

Идеально **для** дополнительного образования

Язык программирования **C++**

3 способа управления

40 готовых программ

Видеоинструкция по сборке

Не надо паять!

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

ЗНАТОК

Car-DUINO

Li-Ion
АККУМУЛЯТОР

МОЩНЫЕ
ФАРЫ

УНИВЕРСАЛЬНЫЕ
ВЕРХНИЕ ОГНИ



Требуются:

Компьютер с Windows 7.0 или выше
подключенный к Интернету



Компьютер с macOS 10.0 или выше
подключенный к Интернету

Возможность управления от смартфона

Подробности можно узнать здесь:



vk.com/znatok_ru



ok.ru/znatokru



youtube.com/znatok

ПРИВЕТСТВИЕ

Конструктор содержит все компоненты, которые используются в реальных современных **электромобилях**, в том числе и **роботизированных**.

При работе с данным набором используется профессиональный язык программирования C++. Можно использовать три способа управления: по Bluetooth от смартфона, по готовой программе, и автономный режим, когда автомобиль сам принимает решения. Конструктор будет интересен не только детям и их родителям, но и педагогам.

Это реальная помощь в образовании!

Мы ориентировались на возраст **13-18 лет**, но не удивимся, если этот диапазон окажется гораздо шире.

Пожалуйста, прежде чем приступить к выполнению проектов, внимательно прочитайте информацию в разделах «ВВЕДЕНИЕ», «ОПИСАНИЕ ДЕТАЛЕЙ» и «УСТАНОВКА ПРОГРАММ».

Это поможет избежать поломок, сэкономит много времени и нервов.

Предлагаемые **QR-коды** позволят увидеть ожидаемый результат или лучше понять принципы работы.

Хотелось бы выразить глубокую признательность людям, вложившим душу в осуществлении этого проекта: Анатолию Вожадеву, Владиславу Зыбенко, Константину Тарасову, Ли Джуню, Чжану Тяньго.

С уважением,
Андрей Бахметьев

P.S. Данная инструкция доступна в электронном виде на сайте www.znatok.ru в ОПИСАНИИ набора CAR-duino.



ВАЖНО! С целью экономии бумаги и бережного отношения к природе, весь раздел ПРОЕКТЫ тоже представлен на указанном сайте в электронном виде.



ВВЕДЕНИЕ

Конструктор абсолютно безопасен и прост в обращении, но, чтобы он прослужил Вам как можно дольше, следует соблюдать некоторые правила:



- Собирайте только те проекты, которые описаны в данной инструкции.
- Никогда не подсоединяйте схемы конструктора к электрическим сетям в вашем доме.
- Большинство проблем в электрических цепях связано с неправильной сборкой, всегда внимательно проверяйте, что ваша цепь действительно соответствует тому, что изображено на рисунке инструкции.
- Удостоверьтесь, что все соединения надежно защелкнуты или прикручены.
- Во время сборки электрической части проекта убедитесь, что выключатель на плате управления находится в положении **OFF**.
- Всегда отключайте аккумулятор, если он, или какой-то элемент стали сильно нагреваться!
- Несмотря на то, что аккумулятор из нашего набора имеет защиту от короткого замыкания, никогда не замыкайте его выход без нагрузки.

СОДЕРЖАНИЕ

• ПЕРЕЧЕНЬ ДЕТАЛЕЙ	4
• ОПИСАНИЕ ДЕТАЛЕЙ	8
• ОПИСАНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ C++	17
• УСТАНОВКА И ЗАГРУЗКА ПРОГРАММ	18
• Для пользователей WINDOWS	18
• Для пользователей MACOS	22
• СБОРКА	28
• ПОДКЛЮЧЕНИЕ И ТЕСТИРОВАНИЕ	34
• ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ ПО BLUETOOTH	52
• ПРОЕКТЫ	54
• ЭТО ИНТЕРЕСНО	55
• Широтно-импульсная модуляция	55
• RGB	56

ПЕРЕЧЕНЬ ДЕТАЛЕЙ

Название элемента	Количество	Изображение	Название элемента	Количество	Изображение
Платформа	1		Подшипник переднего колеса малый	2	
Аккумулятор 7.4V	1		Штифт крепления переднего колеса 2x10mm	2	
Колесо	4		Шестигранный адаптер переднего колеса	2	
Поворотный рычаг	2		Площадка крепления переднего колеса	2	
Передняя полуось	2		Шестигранный адаптер заднего колеса	2	
Подшипник переднего колеса большой	2		Задняя ось 120x3,95 mm	1	

0 cm 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 cm

ПЕРЕЧЕНЬ ДЕТАЛЕЙ

Название элемента	Количество	Изображение	Название элемента	Количество	Изображение
Подшипник задней оси	2		Качалка сервопривода (с винтом)	1	
Несущая скоба	1		Уголок-крепление сервопривода	2	
Мотор-редуктор	1		Рулевая тяга длинная	1	
Шестерня большая	1		Рулевая тяга короткая	1	
Шестерня малая	1		Плата управления	1	
Сервопривод	1		Блок задних огней	1	

ПЕРЕЧЕНЬ ДЕТАЛЕЙ

Название элемента	Количество	Изображение	Название элемента	Количество	Изображение
Блок передних огней	1		Датчик Холла	1	
Фара	2		Стойка верхних огней («люстры»)	2	
Ультразвуковой дальномер	1		Блок верхних огней («люстра»)	1	
Держатель ультразвукового дальномера	1		Гайка верхних огней («люстры»)	2	
Модуль Bluetooth	1		Стойка металлическая h11xM3	4	
Пьезоизлучатель	1		Стойка металлическая h22xM3	4	
Диск с магнитами	1		Стойка металлическая h40xM3	1	

0 cm 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 cm

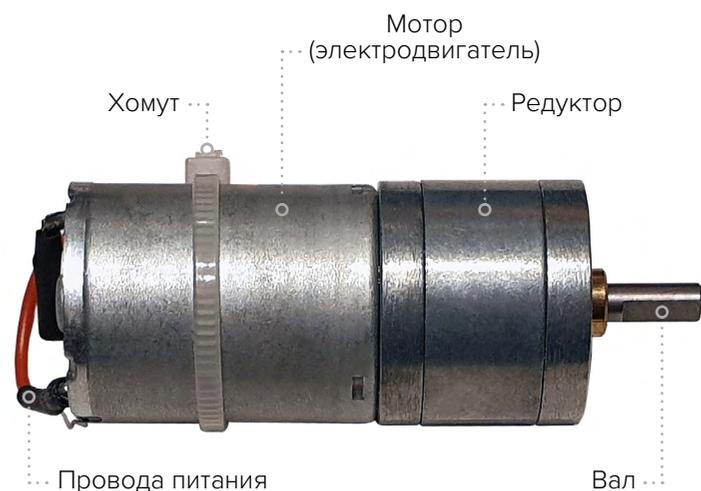
ПЕРЕЧЕНЬ ДЕТАЛЕЙ

Название элемента	Количество	Изображение	Название элемента	Количество	Изображение
Винт M1.5x8 mm	4		Ключ шестигранный	1	
Винт M2.5x10 mm	8		Отвертка крестовая	1	
Винт M3x5 mm	8		Отвертка крестовая малая	1	
Винт M3x8 mm	43		Отвертка плоская	1	
Винт M3x12 mm	2		USB-кабель для заряда аккумулятора	1	
Винт M4x6 mm	2		Кабель USB-miniUSB для подключения к компьютеру	1	
Винт M3 с потайной головкой	2		Кабель (2x17cm)	1	
Гайка M1.5	4		Плоский кабель (3x10cm)	2	
Гайка M3	32		Плоский кабель (3x14cm)	2	
Контргайка M2.5	1		Плоский кабель (4x14cm)	1	
Контргайка M4	2		Плоский кабель (5x12cm)	1	
Ключ универсальный	1		Кабель питания (10cm)	1	

ОПИСАНИЕ ДЕТАЛЕЙ

ПЛАТФОРМА ►

Выполнена из прочного стеклопластика, толщиной 1.6 мм и предназначена для крепления всех основных узлов и механизмов нашего автомобиля. В отличие от металлических платформ не проводит электричество, не деформируется, легко сверлится для установки дополнительного оборудования.



◀ МОТОР-РЕДУКТОР

Приводит в движение автомобиль. Скорость вращения контролируется при помощи энкодера. Красный и черный провода уже припаяны и притянуты пластиковым хомутом, чтобы не разрушить контакты.

ОПИСАНИЕ ДЕТАЛЕЙ

АККУМУЛЯТОР 7.4V

Li-ion аккумулятор с выходным напряжением 7.4V. Для заряда аккумулятора рекомендуется использовать стандартный USB сетевой адаптер 5V/2A (в набор не входит). Если зарядный ток будет меньше 2A, например, 5V/1A, то заряд аккумулятора будет осуществляться медленно и сам аккумулятор может сильно нагреваться. Заряд напряжением 5V возможен благодаря повышающей схеме внутри аккумулятора.

Во время заряда индикатор на аккумуляторе горит красным цветом, по окончании заряда, он гаснет.



Срок службы аккумулятора 7 лет при соблюдении правил эксплуатации (см. сайт www.znatok.ru, набор «CAR-duino», ОПИСАНИЕ, ИНСТРУКЦИЯ).

ОПИСАНИЕ ДЕТАЛЕЙ

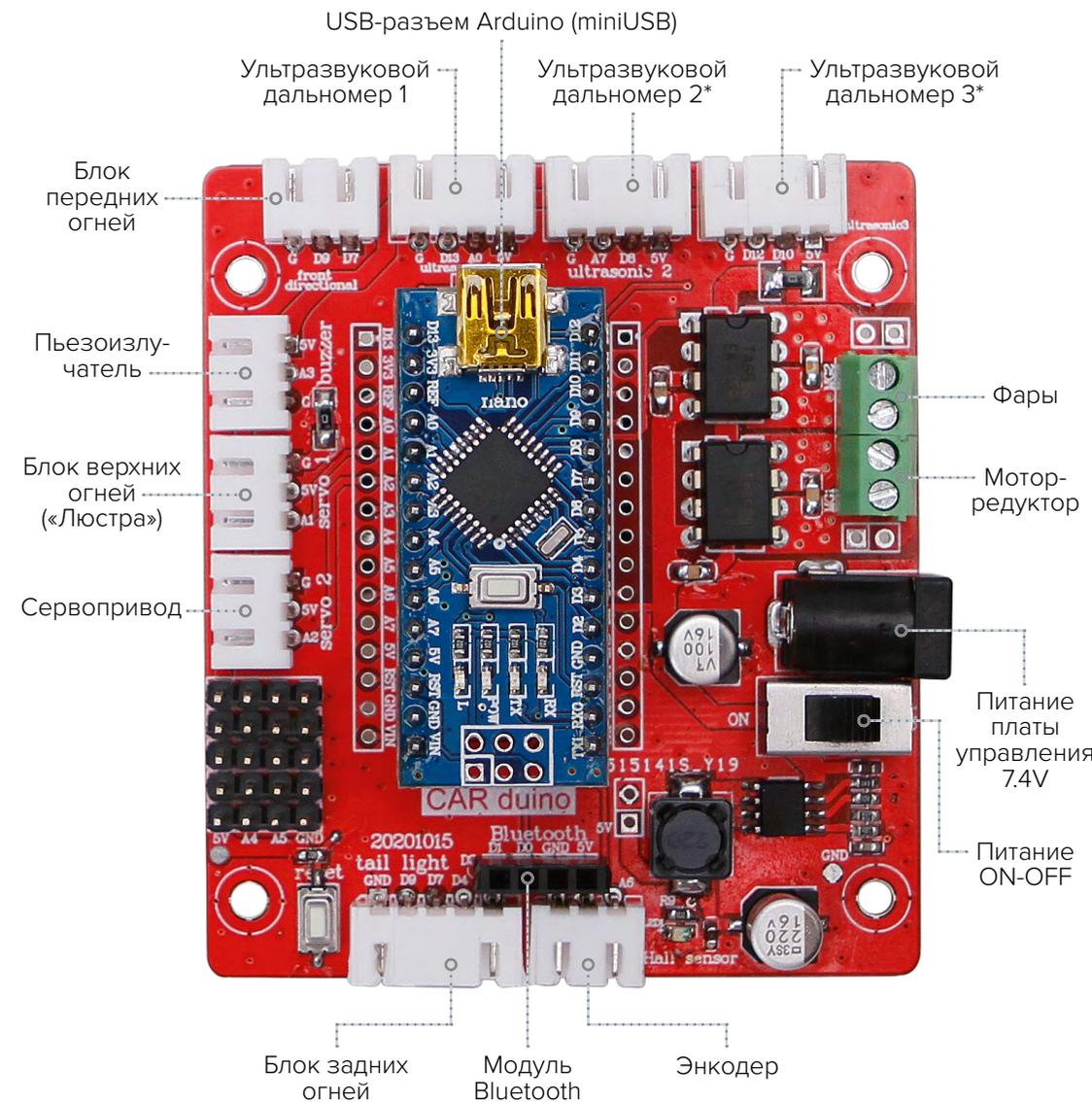
ПЛАТА УПРАВЛЕНИЯ

В основе платы управления лежит микроконтроллер Arduino NANO. Плата, на основе загруженной в микроконтроллер программы и получаемых данных от датчиков управляет автомобилем. Загрузка программ осуществляется при помощи кабеля USB-miniUSB (см. раздел УСТАНОВКА И ЗАГРУЗКА ПРОГРАММ, стр.18).



Кабель USB-miniUSB для подключения к компьютеру

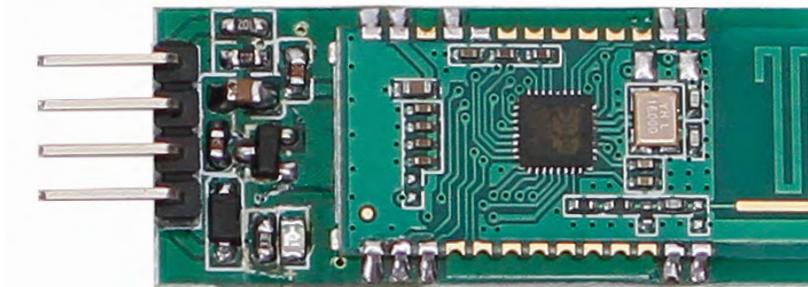
* В данный набор входит только один ультразвуковой дальномер.



ОПИСАНИЕ ДЕТАЛЕЙ

МОДУЛЬ BLUETOOTH ►

В нашем наборе модуль Bluetooth предназначен для управления автомобилем при помощи смартфона. Для этого надо установить на смартфон специальное приложение (см. раздел ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ ПО BLUETOOTH, стр.52). Дальность работы модуля зависит от окружающих условий и колеблется от 5 до 50 метров.



◀ ПЬЕЗОИЗЛУЧАТЕЛЬ

Полное название «пьезоэлектрический излучатель», но его так же называют пьезо, зуммер (summer), бужер (buzzer). Используется для воспроизведения звука или ультразвука. Пьезоизлучатель также может использоваться в качестве пьезоэлектрического микрофона или датчика. В нашем наборе он воспроизводит звуковые сигналы.

ОПИСАНИЕ ДЕТАЛЕЙ

ЭНКОДЕР

Энкодер (encoder) предназначен для измерения скорости передвижения автомобиля и измерения пройденного расстояния. Состоит из диска с 6 магнитами и платы, на которой установлен датчик Холла и логическая схема, формирующая выходные импульсы. Чем выше скорость вращения диска с магнитами, тем выше частота выходных импульсов. А зная количество импульсов и диаметр колеса, можно подсчитать пройденное расстояние.

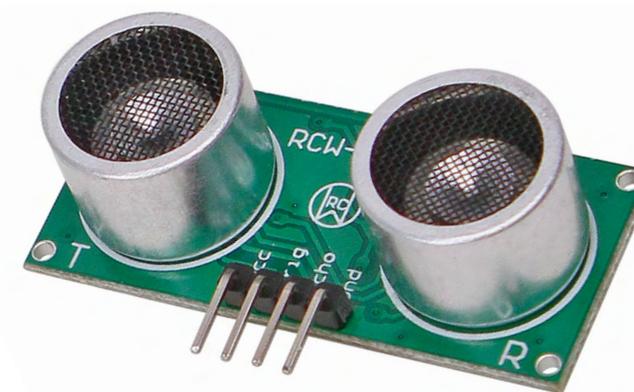
Принцип работы энкодера можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Encoder>



ОПИСАНИЕ ДЕТАЛЕЙ

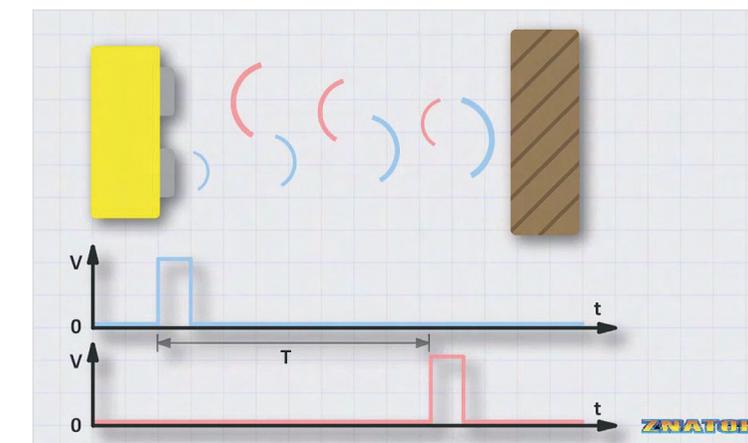
УЛЬТРАЗВУКОВОЙ ДАЛЬНОМЕР

Еще одно его название «ультрасоник» (ultrasonic). Он используется в эхолотах, гидролокаторах, радарх, автономных роботах, автомобильных парктрониках. Принцип работы ультразвукового дальномера заключается в следующем – передатчик (Т) посылает ультразвуковой сигнал, который отражается от объекта, возвращается назад и улавливается приемником (R). По разнице во времени между посылаемым сигналом и отраженным (эхо), рассчитывают расстояние. Чем больше разница во времени, тем больше расстояние.



Принцип работы ультразвукового дальномера можно посмотреть, перейдя по ссылке:

https://znatok.ru/link/?Ultrasonic_HowTo



Есть большая проблема, когда препятствие находится под углом и отраженные волны не попадают на приёмник. Поэтому мы предусмотрели установку дополнительных ультразвуковых дальномеров. Узнать подробнее о проблеме можно, перейдя по ссылке:

https://znatok.ru/link/?Ultrasonic_Problem

ОПИСАНИЕ ДЕТАЛЕЙ

СВЕТОВЫЕ ПРИБОРЫ

- **БЛОК ЗАДНИХ ОГНЕЙ.** Включает в себя поворотные огни («поворотники»), стоп-сигналы, фонарь заднего хода.



- **БЛОК ПЕРЕДНИХ ОГНЕЙ.** Включает в себя поворотные огни. К отверстиям рядом с разъемами крепятся фары.



ОПИСАНИЕ ДЕТАЛЕЙ

- **ФАРЫ.** Мощные светодиодные фары стандартно устанавливаются на блок передних огней и там же подключаются к разъемам.



- **БЛОК ВЕРХНИХ ОГНЕЙ.** Такие огни часто называют «люстра». Представляет собой 8 RGB-светодиодов, которые могут воспроизводить все возможные цвета, имитируя световые сигналы скорой помощи, полицейской машины, safety car, дополнительные огни освещения дороги. «Люстра» крепится на специальные стойки и фиксируется пластиковыми гайками. Может фиксироваться под разными углами – вперёд, вверх, назад.

НЕЛЬЗЯ ДОЛГО СМОТРЕТЬ НА ГОРЯЩИЕ ФАРЫ И «ЛЮСТРУ»!

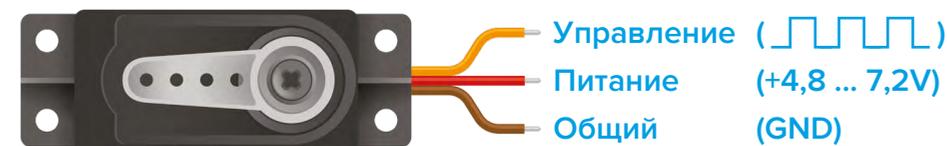
ОПИСАНИЕ ДЕТАЛЕЙ

СЕРВОПРИВОД

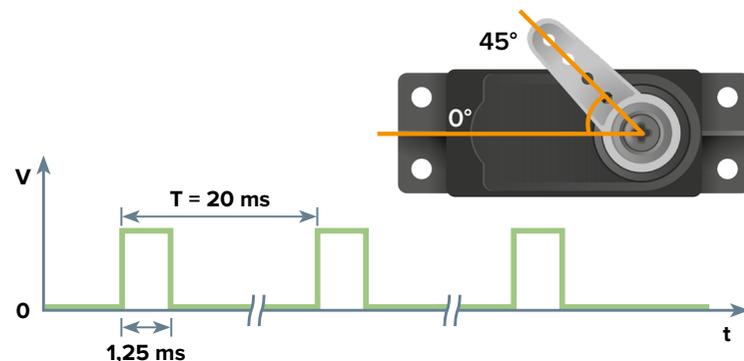
Его часто называют просто серво (servo). В зависимости от сигнала, приходящего на вывод «Управление» вал сервопривода может поворачиваться и фиксироваться в определённом положении. В данном случае сигналом управления является импульс заданной длительности. Вал нашего сервопривода может поворачиваться на 270°, а так как платформа позволяет поворачиваться «качалке» не более чем на 180°, требуется регулировка.

КАК ЭТО РАБОТАЕТ

Для полного поворота на 180° необходимо на вход управления подать импульс, длительность которого t меняется от 1 до 2 миллисекунд (ms) при постоянном периоде T . Когда на вход управления приходят импульсы длительностью 1ms, «качалка», закрепленная на валу, находится в положении 0°. По мере увеличения длительности импульса вал, с небольшой задержкой, будет поворачиваться. Так, при длительности импульса 1.25ms вал повернется на 45°, при 1.5ms – 90°, при 2ms – 180°.



Принцип работы сервопривода можно посмотреть, перейдя по ссылке: https://znatok.ru/link/?Servo_HowTo



ОПИСАНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ



Для программирования на языке C++ предлагается использование среды Arduino IDE, включающую в себя библиотеки для работы с микроконтроллерами. Большая часть программ, написанных для различных устройств, была написана на языках C/C++, а использование модулей Arduino может быть первым серьезным шагом в изучении программирования и микроконтроллеров.

Все программы предоставленные в этом пособии, которые вы можете найти на флэшке, изначально написаны именно для Arduino IDE, и в книге вы можете найти примеры именно для этого варианта.

Примечание: Компания «ЗНАТОК» не занимается разработкой и поддержкой языка программирования C++ и среды программирования Arduino IDE.

Для получения инструкции по работе с языком C++ в данном наборе пройдите по ссылке на сайте www.znatok.ru («CAR-duino», ОПИСАНИЕ, ИНСТРУКЦИЯ): <https://znatok.ru/link/?CARduino-cpp> или скитайте смартфоном этот QR-код:



```
01_Control_LED-10
1 #define BUTTON_PIN 7
2 #define LED_RED 5
3
4 void setup()
5 {
6   pinMode(LED_RED, OUTPUT);
7   pinMode(BUTTON_PIN, INPUT_PULLUP);
8 }
9
10 void loop()
11 {
12   if (digitalRead(BUTTON_PIN) == LOW)
13   {
14     digitalWrite(LED_RED, HIGH);
15   }
16   else
17   {
18     digitalWrite(LED_RED, LOW);
19   }
20 }
21
```

ДЛЯ ПОЛЬЗОВАТЕЛЕЙ WINDOWS

1. По ссылке <https://znatok.ru/link/?Carduino-WorkingFiles-Windows> или на странице набора «Знаток. Car-duino» на сайте znatok.ru скачайте рабочие файлы в архиве.

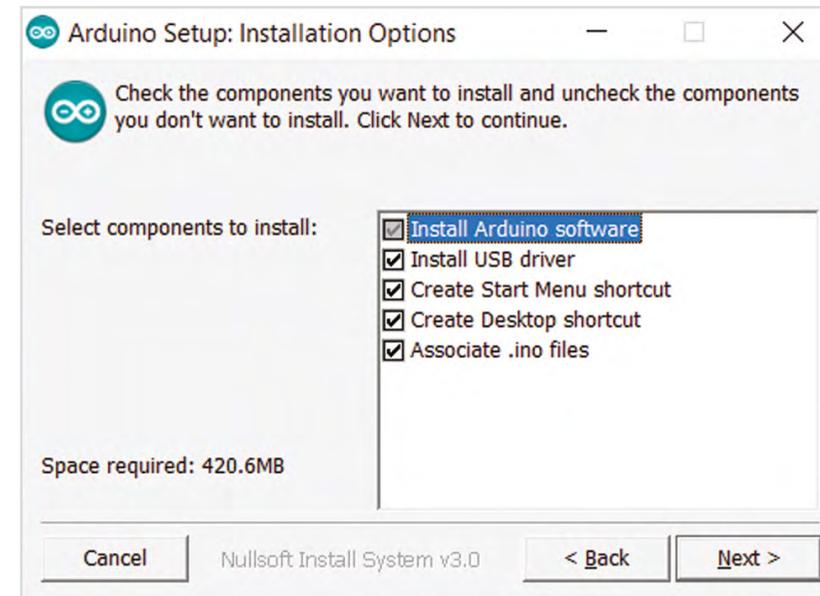
2. Распакуйте архив, в его состав входит:

- Расширенное руководство пользователя: описание подготовки к работе, а так же подробные описания Проектов
- Папка «**Laboratory Projects**», содержащая проекты на языке программирования C++
- Папка «**drivers**», содержащая драйверы для работы с набором
- Папка «**installers**», содержащая файлы установки для Arduino IDE

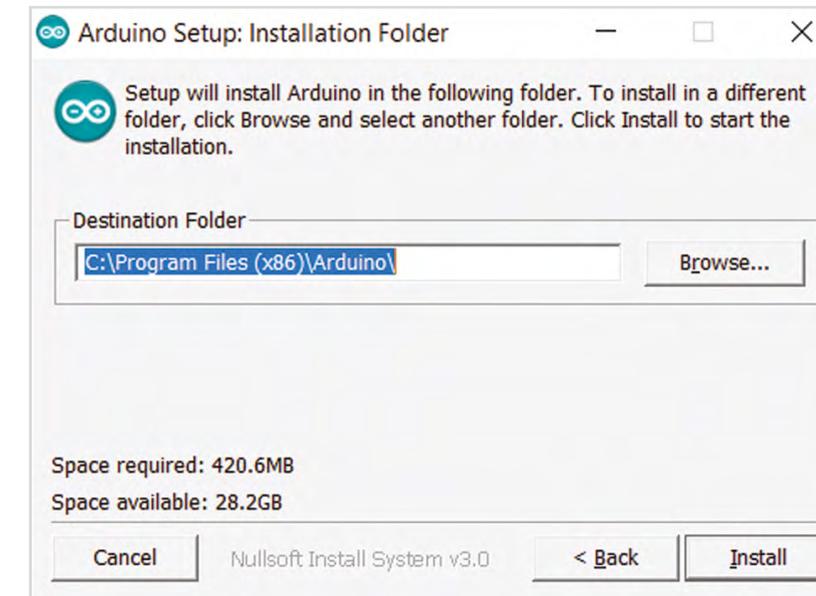
3. Для установки среды программирования Arduino IDE, откройте файл «arduino-1.X.YY-windows.exe» (X.YY – любые цифры, обозначающие версию), расположенную в папке installers

3.1. Следуйте инструкциям установщика: на первом появившемся экране нажмите I agree («Я согласен»).

3.2. Обязательно отметьте пункт Install USB driver. Нажмите Next («Далее»).



3.3. Выберите путь установки (рекомендуем предложенный). Нажмите Install («Установить»).



3.4. Дождитесь конца установки, появится надпись Completed, нажмите Close («Закрыть»), после чего на рабочем столе появится ярлык Arduino.



4. Для установки драйверов (необходимо, если вы их не установили в процессе установки Arduino IDE или возникли сложности с загрузкой программ на модуль Arduino) откройте файл «setup.exe» в папке «drivers/CH341SER».

5. Подключите при помощи USB-кабеля Модуль Arduino к компьютеру. Произойдет установка оборудования. Если у вас отключена функция автоматического поиска драйверов, то в ручном режиме их можно найти в распакованном архиве в папке drivers.

5.1. Запустите Arduino IDE (при помощи ярлыка на рабочем столе).

5.2. В меню **Инструменты** выберите подпункт **Плата** и в выпадающем списке выберите **Arduino Nano**.

5.3. В меню **Инструменты** выберите подпункт **Порт** и в выпадающем списке выберите необходимый порт (часто, бывает один), например, COM6.

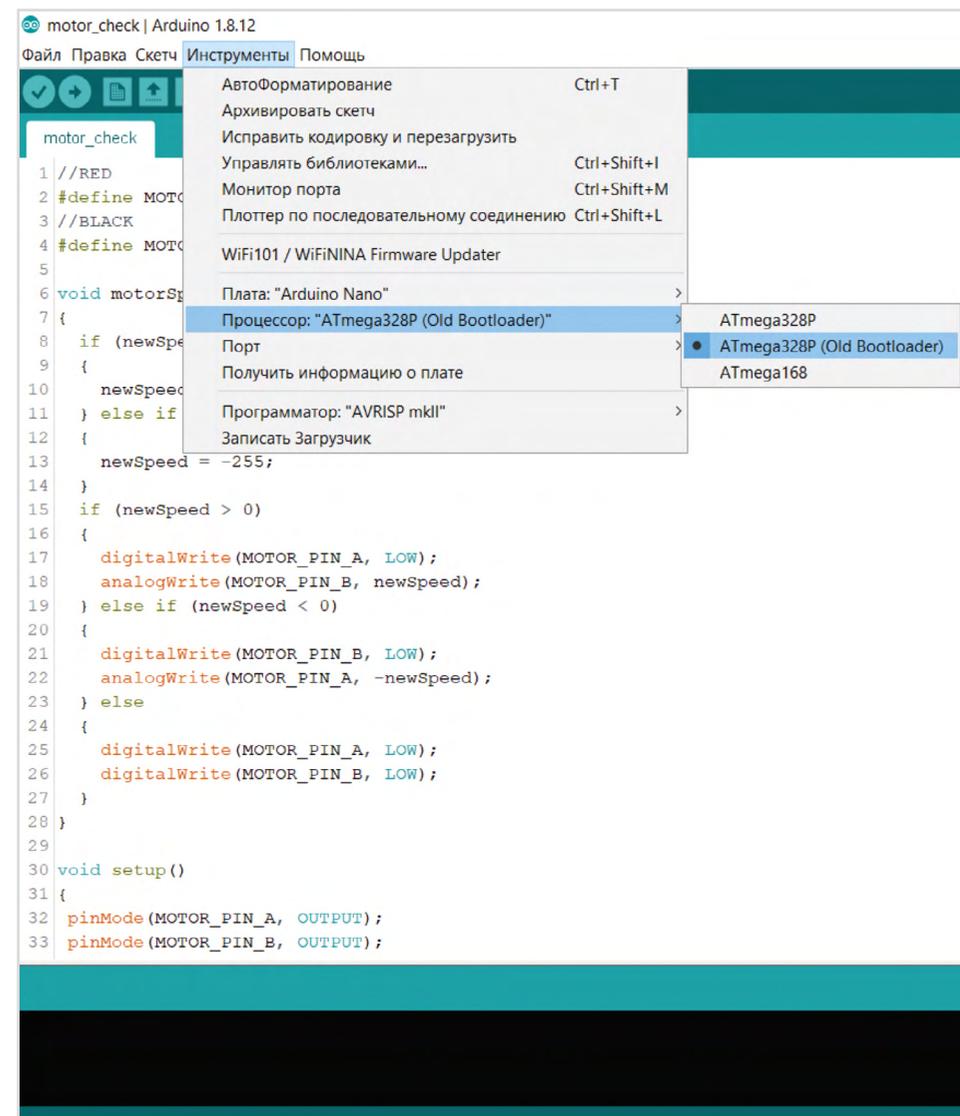
ДЛЯ ПОЛЬЗОВАТЕЛЕЙ WINDOWS

5.4. Выберите процессор:

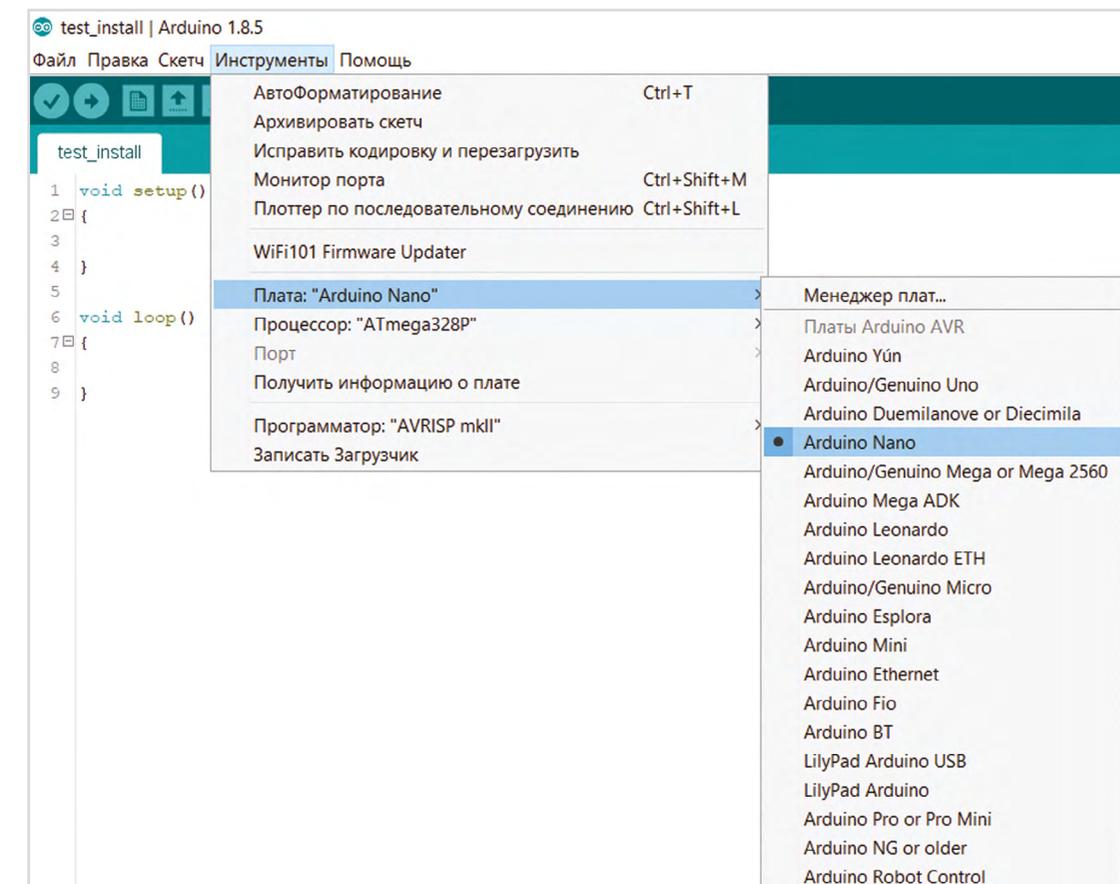
**Инструменты –
Процессор –
Atmega328P (Old Bootloader)**

ИЛИ

**Tools –
Processor –
Atmega328P (Old Bootloader)**



5.5. В меню **Скетч** выберите подпункт **Загрузка**, после чего начнется загрузка программы на Модуль Arduino (в данный момент программа тестовая, ничего выполняться не будет), если все шаги были выполнены верно, то в строке состояния появится сообщение «Загрузка завершена». Если возникла проблема загрузки на Модуль Arduino, то попробуйте изменить Порт (см. п.5.3).

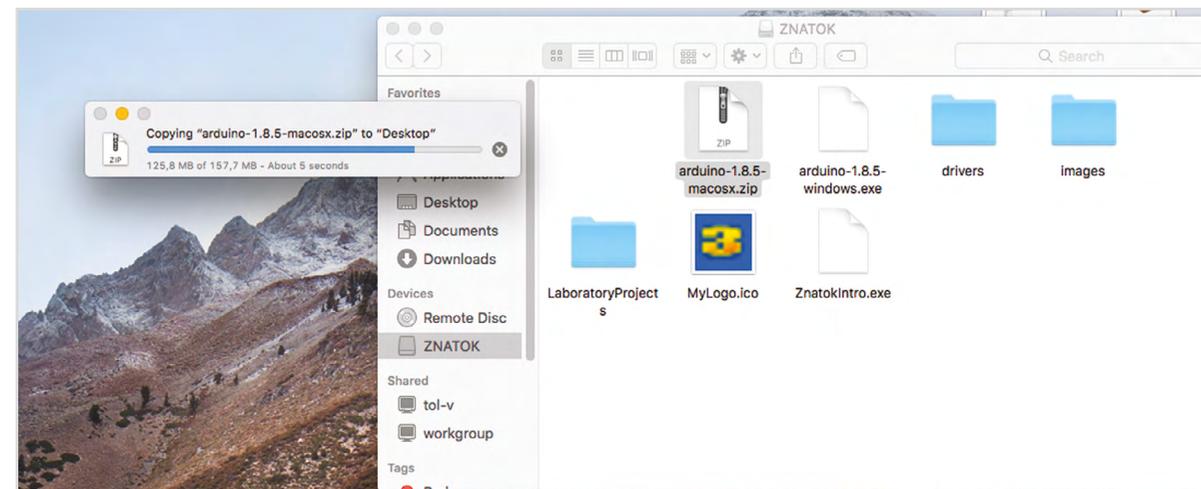


6. Для выполнения лабораторной работы откройте папку «**Laboratory Projects**». В папке «Arduino» расположены лабораторные работы, выберите нужную – у вас откроется Arduino IDE с текстом программы.

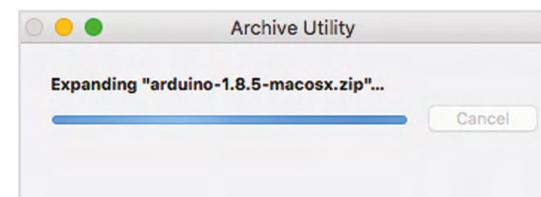
6.1. Для загрузки программы в Arduino необходимо выбрать в меню **Скетч** подпункт **Загрузка** или кликнуть по соответствующей пиктограмме. **НЕ ЗАБУДЬТЕ:** при ошибке загрузки проверьте правильность подключения Платы управления при помощи USB-кабеля к вашему компьютеру, а так же правильность выбора порта (см. п.5.3)

ДЛЯ ПОЛЬЗОВАТЕЛЕЙ MACOS

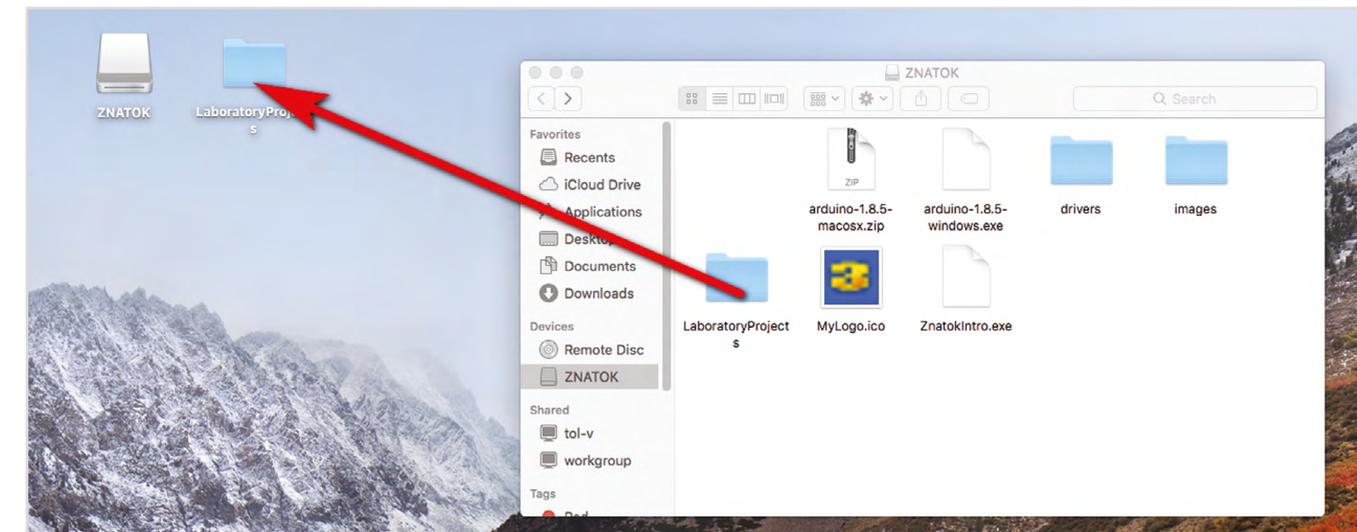
- По ссылке <https://znotok.ru/link/?Carduino-WorkingFiles-macOS> или на странице набора «ЗнатоК. Car-duino» на сайте znotok.ru скачайте рабочие файлы в архиве. Распакуйте архив, в его состав входит:
 - Расширенное руководство пользователя: описание подготовки к работе, а так же подробные описания Проектов
 - Папка «**Laboratory Projects**», содержащая проекты на языке программирования C++
 - Папка «**drivers**», содержащая драйверы для работы с набором
 - Папка «**installers**», содержащая файлы установки для Arduino IDE
- Скопируйте архив arduino-1.X.YY-macosx.zip (X.YY – любые цифры, обозначающие версию), например, на рабочий стол.



- Разархивируйте файл arduino-1.X.YY-macosx.zip в нужное вам место (рекомендуем «Application»), после чего приложение станет доступным для запуска.



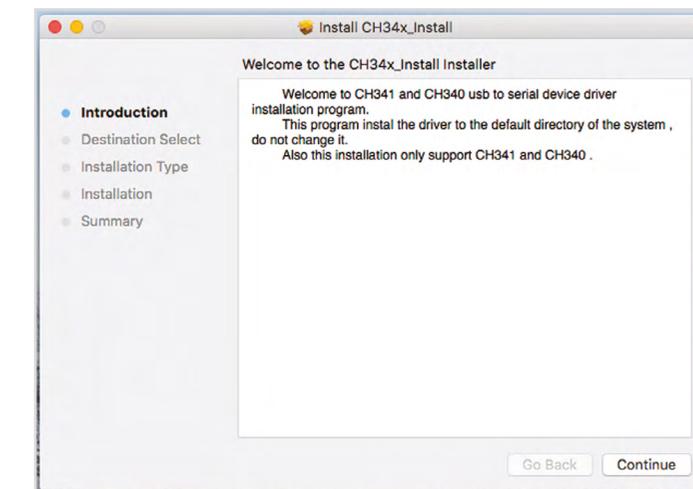
- Скопируйте папку с лабораторными работами Laboratory-Projects в любое удобное вам место.



- В папке drivers\CH341SER_MAC, на прилагаемой флешке, откройте файл установщика:

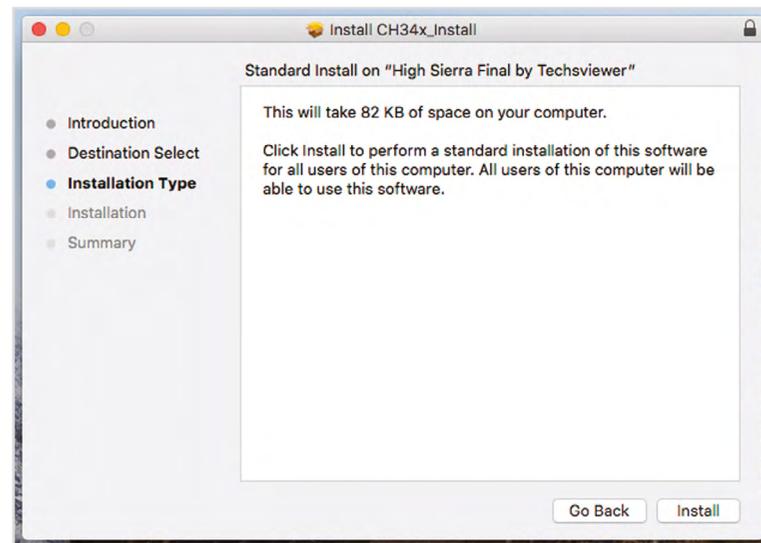


- Следуйте инструкциям установщика:

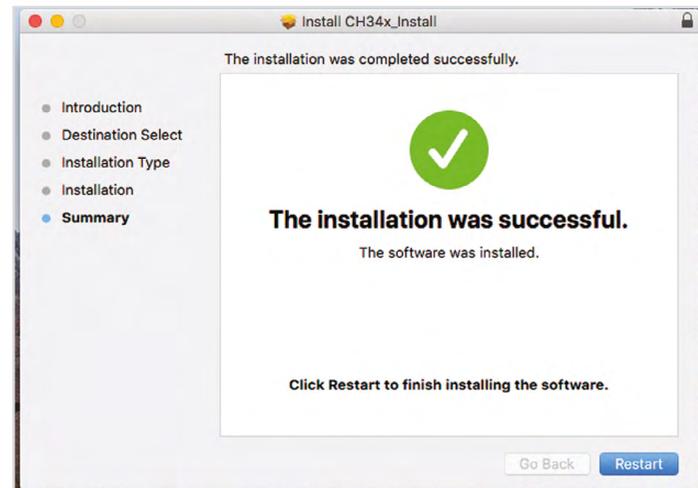
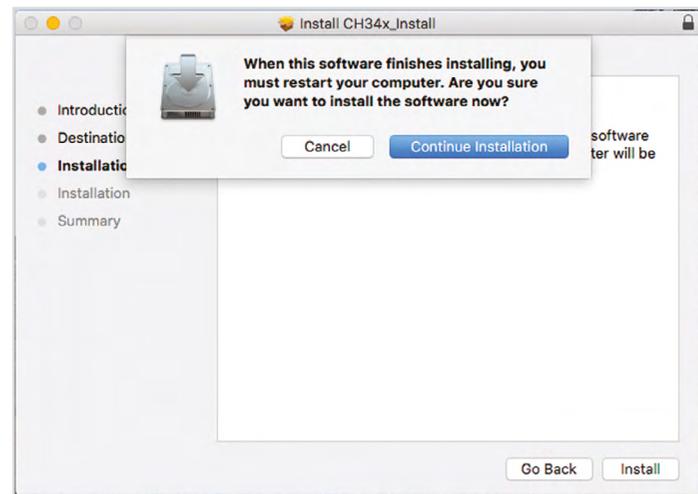


ДЛЯ ПОЛЬЗОВАТЕЛЕЙ MACOS

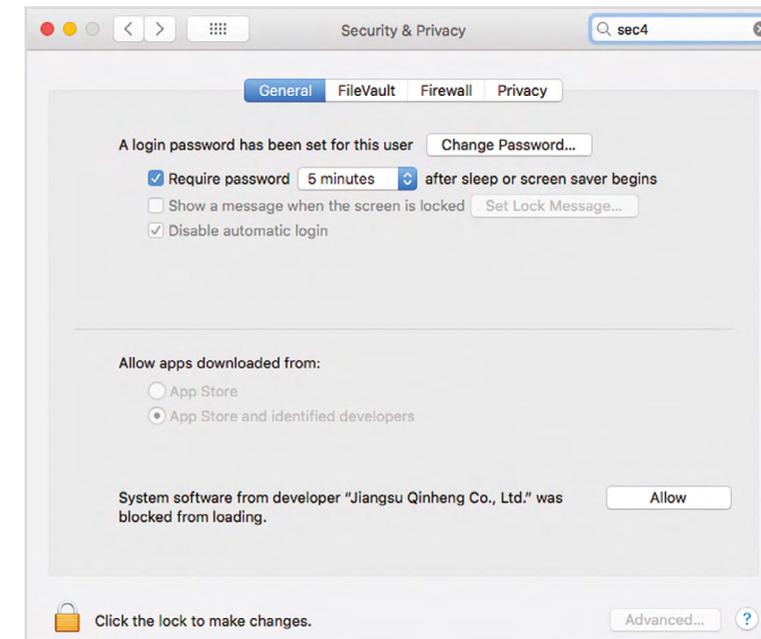
6.1. В процессе установки потребуется ввести пароль пользователя системы:



6.2. После установки, компьютер требуется перезагрузить.

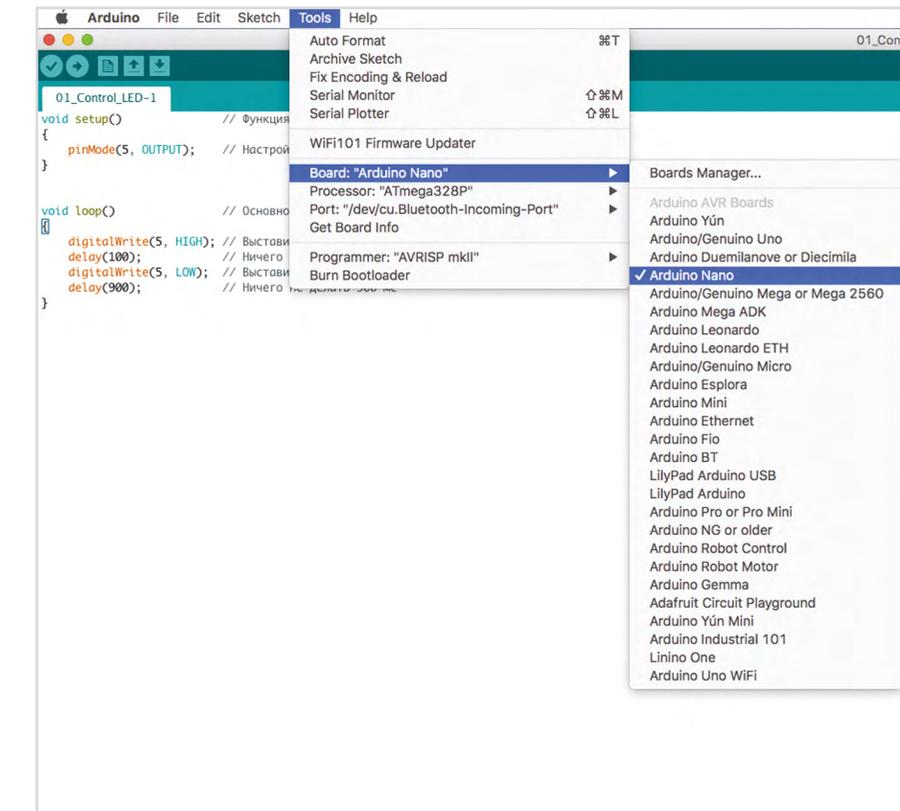


7. В настройках Security&Privacy вашего компьютера необходимо нажать кнопку Allow для корректной работы драйверов.



8. Запустите Arduino IDE (из папки Applications, если вы установили программу туда).

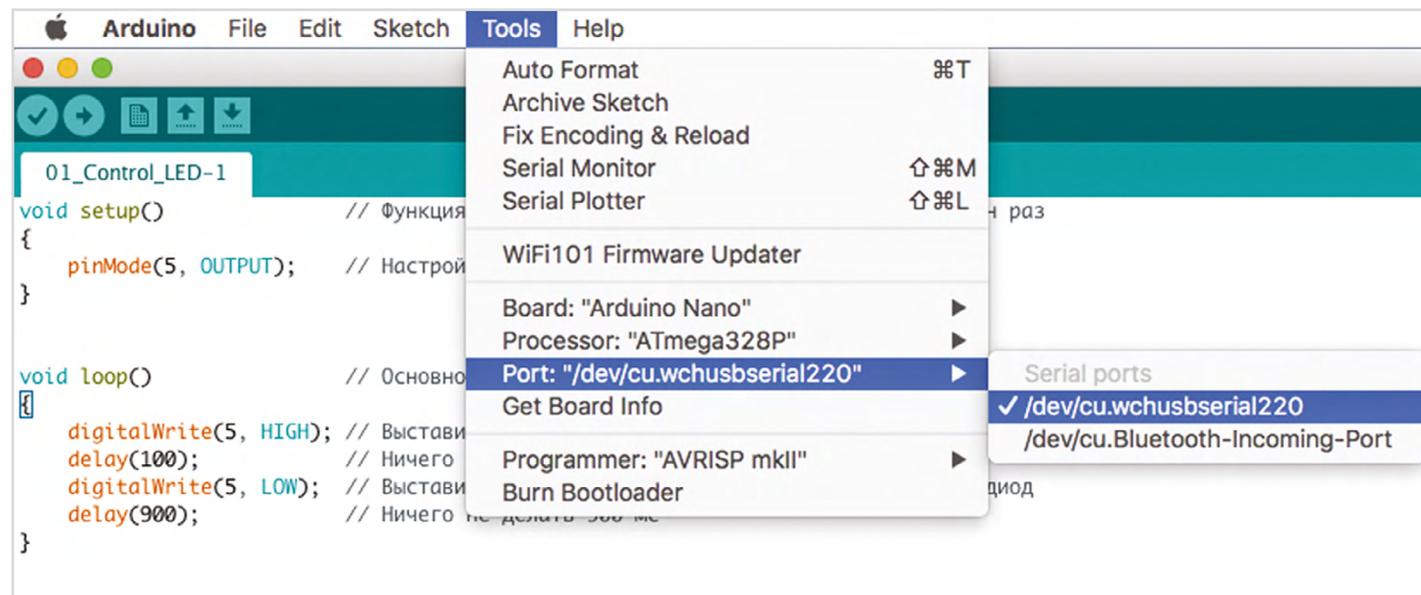
8.1. В меню Tools выберите подпункт Board и в выпадающем списке выберите Arduino Nano.



ДЛЯ ПОЛЬЗОВАТЕЛЕЙ MACOS

8.2. В меню Tools выберите подпункт Port и в выпадающем списке выберите подпункт, указанный на картинке.

8.3. В меню Sketch выберите подпункт Upload, после чего начнется загрузка программы на Модуль Arduino (в данный момент программа тестовая), если все шаги были выполнены верно, то в строке состояния появится сообщение «Загрузка завершена». Если возникла проблема загрузки на Модуль Arduino, то попробуйте изменить Порт (п.8.2).

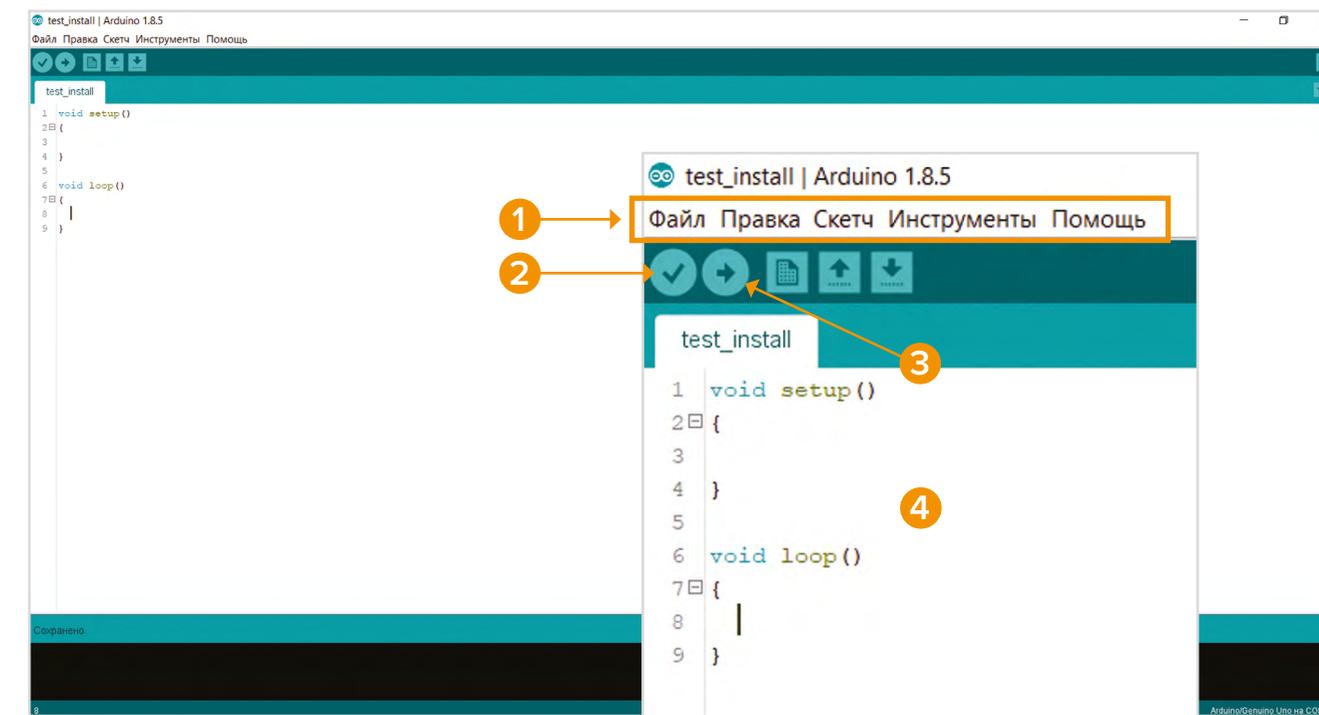


9. Для выполнения лабораторной работы, откройте папку лабораторные работы Znatok Arduino на вашем компьютере. Выберите нужную лабораторную работу и откройте программу – у вас откроется Arduino IDE с текстом программы. Если Arduino IDE уже открыта, то вы можете в меню File выбрать подпункт Open и выбрать нужную программу.

9.1. Для загрузки программы в Arduino необходимо выбрать в меню Sketch подпункт Upload или кликнуть по соответствующей пиктограмме. **НЕ ЗАБУДЬТЕ:** при ошибке загрузки проверьте правильность подключения модуля 111 при помощи USB-кабеля к вашему компьютеру, а так же правильность выбора порта (п.8.2).

ЗАГРУЗКА ПРОГРАММ

Процесс загрузки программы на модуль Arduino вы можете посмотреть, перейдя по ссылке <https://znatok.ru/link/?Carduino-Upload> или отсканировав приведенный QR-код.

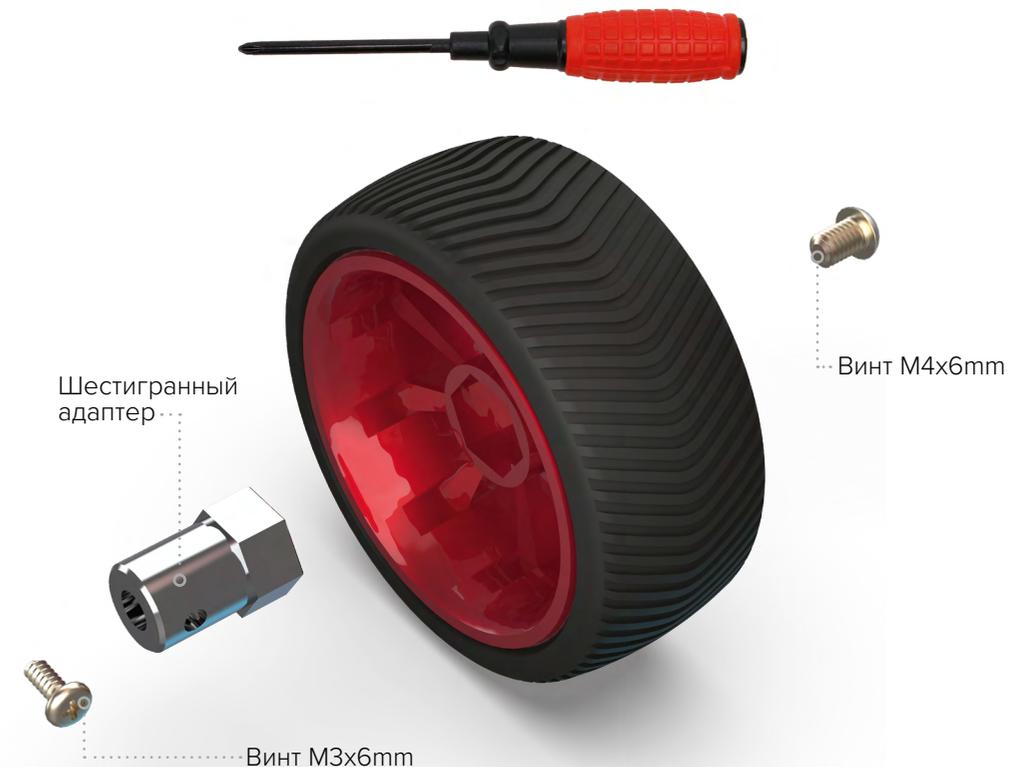


ОПИСАНИЕ ИНТЕРФЕЙСА:

1. Меню.
2. Кнопка проверки программы. В программе ищутся синтаксические (НЕ ЛОГИЧЕСКИЕ) ошибки.
3. Кнопка загрузки программы в модуль Arduino.
4. Основное рабочее поле – тут помещается весь код программы.
5. Строка состояния – показывает стадию загрузки программы.

СБОРКА ЗАДНЕГО КОЛЕСА

- Видеоинструкцию по сборке и установке можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Assemble-BackWheel>
- Для сборки заднего колеса потребуется отвёртка крестовая:



- Конечный результат:



Аналогично собирается и другое заднее колесо.

СБОРКА ПЕРЕДНЕГО КОЛЕСА

- Видеоинструкцию по сборке и установке можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Assemble-FrontWheel>
- Для сборки переднего колеса потребуется ключ универсальный:



- Конечный результат:



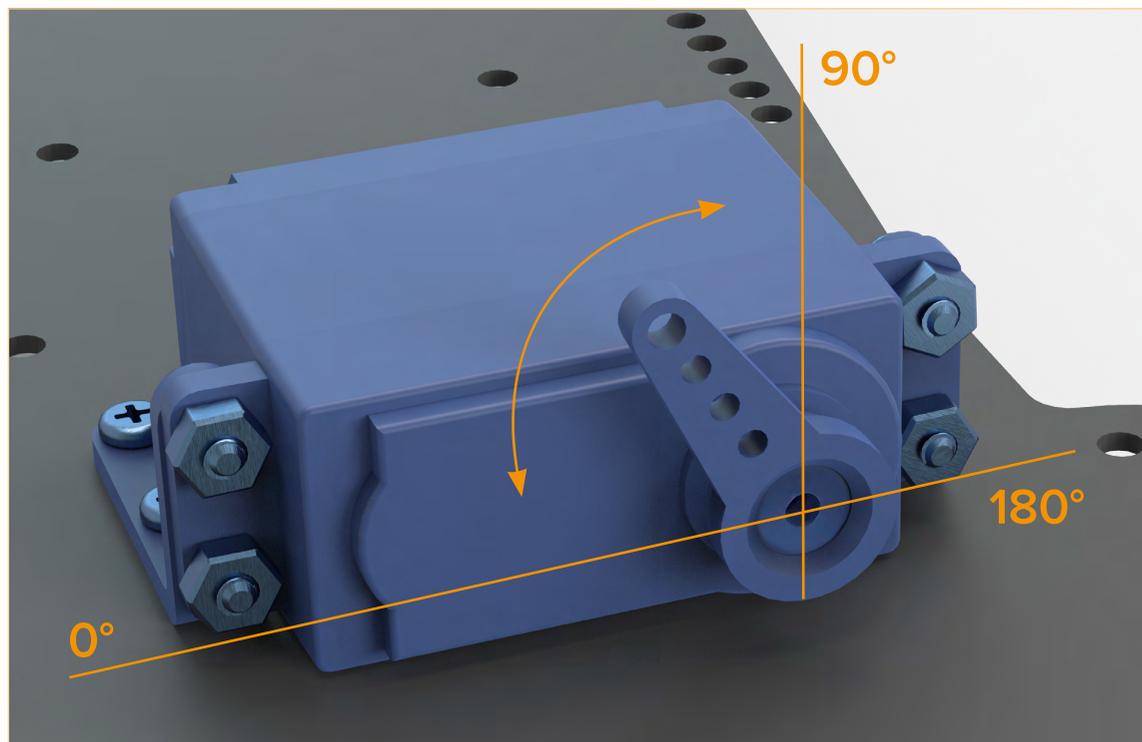
Аналогично собирается и другое заднее колесо.

РЕГУЛИРОВКА СЕРВОПРИВОДА

ЦВЕТ И ДИЗАЙН НЕКОТОРЫХ ДЕТАЛЕЙ И ИНСТРУМЕНТОВ МОЖЕТ ОТЛИЧАТЬСЯ ОТ ПРИВЕДЁННЫХ НА ВИДЕО И ФОТО, НО ЭТО НИКАК НЕ ВЛИЯЕТ НА РАБОТОСПОСОБНОСТЬ. !

Прежде чем закреплять «качалку» необходимо найти её правильное положение:

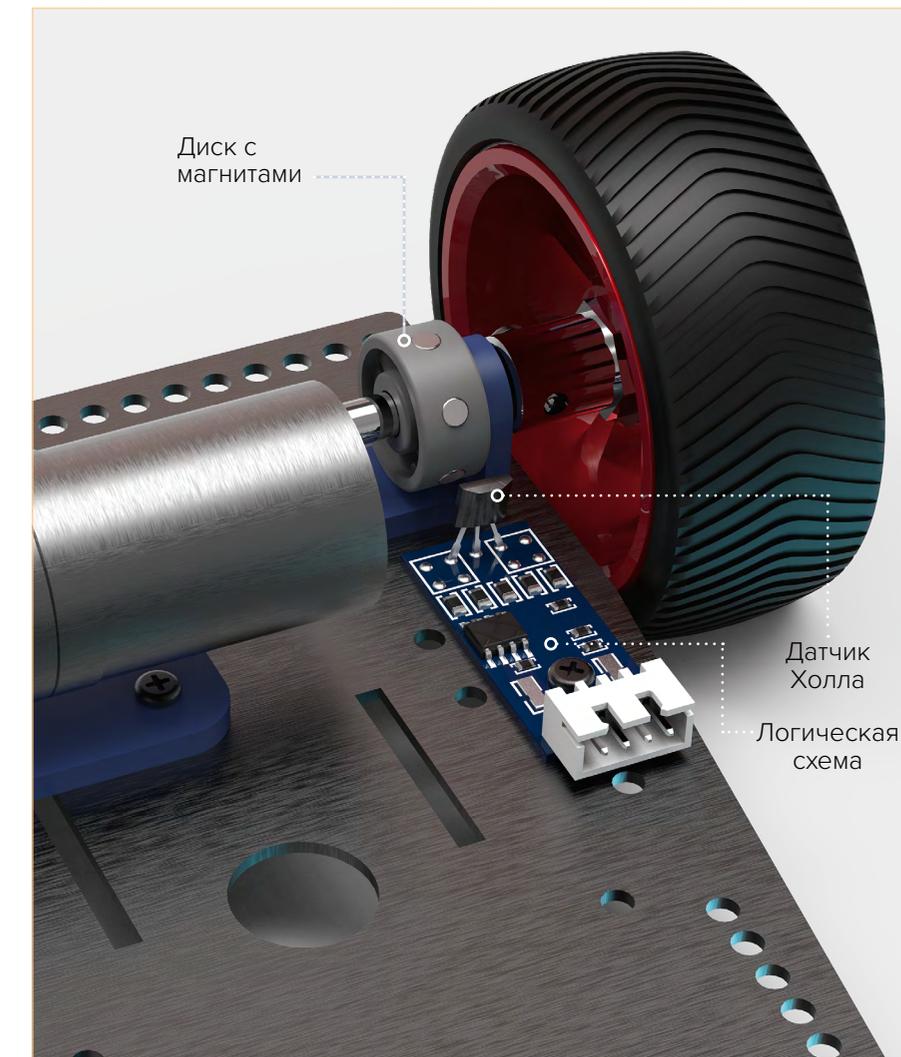
1. Установите «качалку», не закрепляя её винтом, на вал сервопривода в положение 90° (см. рисунок).
2. Медленно, прилагая умеренные усилия, попробуйте повернуть «качалку» до положений 0° и 180° (см. рисунок). Выполняйте это с осторожностью, чтобы не повредить внутренний механизм сервопривода.
3. Если положения 0° или 180° не достигаются, снимите «качалку» и закрепите её под другим углом, так чтобы она смогла поворачиваться от 0° до 180° , как показано на рисунке. Повторите п.2.



РЕГУЛИРОВКА ЭНКОДЕРА

ЦВЕТ И ДИЗАЙН НЕКОТОРЫХ ДЕТАЛЕЙ И ИНСТРУМЕНТОВ МОЖЕТ ОТЛИЧАТЬСЯ ОТ ПРИВЕДЁННЫХ НА ВИДЕО И ФОТО, НО ЭТО НИКАК НЕ ВЛИЯЕТ НА РАБОТОСПОСОБНОСТЬ. !

После закрепления платы с датчиком Холла на платформе при помощи винта и гайки, необходимо передвинуть диск с магнитами на задней оси так, чтобы магниты оказались напротив самого датчика.



СБОРКА ШАССИ

ЦВЕТ И ДИЗАЙН НЕКОТОРЫХ ДЕТАЛЕЙ И ИНСТРУМЕНТОВ МОЖЕТ ОТЛИЧАТЬСЯ ОТ ПРИВЕДЁННЫХ НА ВИДЕО И ФОТО, НО ЭТО НИКАК НЕ ВЛИЯЕТ НА РАБОТОСПОСОБНОСТЬ. !

- Видеоинструкцию по сборке можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Assemble-Chassis>
- Для сборки шасси потребуются следующие инструменты:



Ключ шестигранный



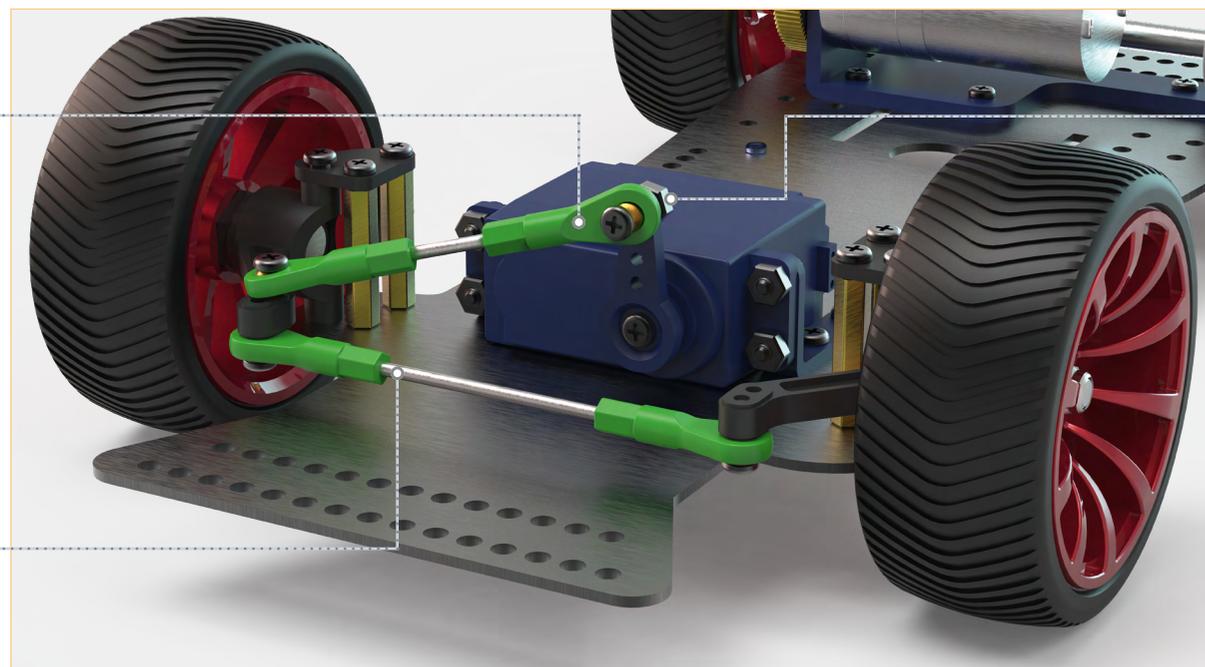
Ключ универсальный



Отвёртка крестовая

Длиной этой тяги регулируется угол поворота правого колеса относительно вала серво-привода

Длиной этой тяги регулируется параллельность колес

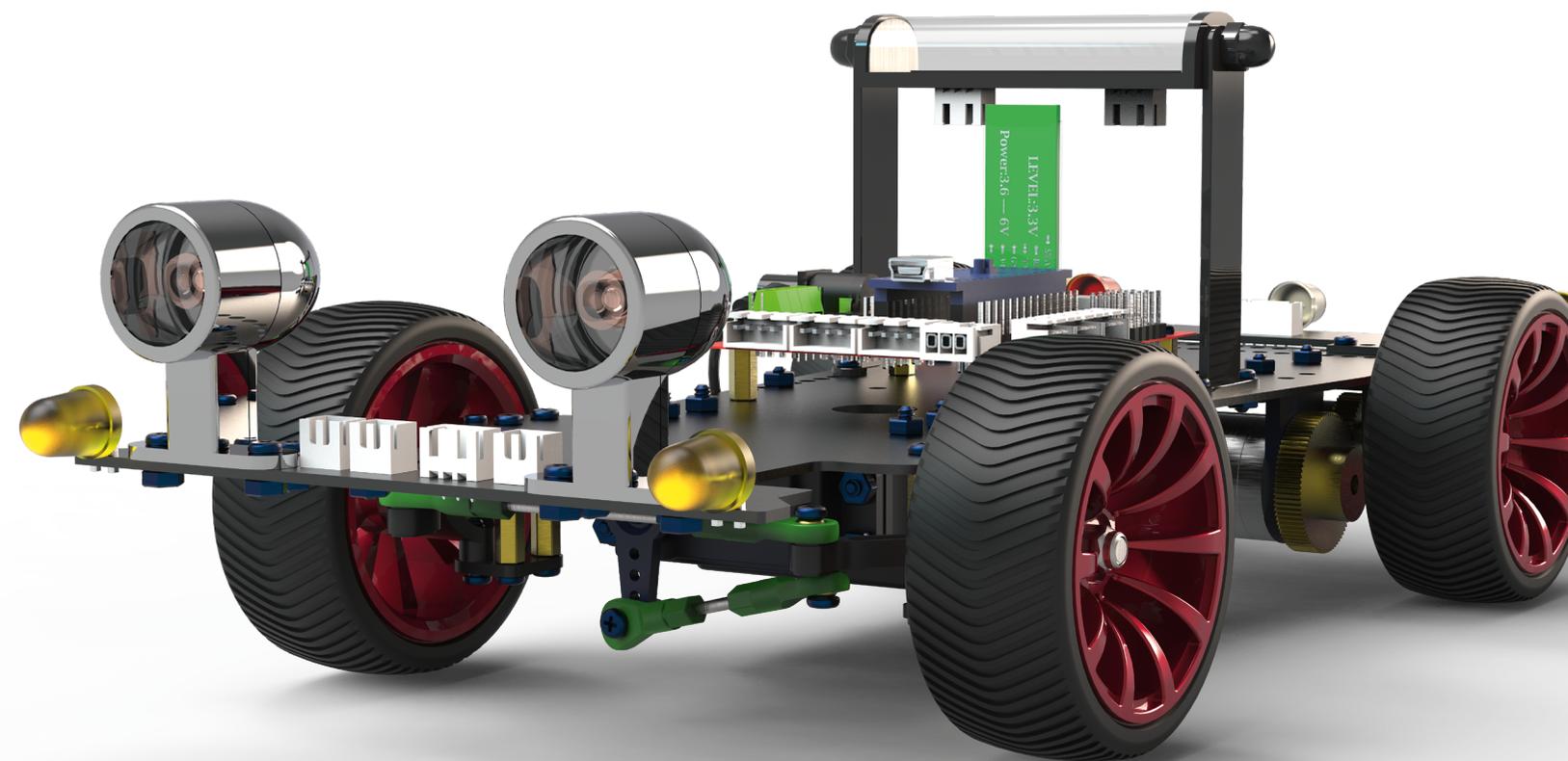


Контргайка M2.5



ПОСЛЕДОВАТЕЛЬНОСТЬ СБОРКИ МОЖЕТ БЫТЬ РАЗЛИЧНОЙ. ПРЕДСТАВЛЕН ОДИН ИЗ ВОЗМОЖНЫХ ВАРИАНТОВ.

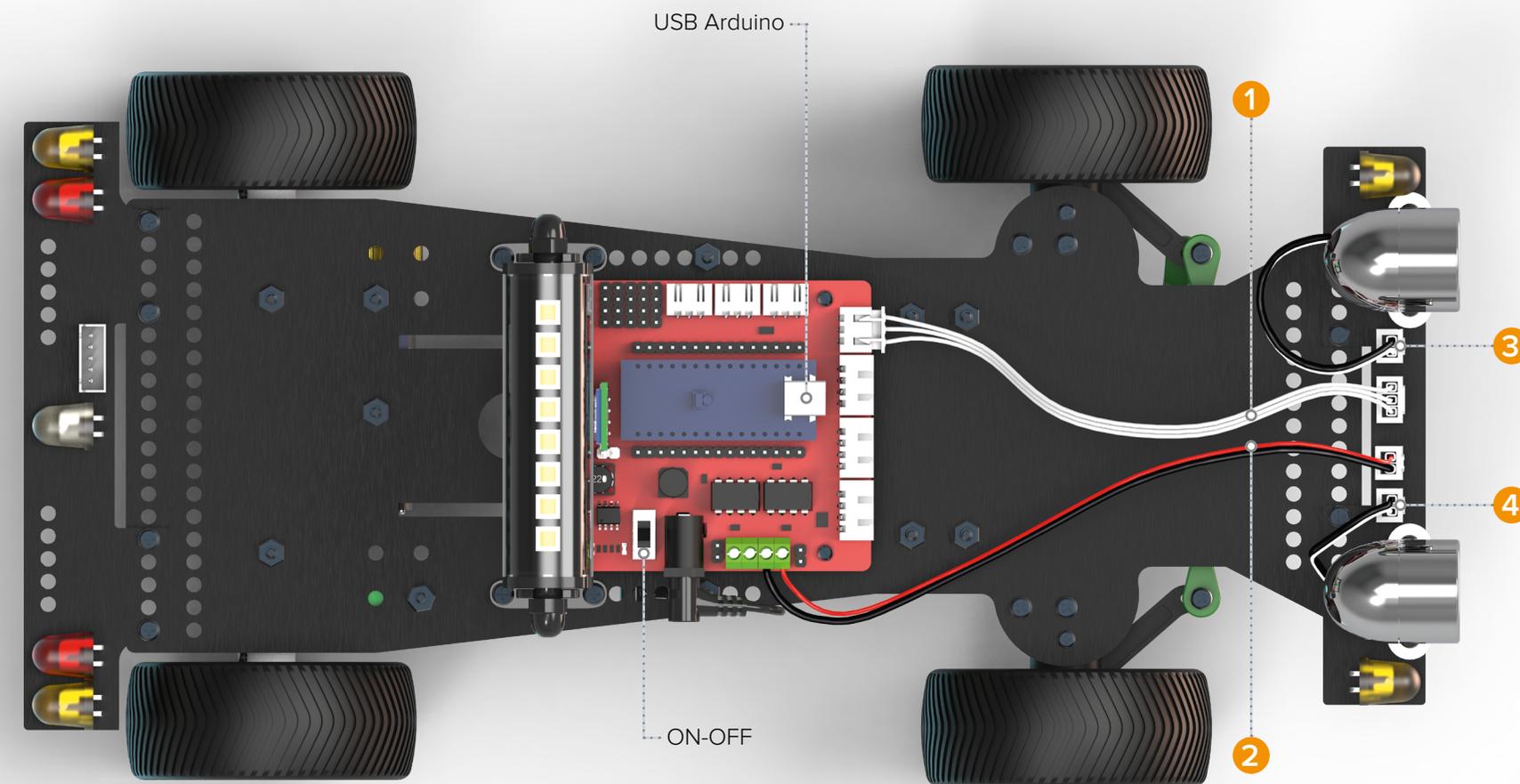
- Конечный результат:



ПОСЛЕДОВАТЕЛЬНОСТЬ ПОДКЛЮЧЕНИЯ МОЖЕТ БЫТЬ РАЗЛИЧНОЙ. НИЖЕ ПРЕДЛОЖЕН ОДИН ИЗ ВАРИАНТОВ.

БЛОК ПЕРЕДНИХ ОГНЕЙ

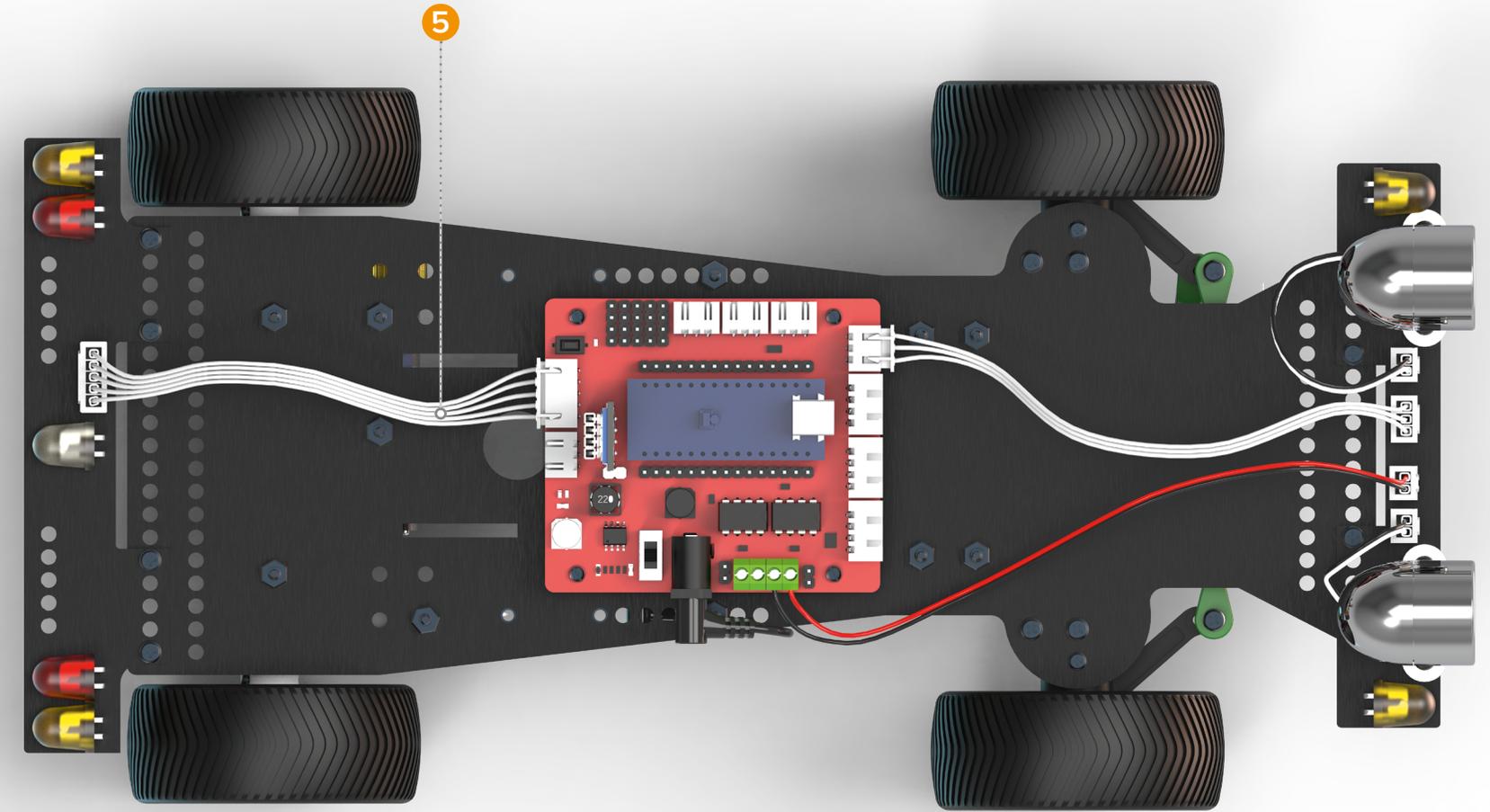
- Поставьте выключатель питания на плате управления в положение OFF.
- Подсоедините кабели №1 (3x14см), №2 (2x17см, красно-черный, соблюдая полярность, при помощи плоской отвертки), №3,4 (чёрно-белые, левая и правая фары).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**front_light_check**» из папки «**LaborotoryProjects**» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления **(ON)**.
- Начнётся автоматическое выполнение программы тестирования блока передних огней. Ожидаемый эффект можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-FrontLight>
- Выключите питание платы управления – разомкните выключатель на плате управления **(OFF)**.



БЛОК ЗАДНИХ ОГНЕЙ

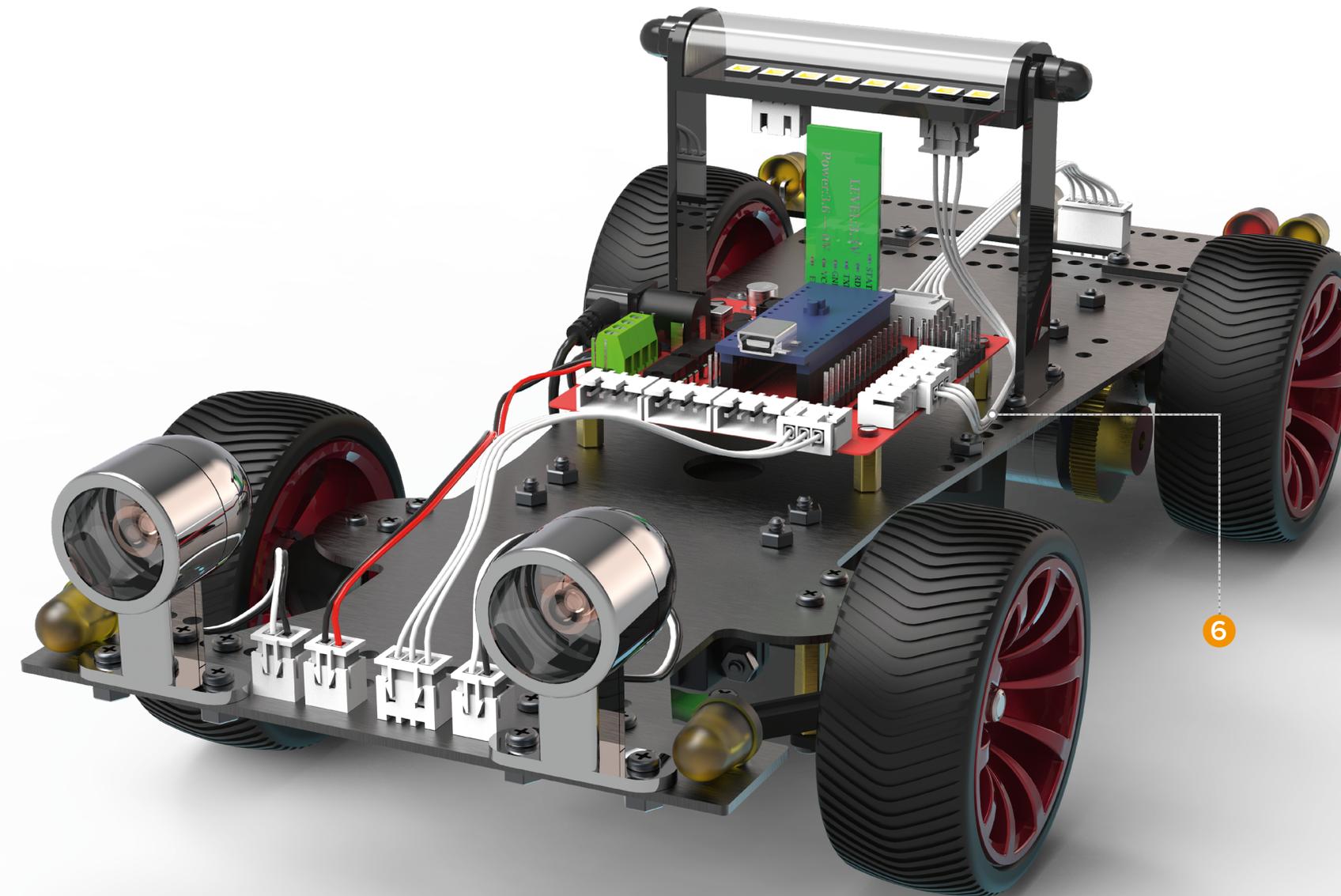
- Подсоедините кабель №5 (5x12cm).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**back_light_check**» из папки «**LaboratoryProjects**» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (**ON**).
- Начнётся автоматическое выполнение программы тестирования блока задних огней. Ожидаемый эффект можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-BackLight>
- Выключите питание платы управления – разомкните выключатель на плате управления (**OFF**).

Для наглядности блок верхних огней («люстра») не показан.



БЛОК ВЕРХНИХ ОГНЕЙ («ЛЮСТРА»)

- Подсоедините кабель №6 (3x10см).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**top_light_check**» из папки «**LaboratoryProjects**» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (**ON**).
- Начнётся автоматическое выполнение программы тестирования блока верхних огней. Ожидаемый эффект можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-TopLight>
- Выключите питание платы управления – разомкните выключатель на плате управления (**OFF**).



СЕРВОПРИВОД

ОБРАЩАЕМ ВНИМАНИЕ, ЧТО «КАЧАЛКА» СЕРВОПРИВОДА ДОЛЖНА БЫТЬ УСТАНОВЛЕНА ТОЧНО ПО ПУНКТУ ИНСТРУКЦИИ, ПРИВЕДЁННОМ В РАЗДЕЛЕ «СБОРКА ШАССИ».

- Подсоедините кабель №7, идущий непосредственно от сервопривода, к соответствующему разъему на плате управления.
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и откройте программу «**servo_check**» из папки «**LaboratoryProjects**» (см. раздел «Установка и загрузка программ»). Одна из важнейших программ! Данная программа позволяет проверить работоспособность сервопривода и откалибровать крайние положения его вала. Научиться это делать совершенно необходимо для управления положением колес, когда сервопривод установлен на платформу и подсоединен к колесам рулевыми тягами. Ниже приведен фрагмент программы, позволяющей установить крайние положения вала сервопривода.

```
const int servo_right_pos = 110;
```

```
const int servo_left_pos = 40;
```

КРАЙНЕЕ ЛЕВОЕ ПОЛОЖЕНИЕ

Выставить значение от 0 до 180 (градусов). Обратите внимание, что для корректной работы данное значение должно быть меньше значения крайне-правого положения.

КРАЙНЕЕ ПРАВОЕ ПОЛОЖЕНИЕ

Выставить значение от 0 до 180 (градусов). Обратите внимание, что для корректной работы данное значение должно быть больше значения крайне-левого положения.

После загрузки программы, отсоедините USB-кабель от модуля Arduino и включите питание – выключатель питания на плате управления в положение **ON**.

При установленных рулевых рычагах необходимо определить крайние правое и левое положение колёс.

Несмотря на то, что сервопривод способен вращаться на 270 градусов, мы его установили так, что он может вращаться на 180, реальный необходимый угол поворота – около 120 градусов – это связано с особенностью крепления колес.

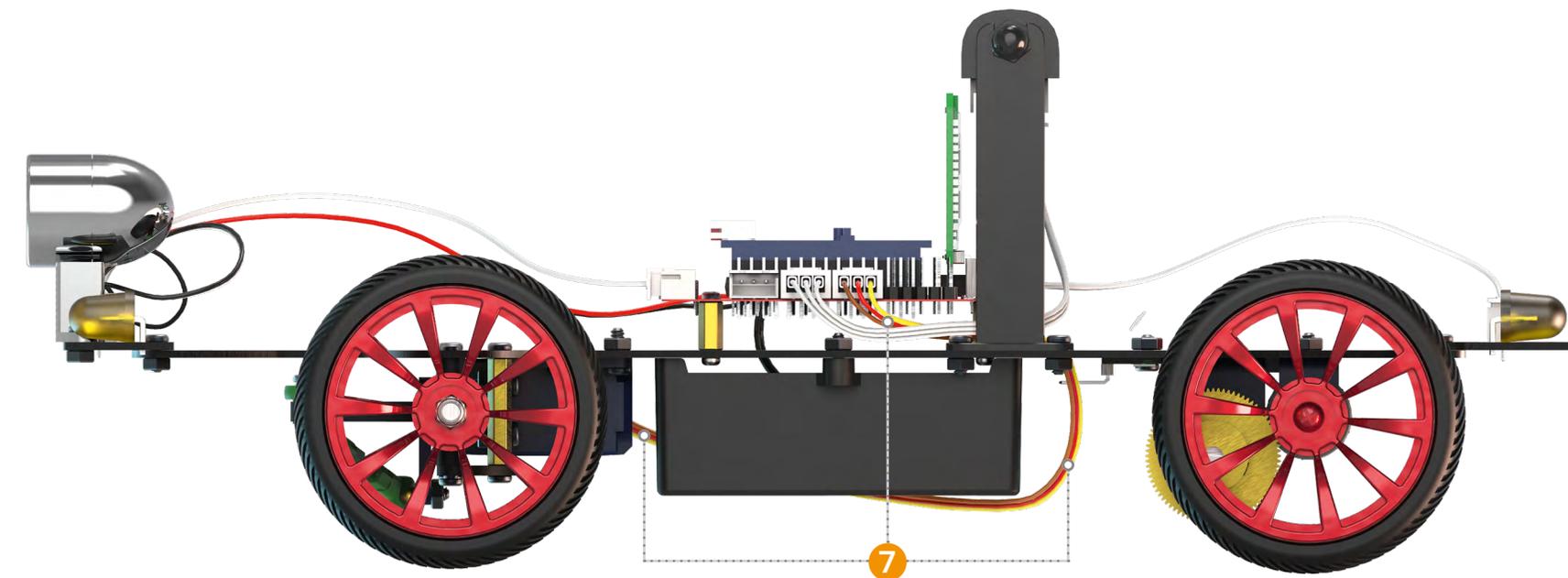
Запустите программу и отметьте, поворачиваются ли колёса до конца в каждом из направлений. Если колеса поворачиваются вправо (влево) не до конца, то необходимо увеличить (уменьшить) соответствующее значение, и наоборот – если колеса начинают упираться в корпус, то данное значение необходимо уменьшить (увеличить).

Рекомендуемый шаг изменения – 10 (например, если сейчас значение – 140, то скорректированное – 130 или 150). Для более точной настройки можно использовать шаг 5.

После каждой корректировки значений, необходимо загружать программу заново.

Значения, полученные в данной настройке понадобятся при дальнейшей работе.

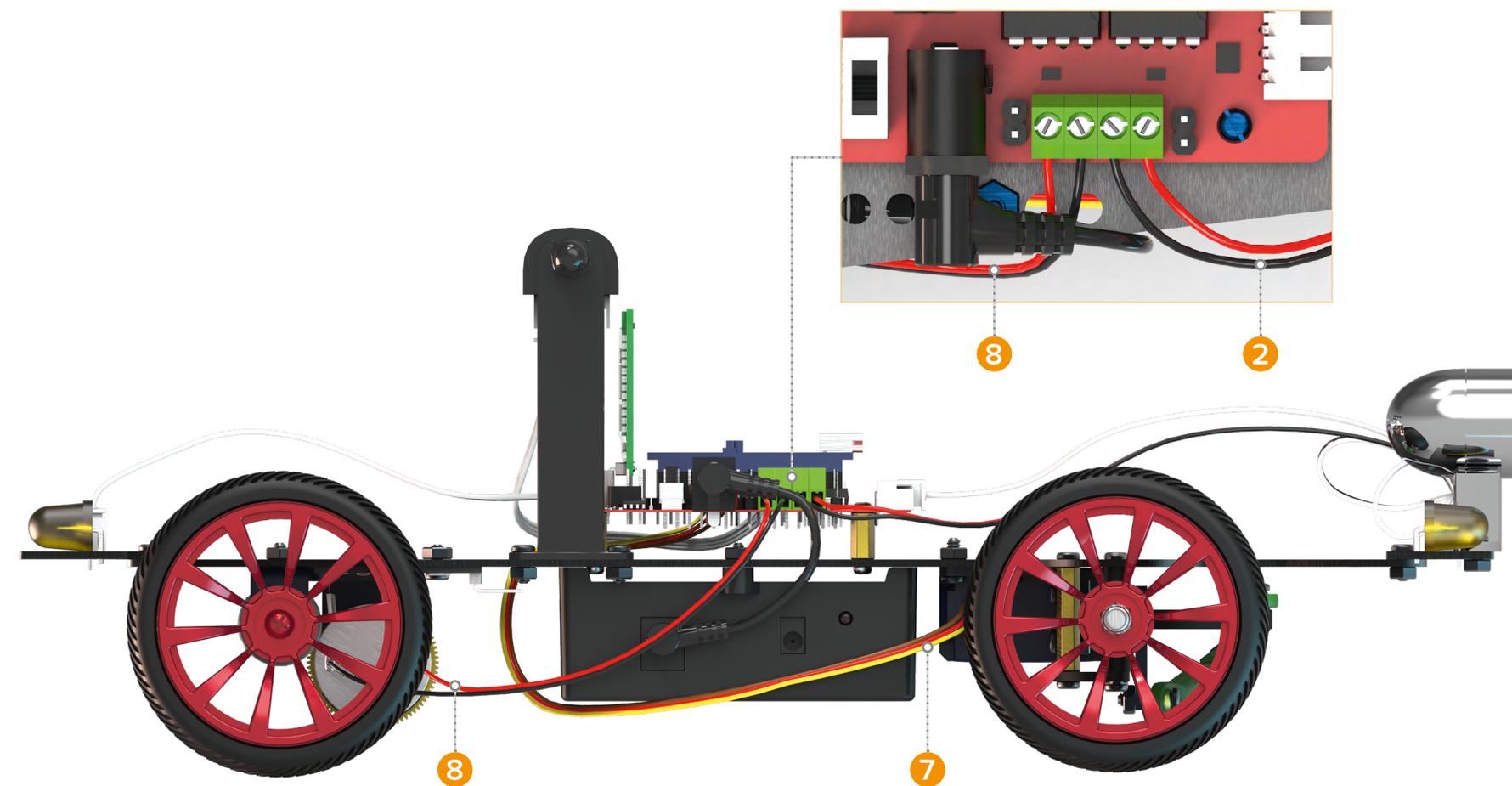
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (**ON**).
- Начнётся автоматическое выполнение программы тестирования сервопривода. Ожидаемый эффект можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-Servo>
- Выключите питание платы управления – разомкните выключатель на плате управления (**OFF**).



МОТОР-РЕДУКТОР

ПРИ ВЫПОЛНЕНИИ ДАННОГО ПУНКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ. 

- Соблюдая полярность, подсоедините при помощи плоской отвертки кабель №8, идущий непосредственно от мотор-редуктора, к соответствующему разъему на плате управления.
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**motor_check**» из папки «**LaborotoryProjects**» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления **(ON)**.
- Начнётся автоматическое выполнение программы тестирования мотор-редуктора. Ожидаемый эффект можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-Motor>
- Выключите питание платы управления – разомкните выключатель на плате управления **(OFF)**.



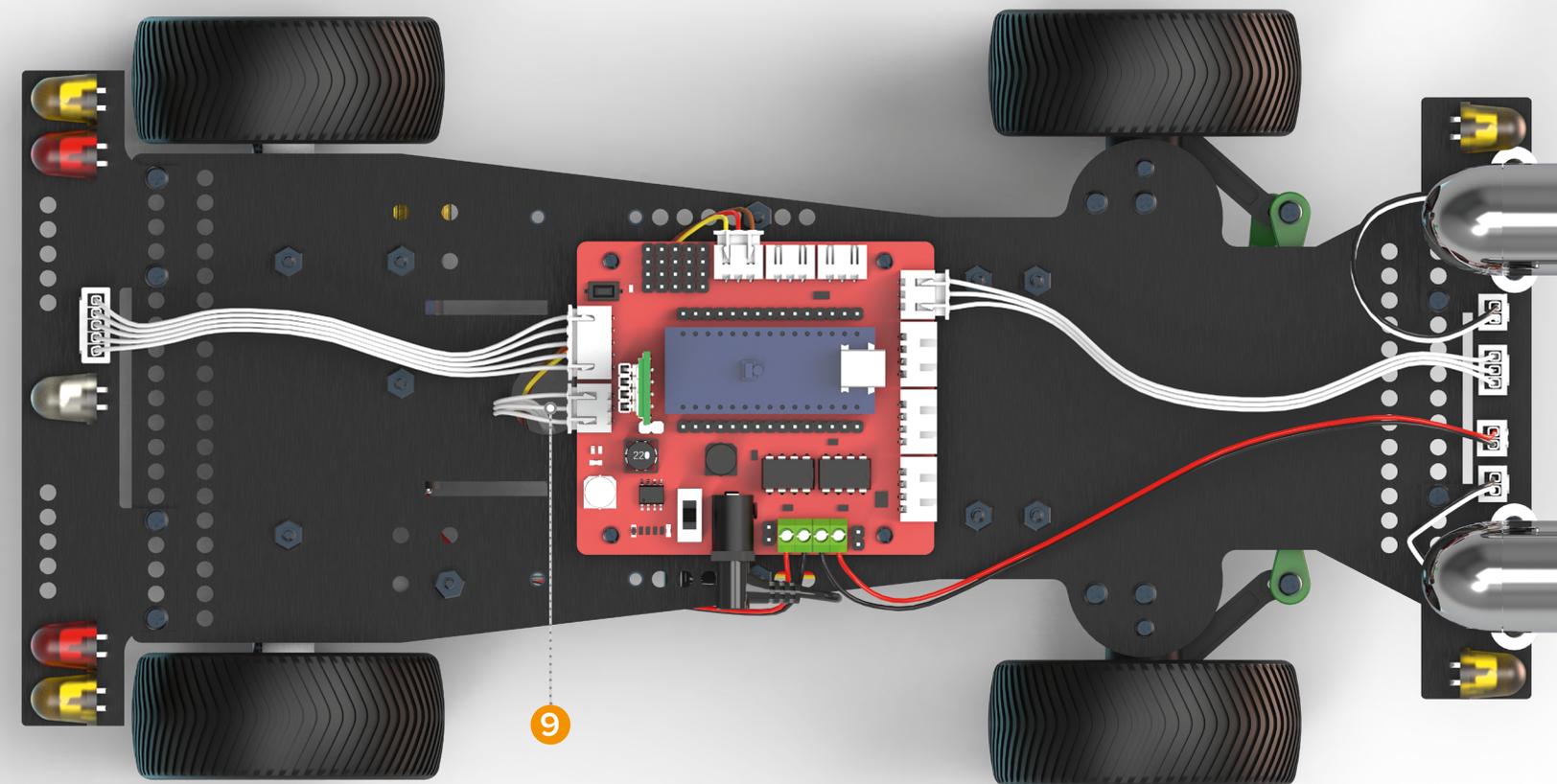
ЭНКОДЕР

ПРИ ВЫПОЛНЕНИИ ДАННОГО ПУНКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ. !

- Подсоедините кабель №9 (3x10см). Датчик Холла расположен рядом с задней осью. Кабель можно продеть через большое отверстие в платформе.
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**hall_sensor_check**» из папки «**LaborotoryProjects**» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (**ON**).
- Начнётся автоматическое выполнение программы тестирования энкодера – частота мигания всех огней будет зависеть от скорости вращения колёс, т.е. частоты прохождения магнитов рядом с датчиком Холла. Ожидаемый эффект можно посмотреть, перейдя по ссылке: <https://znotok.ru/link/?Carduino-Check-Encoder>
- Выключите питание платы управления – разомкните выключатель на плате управления (**OFF**).

Энкодер состоит из двух частей – плата с датчиком Холла и диска с магнитами, закреплённого на задней оси. Кабель №9 соединяет плату управления и плату с датчиком Холла (её на картинке на видно). Расположение платы с датчиком Холла смотрите в разделе РЕГУЛИРОВКА ЭНКОДЕРА.

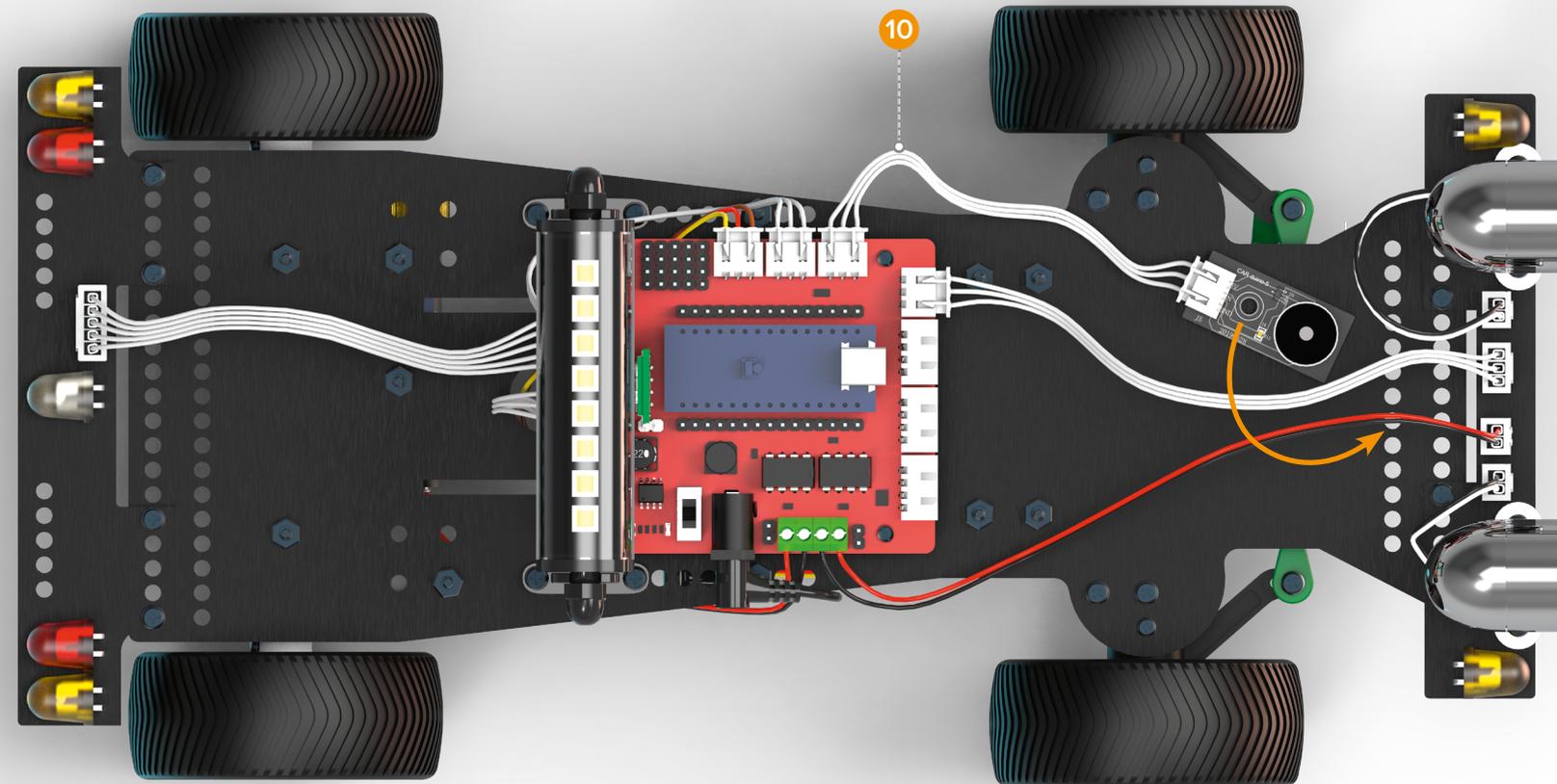
Для наглядности блок верхних огней («люстра») не показан.



ПЬЕЗОИЗЛУЧАТЕЛЬ

- Подсоедините кабель №10 (3x14см). Стрелками указаны возможные места установки пьезоизлучателя при помощи винта и гайки.
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**buzzer_check**» из папки «**LaboratoryProjects**» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (**ON**).
- Начнётся автоматическое выполнение программы тестирования пьезоизлучателя – он должен воспроизвести мелодию.
- Выключите питание платы управления – разомкните выключатель на плате управления (**OFF**).

Пьезоизлучатель может устанавливаться на любое свободное место. Например, крепиться винтом и гайкой к одному из отверстий платформы рядом с блоком передних или задних огней. Стрелкой показано одно из таких мест.



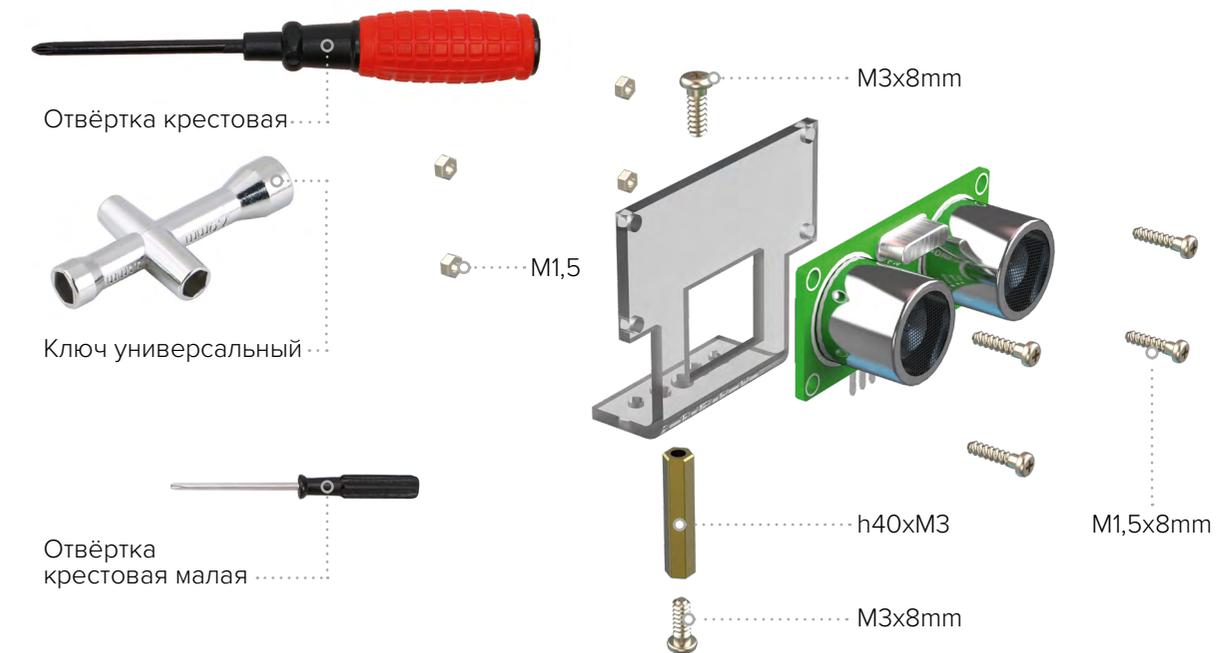
УЛЬТРАЗВУКОВОЙ ДАЛЬНОМЕР.

СБОРКА УЛЬТРАЗВУКОВОГО ДАЛЬНОМЕРА

- Видеоинструкцию по сборке и установке можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Assemble-Ultrasonic>

НА ВИДЕО ПОКАЗАН ТОЛЬКО ОДИН ВАРИАНТ УСТАНОВКИ УЛЬТРАЗВУКОВОГО ДАЛЬНОМЕРА. ВЫ МОЖЕТЕ УСТАНОВИТЬ ЕГО В ДРУГОЕ МЕСТО И НАПРАВИТЬ В НУЖНУЮ ВАМ СТОРОНУ.

- Для сборки ультразвукового дальномера потребуются следующие детали и инструменты:

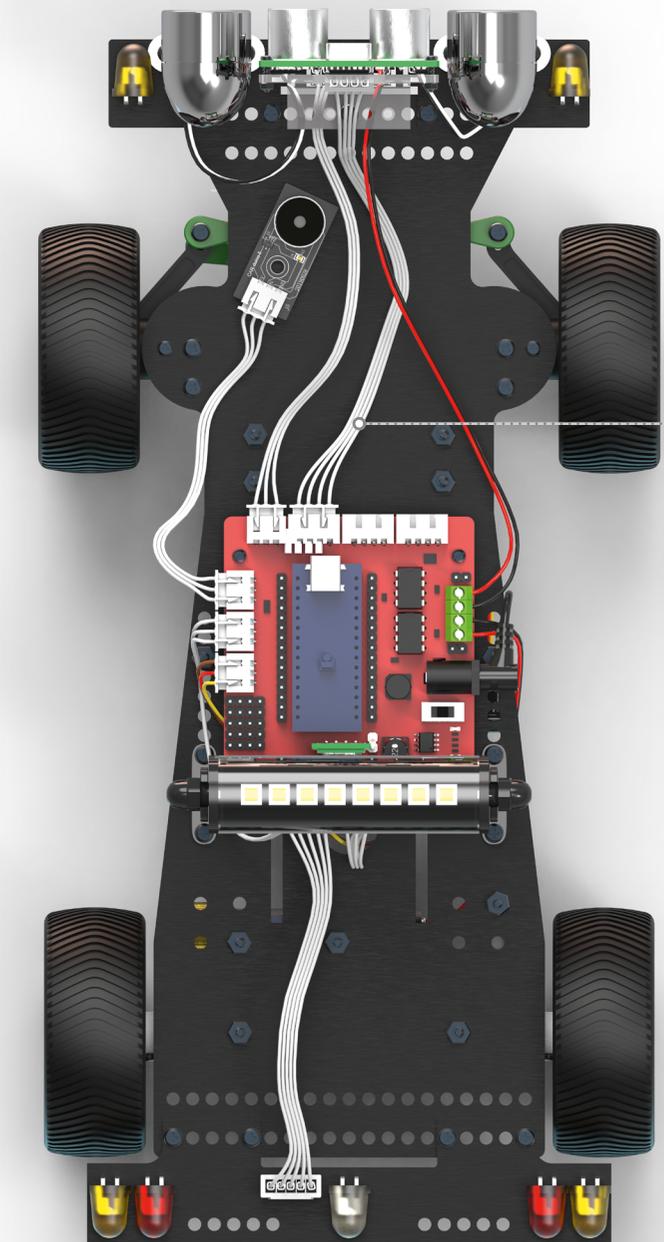


- Конечный результат:



ПОДКЛЮЧЕНИЕ УЛЬТРАЗВУКОВОГО ДАЛЬНОМЕРА

- Подсоедините кабель №11 (4x14cm).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «ultrasonic_check» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Расположите автомобиль на полу и поместите перед ним какой-нибудь плоский предмет. Включите питание – замкните выключатель на плате управления (**ON**).
- Начнётся автоматическое выполнение программы тестирования ультразвукового дальномера – он будет держать определённую дистанцию между автомобилем и препятствием. Ожидаемый эффект можно посмотреть, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-Ultrasonic>
- Выключите питание платы управления – разомкните выключатель на плате управления (**OFF**).

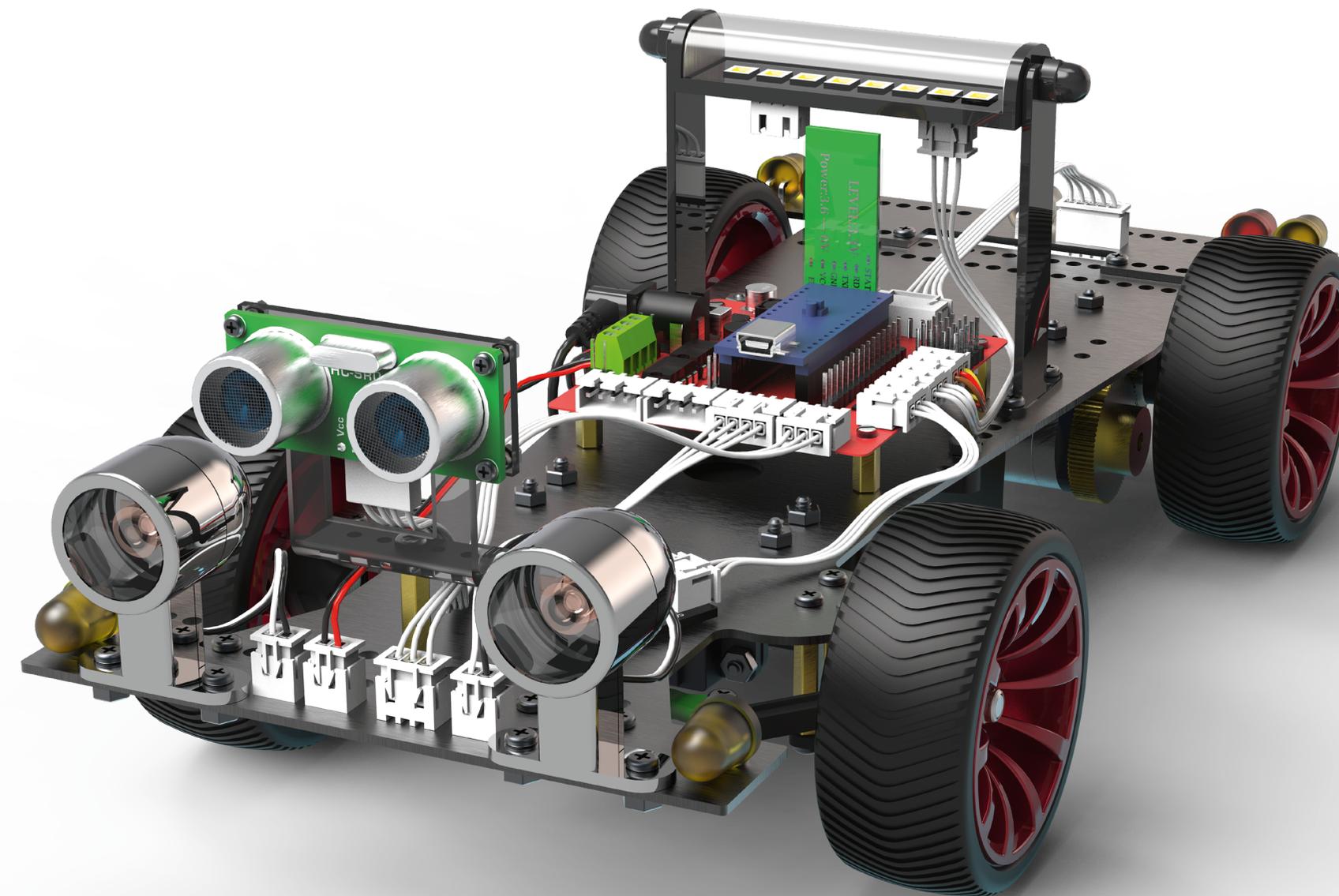


ПОЛНОСТЬЮ СОБРАННАЯ МОДЕЛЬ

Представленная на картинке модель может работать в трёх режимах:

- Езда по заданной программе. Вы можете воспользоваться одной из готовых программ, а можете написать свою.
- Управление от смартфона по Bluetooth.
- Автономный режим, когда автомобиль, получая данные от датчиков, сам принимает решения. Причём, плата управления позволяет подключать дополнительные датчики или подключать дополнительные устройства взамен имеющихся. Например, вместо «люстры» можно подключить ещё один сервопривод.

Вы можете собрать модель с другой компоновкой деталей, сделать дополнительные отверстия для креплений, установить дополнительные элементы, придумать дизайн кузова. Дерзайте!



ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ ПО BLUETOOTH

УСТАНОВКА ПРОГРАММЫ УПРАВЛЕНИЯ НА МОБИЛЬНОЕ УСТРОЙСТВО ANDROID

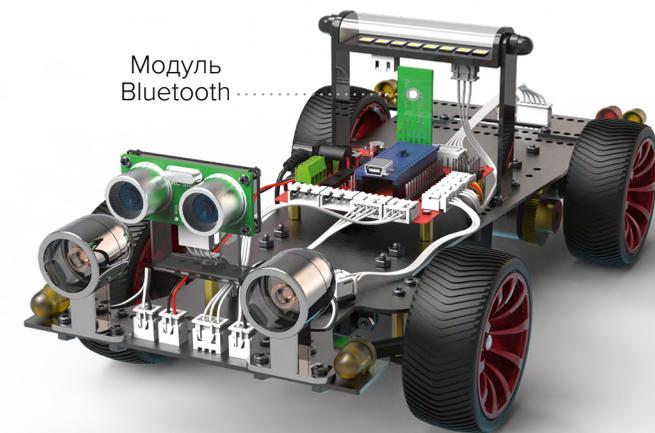
Во многих проектах используется модуль Bluetooth. Убедитесь, что модуль установлен на плате управления. Для его функционирования необходимо установить на своё мобильное устройство (смартфон или планшет) приложение Znatok.Mobile.

Для этого, считайте приведенный здесь QR-код или перейдите на сайт https://znatok.ru/link/?mobile_app и следуйте указанным там инструкциям для установки.



После установки приложения у вас на мобильном устройстве должен появиться такой значок:

После этого необходимо произвести настройку модуля Bluetooth.



НАСТРОЙКА МОДУЛЯ BLUETOOTH ДЛЯ РАБОТЫ

В НЕКОТОРЫХ ВЕРСИЯХ НЕВОЗМОЖНО ИЗМЕНИТЬ ИМЯ МОДУЛЯ И ОН ОСТАНЕТСЯ «MLT-BT05», НО ЗАПУСК ДАННОЙ ПРОГРАММЫ ВСЕ РАВНО НЕОБХОДИМ ДЛЯ КОРРЕКТНОЙ РАБОТЫ.

Соберите полностью модель (в том числе подключите модуль Bluetooth) и включите питание на Плате управления.

Активируйте на вашем мобильном устройстве Bluetooth. При поиске устройств Bluetooth на вашем мобильном устройстве отобразится новое устройство с именем «MLT-BT05» – это имя вашего модуля, стоящего на плате управления.

Подсоедините модуль Arduino на плате управления при помощи USB-кабеля к компьютеру и откройте на нем программу **bluetooth_settings** из папки **LaboratoryProjects** в редакторе Arduino IDE. Найдите в тексте программы этот фрагмент:

```
String uartName = "ZNATOK_CARDUINO";
```

Новое имя модуля Bluetooth

Измените данное имя на новое (сохранив кавычки).

Длина имени может составлять максимум 32 символа и не должна содержать буквы русского алфавита.

Измените указанный фрагмент на новое имя и загрузите программу.

Отключите USB-кабель от компьютера. Переведите положение переключателя питания в положение OFF.

Включите питание на Плате управления, подождите 10-

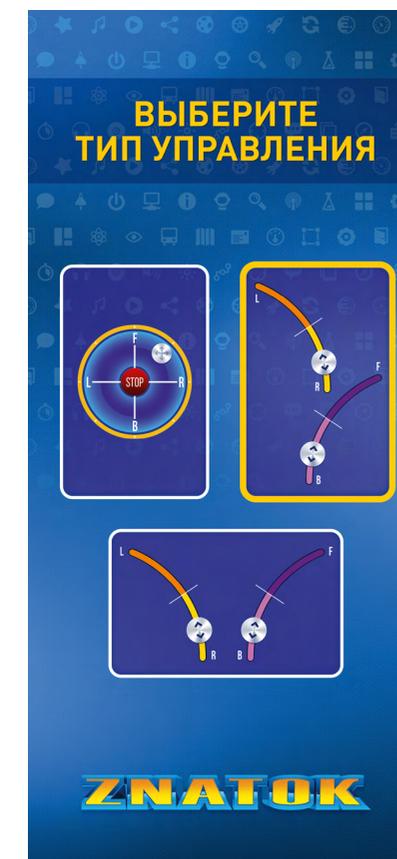
15 секунд и снова проверьте список устройств Bluetooth находящихся рядом при помощи вашего мобильного устройства – там должно появиться новое имя, которое вы присвоили модулю.

Теперь вы можете загружать программы, использующие модуль Bluetooth. После загрузки программы, на мобильном устройстве запустите приложение, дотронувшись до значка:



Далее следуйте инструкциям, появляющимся на вашем смартфоне или планшете, с учётом заданного вами нового имени модуля Bluetooth.

Выберите тип управления:



БЛОК ПЕРЕДНИХ ОГНЕЙ



ПРОЕКТ «ДЕМОНСТРАЦИЯ»

Данную программу вы использовали в разделе «Подключение и тестирование» (стр. 34). Перед переходом к другим проектам, рекомендуем убедиться, что блок передних огней по-прежнему работает.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу **«front_lights_check»** из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы тестирования блока передних огней: поворотники с фарами начнут включаться и выключаться в цикле, при этом фары гаснуть будут плавно.
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?CarDuino-Check-FrontLight>

РАЗБОР ПРОГРАММЫ

В программировании слова «функция» и «команда» имеют практически одинаковый смысл, но в зависимости от контекста мы будем использовать либо слово «функция», либо «команда».

- Каждая программа содержит 2 обязательных блока – **setup** и **loop** (см. рисунок) и дополнительные блоки с данными, константами и пр. Все команды внутри **setup** выполняются один раз при подаче питания на модуль Arduino, далее, в бесконечном цикле, выполняются команды из блока **loop**.
- В программах на языке C++ начало и конец блока/функции обозначаются при помощи открывающейся и закрывающейся фигурных скобок **{** и **}**.
- Каждая программа для Arduino состоит из набора функций (команд), которые выполняются последовательно, друг за другом.
- Команды (почти все) отделяются друг от друга знаком **;**, а для того, чтобы нам было удобно их читать и изменять, команды пишутся на отдельных строках.
- Программист может оставлять комментарии в программе. Они не воспринимаются как команды для Arduino и предназначены, в основном, для того, чтобы кратко,

“на человеческом” языке, пояснить зачем нужен тот или иной кусок программы.

- Один из вариантов комментариев приведен на рисунке – на любой строчке всё, что идет после двойного слэша **//**, будет являться комментарием, и Arduino не будет переводить это на язык машинных команд.

ПРИМЕР ТИПОВОЙ ПРОГРАММЫ

```
front_lights_check
1 #define LAMP_PIN 3 //Фары
2 #define LIGHT_LEFT_PIN 9 //Левый поворотник
3 #define LIGHT_RIGHT_PIN 7 //Правый поворотник
4
5 void setup()
6 {
7     //Настройка всех выводов на "Выход"
8     pinMode(LIGHT_LEFT_PIN, OUTPUT);
9     pinMode(LIGHT_RIGHT_PIN, OUTPUT);
10    pinMode(LAMP_PIN, OUTPUT);
11 }
12
13 void loop()
14 {
15     digitalWrite(LIGHT_RIGHT_PIN, HIGH); //Включить правый поворотник
16     digitalWrite(LIGHT_LEFT_PIN, HIGH); //Включить левый поворотник
17     digitalWrite(LAMP_PIN, HIGH); //Включить фары
18     delay(1000); //Подождать 1 секунду
19
20     digitalWrite(LIGHT_RIGHT_PIN, LOW); //Выключить правый поворотник
21     digitalWrite(LIGHT_LEFT_PIN, LOW); //Выключить левый поворотник
22     for (int i = 255; i > 0; i--) //Плавно уменьшать яркость фар
23     {
24         analogWrite(LAMP_PIN, i);
25         delay(10);
26     }
27 }
```

Константы

setup

loop

Комментарии

БЛОК ПЕРЕДНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ОДНОВРЕМЕННОЕ МИГАНИЕ ПОВОРОТНИКАМИ»

В данном проекте показывается, как можно включать и выключать поворотники, а так же выставлять время, на которое они будут включаться.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**front_lights_01**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: поворотники начнут циклически включаться на 1 секунду, и выключаться на 1 секунду.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-FrontLights-1>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численные значения в программе можно регулировать частоту включения светодиодов-поворотников. Ниже приведен кусок программы и выделены параметры, изменяющие режим работы.

```
digitalWrite(9, HIGH);
digitalWrite(7, HIGH);
delay(1000);
digitalWrite(9, LOW);
digitalWrite(7, LOW);
delay(1000);
```

ЧАСТОТА ВКЛЮЧЕНИЯ И ВЫКЛЮЧЕНИЯ ПОВОРОТНИКОВ

Уменьшите оба указанных значения до 500 и убедитесь в том, что поворотники стали моргать в 2 раза чаще.

РАЗБОР ПРОГРАММЫ

- В данной программе главными являются 2 функции с именами: **digitalWrite** и **delay**. Каждая из них отвечает за то или иное действие.
- В круглых скобках после имени функций идут **параметры функций**: числа или значения, в зависимости от которых функция может работать по-разному.
- Функция **digitalWrite** позволяет подавать или убирать напряжение с **пина** Arduino (порта входа или выхода), в нашем случае – позволяет включать или выключать светодиоды-поворотники.
- Каждый пин Arduino имеет свой номер, в CAR-duino передние поворотники подключены к пинам с номерами **9** и **7**.
- Параметры функции **digitalWrite** позволяют выбрать нужный вывод и подать на него высокий **HIGH** или низкий **LOW** уровень напряжения. Например, функция **digitalWrite(9, HIGH)** подаёт на вывод 9 высокий уровень, который включает светодиод, подключенный к нему.
- Выполнение каждой функции (команды) занимает очень маленькое время. И если после команды включения светодиода, мы напишем команду, которая его выключает, то не заметим, что светодиод горел. Чтобы избежать этого, используется функция **delay** (задержка).
- Функция **delay** позволяет задать время в течение которого Arduino приостанавливает свою работу, но не выключается. Время задается в миллисекундах (в 1 секунде – 1000 миллисекунд). Например, в нашей программе мы включаем 2 светодиода-поворотника и выполняем функцию **delay(1000)**, т.е. в течение 1 секунды ничего не будет происходить, светодиоды будут гореть, и только потом продолжат выполняться команды ниже.

БЛОК ПЕРЕДНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ПОПЕРЕМЕННОЕ МИГАНИЕ ПОВОРОТНИКАМИ»

В данном проекте показывается как можно независимо включать и выключать поворотники и регулировать частоту их включения.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «front_lights_02» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: поворотники начнут включаться и выключаться по очереди с периодичностью в 1 секунду.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-FrontLights-2>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численные значения в программе можно регулировать частоту включения светодиодов-поворотников. Ниже приведен кусок программы и выделены параметры, изменяющие режим работы.

```
digitalWrite(LIGHT_LEFT_PIN, HIGH);
digitalWrite(LIGHT_RIGHT_PIN, LOW);
delay(1000);
digitalWrite(LIGHT_LEFT_PIN, LOW);
digitalWrite(LIGHT_RIGHT_PIN, HIGH);
delay(1000);
```

ЧАСТОТА ВКЛЮЧЕНИЯ И ВЫКЛЮЧЕНИЯ ПОВОРОТНИКОВ

Уменьшите оба указанных значения до 500 и убедитесь в том, что поворотники стали моргать в 2 раза чаще.

РАЗБОР ПРОГРАММЫ

- Часто, проекты бывают большими и не всегда можно запомнить какой пин к какому устройству подключен, а в процессе работы над программой этот пин может поменяться. Чтобы решить эту проблему придумали т.н.

константы:

```
#define LIGHT_LEFT_PIN 9 //Левый поворотник
#define LIGHT_RIGHT_PIN 7 //Правый поворотник
```

- В фрагменте программы выше создали 2 константы и присвоили им значения: создали константу **LIGHT_LEFT_PIN** и присвоили ей значение **9**, аналогично создали и вторую. Теперь, если в тексте программы написать **LIGHT_LEFT_PIN**, система автоматически заменит это на **9**, а **LIGHT_RIGHT_PIN** на **8**.

- Для того, чтобы создать константу, необходимо на новой строке (удобнее делать это перед блоками **setup** и **loop**) написать:

```
#define ИМЯ_КОНСТАНТЫ ЗНАЧЕНИЕ
```

где ИМЯ_КОНСТАНТЫ – это придуманное программистом слово-имя (из латинских букв), а значение – как правило какое-то число.

- Хорошим тоном является выбор “человеческого” имени, при этом состоящих только из больших букв – по имени можно понять что делает константа в программе (в данной программе **LIGHT_LEFT_PIN** означает “пин к которому подключен передний левый свет”), а большие буквы помогут отличить её от имён функций и пр.
- В блоке **setup** встречается функция **pinMode** – это обязательная функция, которая позволяет настроить работу пинов.

pinMode(LIGHT_LEFT_PIN, OUTPUT) сообщает Arduino, что пин с номером **LIGHT_LEFT_PIN** (константа со значением 9) работает в режиме **OUTPUT** – на выход: на нем можно выставлять высокий и низкий уровни напряжения. Пока будут использоваться пины только с таким режимом, но в дальнейшем мы познакомимся и с другими.

БЛОК ПЕРЕДНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «МИГАНИЕ ПЕРЕДНИХ ФАР»

В данном проекте показывается как можно независимо включать и выключать передние фары и поворотники, а также регулировать частоту их включения.

Несмотря на то, что по сути в данном проекте выполняется такое же действие как и в предыдущих (мигание светодиодом) – в программе присутствуют новшества, с которыми вы можете ознакомиться ниже.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «front_lights_03» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: фары начнут моргать с частотой 2 раза в секунду.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-FrontLights-3>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численные значения в программе можно регулировать частоту включения светодиодов-поворотников. Ниже приведен кусок программы и выделены параметры, изменяющие режим работы.

```
analogWrite(LAMP_PIN, 255);
delay(500);

analogWrite(LAMP_PIN, 0);
delay(500);
```

ЧАСТОТА ВКЛЮЧЕНИЯ И ВЫКЛЮЧЕНИЯ ПОВОРОТНИКОВ

Увеличьте оба данных значения до 1000 и убедитесь в том, что фары начали моргать в 2 раза реже.

РАЗБОР ПРОГРАММЫ

- В CAR-duino существует возможность зажигать фары с разной яркостью. Это можно сделать при помощи функции **analogWrite**. Как и в случае с функцией **digitalWrite**, первый параметр отвечает за номер пина которым мы хотим управлять, а второй – это число показывающее с какой мощностью мы хотим включить фару
- Второй параметр – это целое число от 0 до 255. При 0 – фара полностью выключается, при 255 – включается с максимальной яркостью.
- Функция **analogWrite** на самом деле использует ШИМ для регулировки мощности. Для более подробной информации о ШИМ – см. страницу 55

ЗАДАНИЕ

«ВКЛЮЧЕНИЕ И ВЫКЛЮЧЕНИЕ СВЕТОДИОДОВ»

Изучите программы из проектов выше и напишите собственную программу, которая последовательно будет зажигать и гасить в бесконечном цикле: левый поворотник, фары, правый поворотник.

Для того чтобы посмотреть, что должно получиться в итоге, вы можете посмотреть видео, перейдя по ссылке: <https://znatok.ru/link/?CarDuino-Task-1>

БЛОК ПЕРЕДНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «СТУПЕНЧАТАЯ ЯРКОСТЬ ФАР»

В данном проекте показывается как можно регулировать яркость фар так же как это делается в реальных автомобилях: ближний свет, дальний свет, габариты.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «front_lights_04» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы. Фары в цикле начнут менять режимы работы: Дальний свет – ближний свет – габариты – выключены.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-FrontLights-4>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численные значения в программе можно регулировать частоту переключения режимов работы фар. Ниже приведен кусок программы и выделены параметры, изменяющие режим работы.

```
int switchDelay;
switchDelay = 1000;
```

ЧАСТОТА ВКЛЮЧЕНИЯ И ВЫКЛЮЧЕНИЯ ПОВОРОТНИКОВ

Уменьшите данный параметр до 500 и убедитесь в том, что режимы стали сменяться в 2 раза чаще.

РАЗБОР ПРОГРАММЫ

- Кроме констант, в программах можно создавать **переменные**, которые в отличие от констант в процессе работы программы могут менять своё значение (хотя именно в этой программе переменная не меняет своё значение, но эта её возможность пригодится нам в дальнейшем)
- **Переменная** – это ячейка в памяти компьютера/микроконтроллера/Arduino, которая хранит в себе какое-то значение. Обычно, в таких ячейках хранятся числа.
- Чтобы создать переменную, в программе пишется строчка:

```
int switchDelay;
```

При помощи служебного слова **int** создается переменная-ячейка, в которой будет храниться целое число (англ. integer – “целое”), а через пробел указано имя, которое придумано программистом для обозначения этой ячейки.

- Как и в случае с константами, хорошим тоном является использование имен, которые несут в себе смысл.

Здесь, “**switchDelay**” в переводе обозначает “задержка переключения”, т.е. время, которое будет проходить между переключением режимов работы фары.

- С переменными можно делать много разных операций, но самое простое что можно сделать – это задать значение, записать какое-то значение в ячейку.

```
switchDelay = 1000;
```

В данном примере, мы в ячейку с именем **switchDelay** записали число **1000**. Обратите внимание, что знак **=** – это т.н. операция “присваивания”, а не математическое “равно”, к которому многие привыкли. О математических операциях мы поговорим позднее.

- **Переменные**, как и **константы**, можно использовать в **функциях**. Например, во всех трёх использованиях **delay(switchDelay)** в данной программе, используется значение **switchDelay** как параметра, а он задается только один раз в самом начале программы.

БЛОК ПЕРЕДНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ПЛАВНОЕ УВЕЛИЧЕНИЕ ЯРКОСТИ ФАР»

В данном проекте показывается как можно плавно увеличивать яркость фар.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «front_lights_05» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле фары будут плавно увеличивать яркость до максимума, а потом резко выключаться.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-FrontLights-5>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать пороговую яркость фар. Ниже приведен кусок программы и выделен параметр, изменяющий режим работы.

```
while (curBrightness < 255)
```

ПОРОГОВОЕ ЗНАЧЕНИЕ ЯРКОСТИ

Уменьшите данный параметр до 100 и убедитесь в том, что максимальная яркость у фары сильно уменьшилась.

РАЗБОР ПРОГРАММЫ

- Часто, в программах требуется много раз выполнять похожие действия с небольшими различиями ПОКА выполняется какое-то условие. Для этого используется цикл while (англ. while – “пока”):

```
while (curBrightness < 255)
{
  analogWrite(LAMP_PIN, curBrightness);
  delay(15);

  curBrightness++;
}
```

- Команды-функции внутри фигурных скобок выполняются ПОКА выполняется УСЛОВИЕ внутри круглых скобок, в данном случае “ПОКА значение переменной curBrightness меньше 255, последовательно выполняются 3 команды внутри скобок” Условие проверяется каждый виток цикла и как только оно будет неверным – ложным (значение curBrightness станет равным 255),

программа выйдет из этого цикла – начнется выполнение дальше.

- **Условия** могут быть разными, в основном мы будем работать с математическими условиями (< – меньше, > – больше, == – равно, не путать с присваиванием =, != – не равно, >= – больше или равно, <= – меньше или равно)
- Еще одна операция с численными переменными – уменьшение или увеличение значения. Операция **curBrightness++** увеличивает значение переменной на 1.
- Чтобы понять что делает программа, полезно прочитать её по строкам проговаривая что делает каждая функция. Например, данный фрагмент: “Пока значение переменной меньше 255, мы будем выполнять следующие действия: выставим яркость фар равной значению переменной, подождем небольшое время и увеличим значение переменной на 1, после чего проверим – переменная все еще меньше 1? И так по кругу”
- Важно понимать, что для использования переменной, она обязательно должна быть создана, а в нашей программе она создается каждый новый виток блока **loop** и её значение обнуляется в самом начале.

БЛОК ПЕРЕДНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ПЛАВНОЕ УВЕЛИЧЕНИЕ И УМЕНЬШЕНИЕ ЯРКОСТИ ФАР»

В данном проекте показывается как можно плавно увеличивать яркость фар.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «front_lights_06» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле фары будут плавно увеличивать яркость до максимума, а потом плавно уменьшать до 0
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?CarDuino-FrontLights-6>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать пороговую яркость фар. Ниже приведен кусок программы и выделены параметры, изменяющие режим работы.

```
for (int i = 0; i < 255; i++)
{
  analogWrite(LAMP_PIN, i);
  delay(15);
}

for (int i = 255; i > 0; i--)
{
  analogWrite(LAMP_PIN, i);
  delay(15);
}
```

СКОРОСТЬ ИЗМЕНЕНИЯ ЯРКОСТИ

Увеличьте данный параметр до 30 и убедитесь в том, что яркость стала изменяться намного медленнее

РАЗБОР ПРОГРАММЫ

- Еще один тип цикла – цикл **for**. Работает он схожим образом, но записывается гораздо короче. Ниже представлено сопоставление между циклом **while** и **for** (циклы выполняют одну и ту же операцию):

```
int i = 0;
while (i < 255)
{
  analogWrite(LAMP_PIN, i);
  delay(15);

  i++;
}

for (int i = 0; i < 255; i++)
{
  analogWrite(LAMP_PIN, i);
  delay(15);
}
```

Операция, выполняемая до старта цикла

Условие цикла

Тело цикла

Операция, выполняемая в конце каждого витка.

- Стоит обратить внимание, что переменные существуют только внутри тех фигурных скобок, в которых они были созданы. Так, в данной программе переменная **i** создается 2 раза, потому что каждый раз она существует только внутри соответствующего цикла **for**

ЗАДАНИЕ «РАБОТА С ПЕРЕДНИМ БЛОКОМ ОГНЕЙ»

Изучите программы из проектов выше и напишите собственную программу, которая будет в бесконечном цикле выполнять следующие операции:

1. Включить поворотники и фары с максимальной яркостью

2. Выключить поворотники

3. Плавно опустить яркость фар до половины

Для того чтобы посмотреть, что должно получиться в итоге, вы можете посмотреть видео, перейдя по ссылке: <https://znatok.ru/link/?CarDuino-Task-2>

БЛОК ЗАДНИХ ОГНЕЙ

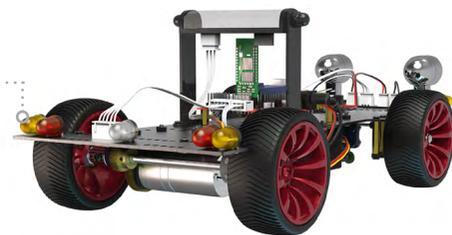
ПРОЕКТ «ДЕМОНСТРАЦИЯ»

Данную программу вы использовали в разделе «Подключение и тестирование» (стр. 36). Перед переходом к другим проектам, рекомендуем убедиться, что блок задних огней по-прежнему работает.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу **«back_lights_check»** из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).

- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы тестирования блока задних огней: все огни на заднем блоке будут в бесконечном цикле зажигаться и гаснуть
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-BackLight>



ПРОЕКТ «ОДНОВРЕМЕННОЕ МИГАНИЕ ПОВОРОТНИКАМИ»

В данном проекте показывается как можно включать и выключать поворотники, а так же выставлять время на которое они будут включаться. Обратите внимания, что вместе с задними поворотниками включаются и передние – это происходит потому, что они подключены параллельно.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу **«back_lights_01»** из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле поворотники начнут 5 раз одновременно включаться и выключаться с периодичностью в 1 секунду, а потом 5 раз с периодичностью в 0.3 секунды.

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-BackLights-1>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численные значения в программе, можно регулировать число миганий поворотниками. Ниже приведен кусок программы и выделен параметр, изменяющий режим работы.

```
int count = 5;
```

ЧИСЛО ВКЛЮЧЕНИЙ И ВЫКЛЮЧЕНИЙ ПОВОРОТНИКОВ

Уменьшите данный параметр до 2 и убедитесь в том, что теперь поворотники будут включаться по 2 раза.

БЛОК ЗАДНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «МИГАНИЕ СТОП-СИГНАЛОВ»

В данном проекте показывается как можно включать и выключать стоп-сигналы и выставлять частоту их моргания.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**back_lights_02**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле стоп-сигналы начнут часто моргать.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?CarDuino-BackLights-2>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численные значения в программе можно регулировать число миганий поворотниками. Ниже приведен кусок программы и выделены параметры, изменяющие режим работы.

```
digitalWrite(LIGHT_TAIL_STOP_PIN, HIGH);
delay(200);
digitalWrite(LIGHT_TAIL_STOP_PIN, LOW);
delay(200);
```

ЧАСТОТА МОРГАНИЯ СТОП-СИГНАЛОВ

Увеличьте данное значение до 400 и убедитесь в том, что стоп-сигналы стали моргать в 2 раза реже.

ПРОЕКТ «ПОПЕРЕМЕННОЕ МИГАНИЕ СИГНАЛАМИ ЗАДНЕГО БЛОКА ФАР»

В данном проекте показывается как можно попеременно зажигать и гасить все сигналы на заднем блоке фар.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**back_lights_03**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле сигналы заднего блока будут поочередно включаться и выключаться.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?CarDuino-BackLights-3>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать частоту переключения огней. Ниже приведен

кусок программы и выделены параметры, изменяющие режим работу.

```
int switchDelay = 1000;
```

ЧАСТОТА ПЕРЕКЛЮЧЕНИЯ

Увеличьте данный параметр до 2000 и убедитесь в том, что огни стали переключаться в 2 раза реже.

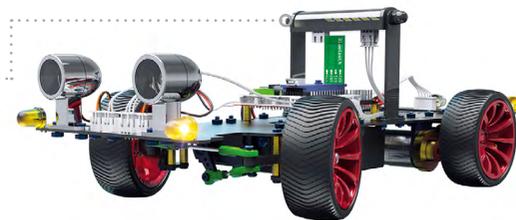
РАЗБОР ПРОГРАММЫ

- Обратите внимание, что в блоке **setup** принудительно выключаются все светодиоды. Несмотря на то, что они и так выключены по умолчанию, при работе, иногда полезно это писать явно, особенно если в дальнейшем какие-либо из светодиодов в начале работы программы должны быть включены.

ЗАДАНИЕ «СВЕТОВАЯ СИГНАЛИЗАЦИЯ»

Изучите программы из двух глав про задний и передний блоки огней и напишите программу, которая позволяет в бесконечном цикле включать и выключать одновременно все огни на переднем и заднем блоках огней каждые 0.5 секунды. Дополните программу тем, чтобы при запуске все огни горели 3 секунды и только потому начали мигание. Для того чтобы посмотреть, что должно получиться в итоге, вы можете посмотреть видео, перейдя по ссылке: <https://znatok.ru/link/?CarDuino-Task-3>

БЛОК ВЕРХНИХ ОГНЕЙ («ЛЮСТРА»)



ПРОЕКТ «ДЕМОНСТРАЦИЯ»

Данную программу вы использовали в разделе «Подключение и тестирование» (стр. 38). Перед переходом к другим проектам, рекомендуем убедиться, что блок верхних огней по-прежнему работает.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**top_light_check**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).

- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы тестирования блока задних огней: все огни на люстре будут переливаться радугой.
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-TopLight>

ПРОЕКТ «КАЛИБРОВКА ЯРКОСТИ»

В данном проекте показывается как можно попеременно зажигать и гасить все сигналы на верхнем блоке фар.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**top_lights_01**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле все огни люстры будут включаться и выключаться белым светом
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать максимальную яркость люстры. Ниже приве-

ден кусок программы и выделен параметр, изменяющий режим работы.

```
#define MAX_BRIGHTNESS 20
```

ЯРКОСТЬ ЛЮСТРЫ

Увеличьте данный параметр до 100 и убедитесь в том, что люстра стала гореть намного ярче. **Данный параметр может принимать значение от 0 до 255**

ВНИМАНИЕ! Данный параметр присутствует в каждом проекте с использованием блока верхних огней – изменяйте его внимательно, т. к. «люстра» может гореть очень ярко берегите глаза.

РАЗБОР ПРОГРАММЫ

- В программе появилось много нового по сравнению с предыдущими проектами. Постепенно мы познакомимся со всем.
- До данного момента, использовались сравнительно простые функции, которые выполняли простые команды: «подать напряжение на пин», «подождать секунду»,

БЛОК ВЕРХНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «КАЛИБРОВКА ЯРКОСТИ» ПРОДОЛЖЕНИЕ

«настроить пин на вывод». Адресные светодиоды (из которых состоит блок верхних огней), сервоприводы, и т. д. – более сложные устройства и для их работы были созданы отдельные функции.

- Подробнее об управлении адресными светодиодами – см. «Это интересно. Адресные светодиоды»
- Множество функций, предназначенных для работы с чем-то одним, объединяют в библиотеки. Для управлением люстрой, состоящей из адресных светодиодов, используется сторонняя библиотека с именем FastLED.
- Подключение библиотеки выполняется при помощи строчки:

```
#define MAX_BRIGHTNESS 20
```

- Библиотеки бывают разных типов, некоторые из них идут в комплекте с ArduinoIDE, а в данном случае – библиотека расположена в папке со скетчем – вы можете рядом с файлом проекта увидеть папку **src**
- Для изменения максимальной яркости используется функция **FastLED.setBrightness**. Параметр может принимать значение от 0 до 255 (максимальная яркость). Не пугайтесь точки в середине. Мы разберемся, откуда

да она берется в этой функции, позднее.

- Один из способов задать цвет того или иного светодиода – использовать конструкцию **strip[X] = CRGB::White; strip** – это наша люстра. Число X в квадратных скобках – это номер светодиода в ленте, цвет которого мы хотим задать. В нашем случае, в люстре 8 светодиодов, которые имеют порядковые номера от 0 до 7. **CRGB::White** – белый цвет
- Цвет светодиода может быть стандартным: **CRGB::White** – белый, **CRGB::Red** – красный, **CRGB::Green** – Зеленый, **CRGB::Blue** – синий, **CRGB::Black** – отсутствие света. В дальнейших проектах мы узнаем о том как задавать свои цвета.
- Список стандартных цветов и их имён можно увидеть перейдя по ссылке <https://github.com/FastLED/FastLED/wiki/Pixel-reference#predefined-colors-list>, реальный цвет люстры может немного отличаться от того, что вы увидите на мониторе.
- После того как мы выберем цвета светодиодов, необходимо передать эту информацию на люстру – для этого выполняется функция **FastLED.show()**.
- Обратите внимание что у этой функции нет параметров, но мы все равно пишем круглые скобки с пустым содержимым.

ПРОЕКТ «ПОПЕРЕМЕННАЯ СМЕНА ЦВЕТА ВСЕЙ ЛЮСТРЫ»

В данном проекте показывается как можно менять цвет всей люстры.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**top_lights_02**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле цвет люстры будет меняться – красный, зеленый, синий
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-TopLights-2>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регу-

лировать частоту переключения цветов. Ниже приведен кусок программы и выделены параметры, изменяющие режим работу.

```
int switchDelay = 1000;
```

ЧАСТОТА ПЕРЕКЛЮЧЕНИЯ ЦВЕТОВ

Уменьшите данный параметр до 500 и убедитесь в том, что цвет люстры стал меняться в 2 раза чаще.

РАЗБОР ПРОГРАММЫ

- Нередко, возникает необходимость создать много переменных со схожим смыслом. Например, нам надо записать информацию о температуре за каждый день года в память Arduino. Мы могли бы вручную создать 356(366) переменных с именами, вроде, day1, day2, day3, day4..., day365, но, обязательно, где-нибудь ошиблись бы. Решением данной проблемы является использование **массивов**.
- Простыми словами, **массив** – это много переменных одного типа с одним именем, но разными порядковыми номерами. Создать массив можно следующим образом:

```
int dayTemperatures[365];
```

БЛОК ВЕРХНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ПОПЕРЕМЕННАЯ СМЕНА ЦВЕТА ВСЕЙ ЛЮСТРЫ» ПРОДОЛЖЕНИЕ

Здесь, создали массив из 365 переменных типа **int** (целые числа). К каждому отдельному элементу можно обращаться по индексу (число в квадратных скобках) как к обычной переменной:

```
dayTemperatures[4] = 24;
dayTemperatures[90] = 45;
```

- Нумерация в массиве начинается с 0. Например в массиве размером 10, первый элемент будет иметь индекс – 0, а последний – 9
- В нашей программе создается массив из цветов, которыми зажигается люстра:

```
CRGB colors[COLOR_COUNT]
```

Обратите внимание, что в самом начале стоит не **int** (целые числа), как в примере, а **CRGB** – это тип который позволяет хранить цвет. Данный тип используется в библиотеке FastLED, «прикрепленной» к этому проекту.

COLOR_COUNT – это константа, которая в нашем случае равна 3. Т.е. мы создали массив из 3 цветов.

- Чтобы сразу заполнить массив цветами, после создания мы пишем знак присваивания и в фигурных скобках, через запятую, перечисляем из чего наш массив будет состоять:

```
CRGB colors[COLOR_COUNT] = {CRGB::Red, CRGB::Green, CRGB::Blue};
```

Мы заполнили наш массив 3 цветами: красным, зеленым, синим. Теперь, если мы напишем в программе **colors[0]**, то обратимся к первому элементу массива, т.е. в данном случае – к красному.

- Идея программы: в основном блоке **loop** – последовательно, для каждого светодиода из люстры, задать цвет из массива цветов **colors**, подождать секунду, и повторить тоже самое только для следующего цвета. Когда цвета закончатся – начать все сначала. У нас есть повторяющееся действие – значит будут использоваться **циклы**.

- Циклы можно «вкладывать» друг в друга. В данном проекте, используется цикл с переменной **c**, изменяющей значения от 0 до **COLOR_COUNT-1** (с помощью него будут перебираться цвета), а внутри него – цикл с переменной **i**, изменяющей значение от 0 до **LED_COUNT-1** (с помощью него будут перебираться светодиоды). Получается, для каждого значения из **c**, «перебираются» все значения из **i**.
- Здесь, мы в переменной **c** – храним цвет, который выставляем в данный момент, а в переменной **i** – номер светодиода в люстре с которым работаем:

```
strip[i] = colors[c];
```

Например, в самом начале, **c=0**. Это значит, что вложенный цикл **for i** выставляет цвет всех светодиодов с номерами от 0 до 7 равный **colors[c]**, т.е. **colors[0]**, т.е. красный. Далее **c** увеличит своё значение на 1 и цикл **for i** повторится (но **c** будет равно уже 1 и цвет будет зеленым)

- Можно отметить, что в программе так же присутствует массив из цветов **strip**. Это массив созданный для «связи» со светодиодами люстры. Поэтому его размер равен 8.

ЗАДАНИЕ «РАДУГА»

Изучите и измените предыдущую программу таким образом, чтобы в бесконечном цикле загорались 7 цветов радуги (красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый)

Для того чтобы посмотреть, что должно получиться в итоге, вы можете посмотреть видео, перейдя по ссылке: <https://znatok.ru/link/?CarDuino-Task-4>

БЛОК ВЕРХНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «БЕГУЩИЙ ОГОНЕК»

Изучите и измените предыдущую программу таким образом, чтобы в бесконечном цикле зажигались 7 цветов радуги (красный, оранжевый, желтый, зеленый, где, синий, фиолетовый)

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**top_lights_03**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле красный огонек будет пробегать по всем светодиодам из люстры.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-TopLights-3>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численные значения в программе можно регулировать частоту переключения цветов. Ниже приведен кусок программы и выделены параметры, изменяющие режим работы.

```
int switchDelay = 100;
CRGB color = CRGB::Red;
```

ЦВЕТ ОГОНЬКА

Измените цвет огонька на любой другой, используемый в предыдущих проектах

ЧАСТОТА ПЕРЕКЛЮЧЕНИЯ ЦВЕТОВ

Уменьшите данный параметр до 50 и убедитесь в том, что огонек стал бежать в 2 раза быстрее

РАЗБОР ПРОГРАММЫ

- В данной программе, перед тем как зажигать конкретный светодиод из люстры, используется функция **FastLED.clear();**, которая позволяет очистить (погасить) все цвета из люстры.

ПРОЕКТ «СВЕТОФОР»

В данном проекте показывается как люстра имитирует работу светофора.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**top_lights_04**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: люстра последовательно включает зеленый, желтый, красный сигналы светофора.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-TopLights-4>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численные значения в программе можно регу-

лировать длительность работы красного, желтого и зеленого сигнала светофора.

```
int redTime = 3000;
int yellowTime = 1000;
int greenTime = 6000;
```

ДЛИТЕЛЬНОСТЬ РАБОТЫ ЗЕЛЕННОГО СИГНАЛА

Уменьшите данный параметр до 2000, чтобы зеленый сигнал работал в течение 2000мс (2 секунд)

ДЛИТЕЛЬНОСТЬ РАБОТЫ ЖЕЛТОГО СИГНАЛА

Уменьшите данный параметр до 0, чтобы желтый сигнал совсем не включался

ДЛИТЕЛЬНОСТЬ РАБОТЫ КРАСНОГО СИГНАЛА

Увеличьте данный параметр до 6000, чтобы красный сигнал работал в течение 6000мс (6 секунд)

БЛОК ВЕРХНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «СВЕТОФОР» ПРОДОЛЖЕНИЕ

РАЗБОР ПРОГРАММЫ

- Для того, чтобы задавать цвет светодиодов в данном проекте, используется новая функция – **setRGB**:

```
strip[i].setRGB(255, 255, 0);
```

Здесь, светодиоду из люстры с номером **i**, задается цвет из палитры RGB с составляющими R=255, G=255, B=0 – получается желтый цвет.

- Подробнее об RGB можно узнать в разделе «Что такое RGB». Каждая из составляющая R(красный), G(зеленый), B(синий) цветов задается числом от 0 до 255, так setRGB(255, 255, 0) по сути смешивает максимальное кол-во красного и зеленого и в результате, при смешении, получается желтый. Попробуйте создавать свои цвета!

ЗАДАНИЕ «СПЕЦИАЛЬНЫЕ СИГНАЛЫ»

Напишите программу, которая будут позволять люстре воспроизводить следующие спец.сигналы:

- Сигнал полиции
- Сигнал скорой медицинской помощи
- Сигнал safety-car
- Сигнал машины сафари

Для того чтобы посмотреть, что может получиться в итоге, вы можете посмотреть видео, перейдя по ссылке: <https://znatok.ru/link/?Carduino-Task-4-1>

ПРОЕКТ «БЛИЖНИЙ\ДАЛЬНИЙ СВЕТ»

В данном проекте показывается как люстра синхронно работает с фарами.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «top_lights_05» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: люстра синхронно с фарами на переднем блоке управления в бесконечном цикле будет включать свет на максимум и приглушать.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-TopLights-5>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численные значения в программе можно регулировать яркость ближнего и дальнего света

```
int nearLight = 30;
int farLight = 200;
```

УРОВЕНЬ ЯРКОСТЬ ДАЛЬНОГО СВЕТА

Уменьшите данный параметр до 100 для уменьшения яркости дальнего света. Данный параметр может принимать значения от 0 до 255

УРОВЕНЬ ЯРКОСТЬ БЛИЖНЕГО СВЕТА

Увеличьте данный параметр до 60 для увеличения яркости ближнего света. Данный параметр может принимать значения от 0 до 255

БЛОК ВЕРХНИХ ОГНЕЙ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «РЕЖИМЫ РАБОТЫ СВЕТА ПРИ ЕЗДЕ»

В данном проекте показывается как световые сигналы автомобиля работают при различных режимах езды:

1. Обычная езда – работает люстра, фары и ходовые огни
2. Поворот – фары и люстра приглушаются, включается поворотник
3. Торможение – зажимаются стоп-сигналы, люстра начинает гореть красным
4. Аварийная остановка – поворотники моргают, люстра моргает оранжевым цветом

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля

к компьютеру и загрузите программу **«top_lights_06»** из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).

- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: все огни на машине будут последовательно, в бесконечном цикле, каждые 3 секунды переключать 4 режима работы
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-TopLights-6>

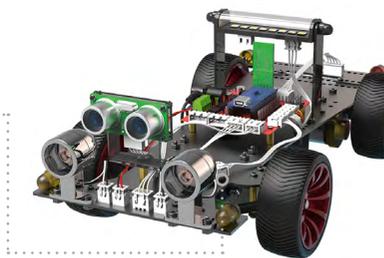
ПЬЕЗОИЗЛУЧАТЕЛЬ

ПРОЕКТ «ДЕМОНСТРАЦИЯ»

Данную программу вы использовали в разделе «Подключение и тестирование» (стр. 46). Перед переходом к другим проектам, рекомендуем убедиться, что пьезоизлучатель по-прежнему работает.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу **«buzzer_check»** из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).



- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы тестирования пьезоизлучателя - по кругу будет проигрываться простая мелодия.
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-Buzzer>

ПЬЕЗОИЗЛУЧАТЕЛЬ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ЗВУКОВОЙ СИГНАЛ ЗАДНЕГО ХОДА»

В данном проекте показывается как можно при помощи пьезоизлучателя имитировать предупредительные сигналы при движении назад.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**buzzer_01**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле пьезоизлучатель будет издавать короткие звуковые сигналы
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Buzzer-1>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать длительность сигналов пьезоизлучателя

```
int beepTime = 50;
```

ПРОДОЛЖИТЕЛЬНОСТЬ СИГНАЛОВ

Увеличьте данный параметр до 200 и убедитесь в том, что сигналы стали намного протяжнее.

РАЗБОР ПРОГРАММЫ

- Для генерации звука при помощи пьезоизлучателя используется функция **tone(pin,frequency)**, где pin – это номер пина к которому подключен пьезоизлучатель, а frequency – это частота сигнала. Для остановки воспроизведения используется функция noTone().
- Другой вариант функции – **tone(pin,frequency, duration)** – первые два параметра совпадают с вышеназванными, а 3 параметр **duration** – длительность сигнала

ПРОЕКТ «ЗВУКОВОЙ СИГНАЛ СКОРОЙ ПОМОЩИ»

В данном проекте показывается как можно при помощи пьезоизлучателя имитировать сигнал машины скорой помощи.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**buzzer_02**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле пьезоизлучатель будет издавать сигналы, имитирующие работу скорой помощи.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Buzzer-2>

РАЗБОР ПРОГРАММЫ

- millis() и параллельная работа функций
- В программе используется новая конструкция – **условие**. Она уже встречалась в собственной функции управления скоростью электродвигателя.
- Общий вид условия:

```
if (a < b)
{
    блок 1;
}else
{
    блок 2;
}
```

Выполняется это следующим образом: «ЕСЛИ условие (a<b) – верно, ТО выполняется блок 1, ИНАЧЕ выполняется блок 2». Это очень похоже на цикл, но проверка условия выполняется только один раз. Стоит отметить что блок ELSE-ИНАЧЕ является необязательным.

- Часто, в программах необходимо выполнять какие-то действия параллельно (например, проигрывать мелодию и менять направление движения), а функция **delay()** – останавливает выполнение программы.

ПЬЕЗОИЗЛУЧАТЕЛЬ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ЗВУКОВОЙ СИГНАЛ СКОРОЙ ПОМОЩИ» ПРОДОЛЖЕНИЕ

Для решения данной проблемы используется «трюк» с функцией `millis()`.

```
curTime = millis();
```

Функция `millis()` **возвращает** значение – сколько прошло миллисекунд с момента запуска Arduino. Это значит, что если с момента запуска прошло ровно 3 секунды, то после выполнения функции `millis()` указанная строчка «превратится» в

```
curTime = 3000;
```

- Звуковой сигнал скорой помощи в данном проекте создается попеременной сменой частоты пьезоизлучателя каждую секунду, т. е. От 0 до 1000мс проигрывается одна частота, от 1000мс до 2000мс – вторая и так по кругу.
- Идея программы заключается в том, что мы в начале, при помощи функции **millis()** отмечаем какое-то время и записываем его в переменную **startTime** (время начала проигрывания первой частоты).

- Далее, каждый виток цикла **loop** проверяется разность **curTime – startTime** – определяется сколько времени прошло с того момента, как мы «поставили отметку» **startTime**. Если эта разность меньше 1000 – то мы должны все еще проигрывать первую частоту, если больше 1000 но меньше 2000, то мы должны проигрывать вторую частоту, если вышли за пределы 2000, то меняем положение стартовой отметки **startTime** на текущее время и все повторяется сначала.

ПРОЕКТ «ЗВУКОВОЙ СИГНАЛ ПОЛИЦЕЙСКОЙ МАШИНЫ»

В данном проекте показывается как можно при помощи пьезоизлучателя имитировать сигнал полицейской машины.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу **«buzzer_03»** из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнется автоматическое выполнение программы: в бесконечном цикле пьезоизлучатель будет издавать сигналы, имитирующие работу полицейской машины
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Buzzer-3>

ПРОЕКТ «ЗВУКОВОЙ СИГНАЛ ПОЖАРНОЙ МАШИНЫ»

В данном проекте показывается как можно при помощи пьезоизлучателя имитировать сигнал пожарной машины.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу **«buzzer_04»** из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнется автоматическое выполнение программы: в бесконечном цикле пьезоизлучатель будет издавать сигналы, имитирующие работу машины скорой помощи
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Buzzer-4>

СЕРВОПРИВОД

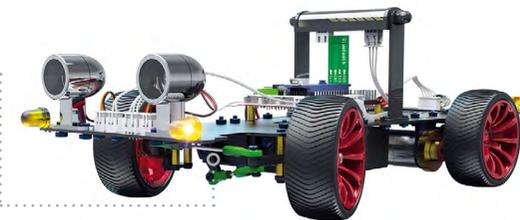
ПРОЕКТ «КАЛИБРОВКА»

Данную программу вы использовали в разделе «Подключение и тестирование» (стр. 40). Перед переходом к другим проектам, рекомендуем убедиться, что сервопривод по-прежнему работает.

Одна из важнейших программ! Данная программа позволяет проверить работоспособность сервопривода и откалибровать крайние положения его вала. Научиться это делать совершенно необходимо для управления положением колес, когда сервопривод установлен на платформу и подсоединен к колесам рулевыми тягами. Ниже приведен фрагмент программы, позволяющей установить крайние положения вала сервопривода.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру.
- Ниже приведен фрагмент программы, позволяющей установить крайние положения вала сервопривода



```
const int servo_right_pos = 110;
const int servo_left_pos = 40;
```

КРАЙНЕЕ ЛЕВОЕ ПОЛОЖЕНИЕ

Выставить значение от 0 до 180 (градусов). Обратите внимание, что для корректной работы данное значение должно быть больше значения крайне-левого положения.

КРАЙНЕЕ ПРАВОЕ ПОЛОЖЕНИЕ

Выставить значение от 0 до 180 (градусов). Обратите внимание, что для корректной работы данное значение должно быть меньше значения крайне-правого положения.

- После загрузки программы, отсоедините USB-кабель от модуля Arduino и включите питание – выключатель питания на плате управления в положение ON.
- При установленных рулевых рычагах необходимо определить крайние правое и левое положение колёс.

- Несмотря на то, что сервопривод способен вращаться на 270 градусов, мы его установили так, что он может вращаться на 180, реальный необходимый угол поворота – около 120 градусов – это связано с особенностью крепления колес.

- Запустите программу и отметьте, поворачиваются ли колёса до конца в каждом из направлений. Если колёса поворачиваются вправо (влево) не до конца, то необходимо увеличить (уменьшить) соответствующее значение, и наоборот – если колёса начинают упираться в корпус, то данное значение необходимо уменьшить (увеличить).

- Рекомендуемый шаг изменения – 10 (например, если сейчас значение – 140, то скорректированное – 130 или 150). Для более точной настройки можно использовать шаг 5.

- После каждой корректировки значений, необходимо загружать программу заново

- Значения, полученные в данной настройке понадобятся при дальнейшей работе.

- Отсоедините USB-кабель от модуля Arduino.

- Включите питание – замкните выключатель на плате управления (ON).

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-Servo>

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).

РАЗБОР ПРОГРАММЫ

- Для управления серводвигателем в программе используется библиотека Servo.
- Подключение библиотеки выполняется при помощи строки:

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

Это немного отличается от того, что мы использовали в блоке верхних огней (мы указывали путь к библиотеке) потому что данная библиотека является одной из стандартных и поставляется вместе с ArduinoIDE.

- В программе используются **глобальные переменные и константы**:

```
Servo servo;
```

```
const int servo_right_pos = 110;
const int servo_left_pos = 40;
```

Глобальные – потому что они могут использоваться в любой из функций. Создание такой переменной происходит вне всех фигурных скобок.

СЕРВОПРИВОД (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «КАЛИБРОВКА» ПРОДОЛЖЕНИЕ

- Для управления серводвигателем создается переменная типа Servo с именем servo. Обратите внимание в том, что в именах важно написание строчных и прописных букв.
- Еще один способ создания константы – создать переменную и написать перед объявлением служебное слово **const**: “const int...” создает переменную типа int и обозначает её константой – неизменяемым значением.
- **servo** – это переменная-объект. Она отличается от обычной переменной, в частности, тем, что в случае обычной переменной в памяти хранятся просто какие-то данные, а в случае объекта – еще набор действий, которые можно выполнять с ними, например в функции **setup** мы производим настройку:

```
servo.attach(SERVO_PIN);
```

Мы пишем имя переменной и через точку мы указываем действие, которое необходимо совершить – **attach**

(от англ. «привязать»), в скобках указываем номер пина к которому подключен сервопривод. Формально, такие действия, которые относятся к объектам называют **методами**, но по сути своей – это **функции**.

- Для того, чтобы задать угол поворота качельки сервопривода, используется метод **write**:

```
servo.write(servo_right_pos);  
delay(2000);
```

Параметром является угол поворота. Внимательно изучите раздел «Загрузка и запуск» данного проекта, чтобы понять ограничения. Кроме того, сама команда на изменение положения сервопривода подается почти мгновенно, но сервопривод поворачивает качельку не так быстро и это время зависит от типа сервопривода и нагрузки на него, поэтому задержка delay после команды write выбирается программистом на свое усмотрение и часто подбирается опытным путем.

МОТОР-РЕДУКТОР

ПРОЕКТ «ДЕМОНСТРАЦИЯ»

Данную программу вы использовали в разделе «Подключение и тестирование» (стр. 42). Перед переходом к другим проектам, рекомендуем убедиться, что мотор-редуктор по-прежнему работает.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «motor_check» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).



- Начнётся автоматическое выполнение программы тестирования мотор-редуктор. Он должен будет вращаться в обоих направлениях с разными скоростями.
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-Motor>

МОТОР-РЕДУКТОР (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ВКЛЮЧЕНИЕ И ВЫКЛЮЧЕНИЕ МОТОР-РЕДУКТОРА»

В данном проекте показывается как включать и выключать мотор-редуктор.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**motor_01**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: мотор-редуктор в бесконечном цикле будет включаться и выключаться.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Motor-1>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать частоту включений мотор-редуктора

```
int switchDelay = 3000;
```

ЧАСТОТА ВКЛЮЧЕНИЯ

Увеличьте данный параметр до 6000 и убедитесь в том, что мотор-редуктор стал вращаться в 2 раза дольше.

РАЗБОР ПРОГРАММЫ

- Мотор-редуктор подключен при помощи драйвера двигателя (вспомогательное устройство для управления мощными устройствами) к 2 пинам.
- Для остановки мотор-редуктора на каждый из этих пинов подается низкое (0В/**LOW**) напряжение – **digitalWrite(MOTOR_PIN_A, LOW);**
- Для включения – только на один из пинов подается высокий (5В/**HIGH**) уровень напряжения – **digitalWrite(MOTOR_PIN_B, HIGH);**

ПРОЕКТ «ИЗМЕНЕНИЕ НАПРАВЛЕНИЯ ДВИЖЕНИЯ»

В данном проекте показывается как изменять направление вращения мотор-редуктора.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**motor_02**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: мотор-редуктор в бесконечном цикле будет изменять направление вращения каждые 5 секунд, останавливаясь на 1 секунду.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Motor-2>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать частоту включений мотор-редуктора

```
int switchDelay = 5000;
```

ЧАСТОТА ВКЛЮЧЕНИЯ

Уменьшите данный параметр до 2000 (2 секунду) и убедитесь в том, что мотор-редуктор стал изменять направление движения намного чаще.

РАЗБОР ПРОГРАММЫ

- Для движения в одну сторону - на один пин подключения мотор-редуктора подается высокий уровень (**HIGH**), а на другой низкий (**LOW**), для того чтобы поменять направление вращения - необходимо на первый подавать **LOW**, а на второй – **HIGH**.

МОТОР-РЕДУКТОР (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ПЛАВНОЕ УВЕЛИЧЕНИЕ СКОРОСТИ С РЕЗКИМ ТОРМОЗОМ»

В данном проекте показывается как можно плавно увеличивать скорость вращения мотор-редуктора.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**motor_03**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле мотор-редуктор плавно, в течение примерно 2.5 секунд будет наращивать свою скорость до максимальной, а потом резко останавливаться.
- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Motor-3>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать частоту включений мотор-редуктора

```
int spDelay = 10;
```

ДЛИТЕЛЬНОСТЬ УСКОРЕНИЯ

Увеличьте данный параметр до 20 и убедитесь в том, что мотор-редуктор стал набирать скорость медленнее.

РАЗБОР ПРОГРАММЫ

- Управление скоростью мотор-редуктора схоже с управлением яркостью светодиода с той лишь разностью, что у мотор-редуктора существует 2 направления вращения и в зависимости от этого функцию analogWrite нужно применять к одному из двух пинов управления мотор-редуктором.

ПРОЕКТ «ПЛАВНОЕ УВЕЛИЧЕНИЕ СКОРОСТИ С ПЛАВНЫМ УМЕНЬШЕНИЕМ»

В данном проекте показывается как можно плавно увеличивать и уменьшать скорость вращения мотор-редуктора, меняя при этом направление.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**motor_04**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: в бесконечном цикле мотор-редуктор плавно, в течение примерно 2.5 секунд будет наращивать свою скорость до максимальной в одну сторону, плавно её уменьшать, затем повторять это в другую сторону.

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Motor-4>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать частоту включений мотор-редуктора

```
int spDelay = 10;
```

ДЛИТЕЛЬНОСТЬ УСКОРЕНИЯ

Увеличьте данный параметр до 20 и убедитесь в том, что мотор-редуктор стал изменять скорость медленнее.

РАЗБОР ПРОГРАММЫ

- Важной частью программирования является создание собственных функций. Если в коде существует последовательность команд, которая повторяется несколько раз и предназначена для выполнения одного задания – разумно написать функцию.

МОТОР-РЕДУКТОР (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ПЛАВНОЕ УВЕЛИЧЕНИЕ СКОРОСТИ С ПЛАВНЫМ УМЕНЬШЕНИЕМ» ПРОДОЛЖЕНИЕ

- В текущем и последующих проектах появляется необходимость выставить скорость движения Car-Duino, а это действие объединяет несколько простых действий по работе с пинами и оно будет точно повторяться множество раз – значит разумно написать функцию.
- Изначально, функцию нужно **описать** – определить, что функция будет делать, далее в программе эту функцию можно использовать – **вызывать**.
- Для создания функции, вне блоков **setup** и **loop** напишем:

```
void motorSpeed(int newSpeed)
{

}
```

- Служебное слово **void** – говорит о том, что мы хотим создать функцию, которая просто выполнит действие (о других видах будет говориться далее), **motorSpeed** – это придуманное программистом имя (motorSpeed – англ. «Скорость Мотора»), в скобках после имени идут переменные-параметры, которые будут использовать-

ся в данной функции. В данном примере мы хотим задать скорость мотор-редуктора целым числом от -255 до 255: **int newSpeed** – определяется переменная целое число, с придуманным нами именем. Параметров могло быть несколько – тогда они писались бы через запятую или не было бы совсем – скобки остались бы пустыми.

- Вся строка называется **заголовком функции**. Команды, которые будут выполняться пишутся внутри фигурных скобок {} и называются **телом функции**.
- В теле данной функции описано изменение скорости вращения мотор-редуктора, подробнее разбор будет в следующих проектах, важно заметить – внутри используется параметр **newSpeed** как обычная переменная.
- Уже в основной программе мы можем **вызывать** функцию. Написав где-то **motorSpeed(200)**; начнется выполнение тела написанной нами функции, при этом внутри **newSpeed** будет равен 200. После, выполнение вернется к строке идущей за вызовом функции.
- Каждый раз когда мы пишем функцию **delay(1000)** или **digitalWrite(5, HIGH)** происходит аналогичный процесс, только функции **delay** и **digitalWrite** описаны не в

тексте нашей программе, а в стандартной библиотеке. Кроме того, можно заметить, что блоки **loop** и **setup** так же по сути являются функциями.

ЗАДАНИЕ «КОРОБКА ПЕРЕДАЧ»

Изучите проекты выше и напишите программу, которая в цикле переключает 4 установленные скорости вращения мотор-редуктора (движение назад, остановка, медленное движение вперед, быстрое движение вперед)

Для того чтобы посмотреть, что должно получиться в итоге, вы можете перейдя по ссылке: <https://znatok.ru/link/?CarDuino-Task-5> посмотреть видео.

УЛЬТРАЗВУКОВОЙ ДАЛЬНОМЕР

ПРОЕКТ «ДЕМОНСТРАЦИЯ – СОБЛЮДЕНИЕ ДИСТАНЦИИ»

Данную программу вы использовали в разделе «Подключение и тестирование» (стр. 48). Перед переходом к другим проектам, рекомендуем убедиться, что УЗ-дальномер по-прежнему работает.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**ultrasonic_check**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).

- Начнётся автоматическое выполнение программы тестирования работы ультразвукового дальномера – он будет держать определённую дистанцию между автомобилем и препятствием
- Ожидаемый эффект можно увидеть по ссылке: <https://znatok.ru/link/?Carduino-Check-Ultrasonic>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать дистанцию, которую будет пытаться удерживать автомобиль

```
int distanceThreshold= 20;
```

ДИСТАНЦИЯ

Увеличьте данный параметр до 40 и убедитесь в том, что автомобиль стал держать увеличенную дистанцию - 40 см.

РАЗБОР ПРОГРАММЫ

- Для работы с УЗ-датчиком, используется библиотека Ultrasonic:

```
#include «src/Ultrasonic.h»
```

- Для настройки ультразвукового дальномера в программе:

```
Ultrasonic ultrasonic(US_PIN_1, US_PIN_2);
```

- Для того, чтобы узнать расстояние перед ультразвуковым дальномером в сантиметрах используется метод **distanceRead()**:

```
int lSensor = ultrasonic.distanceRead();
```

УЛЬТРАЗВУКОВОЙ ДАЛЬНОМЕР (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ОБЪЕЗД ПРЕПЯТСТВИЯ»

Данный проект позволит автомобилю объезжать препятствия, избегая столкновений.

ВНИМАНИЕ! РЕКОМЕНДУЕМ ДАННЫЙ ПРОЕКТ ВЫПОЛНЯТЬ НА ПОЛУ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «ultrasonic_02» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы избегания столкновений с ультразвуковым дальномером – при обнаружении препятствия перед автомобилем, он будет немного отъезжать и выполняя небольшой поворот, продолжая движение прямо.

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Ultrasonic-2>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать расстояние от препятствия на котором автомобиль будет начинать выполнять маневр

```
int distanceThreshold= 30;
```

ДИСТАНЦИЯ

Увеличьте данный параметр до 50 и убедитесь в том, что автомобиль стал выполнять маневр намного раньше.

ПРОЕКТ «АВАРИЙНЫЙ СТОП»

Данный проект позволит автомобилю при обнаружении препятствия впереди себя останавливаться и включать аварийный режим.

ВНИМАНИЕ! РЕКОМЕНДУЕМ ДАННЫЙ ПРОЕКТ ВЫПОЛНЯТЬ НА ПОЛУ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «ultrasonic_03» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы аварийной остановки - при обнаружении препятствия перед автомобилем, он остановится и включит световые и звуковые аварийные сигналы.

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Ultrasonic-3>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать расстояние от препятствия на котором автомобиль будет останавливаться перед препятствием.

```
int distanceThreshold= 30;
```

ДИСТАНЦИЯ

Увеличьте данный параметр до 50 и убедитесь в том, что автомобиль стал останавливаться на увеличенном расстоянии – 50 см.

BLUETOOTH

УСТАНОВКА ПРОГРАММЫ УПРАВЛЕНИЯ НА МОБИЛЬНОЕ УСТРОЙСТВО ANDROID

В данной главе в схемах используется Bluetooth модуль 98, и для его функционирования необходимо установить на своё мобильное устройство (смартфон или планшет) приложение Znatok.Mobile.

Для этого, считайте приведенный здесь QR-код или перейдите на сайт https://znatok.ru/link/?mobile_app и следуйте указанным там инструкциям для установки

После установки приложения у вас на мобильном устройстве должен появиться такой значок: 

После этого необходимо произвести настройку модуля Bluetooth.

ПРОЕКТ «НАСТРОЙКА BLUETOOTH»

Данный проект позволит проверить работоспособность модуля Bluetooth и сменить его имя.

- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**bluetooth_settings**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).

- Активируйте на вашем мобильном устройстве Bluetooth. При поиске устройств Bluetooth на вашем мобильном устройстве отобразится новое устройство с именем «MLT-BT05» — это имя вашего модуля Bluetooth.

- Отсоедините USB-кабель от модуля Arduino.

- Включите питание – замкните выключатель на плате управления (ON).

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя значение в программе можно изменить имя модуля, отображаемое на мобильном устройстве.

```
String uartName = "MLT-BT05";
```

НОВОЕ ИМЯ МОДУЛЯ

Измените данное имя на новое (сохранив кавычки). Длина имени может составлять максимум 32 символа и не должна содержать буквы русского алфавита.

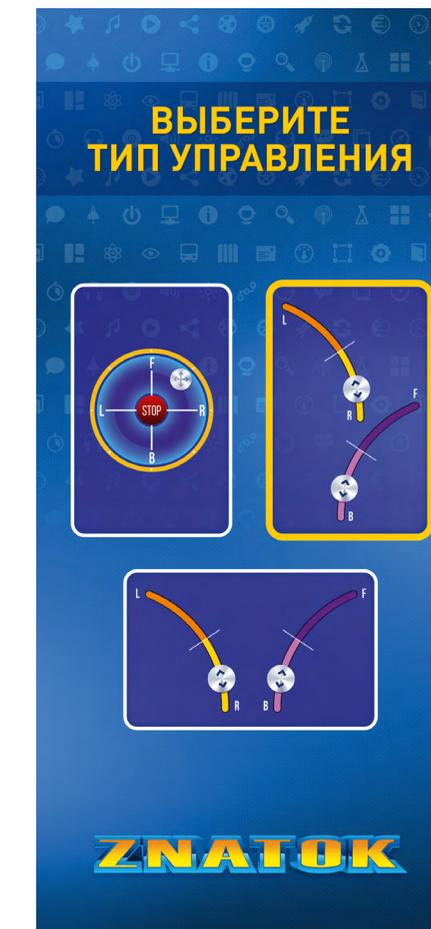
ПРОЕКТ «ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ CAR-DUINO»

Данный проект позволит управлять Car-Duino при помощи мобильного устройства.

ВНИМАНИЕ! РЕКОМЕНДУЕМ ДАННЫЙ ПРОЕКТ ВЫПОЛНЯТЬ НА ПОЛУ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**bluetooth_01**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Откройте на вашем мобильном устройстве приложение Znatok.Mobile и выберите в меню «Электромобиль» один из вариантов управления. Открывшийся пульт управления позволит дистанционно управлять электромобилем: выбирать скорость, включать поворотники, а так же включать спецсигналы.
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Bluetooth-Guide>



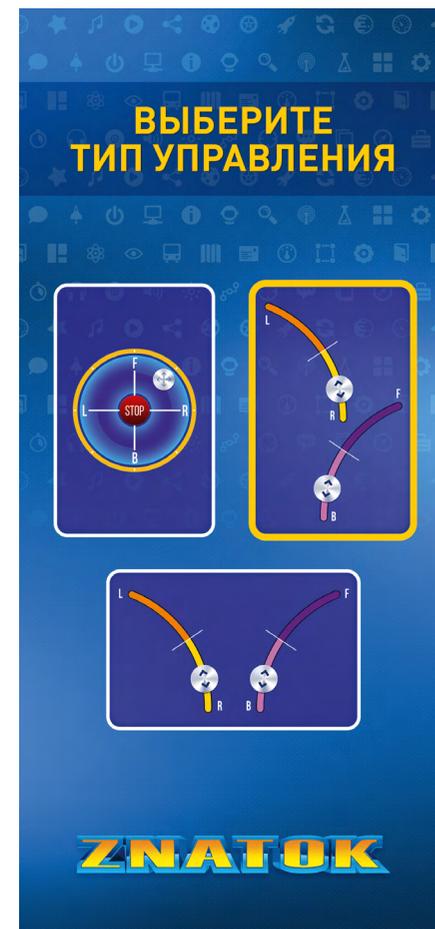
BLUETOOTH (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «МОБИЛЬНЫЙ ИЗМЕРИТЕЛЬ CAR-DUINO»

Данный проект позволит управлять Car-Duino при помощи мобильного устройства.

ВНИМАНИЕ! РЕКОМЕНДУЕМ ДАННЫЙ ПРОЕКТ ВЫПОЛНЯТЬ НА ПОЛУ.

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**bluetooth_02**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Откройте на вашем мобильном устройстве приложение Znatok.Mobile и выберите в меню «Электромобиль» один из вариантов управления. Открывшийся пульт управления позволит дистанционно управлять электромобилем и получать значения дистанции с ультразвукового дальномера, установленного на Car-Duino.



ЭЛЕКТРОМОБИЛЬ

ПРОЕКТ «ДЕМОНСТРАЦИЯ»

Данный проект позволит проверить правильность подключения и совместной работы сервопривода, мотор-редуктора, пьезоизлучателя, переднего, заднего и верхних блоков огней.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**car_check**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы тестирования всего автомобиля: хаотичное движение сервопривода и вращение мотор-редуктора, случайные вспышки света и случайные звуковые сигналы.

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Mobile-Check>

РАЗБОР ПРОГРАММЫ

- В данной программе выполняются случайные действия с случайными параметрами — в Arduino есть возможность задать любое случайное число при помощи функции random():

```
int power = random(0, 256);
```

- Данная функция позволяет получить случайное число из заданного диапазона. В данном примере диапазон — от 0 (включительно) до 255 (включительно).
- Основная идея программы — сгенерировать случайное число — режим работы, а потом в зависимости от этого числа включать или выключать различные устройства. Например, если сгенерируется число 1 — включить движение вперед, 2 — повернуть сервопривод и т. д.
- В программе используется новый тип переменной — логическая переменная **boolean**. Это такая переменная, которая может принимать только два значения: **true** (истина) и **false** (ложь).

ЭЛЕКТРОМОБИЛЬ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ЕЗДА ПО ПРЯМОЙ»

Данный проект позволит автомобилю проезжать вперед и назад в сопровождении световых и звуковых сигналов.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «car_01» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: автомобиль с включенным ближним светом проедет 2 секунды вперед, плавно затормозит в течение 1 секунды, включив стоп-сигналы и далее в течение 3 секунд

будет плавно сдавать назад, сопровождая движение звуковым сигналом.

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Mobile-1>

РАЗБОР ПРОГРАММЫ

- В данном проекте важной особенностью является то, что параллельно идут 3 процесса — управление светом, управление звуком, управление движением.
- Цикл работы (6 секунд) делится на 3 промежутка (от 0 до 2 секунды — движение вперед, от 2 до 3 секунды — торможение, от 3 до 6 секунды — движение назад), которые повторяются в цикле. При помощи функции millis() определяется какой из промежутков идет сейчас.

ПРОЕКТ «ОПТИМАЛЬНЫЙ РАЗВОРОТ»

Данный проект позволит автомобилю проехать вперед и выполнить оптимальный разворот в 2 движения в сопровождении световых и звуковых сигналов.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «car_02» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: автомобиль с включенным ближним светом проедет 2 секунды вперед, плавно затормозит в течение 1 секунды, с использованием поворотников выполнит движение

влево-назад по дуге, далее, остановившись выполнит движение прямо-вправо по дуге, затем выровняв колеса, поедет прямо (в сторону обратную первоначальному движению).

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Mobile-2>

РАЗБОР ПРОГРАММЫ

- В данной программе параметры поворота и движения зависят от характера поверхности, заряда аккумулятора. Но для наибольшей точности необходимо правильно откалибровать углы поворота сервопривода — см. проект «Калибровка»

ЗАДАНИЕ «SAFETYCAR»

Изучите проекты выше и напишите программу, которая позволит автомобилю двигаться по кругу со средней скоростью, при этом люстра должна работать в режиме SafetyCar.

ЭЛЕКТРОМОБИЛЬ (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «ШИРОКАЯ ЗМЕЙКА»

Данный проект позволит автомобилю выполнить серию из 4 поворотов (2 влево, 2 вправо), сопровождая каждый поворот световыми и звуковыми сигналами.

ВНИМАНИЕ! РЕКОМЕНДУЕМ ДАННЫЙ ПРОЕКТ ВЫПОЛНИТЬ НА РОВНОЙ ПОВЕРХНОСТИ РАЗМЕРОМ НЕ МЕНЕЕ 1Х2 МЕТРА.

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «car_03» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы: автомобиль выполнит серию из 4 поворотов, сопровождая каждый поворот звуковыми сигналами.
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Mobile-3>

РАЗБОР ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать число поворотов.

```
int turnCount = 4;
```

ЧИСЛО ПОВОРОТОВ

Увеличьте данный параметр до 2 и убедитесь в том, что автомобиль стал проезжать намного меньшее расстояние.

РАЗБОР ПРОГРАММЫ

- Чтобы определить какой поворот выполнять в данный момент – влево или вправо — определяется является он четным или нечетным.
- Для определения четности, зная какой по счету поворот выполняется, вычисляется остаток **от деления на 2** – если остаток равен 0, то число четное, иначе – нечетное.

- Остаток от деления вычисляется при помощи оператора %:

```
if (curTurn % 2 == 0)
{
    //Левый поворот
}else
{
    //Правый поворот
}
```

ЭНКОДЕР

ПРОЕКТ «ДЕМОНСТРАЦИЯ»

Данную программу вы использовали в разделе «Подключение и тестирование» (стр. 44). Перед переходом к другим проектам, рекомендуем убедиться, что энкодер по-прежнему работает.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ДЕЛАТЬ ЕГО НА ПОЛУ, ЛИБО ПОДЛОЖИТЬ ПОД ПЛАТФОРМУ КАКОЙ-ТО ПРЕДМЕТ, ЧТОБЫ КОЛЁСА НЕ КАСАЛИСЬ ПОВЕРХНОСТИ.

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу **«hall_sensor_check»** из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).

- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы тестирования работы энкодера: частота моргания всеми огнями будет зависеть от скорости вращения колес, т.е. частоты прохождения магнитов рядом с энкодером - датчиком Холла.
- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Check-Encoder>

ПРОЕКТ «ЕЗДА НА ЗАДАННОЕ РАССТОЯНИЕ»

Данный проект позволит автомобилю проехать заданное расстояние при помощи энкодера.

ВНИМАНИЕ! РЕКОМЕНДУЕМ ДАННЫЙ ПРОЕКТ ВЫПОЛНЯТЬ НА ПОЛУ.

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу **«encoder_01»** из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнётся автоматическое выполнение программы – автомобиль проедет заданное расстояние – 40 см
- Работа энкодера связана с понятиями точность и погрешность - невозможно сделать устройство, которое

будет проезжать заданное расстояние с абсолютной точностью. Запуская автомобиль из раза в раз и замеряя линейкой пройденное расстояние, скорее всего, вы будете получать различные значения между 38см и 42см. Мы считаем, что данная точность позволит смоделировать большинство ситуаций и решить множество задач.

- Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Encoder-1>

ИЗМЕНЕНИЕ ПРОГРАММЫ

Изменяя численное значение в программе можно регулировать расстояние, которое проедет автомобиль.

```
int distance= 40;
```

ДИСТАНЦИЯ

Увеличьте данный параметр до 80 и убедитесь в том, что автомобиль стал проезжать дистанцию в 2 раза длиннее - 80 см.

ЭНКОДЕР (ПРОДОЛЖЕНИЕ)

ПРОЕКТ «КРУИЗ-КОНТРОЛЬ»

Данный проект позволит автомобилю держать фиксированную скорость независимо от нагрузки и уклона дороги.

ВНИМАНИЕ! ПРИ ВЫПОЛНЕНИИ ДАННОГО ПРОЕКТА РЕКОМЕНДУЕМ ЛИБО ПОМЕЩАТЬ НА АВТОМОБИЛЬ ГРУЗ, ЛИБО ИСПОЛЬЗОВАТЬ БОЛЬШУЮ НАКЛОННУЮ ПОВЕРХНОСТЬ И РЕЖИМ КРУГОВОГО ДВИЖЕНИЯ (см. видео по ссылке <https://znatok.ru/link/?CarduinoEncoder-CruiseControl>)

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**encoder_02**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).

- Отсоедините USB-кабель от модуля Arduino.
- Включите питание – замкните выключатель на плате управления (ON).
- Начнется автоматическое выполнение программы – автомобиль начнет езду с малой скоростью. При этом, встретив наклон/спуск или другую поверхность его скорость останется такой же.
- Работа энкодера связана с понятиями точность и погрешность - невозможно сделать устройство, которое будет постоянно ехать с одной скоростью с абсолютной точностью. Чтобы узнать свою скорость автомобилю требуется определенное время, в нашем автомобиле это время может достигать 0.5 секунд (не всегда), поэтому иногда можно замечать резкие изменения в скорости движения автомобиля.

ПРОЕКТ «ПАРАЛЛЕЛЬНАЯ ПАРКОВКА»

Данный проект позволит автомобилю при помощи ультразвукового датчика расстояния и энкодера выполнять параллельную парковку.

ДЛЯ ВЫПОЛНЕНИЯ ДАННОГО ПРОЕКТА НЕОБХОДИМО ПОВЕРНУТЬ УЛЬТРАЗВУКОВОЙ ДАТЧИК РАСТОЯНИЯ НАПРАВО НА 90 ГРАДУСОВ (см. Рисунок).

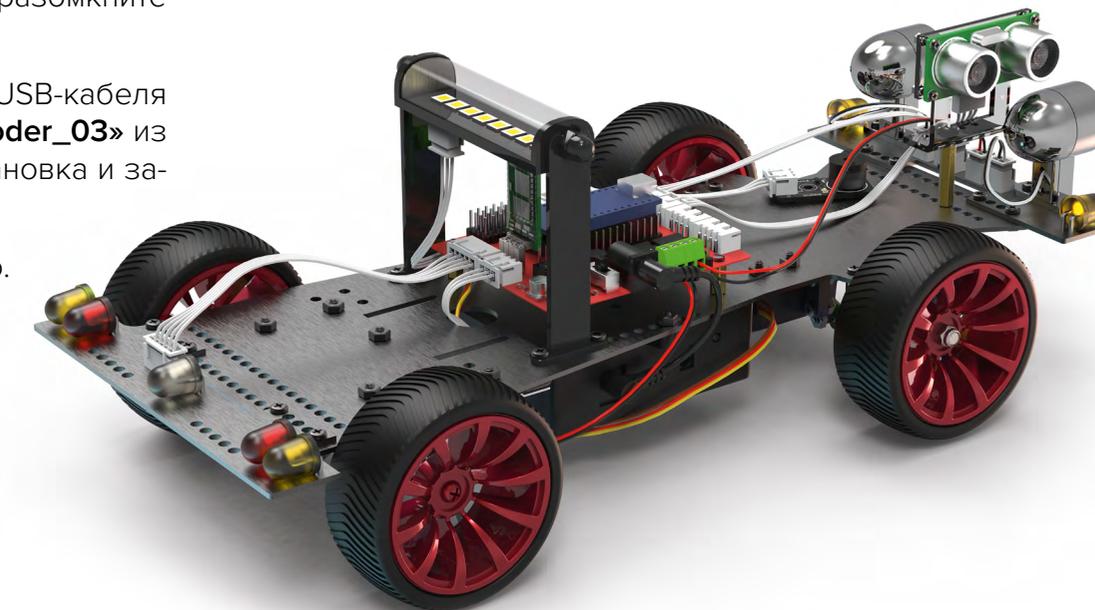
ВНИМАНИЕ! РЕКОМЕНДУЕМ ДАННЫЙ ПРОЕКТ ВЫПОЛНЯТЬ НА ПОЛУ.

Ожидаемый эффект можно увидеть перейдя по ссылке: <https://znatok.ru/link/?Carduino-Encoder-Parking>

ЗАГРУЗКА И ЗАПУСК

- Выключите питание платы управления – разомкните выключатель на плате управления (OFF).
- Подключите модуль Arduino при помощи USB-кабеля к компьютеру и загрузите программу «**encoder_03**» из папки «LaboratoryProjects» (см. раздел «Установка и загрузка программ»).
- Отсоедините USB-кабель от модуля Arduino.

- Включите питание – замкните выключатель на плате управления (ON).
- Начнется автоматическое выполнение программы - автомобиль начнет поиск свободного места для парковки, а потом начнет выполнять маневр по её выполнению.
- Работа энкодера связана с понятиями точность и погрешность - невозможно сделать устройство, которое будет проезжать заданное расстояние с абсолютной точностью. Поэтому для корректной работы данной программы требуется хорошо откалиброванное положение серводвигателя в автомобиле и чуть большее, чем может интуитивно показаться, место для парковки.



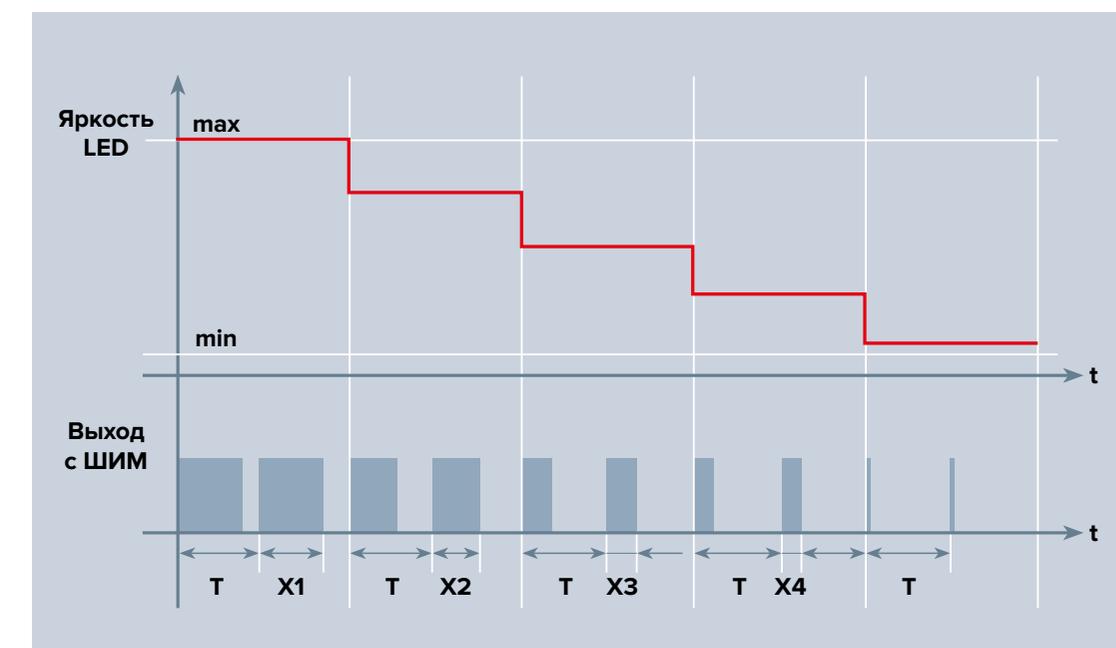
ШИРОТНО-ИМПУЛЬСНАЯ МОДУЛЯЦИЯ

Для регулировки яркости светодиодов или скорости вращения электродвигателей часто используется широтно-импульсная модуляция (ШИМ или PWM). Формальное определение ШИМ звучит так – процесс управления мощностью, подводимой к нагрузке, путём изменения скважности* импульсов, при постоянной частоте.

Принцип работы ШИМ можно посмотреть, считав смартфоном этот QR-код или открыв соответствующую ссылку на сайте www.znatok.ru в ОПИСАНИИ набора «CAR-duino», в разделе ИНСТРУКЦИЯ.



На графике приведен пример управления яркостью светодиода (LED). Из графика видно, что период импульсов T на выходе постоянен, а длительность импульсов X меняется. Чем короче импульсы, тем меньшая мощность поступает на светодиод и яркость его уменьшается. Аналогично управляются лампа накаливания или электродвигатель, у которого будет меняться скорость вращения.

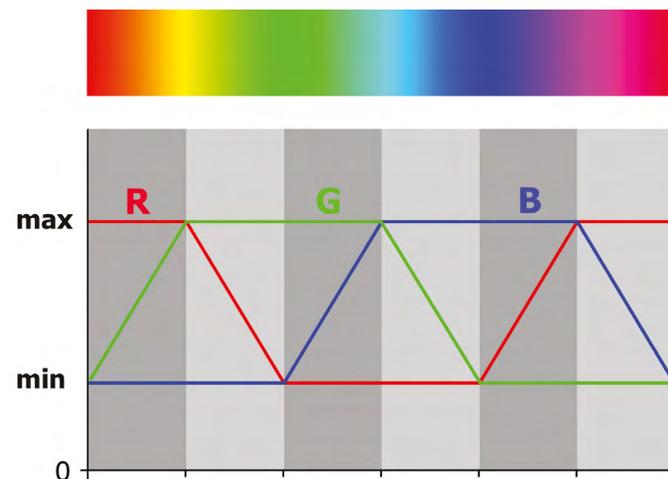
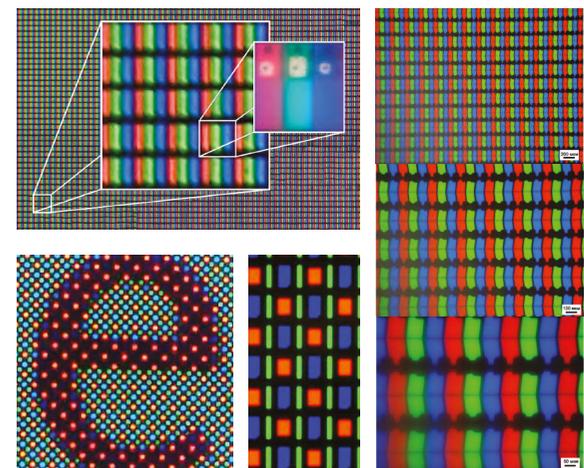


*СКВАЖНОСТЬ (S) – ОТНОШЕНИЕ ПЕРИОДА ИМПУЛЬСОВ T К ДЛИТЕЛЬНОСТИ ИМПУЛЬСА X , Т.Е. $S=T/X$.
У СИГНАЛА ТИПА «МЕАНДР» СКВАЖНОСТЬ РАВНЯЕТСЯ 2, Т.К. ПЕРИОД В ДВА РАЗА БОЛЬШЕ ДЛИТЕЛЬНОСТИ ИМПУЛЬСА.



RGB

Вся гамма цветов, которую вы видели в проектах с «люстрой», видите на экранах смартфонов, мониторах компьютеров, телевизорах, создается при помощи всего трёх цветов – **красного, зелёного и синего (RGB)**! Здесь приведены фотографии различных экранов, сделанные под микроскопом. Форма и размер светодиодов разные, а цветов всего три.



На этом графике показана зависимость получаемого цвета от яркости свечения каждого светодиода. Обратите внимание, что одновременно горят только два из трёх светодиодов. Если подключить ещё и третий светодиод, то цветовая палитра станет ещё разнообразнее.

Смешивание света можно посмотреть, считав смартфоном этот QR-код или открыв соответствующую ссылку на сайте www.znatok.ru в ОПИСАНИИ набора «CAR-duino», в разделе ИНСТРУКЦИЯ.

Но то, что получается со светом, не всегда получается при смешивании красок!



ЕСЛИ ВАС ЗАИНТЕРЕСОВАЛА
ТЕМА RGB, ОБРАТИТЕ
ВНИМАНИЕ НА НАБОР
ARDUINO LIGHT MINI

Подробности можно узнать здесь:



vk.com/znatok_ru



ok.ru/znatokru

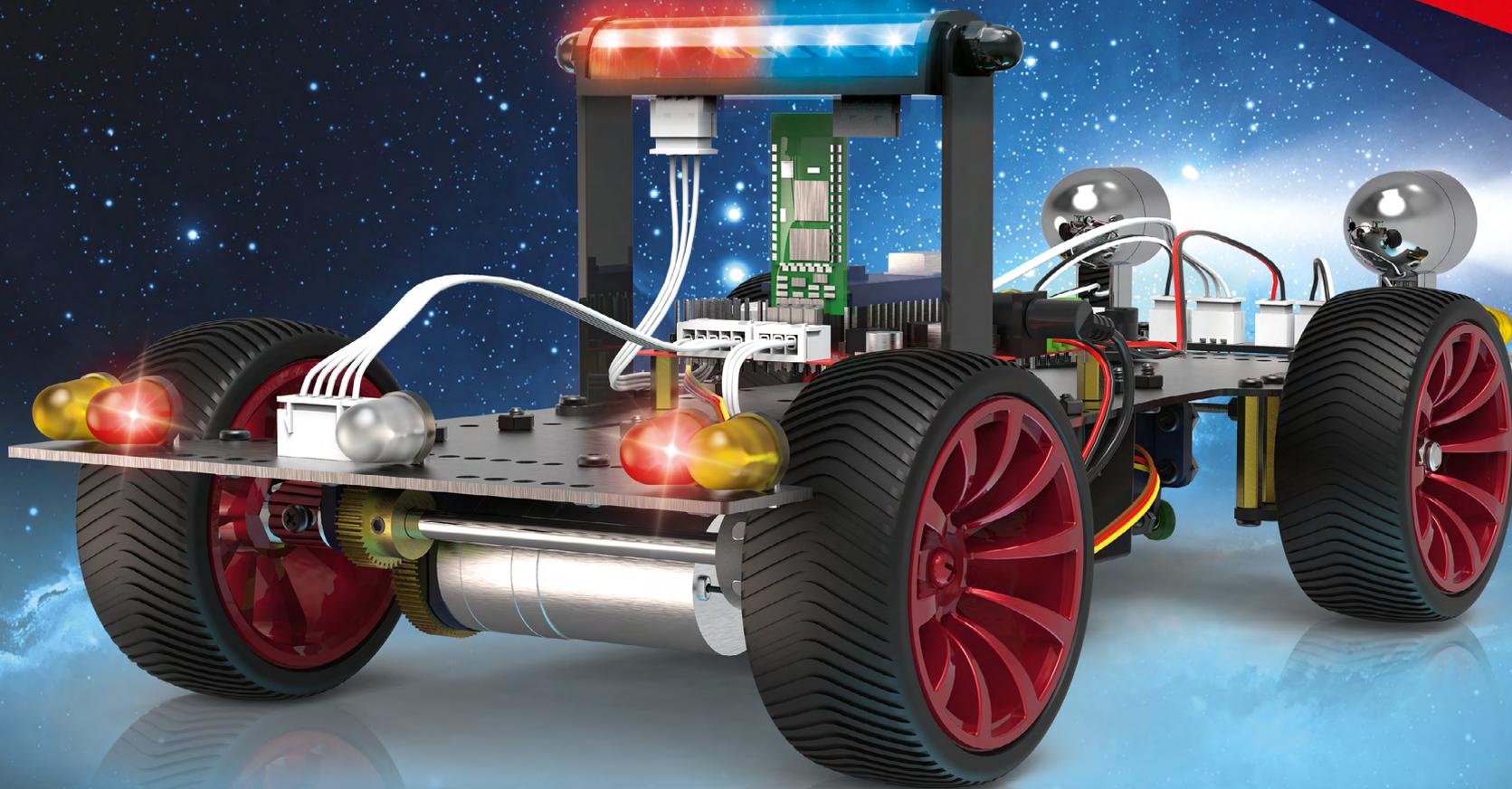


youtube.com/znatok

играем и учимся
ЭЛЕКТРОННЫЙ КОНСТРУКТОР
ЗНАТОК™

Car·DUINO

**ДО НОВЫХ
ВСТРЕЧ!**



Дизайн и цвет элементов может отличаться от приведённого

RUS-01