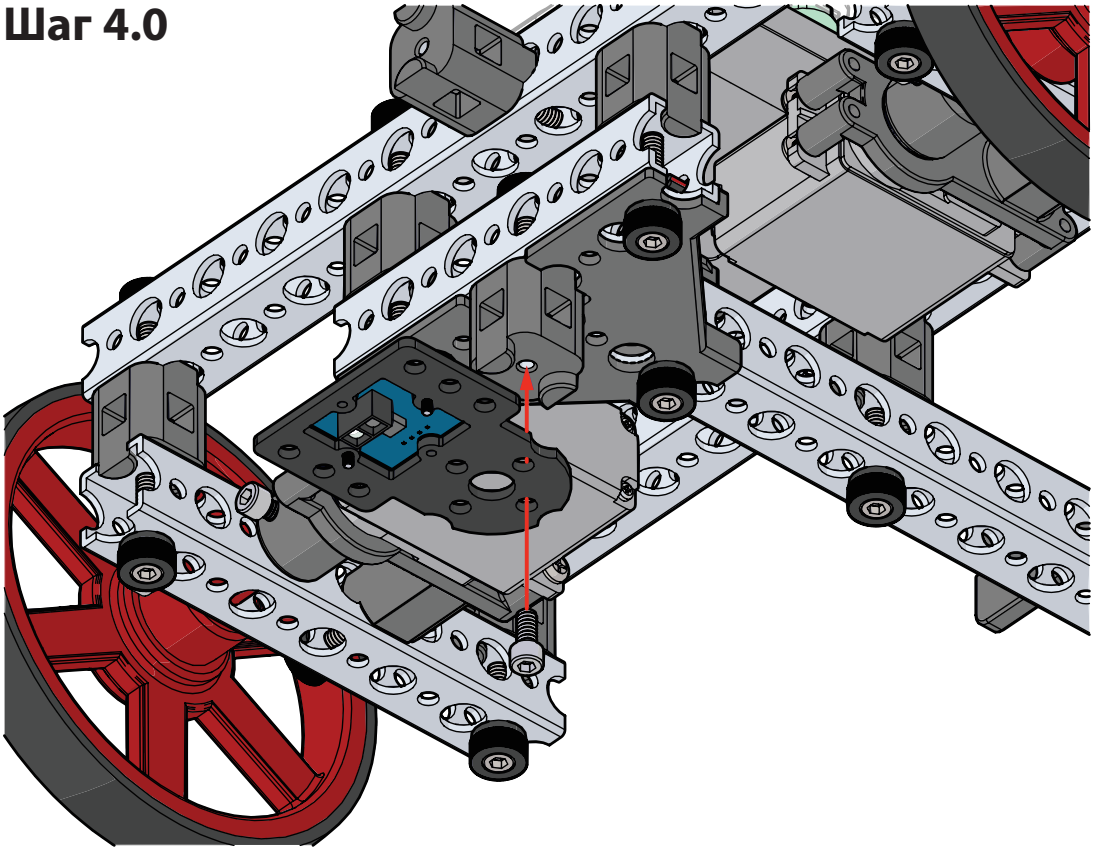


1x
комплект ультразвукового
датчика 43055

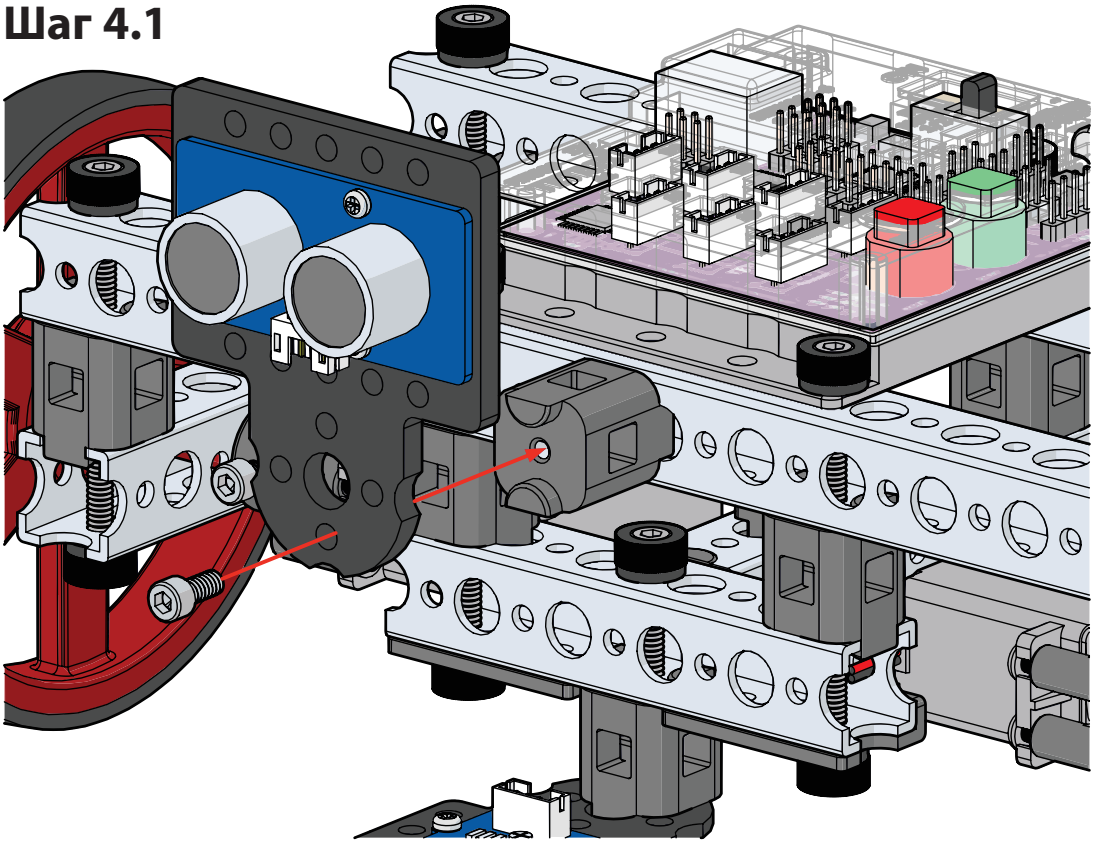
На промежуточном этапе конструкция должна выглядеть так.



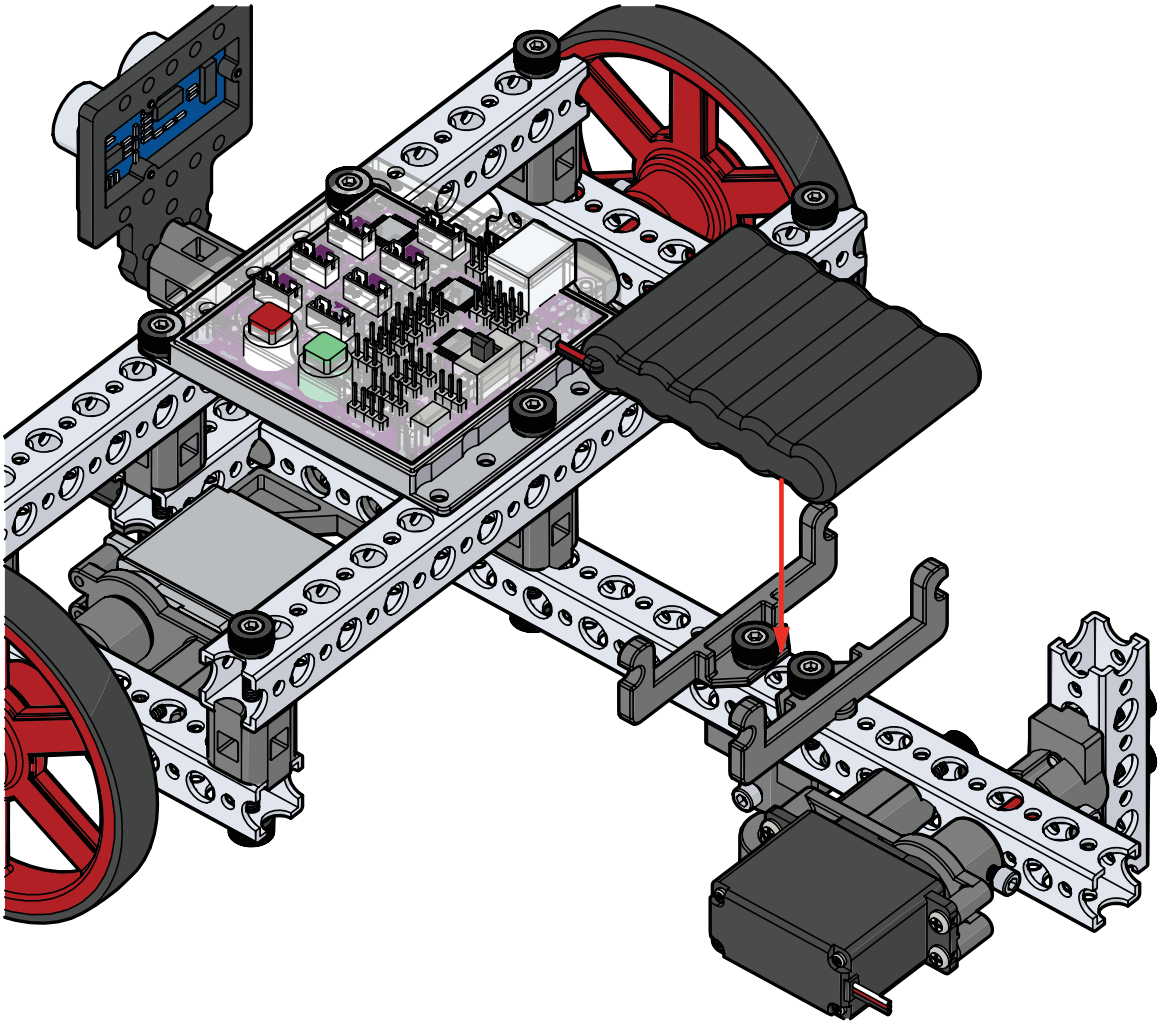
Шаг 4.0



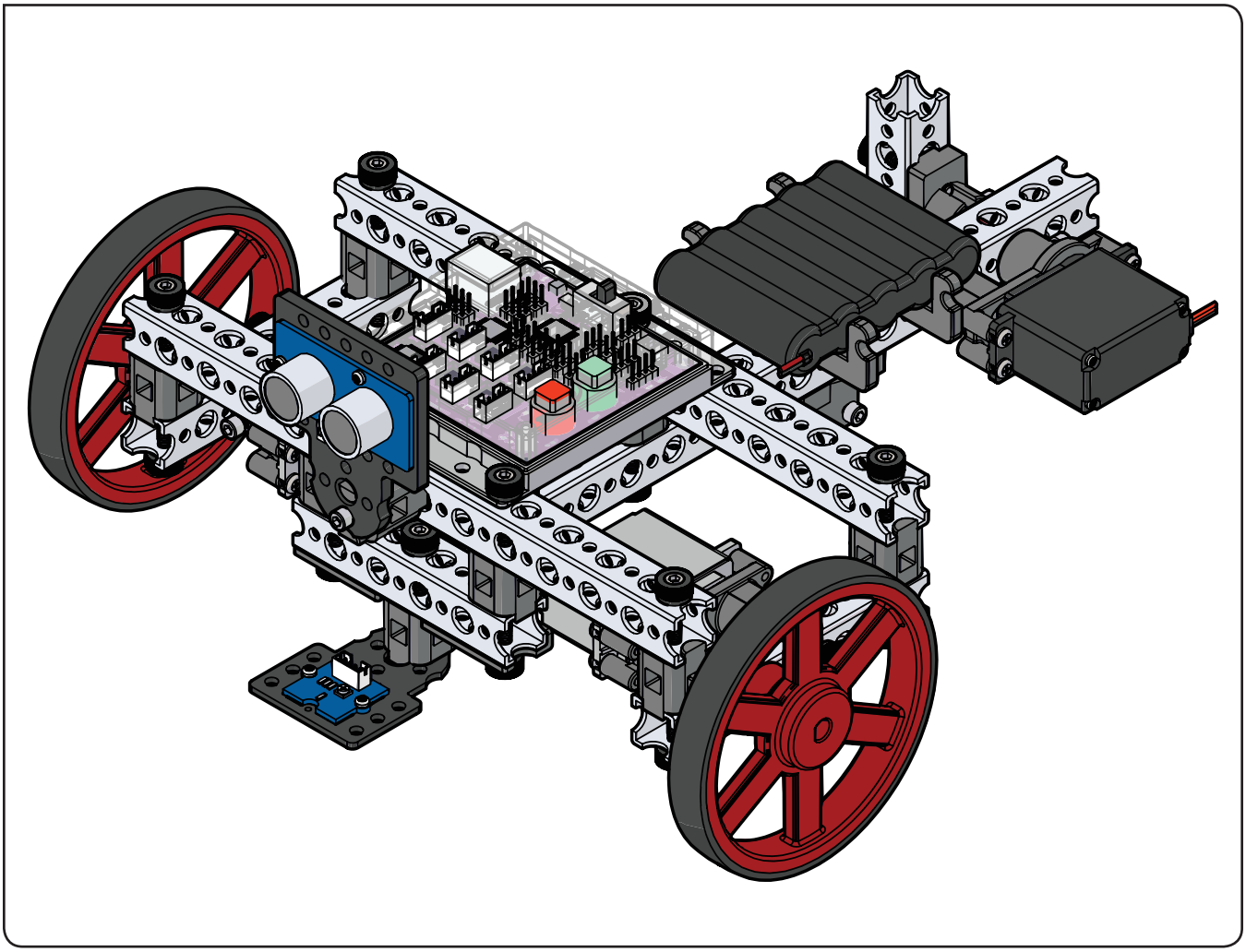
Шаг 4.1



Шаг 4.2



Собранный робот выглядит следующим образом.



Упражнение 7: Движение вперёд

Примечания для учителя: На продолжительность сборки могут повлиять различные факторы, включая такие, как упорядоченность набора и самостоятельная либо парная форма работы. Время указано приблизительно. Фактически затраченное время может отличаться.

Введение

Это ваше первое упражнение по написанию кода для базового робота PULSE, поэтому программа будет простой. В ходе этого упражнения вам предстоит создать скетч, который заставит базового робота двинуться вперёд и по истечении трёх секунд остановиться. На этом программа завершится.

Пользуясь знаниями, полученными в ходе упражнения 2, необходимо установить второй электродвигатель постоянного тока и добиться синхронной работы обоих двигателей. Способность синхронно управлять двигателями является фундаментальным требованием для работы с мобильными роботами.

Необходимое оборудование

Открытие программы

Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты
Examples > Activity_7. Откроется окно нового скетча под заголовком Activity_7 (рис. 45).

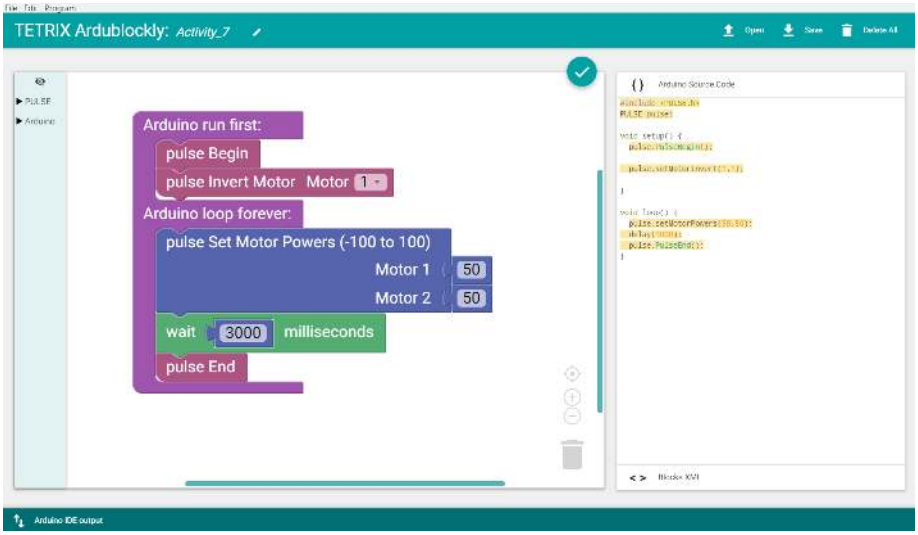


Рисунок 45

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После включения светодиода отсоедините кабель USB и поставьте базового робота на пол.

Поставьте его так, чтобы колёса находились спереди. Если робот едет назад вместо того, чтобы ехать вперёд, поменяйте местами провода электродвигателей постоянного тока в портах контроллера.

Для запуска скетча нажмите на зелёную кнопку "Пуск". Проследите за направлением и продолжительностью движения базового робота. Соответствуют ли его действия приведённому описанию скетча? Базовый робот должен остановиться через три секунды.

Дальнейшее изучение

В скетче представлено три новых блока: pulse Invert Motor, pulse Set Motor Powers и pulse End. Блок pulse Invert Motor (рис. 46) позволяет менять направление вращения двигателя.

Если двигатели установлены с противоположных сторон, с помощью блока можно задать одно направление вращения для обоих двигателей, обеспечив их согласованную работу. Это облегчит вашу работу по программированию. Данная функция включает в себя два параметра. Первый параметр задаёт канал двигателя, второй параметр определяет, менять или не менять направление вращения (0 или 1).

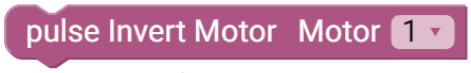


Рисунок 46

Блок pulse Set Motor Powers (рис. 47) позволит одновременно задать мощность электродвигателя 1 и электродвигателя 2. Эти два параметра определяют частоту вращения каждого из двигателей. В рассматриваемом скетче мощность каждого из двигателей установлена на 50 %.

Рисунок 47

Блок pulse End останавливает или завершает выполнение программы. Если бы этого блока не было, программа продолжала бы исполнять цикл до тех пор, пока вы не остановили бы её нажатием на красную кнопку на корпусе контроллера.

Совет: Без использования кодовых датчиков частота вращения электродвигателей постоянного тока, задаваемая через значение мощности, может различаться в зависимости от заряда аккумуляторной батареи.

В этом простом скетче все описанные блоки действуют совместно таким образом, чтобы робот двигался вперёд в течение трёх секунд и затем остановился. Поскольку электродвигатели должны работать синхронно в течение всей программы, блок pulse Invert Motor следует использовать только в разделе настройки в блоке цикла. Блок pulse Set Motor Powers заставляет сразу оба электродвигателя вращаться с мощностью 50 %. Для завершения скетча используйте блок pulse End вместо обращения программы в бесконечный цикл (рис. 48).

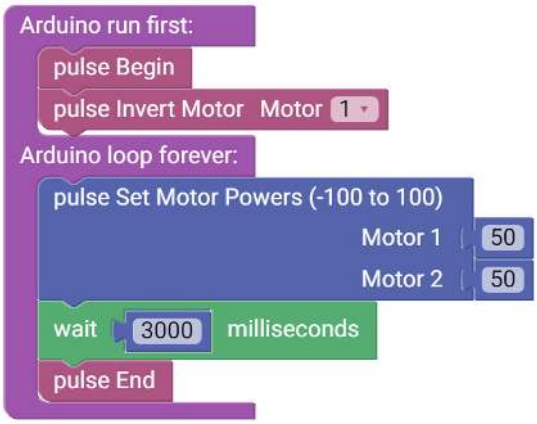


Рисунок 48

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч, который заставит базового робота проехать вперёд и остановиться. Вспомните, чему научились в ходе предыдущих упражнений и попытайтесь сделать так, чтобы базовый робот некоторое время двигался вперёд, а затем задним ходом вернулся в исходную точку.

У вас есть всё необходимое для определения скорости, поскольку вы можете узнать расстояние, пройденное за некоторое время. В скетче используется заданное значение времени, а расстояние, на которое переместился робот, можно измерить линейкой. При изменении параметра мощности должно измениться и пройденное за то же время расстояние.

С учётом этого поставьте задачу, отметив заданное расстояние на полу. На основе собранных данных запрограммируйте базового робота так, чтобы он как можно ближе подъехал к указанной отметке, не заезжая за неё.

Связь с реальным миром

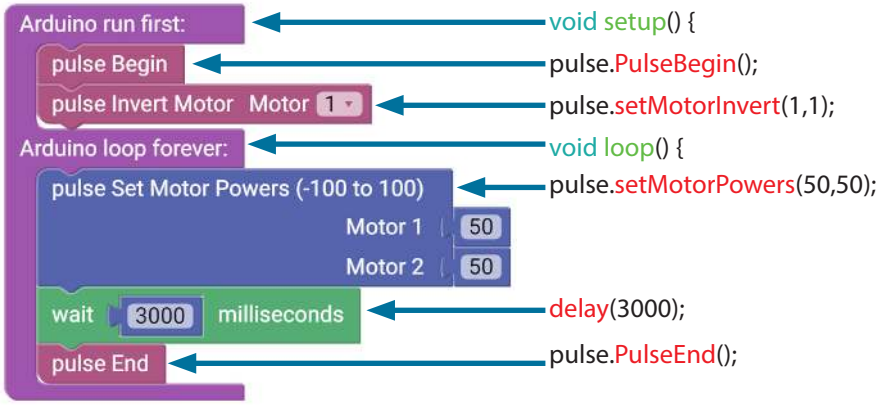
Поезда — это такие средства передвижения, которые приводятся в действие электродвигателями и могут двигаться только вперёд или назад. Их электродвигатели вращаются под воздействием усилия, направленного вперёд или назад. Поезд не может выполнять крутые повороты, поэтому пути плавно загибаются, а не поворачивают под углом 90 градусов.

Профессии: железнодорожный рабочий, машинист, инженер-электровозостроитель

Связь с точными и естественными науками

- Физика
  - Кинетическая энергия
  - Частота вращения
- Технология
  - Прямой привод
  - Электродвигатели постоянного тока
- Техническое конструирование
  - Рулевой механизм
  - Программирование
- Математика
  - Обороты
  - Длина окружности колеса

Связь блоков с текстом



Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

pulse.setMotorInvert(1,1);
}

void loop() {
pulse.
setMotorPowers(50,50);
delay(3000);
pulse.PulseEnd();
}
```

Примечание: Запрограммируйте два электродвигателя на вращение в противоположных направлениях с помощью блока pulse Invert Motor.

Упражнение 8: Движение по кругу

Введение

В ходе восьмого упражнения понадобятся знания об электродвигателях для создания новых схем движения. Способность двигаться по прямой, конечно, важна, но нужно расширять возможности и осваивать технику поворотов. В этом упражнении требуется научить базового робота PULSE ездить по кругу, используя разную мощность синхронно работающих электродвигателей.

Необходимое оборудование

Открытие программы

Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты
Examples > **Activity_8**. Откроется окно нового скетча под заголовком Activity_8 (рис. 49).

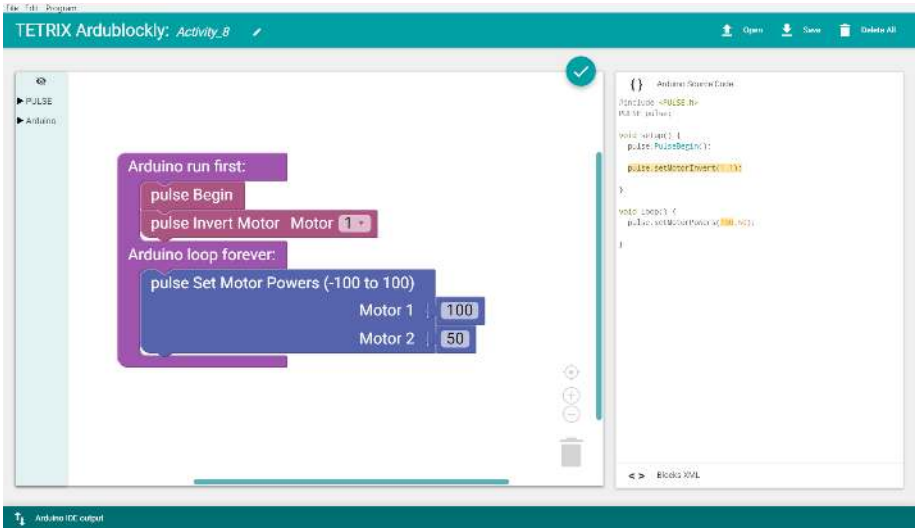


Рисунок 49

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После этого отсоедините кабель USB и поставьте базового робота на пол.

Поставьте его так, чтобы колёса находились спереди. Если робот едет назад вместо того, чтобы ехать вперёд, поменяйте местами провода электродвигателей постоянного тока в портах контроллера.

Для запуска скетча нажмите на зелёную кнопку "Пуск". Проследите за направлением и продолжительностью движения базового робота. Остановите скетч, нажав на красную кнопку сброса параметров/остановки. Соответствуют ли его действия приведённому описанию скетча?

Совет: Возможно, базовому роботу будет трудно преодолевать сцепление с ковровым покрытием. По возможности, испытывайте робота на гладкой поверхности.

Совет: Чтобы робот смог проехать полный круг, потребуется около двух метров свободного пространства на полу.

Дальнейшее изучение

В данном скетче не используются никакие новые блоки, однако он обеспечит более глубокое понимание синхронного действия электродвигателей для выполнения определённых движений.

Все блоки в составе скетча действуют вместе таким образом, чтобы базовый робот ездил по кругу. Поскольку электродвигатели должны работать синхронно в течение всей программы, блок pulse Invert Motor используется только в разделе настройки в блоке цикла. Блок pulse Set Motor Powers даёт команду двум электродвигателям вращаться с разной частотой в пределах одной функции. Заданные значения мощности составляют 100 % для одного двигателя и 50 % для второго, что приводит к движению по кругу.

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч, который заставит базового робота ездить по кругу другого размера. Вспомните, чему научились в ходе предыдущих упражнений, и попробуйте увеличить или уменьшить диаметр круга, экспериментируя с разными параметрами. Поставьте перед собой более сложные задачи: как научить робота ездить по траектории в форме овала или восьмерки?

Связь с реальным миром

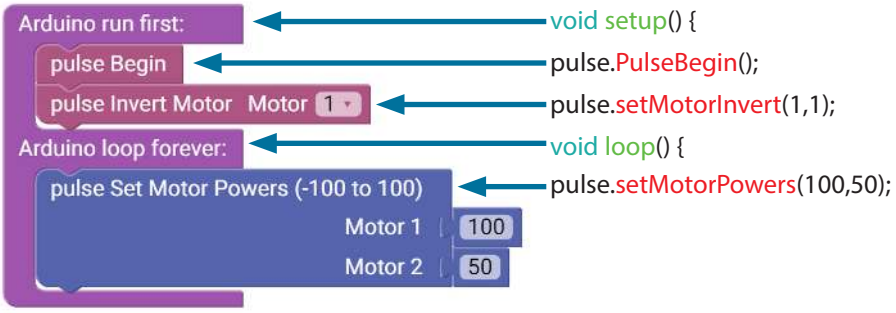
Если вы когда-либо бывали рядом с полем для гольфа, наверно, вы видели работу автоматических систем полива. Эти оросительные системы двигаются по кругу, распыляя воду на 360 градусов. Они запрограммированы на включение и выключение в разное время. Важно, чтобы электродвигатель поворачивал распылительную головку для полного покрытия площади распыления.

Профессии: озеленитель, механик сельскохозяйственного оборудования, мелиоратор

Связь с точными и естественными науками

- Физика
  - Мощность
  - Частота вращения
- Технология
  - Управление приводом
  - Точность привода
- Техническое конструирование
  - Конструирование машин
  - Геометрические фигуры
- Математика
  - Радиус
  - Диаметр

Связь блоков с текстом



Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

pulse.setMotorInvert(1,1);
}

void loop() {
pulse.setMotorPowers(100,50);
}
```

Примечание: Мощность вращения электродвигателя 1 в два раза больше мощности вращения электродвигателя 2.

Упражнение 9: Движение по квадратной траектории

Введение

Чтобы далее усовершенствовать технику управления движением базового робота PULSE, нужно научить робота поворачивать на 90 градусов. С помощью таких поворотов базовый робот сможет ездить по квадрату.

Необходимое оборудование

Открытие программы

Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты
Examples > **Activity_9**. Откроется окно нового скетча под заголовком Activity_9 (рис. 50).

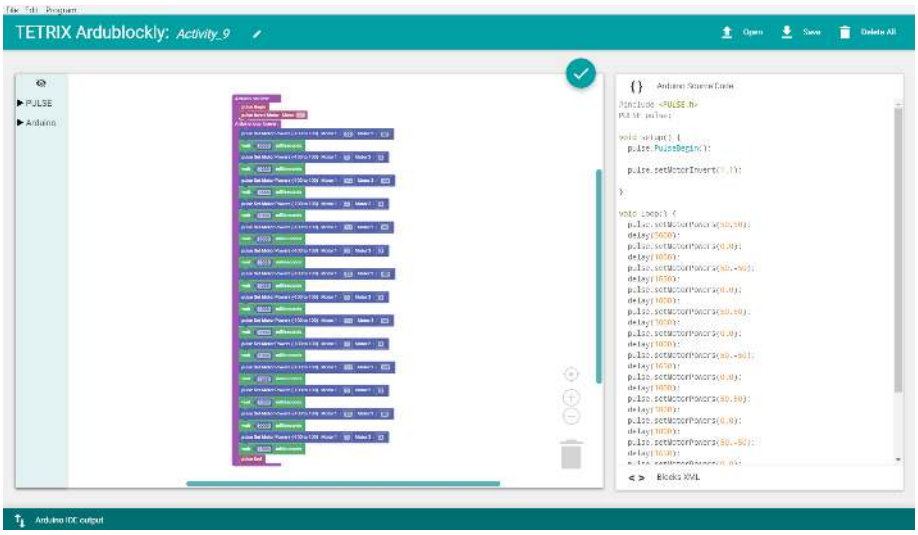


Рисунок 50

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После этого отсоедините кабель USB и поставьте базового робота на пол.

Для запуска скетча нажмите на зелёную кнопку "Пуск". Проследите за направлением и продолжительностью движения базового робота. Соответствуют ли его действия приведённому описанию скетча?

Совет: В зависимости от поверхности, траектория движения робота может оказаться не идеально квадратной.

Дальнейшее изучение

Если у вас на экране много блоков, вам может быть трудно рассматривать целую программу. Воспользуйтесь функцией увеличения и уменьшения. Также некоторые блоки можно преобразовать в строку. Блок pulse Set Motor Powers можно преобразовать в строку. Ниже представлен пример по изменению вида этого блока (рис. 51). Это не повлияет на саму программу.

Рисунок 51

Несмотря на то, что этот скетч отличается большим размером, он состоит из функций, которые вы использовали ранее. Здесь собрано несколько последовательных действий для создания одного сложного действия, в данном случае движения по квадрату.

Программа имеет очень важную для понимания особенность: для программирования робота на движение по квадратной траектории используется точный расчёт. Точный расчёт выражается в использовании параметра времени как основы для управления электродвигателем. Например, для поворота направо электродвигателям дают команду вращаться в противоположных направлениях с мощностью 50 % в течение 1650 миллисекунд.

Вы предполагаете, что если электродвигатель будет вращаться с определённой частотой в течение определённого времени, траектория робота должна приблизиться к повороту направо под углом 90 градусов. Однако такой прогноз не всегда точен ввиду того, что уровень заряда аккумуляторной батареи может различаться. Также на результат влияет проскальзывание колёс по поверхности, на которой запускают робота. Таким образом, чтобы добиться правильной траектории поворота, нужен точный расчёт.

Все функции в составе скетча действуют вместе таким образом, чтобы базовый робот ездил по квадратной траектории (рис. 52). Поскольку электродвигатели должны работать синхронно в течение всей программы, блок pulse Invert Motor следует использовать только в разделе настройки в блоке цикла. Блок pulse Set Motor Powers даёт команду двум электродвигателям вращаться с разной частотой в пределах одной функции.

Структура программы

Рисунок 52

Совет: Если не получилось достичь прямых углов поворота, отрегулируйте время поворота. Например, время можно изменить со 1650 на 1300. Заряд аккумуляторной батареи может повлиять на выполнение поворотов.

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч для движения базового робота по квадратной траектории. Вспомните, чему научились в ходе предыдущих упражнений, и попробуйте увеличить или уменьшить квадрат, экспериментируя с разными параметрами. Поставьте перед собой задачу повышенной сложности: пусть траектория движения робота будет сложнее квадрата. Например, с помощью точных расчётов можно заставить робота изображать буквы алфавита или проехать по простому лабиринту.

Связь с реальным миром

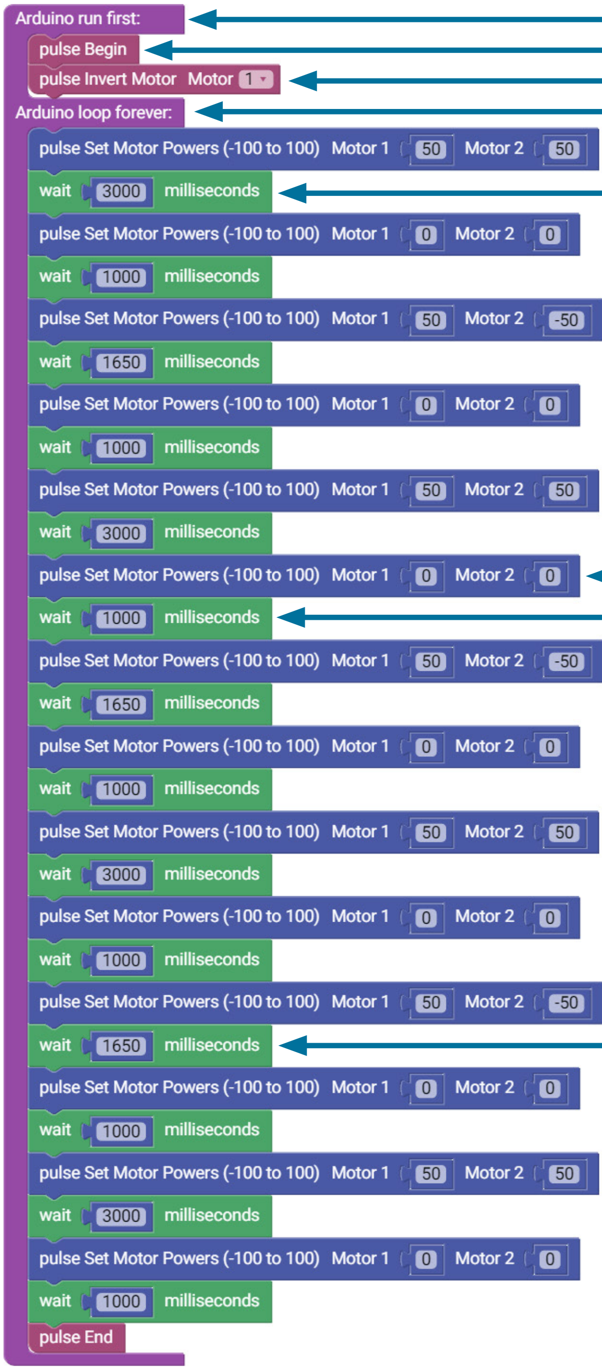
Некоторые транспортные средства периодически проезжают по одному и тому же маршруту. К примеру, на складе такой транспорт может отвезти груз в три разных места и затем вернуться на базу. Далее он принимает новый груз и вновь движется по квадратной траектории, развозя его в пункты назначения.

Профессии: техник по составлению маршрутов, авиадиспетчер, инженер-технолог

Связь с точными и естественными науками

- Физика
  - Расстояние
  - Время
- Технология
  - Микропроцессорное время
  - Радиус поворота
- Техническое конструирование
  - Мощность электродвигателя
  - Торможение
- Математика
  - Углы квадрата
  - Измерение углов

Связь блоков с текстом




```
void setup() {
pulse.PulseBegin();
pulse.setMotorInvert(1,1);
void loop() {
pulse.setMotorPowers(50,50);
delay(3000);
```

Примечание: Электродвигатели вращаются вперёд в течение 3 секунд.

```
pulse.setMotorPowers(0,0);
delay(1000);
```

Примечание: Электродвигатели тормозят на 1 секунду.

```
pulse.setMotorPowers(50,-50);
delay(1650);
```

Примечание: Электродвигатели вращаются в противоположных направлениях в течение 1,65 секунд, при этом робот делает поворот.

Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

pulse.setMotorInvert(1,1);
}

void loop() {
pulse.setMotorPowers(50,50);
delay(3000);
pulse.setMotorPowers(0,0);
delay(1000);
pulse.setMotorPowers(50,-50);
delay(1650);
pulse.setMotorPowers(0,0);
delay(1000);
pulse.setMotorPowers(50,50);
delay(3000);
pulse.setMotorPowers(0,0);
delay(1000);
pulse.setMotorPowers(50,-50);
delay(1650);
pulse.setMotorPowers(0,0);
delay(1000);
pulse.setMotorPowers(50,50);
delay(3000);
pulse.setMotorPowers(0,0);
delay(1000);
pulse.setMotorPowers(50,-50);
delay(1650);
pulse.setMotorPowers(0,0);
delay(1000);
pulse.setMotorPowers(50,50);
delay(3000);
pulse.setMotorPowers(0,0);
delay(1000);
pulse.PulseEnd();
}
```

Упражнение 10: Упрощение скетча для движения по квадратной траектории

Введение

Помните, как в упражнении 9 вы научили робота двигаться по квадратной траектории? В ходе следующего упражнения вы запрограммируете его на те же действия более эффективным способом.

Необходимое оборудование

Открытие программы

Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты
Examples > **Activity_10**. Откроется окно нового скетча под заголовком Activity_10 (рис. 53).

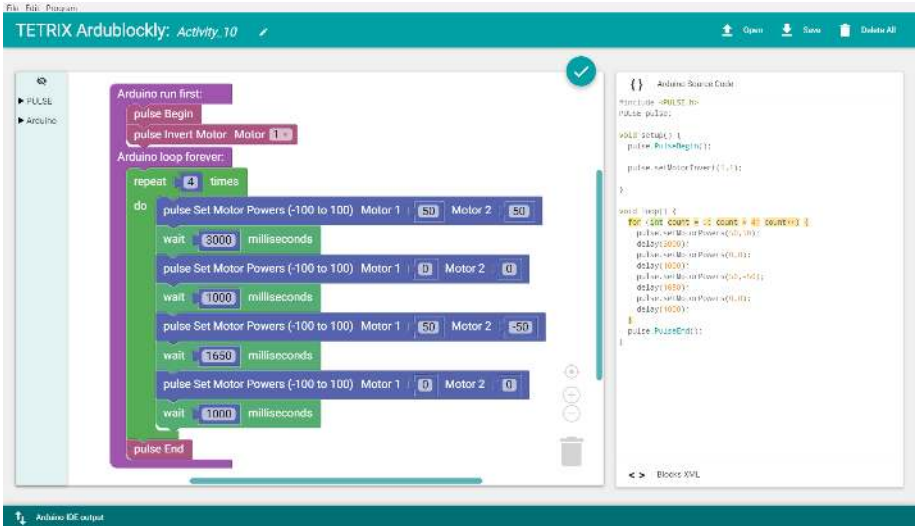


Рисунок 53

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После этого отсоедините кабель USB и поставьте базового робота на пол.

Для запуска скетча нажмите на зелёную кнопку "Пуск". Проследите за направлением и продолжительностью движения базового робота. Соответствуют ли его действия приведённому описанию скетча?

Совет: В зависимости от поверхности, траектория движения робота может оказаться не идеально квадратной.

Дальнейшее изучение

В этом скетче использована одна новая программная структура в форме блока "повтор действия" (repeat-do). Блок repeat-do обеспечивает заданное число повторений включённых в него команд. При этом программа выглядит проще и её легче просматривать.

Блок repeat-do используется для повторения заключенной в нём части кода. Для подсчёта числа повторов до окончания цикла применяют счётчик приращений. Такой блок нужен для того, чтобы сократить количество повторений цикла (рис. 54).

Рисунок 54

Программируя базового робота на движение по квадрату в упражнении 9, вы указывали каждое действие отдельной строкой, в результате программа стала слишком длинной и казалась сложнее, чем на самом деле.

Начните программу с блока repeat-do и задайте необходимое количество повторений цикла. В нашей программе цикл повторится четыре раза. В цикл включены следующие действия робота: движение вперёд и поворот (рис. 55). Такой элемент программы называется вызываемой функцией, поскольку он находится вне блока "настройка-цикл" (setup-loop).

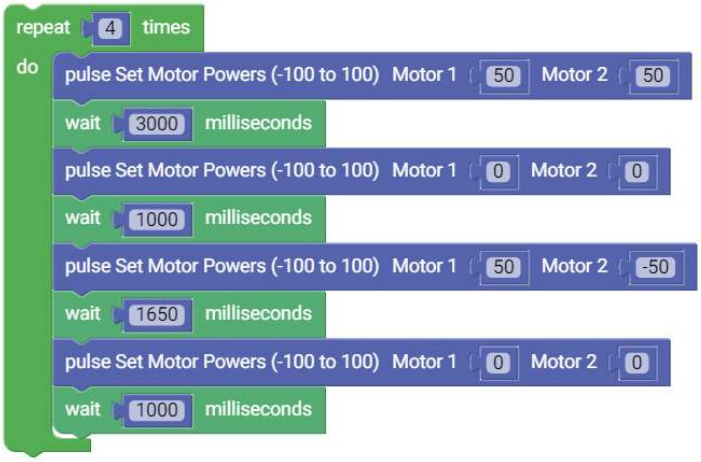


Рисунок 55

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч для движения базового робота по квадратной траектории, используя блок repeat-do и вызываемые функции. Либо для усложнения задачи придумайте траектории более сложные, чем квадрат, например, прямоугольники или шестиугольники.

Вспомните, чему научились в ходе предыдущих упражнений. Сделайте шаг вперёд, создавая роботов собственной уникальной конструкции и применяя полученные знания о кодировании новыми и неожиданными способами.

Связь с реальным миром

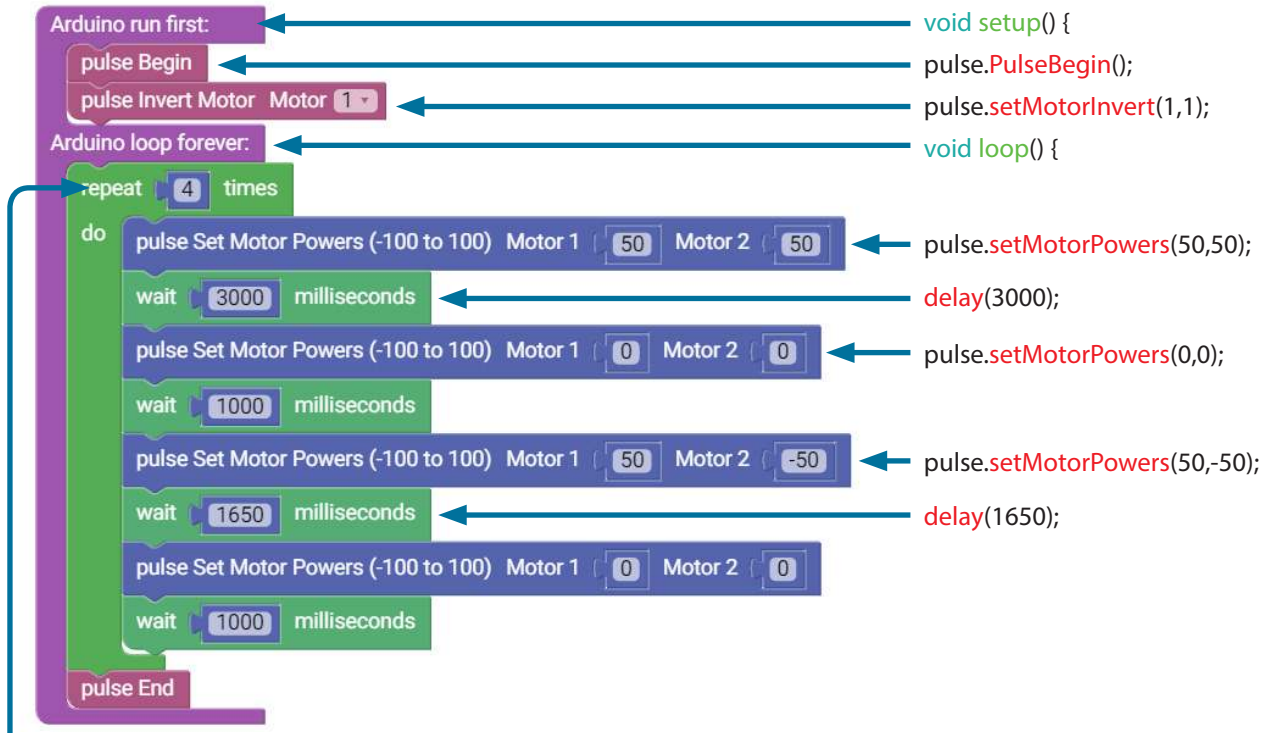
На предприятиях роботы складывают изделия в упаковки. Они должны повторять этот процесс снова и снова. Роботы оборудованы датчиками, которые позволяют определить, есть ли под их рабочим механизмом упаковка.

Профессии: программист ПЛК (программируемые логические контроллеры), программист станков с ЧПУ, техник производства

Связь с точными и естественными науками

- Физика
  - Расстояние
  - Скорость
- Технология
  - Микропроцессоры
  - Радиус поворота
- Техническое конструирование
  - Схемы следования по маршруту
  - Программирование
- Математика
  - Измерение углов
  - Углы квадрата

Связь блоков с текстом



Примечание: Это инициализация, которая происходит в первую очередь и только один раз.

Примечание: Это условие, которое проверяется в каждом цикле.

Примечание: Накопление значений выполнено и условие проверяется снова.

Исходный код Arduino

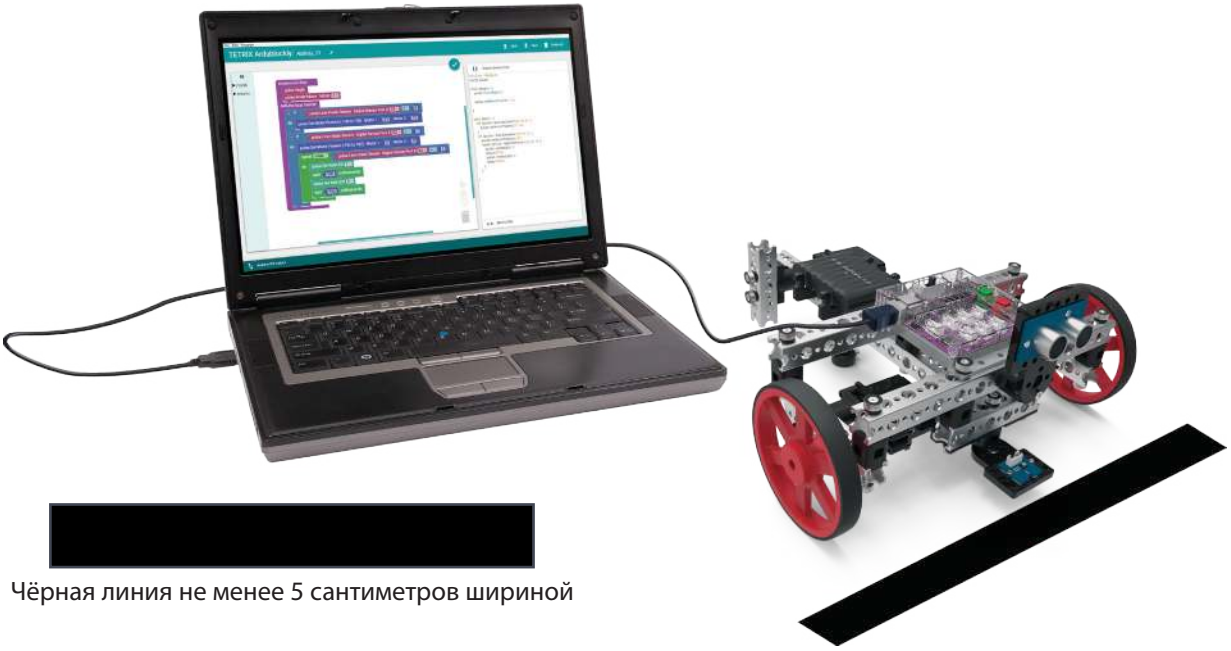
```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

pulse.setMotorInvert(1,1);
}
void loop() {
for (int count = 0; count < 4; count++)
{
pulse.setMotorPowers(50,50);
delay(3000);
pulse.setMotorPowers(0,0);
delay(1000);
pulse.setMotorPowers(50,-50);
delay(1650);
pulse.setMotorPowers(0,0);
delay(1000);
}
pulse.PulseEnd();
}
```

Упражнение 11: Движение до линии и остановка

Введение

Работа над упражнением начнётся с добавления датчика линии к базовому движению вперёд. Благодаря этому базовый робот PULSE сможет останавливаться, учитывая окружающие условия. В этом упражнении базовый робот будет двигаться передним ходом и останавливаться на линии.

Необходимое оборудование



Чёрная линия не менее 5 сантиметров шириной

Открытие программы

Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты
Examples > **Activity_11**. Откроется окно нового скетча под заголовком Activity_11 (рис. 56).

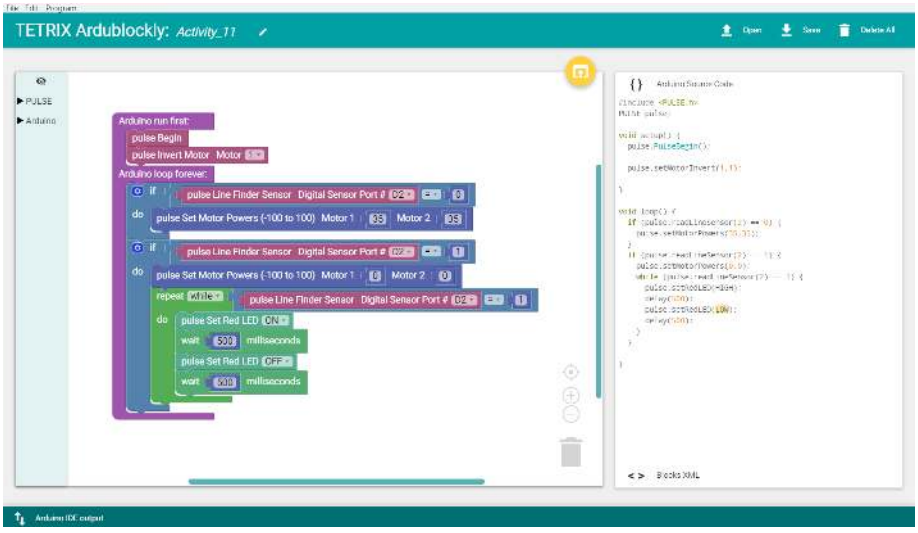


Рисунок 56

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Датчик линии должен быть подсоединён к порту D2. Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После этого отсоедините кабель USB и поставьте базового робота на пол.

Поставьте базового робота на белой или светоотражающей поверхности, направив его в сторону чёрной линии или неотражающей поверхности, которая находится приблизительно в метре от него. Для выполнения этого кода вам потребуется тёмная полоса на белой или отражающей поверхности, по которой будет двигаться робот. Поверхность можно изготовить из глянцевого картона, пенокартона, плакатного щита или несколько листов бумаги, склеенных друг с другом.

Для запуска скетча нажмите на зелёную кнопку "Пуск". Наблюдайте за действиями робота. Остановите скетч, нажав на красную кнопку сброса параметров/остановки. Соответствуют ли его действия приведённому описанию скетча?

Совет: В зависимости от поверхности, траектория движения робота может оказаться не идеально квадратной.

Устранение неисправностей: Проверьте, подсоединён ли датчик линии к правильному порту и настроен ли он надлежащим образом для обнаружения линии. Может потребоваться регулировка по высоте либо настройка с помощью регулировочного винта, расположенного на обратной стороне модуля датчика. Чтобы проверить работу датчика, вручную подвигайте робота вперёд и назад по границе чёрной и белой поверхностей. Красный светодиод должен гореть, когда датчик линии находится на белой поверхности, и гаснуть при въезде на чёрную поверхность.

Дальнейшее изучение

В этом скетче используется два блока "если..., то" (if-do), которые вам знакомы по упражнению 4, а также вводится новая программная структура — блок цикла "пока" (while()).

При выполнении программы блоки if-do проверяют условие датчика линии. Первый блок if-do даёт команду базовому роботу двигаться по белой или светоотражающей поверхности с мощностью 35 %. Показание датчика линии на таких поверхностях будет равно 0 (рис. 57).

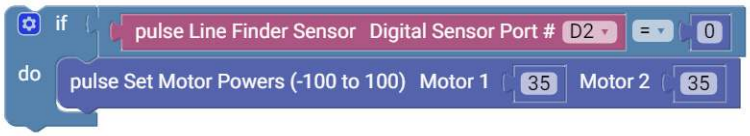


Рисунок 57

Второй блок if-do состоит из двух частей. Первая часть даёт базовому роботу команду остановиться при въезде на чёрную линию или неотражающую поверхность. При обнаружении чёрной поверхности показание датчика линии становится равным 1. Соответственно, мощность электродвигателя становится равной 0 и происходит торможение (рис. 58).

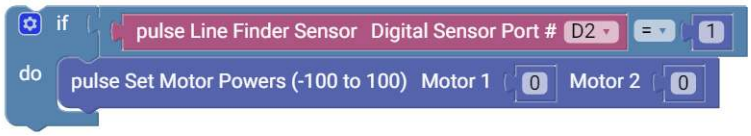


Рисунок 58

Вторая часть блока if-do выполняет блок цикла while(). Блок while() проверяет заданное в блоке условие и будет непрерывно выполнять цикл до тех пор, пока оно не станет ложным. В рассматриваемой программе действие, указанное в цикле while(), будет повторяться всё время, пока показание датчика линии равно 1 (рис. 59). Красный светодиод будет продолжать мигать до изменения переменных в скетче.

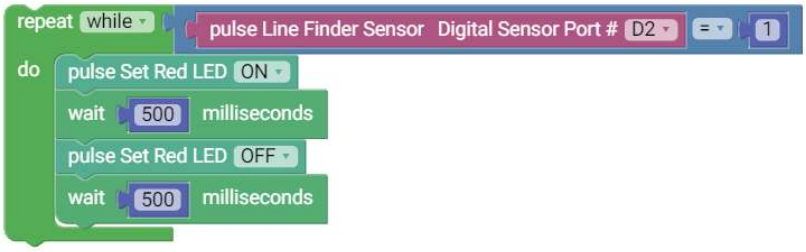


Рисунок 59

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч, который заставит базового робота подъехать к чёрной линии и остановиться. Вспомните, чему научились в ходе предыдущих упражнений. Попробуйте запрограммировать базового робота на выполнение разных действий, когда он подъезжает к линии, например, сдавать назад или менять курс движения.

Связь с реальным миром

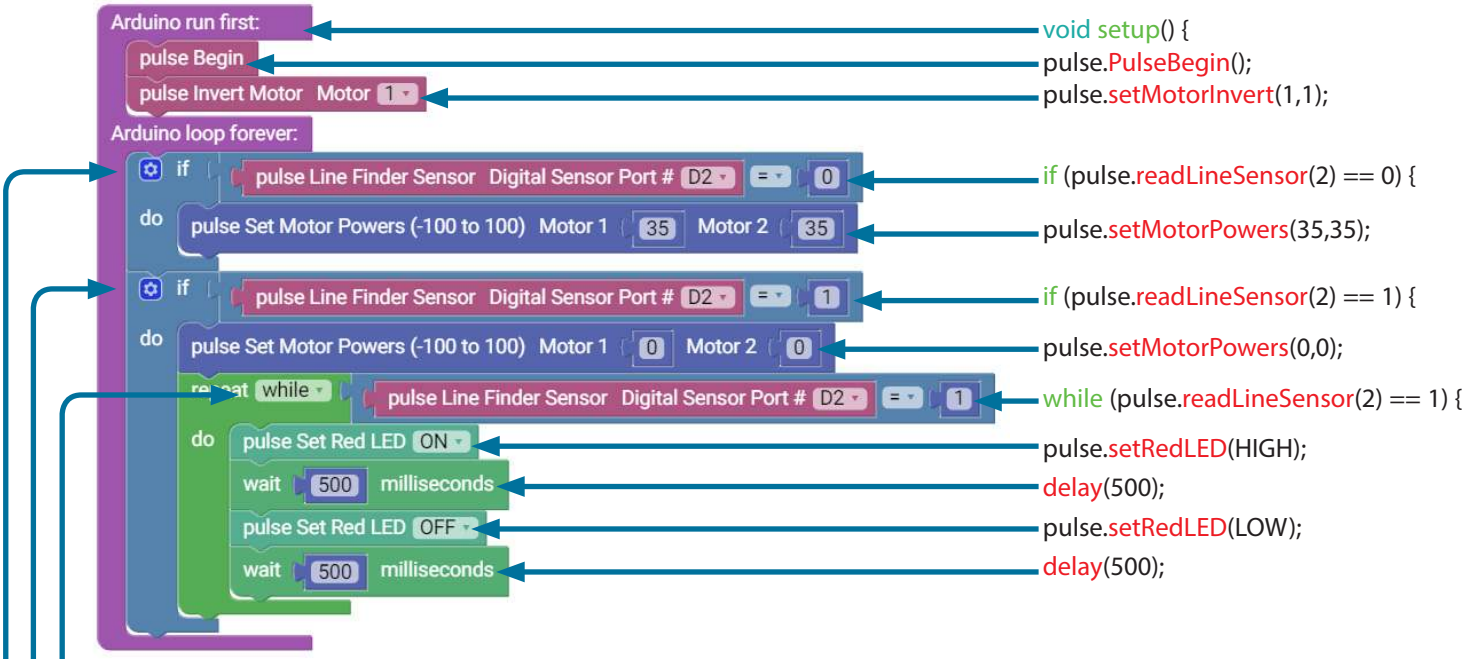
С появлением беспилотных автомобилей потребовались системы, которые могут распознавать разметку на дороге. Когда автомобиль подъезжает к перекрестку, он должен остановиться на линии. Такое действие можно выполнить с помощью датчика, обнаруживающего линию.

Профессии: разработчик автомобильного программного обеспечения, инженер-электрик, цифровой скульптор

Связь с точными и естественными науками

- Физика
  - o Скорость
  - o Ускорение
- Технология
  - o Датчики
  - o Цикл while()
- Техническое конструирование
  - o Решение задач
  - o Движение по линии
- Математика
  - o Логические операторы
  - o Расчёты скорости

Связь блоков с текстом



Примечание: Этот цикл if() приводит в действие электродвигатели, если показание датчика равно 0.

Примечание: Этот цикл if() останавливает электродвигатели, если показание датчика равно 1.

Примечание: Этот цикл while() заставляет красный светодиод мигать, пока показание датчика равно 1.

Исходный код Arduino

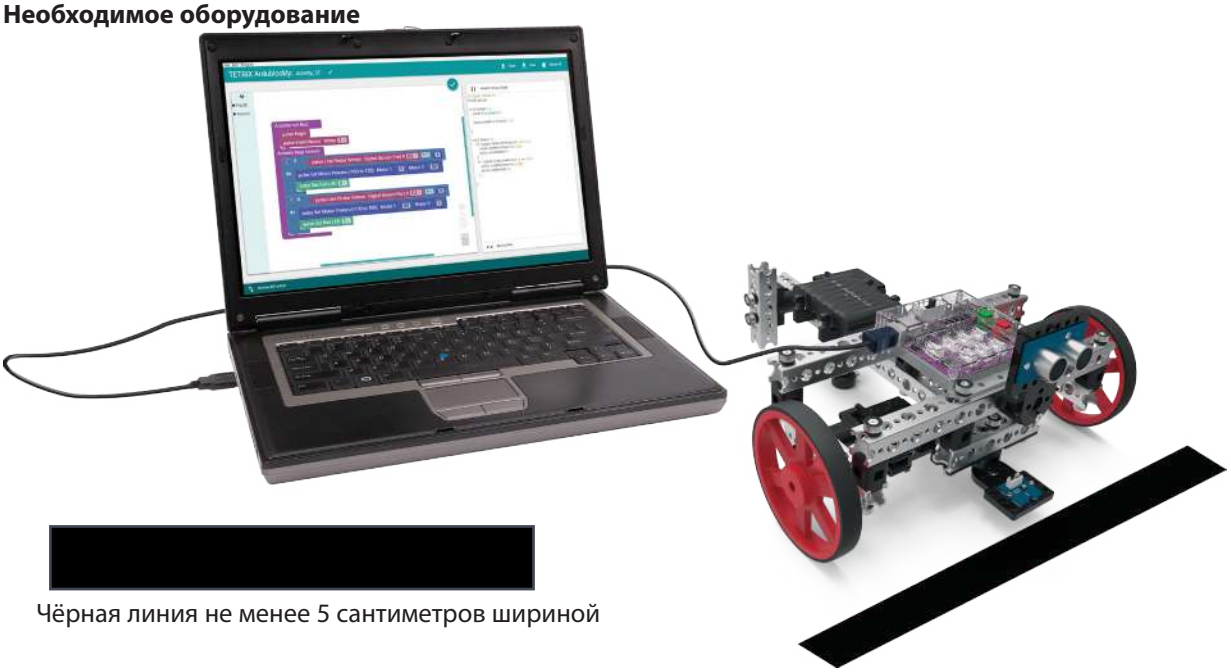
```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

pulse.setMotorInvert(1,1);
}
void loop() {
if (pulse.readLineSensor(2) == 0) {
pulse.setMotorPowers(35,35);
}
if (pulse.readLineSensor(2) == 1) {
pulse.setMotorPowers(0,0);
while (pulse.readLineSensor(2) ==
1) {
pulse.setRedLED(HIGH);
delay(500);
pulse.setRedLED(LOW);
delay(500);
}
}
}
```

Упражнение 12: Движение по линии

Введение

Для этого упражнения обобщите все знания, которые вы усвоили в ходе предыдущей работы, и примените их немного иначе, чтобы робот мог выполнять новые действия. Таким образом вы научите базового робота PULSE двигаться по линии.

Необходимое оборудование



Чёрная линия не менее 5 сантиметров шириной

Открытие программы

Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты
Examples > **Activity_12**. Откроется окно нового скетча под заголовком Activity_12 (рис. 60).

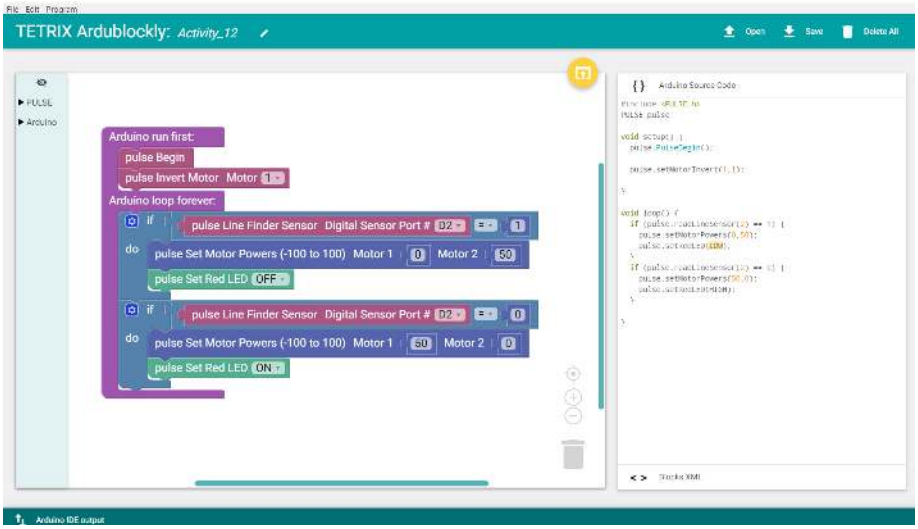


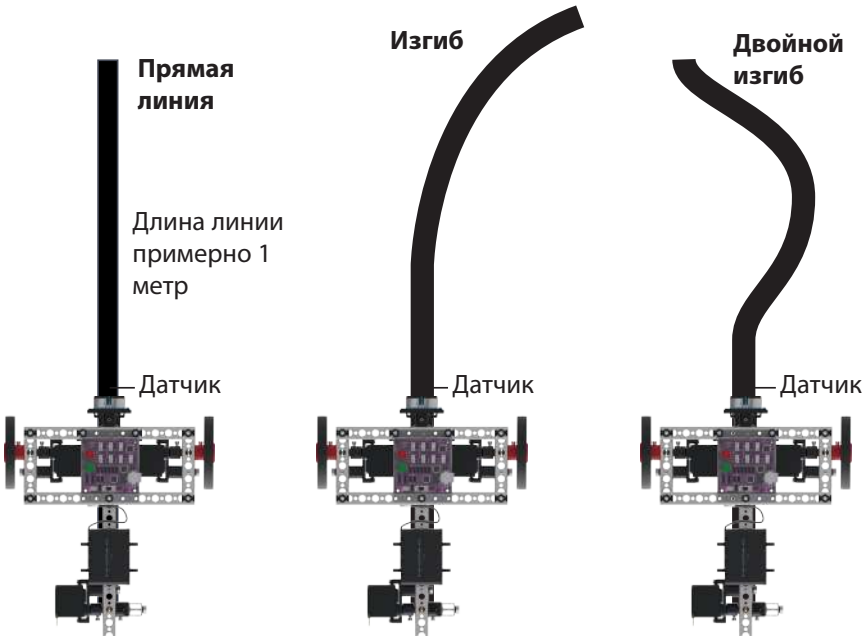
Рисунок 60

Выполнение управляющего кода

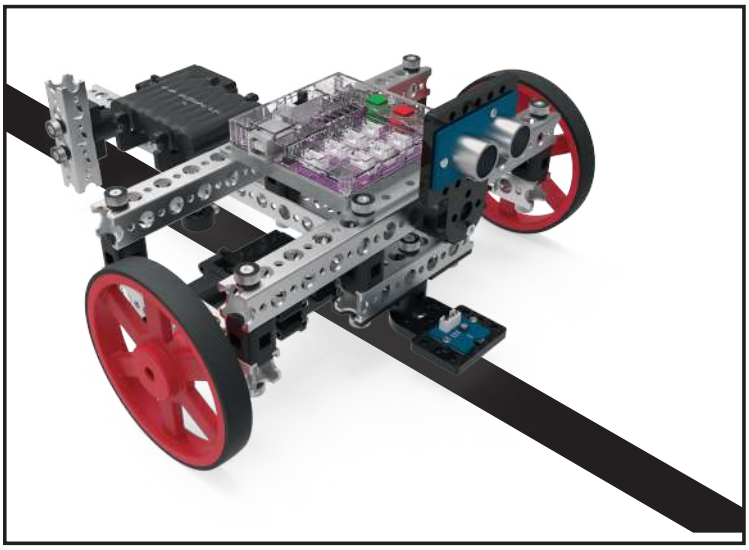
Перед загрузкой скетча в контроллер PULSE проверьте соединения. Датчик линии должен быть подсоединён к порту D2. Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После включения светодиода отсоедините кабель USB и поставьте базового робота на белую или отражающую поверхность.

Для выполнения этого кода вам потребуется тёмная полоса на белой или отражающей поверхности, по которой будет двигаться робот. Поверхность можно изготовить из глянцевого картона, пенокартона, плакатного щита или несколько листов бумаги, склеенных друг с другом.

Для чёрной ленты подойдет обычная изолента. Дорожка для робота может быть как прямой, так и с изгибами. Точность робота при проезде поворотов ограничена, поэтому не создавайте крутых поворотов на трассе.



Желательно поставить базового робота так, чтобы датчик линии находился чуть сбоку от направляющей линии. Для запуска скетча нажмите на зелёную кнопку "Пуск". Наблюдайте за действиями робота. Остановите скетч, нажав на красную кнопку сброса параметров/остановки. Соответствуют ли его действия приведённому описанию скетча?



Совет: Может возникнуть необходимость подтолкнуть робота для начала движения из-за проскальзывания колёс на гладкой поверхности.

Устранение неисправностей: Проверьте, подсоединён ли датчик линии к правильному порту и настроен ли он надлежащим образом для обнаружения линии. Может потребоваться регулировка по высоте либо настройка с помощью регулировочного винта, расположенного на обратной стороне модуля датчика. Чтобы проверить работу датчика, вручную подвигайте робота вперёд и назад по границе чёрной и белой поверхностей. Красный светодиод должен гореть, когда датчик линии находится на белой поверхности, и гаснуть при въезде на чёрную поверхность.

Дальнейшее изучение

В этом скетче используется два блока if-do. В каждом блоке if-do выполняется два действия.

Первое действие связано с электродвигателями. Блок if-do даёт базовому роботу команду повернуть, задавая значение мощности для одного электродвигателя и останавливая другой в зависимости от показаний датчика линии. Блоки if-do выполняют противоположные действия. Благодаря этому базовый робот способен выполнить ряд поворотов, следуя по линии.

Второе действие блока if-do связано с красным светодиодным индикатором на корпусе контроллера PULSE. Блок if-do включает или выключает красный светодиод в зависимости от показаний датчика линии. Включение и выключение синхронизируется с вращением электродвигателей.

Действия в обоих блоках if-do совместно обеспечивают следование базового робота по линии с подачей зрительных сигналов в виде мигания красного светодиода.

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч, позволяющий базовому роботу следовать по чёрной линии. Вспомните, чему научились в ходе предыдущих упражнений. Попробуйте заставить базового робота ехать быстрее, но при этом по-прежнему точно по линии, или измените код так, чтобы базовый робот двигался по линии в обратном направлении.

Связь с реальным миром

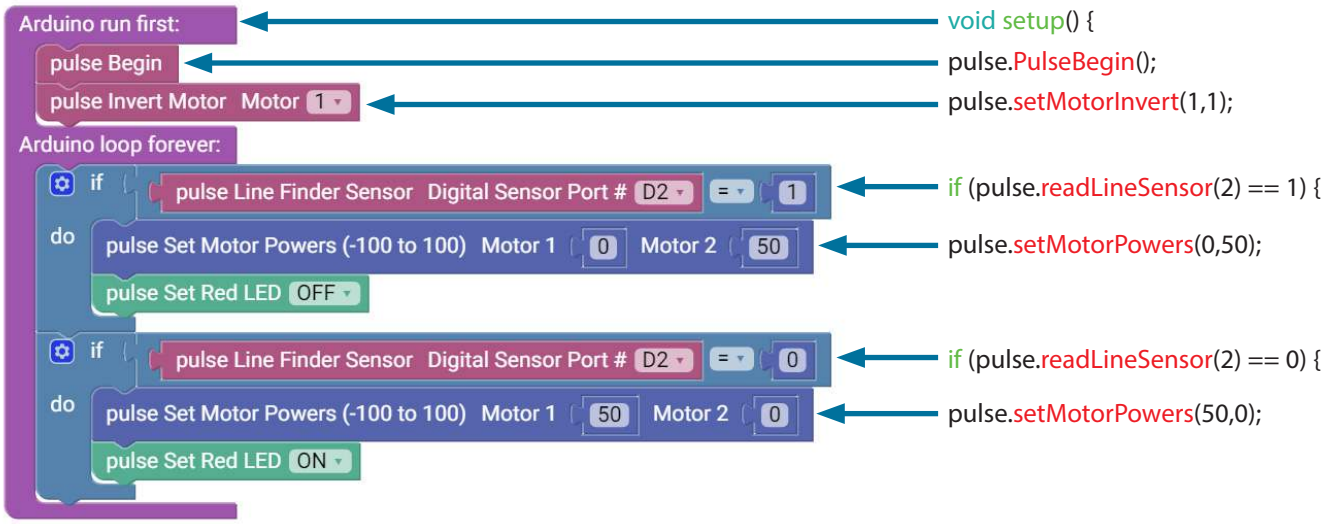
Работа детских аттракционов нового поколения основывается на датчиках следования за линией. Например, благодаря ему миниатюрные электромобили катают малышей по дорожке, а игрушечные поезда ездят по траектории в форме восьмерки.

Профессии: управляющий парка развлечений, технический директор, мастер по системам электроуправления аттракционами

Связь с точными и естественными науками

- Физика
  - o Отражение света
  - o Угол падения
- Технология
  - o Пороговые значения
  - o Аналоговая и цифровая информация
- Техническое конструирование
  - o Станки для обработки края
  - o Беспилотный транспорт
- Математика
  - o Логические операторы
  - o Числовые величины

Связь блоков с текстом



Примечание: Одно колесо будет вращаться, второе — тормозить, в зависимости от показаний датчика линии.

Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

pulse.setMotorInvert(1,1);
}

void loop() {
if (pulse.readLineSensor(2) == 1) {
pulse.setMotorPowers(0,50);
pulse.setRedLED(LOW);
}
if (pulse.readLineSensor(2) == 0) {
pulse.setMotorPowers(50,0);
pulse.setRedLED(HIGH);
}

}
```

Упражнение 13: Движение до стены и остановка

Введение

В ходе этого упражнения вы расширите полученные знания, работая с ультразвуковым датчиком. Мы запрограммируем базового робота PULSE таким образом, чтобы он подъезжал к стене и останавливался на заданном расстоянии.

Необходимое оборудование

Открытие программы

Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты
Examples > **Activity_13**. Откроется окно нового скетча под заголовком Activity_13 (рис. 61).

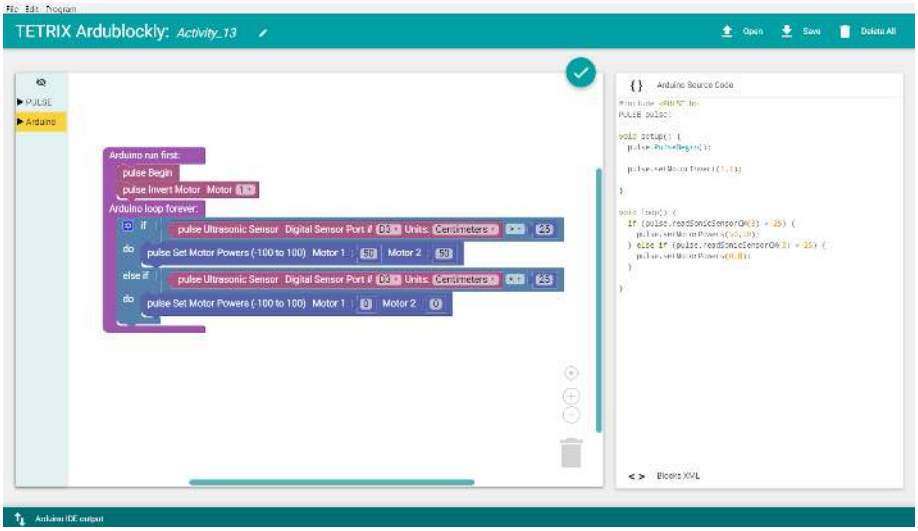


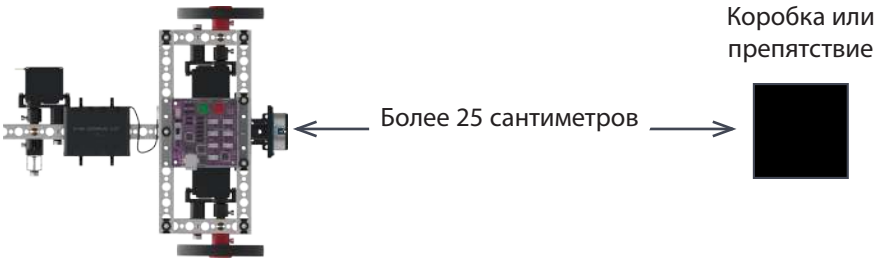
Рисунок 61

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Ультразвуковой датчик должен быть подключен к порту D3. Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После этого отсоедините кабель USB и поставьте базового робота на пол.

Направьте базового робота в сторону предмета или стены, находящейся на расстоянии не менее 25 см от него. Для запуска скетча нажмите на зелёную кнопку "Пуск". Наблюдайте за действиями робота. Что произойдет, если базовый робот или предмет сдвинется до окончания скетча?

Остановите скетч, нажав на красную кнопку сброса параметров/остановки. Соответствуют ли его действия приведённому описанию скетча?



Дальнейшее изучение

В этом скетче используется блок if-else. Блок if-else позволяет более эффективно организовать выполнение кода, чем базовый блок if. С его помощью можно объединить несколько проверок таким образом, чтобы они выполнялись одновременно. Другими словами, часть блока с оператором if даёт команду выполнить определённое действие, если условие истинно, в то время как часть блока с оператором else даёт команду выполнить другое действие, если условие ложно.

В этом скетче блок if-else проверяет показания ультразвукового датчика, и если датчик находится на расстоянии более 25 сантиметров от предмета, блок устанавливает мощность электродвигателей на 50 %. Во второй части блока if-else действует та же проверка, но если датчик находится на расстоянии менее 25 см от предмета, даётся команда на выполнение другого действия, в данном случае — на остановку электродвигателей.

Блок if допускает различные варианты программирования. Можно применить базовый блок if-do ("если..., то") и далее программа будет выполнять действие, включённое в цикл. В ходе этого упражнения используется блок if-do; else if-do ("если..., то; иначе если..., то"). Другой вариант — применение блока if-do, else-do ("если..., то, иначе..., то"). Для изменения вида блока нужно нажать на синий значок шестерни (рис. 62 и 63) и затем перетащить требуемый оператор под блок if справа (рис. 64).

Устранение неисправностей: Проверьте, подключён ли ультразвуковой датчик к правильному порту и верно ли выполнено соединение. Следует учитывать, что если предмет не обладает достаточной площадью поверхности или имеет неровную поверхность, датчик может его не обнаружить или неправильно определить расстояние до него.

Рисунок 62

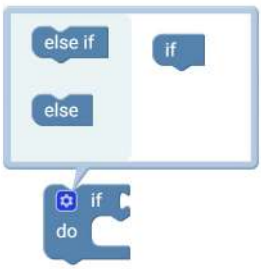


Рисунок 63

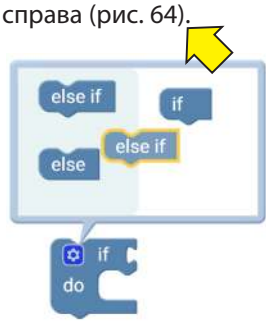


Рисунок 64

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч, который заставит базового робота подъехать к стене или препятствию и остановиться. Вспомните, чему научились в ходе предыдущих упражнений. Попробуйте изменить направление, диапазон обнаружения или скорость. Также можно проверить, предметы какого размера или формы лучше всего обнаруживает датчик и на каком расстоянии.

Связь с реальным миром

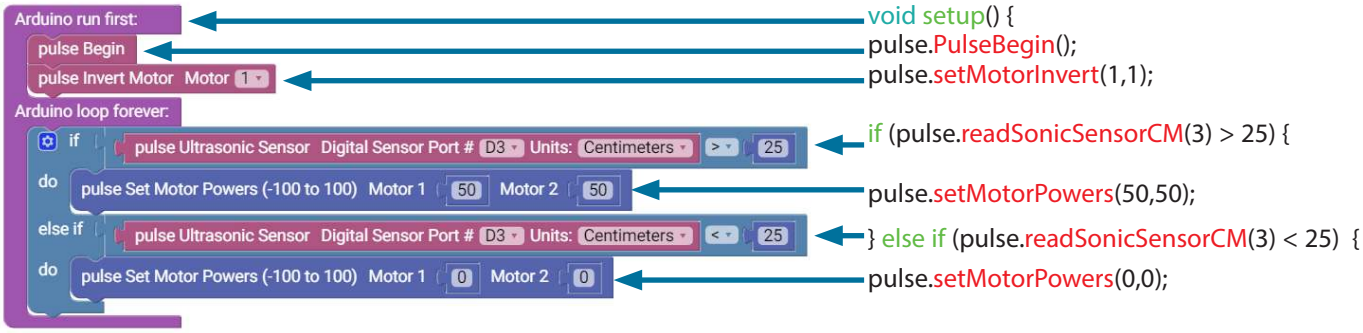
Роботы-пылесосы очищают полы жилых помещений при помощи датчиков, определяющих наличие препятствий. Если на пути робота попадается мебель, программа даёт ему команду сдать назад и развернуться, чтобы избежать столкновения с препятствием. Также роботы-пылесосы оборудованы датчиками, обнаруживающими ступени. По их сигналу робот останавливается до того, как произойдет падение.

Профессии: инженер по вакуумному оборудованию, технолог по вакуумному напылению, чертежник механического оборудования

Связь с точными и естественными науками

- Физика
  - Скорость звука
  - Звуковые волны
- Технология
  - Калибровка датчика
  - Ультразвук
- Техническое конструирование
  - Обнаружение препятствий
  - Торможение
- Математика
  - Расстояние до предметов
  - Метрические и стандартные единицы измерения

Связь блоков с текстом



Примечание: Робот будет ехать до тех пор, пока не обнаружит препятствие на расстоянии 25 см сантиметров, после чего остановится.

Исходный код Arduino

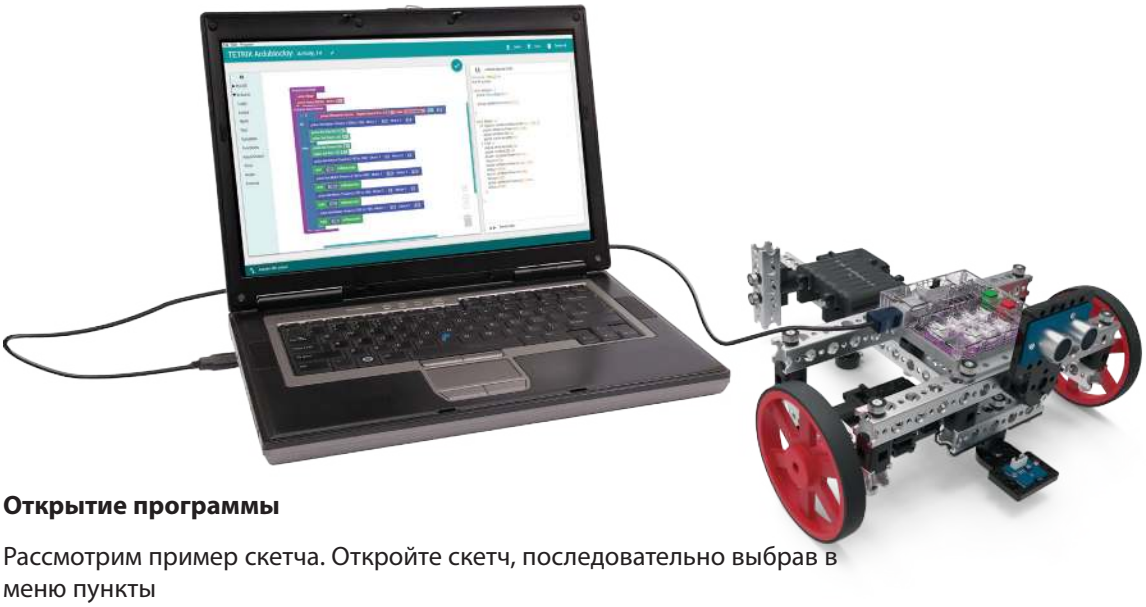
```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

pulse.setMotorInvert(1,1);
}

void loop() {
if (pulse.readSonicSensorCM(3) > 25) {
pulse.setMotorPowers(50,50);
} else if (pulse.readSonicSensorCM(3) < 25)
{
pulse.setMotorPowers(0,0);
}

}
```

Упражнение 14: Объезд препятствий

Введение

Это упражнение предусматривает расширенное применение блока if-else путём добавления действий. Благодаря этому базовый робот PULSE сможет объезжать препятствия на своём пути.

Необходимое оборудование



Открытие программы

Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты
Examples > **Activity_14**. Откроется окно нового скетча под заголовком Activity_14 (рис. 65).

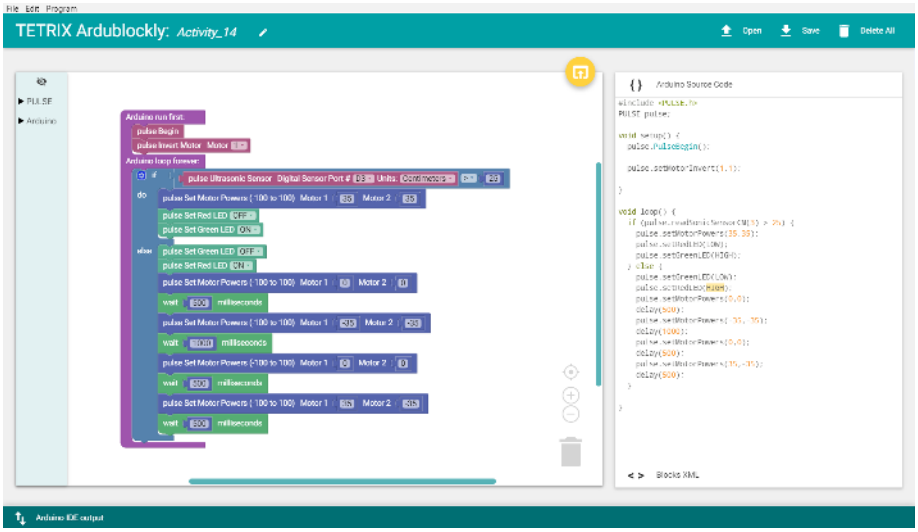


Рисунок 65

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Ультразвуковой датчик должен быть подключен к порту D3. Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После этого отсоедините кабель USB и поставьте базового робота на пол.

Во время выполнения кода разместите перед роботом препятствие на расстоянии, существенно превышающем диапазон обнаружения, равный 25 сантиметрам. Для этой цели хорошо подходят картонные коробки. Слишком тяжелые объекты могут привести к повреждениям робота в случае, если произойдет столкновение робота с препятствием.

Для начала нажмите на зелёную кнопку "Пуск". Робот поедет вперёд и будет двигаться до тех пор, пока не обнаружит препятствие на своём пути на расстоянии 25 см.

После обнаружения препятствия робот остановится, сдаст назад, повернёт направо и продолжит движение. Наблюдайте за действиями робота. Остановите скетч, нажав на красную кнопку сброса параметров/остановки. Соответствуют ли его действия приведённому описанию скетча?

Дальнейшее изучение

В рассматриваемом скетче применяется блок if-else и присутствует несколько действий, заключённых в цикл.

В каждую часть блока if-else включено по несколько действий. При выполнении раздела if базовый робот движется вперёд с мощностью 35 % и включенным зелёным светодиодом, в то время как датчик проверяет, нет ли препятствия. При выполнении раздела else, если базовый робот обнаружил препятствие, зелёный светодиод выключается и включается красный светодиод. При этом базовый робот останавливается, сдаёт назад и поворачивает направо. Затем цикл повторяется.

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч, позволяющий базовому роботу объезжать препятствия. Вспомните, чему научились в ходе предыдущих упражнений.

Проведите эксперименты с различными препятствиями или сделайте полосу препятствий из нескольких предметов. Можно изменять код-пример любым образом, меняя реакцию и действия робота при объезде препятствий.

Совет: Если робот приближается к препятствию под углом, обнаружение препятствия может быть затруднено.

Устранение неисправностей: Проверьте, подключён ли ультразвуковой датчик к правильному порту и верно ли выполнено соединение. Следует учесть, что если предмет очень мал или имеет неровную поверхность, датчик может его не обнаружить или неправильно определить расстояние до него.

Связь с реальным миром

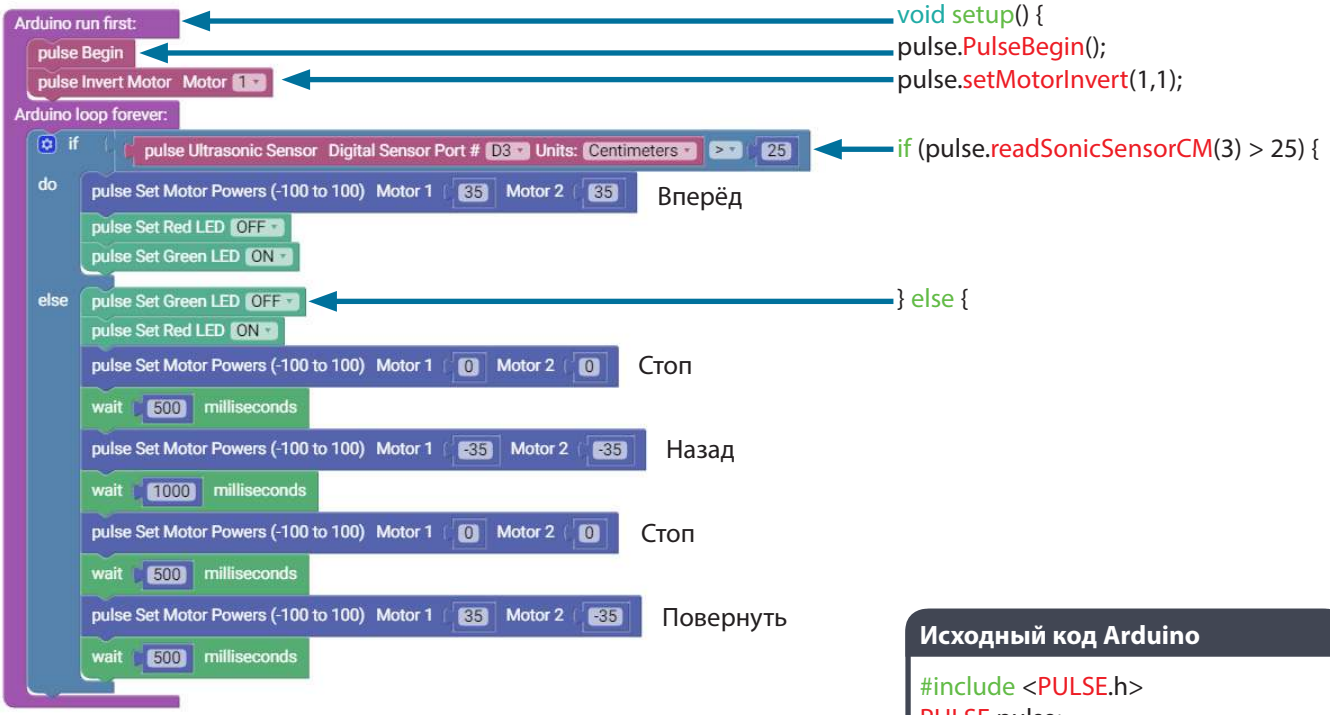
Некоторые автомобили новых поколений оборудованы передними датчиками и способны определять наличие препятствий впереди. Представьте, что на дорогу перед вашим автомобилем выскочил олень. Автомобиль автоматически остановится благодаря тому, что датчики обнаружили животное. Авария от столкновения с оленем будет предотвращена.

Профессии: инженер-испытатель, инженер по разработке материалов, лаборант-метролог

Связь с точными и естественными науками

- Физика
  - Звуковые волны
  - Эхолокация
- Технология
  - Логические операторы
  - Применение электродвигателей
- Техническое конструирование
  - Объезд препятствий
  - Рулевое управление
- Математика
  - Построение графиков
  - Углы

Связь блоков с текстом



Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

pulse.setMotorInvert(1,1);
}

void loop() {
if (pulse.readSonicSensorCM(3) > 25)
{
pulse.setMotorPowers(35,35);
pulse.setRedLED(LOW);
pulse.setGreenLED(HIGH);
} else {
pulse.setGreenLED(LOW);
pulse.setRedLED(HIGH);
pulse.setMotorPowers(0,0);
delay(500);
pulse.setMotorPowers(-35,-35);
delay(1000);
pulse.setMotorPowers(0,0);
delay(500);
pulse.setMotorPowers(35,-35);
delay(500);
}

}
```

Упражнение 15: Сочетание датчиков

Введение

Это упражнение подытоживает все предыдущие упражнения. Базовый робот PULSE будет двигаться по линии, следя за наличием препятствий. При обнаружении препятствия базовый робот остановится, поднимет балку-рычаг и будет ждать до тех пор, пока препятствие не уберут.

Необходимое оборудование

Открытие программы

Рассмотрим пример скетча. Откройте скетч, последовательно выбрав в меню пункты
Examples > **Activity_15**. Откроется окно нового скетча под заголовком Activity_15 (рис. 66).

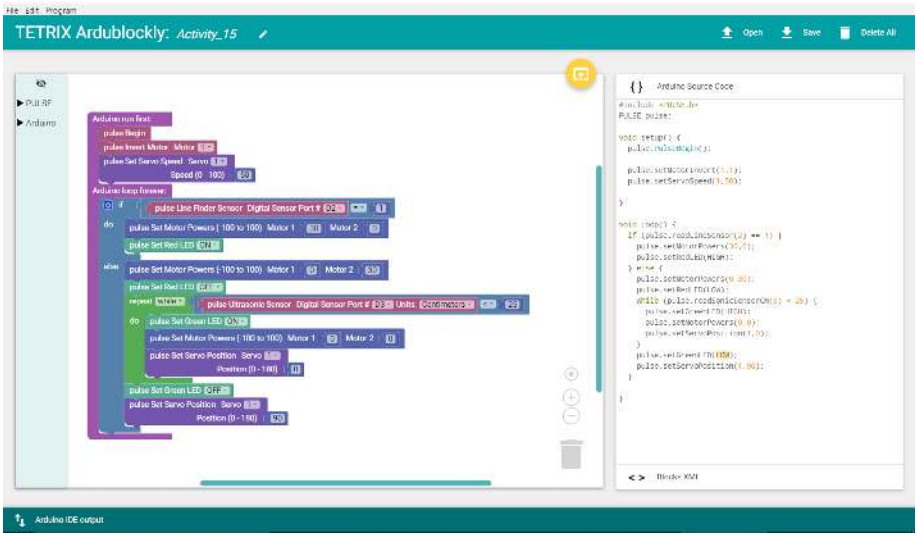


Рисунок 66

Выполнение управляющего кода

Перед загрузкой скетча в контроллер PULSE проверьте соединения. Датчик линии должен быть подсоединён к порту D2, а ультразвуковой датчик — к порту D3.

Датчик линии необходим роботу, чтобы следовать по линии, а ультразвуковой датчик поможет ему обнаруживать препятствия. Согласно этому скетчу-примеру, робот будет двигаться по чёрной линии на белой поверхности и следить, нет ли на его пути препятствий.

При обнаружении предмета на расстоянии менее 25 см робот остановится, чтобы избежать столкновения, далее он поднимет рычаг и будет ждать, пока препятствие не уберут. После устранения препятствия движение может быть продолжено.

Загрузите скетч в контроллер. При этом включится зелёный светодиодный индикатор, что означает готовность к выполнению кода. После этого отсоедините кабель USB и поставьте базового робота на пол.

Желательно поставить базового робота на белой или светоотражающей поверхности так, чтобы датчик линии находился чуть сбоку от направляющей линии.

Во время выполнения кода разместите перед роботом препятствие на расстоянии, существенно превышающем диапазон обнаружения, равный 25 сантиметрам. Для этой цели хорошо подходят картонные коробки. Слишком тяжелые объекты могут привести к повреждениям робота в случае, если произойдет столкновение робота с препятствием.

Для запуска скетча нажмите на зелёную кнопку "Пуск". Наблюдайте за действиями робота. Также можно экспериментировать, перемещая препятствия и наблюдая, как на это отреагирует робот.

Остановите скетч, нажав на красную кнопку сброса параметров/остановки. Соответствуют ли его действия приведённому описанию скетча?

Совет: В случае затруднений в работе с датчиками см. советы по устранению неисправностей в предыдущих упражнениях.

Дальнейшее изучение

В этом скетче используется блок if-else и блок цикла while(), которые рассматривались раньше. В программе отсутствуют новые элементы. Программа основана на сочетании элементов из предыдущих упражнений.

Блок if-else в скетче обеспечивает такое же движение по линии, что и скетч с двумя блоками if в упражнении 12, но с большей степенью контроля. Блок цикла while() отслеживает показания ультразвукового датчика и заставляет робота реагировать при обнаружении предмета в пределах 25 см. При этом базовый робот останавливается, поднимает рычаг при помощи сервопривода и ждет, пока предмет не уберут с его пути. После того, как препятствие будет убрано, базовый робот опустит рычаг и продолжит первоначальное действие — движение по линии.

Дополнительное упражнение

На основе этого примера попытайтесь создать новый скетч, позволяющий базовому роботу двигаться по чёрной линии и останавливаться перед препятствиями. Вспомните, чему научились в ходе предыдущих упражнений.

Попробуйте создать скетч, который заставит базового робота двигаться по линии до препятствия, объезжать препятствие и продолжать движение по линии дальше, а не ждать, пока препятствие уберут.

Связь с реальным миром

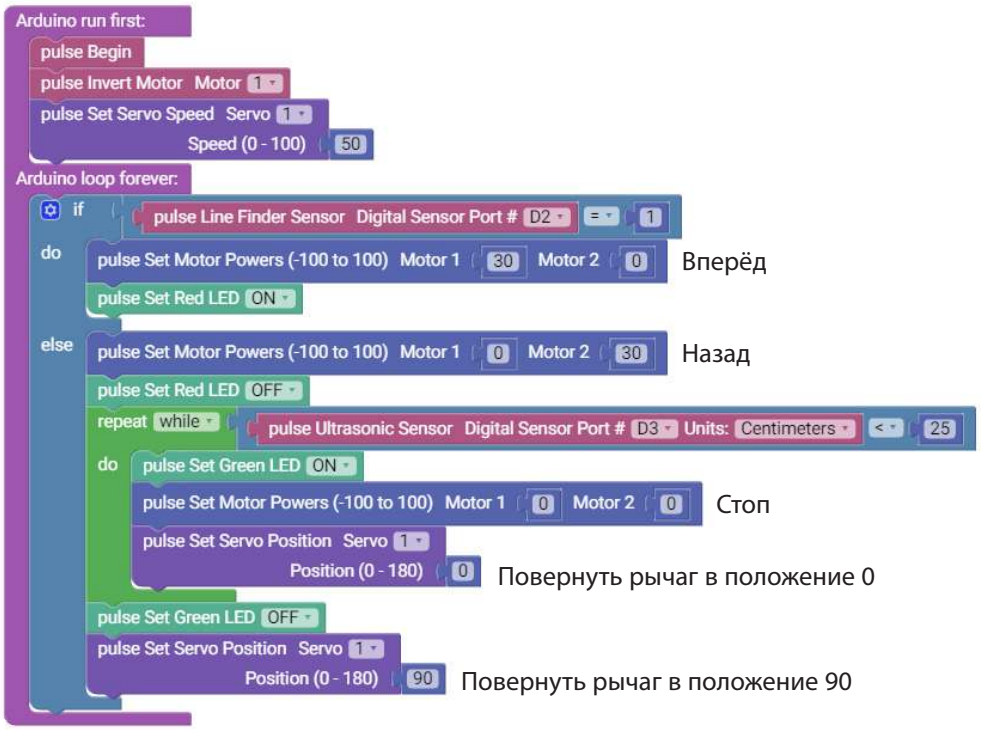
В медицине применяется много разных датчиков. В их числе датчики, определяющие температуру тела, пульс, частоту дыхания и сердечный ритм. Иногда роботы даже проводят хирургические операции или делают снимки внутри человеческого тела.

Профессии: техник-хирург, медицинский техник, инженер-биомедик

Связь с точными и естественными науками

- Физика
  - o Сенсорные устройства
  - o Функции
- Технология
  - o Показания датчика
  - o Разработка маршрута
- Техническое конструирование
  - o Решение задач
  - o Интеграция сенсорных данных
- Математика
  - o Распределение данных
  - o Продолжительность

Связь блоков с текстом



Исходный код Arduino

```
#include <PULSE.h>
PULSE pulse;

void setup() {
pulse.PulseBegin();

pulse.setMotorInvert(1,1);
pulse.setServoSpeed(1,50);
}

void loop() {
if (pulse.readLineSensor(2) == 1) {
pulse.setMotorPowers(30,0);
pulse.setRedLED(HIGH);
} else {
pulse.setMotorPowers(0,30);
pulse.setRedLED(LOW);
while (pulse.readSonicSensorCM(3) < 25)
{
pulse.setGreenLED(HIGH);
pulse.setMotorPowers(0,0);
pulse.setServoPosition(1,0);
}
pulse.setGreenLED(LOW);
pulse.setServoPosition(1,90);
}

}
```

Собирай, программируй, испытывай, учись . . .

двигайся вперёд!

Вы собрали робота, запрограммировали контроллер PULSE, испытали и усовершенствовали технические решения. Что же дальше?

Настоящее руководство предназначено для обучения основам сборки и программирования робота TETRIX PRIME с контроллером PULSE и программным обеспечением *TETRIX Ardublockly* и *Arduino (IDE)*. Однако это всего лишь начало. Существует множество дополнительных источников и возможностей для сборки более крупных роботов, способных выполнять более сложные действия. После того, как вы овладели основами, вас ограничивают только пределы собственной фантазии! Ниже указано несколько популярных источников, с которых можно начать.

Также не забудьте ознакомиться с приложениями к настоящему руководству, где представлена самая разнообразная дополнительная информация.

Хочется более сложных задач?

Выполните упражнения из пройденного руководства в текстовом ПО *Arduino (IDE)*. Перепишите графические программы в текстовом виде. Вносите изменения и выполняйте дополнительные упражнения, пользуясь таким форматом кода. При этом время занятий по руководству может существенно увеличиться.

Подробности про ПО *Arduino (IDE)*, а также тонкости программирования и рекомендации:

Контроллер TETRIX PULSE разработан с учётом архитектуры Arduino, соответственно, для него подходят языки программирования, поддерживающие Arduino. Пользователи PULSE могут обратиться к онлайн-сообществу Arduino, которое опубликовало множество вспомогательных материалов для быстрого освоения программирования. Более подробную информацию можно найти по адресу: **www.arduino.cc**.

Дополнительная информация о конструкторах TETRIX:

Перечисленные в настоящем руководстве детали и элементы — это всего лишь часть конструктора TETRIX PRIME. Более подробная информация о дополнительных строительных элементах, деталях движущихся механизмов и комплектующих, при помощи которых можно расширить ваш набор, представлена на сайте **www.TETRIXrobotics.com**. Также там можно посмотреть обучающие видеоролики и загрузить вспомогательные материалы.

Учебный план

В учебном плане указана примерная продолжительность работы над каждым из упражнений при выполнении всех его разделов. Все учащиеся работают в своём темпе, поэтому время, в действительности затрачиваемое на каждое из упражнений, будет разным. Примерная продолжительность приводится с учётом выполнения всех дополнительных упражнений. Из учебного плана исключено время, необходимое на изучение вводного раздела руководства по программированию.

Урок	Продолжительность
Введение	60 минут
Упражнение 1: Привет, мир!	20 минут
Упражнение 2: Вращение электродвигателей постоянного тока	20 минут
Упражнение 3: Вращение сервоприводов	20 минут
Упражнение 4: Использование датчика линии	30 минут
Упражнение 5: Первое знакомство с ультразвуковым датчиком	30 минут
Упражнение 6: Сборка базового робота PULSE	60 минут
Упражнение 7: Движение вперёд	60 минут
Упражнение 8: Движение по кругу	60 минут
Упражнение 9: Движение по квадратной траектории	60 минут
Упражнение 10: Упрощение скетча для движения по квадратной траектории	60 минут
Упражнение 11: Движение до линии и остановка	60 минут
Упражнение 12: Движение по линии	60 минут
Упражнение 13: Движение до стены и остановка	60 минут
Упражнение 14: Объезд препятствий	60 минут
Упражнение 15: Сочетание датчиков	60 минут

Суммарная продолжительность: 780 минут (13 часов)

Принятые во внимание стандарты

Единые общеобразовательные стандарты штатов (США) — английский язык трудового обучения/развитие речи

Класс 6

- SL.6.1
  - SL.6.1.A
  - SL.6.1.B
  - SL.6.1.C
  - SL.6.1.D
- SL.6.4
- RST.6-8.3
- RST.6-8.4

Класс 7

- SL.7.1
  - SL.7.1.A
  - SL.7.1.B
  - SL.7.1.C
  - SL.7.1.D
- SL.7.4
- RST.6-8.5
- RST.6-8.6

Класс 8

- SL.8.1
  - SL.8.1.A
  - SL.8.1.B
  - SL.8.1.C
  - SL.8.1.D
- SL.8.4
- RST.6-8.10

Единые общеобразовательные стандарты штатов (США) — математика

Математические навыки

- MP1
- MP2

Математические знания

- 6.NS.C.5

Стандарты Международной ассоциации преподавателей технических наук и инженерного дела (ITEEA) по технологической грамотности

Классы 6-8

- 1.F
- 1.G
- 1.H
- 2.M
- 2.N
- 2.O
- 2.V
- 3.D
- 3.E
- 3.F
- 4.D
- 7.D
- 7.E
- 8.E
- 9.H
- 10.H
- 12.H
- 12.I
- 12.J
- 12.K
- 14.G
- 15.G
- 16.G
- 16.H
- 17.H
- 18.F
- 18.G
- 18.I
- 19.F

Толковый словарь

прерывание: принудительная остановка программы или скетча (нажатием на красную кнопку сброса параметров/остановки на корпусе контроллера PULSE)

***ПО Arduino (IDE)*:** программное обеспечение с открытым исходным кодом, используемое для программирования устройств, которые поддерживают язык Arduino, например, таких, как контроллеры PULSE

автономный: робот, совершающий действия под управлением микропроцессора, без вмешательства человека

балка: прямоугольная алюминиевая деталь с расположенными определённым образом отверстиями, используется как конструктивный элемент

действие: физическое действие робота

вызываемая функция: набор команд, выполняемый внутри основной программы или скетча

код: команды программы

оператор сравнения: код, сравнивающий значения двух переменных

условие: код, где используется как логика. так и сравнение значений

точный расчёт: процесс, позволяющий определить будущее положение на основе направления движения и ранее пройденного расстояния

описание: выражение, представляющее собой название определённой функции, переменной или постоянной

задержка (delay): код, устанавливающий интервалы времени между операторами или действиями

дифференциальный привод: система привода, в котором направление хода задаётся изменением частоты вращения электродвигателей, расположенных на противоположных сторонах робота

рабочий орган: прикреплённое к роботу устройство, взаимодействующее с окружающей средой

выполнение: выполнение команды или программы

ожидаемое действие: действие робота, которое он предположительно должен совершить при выполнении набора команд

цикл for: цикл со счётчиком, набор команд, повторяющийся заданное количество раз

функция: процедура в составе компьютерной программы

HIGH: "высокое" значение, обычно устанавливается для включенного переключателя или датчика

"если/иначе" (if/else): оператор-команда, осуществляющий выбор между двумя вариантами действий на основе сравнения значений

"если/то" (if/then): оператор-команда, выполняющий действие на основе сравнения значений

накопление значений: приращение до определённого количества, обычно применяющееся в программных циклах

инициализация: установка исходного значения переменной или начальной точки конфигурации кода

целое число: положительное или отрицательное целое число, включая ноль

светодиод: светоизлучающий диод

датчик линии: устройство, способное распознавать контраст между светлой и тёмной поверхностью и использовать его для управления движением робота

логика: наука о принципах формализованного рассуждения или верных умозаключений; логика руководствуется математическими принципами

LOW: "низкое" значение, обычно устанавливается для выключенного переключателя или датчика

основной цикл: также известный как "void loop" в программировании на основе языка C, этот цикл содержит командный код, управляющий роботом

миллисекунды: одна тысячная секунды

нейтральное положение: центральное положение сервопривода

препятствие: предмет, оказавшийся на пути робота

параметр: фактор или условие, которое должно быть достигнуто при окончательном решении задачи

псевдокод: описание шагов (с помощью простого языка), которые должны выполняться в компьютерной программе

отражённый свет: часть светового потока, которая идёт от источника до предмета и возвращается обратно к источнику

датчик: электронное устройство, применяемое для распознавания света, звука, движения или энергии с последующей передачей информации в процессор

сервопривод: вращающийся привод с точной регулировкой углового положения и частоты вращения, состоит из двигателя и подключенного к нему датчика, передающего информацию о положении привода

скетч: набор команд (компьютерная программа) на платформе ПО *Arduino (IDE)*

стандартный сервопривод: сервопривод, способный вращаться до определённого углового положения (обычно в диапазоне 0...180°) благодаря внутренней обратной связи, осуществляемой при помощи датчика

оператор: одна команда кода; инструкция для выполнения

синтаксис: точный способ написания компьютерного кода, обеспечивающий его считывание компьютером или микропроцессором

***TETRIX Ardublockly*:** подключаемый интерфейс, позволяющий составлять программы в виде графических блоков, а не синтаксического кода, при этом он может отображать код для более эффективного устранения неисправностей и изучения программирования

ультразвуковой датчик: устройство, определяющее расстояние до предмета при помощи звуковых волн

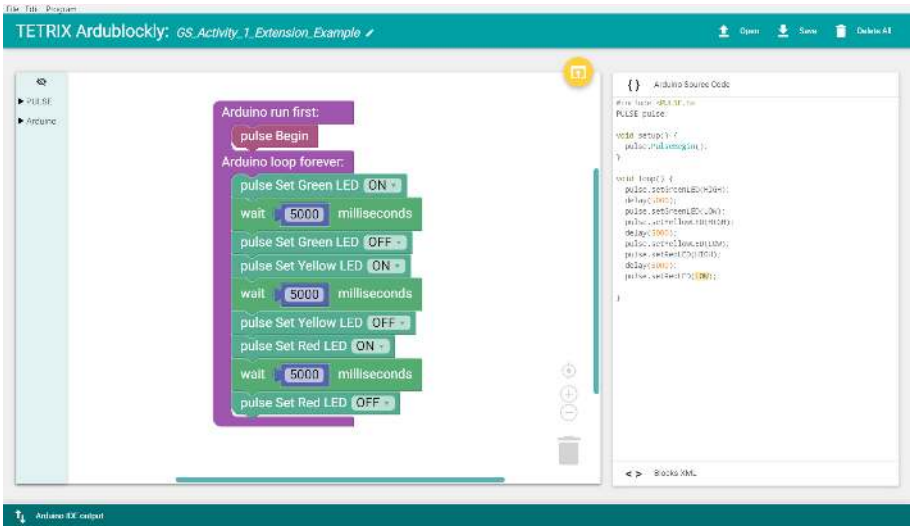
загрузить: перенести скетч или файл на компьютер или в микропроцессор

переменная: элемент данных с именем, который может принимать одно или несколько значений при выполнении программы

void setup: команда, обеспечивающая инициализацию программы

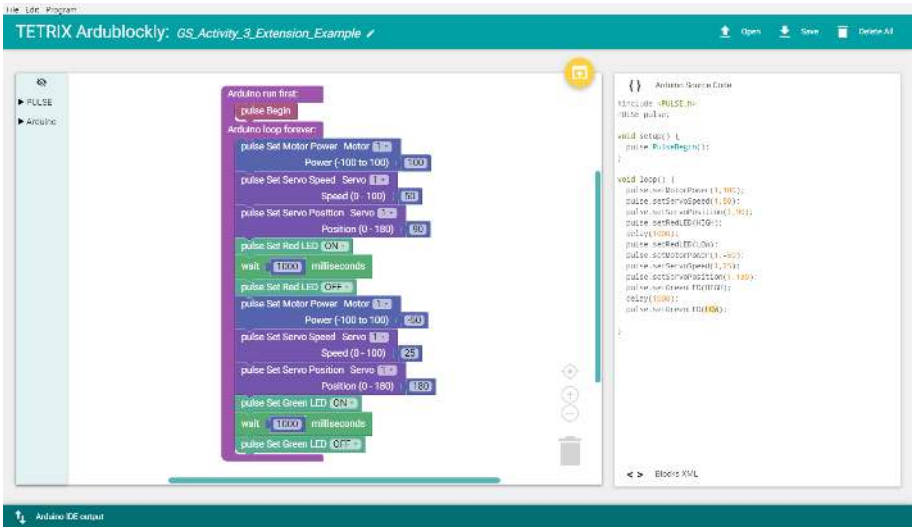
цикл while набор команд, выполняющийся многократно в зависимости от логического оператора

Дополнительные упражнения вводной части

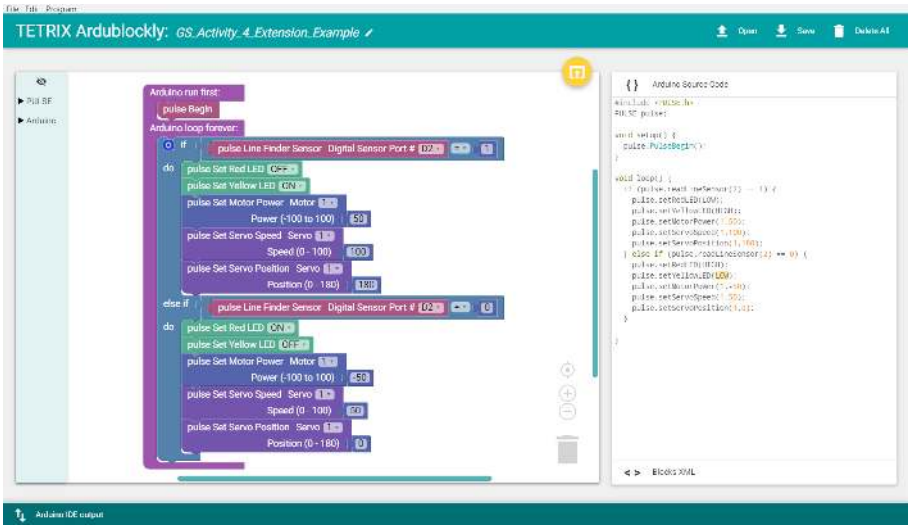


Дополнение к вводному упражнению 1

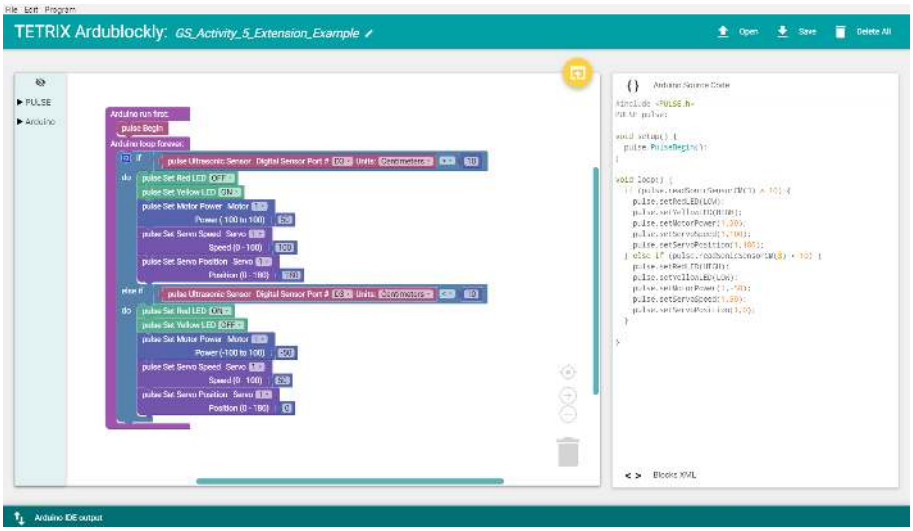
Дополнение к вводному упражнению 2



Дополнение к вводному упражнению 3



Дополнение к вводному упражнению 4



Дополнение к вводному упражнению 5

Полный перечень профессий

- авиадиспетчер
- разработчик автомобильного программного обеспечения
- инженер-биомедик
- конструктор автомобилей
- программист станков с ЧПУ
- цифровой скульптор
- инженер-электрик
- мастер по системам электроуправления аттракционами
- слесарь по электромеханическому оборудованию
- инженер-электронщик
- инженер
- рабочий производства
- механик сельскохозяйственного оборудования
- озеленитель
- инженер промышленного производства
- мелиоратор
- инженер-электровозостроитель
- оператор машинного оборудования
- специалист по техническому обслуживанию машинного оборудования
- слесарь
- инженер-технолог
- техник производства
- техник по составлению маршрутов
- инженер по разработке материалов
- инженер-материаловед
- лаборант-метролог
- чертежник механического оборудования
- инженер-механик
- медицинский техник
- программист ПЛК
- машинист
- железнодорожный рабочий
- специалист по робототехнике
- механик малых двигателей
- разработчик программного обеспечения
- техник по акустическим системам
- инженер-акустик
- техник-хирург
- технический директор
- инженер-испытатель
- управляющий парка развлечений
- инженер службы эксплуатации светофоров
- дорожный мастер
- технолог по вакуумному напылению
- инженер по вакуумному оборудованию

Технические характеристики робототехнического контроллера TETRIX PULSE

Микроконтроллер:	процессор ATmega328P с установленным загрузчиком операционной системы Arduino Optiboot
Память:	программируемое ЭПЗУ объемом 32 КБ (ATmega328P)
Электропитание:	6 вольт постоянного тока от аккумуляторной батареи NiMH 6 В серии TETRIX® PRIME
Порты для электродвигателей постоянного тока:	2 трёхштырьковых разъёма; управление с использованием Н-образного моста и ШИМ; постоянный ток 3 А в каждом канале, максимальный ток 5 А
Совместимый электродвигатель постоянного тока:	электродвигатель постоянного тока 6 В TETRIX PRIME (44298)
Режимы управления электродвигателем постоянного тока:	постоянная мощность (от -100 до 100 %) постоянная частота вращения при помощи ПИД-регулирования (от -100 до 100 градусов в секунду) ПИД-регулирование для достижения и удержания заданного конечного положения с учётом показаний энкодера ПИД-регулирование для достижения и удержания заданного конечного положения с учётом показаний энкодера в градусах
Порты для энкодеров:	2 квадратурных, 5 В постоянного тока, макс. 50 мА; технические характеристики: 360 отсчётов на оборот, 1440 импульсов на оборот; ENC 1 и ENC 2
Разъём USB:	USB типа B
Драйвер USB:	FTDI
Порты для стандартных сервоприводов:	всего 6: каналы для 1-6 сервоприводов
Совместимый сервопривод:	сервопривод HiTec HS322-HD (40538)
Совокупное ограничение по току сервопривода:	постоянный ток макс. 6 В, 6 А
Режимы управления сервоприводами:	Задание частоты вращения сервопривода (от 0 до 100 %) Задание положения сервопривода (0-180 градусов)
Отслеживание напряжения аккумуляторной батареи:	в пределах 0-7,5 В
3 порта для цифровых датчиков (D2-D4):	каждый порт можно настроить на ввод или вывод цифровых сигналов или превратить в порт последовательной связи.
3 порта для аналоговых датчиков (A1-A3):	каждый порт можно настроить на ввод аналоговых сигналов или ввод-вывод цифровых сигналов
1 порт связи между ИС (I2C):	частота 100 кГц. Для этого соединения используется та же шина связи между ИС, что и для внутренних микроконтроллеров. управляющих электродвигателями постоянного тока и сервоприводами. Адреса I2C с 0x01 по 0x06 оставлены контроллером PULSE в запасе.
Порт для подсоединения аккумуляторной батареи:	трёхштырьковый разъём. Используйте только аккумуляторную батарею NiMH 6 В серии TETRIX PRIME.
1 одна зелёная кнопка "Пуск" (START):	программируемая кнопка
1 красная кнопка сброса параметров/остановки (RESET):	непрограммируемая кнопка
1 красный светодиод:	программируемый светодиодный индикатор
1 жёлтый светодиод:	программируемый светодиодный индикатор
1 зелёный светодиод:	программируемый светодиодный индикатор
1 синий светодиод:	включается на время подачи в цепь электропитания
2 жёлтых светодиода:	указывают на последовательную передачу данных через порт USB
1 красный и 1 зелёный светодиод электродвигателей постоянного тока:	сигнализируют о вращении электродвигателей постоянного тока с указанием направления, задаваемого по каждому каналу управления электродвигателями постоянного тока

Схемы расположения контактов контроллера TETRIX PULSE

Порты для датчиков на контроллере PULSE

В контроллере PULSE используются схемы назначения контактов, совместимые с Arduino UNO. Датчики, к которым в библиотеке Arduino для контроллера PULSE есть программные коды, настраиваются автоматически благодаря библиотечным функциям. Поддержка разных типов датчиков будет добавляться по мере их выхода на рынок. Однако функции составления программных кодов на языке Arduino позволяют напрямую пользоваться всеми портами, если требуется написать код для дополнительных датчиков, еще не поддерживаемых в библиотеке для контроллера PULSE. За исключением порта I2C, все остальные можно настроить на ввод или вывод сигналов при помощи функции Arduino pinMode(). Более подробную информацию можно найти в разделе справки по языкам (Language Reference) на сайте**https://www.arduino.cc/en/Reference/HomePage**.

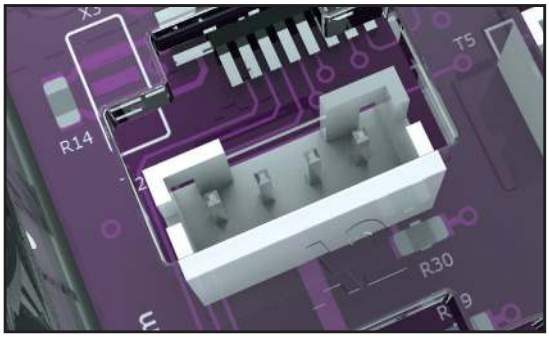


Рисунок 67: Порт для датчиков на контроллере PULSE (контакты слева направо: 1, 2, 3, 4)

Примечание: Порт I2C контроллера PULSE можно использовать только в режиме I2C. Переводить его в аналоговой или цифровой режим нельзя.

Примечание: Порты для аналоговых датчиков A1-A3 можно также настроить на ввод-вывод цифровых сигналов.

Таблица 1: Назначение контактов порта I2C

Контакт	Функция	Назначение контактов в ПО *Arduino (IDE)* ( )
Контакт 1	Заземление	Н/П
Контакт 2	+5 В, 100 мА	Н/П
Контакт 3	SDA (последовательная передача данных через шину связи I2C)	Канал ввода ADC4 (A4) | ввод-вывод цифровых сигналов (18)
Контакт 4	SCL (тактовый сигнал шины связи I2C)	Канал ввода ADC5 (A5) | ввод-вывод цифровых сигналов (19)

Таблица 2: Порт для аналоговых датчиков (A1)

Контакт	Функция	Назначение контактов в ПО *Arduino (IDE)* ( )
Контакт 1	Заземление	Н/П
Контакт 2	+5 В, 100 мА	Н/П
Контакт 3	Неиспользуемый	Н/П
Контакт 4	Ввод аналоговых или ввод-вывод цифровых сигналов	Ввод аналоговых сигналов (A1) | ввод-вывод цифровых сигналов (15)

Таблица 3: Порт для аналоговых датчиков (A2)

Контакт	Функция	Назначение контактов в ПО *Arduino (IDE)* ( )
Контакт 1	Заземление	Н/П
Контакт 2	+5 В, 100 мА	Н/П
Контакт 3	Неиспользуемый	Н/П
Контакт 4	Ввод аналоговых или ввод-вывод цифровых сигналов	Ввод аналоговых сигналов (A2) | ввод-вывод цифровых сигналов (16)

Таблица 4: Порт для аналоговых датчиков (A3)

Контакт	Функция	Назначение контактов в ПО *Arduino (IDE)* ( )
Контакт 1	Заземление	Н/П
Контакт 2	+5 В, 100 мА	Н/П
Контакт 3	Неиспользуемый	Н/П
Контакт 4	Ввод аналоговых или ввод-вывод цифровых сигналов	Ввод аналоговых сигналов (A3) | ввод-вывод цифровых сигналов (17)

Таблица 5: Порт для цифровых датчиков (D2)

Контакт	Функция	Назначение контактов в ПО *Arduino (IDE)* ( )
Контакт 1	Заземление	N/A
Контакт 2	+5 В, 100 мА	Н/П
Контакт 3	Ввод-вывод цифровых сигналов	Ввод-вывод цифровых сигналов (9)
Контакт 4	Ввод-вывод цифровых сигналов	Ввод-вывод цифровых сигналов (2)

Таблица 6: Порт для цифровых датчиков (D3)

Контакт	Функция	Назначение контактов в ПО *Arduino (IDE)* ( )
Контакт 1	Заземление	Н/П
Контакт 2	+5 В, 100 мА	Н/П
Контакт 3	Ввод-вывод цифровых сигналов	Ввод-вывод цифровых сигналов (10)
Контакт 4	Ввод-вывод цифровых сигналов	Ввод-вывод цифровых сигналов (3)

Таблица 7: Порт для цифровых датчиков (D4)

Контакт	Функция	Назначение контактов в ПО *Arduino (IDE)* ( )
Контакт 1	Заземление	Н/П
Контакт 2	+5 В, 100 мА	Н/П
Контакт 3	Ввод-вывод цифровых сигналов	Ввод-вывод цифровых сигналов (11)
Контакт 4	Ввод-вывод цифровых сигналов	Ввод-вывод цифровых сигналов (4)

Примечание: Порты для цифровых датчиков D2-D4 можно также превратить в порты последовательной связи с программным управлением, используя для этого библиотеку функций последовательной передачи данных в ПО *Arduino (IDE)*.

Таблица библиотечных функций Arduino для контроллера серии TETRIX PULSE

Описание	Функция	Пример кода
Начать работу контроллера PULSE Вызывается в цикле настройки() кода на языке Arduino. Начинает работу контроллера PULSE.	**PulseBegin**(); Тип данных: Отсутствуют	**PulseBegin**(); *Сбросить параметры и инициализировать контроллер PULSE.*
Закончить работу контроллера PULSE После вызова сразу же завершает программу и возвращает контроллер PULSE в исходное состояние.	**PulseEnd**(); Тип данных: Отсутствуют	**PulseEnd**(); *Завершить программу контроллера PULSE и вернуть контроллер в исходное состояние.*
Настроить красный светодиод Даёт красному светодиоду контроллера PULSE команду на включение или выключение.	**setRedLED**(состояние); Тип данных: *состояние* = целое число Диапазон данных: *состояние* = 1 или 0 или *состояние* = HIGH или LOW	**setRedLED**(HIGH); *или* **setRedLED**(1); *Включить красный светодиод.* **setRedLED**(LOW); *или* **setRedLED**(0); *Выключить красный светодиод.*
Настроить жёлтый светодиод Даёт жёлтому светодиоду контроллера PULSE команду на включение или выключение.	**setYellowLED**(состояние); Тип данных: *состояние* = целое число Диапазон данных: *состояние* = 1 или 0 или *состояние* = HIGH или LOW	**setYellowLED**(HIGH); *или* **setYellowLED**(1); *Включить красный светодиод.* **setYellowLED**(LOW); *или* **setYellowLED**(0); *Выключить красный светодиод.*
Настроить зелёный светодиод Даёт зелёному светодиоду контроллера PULSE команду на включение или выключение.	**setGreenLED**(состояние); Тип данных: *состояние* = целое число Диапазон данных: *состояние* = 1 или 0 или *состояние* = HIGH или LOW	**setGreenLED**(HIGH); *или* **setGreenLED**(1); *Включить зелёный светодиод.* **setGreenLED**(LOW); *или* **setGreenLED**(0); *Выключить зелёный светодиод.*
Задать мощность электродвигателя постоянного тока Задаёт уровень мощности и направление вращения электродвигателя постоянного тока TETRIX, подсоединённого к соответствующим портам на контроллере PULSE. Диапазон уровня мощности — от 0 до 100. Направление вращения задаётся знаком (+/-) перед уровнем мощности. Уровень мощности 0 = остановить электродвигатель.	**setMotorPower**(электродвигатель №, мощность); Тип данных: *электродвигатель №* = целое число *мощность* = целое число Диапазон данных: *электродвигатель №* = 1 или 2 *мощность* = от -100 до 100	**setMotorPower**(1, 50); *Вращать электродвигатель 1 по часовой стрелке с мощностью 50 %.* **setMotorPower**(2, -50 %); *Вращать электродвигатель 2 против часовой стрелки с мощностью 50 %.* **setMotorPower**(1, 0); *Выключить электродвигатель 1.*
Задать мощность электродвигателей постоянного тока Одновременно задаёт уровень мощности и направление вращения **обоих** электродвигателей постоянного тока TETRIX, подсоединённых к соответствующим портам на контроллере PULSE. Параметры каналов контроллера PULSE для электродвигателя 1 и электродвигателя 2 задаются при помощи одного оператора. Диапазон уровня мощности — от 0 до 100. Направление вращения задаётся знаком (+/-) перед уровнем мощности. Уровень мощности 0 = остановить электродвигатель.	**setMotorPowers**(мощность1, мощность2); Тип данных: *мощность1* = целое число *мощность2* = целое число Диапазон данных: *мощность1* = от -100 до 100 *мощность2* = от -100 до 100	**setMotorPowers**(50, 50); *Вращать электродвигатель 1 и электродвигатель 2 по часовой стрелке с мощностью 50 %.* **setMotorPowers**(-50, 50); *Вращать электродвигатель 1 и электродвигатель 2 против часовой стрелки с мощностью 50 %.* **setMotorPowers**(0, 0); *Выключить электродвигатель 1 и электродвигатель 2.*

Описание	Функция	Пример кода
Задать частоту вращения электродвигателя постоянного тока Задаёт постоянную частоту вращения электродвигателя постоянного тока TETRIX, оборудованного энкодером TETRIX, при помощи ПИД-регулятора скорости. Параметр *частота вращения* имеет диапазон от 0 до 720 градусов в секунду (град/с). Направление вращения задаётся знаком (+/-) параметра *частота вращения*.	**setMotorSpeed** (электродвигатель №, частота вращения); Тип данных: *электродвигатель №* = целое число *частота вращения* = целое число Диапазон данных: *электродвигатель №* = 1 или 2 *частота вращения* = от -720 до 720	**setMotorSpeed**(1, 360); *Вращать электродвигатель1 по часовой стрелке с постоянной частотой 360 град/с.* **setMotorSpeed**(1, -360); *Вращать электродвигатель 1 против часовой стрелки с постоянной частотой 360 град/с.*
Задать частоту вращения электродвигателей постоянного тока Задаёт постоянную частоту вращения одновременно для **обоих** каналов электродвигателей постоянного тока TETRIX, оборудованного энкодерами TETRIX, при помощи ПИД-регулятора скорости. Параметры каналов контроллера PULSE для электродвигателя 1 и электродвигателя 2 задаются при помощи одного оператора. Параметр частота вращения имеет диапазон от 0 до 720 градусов в секунду (град/с). Направление вращения задаётся знаком (+/-) параметра *частота вращения*.	**setMotorSpeeds**(частота вращения 1, частота вращения 2); Тип данных: *частота вращения 1* = целое число *частота вращения 2* = целое число Диапазон данных: *частота вращения 1* = от -720 до 720 *частота вращения 2* = от -720 до 720	**setMotorSpeeds**(360, 360); *Вращать электродвигатель 1 и электродвигатель 2 по часовой стрелке с постоянной частотой 360 град/с.* **setMotorSpeeds**(360, -360); *Вращать электродвигатель 1 и электродвигатель 2 против часовой стрелки с постоянной частотой 360 град/с.* **setMotorSpeeds**(360, -180); *Вращать электродвигатель 1 и электродвигатель 2 против часовой стрелки с постоянной частотой 180 град/с.*
Задать конечное положение электродвигателей постоянного тока Задаёт постоянную частоту вращения, а также конечное удерживаемое положение по счётчику энкодера, при помощи ПИД-регулятора скорости и положения, одновременно для **обоих** каналов электродвигателей постоянного тока TETRIX, каждый из которых оборудован энкодером TETRIX. Параметры каналов контроллера PULSE для электродвигателя 1 и электродвигателя 2 задаются при помощи одного оператора. Параметр частота вращения имеет диапазон от 0 до 720 градусов в секунду (град/с). Конечное положение по счётчику энкодера представляет собой длинное целое число со знаком от -2 147 483 648 до 2 147 483 647. Разрешающая способность энкодера = 1/4 градуса на отсчёт.	**setMotorTargets**(частота вращения 1, конечное значение 1, частота вращения 2, конечное значение 2); Тип данных: *частота вращения 1* = целое число *конечное значение 1* = длинное целое *частота вращения 2* = целое число *конечное значение 2* = длинное целое Диапазон данных: *частота вращения 1* = от 0 до 720 *конечное значение1* = от -2147483648 до 2147483647 *частота вращения 2* = от 0 до 720 *конечное значение2* = от -2147483648 до 2147483647	**setMotorTargets**(360, 1440, 360, 1440); *Вращать электродвигатель 1 и электродвигатель 2 с постоянной частотой 360 град/с, пока значение энкодера на каждом из электродвигателей не достигнет 1440. Когда электродвигатель достигнет заданного конечного положения по показаниям энкодера, задержать положение в режиме сервопривода.* **setMotorTargets**(360, 1440, 180, 2880); *Вращать электродвигатель 1 с постоянной частотой 360 град/с, пока значение энкодера 1 не достигнет 1440. Вращать электродвигатель 2 с постоянной частотой 180 град/с, пока значение энкодера 2 не достигнет 2880. Каждый из двигателей задержится, подобно сервоприводу, в том положении, которое было у него на момент достижения энкодером заданного конечного значения.* **Примечание:** *Разрешающая способность энкодера = 1/4 градуса на отсчёт. Например, 1 оборот электродвигателя равен 1440 отсчётов энкодера (1440 / 4 = 360).*
Задать положение электродвигателя в градусах Задаёт постоянную частоту вращения, а также конечное удерживаемое положение в градусах электродвигателя постоянного тока TETRIX, оборудованного энкодером TETRIX, при помощи ПИД-регулятора скорости и положения. Параметр *частота вращения* имеет диапазон от 0 до 720 градусов в секунду (град/с). Конечное положение по показаниям энкодера в градусах представляет собой длинное целое число со знаком от -536 870 912 до 536 870 911 с разрешающей способностью энкодера 1 градус.	**setMotorDegree**(электродвигатель №, частота вращения, градусы); Тип данных: *электродвигатель №* = целое число *частота вращения* = целое число градусы = длинное целое Диапазон данных: *электродвигатель №* = 1 или 2 *частота вращения* = от 0 до 720 *градусы* = от -536870912 до 536870911	**setMotorDegree**(1, 180, 360); *Вращать электродвигатель 1 с постоянной частотой 180 град/с, пока значение энкодера 1 в градусах не достигнет 360. При достижении конечного значения энкодера в градусах задержать положение в режиме сервопривода.* **setMotorDegree**(2, 90, 180); *Вращать электродвигатель 2 с постоянной частотой 90 град/с, пока значение энкодера 2 в градусах не достигнет 180. При достижении конечного значения энкодера в градусах задержать положение в режиме сервопривода.*

Описание	Функция	Пример кода
Задать положение электродвигателей в градусах Задаёт постоянную частоту вращения, а также конечное удерживаемое положение в градусах, при помощи ПИД-регулятора скорости и положения, для обоих каналов электродвигателей постоянного тока TETRIX, оборудованных энкодерами TETRIX. Параметры каналов контроллера PULSE для электродвигателя 1 и электродвигателя 2 задаются при помощи одного оператора. Параметр частота вращения имеет диапазон от 0 до 720 градусов в секунду (град/с). Заданное конечное положение по показаниям энкодера в градусах представляет собой длинное целое число со знаком от -536 870 912 до 536 870 911 с разрешающей способностью энкодера 1 градус.	**setMotorDegrees**(частота вращения 1, градусы 1, частота вращения 2, градусы 2); Тип данных: *частота вращения 1* = целое число *градусы 1* = длинное целое *частота вращения 2* = целое число *градусы 2* = длинное целое Диапазон данных: *частота вращения 1* = от 0 до 720 *градусы 1* = от -536870912 до 536870911 *частота вращения 2* = от 0 до 720 *градусы 2* = от -536870912 до 536870911	**setMotorDegrees**(180, 360, 180, 360); *Вращать электродвигатель 1 и электродвигатель 2 с постоянной частотой 180 град/с, пока значение энкодера в градусах на каждом из электродвигателей не достигнет 360. Когда электродвигатель достигнет заданного конечного положения по показаниям энкодера в градусах, задержать положение в режиме сервопривода.* **setMotorDegrees**(360, 720, 90, 360); *Вращать электродвигатель 1 с постоянной частотой 360 град/с, пока значение энкодера 1 в градусах не достигнет 720. Вращать электродвигатель 2 с постоянной частотой 90 град/с, пока значение энкодера 2 в градусах не достигнет 360. Каждый из двигателей задержится, подобно сервоприводу, в том положении, которое было у него на момент достижения энкодером заданного конечного значения.*
Задать обратное направление электродвигателю Делает обратной схему вращения вперёд-назад в канале электродвигателя постоянного тока. Функция предназначена для согласования прямого и обратного вращения электродвигателей на противоположных ботах шасси робототехнической модели с бортовым поворотом. Замена заданных действий обратными в одном канале управления электродвигателями может потребовать более интуитивного программирования противолежащих электродвигателей постоянного тока, работающих в паре. Параметр *обратное действие* со значением 1 = обратное действие. Параметр *обратное действие* со значением 0 = отсутствие обратного действия. По умолчанию установлено отсутствие обратного действия.	**setMotorInvert**(электродвигатель №, обратное действие); Тип данных: *электродвигатель №* = целое число *обратное действие* = целое число Диапазон данных: *электродвигатель №* = 1 или 2 *обратное действие* = 0 или 1	**setMotorInvert**(1, 1); *Заменить схему направления вращения электродвигателя № 1 на обратную.* **setMotorInvert**(2, 1); *Заменить схему направления вращения электродвигателя № 2 на обратную.* **setMotorInvert**(1, 0); *Не менять схему направления вращения электродвигателя № 1 на обратную.* **setMotorInvert**(2, 0); *Не менять схему направления вращения электродвигателя № 2 на обратную.* **Примечание:** *По умолчанию контроллер PULSE настроен на отсутствие обратного действия после включения питания или сброса параметров.*
Считать состояние занятости электродвигателя Проверяет по флажку занятости состояние электродвигателя постоянного тока, который работает в режиме ПИД-регулирования положения. В состоянии занятости электродвигателя будет возвращено значение "1", если он движется к заданному конечному положению (в градусах или в отсчётах энкодера). Когда он достигнет заданного конечного положения и задержится в нём, в состоянии занятости будет возвращено значение "0".	**readMotorBusy**(электродвигатель №); Тип данных: *электродвигатель №* = целое число Диапазон данных: *электродвигатель №* = 1 или 2 Тип возвращаемых данных: *значение* = целое число (0 или 1)	**readMotorBusy**(1); *Вернуть состояние занятости электродвигателя №1.* **readMotorBusy**(2); *Вернуть состояние занятости электродвигателя №2.*

Описание	Функция	Пример кода
Считать показания счётчика энкодера Считывает число отсчётов энкодера. Контроллер PULSE использует число импульсов, выданных энкодером, для управления по ПИД-алгоритму электродвигателем постоянного тока TETRIX, подсоединённым к соответствующим портам. Контроллер отсчитывает импульсы в заданном отрезке времени и на основе этого точно регулирует скорость и положение. Каждая 1/4 градуса равна одному импульсу или отсчёту, либо 1 угловой градус равен 4 отсчётам энкодера. Считав текущее показание счётчика, можно определить положение вала электродвигателя. Общие показания счётчика находятся в диапазоне от -2 147 483 648 до 2 147 483 647. Вращение по часовой стрелке увеличивает показания счётчика, вращение против часовой стрелки — уменьшает. Значения энкодера обнуляются в случае включения питания и сброса параметров.	**readEncoderCount**(энкодер №); Тип данных: *энкодер №* = целое число Диапазон данных: *энкодер №* = 1 или 2 Тип возвращаемых данных: *значение* = длинное целое	**readEncoderCount**(1); *Считать текущее показание счётчика энкодера 1 (порт ENC 1).* **readEncoderCount**(2); *Считать текущее показание счётчика энкодера 2 (порт ENC 2).*
Считать показания энкодера в градусах Считывает число отсчитанных энкодером градусов. Контроллер PULSE использует число импульсов, выданных энкодером, для управления по ПИД-алгоритму электродвигателем постоянного тока TETRIX, подсоединённым к соответствующим портам. Контроллер отсчитывает импульсы в заданном отрезке времени и на основе этого точно регулирует скорость и положение. Эта функция аналогична функции счётчика энкодера, но вместо возврата приблизительных показаний счётчика энкодера она возвращает положение вала электродвигателя в градусах. Общие показания энкодера в градусах находятся в диапазоне от -536 870 912 до 536 870 911. Вращение по часовой стрелке увеличивает показания счётчика, вращение против часовой стрелки — уменьшает. Значения энкодера обнуляются в случае включения питания и сброса параметров.	**readEncoderDegrees**(энкодер №); Тип данных: *энкодер №* = целое число Диапазон данных: *энкодер №* = 1 или 2 Тип возвращаемых данных: *значение* = длинное целое	**readEncoderDegrees**(1); *Считать текущее показание энкодера 1 в градусах (порт ENC 1).* **readEncoderDegrees**(2); *Считать текущее показание энкодера 2 в градусах (порт ENC 2).*
Обнулить счётчик энкодера Вызов этой функции приведет к обнулению показаний энкодера.	**resetEncoder**(энкодер №); Тип данных: *энкодер №* = целое число Диапазон данных: *энкодер №* = 1 или 2	**resetEncoder**(1); *Обнулить показания счётчика энкодера 1.* **resetEncoder**(2); *Обнулить показания счётчика энкодера 2.*
Обнулить счётчики обоих энкодеров (1 и 2) Обнулить показания счётчика энкодера 1 и энкодера 2.	**resetEncoders**(); Тип данных: Отсутствуют	**resetEncoders**(); *Обнулить показания счётчика энкодера 1 и энкодера 2.*

Описание	Функция	Пример кода
Считать выходной сигнал датчика линии Считывает цифровой выходной сигнал датчика линии, подсоединённого к порту для датчиков контроллера PULSE. Считываемое значение равно 0, если уловлено отражение света (обнаружена светлая поверхность), и 1, если отражение света не уловлено (обнаружена тёмная поверхность, например, линия).	**readLineSensor**(порт №); Тип данных: *порт №* = целое число Диапазон данных: *порт №* (см. примечание в соседней колонке) Тип возвращаемых данных: *значение* = целое число (0 или 1)	**readLineSensor**(2); *Считать цифровое значение датчика линии на порте D2 для цифровых датчиков.* **Примечание:** *Датчик линии можно подсоединить к любому цифровому порту D2-D4 или аналоговому порту A1-A3, настроенному на прием цифровых сигналов.*
Считать показания ультразвукового датчика в сантиметрах Считывает расстояние в сантиметрах до предмета перед ультразвуковым датчиком. Частота датчика модулируется на уровне 42 к Гц, а диапазон измерения составляет от 3 до 400 сантиметров. Считываемое значение представляет собой целое число.	**readSonicSensorCM**(порт №); Тип данных: *порт №* = целое число Диапазон данных: *порт №* (см. примечание в соседней колонке) Тип возвращаемых данных: *значение* = целое число (от 3 до 400) Возможны небольшие отличия минимальных и максимальных значений.	**readSonicSensorCM**(3); *Считать расстояние в сантиметрах до предмета перед ультразвуковым датчиком, подсоединённым к порту D3 для цифровых датчиков.* **Примечание:** *Ультразвуковой датчик можно подсоединить к любому цифровому порту D2-D4 или аналоговому порту A1-A3, настроенному на прием цифровых сигналов.*
Считать показания ультразвукового датчика в дюймах Считывает расстояние в дюймах до предмета перед ультразвуковым датчиком. Частота датчика модулируется на уровне 42 к Гц, а диапазон измерения составляет от 2 до 150 дюймов. Считываемое значение представляет собой целое число.	**readSonicSensorIN**(порт №); Тип данных: *порт №* = целое число Диапазон данных: *порт №* (см. примечание в соседней колонке) Тип возвращаемых данных: *значение* = целое число (от 2 до 150) Возможны небольшие отличия минимальных и максимальных значений.	**readSonicSensorIN**(4); *Считать расстояние в дюймах до предмета перед ультразвуковым датчиком, подсоединённым к порту D4 для цифровых датчиков.* **Примечание:** *Ультразвуковой датчик можно подсоединить к любому цифровому порту D2-D4 или аналоговому порту A1-A3.*
Считать напряжение аккумуляторной батареи Считывает напряжение аккумуляторной батареи TETRIX, питающей контроллер PULSE. Считываемое значение представляет собой целое число.	**readBatteryVoltage**(); Тип данных: Отсутствуют Тип возвращаемых данных: *значение* = целое число	**readBatteryVoltage**(); *Считывает напряжение аккумуляторной батареи TETRIX, питающей контроллер PULSE.* *Примеры: Значение 918 равно 9,18 В.*
Считать состояние пусковой кнопки Считывает состояние зелёной пусковой кнопки контроллера PULSE. Возвращение значения "1" указывает на то, что кнопка нажата. Возвращение значения "0" указывает на то, что кнопка не нажата.	**readStartButton**(); Тип данных: Отсутствуют Тип возвращаемых данных: *значение* = целое число (0 или 1)	**readStartButton**(); *Считать кнопку "Пуск". Значение "1" означает, что кнопка нажата. Значение "0" означает, что кнопка нажата.*

Описание	Функция	Пример кода
Задать частоту вращения сервопривода Задаёт частоту вращения сервопривода, подсоединённого к портам для сервоприводов 1-6 контроллера PULSE. Параметр *частота вращения* способен принимать значения из диапазона от 0 до 100 %. Параметр "канал сервопривода" может иметь любое значение от 1 до 6. Если не указано иное, частота вращения сервопривода по умолчанию равна 100 (максимальная частота). После того, как частота вращения сервопривода установлена, он всегда будет вращаться с установленной частотой до её изменения. Если не менять частоту вращения, эта функция вызывается только один раз, в начале программы.	**setServoSpeed**(сервопривод №, частота вращения); Тип данных: *сервопривод №* = целое число *частота вращения* = целое число Диапазон данных: *сервопривод №* = от 1 до 6 *частота вращения* = от 0 до 100	**setServoSpeed**(1, 25); *Установить частоту вращения по каналу сервопривода 1 на 25 %.* **setServoSpeed**(2, 50); *Установить частоту вращения по каналу сервопривода 2 на 50%.*
Задать частоту вращения всех сервоприводов Задаёт частоту вращения по всем шести каналам сервоприводов одновременно посредством одной команды. Все шесть значений частоты вращения задаются последовательно в диапазоне от 0 до 100 %. Все шесть значений могут быть одинаковыми или разными.	**setServoSpeeds**(частота вращения1, частота вращения2, частота вращения3, частота вращения4, частота вращения5, частота вращения6); Тип данных: *частота вращения1-частота вращения6* = целое число Диапазон данных: *частота вращения1-частота вращения6* = от 0 до 100	**setServoSpeeds**(25, 25, 25, 25, 25, 25); *Установить частоту вращения по всем шести каналам сервоприводов на 25 %* **setServoSpeeds**(25, 35, 45, 55, 65, 75); *Частота вращения сервопривода 1 = 25 %* *Частота вращения сервопривода 2 = 35 %* *Частота вращения сервопривода 3 = 45 %* *Частота вращения сервопривода 4 = 55 %* *Частота вращения сервопривода 5 = 65 %* *Частота вращения сервопривода 6 = 75 %*
Задать положение сервопривода Задаёт угловое положение сервопривода, подсоединённого к портам для сервоприводов 1-6 контроллера PULSE. Параметр *положение* способен принимать любое значение от 0 до 180 градусов. Любое значение, выходящее за пределы данного диапазона, игнорируется. Сервоприводы могут иметь разную конструкцию, поэтому при их эксплуатации на границе диапазона соблюдайте осторожность. Прислушивайтесь к звуку работы сервопривода. Шум говорит о том, что сервопривод прижат к механическому ограничителю, что влечет риск его повреждения. Если это произошло, следует уменьшить диапазон до значений чуть больше 0 и чуть меньше 180 во избежание повреждений.	**setServoPosition**(сервопривод №, положение); Тип данных: *сервопривод №* = целое число *положение* = целое число Диапазон данных: *сервопривод №* = от 1 до 6 *положение* = от 0 до 180	**setServoPosition**(1, 90); *Установить угловое положение сервопривода 1 на 90 градусов.* **setServoPosition**(2, 130); *Установить угловое положение сервопривода 2 на 130 градусов.*
Задать положение всех сервоприводов Задаёт угловое положение всех шести сервоприводов, подсоединённых к портам для сервоприводов 1-6 контроллера PULSE. Параметр положение способен принимать любое значение от 0 до 180 градусов. Любое значение, выходящее за пределы данного диапазона, игнорируется. Сервоприводы могут иметь разную конструкцию, поэтому при их эксплуатации на границе диапазона соблюдайте осторожность. Прислушивайтесь к звуку работы сервопривода. Шум говорит о том, что сервопривод прижат к механическому ограничителю, что влечет риск его повреждения. Если это произошло, следует уменьшить диапазон до значений чуть больше 0 и чуть меньше 180 во избежание повреждений.	**setServoPositions**(положение1, положение2, положение3, положение4, положение5, положение6); Тип данных: *положение1-положение6* = целое число Диапазон данных: *положение1-положение6* = от 0 до 180	**setServoPositions**(90, 90, 90, 90, 90, 90); *Установить угловое положение всех шести сервоприводов, подсоединённых к портам для сервоприводов 1-6 контроллера PULSE, на 90 градусов.* **setServoPositions**(10, 20, 30, 40, 50, 60); *Установить угловое положение всех шести сервоприводов, подсоединённых к портам для сервоприводов 1-6 контроллера PULSE.* *Положение сервопривода 1 = 10 градусов* *Положение сервопривода 2 = 20 градусов* *Положение сервопривода 3 = 30 градусов* *Положение сервопривода 4 = 40 градусов* *Положение сервопривода 5 = 50 градусов* *Положение сервопривода 6 = 60 градусов*

Описание	Функция	Пример кода
Считать положение сервопривода Считывает соответствующее самой последней команде положение сервопривода, подсоединённого к портам 1-6 для сервоприводов на контроллере PULSE. Возвращаемое значение составит 0-180.	**readServoPosition**(сервопривод №); Тип данных: *сервопривод №* = целое число Диапазон данных: *сервопривод №* = от 1 до 6 Тип возвращаемых данных: *значение* = целое число (от 0 до 180)	**readServoPosition**(1); *Считать соответствующее самой последней команде положение сервопривода 1.* **readServoPosition**(2); *Считать соответствующее самой последней команде положение сервопривода 2.*

Памятка по библиотечным функциям Arduino для контроллера серии TETRIX PULSE

Ниже представлен указатель всех операторов функций на языке Arduino в библиотеке для робототехнического контроллера TETRIX PULSE.

PulseBegin();

PulseEnd();

setRedLED(HIGH/LOW);

setYellowLED(HIGH/LOW);

setGreenLED(HIGH/LOW);

setMotorPower(электродвигатель №, мощность);

setMotorPowers(мощность1, мощность2);

setMotorSpeed(электродвигатель №, частота вращения);

setMotorSpeeds(частота вращения1, частота вращения2);

setMotorTarget(электродвигатель №, частота вращения, конечное значение);

setMotorTargets(частота вращения1, конечное значение1, частота вращения2, конечное значение2);

setMotorDegree(электродвигатель №, частота вращения, градусы);

setMotorDegrees(частота вращения1, градусы1, частота вращения2, градусы2);

setMotorInvert(электродвигатель №, обратное действие);

readMotorBusy(электродвигатель №);

readEncoderCount(энкодер №);

readEncoderDegrees(энкодер №);

resetEncoder(энкодер №);

resetEncoders();

readLineSensor(порт №);

readSonicSensorCM(порт №);

readSonicSensorIN(порт №);

readBatteryVoltage();

readStartButton();

setServoSpeed(сервопривод №, частота вращения);

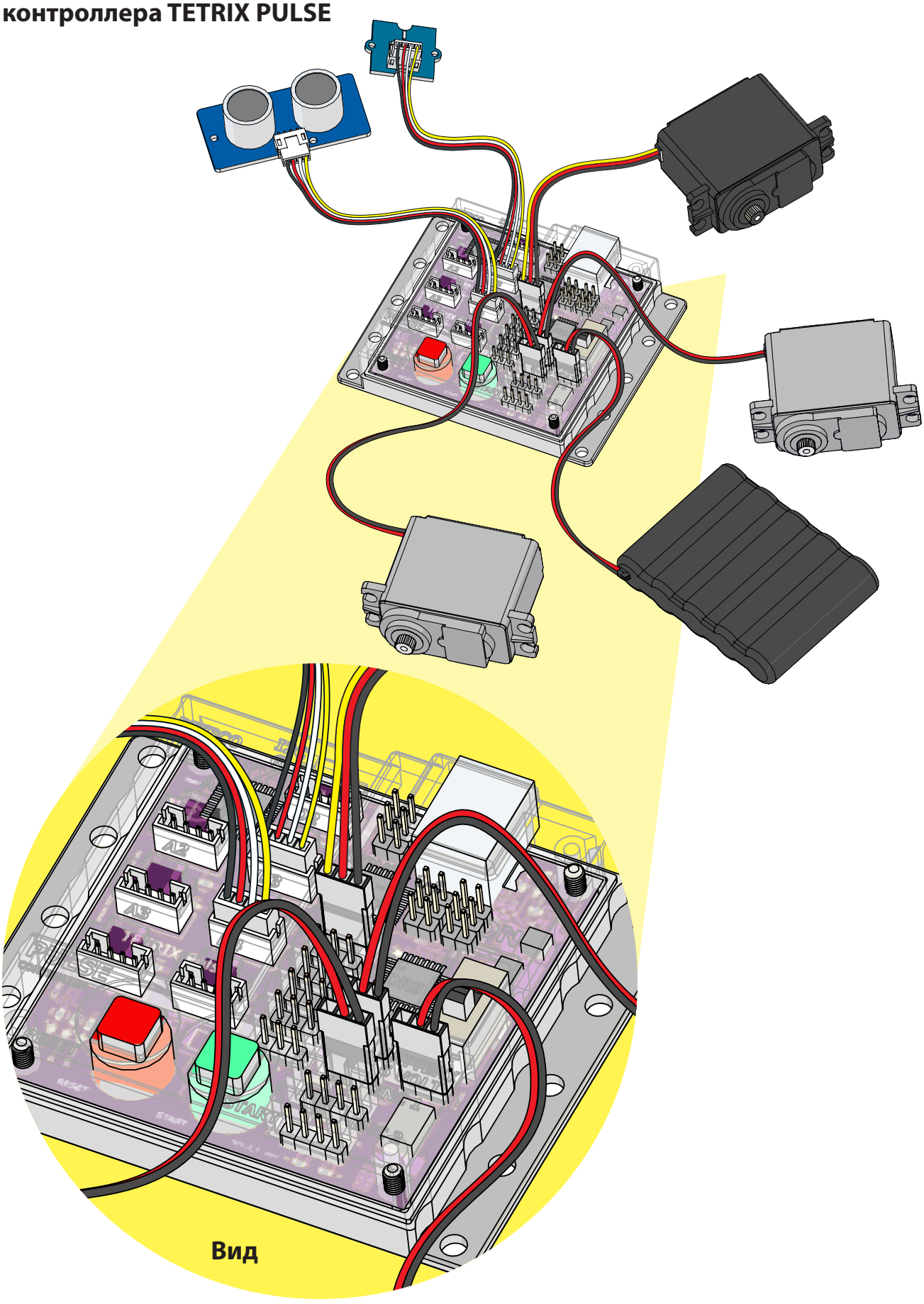
setServoSpeeds(частота вращения1, частота вращения2, частота вращения3, частота вращения4, частота вращения5, частота вращения6);

setServoPosition(сервопривод №, положение);

setServoPositions(положение1, положение2, положение3, положение4, положение5, положение6);

readServoPosition(сервопривод №);

Схема электрических соединений контроллера TETRIX PULSE



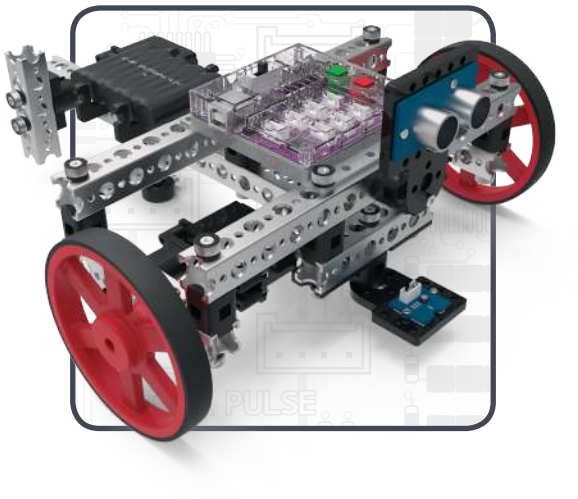
Вид

PITSCO

TETRIX®

PRIME

Робототехнический контроллер TETRIX® PULSE™
Советы по программированию



Бесплатный звонок
800•835•0686

Загляните на наш сайт
TETRIXrobotics.com

PITSCO EDUCATION