Муниципальное автономное учреждение
дополнительного образования
«Центр внешкольной работы»
627070 Тюменская обл. с. Омутинское ул. Тимирязева, 1А, помещение 1; тел. 8(34544) 3-18-02, cvr_omut@mail.ru

Программа принята на заседании педагогического совета «19» апреля 22г.

УТВЕРЖДАЮ
Директор МАУ
«Центр информационно-библиотечного обслуживания населения Омутинского района»
Т.А.Бабенкова
«20» апреля 22г.

УТВЕРЖДАЮ
Директор МАУ ДО
«Центр внешкольной работы»
С.В. Жукова
«21» апреля 22 г.

Дополнительная общеобразовательная общеразвивающая разноуровневаяпрограмма технической направленности по робототехнике «ArduЛаб»

Возраст обучающихся:12-17 лет
Срок освоения:3 года

Автор - составитель:
Белкин Д.В.
педагог дополнительного образования

с. Омутинское, 2022

СОДЕРЖАНИЕ

Пояснительная записка

В современном мире с каждым годом растет потребность в специалистах, обладающих навыками построения робототехники, так как данное направление востребовано практических для всех областей экономики. Перед педагогами стоит задача уже со школьной скамьи готовить будущих инженеров и техников, следовательно, возникает и необходимость в разработке и реализации программ дополнительного образования инженерно-технической направленности.

В Концепции развития дополнительного образования детей до 2030 года сказано, что в рамках реализации дополнительных общеобразовательных программ технической направленности необходимо создать условия для вовлечения детей в создание искусственно-технических и виртуальных объектов, построенных по законам природы, в приобретение навыков в области обработки материалов, электротехники и электроники, системной инженерии, 3D-прототипирования, цифровизации, работы с большими данными, освоения языков программирования, машинного обучения, автоматизации и робототехники, технологического предпринимательства, содействовать формированию у обучающихся 14 современных знаний, умений и навыков в области технических наук, технологической грамотности и инженерного мышления.

В наше время робототехники и компьютеризации, подростков необходимо учить решать задачи с помощью автоматов, которые он сам может спроектировать, защищать свое решение и воплотить его в реальной модели, т.е. непосредственно сконструировать и запрограммировать. Предмет робототехники – это создание и применение роботов, других средств робототехники и основанных на них технических систем и комплексов различного назначения. Робототехника - это прикладная наука, занимающаяся разработкой и эксплуатацией интеллектуальных автоматизированных технических систем для реализации их в различных сферах человеческой деятельности.

Одна из задач дополнительного образования - создание комфортной среды, позволяющей ученику раскрыть собственный потенциал. Это позволит ему свободно действовать, познавая эту среду, а через неё и окружающий мир. Новая роль педагога состоит в том, чтобы побуждать ребёнка к познанию и к деятельности. В связи с этим образовательная робототехника в школе приобретает все большую значимость и актуальность в настоящее время, так как позволяет заинтересовать обучающихся, разнообразить учебную деятельность, использовать активные формы и методы обучения, работу в группах, решать задачи практической направленности.

Актуальность программы обусловлена тем, что в настоящее время владение навыками построения робототехники сейчас востребовано практически в любой сфере экономики, а высококвалифицированные специалисты, обладающие знаниями в области робототехники и мехатроники крайне востребованы. С учётом того, как быстро в современном мире развиваются технологии и как растут информационные объёмы, специалистов по данному направлению лучше всего начинать готовить именно со школьной скамьи.

В настоящий момент в нашей стране развиваются нанотехнологии, электроника, механика и программирование. Созревает благодатная почва для развития компьютерных технологий и робототехники. Успехи страны в XXI веке будут определять не природные ресурсы, а уровень интеллектуального потенциала, который определяется уровнем самых передовых на сегодняшний день технологий. Уникальность образовательной робототехники заключается в возможности объединить конструирование и программирование в одном курсе, что способствует интегрированию преподавания информатики, математики, физики, черчения, естественных наук с развитием инженерного мышления, через техническое творчество. Техническое творчество — мощный инструмент синтеза знаний, закладывающий прочные основы системного мышления. Таким образом, инженерное творчество и лабораторные исследования —

многогранная деятельность, которая должна стать составной частью повседневной жизни каждого обучающегося.

К особенностям реализации данной Программы можно отнести возможность обучения с применением дистанционных технологий. В настоящее время, количество и качество онлайн-сервисов позволяет с легкостью проводить обучение в ставшем уже привычном для нас онлайн формате, делая процесс обучения более гибким и доступным.

Дополнительная общеобразовательная общеразвивающая разноуровневая программа **технической направленности** по робототехнике «ArduЛаб» сокращенно от «Лаборатория Arduino» (далее – Программа) направлена на привлечение обучающихся к современным технологиям конструирования, программирования и использования роботизированных устройств на базе платформы Arduino.

В современных требованиях к обучению, воспитанию и подготовке детей к труду важное место отведено формированию активных, творческих сторон личности. Применение робототехники на базе микропроцессоров Arduino, различных электронных компонентов (датчиков и модулей расширения) в учебном процессе формирует инженерный подход к решению задач, дает возможность развития творческого мышления у детей, привлекает воспитанников к самостоятельным исследованиям в межпредметных областях.

Данная Программа разработана в соответствии со следующими нормативно-правовыми документами:

- Федеральный Закон от 29.12.2012 № 273-ФЭ «Об образовании в РФ»;

- Концепция развития дополнительного образования детей до 2030 года (Распоряжение Правительства РФ от 31 марта 2022 г. № 678-р);

- Постановление Главного государственного санитарного врача РФ от 28.09.2020 № 28 «Об утверждении СанПиН 2.4.3648-20 «Санитарно эпидемиологические требования к организациям воспитания и обучения, отдыха и оздоровления детей и молодежи»;

- Приказа Минобрнауки России от 23.08.2017 № 816 «Об утверждении Порядка применения организациями, осуществляющими образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ» (Зарегистрировано в Минюсте России 18.09.2017 N 48226);

- Приказ Министерства просвещения России от 09 ноября 2018 г. № 196 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам»;

- Письмо Минобрнауки России от 11.12.2006 г. № 06-1844 «О примерных требованиях к программам дополнительного образования детей»;

- Письмо Департамента государственной политики в сфере воспитания детей и молодежи Министерства образования и науки Российской Федерации от 18.11.2015 № 09-3242 «Методические рекомендации по проектированию дополнительных общеобразовательных программ (включая разноуровневые программы)»;

- ГОСТ Р 7.0.5-2008 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая ссылка. Общие требования и правила составления»;

- Уставом МАУ ДО "ЦВР" (приказ АОМР № 887-п от 02.11.2018 г.);

- Положение о разработке и утверждения ДООП в МАУ ДО "ЦВР" (Приложение №8 к приказу МАУ ДО "ЦВР" № 32-од от 30.08.2021г.)

Новизна Программы состоит том, что в качестве материально-технического обеспечения на занятиях с платформой Arduino используется набор Arduino GyverKIT, автором и вдохновителем которой является известный в «технических кругах» российский техноблогер AlexGyver (Алекс Гайвер)*.

---

*Алекс Гайвер – популярный российский блогер-милионник, «ардуинщик», инженер, изобретатель, ведущий канала на Youtube, Яндекс Дзен, ВК. Студент аэрокосмического факультета МГТУ имени Н.Э. Баумана.

В отличие от ставших привычными всем наборов «Матрешка Z» или «Амперка», данный набор содержит значительное количество датчиков и компонентов, позволяющих выйти за рамки привычных образовательных программ по Arduino, огромную методическую базу различных примеров и проектов по работе с ним. В отличие от LEGO роботов, которые собираются из готовых деталей, робототехника на основе Arduino открывает больше творческих возможностей, где можно использовать практически все, что есть под руками. Цикл занятий по Программе охватывает все стандартные операторы и функции Arduino, и построен таким образом, что от занятия к занятию у обучающегося идёт плавное формирование “базы”. Каждый последующий урок содержит в себе информацию предыдущих или тесно с ними взаимосвязан, далее уроки усложняются и становятся комплексными.

Учебный материал программы консолидирован по принципу разноуровневости, которая позволяет более вариативно подходить к организации образовательного процесса и активно подстраиваться под потребности обучающихся. Минимальным уровнем сложности общеразвивающей программы должен быть уровень, обеспечивающий прием всех обучающихся без предъявления специальных требований. Именно поэтому содержание данной Программы, учитывает приём обучающихся без проведения индивидуального отбора. На стартовом уровне, нами был предусмотрен краткосрочный и небольшой по объему ознакомительный модуль, который носит мотивационный, общекультурный характер, предполагает овладение первичными простейшими навыками по робототехнике на базе платформы Arduino, содействуют самопознанию, самоопределению обучающихся, имеет междисциплинарный характер и призван обеспечить возможность занятий по программе всем детям независимо от способностей и уровня общего развития. Обеспечение «пробного доступа» на программу в виде ознакомительного модуля позволяет определить уровень, к освоению которого готов ребенок в данном виде деятельности.

Большую часть времени, обучающиеся в объединении заняты практической работой (особенно это касается базового и продвинутого уровня).

Для этой цели как нельзя лучше подходит метод проектов, который реализует главный смысл и назначение обучения – создает условия для сотрудничества в сообществе исследователей, тем самым помогает воспитанникам выявить свои таланты. Метод проектов ориентирован на достижение целей обучающихся. Он формирует большое количество умений и навыков, опыт деятельности.

Больших успехов, обучающиеся достигают при самостоятельном изучении материала, соотнеся объект изучения с действительностью или конкретным объектом. Задача педагога состоит лишь в том, чтобы верно мотивировать обучающегося на выполнение задач и выдать все необходимые для этого инструменты.

Так, постигая основы создания технических проектов, воспитанники имеют возможность на практике выявить и устранить имеющиеся недостатки своих решений, опираясь на конкретные цели и задачи которые должно выполнять их устройство.

Использование проектной деятельности в образовательном процессе является эффективным способом мотивации к занятию конструкторской деятельностью, а также к самостоятельному поиску и выбору оптимального решения. Пережитый опыт подготовки может послужить начальным фундаментом для выбора будущей профессии.

Отличительной особенностью данной Программы является также наличие уникальной рейтинговой системы Приложение №.1 Построенной на принципе игрофикации, но имеющей вполне материальный результат. Являясь способом мотивации, а также выявления и поощрения результативных учащихся.

Педагогическая целесообразность заключается в том, что, занимаясь по Программе, неизбежно происходят изменения в восприятия у обучающихся технических дисциплин, переход из разряда умозрительных в разряд прикладных. Применение на практике теоретических знаний, полученных на

математике или физике, ведет к более глубокому пониманию основ, закрепляет полученные навыки, формируя образование в его наилучшем смысле.

Обучающиеся в процессе освоения Программы задают вопросы и решают поставленные задачи. Материал программы не дает обучающимся всего того, что им нужно знать. Вместо этого они задаются вопросом о том, что знают, и изучают еще не освоенные моменты. В процессе работы с данным оборудованием воспитанники овладевают ключевыми компетенциями:

- коммуникативные компетенции;
- учебно-познавательные компетенции;
- информационно-коммуникационные технологии;
- речевые компетенции;
- компетенции деятельности;
- ценностно-смысловые компетенции;
- компетенции личностного самосовершенствования;
- читательские компетенции.

Целесообразность изучения данного курса определяется:

- востребованностью специалистов в области программируемой микроэлектроники в современном мире;
- возможностью развить и применить на практике знания, полученные на уроках математики, физики, информатики;
- возможностью предоставить ученику образовательную среду, развивающую его творческие способности и амбиции, формирующую интерес к обучению, поддерживающую самостоятельность в поиске и принятии решений.

Практическая значимость изучаемого предмета заключается в мотивировании детей с различными образовательными потребностями и возможностями обучения по Программе. В связи с тем, что дополнительное образование играет значительную роль в ориентации и профилировании детей

и молодёжи относительно будущей профессии, можно с уверенностью говорить о том, что инженерные и технические профессии должны стать самыми престижными, востребованными и высокооплачиваемыми.

Разнообразие и вариативность Программы, предусматривающие получение детьми навыков и умений разного уровня (ознакомительный, базовый и продвинутый), использование современных образовательных технологий, реализация проектов, адаптированных для сельской местности, сетевое взаимодействие, модернизация оборудования – все это позволяет обеспечить переносу полученных знаний и умений, как в аналогичные, так и в измененные условия.

В **сетевом взаимодействии** реализации дополнительной общеобразовательной общеразвивающей программы участвует организация, обладающая ресурсами, необходимыми для осуществления образовательной деятельности: Муниципальное автономное учреждение «Центр информационно-библиотечного обслуживания населения Омутинского района» с. Омутинское, ул. Советская 128.

На основе использования возможностей партнера образовательного процесса обеспечивается:

- вариативное пространство для работы учащихся, создавая условия для усиления учебной мобильности;
- предоставление в безвозмездное пользование помещений для реализации образовательной Программы, проведения соревнований, конкурсов, выставок;
- оказание методической помощи воспитанникам и педагогу в части поиска, подбора информации;
- обобщение и тиражирование педагогического опыта в условиях сотрудничества;
- совместное проведение специализированных мероприятий, соревнований, конкурсов, выставок;

- сотрудники сетевого партнера принимают участие в оценивании результатов проектной деятельности согласно выданных рекомендаций.

Адресат программы: обучающиеся 12-17 лет, принимаются все желающие на основе входного контроля. Условия формирования групп: разновозрастные. Набор на второй и третий год обучения производится после освоения ознакомительного модуля, а также на основании результатов тестирования, наличия базовых знаний, собеседования.

Аудиторные занятия проводятся по адресу: Тюменская область, Омутинский район, с. Омутинское, ул. Тимирязева, 1А., а также с. Омутинское, ул. Советская 128.

Обучение с применением **дистанционных образовательных технологий** организуется в соответствии с положением об организации образовательного процесса с применением электронного обучения и дистанционных образовательных технологий при реализации дополнительных общеобразовательных общеразвивающих программ МАУ ДО «Центр внешкольной работы» https://cvr-omut.tmn.muzkult.ru/media/2022/02/11/1293921421/Polozhenie_po_Dist.Obuch_2.pdf

Под дистанционными образовательными технологиями понимаются образовательные технологии, реализуемые в основном с применением информационно-телекоммуникационных сетей при опосредованном (на расстоянии) взаимодействии обучающихся и педагогических работников.

Воспитанники вправе выбирать наиболее удобную платформу для проведения занятий: Discord, Viber или Zoom.

Целесообразность применения дистанционных технологий в данной программе заключается в том, что они позволяют расширить возможности получения дополнительного образования детьми, проживающими в отдалённых сельских поселениях района либо детьми с ограниченными возможностями здоровья. Порой использование дистанционных технологий становится просто необходимым, например, в период карантина либо самоизоляции.

Благодаря современным информационным технологиям обучающиеся могут использовать различные информационные ресурсы. Это актуально на сегодняшний день и очень востребовано. Обучающиеся самостоятельно используют самые разные источники информации, тем самым, приобретают знания, сами для себя определяют способы познавательной деятельности.

Цель и задачи программы

Цель программы: развитие научно-технического и творческого потенциала личности ребенка посредством обучения основам электротехники и программирования на базе платформы Arduino.

Задачи программы:

Образовательные:

- научить выстраивать гипотезу и сопоставлять с полученным результатом;
- научить навыками научно-технического конструирования и моделирования;
- научить четко и точно излагать свои мысли и технические замыслы;
- дать первоначальные знания о конструкции робототехнических устройств;
- дать первоначальные знания по основным законам электричества и ознакомить учащихся с основами электротехники;
- научить основам программирования микроконтроллера Arduino на языке C++;
- обучить самостоятельному проектированию и программированию устройства, которое решает практическую задачу;

Развивающие:

- сформировать представление об основных законах робототехники;
- развивать интерес к научно-техническому творчеству, технике, современным технологиям;
- развивать алгоритмическое и логическое мышление;

- развивать способности учащихся творчески подходить к проблемным ситуациям и самостоятельно находить решения;
- развивать навыки, связанные с правильным поиском, обработкой информации в сети Интернет и представлением результатов своей деятельности;
- развивать способности работы индивидуально и в командах разного качественного и количественного состава группы;
- способствовать заинтересованности в самостоятельном расширении кругозора в области конструирования робототехнических систем.
- развивать нестандартное мышление, а также поисковые навыки в решении прикладных задач;
- развивать способность работать в коллективе, умение оказывать поддержку в реализации чужих идей и взаимодействие для достижения общих целей;
- развивать творческий потенциал в процессе конструирования и программирования роботов;
- развивать познавательный интерес и мотивацию к учению и выбору инженерных специальностей;
- формировать интерес к практическому применению знаний, умений и навыков в повседневной жизни и в дальнейшем обучении;
- формировать навыков коллективного труда;
- развивать коммуникативные навыки;
- стимулировать интерес к смежным областям знаний: математике, геометрии, физике, биологии;
- формировать информационную культуру, умение ориентироваться и работать с разными источниками информации.

Воспитательные:

- воспитывать интереса к конструированию и программированию;
- воспитывать педантичность и внимание к деталям;

- воспитать ценностно-личностные качества: трудолюбие, ответственность, аккуратность, культуру поведения;
- прививать культуру организации рабочего места, правила обращения со сложными и опасными инструментами;
- воспитывать устойчивый интерес к методам технического моделирования, проектирования, конструирования и программирования;
- поддерживать представление учащихся о значимости общечеловеческих нравственных ценностей, доброжелательности, сотрудничества;
- поощрять целеустремленность, усердие, настойчивость, оптимизм, веру в свои силы;
- Пподтверждать высокую ценность таких способностей и качеств, как эмоциональная уравновешенность, рассудительность, эмпатия;
- воспитывать позитивные нравственно-этические установки по отношению к сверстникам и старшему поколению;
- формировать общероссийскую гражданскую идентичность, патриотизм и гражданскую ответственность.

Также задачи каждого из трёх уровней Программы можно сформулировать следующим образом:

Стартовый уровень **«START»** – «репродукция» (воспитанник понимает, может воспроизвести без ошибок);

Базовый уровень **«Arduino BASE»** – «интерпретация» (воспитанник понимает, может применить с изменениями в похожей ситуации);

Продвинутый уровень **«Arduino PRO»** – «изобретение» (воспитанник может самостоятельно спроектировать, сконструировать и запрограммировать устройство, решающее поставленную перед ним практическую задачу).

Ожидаемые результаты

Предметные:

- ранняя профориентация, обеспечивающая ознакомление с современными профессиями и профессиями будущего;
- умение использовать платформу Arduino и электронные компоненты;
- знание синтаксиса языка C++;
- умение работать в программах и сервисах: ScratchDuino, mBlock, ArduBlock, Tinkercad, RemoteXY, Virtuino, Blynk, IoT Wi-Fi контроллер, Bluino Loader, Fritzing, Arduino IDE, EasyEDA;
- умение проводить и анализировать конструирование механизмов, простейших роботов, позволяющих решить конкретные задачи (с помощью стандартных простых механизмов, с помощью материального или виртуального конструктора);
- умение работы с роботами и электронными устройствами;
- умение осуществлять пайку электронных компонентов;
- владение основными терминами Arduino-робототехники и использование их при проектировании и конструировании робототехнических систем;
- освоение основных принципов и этапов разработки проектов;
- владение методами ТРИЗ;
- умение самостоятельно работать с технической документацией, схемами, справочниками;
- умение поиска технической документации в сети Интернет;
- умение снимать данные с датчиков, роботизированных устройств;
- умение проводить настройку и отладку конструкции или программного кода.

Личностные:

- проявление познавательных интересов и активности в технической области;

- воспитание аккуратности и дисциплинированности при выполнении работы;
- формирование общей культуры поведения, навыков культуры труда; воспитание воли, усидчивости, трудолюбия, уважения к своему труду и труду окружающих, стремление к достижению результата поставленной цели;
- формирование опыта совместной деятельности;
- развитие трудолюбия и ответственности за качество своей деятельности;
- формирование нравственно-моральных качеств личности;
- формирование профессиональной ориентации, социализация личности;
- развитие уважительное отношение к товарищам, чувство взаимопомощи;
- формирование стремления к получению качественного законченного результата.

Метапредметные:

- проявление нестандартного подхода к решению учебных и практических задач в процессе моделирования изделия или технологического процесса;
- самостоятельная организация и выполнение различных технических проектов;
- приведение примеров, подбор аргументов, формулирование выводов по обоснованию технико-технологического и организационного решения; отражение в устной или письменной форме результатов своей деятельности;
- Развивать образное и вариативное мышление, воображение, творческие способности;
- Развивать и формировать мыслительные операции (анализа, синтеза, сравнения, обобщения, классификации, аналогии);
- Развивать исследовательскую активность, умение наблюдать и экспериментировать.

Содержание программы

Программа рассчитана на 3 уровня освоения материала– **«START»**, **«Arduino BASE» и «Arduino PRO»** .

Продолжительность освоения Программы составляет три года - 360 академических часов. По индивидуальному маршруту срок обучения варьируется, исходя из индивидуальных особенностей каждого конкретного обучающегося.

Программой, предусмотрена возможность обучения и переход на любой из уровней всем детям, в зависимости от способностей и уровня общего развития, исходя из диагностики их возможностей. Возможно обучение по индивидуальной образовательной траектории, включение в состав одной учебной группы обучающихся с разным уровнем подготовки и прохождение обучения по индивидуальному образовательному маршруту.

Срок реализации программы – 360 часов.

Программа рассчитана на возрастную категорию детей от 12 до 17 лет. Группы обучающихся формируются на основе свободного набора, постоянного состава. Группа от 5 до 10 человек.

Приемы проведения занятия: практическая работа, коллективная работа, самостоятельная работа.

Занятия в зависимости от уровня освоения программы проводятся 1-2 раза в неделю, 2 академических часа, 45 минут, всего 360 часов. При реализации программы с применением дистанционных технологий, также 1-2 раза в неделю, 2 академических часа, 30 минут, всего 360 часов.

Данная программа является логическим продолжением дополнительной общеобразовательной общеразвивающей программы по робототехнике «РобоЛаб» https://cvr-omut.tmn.muzkult.ru/media/2022/03/16/1292437848/DOOP_RoboLab_2022.pdf, реализуемой МАУ ДО «Центр внешкольной работы».

	«START»	«Arduino BASE»	«Arduino PRO»
Число обучающихся в группах (чел.)	до 10	до 5	до 5
Срок обучения (год/ак.часы)	1 год (72 часа)	1 год (144 часа)	1 год (144 часов)
Режим занятий	2 часа в неделю (36 уч.нед.)	4 часа в неделю (36 уч.нед.)	4 часа в неделю (36 уч.нед.)
Самостоятельная работа учащихся в рамках учебного занятия*	-	до 50% учебного времени*	до 70% учебного времени*

*В рамках учебных занятий уделяется большое внимания к самостоятельной работе обучающихся: разбор теоретического материла, составление плана работы над проектом, оформление и подготовка к защите.

Календарный учебный график

Наименование группы/год обучения	Срок учебного года (продолжительность обучения)	Кол-во занятий в неделю, продолжи-тельность одного занятия min	Наименование дисциплины	Всего ак.ч в год	Количество ак.ч. в неделю
START	с 1 сентября по 30 сентября (4 учебных недели)	2 занятия по 45 мин (2 ак.ч.) 1 раз в неделю	Ознакомительный модуль	8	2
	с 1 ноября по 31 мая (32 учебных недели)		Основы электротехники	56	
			Основы пайки	8	
Arduino BASE	с 1 сентября по 15 сентября (2 учебных недели)	2 занятия по 45 мин (2 ак.ч.) 2 раз в неделю	Начало работы с платформой Arduino	8	4
	с 16 сентября по 4 октября (3 учебных недели)		Программирование	22	
	с 5 октября по 30 ноября с 9 января по 30 апреля (23 учебных недели)		Управление	30	
			Индикация	22	
			Датчики	20	
			Моторы, коммутация	10	
	с 1 декабря по 30 декабря с 1 мая по 31 мая (8 учебных недель)		Проектная лаборатория	28	
Arduino PRO	с 1 сентября по 30 ноября с 9 января по 30 апреля (28 учебных недель)	2 занятия по 45 мин (2 ак.ч.) 2 раз в неделю	Управление	12	4
			Индикация	8	
			Датчики	12	
			Моторы, коммутация	8	
			Умный дом	32	
			Интернет вещей	40	
	с 1 декабря по 30 декабря с 1 мая по 31 мая (8 учебных недель)		Проектная лаборатория	32	

Учебный план

Уровень обучения	Дисциплины (модули)	Трудоемкость (количество академических часов)			Формы промежуточной/ итоговой аттестации	
		всего	теория	практика	Очная форма	С применением дистанционных технологий
START	Ознакомительный модуль	8	2	6	Итоговый проект	Проект в среде Tinkercad
					Итоговое тестирование	Итоговое онлайн-тестирование
	Основы электротехники	56	18	38	Наблюдение, итоговая работа. Конкурс задач «Знаток»	Онлайн конкурс задач
	Основы пайки	8	1	7	Итоговый проект	Проект в EasyEDA
	Итого	**72**	**21**	**51**		
Arduino BASE	Начало работы с платформой Arduino	8	2	6	Тестирование	Онлайн-тестирование
	Программирование	22	4	18	Хакатон по программированию	Онлайн-хакатон по программированию
	Управление	30	-	30	Мини проекты	Мини проекты в среде Fritzing , Tinkercad
	Индикация	22	-	22		
	Датчики	20	-	20		
	Моторы, коммутация	10	-	10		
	Проектная лаборатория	32	4	28	Очная защита проекта	Онлайн защита проекта
	Итого	**144**	**12**	**132**		
Arduino PRO	Управление	12	-	12	Мини проекты	Мини проекты в среде Fritzing, Tinkercad
	Индикация	8	-	8		
	Датчики	12	-	12		
	Моторы, коммутация	8	-	8		
	Умный дом	32	2	30	Макет «умного» дома	Проекты в среде Fritzing, Tinkercad
	Интернет вещей	40	4	36		
	Проектная лаборатория	32	2	30	Очная защита проекта	Онлайн защита проекта
	Итого	**144**	**8**	**136**		
ВСЕГО		**360**	**41**	**319**		

Содержание программного материала

Уровень «START»

1. Ознакомительный модуль

Данный структурный элемент программы носит мотивационный, общекультурный характер, предполагает овладение первичными простейшими навыками по робототехнике, содействуют самопознанию, самоопределению обучающихся, имеет междисциплинарный характер и призван обеспечить возможность занятий по программе всем детям независимо от способностей и уровня общего развития.

Итоговый контроль в виде тестирования (Приложение №2) позволяет оценить готовность учащихся к занятиям робототехникой на базе микроконтроллера Arduino и предстоящей самостоятельной работе.

Краткосрочный и небольшой по объему ознакомительный модуль призван обеспечить каждому желающему «пробное» вхождение в программу, чтобы любой учащийся смог ощутить себя в роли разработчика, робототехника, программиста и понять подходит ли такая деятельность именно для него, учитывает ли она его интересы и потребности в самореализации.

Цель модуля: Знакомство с основными видами деятельности по программе АрдуЛаб, на примере создания конкретного технического проекта.

Задачи модуля:

Образовательные

- Дать первоначальные знания о конструкции робототехнических устройств;
- Сформировать первоначальные представления об основных законах робототехники и о конструировании роботов;
- Познакомить с основами разработки алгоритмов при создании робототехнических конструкций;

Развивающие

- Развивать интерес к научно-техническому творчеству, технике, современным технологиям;
- Развивать интерес к смежным областям знаний: математике, геометрии, физике, информатике;

Воспитательные

- Воспитывать интерес к конструированию и программированию;
- Воспитывать ценностно-личностные качества: трудолюбие, ответственность, аккуратность, культуру поведения.

Ожидаемые результаты:

- Возникновение интереса к занятиям техническим творчеством программированию созданию собственных устройств и проектов, стимулирование к изучению теоретической части и занятиям робототехникой;
- Сформированность представлений об основной деятельности на занятиях робототехникой.

Тема проекта ознакомительного модуля может варьироваться исходя из возрастных особенностей и интересов учащихся. На данном этапе гораздо более важно продемонстрировать все этапы создания технического проекта на базе контроллера Arduino и выдать рекомендации по дальнейшей возможности освоения полного курса программы.

1.1. Введение в робототехнику. Цели и задачи работы объединения. Краткий инструктаж по ТБ, ПДД, ППБ.

Знакомство с учащимися. Обзор работы объединения по робототехнике. История создания робототехники, понятие робот и современные образцы робототехники. Характеристика микроконтроллеров и их возможности. Инструктаж по технике безопасности при обращении с ТСО, устройствами, работающими от сети, правилах безопасного обращения с деталями набора Arduino, правилах следования к месту занятий от школы или из дома, безопасный маршрут. Правила поведения на занятиях и в учреждении, режим и расписание, рейтинговая система.

1.2. Проект

Создание проекта «Велосипедный спидометр».

Практика:

1. Изучение принципа работы. Подбор компонентов: Датчик холла или геркон, магнит, плата Arduino NANO, сервопривод MG90, напечатанный на 3D принтере корпус проекта, батарейный отсек, включатель. Домашнее задание: Измерить радиус переднего колеса и рассчитать длину окружности.

2. Сборка устройства. Внесение полученных измерений в скетч Arduino IDE.

3. Установка устройства. Проведение «полевых» испытаний. Доработка кода. Подведение итогов. Проведение итоговой аттестации.

2. Основы электротехники

2.1. Электронный конструктор «Знаток» Введение. Методика сборки. Перечень компонентов.

2.2. Базовые понятия. Понятие электричества.

Электрическая цепь.

Практика: Построение простой электрической цепи.

2.3. Принципиальные схемы.

Практика: Чтение принципиальных схем. Сборка электрических цепей.

2.4. Основные законы электричества. Закон Ома.

Практика: Расчет токов в цепи. Расчет мощности потребителя и источника.

2.5. Управление электричеством.

Практика: Способы коммутации. Ручное и автоматическое управление электричеством.

2.6. Батарейки и аккумуляторы

Принцип работы. Отличия батарейки от аккумулятора. История создания. Виды аккумуляторов. Эффект памяти. Последовательное и параллельное включение батарей.

Практика: Создание батарейки из картошки.

2.7. Переключатели.

Выключатель, кнопка, геркон, сенсорный выключатель.

Практика: Параллельное и последовательное подключение переключателей. Музыкальный дверной звонок управляемый сенсором. Сигнализация на герконе.

2.8. Источники света. Лампы и светодиоды.

Историческая справка. Лампа и светодиод. ВАХ светодиодов. Параллельное и последовательное подключение. Схемы включения. Влияние силы тока на яркость светодиода.

Практика: Подключение лампы и светодиода. Подключение диода без токоограничивающего резистора и с резистором. Расчет резистора.

2.9. SMD светодиоды. Светодиодные ленты.

Структура SMD светодиода. Типы светодиодов. Расчет потребления светодиодной ленты и подбор источника питания.

Практика: Подбор мощности источника питания для различных отрезков и типов лент.

2.10. RGB светодиоды и ленты.

Принципа работы. Особенности подключения. Разновидности RGB светодиодов.

Практика: Подключение RGB светодиодной ленты.

2.11. Электродвигатель и генератор.

Историческая справка. Устройство, принцип действия, условные обозначения. Схема подключения, реверсирование. Зависимость скорости вращения от напряжения. Электрогенератор.

Практика: Простейший электродвигатель из медной проволоки и магнита. Сборка схем на конструкторе.

2.12. Коллекторный двигатель. Коммутация. Н-мост.

Типы электродвигателей и их применение в робототехнике. Способы управления и коммутации.

Практика: Сборка схем коммутации. Реверсирование и изменение скорости вращения коллекторного двигателя.

2.13. Резистор и реостат.

Историческая справка. Внешний вид и обозначение. Резистор как ограничитель тока. Делитель напряжения на резисторе и переменном резисторе. Маркировка резисторов.

Практика: Простейший резистор из грифельного карандаша. Сборка схем на конструкторе.

2.14. Параллельное и последовательное соединение.

Практика: Сборка принципиальных схем.

2.15. Проводники и диэлектрики.

Историческая справка. Определение. Тестеры электропроводимости.

Практика: Определение электропроводимости предметов с помощью гальванометра из конструктора «Знаток».

2.16. Катушка индуктивности.

Внешний вид, устройство и условное обозначение. Применение в схемотехнике. Принцип действия. Электромагнит. Генератор Фарадея.

Практика: Опыт с компасом. Создание фонарика, работающего без батареек.

2.17. Электроизмерительные приборы. Работа с мультиметром.

Гальванометр из набора. Амперметр и Вольтметр, Ваттметр. Правила работы с мультиметром, измерение характеристик. Единицы и диапазоны измерений.

Практика: Изготовление амперметра и вольтметра из гальванометра. Практическая работа с Мультиметром.

2.18. Конденсатор.

Историческая справка. Внешний вид, условные обозначения и принцип работы. Применение в схемотехнике. Последовательное и параллельное подключение в цепи. Переменный конденсатор. Зарядка и разрядка конденсатора. Характеристики. Емкость конденсатора.

Практика: Схема плавного включения света.

2.19. Диод.

Характеристики, внешний вид и условные обозначения. Что общего между диодом и футбольным мячом? Проверка проводимости. Защитные функции. Падение напряжения на диоде. Диодный мост.

Практика: Сборка схем с диодом.

2.20. Транзисторы. Биполярный транзистор и полевой транзистор.

Историческая справка. Характеристики, обозначение и принцип работы. Что общего между транзистором и водопроводным краном? Различие между полевым и биполярным транзисторами. NPN и PNP транзисторы. Применение в цепи. Ключ. Усилитель. Составной транзистор.

Практика: Сборка схем с транзистором. Управление нагрузкой при помощи транзистора.

2.21. Тиристор.

Характеристики, обозначение и принцип работы.

Практика: Включение лампы при помощи тиристора.

2.22. Фоторезистор.

Характеристики, обозначение и принцип работы. Световая и ВАХ характеристики. Применение.

Практика: Защитная сигнализация. Автоматический уличный фонарь.

2.23. Микрофон.

Историческая справка. Электростатические и пьезоэлектрические микрофоны. Принцип действия.

Практика: Сборка схем с применением микрофона.

2.24. Интегральные схемы.

Историческая справка. Краткая теория. Применение.

Практика: Обзор интегральных схем. Разборка схем из состава конструктора «Знаток»

2.25. Цифровая техника. Семисегментный индикатор.

Принцип работы, назначение. Способы управления.

Практика: Вывод различных букв и цифр на индикатор.

2.26. Цифровая техника. Логические элементы.

Понятие логических элементов. Применение.

Практика: Сборка схем логических элементов «И», «НЕ», «ИЛИ-НЕ».

3. Основы пайки

3.1. Введение.

Техника безопасности при пайке. Инструменты и приспособления для пайки. Паяльник, паяльная станция, Типы жал. Лужение жала. Третья рука.

3.2. Практика: Пайка проводов. Пайка компонентов. Распайка компонентов.

3.3. Практика: Изготовление печатных плат. Разводка печатной платы в EasyEDA.

3.4. Практика: Травля и лужение печатной платы. Пайка компонентов.

Уровень «Arduino BASE»

1. Начало работы с платформой Arduino.

Плата Arduino Nano. Обзор. Элементы. Распиновка. Характеристики. Базовые понятия. Возможности микроконтроллера. Питание схемы.

Практика: Что такое скетч и почему светодиод мигает.

1.1. Возможности микроконтроллера.

Переменные и константы. Функции. Составление арифметических выражений. Электрический сигнал. Аналоговый сигнал. Цифровой сигнал. Широтно импульсная модуляция (ШИМ, PWM).

1.2. Работа с Arduino IDE.

Сборка схемы из модулей. Возможные ошибки, FAQ. Установка библиотек.

Практика: Загрузка прошивки.

1.3. Приложения для работы с Arduino.

Обзор приложений для ПК: ScratchDuino, mBlock, ArduBlock. Онлайн сервис «Цепи» в Tinkercad. Обзор приложений для Android: Android RemoteXY, Virtuino, Blynk, IoT Wi-Fi контроллер, Справочник по Arduino, Bluino Loader.

Практика: Подбор оптимальных приложения для работы на ПК и Android.

2. Программирование.

Данный раздел носит чисто практический характер, посвящен изучению стандартных команд для языка C++ и Arduino Wiring. За основу практических занятий взят материал Youtube канала «Заметки Ардуинщика» https://www.youtube.com/c/ЗаметкиАрдуинщика/featured. В качестве опорных конспектов, воспитанникам рекомендуется использовать следующие материалы: уроки Arduino, блокнот Arduino программиста и краткая шпаргалка по основным функциям Arduino Wiring и языку C++. (Приложение №3, Приложения №4, Приложения №5. Данные приложения оформлены с комментариями и примерами в виде кода для большей наглядности и возможности сразу «почувствовать» его и запомнить, как он выглядит. Для полной информации по каждой главе можно обращаться по ссылкам. Можно воспользоваться смартфоном для того, чтобы иметь в одном месте список всех необходимых инструментов для работы с платформой Arduino. Также учащиеся могут воспользоваться приложением для разработчиков «Справочник по Arduino» (Приложение № 6) которое содержит offline мобильный справочник по синтаксису языка C++.

2.1. Синтаксис и структура кода.

2.2. Типы данных, переменные.

2.3. Математические операции.

2.4. Массивы.

2.5. Сравнения, условия и выбор.

2.6. Циклы.

2.7. String-строки.

2.8. Си-строки (массивы символов).

2.9.Функции.

2.10. Монитор порта, отладка.

2.11. Функции времени.

2.12. Цифровые пины.

2.13. Аналоговые пины.

2.14. ШИМ сигнал.

2.15. Аппаратные прерывания.

2.16. Случайные числа.

2.17. Многозадачность в Arduino.

2.18. Переключение задач.

2.19. Избавляемся от циклов.

2.20. Разбивка на вкладки.

2.21. Как написать крупный проект.

3. Управление.

Работа с разделом осуществляется при помощи базы примеров и уроков по компонентам из набора GyverKIT https://kit.alexgyver.ru/tutorials-category/control/.

3.1. Arduino и кнопка.

Практика: Подключение и подтяжка, отработка нажатия, дребезг контактов. Расширенная работа с кнопкой.

3.2. Arduino и потенциометр.

Смежная тема: RGB светодиод, сервопривод.

Проект: Светильник с управляемой яркостью.

3.3. Arduino и энкодер.

Смежная тема: Arduino и дисплей TM1637.

Проект: Кухонный таймер.

3.4. Arduino и джойстик.

Смежная тема: Сервопривод.

Проект: манипулятор.

3.5. Arduino и ИК пульт.

Смежная тема: Arduino и адресная светодиодная лента.

Проект: «Огненный» светильник.

3.6. Arduino и сенсорная кнопка.

Смежная тема: Arduino и реле.

Проект: Светильник на 220 Вольт.

3.7. Arduino и Bluetooth JDY-31.

Смежная тема: Мотор с редуктором, Android приложение Remote XY.

Проект: Bluetooth платформа.

4. Индикация.

Работа с разделом осуществляется при помощи базы примеров и уроков по компонентам из набора GyverKIT https://kit.alexgyver.ru/tutorials-category/indication/.

4.1. Arduino и светодиод.

Проект: Маячок. Маячок с нарастающей яркостью. Бегущие огни.

4.2. Arduino и RGB светодиод.

Смежная тема: Arduino и кнопка.

Проект: Светильник с управляемой яркостью.

4.3. Arduino и зуммер.

Смежная тема: Arduino и кнопка.

Проект: Пианино.

4.4. Arduino и матрица MAX7219.

Проект: Бегущая строка.

4.5. Arduino и дисплей TM1637.

Смежная тема: Arduino и кнопка.

Проект: Счётчик нажатий. Секундомер..

4.6. Arduino и дисплей LCD1602.

Проект: Тестер батареек.

5. Датчики.

Работа с разделом осуществляется при помощи базы примеров и уроков по компонентам из набора GyverKIT https://kit.alexgyver.ru/tutorials-category/sensors/.

5.1. Arduino и фоторезистор.

Смежная тема: Сервопривод.

Проект: Трекер для солнечных батарей.

5.2. Arduino и датчик расстояния HC-SR04.

Смежная тема: Arduino и зуммер. Arduino и дисплей TM1637.

Практика: Парктроник. Дальномер.

5.3. Arduino и термистор.

Смежная тема: Arduino и дисплей TM1637. Кнопка.

Проект: Термостат с таймером.

5.4. Arduino и термометр DS18B20.

Смежная тема: Arduino и дисплей LCD1602.

Проект: Термометр.

5.5. Arduino и часы RTC DS3231.

Смежная тема: Arduino и дисплей TM1637.

Проект: Часы реального времени.

6. Моторы, коммутация.

Работа с разделом осуществляется при помощи базы примеров и уроков по компонентам из набора GyverKIT https://kit.alexgyver.ru/tutorials-category/output/.

6.1. Arduino и сервопривод.

Смежная тема: Arduino и потенциометр.

Проект: Манипулятор.

6.2. Драйвер моторов L9110S.

Смежная тема: Мотор с редуктором.

Проект: Сборка приводной платформы.

6.3. Мотор с редуктором.

Смежная тема: Arduino и реле, Arduino и MOSFET модуль, Arduino и MOSFET транзистор.

Практика: Коммутация мотора различными способами.

7. Проектная лаборатория.

При освоении технологий конструирования робототехнических систем зачастую возникают реальные технические противоречия, которые необходимо

преодолеть. Именно поэтому теоретическая часть раздела включает в себя знакомство с методикой поиска идей для будущих проектов, приемами и алгоритмами, разработанными в рамках теории решения изобретательских задач (ТРИЗ). Воспитанники знакомятся с такими известными методами как: мозговой штурм, синектика или метод аналогий, морфологический анализ, метод фокальных объектов и их разновидностями. Эти и другие методы на примере работы объединения «ArduЛаб» используются для развития креативного подхода к решению поставленных задач.

Цель раздела: Закрепление полученных знаний в ходе самостоятельной работы над конкретным техническим проектом.

Задачи раздела:

- Научить методам поиска решений изобретательских задач;
- Научить четко и точно излагать свои мысли и технические замыслы;
- Обучить самостоятельному проектированию и программированию устройства, которое решает практическую задачу;
- Развивать умение выстраивать гипотезу и сопоставлять с полученным результатом;
- Развивать способности работы индивидуально и в командах разного качественного и количественного состава группы;
- Развивать нестандартное, креативное мышление, а также поисковые навыки в решении прикладных задач;
- Содействовать саморазвитию в формировании успешных личных стратегий коммуникации и развитию компетенций при участии учеников в командной работе;
- Способствовать заинтересованности в самостоятельном расширении кругозора в области конструирования робототехнических систем.

Ожидаемые результаты:

- Умение самостоятельно спроектировать, сконструировать и запрограммировать устройство, решающее поставленную перед ним практическую задачу;

- Умение самостоятельно применять на практике методы поиска решений изобретательских задач;
- Развитие коммуникативных навыков, навыков самопрезентации и презентации проекта.

Раздел «Проектная лаборатория» призван дать возможность воспитанникам на основе полученных знаний разработать технический проект на платформе Arduino, пройти существующие этапы работы над проектом и представить свое решение в формате очной либо дистанционной защиты. Технический проект рассматривается как обобщение опыта усвоения текущего уровня, систематизирует знания, практические умения и навыки, а также способы творческой деятельности, полученные в ходе практических занятий, выполнения практических работ.

Данный раздел исполняет роль самостоятельной работы, технического практикума, который служит формой промежуточной аттестации и способом оценки уровня полученных знаний.

Воспитанники должны подготовить необходимую документацию по проекту: схемы, чертежи, программный код, описание проекта в виде презентации. Работа над проектом практическая, индивидуально либо в группах. Каждый воспитанник вправе выбирать, работать самостоятельно либо в группе. Проектная группа может включать до трех человек, обучающихся на разных уровнях программы. После успешной защиты, проект, включается в портфолио работ воспитанников, может быть доработан и рекомендован к участию в тематической выставке либо конкурсе в соответствии с планом воспитательной работы по программе. Таким образом, при успешном освоении образовательной программы, воспитанники будут иметь в своем портфолио уже как минимум 4 проекта на базе контролера Arduino.

Занятия по данному разделу проводятся в соответствии с календарным учебным графиком, не реже двух раза в год. Режим занятий 2 часа 2 раза в неделю, всего 32 часа, 2 календарных месяца.

Защита проекта представляет собой выступление с кратким сообщением (время выступления не ограничивается) о сути и результатах своей практической деятельности, с последующими ответами на вопросы. Может проходить в очном

либо дистанционном формате посредством Zoom, Viber, Discord и т.п. Для защиты проектов привлекаются сотрудники сетевого партнера программы в соответствии с договором о сотрудничестве. Состав и количество членов жюри определяется второй стороной. Критерии оценки творческих проектов и требования к содержанию и оформлению конкурсных работ изложены в (Приложение 7)

Поощрение участников осуществляется в соответствии с рейтинговой системой учреждения (Приложение 1).

Уровень «Arduino PRO»

1. Управление.

Работа с разделом осуществляется при помощи базы примеров и уроков по компонентам из набора GyverKIT https://kit.alexgyver.ru/tutorials-category/control/.

1.1. Arduino и радио модули 433 МГц.

Мотор с редуктором, сервопривод,

Смежная тема: Arduino и джойстик.

Проект: Платформа на радиоуправлении.

1.2. Arduino и мембранная клавиатура.

Смежная тема: сервопривод

Проект: Сейф с кодовым замком.

2. Индикация.

Работа с разделом осуществляется при помощи базы примеров и уроков по компонентам из набора GyverKIT https://kit.alexgyver.ru/tutorials-category/indication/.

2.1. Arduino и адресная светодиодная лента.

Смежная тема: Arduino и кнопка.

Проект: Новогодняя гирлянда.

2.2. Arduino и OLED дисплей.

Смежная тема: Arduino и часы RTC DS3231, Arduino и кнопка.

Проект: Электронные часы.

3. Датчики.

Работа с разделом осуществляется при помощи базы примеров и уроков по компонентам из набора GyverKIT https://kit.alexgyver.ru/tutorials-category/sensors/.

3.1. Arduino и датчик препятствия.

Смежная тема: Мотор с редуктором, Драйвер мотора.

Проект: Автомобиль для езды по черной линии.

3.2. Arduino и микрофон.

Смежная тема: Arduino и матрица MAX7219.

Проект: Анализатор спектра.

3.3. Arduino и акселерометр MPU6050.

Смежная тема: Сервоприводы.

Проект: Стедикам для телефона.

4. Моторы, коммутация.

Работа с разделом осуществляется при помощи базы примеров и уроков по компонентам из набора GyverKIT https://kit.alexgyver.ru/tutorials-category/output/.

4.1. Arduino и шаговый мотор.

Смежная тема: Arduino и энкодер, Arduino и часы RTC DS3231, Arduino и OLED дисплей

Проект: Автоматические жалюзи.

4.2. Arduino и реле.

Практика: Управление нагрузкой 220Вольт.

5. Умный дом

«Умный дом» — система управления и автоматизации инженерными системами дома. Данная система может управлять отоплением, вентиляцией, включением света, а также выполнять функции сигнализации о несанкционированном проникновении в дом или утечке воды на пол. Все эти функции реализуются при помощи специальных датчиков, подключенных к хабу

(устройству управления). Хаб обрабатывает сигналы от датчиков и отправляет команды устройствам исполнителям.

Целью раздела является практическая разработка модулей системы «умный дом», а так же разработка и исследование алгоритмов системы, позволяющих увеличить экономию ресурсов, обезопасить жизнь человека и улучшить комфорт проживания.

Для наглядности, разработка мини-проектов осуществляется на специально изготовленном стенде, представляющем макет жилого дома в масштабе.

5.1. Arduino и ИК пульт.

Смежная тема: Arduino и MOSFET транзисторы.

Проект: Светодиодный контроллер.

5.2. Arduino и считыватель RFID RC522.

Смежная тема: Сервопривод.

Проект: Домофон.

5.3. Arduino и метеодатчик BME280/BMP280.

Смежная тема: Arduino и OLED дисплей.

Проект: Домашняя метеостанция.

5.4. Arduino и датчик HTU21D.

Смежная тема: Arduino и сервопривод.

Проект: Предсказатель погоды.

5.5. Arduino и ИК датчик движения.

Смежная тема: Arduino и реле, Мотор с редуктором.

Проект: Система открывания дверей.

5.6. Arduino и водяная помпа.

Смежная тема: Arduino и датчик влажности почвы, Arduino и энкодер, Arduino и дисплей LCD1602.

Проект: Система автополива растений.

6. Интернет вещей

Интернет вещей (англ. Internet of things, IoT) – сеть физических устройств, в которые встроены датчики, софт и другие технологии для сбора, обработки и обмена информацией с другими умными устройствами и IoT-платформами.

Целью раздела является практическая разработка модулей обменивающихся друг с другом информацией посредством технологий беспроводной связи.

Так же как и в предыдущем разделе, разработка мини-проектов осуществляется на специально изготовленном стенде, представляющем макет жилого дома в масштабе.

Разработка ведется с использованием следующих Android приложений: RemoteXY, Virtuino, Blynk, IoT Wi-Fi контроллер и т.п. Использование подобных приложений также значительно упрощает образовательный процесс, осуществляемый при помощи дистанционных технологий, так как разработку IoT устройств можно вести даже не выходя из дома.

Работа с разделом осуществляется при помощи базы примеров и уроков по компонентам из набора GyverKIT https://kit.alexgyver.ru/tutorials/wemos-basic/.

6.1. Плата Wemos Mini (esp8266).

Подключение к роутеру. Создание точки доступа. Настройка для Wemos (esp8266), библиотеки.

Практика: Подключение esp8266 к сети WiFi. Обмен данными между платами.

6.2. WiFi розетка.

6.3. Датчик протечек.

6.4. Датчик температуры и влажности.

6.5. Концевики дверей и окон.

6.6. Охранная сигнализация.

6.7. WiFi сервер «умного дома».

7. Проектная лаборатория.

См. **Уровень «Arduino BASIC»** пункт 7.

Методические материалы

Формы обучения и виды занятий

Форма организации занятий: очная, с применением дистанционных технологий.

Обучение очное с возможностью дистанционного обучения. Виды занятий: лекции, практические занятия, зачетные занятия.

Формы работы

- практическое занятие;
- занятие – соревнование;
- педагогический коучинг;
- workshop (рабочая мастерская - групповая работа, где все участники активны и самостоятельны);
- консультация;
- мозговой штурм;
- фронтальная;
- индивидуальная;
- групповая.

Виды учебной деятельности:

- объяснение и интерпретация наблюдаемых явлений;
- выполнение практических работ;
- публичное выступление;
- решение инженерных задач;
- систематизация знаний;
- поиск информации в сети Интернет;
- выполнение тестовых заданий;
- наблюдение за демонстрациями преподавателя;
- анализ графиков, таблиц, схем;
- изучение устройства механизмов и принципа их действия;
- анализ производственных ситуаций, ситуативных задач;

- изучение последовательности выполнения операций;
- моделирование и конструирование;
- Выполнение работ практикума.

Конечно, этим списком многообразие видов учебной деятельности не исчерпывается. Задача педагога - искать и находить новые, более эффективные в современной информационной образовательной среде виды деятельности обучающихся, ориентированные на достижение современных образовательных результатов. Это один из критериев того, что образовательная программа ежегодно должна повергаться изменениям.

Важно понимать, что вся учебная деятельность должна быть представлена как система неких учебных задач. Они даются в определённых учебных ситуациях и предполагают определённые учебные действия - предметные, контрольные, вспомогательные и т. д. Учебная ситуация рассматривается как организация учебной деятельности, в которой обучаемые (возможно, при помощи преподавателя) не только обнаруживают предмет своего действия, но и решают конкретные задачи, направленные на выработку ключевых компетенций (сравнение, установление взаимосвязей, определение причин и следствий, решение противоречий и др.).

В ходе отбора видов учебной деятельности педагог может опираться на следующую классификацию типов учебных ситуаций для построения учебного процесса в информационной образовательной среде:

- ситуация-проблема - прототип реальной проблемы, которая требует оперативного решения (вырабатывается умение находить оптимальное решение);
- ситуация-оценка - прототип реальной ситуации с готовым предполагаемым решением, которое следует оценить, а затем предложить своё адекватное решение;
- ситуация-тренинг - образец стандартной или другой ситуации (предлагается описать или решить ситуацию).

Кроме того, на учебных занятиях возможны:

- *классическая ситуация* - даётся чёткое описание ситуации, взятой из практики или искусственно сконструированной; обучающиеся должны самостоятельно вычленить из её контекста вопрос, по поводу чего им следует принять решение;
- *живая ситуация* - берётся событие из жизни обучающегося, на производстве, принятое решение неизвестно, его надо найти, а развитие действия описать в той последовательности, в которой оно происходило;
- *действия по алгоритму, по инструкции, по стандарту* - обучающимся предлагаются ситуация и нормативный документ, в соответствии с которым должно быть принято решение.
- *ситуация-иллюстрация* - прообраз жизненной ситуации, которая включается в качестве факта в лекционный материал (визуальная образная ситуация, представленная в виде игры).

Основными принципами обучения являются:

1. *Научность*. Этот принцип предопределяет сообщение обучаемым, только достоверных, проверенных практикой сведений, при отборе которых учитываются новейшие достижения науки и техники.

2. *Доступность*. Предусматривает соответствие объема и глубины учебного материала уровню общего развития учащихся в данный период, благодаря чему, знания и навыки могут быть сознательно и прочно усвоены.

3. *Связь теории с практикой*. Обязывает вести обучение так, чтобы обучаемые могли сознательно применять приобретенные ими знания на практике.

4. *Воспитательный характер обучения.* Процесс обучения является воспитывающим, ученик не только приобретает знания и нарабатывает навыки, но и развивает свои способности, умственные и моральные качества.

5. *Сознательность и активность обучения.* В процессе обучения все действия, которые отрабатывает ученик, должны быть обоснованы. Нужно учить, обучаемых, критически осмысливать, и оценивать факты, делая выводы, разрешать все сомнения с тем, чтобы процесс усвоения и наработки необходимых навыков

происходили сознательно, с полной убежденностью в правильности обучения. Активность в обучении предполагает самостоятельность, которая достигается хорошей теоретической и практической подготовкой и работой педагога.

6. *Наглядность*. Объяснение техники сборки робототехнических средств на конкретных изделиях и программных продукта. Для наглядности применяются существующие видео материалы, а также материалы своего изготовления.

7. *Систематичность и последовательность*. Учебный материал дается по определенной системе и в логической последовательности с целью лучшего его освоения. Как правило, этот принцип предусматривает изучение предмета от простого к сложному, от частного к общему.

8. *Прочность закрепления знаний, умений и навыков*. Качество обучения зависит от того, насколько прочно закрепляются знания, умения и навыки учащихся. Не прочные знания и навыки обычно являются причинами неуверенности и ошибок. Поэтому закрепление умений и навыков должно достигаться неоднократным целенаправленным повторением и тренировкой.

9. *Индивидуальный подход в обучении.* В процессе обучения педагог исходит из индивидуальных особенностей детей и, опираясь на сильные стороны ребенка, доводит его подготовленность до уровня общих требований.

Методы обучения:

Традиционные:

- объяснительно-иллюстративный метод (лекция, рассказ, работа с литературой и т.п.);
- репродуктивный метод;
- метод проблемного изложения;
- частично-поисковый (или эвристический) метод;
- исследовательский метод.

Современные:

- метод проектов:
- метод обучения в сотрудничестве;
- метод портфолио;

- метод взаимообучения.

Особенности организации образовательного процесса

-очно, дистанционно в условиях сетевого взаимодействия.

Методы обучения (словесный, наглядный практический; объяснительно-иллюстративный, репродуктивный, частично-поисковый, исследовательский проблемный; игровой, дискуссионный, проектный и др.) и воспитания (убеждение, поощрение, упражнение, стимулирование, мотивация и др.);

Формы организации образовательного процесса: групповая, возможна организация индивидуально-группового обучения, а также дистанционно.

Формы организации учебного занятия - беседа, выставка, диспут, защита проектов, игра, конкурс, конференция, круглый стол, лекция, мастер-класс, открытое занятие, практическое занятие, презентация, соревнование, фестиваль, экскурсия**.**

Педагогические технологии:

- технология разноуровневого обучения;
- технология проблемного обучения;
- технология дистанционного обучения;
- технология игровой деятельности;
- технология проектной деятельности;
- здоровьесберегающая технология.

Формы аттестации и контроля

Оценочные материалы – пакет диагностических методик, позволяющих определить достижение учащимися планируемых результатов

Систематизированные материалы наблюдений (оценочные листы, материалы и листы наблюдения и т.п.) за процессом овладения знаниями, умениями, навыками, компетенциями, предусмотренными образовательной программой

Для успешной реализации программы предлагается систематическое отслеживание результатов учебной деятельности воспитанников.

В программе предусмотрена безоценочная система оценивания. Однако существует уникальная рейтинговая система, как дополнительный способ мотивации воспитанников (Приложение 1). По мере освоения программного материала предусмотрена организация проверочных работ, тестов, активно могут быть использованы сервисы тестирования от Google и VK. На ознакомительный модуль уровня «START» принимаются все желающие, по окончанию освоения раздела «Основы электротехники», проходят итоговую аттестацию (Приложение 9). При желании обучающегося заниматься на уровне «Arduino BASIC», предлагается пройти процедуру тестирования стартового уровня Приложение 2 и Приложение 9. На уровне «Arduino BASIC» воспитанники совершенствуют свои умения и навыки полученные ранее, изучают синтаксис языка С++, знакомятся с приложениями для работы с контроллером Arduino. Промежуточный контроль осуществляется при помощи тестирования (Приложение 10). В качестве промежуточного и итогового контроля используется раздел «Проектная лаборатория». После прохождения которого, воспитанники выполняют защиту своего творческого проекта. Аналогичным образом организуются формы контроля на уровне «Arduino PRO».

Система оценивания по Программе предусматривает следующую организацию проверочных работ:

- защита творческих проектов;
- участие в конкурсах;
- участие во внутренних соревнованиях, районных и областных;
- тестирование;
- зачет.

Результативность освоения программы определятся в несколько этапов.

Входной контроль: собеседование. Задача контроля - определить начальную подготовку, желание заниматься в этом направлении, личные качества ребенка и др.

Текущий контроль: опрос, соревнование, наблюдение, анализа результатов участия обучающихся в конкурсах и соревнованиях.

Подведение итогов реализации программы: соревнования или презентация (защита) творческого проекта.

Результаты фиксируются в диагностической карте. (Приложение 8).

Требования для перевода обучающихся на следующий уровень программы

Основанием для перевода учащегося на след уровень освоения является:

- успешное освоение теоретического и практического материала образовательной программы:
- успешное прохождение промежуточной аттестации в соответствии с установленными формами контроля обучающихся;
- успешное выполнение итоговой аттестация в соответствии с установленными формами контроля обучающихся;

Выпуск обучающихся, предусмотрен только после выполнения выпускной работы, её защиты и сдачи материала.

Учебно-методическое обеспечение и информационное обеспечение программы

1. Личный кабинет педагога на сайте «Инфоурок»: https://infourok.ru/user/belkin-dmitriy-vladimirovich;
2. База идей педагога на сайте «Pinterest»: https://www.pinterest.ru/tm_belkin/_saved/;
3. База знаний Амперки: http://wiki.amperka.ru/
4. Методическая база работы с набором Ардуино GyverKIT: https://kit.alexgyver.ru/;
5. Канал Alex Gyver: https://www.youtube.com/c/AlexGyverShow;
6. Официальный сайт платформы: https://www.arduino.cc/;
7. Канал «Заметки Ардуинщика»: https://www.youtube.com/c/ЗаметкиАрдуинщика;

8. База уроков Arduino и робототехники: https://alexgyver.ru/lessons/.

Условия реализации программы

Материально-техническое обеспечение программы

1. Набор Ардуино GyverKIT PRO (10 шт.).
2. Компьютерный класс, площадью не менее 48м2 с обязательным доступом в сеть Интернет.
3. Ноутбуки или планшеты – 10 шт.
4. Колонки или иная акустическая система.
5. Мультимедийный проектор и экран (или интерактивная доска).
6. 3D принтер Hercules – 1 шт.
7. Пластик PLA – 1 кг.

Кадровое обеспечение реализации программы

Педагог дополнительного образования, имеющий высшее профессиональное образование технической направленности без предъявления к стажу педагогической работы.

Организация мероприятий с обучающимися и родителями вне учебного плана

В соответствии с программой воспитательной работы МАУ ДО «Центра внешкольной работы» на 2021-2022 год, в данном разделе представлен план традиционных мероприятий, организуемых для обучающихся и их родителей за рамками учебного плана для организации досуга, формирования ценностных ориентиров, профилактической работы, участия в конкурсной и соревновательной деятельности и т. д. Сроки проведения мероприятий и условия участия в них конкретизируются непосредственно в течение учебного года.

https://cvr-omut.tmn.muzkult.ru/media/2021/07/22/1303684579/PROGRAMMA_VOSP.RABOTY_CZVR_21-22gg.pdf

№ п/п	Наименование (название) мероприятия	Реализация компонента	Форма проведения мероприятия	Сроки проведения
1	День открытых дверей МАУ ДО «Центра внешкольной работы»	Социокультурное и медиакультурное воспитание	Массовое мероприятие, игровая программа.	01.09
2	«А вот и мы!»	Воспитание здорового образа жизни	Внутри объединения. Спортивная игровая программа, направленная на сплочение коллектива, воспитание толерантности.	02.09 – 10.09
3	Родительское собрание	-	Знакомство с родителями. Ознакомление с деятельностью объединения	10.09 – 14.09
4	Областной чемпионат по робототехнике и программированию на Кубок Губернатора Тюменской области	Техническое воспитание	Робототехнические соревнования	15.09 – 20.09
5	Профилактическая акция «Внимание дети!» «Безопасный путь домой»	Правовое воспитание и культура безопасности.	Массовое мероприятие. Участие в месячнике безопасности детей: профилактическое занятие с воспитанниками 1-4-х классов по теме «Безопасный путь домой». Оформление маршрутного листа, участие воспитанников в викторине по ПДД.	25.09 – 30.09
6	«День Матери»	Воспитание семейных ценностей	Массовое мероприятие. Конкурсно- развлекательная программа для родителей с детьми	29.12
7	Новогодние утренники	Социокультурное и медиакультурное воспитание	Обучающиеся, отличившиеся в освоении образовательной программы, имеют право на посещение детских утренников в МАУ ДО «Центр внешкольной работы».	25.12 – 30.12

8	Новогодний «РобоКвест»	Техническое воспитание	Робототехнические соревнования. Проводится в новогодние каникулы. Принять участие может любой желающий. Воспитанники выступают в роли судей, наставников и организаторов соревнований.	05.01 – 10.01
9	«Есть такая профессия – Родину защищать»	Гражданско-патриотическое воспитание	Внутригрупповая работа. Выставка вооружения	23.02
10	8 Марта	Воспитание семейных ценностей	Игровая программа для родителей и детей. Организация семейного досуга.	08.03
11	Окружная выставка-конкурс технического творчества и робототехники «Техническое творчество – дорога в будущее» с. Голышманово	Техническое творчество	Выставка представлена всеми видами технического творчества. Воспитанники проводят презентацию и защиту проектов по робототехнике.	20.03 – 25.03
13	Областная выставка научно-технического творчества и робототехники г. Тюмень	Техническое творчество	Выставка представлена всеми видами технического творчества. Воспитанники проводят презентацию и защиту проектов по робототехнике.	10.04 – 20.04
14	Межрайонные робототехнические соревнования «РобоКвест»	Техническое воспитание	Робототехнические соревнования по робототехнике на специальном полигоне	25.04 – 30.04
15	«Мой край»	Краеведческое воспитание	Внутригрупповая работа. Экскурсия в краеведческий музей Омутинского района.	05.05 – 10.05
16	Районные робототехнические соревнования «РобоОмут»	Техническое воспитание	Робототехнические соревнования по соревновательной робототехнике	15.05 -20.05
17	«Ура! Каникулы!»	Социокультурное и медиакультурное воспитание	Внутригрупповая работа. Игровое мероприятие. Профилактика детских правонарушений в летний период	25.05 – 30.05
18	День защиты детей	Социокультурное и медиакультурное воспитание	Игровая программа для детей. Организация содержательного досуга.	01.06
19	Смена в лагере	Социокультурное и медиакультурное воспитание	Летний лагерь дневного пребывания «Калейдоскоп приключений»	06.06 – 28.06.

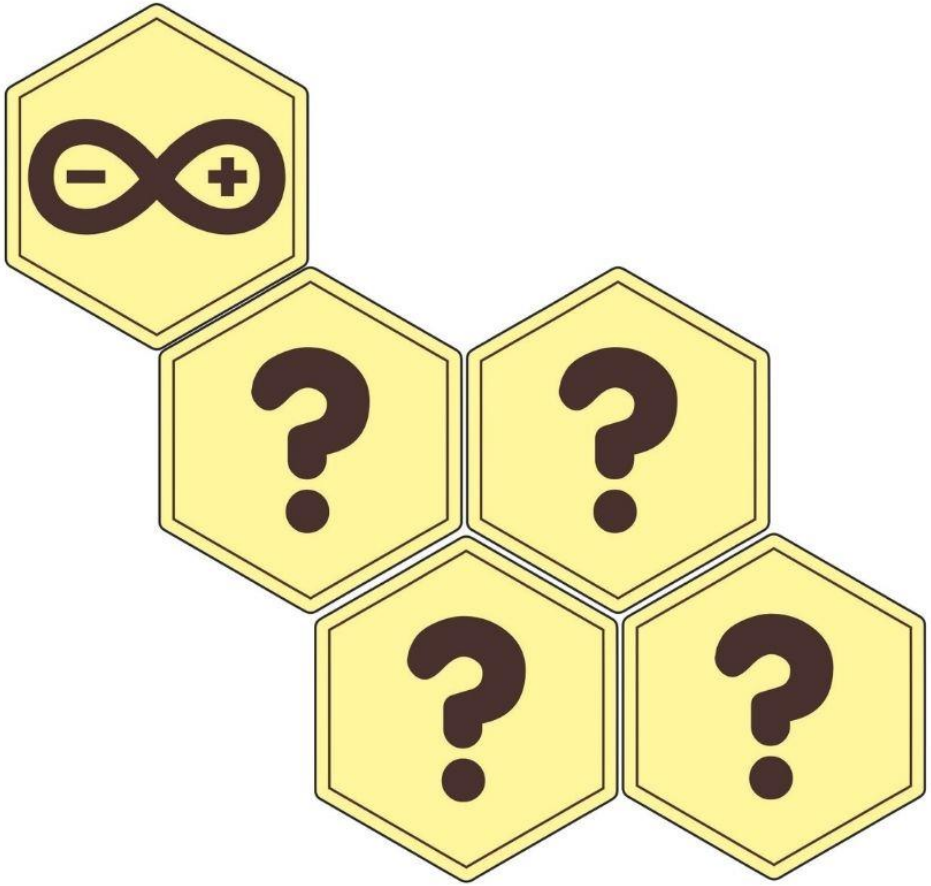
Литература и источники

1. Викентьев, И.Л., Кайков, И.К. Лестница идей: Основы ТРИЗ в примерах и задачах. [Текст]\ И.Л.Викентьев, И.К.Кайков,— Новосибирск, 1992.
2. Петин – Электроника. Проекты с использованием Arduino.
3. Джереми Блюм – Изучаем Arduino. Инструменты и методы технического волшебства.
4. Информационно-методический сборник регионального модельного центра «Реализация приоритетного проекта «Доступное образование для детей» На территории Тюменской области», Тюмень 2017г. Выпуск №1.
5. Максим Иванов (aka e-maxx). Сборник алгоритмов на C++.
6. Официальный сайт кружка робототехники "Робикс" https://robx.org/
7. Саймон Монк – Программируем Arduino. Профессиональная работа со скетчами.
8. Теро Карвинен и др. – Делаем сенсоры.
9. Улли Соммер – Электроника. Программирование Arduino.
10. Юрий Менщиков – Ардуино на пальцах.
11. Юрий Ревич. Занимательная электроника.
12. Francis Perea – Arduino Essentials (англ.).

Рейтинговая система

"ArduЛАБ"

Значки достижений



Выдаются за успешное выполнение заданий по указанной теме.

В случае дистанционного обучения, могут выдаваться за выполнение "Домашнего задания" либо самостоятельной работы по теме.

"Валюта" инженеров Metcoin (Металлолом)

Как заработать:

В ходе занятия

За успешное выполнение особо сложных либо дополнительных заданий

За участие

За личное участие в выставках либо соревнованиях, за инженерные идеи

В качестве поощрения

За примерное поведение, качественный и творческий подход к выполнению поставленной задачи
Выдается на усмотрение педагога.

За помощь

За оказание помощи другу или педагогу во время занятия

по легенде, только настоящий робототехник сможет смастерить из любого металлолома все что угодно

По окончании учебного года, металлолом можно обменять:

Электроника

Мерч

Собственное производство

Составляется рейтинг 3-х наиболее успешных участников

Приложение 2

Итоговое тестирование
Ознакомительный модуль

1. Расставьте по порядку три основных закона робототехники.	
2	Робот должен повиноваться всем приказам, которые даёт человек, кроме тех случаев, когда эти приказы противоречат Первому Закону
3	Робот должен заботиться о своей безопасности в той мере, в которой это не противоречит Первому или Второму Законам
1	Робот не может причинить вред человеку или своим бездействием допустить, чтобы человеку был причинён вред

2. Название, под которым подразумевается человекоподобный робот.	
+	Андроид
	Робот
	Машина
	Механизм

3. Каких видов робототехники не существует?	
	Промышленная
+	Предшкольная
	Военная
+	Строительная
	Космическая

4. Робототехника – это…	
	(от техника; англ. texniks) наука, занимающаяся разработкой технических систем
+	(от робот и техника; англ. robotics) прикладная наука, занимающаяся разработкой автоматизированных технических систем
	(от роботать; англ. roboticsistems) наука, занимающаяся разработкой технических систем

5.	Что первым делом учитывается при разработке робота с точки зрения электроники?
	Квалификация пользователя
+	Напряжение в цепи
	Квалификация программиста
	Формат данных, передаваемых с датчиков

6.	Какие признаки подскажут, что для этой работы нужен робот?
+	Экстремальные условия и труднодоступность рабочих объектов
	Низкая квалификация сотрудников
	Использование необычных инструментов

7.	Что помогло бы улучшить грузоподъемность рабочих на заводе?
	RPA
	Роверы
	Манипуляторы
+	Экзоскелеты

8.	Какой элемент связывает действия робота и показания датчиков между собой?
	Система датчиков
	Исполняющее устройство
+	Алгоритм

9.	Что помогает новому роботу-пылесосу в построении карты?
	База данных с расположением комнат и препятствий
	Заполненный граф на основе данных всех роботов-пылесосов
+	Построение графов при непосредственном прохождении комнат
	GPS

10. Выберите правильное определение робота:	
+	Автоматическое или автоматизированное устройство, включающее в себя систему датчиков, контроллер и исполняющее устройство, выполняющее некоторые операции по заранее заданной программе, самостоятельно или по команде человека.
	Система, оснащенная искусственным интеллектом для принятия решения.
	Механическое устройство, выполняющее операции в автоматическом режиме.
	Системы климат-контроля.

11. Что обязательно понадобится для того, чтобы роботизировать террариум?	
+	Датчики влажности и температуры, контроллер и система нагрева
	Датчик движения, датчик света и видеокамера
	Датчик температуры, сервопривод
	Термопара и датчик тепла

12. У вас есть робот-манипулятор, задача которого — раскладывать в хранилище бумажные документы. Хранилище состоит из двух комнат. Чем должен обладать новый робот, чтобы успешно выполнять работу?	
	Датчик цвета и система питания на солнечной энергии
+	Система перемещения и шарнир, позволяющий перемещать рычаг манипулятора по трем осям

13. Что сегодня не умеют делать роботы в сфере подбора сотрудников?	
	Отбирать резюме по нужным критериям
+	Искать и нанимать топ-менеджеров
	Отвечать на вопросы кандидатов

14. Выполнение каких задач пока еще нельзя передать роботам?	
	Исследования вулканов и поверхности морского дна
	Выращивание семян на космической станции
	Заполнение и обработка данных из заявлений

+	Назначение медицинских препаратов и диагностика состояния больного

15. Какое название имеет автоматическая машина, состоящая из исполнительного устройства в виде манипулятора?	
	Мобильный робот
+	Манипуляционный робот
	Управляющий робот

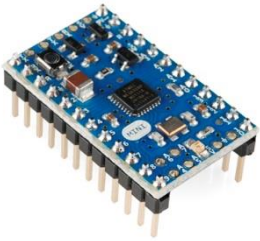
16. Сервомотор – это…	
	Устройство для определения цвета
	Устройство для хранения данных
+	Устройство для движения робота
	Устройство для проигрывания звука

17. Кем было придумано слово "робот"?	
+	Карел Чапек
	Йозеф
	Карел Чапек и Йозеф

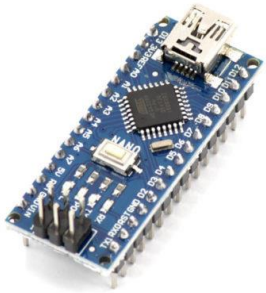
18. Что такое микроконтроллер?	
+	Очень маленькой компьютер, который запускает программу.
	Исполнительное устройство для перемещения робота.
	Устройство, от которого получает электрическую энергию робот.

19. Алгоритм – это...	
+	Описание последовательности действий (план), строгое исполнение которых приводит к решению поставленной задачи за конечное число шагов
	Область памяти, адрес которой можно использовать для осуществления доступа к данным

20. Соотнесите существующие платы Arduino

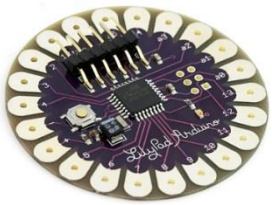


Arduino LilyPad



Arduino Mini

Arduino Micro



Arduino Nano

Arduino Mega

Arduino Uno

Вопрос на 2 балла

	21. Пешеход за 4 часа прошел 16 км. С какой скоростью двигался пешеход?
	4 км
+	4 км/ч
	12 км

Вопрос на 3 балла

	22. Длина окружности велосипедного колеса составляет 127,6 см укажите диаметр колеса в дюймах. Один дюйм = 2,54 см.
	40 дюймов
+	16 дюймов

	50 дюймов
	324 дюйма

Тест состоит из 22 вопросов.
Максимальное количество баллов – 25.
«Проходное» количество баллов – 18.

Приложение 3

УРОКИ ARDUINO

Данное пособие является сборником команд, операторов и функций (ШПАРГАЛКА), стандартных для языка C++ и *Arduino Wiring*. Пособие создано с целью иметь в одном месте список всех «инструментов» для работы с Arduino, чтобы всегда можно было его открыть и сразу найти нужное.

Ссылка для скачивания: https://disk.yandex.ru/i/WJYIfjNDYyBg1

Оглавление

Структура программы

// Однострочный комментарий (комментарии не занимают память)

/* Многострочный комментарий (комментарии не занимают память) ***/**

#include - подключить файл (библиотеку)

void setup() {} - всё находящееся внутри {} будет выполнено 1 раз при загрузке Ардуино

void loop() {} - всё находящееся внутри {} бесконечно повторяется из начала в конец

После каждого «действия» ставится точка запятой ;

Типы данных

Переменная объявляется вот таким образом: **<тип данных> <имя>;**

```
int my_val;              // объявить переменную my_val
```

Также можно сразу присвоить значение **<тип данных> <имя> = <значение>;**

```
int my_val = 2300;     // объявить переменную my_val и присвоить ей число 2300
```

Также можно объявить несколько переменных одного типа сразу

```
int my_val = 2300, my_val2, my_val4, lolkek = 5;          // объявить переменные
```

Название	Альтернативное название	Вес	Диапазон	Особенность
boolean		1 байт	0 или 1	Логическая переменная, может принимать значения **true** (1) и **false** (0)
char	int8_t	1 байт	-128… 127	Хранит номер символа из таблицы символов ASCII
byte	uint8_t	1 байт	0… 255	
int	int16_t	2 байта	-32 768… 32 767	
unsigned int	uint16_t	2 байта	0… 65 535	
word		2 байта	0… 65 535	То же самое, что unsigned int
long	int32_t	4 байта	-2 147 483 648… 2 147 483 647	- 2 миллиарда… 2 миллиарда
unsigned long	uint32_t	4 байта	0… 4 294 967 295	0… 4 миллиарда…
float		4 байта	-3.4028235E+38… 3.4028235E+38	Хранит числа с плавающей точкой (десятичные дроби). **Точность: 6-7 знаков**
double		4 байта		То же самое, что float

Особенности использования переменных

- Внимательно следите за значением, которое принимает переменная. Если значение превысит максимальное или принизит минимальное (выйдет из диапазона) для этого типа данных, то переменная сбросится в 0, а затем продолжит увеличение или уменьшение в том же направлении (например если присвоить byte значение 300, то она примет 300-255=45). Такую ошибку потом будет трудно отследить.
- Тип данных указывается при объявлении переменной ТОЛЬКО ОДИН РАЗ, далее переменная используется чисто по имени (обращение к переменной). При попытке сменить тип переменной (переобъявить переменную) вы получите ошибку. Но только в том случае, если переменная глобальная, либо когда локальная переобъявляется внутри функции, в которой она уже была объявлена.

Особенности float

- Присваивать только значение с точкой, даже если оно целое (10.0)
- Делить тоже только на числа с точкой, даже если они целые (переменная / 2.0)
- При делении целочисленного типа с целью получить число с плавающей точкой, писать (float) перед вычислением!
- Операции с числами типа float занимают гораздо больше времени, чем с целыми! Если нужна высокая скорость вычислений, лучше применять всякие хитрости, в стиле выполнения всех вычислений типом long, и результат уже переводить во float. Например вместо 3.25 вычислять в 100 раз большие числа, то есть 325

Типы переменных

Глобальная переменная – объявляется ВНЕ функций, например в самом начале скетча или между функциями. Обращаться к глобальной переменной (использовать её значение) можно использовать ВЕЗДЕ.

Локальная переменная – объявляется ВНУТРИ функции, и обращаться к ней можно только внутри этой функции.

Локальных переменных может быть несколько с одинаковым именем, но разными значениями. Это связано с тем, что локальная переменная выгружается из оперативной памяти микроконтроллера при выходе из функции.

Константы

const <тип> <имя> = <значение>; - объявить константу

```
const int my_val = 2300;                    // объявить константу my_val и присвоить ей число 2300
```

#define <имя> <значение> - объявить константу через define (точка запятой НЕ НУЖНА)

```
#define my_val  2300                        // определить константу my_val и присвоить ей число 2300
```

Константа, объявленная через #define, работает немного по-другому: на этапе компиляции кода указанное название ЗАМЕНЯЕТСЯ на указанное значение, и хранится во флэш-памяти МК.
При попытке сменить значение константы ПОСЛЕ её объявления, вы получите ошибку!

Математические операторы

+, -, *, / // сложить, вычесть, умножить...

pow(x, a); // возвести "x" в степень "a" (x^a), **pow** может возводить в дробную степень!

sq(x); // возвести число "x" в квадрат (x^2)

sin(x), cos(x), tan(x); // синус, косинус, тангенс

round(x); // математическое округление (если после запятой больше или равно 5, то округляем в большую сторону)

ceil(x); // округлить в бОльшую сторону

floor(x); // округлить в меньшую сторону

x += a; // прибавить "a" к "x"

x -= a; // вычесть "a" из "x"

x *= a; // домножить "x" на "a"

Операторы библиотеки Serial

Serial - объект библиотеки Serial для работы с последовательным портом (COM портом)
Serial.begin(<скорость>); - открыть порт
Serial.begin(9600); // открыть порт на 9600 БОД
ВНИМАНИЕ! Скорость, установленная в begin(), должна быть равна скорости монитора порта (в самом мониторе правый нижний угол). Иначе в выводе получите крокозябры!

Serial.print(); // вывод в порт. Переменные и цифры напрямую, текст – в кавычках " "

Serial.println(); // вывод с переводом строки

Serial.println(val, n); // вывод *переменной val (типа float)* с n числом знаков после запятой

Serial.println(val, <базис>); // вывод с указанным базисом:

- **DEC** - десятичный (человеческие числа)
- **HEX** - 16-ричная система
- **OCT** - 8-ричная система
- **BIN** - двоичная система

Данные с компьютера попадают в буфер с объёмом 64 байта, и ждут обработки

Serial.available(); // проверить буфер на наличие входящих данных

Serial.read(); // прочитать входящие данные в символьном формате! Согласно ASCII

Serial.read() - '0'; // прочитать данные в целочисленном формате. По одной цифре!
Serial.parseInt(); // прочитать данные в целочисленном формате. Число целиком!
Serial.parseFloat(); // прочитать данные в формате с плавающей точкой. Число целиком!
Serial.setTimeout(); // поставить таймаут для parseInt и parseFloat. Можно поставить около 50
Serial.flush(); // очистить буфер порта

Код		Код		Код		Код		Код		Код		Код		Код	
32	пробел	44	,	56	8	68	D	80	P	92	\	104	h	116	t
33	!	45	-	57	9	69	E	81	Q	93	]	105	i	117	u
34	"	46	.	58	:	70	F	82	R	94	^	106	j	118	v
35	#	47	/	59	;	71	G	83	S	95	_	107	k	119	w
36	$	48	0	60	<	72	H	84	T	96	`	108	l	120	x
37	%	49	1	61	=	73	I	85	U	97	a	109	m	121	y
38	&	50	2	62	>	74	J	86	V	98	b	110	n	122	z
39	'	51	3	63	?	75	K	87	W	99	c	111	o	123	{
40	(	52	4	64	@	76	L	88	X	100	d	112	p	124	|
41	)	53	5	65	A	77	M	89	Y	101	e	113	q	125	}
42	*	54	6	66	B	78	N	90	Z	102	f	114	r	126	~
43	+	55	7	67	C	79	O	91	[	103	g	115	s	127	⌂

Условный оператор if

if () {} - условный оператор, проверяет истинность в () и выполняет код в {}, если оно верно

```
if () {          // проверяет условие, если верно,
// выполняет эту часть кода

} else {         // если неверно

// выполняет вот эту часть кода
```

```
if () {          // проверяет условие, выполняет если верно
} else if () {   // если неверно, проверяет новое

} else if () {   // если неверно, проверяет новое

} else if () {   // если неверно, проверяет новое

} else {         // если неверно, выполняет то, что ниже
}
```

В условии может быть как логическое выражение (a > b), так и логическая переменная со значением true или false. Или обычная переменная со значением 1 или 0.

Операторы сравнения

a == b - если а равно b

a != b - если а не равно b

> - если а больше b (строго)

< - если а меньше b (строго)

>= - если а больше или равна b

<= - если а меньше или равна b

Логические операторы

&& - логическое И (одно условие И второе)

|| - логическое ИЛИ (либо одно, либо второе)

! – отрицание (например if (!val) - если val - ложь, т.е. 0)

Добавлено от WakeUp4L1fe

*Использовать **boolean (bool)** лучше со значениями **true** и **false**.*

*С++ приравнивает ноль к **false**, а любое число к **true**.*

К примеру:

```
bool x = 2;

if (x == 1) then {
Serial.println("истина");

} else {
Serial.println("ложь");
```

В порт выведется слово истина, хотя присваивали двойку!

Добавлено от Alexei Belousov

Небольшое добавление по условиям. Существует и укороченная запись условий

```
(a > b) ? c = true : c = false;

(a > b) ? Serial.println("a больше b") : Serial.println("b больше a");
```

Если А больше В то С равно истина, иначе С равно ложь..

Также имеет место запись присваивания переменной значения результата сравнения:

```
c = (a > b);      // c принимает true, если a > b. Или false, если нет
```

Оператор выбора switch.. case

```
switch (val) {          // рассматриваем переменную val

case 1:                 // если она равна 1, выполнить код здесь
break;
case 2:                 // если она равна 2, выполнить код здесь
break;
default:
break;

}
```

Добавлено от WakeUp4L1fe

Еще стоило указать возможность использования одновременно нескольких условий ***switch*** *оператора:*

```
switch (val) {
case 1:

case 2:

Serial.println("1 или 2");
break;

case 3:
Serial.println("3");
```

// то есть при пропуске ***break;*** *будут проверены и отработаны оба условия*

Функции задержек

delay() - задержка, в скобках указывается число миллисекунд (в 1 сек 1'000 миллисекунд). Максимальное значение типа *unsigned long* (4 байта), 4’294’967’295 мс, или около 1200 часов, или 50 суток.

delayMicroseconds() - задержка, в скобках указывается число микросекунд (в 1 сек 1'000'000 микросекунд). Максимальное значение 16'383 мкс, или 16 миллисекунд.

ИСПОЛЬЗОВАТЬ ЗАДЕРЖКИ НЕ РЕКОМЕНДУЕТСЯ! ОНИ ПОЛНОСТЬЮ "ВЕШАЮТ" СИСТЕМУ!

Функции таймера

millis() - возвращает количество миллисекунд, прошедших с момента включения МК.

- Макс. значение: 4’294’967’295 мс или 50 суток.
- Разрешение: 1 миллисекунда.

micros() - возвращает количество микросекунд, прошедших с момента включения МК.

- Макс. значение: 4’294’967’295 мкс или 70 мин
- Разрешение: 4 микросекунды

Пишем (long) перед умножением, чтобы программатор выделил нужное количество памяти для проведения операции! (для работы с большими числами).

Пример: ***(long)23*24*60*60*1000****, если хотим получить ПРАВИЛЬНЫЙ результат умножения в виде числа миллисекунд, равного 23 дням.*

На основе millis(), переменной unsigned long и условия можно сделать простой таймер, который будет срабатывать через указанные промежутки вермени.

```
unsigned long last_time;                // глобальная переменная!!!

void loop() {
 if (millis() - last_time >= 5000)  {    // если прошло больше 5000 мс (5 секунд)
 // код, который выполняется каждые 5 секунд
 last_time = millis();                  // сброс таймера
```

Режимы работы цифровых портов

- Аналоговые и цифровые пины могут работать как ВХОДЫ и как ВЫХОДЫ
- По умолчанию все пины работают КАК ВХОДЫ
- «Аналоговые» пины – это заблуждение. **Все пины цифровые**, но у некоторых есть АЦП (аналогово-цифровой преобразователь), у Нано и Уно эти пины подписаны как А. В то же время у Нано есть пины А6 и А7, к которым подключен ТОЛЬКО АЦП! То есть эти могут только измерять напряжение при помощи analogRead(), другие функции для них недоступны!

pinMode(pin, mode); - настроить порт

- **pin** - номер порта. Цифровые: 0 – 13. Аналоговые: 14 - 19, либо A0 - A5
- **mode** - режим работы порта
  - **INPUT** - вход, принимает сигнал
  - **OUTPUT** - выход, выдаёт 0 или 5 Вольт
  - **INPUT_PULLUP** - вход с подтяжкой к 5 В

Генерация цифрового сигнала

digitalWrite(pin, signal); - подать цифровой сигнал

- **pin** - номер порта. Цифровые: 0 – 13. Аналоговые: 14 - 19, либо A0 - A5
- **signal** - какой сигнал подаём
  - **LOW**, или 0 (ноль), или false - 0 Вольт
  - **HIGH**, или 1, или true - 5 Вольт

Чтение цифрового сигнала

digitalRead(pin); - прочитать цифровой сигнал

- **pin** - номер порта. Цифровые: 0 – 13. Аналоговые: 14 - 19, либо A0 - A5

Чтение аналогового сигнала

analogRead(pin); - прочитать аналоговый сигнал (оцифровать)

• **pin** - номер пина. Аналоговые: 0 - 7

Функция возвращает значение 0.. 1023 в зависимости от напряжения на пине от 0 до опорного напряжения. По умолчанию опорное напряжение равно напряжению питания, при питании от USB это около 4.7 Вольт. Можно выбрать источник опорного напряжения:

- analogReference(**DEFAULT**) – напряжение питания как опорное
- analogReference(**INTERNAL**) – встроенное опорное на 1.1 Вольт
- analogReference(**EXTERNAL**) – опорное напряжение подаётся на пин Aref

Изменение диапазона значений

map(val, min, max, new_min, new_max); - возвращает величину в новом диапазоне

- **val** - входная величина
- **min, max** - минимальное и максимальное значение на входе в map
- **new_min, new_max** – соответственно мин. и макс. значения на выходе

```
val = analogRead(0);                // читаем с пина (0-1023)
val = map(val, 0, 1023, 50, 100);   // переводим диапазон в 50-100
```

При изменении val от 0 до 1023 мы получим плавное изменение значения на выходе от 50 до 100

constrain(val, min, max); - ограничить диапазон переменной val до min и max

```
val = analogRead(0);                // читаем с пина (0-1023)
val = constrain(val, 50, 100);      // ограничим диапазон до 50-100
```

При изменении val от 0 до 50 получим постоянные 50, от 50 до 100 получим изменение от 50 до 100, а от 100 до 1023 получим постоянные 100, так как диапазон у нас именно **ограничен**.

Генерация ШИМ сигнала

analogWrite(pin, duty);

• **pin** – пин, На котором генерировать ШИМ

• **duty** – величина 0.. 255, соответствует скважности ШИМ 0.. 100%

ШИМ пины Arduino NANO, UNO: 3, 5, 6, 9, 10, 11

ШИМ пины Arduino MEGA: все до 13

Цикл for, «счётчик»

for (counter; condition; change) {} - цикл for

- **counter** – переменная счётчика, обычно создают новую «локальную», в стиле **int i = 0**;
- **condition** – условие, при котором выполняется цикл, например *«счётчик меньше 5»* **i < 5;**
- **change** – изменение, т.е. увеличение или уменьшение счётчика, например **i++**, **i--**, **i += 10**;

Пример:

```
for (byte i = 0; i < 100; i++) {  // счётчик от 0 до 99
Serial.println(i);                // вывести в монитор порта числа от 0 до 99
```

Цикл while, «с предусловием»

while (condition) {}

• **condition** – условие, при котором выполняется блок кода, заключённый в **{}**

Пример:

```
while (flag) {
// какой-то кусок кода, который выполняется, пока flag равен логической 1
}
```

Цикл do while, «с постусловием»

do {} while (condition) ;

• **condition** – условие, при котором выполняется блок кода, заключённый в **{}**

Пример:

```
do {
// какой-то кусок кода, который выполняется, пока flag равен логической 1
// но в отличие от предыдущего цикла, выполнится ХОТЯ БЫ ОДИН РАЗ, даже если Flag равен 0
} while (flag);
```

break – выход из цикла

Пример:

```
for (byte i = 0; i < 100; i++) {        // счётчик от 0 до 99
if (i > 50) break;                      // выйти из цикла, если i больше 50
```

```
Serial.println(i);          // вывести в монитор порта числа от 0 до 50!!!
}
```

continue – пропустить ход

Пример:

```
for (byte i = 0; i < 100; i++) {          // счётчик от 0 до 99
if (i > 50 && i < 60) continue;           // если i от 50 до 60, перейти в начало цикла
Serial.println(i);
```

Функция, которая ничего не берёт и не возвращает

void myFunction() {}

- **void** – слово, показывающее, что функция ничего не возвращает
- **myFunction** – название функции

Пример:

```
void myFunction() {          // задаём функцию
Serial.println("HELLO!");    // которая выведет в порт HELLO!
}
```

В другом месте программы вызываем функцию как **myFunction();** и в этом месте будет выслано слово HELLO! в порт.

Функция, которая берёт и ничего не возвращает

Пример:

```
void myFunction(int val) {   // создаём функцию, на вход которой подаётся одно число
Serial.println(val);         // и выводим его в порт
```

В другом месте программы вызываем функцию как **myFunction(50);** и в этом месте будет выслано в порт число 50.

Функция, которая берёт и возвращает

Пример:

```
int mySum(int val1, int val2) {   // создаём функцию, на вход которой подаётся два числа
return val1 + val2;               // возвращаем сумму чисел
```

• **return** – оператор, возвращающий результат

В другом месте программы вызываем функцию как **mySum(50, 70);** и функция вернёт результат 120. Как пример:

Serial.println(mySum(50, 70)); // в порт будет послана сумма чисел, т.е. 120

Генерируем случайные числа

randomSeed(value); // функция, задающая начало отсчёта генератору псевдослучайных чисел

• **value** – любое число типа *long*

random(min, max); // функция, возвращающая случайное число в диапазоне от **min** до **max - 1**

random(max); // то же самое, но возвращает от 0 до **max - 1**

Пример:

Serial.println(random(20)); // вывести в порт случайное число от 0 до 19

Как получать максимально случайную последовательность чисел?

randomSeed(analogRead(0)); // в качестве опорного числа взять сигнал с НЕПОДКЛЮЧЕННОГО НИКУДА аналогового пина

Создание массива

<тип данных> <имя массива>[<число элементов>];

<тип данных> <имя массива>[<число элементов>] = {элемент1, элемент2…};

Если не указываются элементы, то обязательно нужно указать размер массива, чтобы под него выделилось место в памяти. Размер можно не указывать в том случае, если сразу указываются все элементы. Примеры:

int myInts[6];

int myPins[] = {2, 4, 8, 3, 6};

int mySensVals[6] = {2, 4, -8, 3, 2};

Чтение – запись

Главное помнить, что нумерация элементов НАЧИНАЕТСЯ С НУЛЯ!

myArray[5] = 10; // присвоить пятому элементу число 10

if (myArray[5] == 20) ….. // если элемент массива под номером 5 равен 20…

Пример. Забивка массива случайными числами

```
byte myArray[50];                 // создать массив myArray на 50 ячеек
for (byte i = 0; i < 50; i++) {   // счётчик от 0 до 49
myArray[i] = random(100);         // присвоить случайное число от 0 до 99 элементам массива под номерами 0.. 49
```

Аппаратные прерывания

attachInterrupt(pin, function, state); - подключить прерывание

detachInterrupt(pin); - отключить прерывание

- **pin** – пин прерывания, для NANO и UNO это пины D2 и D3, соответствуют номерам 0 и 1
- **function** – название функции, которая будет вызвана при срабатывании прерывания
- **state** – режим обработки, их несколько:
  - **LOW** - вызывает прерывание, когда на порту LOW
  - **CHANGE** - прерывание вызывается при смене значения на порту, с LOW на HIGH инаоборот
  - **RISING** - прерывание вызывается только при смене значения на порту с LOW на HIGH
  - **FALLING** - прерывание вызывается только при смене значения на порту с HIGH на LOW

Пример:

Кнопка подключена к D2 и GND

```
void setup() {
 pinMode(2, INPUT_PULLUP);                  // пин D2 подтянут к питанию
 attachInterrupt(0, myInterrupt, FALLING);  // подключить прерывание на пин D2, обрабатывать при падении сигнала и вызывать функцию myInterrupt
}

void myInterrupt() {                        // функция обработчика прерываний
 Serial.println("INTERRUPT!");              // при срабатывании вывести в порт слово INTERRUPT
}
```

Приложение 4

Блокнот Arduino-программиста

Оглавление

структура

Базовая структура программы для Arduino довольно проста и состоит, по меньшеймере, из двух частей. В этих двух обязательных частях, или функциях, заключён выполняемый код.

```
void setup()
{
  statements;
}

void loop()
{
  statements;
}
```

Где setup() — это подготовка, а loop() — выполнение. Обе функции требуются для работы программы.

Перед функцией setup - в самом начале программы, обычно, идёт, объявление всех переменных. setup - это первая функция, выполняемая программой, и выполняемая только один раз, поэтому она используется для установки режима работы портов (pinMode()) или инициализации последовательного соединения

Следующая функция loop содержит код, который выполняется постоянно — читаются входы, переключаются выходы и т.д. Эта функция — ядро всех программ Arduino и выполняет основную работу.

setup()

Функция setup() вызывается один раз, когда программа стартует. Используйте её для установки режима выводов или инициализации последовательного соединения. Она должна быть включена в программу, даже если в ней нет никакого содержания.

```
void setup()
{
  pinMode(pin, OUTPUT);       //  устанавливает 'pin' как выход
}
```

loop()

После вызова функции setup() – управление переходит к функции loop() , которая делает в точности то, что означает её имя — непрерывно выполняется, позволяя

```
void loop()
{
  digitalWrite(pin, HIGH);    // включает 'pin'
  delay(1000);                // секундная пауза
  digitalWrite(pin, LOW);     // выключает 'pin'
  delay(1000);                // секундная пауза
}
```

программе что-то изменять, отвечать и управлять платой Arduino.

функции

Функция — это блок кода, имеющего имя, которое указывает на исполняемый код, который выполняется при вызове функции. Функции void setup() и void loop() уже обсуждались, а другие встроенные функции будут рассмотрены позже.

Могут быть написаны различные пользовательские функции, для выполнения повторяющихся задач и уменьшения беспорядка в программе. При создании функции, первым делом, указывается тип функции. Это тип значения, возвращаемого функцией, такой как 'int' для целого (integer) типа функции. Если функция не возвращает значения, её тип должен быть void. За типом функции следует её имя, а в скобках параметры, передаваемые в функцию.

```
type functionName(parameters)
{
  statements;
}
```

Следующая функция целого типа delayVal() используется для задания значения паузы в программе чтением значения с потенциометра. Вначале объявляется локальная переменная v, затем v устанавливается в значение потенциометра, определяемое числом между 0 — 1023, затем это значение делится на 4, чтобы результирующее значение было между 0 и 255, а затем это значение возвращается в основную программу.

```
int delayVal()
{
  int v;                      // создаём временную переменную 'v'
  v  = analogRead(pot);       // считываем значение с потенциометра
  v /= 4;                     // конвертируем 0 - 1023 в 0 - 255
  return v;                   // возвращаем конечное значение
}
```

{} фигурные скобки

Фигурные скобки (также упоминаются как просто «скобки») определяют начало и конец блока функции или блока выражений, таких как функция void loop() или выражений (statements) типа for и if.

```
type function()
{
  statements;
}
```

За открывающейся фигурной скобкой { всегда должна следовать закрывающаяся фигурная скобка }. Об этом часто упоминают, как о том, что скобки должны быть «сбалансированы». Несбалансированные скобки могут приводить к критическим, неясным ошибкам компиляции, вдобавок иногда и трудно выявляемым в больших программах.

Среда разработки Arduino, включает возможность удобной проверки баланса фигурных скобок. Достаточно выделить скобку, или даже щёлкнуть по точке вставки сразу за скобкой, чтобы её пара была подсвечена.

; точка с запятой

Точка с запятой должна использоваться в конце выражения и разделять элементы программы. Также точка с запятой используется для разделения элементов циклаfor.

```
int x = 13;    // объявляет переменную 'x' как целое 13
```

Примечание: Если забыть завершить стоку точкой с запятой, то это приведёт к возникновению ошибки компиляции. Текст ошибки может быть очевиден и указывать на пропущенную точку с запятой, но может быть и не таким очевидным. Если появляется маловразумительная или нелогичная ошибка компилятора, первое, что следует проверить — не пропущена ли точка с запятой вблизи строки, где компилятор выразил своё недовольство.

/* ... */ блок комментария

Блок комментария или однострочный комментарий —– это область текста, которая игнорируется программой и используется для добавления текста с описанием кода или примечаний. Комментарий помогают другим понять эту часть программы. Он начинается с /* и заканчивается */ и может содержать множество строк.

```
/* это «огороженный» блок комментария,и
не забудьте «закрыть» комментарий - он
должен быть сбалансирован!
*/
```

Поскольку комментарии игнорируются программой, а, следовательно, не занимают места в памяти, они могут быть достаточно ёмкими, но кроме того, они могут использоваться для «пометки» блоков кода с отладочной целью.

Примечание: Хотя допускается вставка однострочного комментария в блоке комментария, второй блок комментария не допускается.

// однострочный комментарий

Однострочный комментарий начинается с // и заканчивается (внутренним) кодом перехода на другую строку. Как и блок комментария, он игнорируется программой и не занимает места в памяти.

```
// вот так выглядит однострочный комментарий
```

Однострочный комментарий часто используется после действенного выражения, чтобы дать больше информации о том, что выражение выполняет или в качестве напоминания на будущее.

переменные

Переменные — это способ именовать и хранить числовые значения для последующего использования программой. Само название - переменные, говорито том, что переменные - это числа, которые могут последовательно меняться, в отличие от констант, чьё значение никогда не меняется. Переменные нужно декларировать (объявлять), и, что очень важно - им можно присваивать значения, которые нужно сохранить. Следующий код объявляет переменную inputVariable, а затем присваивает ей значение, полученное от 2-го аналогового порта:

```
int inputVariable = 0;          // объявляется переменная и
                                // ей присваивается значение 0
inputVariable = analogRead(2); // переменная получает значение
                                // аналогового вывода 2
```

'inputVariable' — это наша переменная. Первая строка декларирует, что она будет содержать int, короткое целое. Вторая строка присваивает ей значение аналогового вывода 2. Это делает значение на выводе 2 доступным в любом месте программы.

Когда переменной присвоено значение, или пере-присвоено, вы можете проверить это значение, если оно встречается в некотором условии, или использовать его непосредственно. Рассмотрим пример, иллюстрирующий три операции с переменными. Следующий код проверяет, не меньше ли 100 значение переменной, а если так, переменной inputVariable присваивается значение 100, а затем задаётся пауза, определяемая переменной inputVariable, которая теперь, как минимум, равна 100:

```
if (inputVariable < 100) // проверяем, не меньше ли 100 переменная
{
  inputVariable = 100;   // если так, присваиваем ей значение 100
}
delay(inputVariable);    // используем переменную как паузу
```

Примечание: Переменные должны иметь наглядные имена, чтобы код был удобочитаемый. Имена переменных как tiltSensor или pushButton помогают программисту при последующем чтении кода понять, что содержит эта переменная. Имена переменных как var или value, с другой стороны, мало делают для понимания кода, и здесь используются только в качестве примера. Переменные могут быть названы любыми именами, которые не являются ключевыми словами языка программирования Arduino.

объявление переменных

Все переменные должны быть задекларированы до того, как они могут использоваться. Объявление переменной означает определение типа её значения: int, long, float и т.д., задание уникального имени переменной, и дополнительно ей можно присвоить начальное значение. Всё это следует делать только один раз в программе, но значение может меняться в любое время при использовании арифметических или других разных операций.

Следующий пример показывает, что объявленная переменная inputVariable имееттип int, и её начальное значение равно нулю. Это называется простым присваиванием.

```
int  inputVariable = 0;
```

Переменная может быть объявлена в разных местах программы, и то, где это сделано, определяет, какие части программы могут использовать переменную.

границы переменных

Переменные могут быть объявлены в начале программы перед void setup(), локально внутри функций, и иногда в блоке выражений таком, как цикл for. То, где объявлена переменная, определяет её границы (область видимости), или возможность некоторых частей программы её использовать.

Глобальные переменные таковы, что их могут видеть и использовать любые функции и выражения программы. Такие переменные декларируются в начале программы перед функцией setup().

Локальные переменные определяются внутри функций или таких частей, как цикл for. Они видимы и могут использоваться только внутри функции, в которой объявлены. Таким образом, могут существовать несколько переменных с одинаковыми именами в разных частях одной программы, которые содержат разные значения. Уверенность, что только одна функция имеет доступ к её переменной, упрощает программу и уменьшает потенциальную опасность возникновения ошибок.

Следующий пример показывает, как декларировать несколько разных типов переменных, и демонстрирует видимость каждой переменной:

```
int value;                  // 'value' видима
                            // для любой функции
void setup()
{
  // no setup needed
}

void loop()
{
  for (int i=0; i<20;)      // 'i' видима только
  {                         // внутри цикла for
    i++;
  }
  float f;                  // 'f' видима только
}                           // внутри loop
```

byte

Байт хранит 8-битовое числовое значение без десятичной точки. Он имеетдиапазон от 0 до 255.

```
byte someVariable = 180;   // объявление 'someVariable'
                           // как имеющей тип byte
```

int

Целое — это первый тип данных для хранения чисел без десятичной точки, ихранит 16-битовое значение в диапазоне от 32767 до -32768.

```
int someVariable = 1500;   // объявляет 'someVariable'
                           // как переменную целого типа
```

Примечание: Целые переменные будут переполняться, если форсировать их переход через максимум или минимум при присваивании или сравнении.Например, если x = 32767 и следующее выражение добавляет 1 к x, x = x +1 или x++, в этом случае x переполняется и будет равен -32678.

long

Тип данных увеличенного размера для больших целых, без десятичной точки, сохраняемый в 32-битовом значении с диапазоном от 2147383647 до -2147383648.

```
long someVariable = 90000; // декларирует 'someVariable'
                           // типа long
```

float

Тип данных для чисел с плавающей точкой или чисел, имеющих десятичную точку.Числа с плавающей точкой имеют большее разрешение, чем целые и сохраняются как 32-битовые значения в диапазоне от 3.4028235E+38 до -3.4028235E+38.

```
float someVariable = 3.14; // объявление 'someVariable'
                           //   как floating-point тип
```

Примечание: Числа с плавающей точкой не точные, и могут выдавать странные результаты при сравнении. Вычисления с плавающей точкой медленнее, чем вычисления целых при выполнении расчётов, так что, без нужды, их следуетизбегать.

массивы

```
int myArray[] = {value0, value1, value2...}
```

Массив — это набор значений, к которым есть доступ через значение индекса. Любое значение в массиве может быть вызвано через вызов имени массива и индекса значения. Индексы в массиве начинаются с нуля с первым значением, имеющим индекс 0. Массив нуждается в объявлении, а дополнительно может заполняться значениями до того, как будет использоваться.

Схожим образом можно объявлять массив, указав его тип и размер, а позже присваивать значения по позиции индекса:

```
int myArray[5];     // объявляет массив целых длиной в 6 позиций
myArray[3] = 10;    // присваивает по 4у индексу значение 10
```

Чтобы извлечь значение из массива, присвоим переменной значение по индексу массива:

```
x = myArray[3];     // x теперь равно 10
```

Массивы часто используются в цикле for, где увеличивающийся счётчик применяется для индексации позиции каждого значения. Следующий пример использует массив для мерцания светодиода. Используемый цикл for со счётчиком, начинающимся с 0, записывает значение из позиции с индексом 0 массива flicker[], в данном случае 180, на PWM-вывод (широтно-импульсная модуляция) 10; затем пауза в 200 ms, а затем переход к следующей позиции индекса.

```
int ledPin = 10;                        // LED на выводе 10
byte flicker[] = {180, 30, 255, 200, 10, 90, 150, 60};
                                        // выше массив из 8
void setup()                            // разных значений
{
  pinMode(ledPin, OUTPUT);              //задаём OUTPUT вывод
}

void loop()
{
  for(int i=0; i<7; i++)                // цикл равен числу
  {                                     // значений в массиве
    analogWrite(ledPin, flicker[i]);    //пишем значение по индексу
    delay(200);                         // пауза 200 мС
  }
}
```

арифметика

Арифметические операции включают сложение, вычитание, умножение и деление. Они возвращают сумму, разность, произведение или частное (соответственно) двух операндов.

```
y = y + 3;
x = x - 7;
i = j * 6;
r = r / 5;
```

Операция управляется используемым типом данных операндов, так что, например, 9/4 даёт 2 вместо 2.25, поскольку 9 и 4 имеют тип int и не могут использовать десятичную точку. Это также означает, что операция может вызвать переполнение, если результат больше, чем может храниться в данном типе.

Если используются операнды разного типа, то для расчётов используется больший тип. Например, если одно из чисел (операндов) типа float, а второе целое, то для вычислений используется тип с плавающей точкой.

Выбирайте типы переменных достаточные для хранения результатов ваших вычислений. Прикиньте, в какой точке ваша переменная переполнится, а также, что случится в другом направлении, то есть, (0-1) или (0- -32768). Для вычислений, требующих дробей, используйте переменные типа float, но остерегайтесь их недостатков: большой размер и маленькая скорость вычислений.

Примечание: Используйте оператор приведения типа (нзвание типа) для округления, то есть, (int)myFloat - для преобразования переменной одного типа в другой «на лету». Например, i = (int) 3.6 - поместит в i значение 3.

смешанное присваивание

Смешанное присваивание сочетает арифметические операции с операциями присваивания. Чаще всего встречается в цикле for, который описан ниже. Наиболее общее смешанное присваивание включает:

```
x ++       // то же, что x = x + 1, или увеличение   x на +1
x --       // то же, что x = x - 1, или уменьшение   x на -1
x += y     // то же, что x = x + y, или увеличение   x на +y
x -= y     // то же, что x = x - y, или уменьшение   x на -y
x *= y     // то же, что x = x * y, или умножение    x на y
x /= y     // то же, что x = x / y, или деление   x на y
```

Примечание: Например, x *= 3 утроит старое значение x и присвоит полученный результат x.

операторы сравнения

Сравнения одной переменной или константы с другой используются в выражении для if, чтобы проверить истинность заданного условия. В примерах на следующих страницах ?? используется для обозначения любого из следующих условий:

```
x == y      // x равно y
x != y      // x не равно y
x <  y      // x      меньше, чем   y
x >  y      // x      больше, чем   y
x <= y      // x      меньше, чем или равно  y
x >= y      // x      больше, чем или равно  y
```

логические операторы

Логические операторы, чаще всего, это способ сравнить два выражения и вернуть ИСТИНА или ЛОЖЬ, в зависимости от оператора. Есть три логических оператора: AND, OR и NOT, часто используемые в конструкциях if:

```
Logical AND:
if (x > 0 && x < 5)      // true, только если оба
                         // выражения true

Logical OR:
if (x > 0 || y > 0)      // true, если любое из
                         // выражений true

Logical NOT:
if (!x > 0)              // true, если только
                         // выражение false
```

константы

Язык Arduino имеет несколько предопределённых величин, называемых константами. Они используются, чтобы сделать программу удобной для чтения. Константы собраны в группы.

true/false

Это Булевы константы, определяющие логические уровни. FALSE легко определяется как0 (ноль), а TRUE, как 1, но может быть и чем-то другим, отличным от нуля. Так что в Булевом смысле -1, 2 и 200 — это всё тоже определяется как TRUE.

```
if (b == TRUE);
{
  doSomething;
}
```

high/low

Эти константы определяют уровень выводов как HIGH или LOW и используются при чтении или записи на логические выводы. HIGH определяется как логический уровень 1, ON или 5 вольт(3-5), тогда как LOW — 0, OFF или 0 вольт(0-2).

```
digitalWrite(13, HIGH);
```

input/output

Константы используются с функцией pinMode() для задания режима работы цифровых выводов: либо как INPUT (вход), либо как OUTPUT (выход).

```
pinMode(13, OUTPUT);
```

управление программой

if

Конструкция if проверяет, будет ли выполнено некое условие, такое, как, например, будет ли аналоговое значение больше заданного числа, и выполняет какое-то выражение в скобках, если это условие true (истинно). Если нет, то выражение в скобках будет пропущено. Формат для if следующий:

```
if (someVariable ?? value)
{
  doSomething;
}
```

Пример выше сравнивает someVariable со значением (value), которое может быть и переменной, и константой. Если выражение или условие в скобках истинно, выполняется выражение в фигурных скобках. Если нет, выражение в фигурных скобках пропускается, и программа выполняется с оператора, следующего заскобками.

Примечание: Остерегайтесь случайного использования «=», как в if (x = 10), что технически правильно, определяя x равным 10, но результат этого всегда true. Вместо этого используйте «==», как в if (x == 10), что осуществляет проверку значения x — равно ли оно 10 или нет. Запомните «=» - равно, а «==» - равно ли?

if...else

Конструкция if...else позволяет сделать выбор «либо, либо». Например, если вы хотите проверить цифровой вход и выполнить что-то, если он HIGH, или выполнить что-то другое, если он был LOW, вы должны записать следующее:

```
if (inputPin == HIGH)
{
  doThingA;
}
else
{
  doThingB;
}
```

else может также предшествовать другой проверке if так, что эти множественные, взаимоисключающие проверки могут запускаться одновременно. И возможно даже неограниченное количество подобных else переходов. Хотя следует помнить, что только один набор выражений будет выполнен в зависимости от результата проверки:

```
if (inputPin < 500)
{
  doThingA;
}
else if (inputPin >= 1000)
{
  doThingB;
}
else
{
  doThingC;
}
```

Примечание: Конструкция if просто проверяет, будет ли выражение в круглых скобках истинно или ложно. Это выражение может быть любым правильным,относительно языка Си, выражением, как в первом примере if (inputPin == HIGH). В этом примере if проверяет только то, что означенный вход в состоянии высокого логического уровня или действительно ли напряжение на нём 5 вольт.

for

Конструкция for используется для повторения блока выражений, заключённых в фигурные скобки заданное число раз. Наращиваемый счётчик часто используется для увеличения и прекращения цикла. Есть три части, разделённые точкой с запятой, в заголовке цикла for:

```
for ( инициализация; условие; выражение )
{
  doSomething;
}
```

«Инициализация» локальной переменной, или счётчика, имеет место в самом начале и происходит только один раз. При каждом проходе цикла проверяется «условие». Если условие остаётся истинным, то следующее выражение и блок выполняются, а условие проверяется вновь. Когда условие становится ложным, цикл завершается.

Следующий пример начинается с целого i равного 0, проверяет, остаётся ли i ещё меньше 20, и, если так, увеличивает i на 1 и выполняет блок в фигурных скобках:

```
for (int i=0; i<20; i++)  // декларирует i, проверяет меньше ли
{                         // чем 20, увеличивает i на 1
  digitalWrite(13, HIGH); // устанавливает вывод 13 в ON
  delay(250);             // пауза в 1/4 секунды
  digitalWrite(13, LOW);  // сбрасывает вывод 13 в OFF
  delay(250);             // пауза в 1/4 секунды
}
```

Примечание: В Си цикл for более гибок, чем это можно обнаружить в других языках программирования, включая Basic. Любые или все три элемента заголовка могут быть опущены, хотя точка с запятой требуется. Также выражения для инициализации, условия и выражения могут быть любыми правильными выражениями Си с несвязанными переменными. Такие необычные типы выражений могут помочь в решении некоторых редких программных проблем.

while

Цикл while продолжается, и может продолжаться бесконечно, пока выражение в скобках не станет false (ложно). Что-то должно менять проверяемую переменную, иначе из цикла никогда не выйти. И это должно быть в вашем коде, как, скажем, увеличение переменной, или внешнее условие, как, например, проверяемый сенсор.

```
while (someVariable ?? value)
{
  doSomething;
}
```

Следующий пример проверяет, будет ли someVariable меньше 200, и если да, то выполняются выражения в фигурных скобках, и цикл продолжается, пока someVariable остаётся меньше 200.

```
While (someVariable < 200) // проверяет, меньше ли 200
{
  doSomething;             //  выполняет выражение в скобках
  someVariable++;          //  увеличивает переменную на 1
}
```

do...while

Цикл do управляемый «снизу» цикл, работающий на манер цикла while, с тем отличием, что условие проверки расположено в конце цикла, таким образом, цикл выполнится хотя бы один раз.

```
do
{
  doSomething;
} while (someVariable ?? value);
```

Следующий пример присваивает readSensor переменной x, делает паузу на 50 миллисекунд, затем цикл выполняется, пока x меньше, чем 100.

```
do
{
  x = readSensors();     // присваиваем значение
                         // readSensors() переменной x
  delay(50);             //  пауза 50 миллисекунд
} while (x < 100);       //продолжение цикла, если x меньше 100
```

цифровой ввод/вывод

pinMode (pin, mode)

Используется в void setup () для конфигурации заданного вывода, чтобы он работал на вход (INPUT) или на выход (OUTPUT).

```
pinMode(pin, OUTPUT);      // устанавливает 'pin' на выход
```

Цифровые выводы в Arduino предустановлены на вход, так что их нет нужды явно объявлять как INPUT с помощью pinMode (). Выводы, сконфигурированные как INPUT, подразумеваются в состоянии с высоким импедансом (сопротивлением).

В микроконтроллере Atmega, есть также удобные, программно доступные подтягивающие резисторы 20 кОм. Эти встроенные подтягивающие резисторы доступны следующим образом:

```
pinMode(pin, INPUT);           // настраиваем 'pin' на вход
digitalWrite(pin, HIGH);  // включаем подтягивающие резисторы
```

Подтягивающие резисторы, как правило, используются при соединении входов с переключателями. Заметьте, что в примере выше нет преобразования pin на выход, это просто метод активизации встроенных подтягивающих резисторов.

Выводы, сконфигурированные как OUTPUT, находятся в низкоимпедансном состоянии и могут отдавать 40 мА в нагрузку (цепь, другое устройство). Это достаточный ток для яркого включения светодиода (не забудьте последовательный токоограничительный резистор!), но не достаточный для включения реле, соленоидов или моторов.

Короткое замыкание выводов Arduino или слишком большой ток могут повредить выходы или даже всю микросхему Atmega. Порой, не плохая идея — соединять OUTPUT вывод через последовательно включённый резистор в 470 Ом или 1 кОм.

digitalRead (pin)

Считывает значение заданного цифрового вывода (pin) и возвращает результат HIGH или LOW. Вывод должен быть задан либо как переменная, либо как константа (0-13).

```
value = digitalRead(Pin);    // задаёт 'value' равным
                             // входному выводу 'Pin'
```

digitalWrite (pin, value)

Выводит либо логический уровень HIGH, либо LOW (включает или выключает) на заданном цифровом выводе pin. Вывод может быть задан либо как переменная, либо как константа (0-13).

```
digitalWrite(pin, HIGH);   // sets 'pin' to high
```

Следующий пример читает состояние кнопки, соединённой с цифровым входом, и включает LED (светодиод), подключённый к цифровому выходу, когда кнопканажата:

```
int led   = 13;   // соединяем LED с выводом 13
int pin   =  7;   // соединяем кнопку с выводом 7
int value =  0;// переменная для хранения прочитанного значения

void setup()
{
  pinMode(led, OUTPUT);     // задаём вывод 13 как выход
  pinMode(pin, INPUT);      // задаём вывод 7 как вход
}

void loop()
{
  value = digitalRead(pin);  // задаём 'value' равной
                             // входному выводу
  digitalWrite(led, value);  // устанавливаем 'led' в
}                            // в значение кнопки
```

analogRead (pin)

Считывает значение из заданного аналогового входа (pin) с 10-битовым разрешением. Эта функция работает только на аналоговых портах (0-5). Результирующее целое значение находится в диапазоне от 0 до 1023.

```
value = analogRead(pin);   // задаёт 'value' равным 'pin'
```

Примечание: Аналоговые выводы не похожи на цифровые, и нет необходимости предварительно объявлять их как INPUT или OUTPUT (если только вы не планируете использовать их в качестве цифровых портов 14-18).

analogWtite (pin, value)

Записывает псевдо-аналоговое значение, используя схему с широтно-импульсной модуляцией (PWM), на выходной вывод, помеченный как PWM. На новом модуле Arduino с ATmega168 (328), эта функция работает на выводах 3, 5, 6, 9, 10 и 11. Старый модуль Arduino с ATmega8 поддерживает только выводы 9, 10 и 11. Значение может быть задано как переменная или константа в диапазоне 0-255.

```
analogWrite(pin, value);  // записывает 'value' в аналоговый 'pin'
```

Значение 0 генерирует устойчивое напряжение 0 вольт на выходе заданного вывода; значение 255 генерирует 5 вольт на выходе заданного вывода. Для значений между 0 и 255 вывод быстро переходит от 0 к 5 вольтам — чем больше значение, тем чаще вывод в состоянии HIGH (5 вольт). Например, при значении 64 вывод будет в 0 три четверти времени, а в состоянии 5 вольт одну четверть; при значении 128 половину времени будет вывод будет в 0, а половину в 5 вольт; при значении 192 четверть времени вывод будет в 0 и три четверти в 5 вольт.

Поскольку эта функция схемная (встроенного модуля), вывод будет генерировать устойчивый сигнал после вызова analogWrite в фоновом режиме, пока не будет следующего вызова analogWrite (или вызова digitalRead или digitalWrite на тот же вывод).

Примечание: Аналоговые выводы, не такие как цифровые, и не требуют предварительной декларации их как INPUT или OUTPUT.

Следующий пример читает аналоговое значение с входного аналогового вывода, конвертирует значение делением на 4 и выводит PWM сигнал на PWM вывод:

```
int led = 10;     // LED с резистором на выводе 10
int pin =  0;     // потенциометр на аналоговом выводе 0
int value;        // значение для чтения

void setup(){}    // setup не нужен

void loop()
{
  value =  analogRead(pin);   // задаёт 'value' равной 'pin'
  value /= 4;                 // конвертирует 0-1023 в 0-255
  analogWrite(led, value);    // выводит PWM сигнал на LED
}
```

время и математика

delay (ms)

Приостанавливает вашу программу на заданное время (в миллисекундах), где1000 равно 1 секунде.

```
delay(1000);    // ждём одну секунду
```

millis()

Возвращает число миллисекунд, как unsigned long, с момента старта программы вмодуле Arduino.

```
value = millis();   // задаёт 'value' равной millis ()
```

Примечание: Это число будет переполняться (сбрасываться в ноль), после,примерно, 9 часов.

min (x, y)

Вычисляется минимум двух чисел любого типа данных и возвращает меньшеечисло.

```
value = min(value, 100); // устанавливает 'value' в наименьшее из
                         // value и 100, обеспечивая, что
                         // оно никогда не превысит 100
```

max (x, y)

Вычисляется максимум двух чисел любого типа данных и возвращает большеечисло.

```
value = max(value, 100); // устанавливает 'value' в наибольшее из
                         // value и 100, обеспечивая, что
                         // оно всегда не меньше 100
```

случайные числа

randomSeed (seed)

Устанавливает значение, или начальное число, в качестве начальной точки функцииrandom().

```
randomSeed(value);     // задаёт 'value' как начальное значение random
```

Поскольку Arduino не может создавать действительно случайных чисел, randomSeed позволяет вам поместить переменную, константу или другую функцию в функцию random, что помогает генерировать более случайные «random» числа. Есть множество разных начальных чисел, или функций, которые могут быть использованы в этой функции, включая millis(), или даже analogRead() для чтения электрических шумов через аналоговый вывод.

random (max)
random (min, max)

Функция random позволяет вам вернуть псевдослучайное число в диапазоне, заданном значениями min и max.

```
value = random(100, 200);   // задаёт 'value' случайным
                            // числом между 100 и 200
```

Примечание: Используйте это после использования функции randomSeed().

Следующий пример создаёт случайное число между 0 и 255 и выводит PWM сигнал на PWM вывод, равный случайному значению:

```
int randNumber;   // переменная для хранения случайного значения
int led = 10;     // LED с резистором на выводе 10

void setup() {}   // setup не нужен

void loop()
{
  randomSeed(millis());           // задаёт millis() начальным числом
  randNumber = random(255);       // случайное число из 0-255
  analogWrite(led, randNumber);   // вывод PWM сигнала
  delay(500);                     // пауза в полсекунды
}
```

последовательный обмен

Serial.begin (rate)

Открывает последовательный порт и задаёт скорость для последовательной передачи данных. Типичная скорость обмена для компьютерной коммуникации — 9600, хотя поддерживаются и другие скорости.

```
void setup()
{
  Serial.begin(9600);    // открывается последовательный порт
}                        // задаётся скорость обмена 9600
```

Примечание: При использовании последовательного обмена, выводы 0 (RX) и 1 (TX) не могут использоваться одновременно как цифровые.

Serial.println (data)

Передаёт данные в последовательный порт, сопровождая автоматическим возвратом каретки и переходом на новую строку. Команда такая же, что и Serial.print(), но легче для последующего чтения на данных в терминале.

```
Serial.println(analogValue);  // отправляет значение
                              // 'analogValue'
```

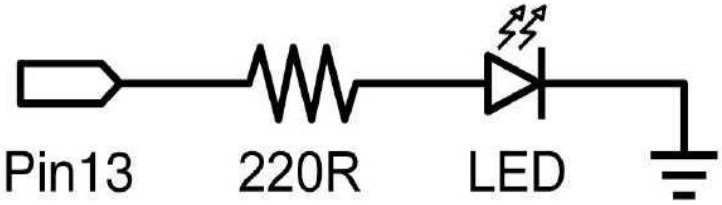
Примечание: За дальнейшей информацией о различных изменениях Serial.println() и Serial.print() обратитесь на сайт Arduino.

Следующий простой пример читает аналоговый вывод 0 и отсылает эти данные на компьютер каждую секунду.

```
void setup()
{
  Serial.begin(9600);              // задаём скорость 9600 bps
}

void loop()
{
  Serial.println(analogRead(0)); // шлём аналоговое значение
  delay(1000);                   // пауза 1 секунда
}
```

цифровой выход



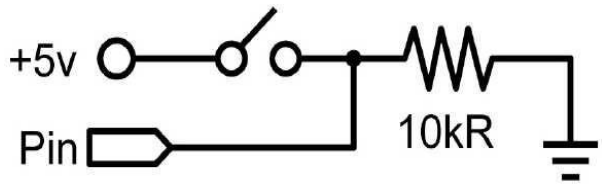
Это базовая программа «hello world», используемая для включения и выключения чего-нибудь. В этом примере светодиод подключён к выводу 13 и мигает каждую секунду. Резистор в данном случае может быть опущен, поскольку на 13-м порту Arduino уже есть встроенный резистор.

```
int ledPin = 13;                  // LED на цифровом выводе 13

void setup()                      // запускается один раз
{
  pinMode(ledPin, OUTPUT);        // устанавливае вывод 13 на выход
}

void loop()                       // запускаем вновь и вновь
{
  digitalWrite(ledPin, HIGH);     // включаем LED
  delay(1000);                    // пауза 1 секунда
  digitalWrite(ledPin, LOW);      // выключаем LED
  delay(1000);                    // пауза 1 секунда
}
```

цифровой ввод



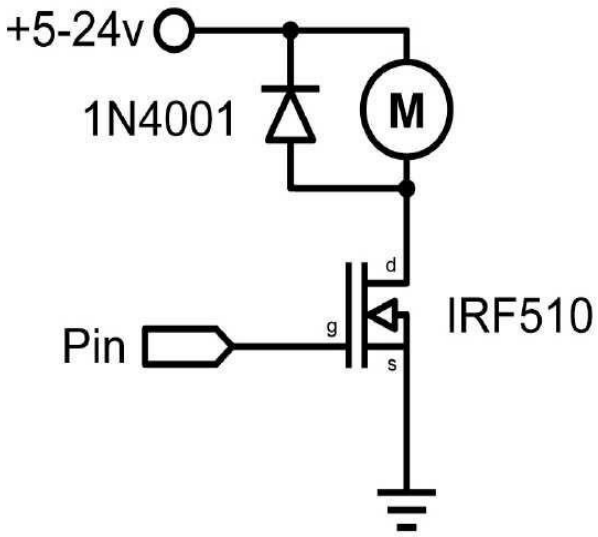
Это простейшая форма ввода с двумя возможными состояниями: включено или выключено. В примере считывается простой переключатель или кнопка, подключённая к выводу 2. Когда выключатель замкнут, входной вывод читается как HIGH и включает светодиод.

```
int ledPin = 13;                // выходной вывод для LED
int inPin = 2;                  // входной вывод (для switch)

void setup()
{
  pinMode(ledPin, OUTPUT);   // объявляем LED как выход
  pinMode(inPin, INPUT);     // объявляем switch как вход
}

void loop()
{
  if (digitalRead(inPin) == HIGH) // проверяем вход, HIGH?
  {
    digitalWrite(ledPin, HIGH);   // включаем LED
    delay(1000);                  // пауза 1 секунда
    digitalWrite(ledPin, LOW);    // выключаем LED
    delay(1000);                  // пауза 1 секунда
  }
}
```

сильноточный выход



Иногда возникает необходимость в управлении более, чем 40 мА от Arduino. В этом случае может использоваться транзистор MOSFET для коммутации сильноточной нагрузки. В следующем примере MOSFET быстро включается и выключается 5 раз в секунду.

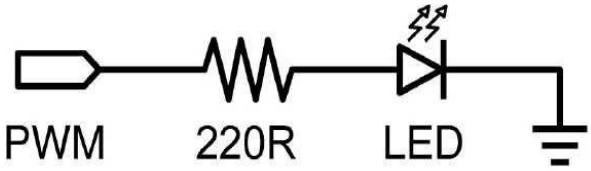
Примечание: Схема показывает мотор и диод защиты, но другие, не индуктивные, нагрузки могут включаться без диода.

```
int outPin  =  5;          // выходной вывод для MOSFET

void setup()
{
  pinMode(outPin, OUTPUT);       // задаём вывод 5 как выход
}

void loop()
{
  for (int i=0; i<=5; i++)       // цикл 5 раз
  {
    digitalWrite(outPin, HIGH); // включаем MOSFET
    delay(250);                 // пауза в 1/4 секунды
    digitalWrite(outPin, LOW);  // выключаем MOSFET
    delay(250);                 // пауза в 1/4 секунды
  }
  delay(1000);                  // пауза 1 секунда
}
```

pwm выход



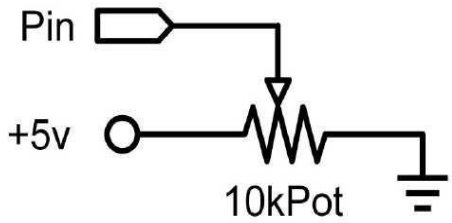
Широтно-импульсная модуляция (PWM) — это способ имитировать аналоговый выход с помощью импульсного сигнала. Это можно использовать для гашения и увеличения яркости светодиода или позже для управления сервомотором. Следующий пример медленно увеличивает яркость и гасит LED, используя цикл for.

```
int ledPin  =  9;      // PWM вывод для LED

void setup(){}         //  setup не нужен

void loop()
{
  for (int i=0; i<=255; i++)  // растущее значение для i
  {
    analogWrite(ledPin, i);   //устанавливаем уровень яркости для i
    delay(100);               //  пауза 100 мС
  }
  for (int i=255; i>=0; i--)  // спадающее значение для i
  {
    analogWrite(ledPin, i);   //устанавливаем уровень яркости для i
    delay(100);               //  пауза 100 мС
  }
}
```

вход с потенциометра



Использование потенциометра и одного из аналоговых портов Arduino (аналого-цифрового преобразователя (ADC)) позволяет читать аналоговые значения в диапазоне 0-1023. Следующий пример показывает использование потенциометра для управления временем мигания светодиода LED.

```
int potPin = 0;     // входной вывод для потенциометра
int ledPin = 13;    // выходной вывод для LED

void setup()
{
  pinMode(ledPin, OUTPUT);  // объявляем ledPin как OUTPUT
}

void loop()
{
  digitalWrite(ledPin, HIGH);     // включаем ledPin
  delay(analogRead(potPin));      // пауза
  digitalWrite(ledPin, LOW);      // выключаем ledPin
  delay(analogRead(potPin));      // пауза
}
```

вход от переменного резистора

Переменные резисторы включают фотоприёмники, термисторы, тензодатчики и т.д. Данный пример использует функцию чтения аналогового значения и задаёт время паузы. Этим управляется скорость, с которой меняется яркость светодиодаLED.

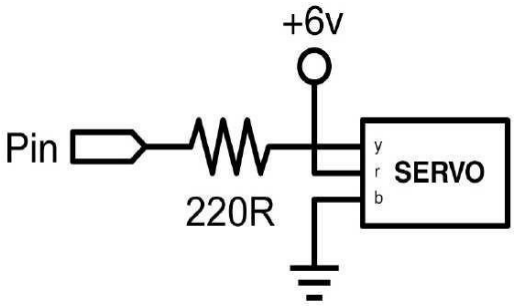
```
int ledPin    =  9;    // PWM вывод для LED
int analogPin =  0;    // переменный резистор на аналоговом выводе 0

void setup(){}        //   setup не нужен

void loop()
{
  for (int i=0; i<=255; i++)  // увеличивающееся значение для i
  {
    analogWrite(ledPin, i);   // устанавливаем уровень яркости по i
    delay(delayVal());        // берём значение времени и паузы
  }
  for (int i=255; i>=0; i--)  //  уменьшающееся значение для i
  {
    analogWrite(ledPin, i);   //  устанавливаем уровень яркости по i
    delay(delayVal());        //  берём значение времени и паузы
  }
}

int delayVal()
{
  int v;                      //  создаём временную переменную
  v  = analogRead(analogPin); //  читаем аналоговое значение
  v /= 8;                     //  конвертируем 0-1024 в 0-128
  return v;                   //  возвращаем окончательное значение
}
```

серво вывод



Любительские сервомашинки — это разновидность полу-автономного мотор- редуктора, который может поворачиваться на 180^0. Всё, что нужно — это отправлять импульсы каждые 20 мС. В данном примере используется функция servoPulse для поворота мотора от 10^0 до 170^0 и обратно.

```
int servoPin = 2;    // привод соединён с цифровым выводом 2
int myAngle;         // угол привода грубо 0-180
int pulseWidth;      // servoPulse функции переменная

void setup()
{
  pinMode(servoPin, OUTPUT);    // устанавливаем вывод 2 на выход
}

void servoPulse(int servoPin, int myAngle)
{
  pulseWidth = (myAngle * 10) + 600;  // определяем паузу
  digitalWrite(servoPin, HIGH);       // устанавливаем привод в high
  delayMicroseconds(pulseWidth);      // микросекундная пауза
  digitalWrite(servoPin, LOW);        // устанавливаем привод в low
}

void loop()
{
  // привод стартует с 10 градусов и поворачивается до 170
  for (myAngle=10; myAngle<=170; myAngle++)
  {
    servoPulse(servoPin, myAngle);    // отправляем вывод и угол
    delay(20);                        // обновляем цикл
  }
  // привод стартует со 170 и поворачивается до 10 градусов
  for (myAngle=170; myAngle>=10; myAngle--)
  {
    servoPulse(servoPin, myAngle);    // отправляем вывод и угол
    delay(20);                        // обновляем цикл
  }
}
```

Приложение 5

Краткая шпаргалка по основным функциям Arduino Wiring и языку C++

Краткая шпаргалка по основным функциям Arduino Wiring и языку C++. Оформлено с комментариями и примерами в виде кода: для большей наглядности и возможности сразу почувствовать код и запомнить, как он выглядит. Для полной информации по каждой главе обращайтесь по ссылкам.

Оглавление

Синтаксис

```
// полный урок тут: https://alexgyver.ru/lessons/syntax/

// однострочный комментарий

/*
  многострочный
  комментарий
*/

// каждая команда оканчивается ;
// каждой скобке ( { < соответствует закрывающая > } )

// === ПРЕПРОЦЕССОР ===
#include <Servo.h>     // подключает библиотеку. Ищет в папке с библиотеками
#include "Servo.h"     // ищет в папке со скетчем, а потом в папке с библиотеками
#define MY_CONST 10    // объявить "жёсткую" константу MY_CONST равной 10

// эта функция обязательно должна быть в скетче в одном экземпляре
void setup() {
  // код выполнится 1 раз при старте программы
}

// эта функция обязательно должна быть в скетче в одном экземпляре
void loop() {
  // код будет выполняться циклично после setup
}
```

Переменные и типы данных

```
// полный урок тут: https://alexgyver.ru/lessons/syntax/

boolean flag1;                  // объявить
boolean flag1, flag2;           // объявить несколько
boolean flag1 = true;           // объявить и инициализировать

// === ТИПЫ ДАННЫХ ===
boolean или bool                //  1 байт, логическая. true/false или 1/0
int8_t                          //  1 байт, целочисл., -128… 127
char                            //  1 байт, символьная, -128… 127 или 'a'
uint8_t, byte                   //  1 байт, целочисл., 0… 255
int16_t, int, short             //  2 байта, целочисл., -32 768… 32 767
uint16_t, unsigned int,  word   //  2 байта, целочисл., 0… 65 535
int32_t, long                   //  4 байта, целочисл., -2 147 483 648… 2 147 483 647
uint32_t, unsigned long         //  4 байта, целочисл., 0… 4 294 967 295
float, double                   //  4 байта, дробн., -3.4028235E+38… 3.4028235E+38
// примеч.: на других платформах double имеет размер 8 бит и бОльшую точность

'a'     // символ
"abc"   // строка или массив символов

// === МАССИВЫ ===
int myInts[6];     // указываем количество ячеек
int myPins[] = {2, 4, 8, 3, 6};   // указываем содержимое ячеек
float Sens[3] = {0.2, 0.4, -8.5}; // указываем и то и то, количество ячеек должно
совпадать
char message[6] = "hello";        // храним символы

// === СПЕЦИФИКАТОРЫ ===
const     // константа, такую переменную нельзя изменить. const int val = 10;
static    // статическая переменная (см. ниже)
volatile // не оптимизировать переменную. Использовать для работы в прерываниях
extern    // указывает компилятору, что эта переменная объявлена в другом файле
программы
```

Область видимости

```
// === ГЛОБАЛЬНАЯ ===
// Глобальная переменная объявляется вне функций и доступна
// для чтения и записи в любом месте программы, в любой её функции.
byte var;
void setup() {
  // спокойно меняем глобальную переменную
  var = 50;
}
void loop() {
  // спокойно меняем глобальную переменную
  var = 70;
}

// === ЛОКАЛЬНАЯ ===
// Локальная переменная живёт внутри функции или внутри любого блока кода,
// заключённого в { фигурные скобки }, доступна для чтения и записи только внутри него.
void setup() {
  byte var;  // локальная для setup переменная
  // спокойно меняем локальную переменную
  var = 50;
}
void loop() {
  // приведёт к ошибке, потому что в этом блоке кода var не объявлена
  var = 70;

  // сделаем тут отдельный блок кода
  {
    byte var2 = 10;
    // var2 существует только внутри этого блока!
  }
  // вот тут var2 уже будет удалена из памяти
}

// === СТАТИЧЕСКАЯ ЛОКАЛЬНАЯ ===
// статическая локальная переменная не удаляется из памяти
// после выхода из функции
void setup() {
  myFunc(); // вернёт 20
  myFunc(); // вернёт 30
  myFunc(); // вернёт 40
  myFunc(); // вернёт 50
}
void loop() {
}
byte myFunc() {
  static byte var = 10;
  var += 10;
  return var;
}
```

Строки

```
// полный урок тут: https://alexgyver.ru/lessons/strings/

String string0 = "Hello String";              // заполняем словами в кавычках
String string1 = String("lol ") + String("kek");  // сумма двух строк
String string2 = String('a');                 // строка из символа в одинарных кавычках
String string3 = String("This is string");    // конвертируем строку в String
String string4 = String(string3 + " more");   // складываем строку string3 с текстом в
кавычках
String string5 = String(13);                  // конвертируем из числа в String
String string6 = String(20, DEC);             // конвертируем из числа с указанием
базиса (десятичный)
```

```
String string7 = String(45, HEX);              // конвертируем из числа с указанием базиса (16-ричный)
String string8 = String(255, BIN);             // конвертируем из числа с указанием базиса (двоичный)
String string9 = String(5.698, 3);             // из float с указанием количества знаков после запятой (тут 3)

// длина строки
String textString = "Hello";
sizeof(textString);   // вернёт 6
textString.length();  // вернёт 5
// полный набор инструментов String тут https://alexgyver.ru/lessons/strings/

// ====== МАССИВЫ СИМВОЛОВ ======
// объявить массив текста длиной 6 символов
// и задать текст
char helloArray[] = "Hello!";

// объявить массив текста длиной 100 символов
// и задать в его начало текст
char textArray[100] = "World";

// длина строки
char textArray[100] = "World";
sizeof(textArray); // вернёт 100
strlen(textArray);  // вернёт 5
```

Serial

```
// полный урок тут: https://alexgyver.ru/lessons/serial/

// === СТАРТ/СТОП ===
Serial.begin(Speed);    // открыть порт на скорости
Serial.end();           // закрыть порт
Serial.available();     // дозвращает количество байт в буфере приёма

// === ПЕЧАТЬ ===
// Отправляет в порт значение val – число или строку
Serial.print(val);
Serial.print(val, format);

// Отправляет и переводит строку
Serial.println(val);
Serial.println(val, format);

Serial.print(78);         // выведет 78
Serial.print(1.23456);    // 1.23 (умолч. 2 знака)
Serial.print('N');        // выведет N
Serial.print("Hello world."); // Hello world.
Serial.print(78, BIN);    // вывод "1001110"
Serial.print(78, OCT);    // вывод "116"
Serial.print(78, DEC);    // вывод "78"
Serial.print(78, HEX);    // вывод "4E"
Serial.print(1.23456, 0); // вывод "1"
Serial.print(1.23456, 2); // вывод "1.23"
Serial.print(1.23456, 4); // вывод "1.2345"

// === ПАРСИНГ ===
Serial.setTimeout(value); // таймаут ожидания приёма данных для парсинга, мс. По умолчанию 1000 мс (1 секунда)
Serial.readString();      // принять строку
Serial.parseInt();        // принять целочисленное
Serial.parseFloat();      // принять float
```

Условия и выбор

```
// полный урок тут: https://alexgyver.ru/lessons/conditions/

// === Сравнение и логика ===
== , != , >= , <= ;    // равно, не равно, больше или равно, меньше или равно
! , && , || ;          // НЕ, И, ИЛИ

// === if-else ===
// при выполнении одного действия {} необязательны
if (a > b) c = 10; // если a больше b, то c = 10
else c = 20;       // если нет, то c = 20

// вместо сравнения можно использовать лог. переменную
boolean myFlag, myFlag2;
if (myFlag) c = 10;

// сложные условия
// если оба флага true - c = 10
if (myflag && myFlag2) c = 10;

// при выполнении двух и более {} обязательны
if (myFlag) {
  c = 10;
  b = c;
} else {
  c = 20;
  b = a;
}

// === else if ===
byte state;
if (state == 1) a = 10;       // если state 1
else if (state == 2) a = 20; // если нет, но если state 2
else a = 30;                  // если и это не верно, то вот

// === Оператор ? ===
// "Короткий" вариант if-else
int c = (a > b) ? 10 : -20;  // если a > b, то c = 10. Если нет, то c = -20
Serial.println( (flag) ? ("флаг поднят") : ("флаг опущен") );

// === Оператор выбора ===
switch (val) {
case 1: // выполнить, если val == 1
  break;
case 2: // выполнить, если val == 2
  break;
default: // выполнить, если val ни 1 ни 2
  // default опционален
  break;
}

// Оператор break очень важен, позволяет выйти из switch
// Можно использовать так:
switch (val) {
case 1:
case 2:
case 3:
case 4:
  // выполнить, если val == 1, 2, 3 или 4
  break;
case 5:
  // выполнить, если val == 5
  break;
}
```

Циклы

```
// полный урок тут: https://alexgyver.ru/lessons/loops/

// === for ===
for (int i = 0; i < 10; i++) {
  Serial.println(i);    // вывод в порт 0, 1.. 9
}

// === while ===
while (a < b) {
  // выполняется, пока a меньше b
}

// === do while ===
// Отличается от while тем, что выполнится хотя бы один раз
do {
  // выполняется, пока a меньше b
} while (a < b);

// === Дополнительно ===
continue;   // перейти к след. итерации цикла
break;      // выйти из цикла
```

Математика, вычисления

```
// полный урок тут: https://alexgyver.ru/lessons/compute/

+ , - , * , / , % ; // сложить, вычесть, умножить, разделить, остаток отделения
a = b + c / d;

++ , -- , += , -= , *= , /= ; // прибавить 1, вычесть 1, прибавить, вычесть,
умножить, разделить
a++;        // ~ a = a + 1;
a /= 10;    // ~ a = a / 10;

// === БОЛЬШИЕ ВЫЧИСЛЕНИЯ ===
// ВАЖНО! Для арифметических вычислений по умолчанию используется ячейка long (4 байта)
// но при умножении и делении используется int (2 байта)
// Если при умножении чисел результат превышает 32’768, он будет посчитан некорректно.
// Для исправления ситуации нужно писать (long) перед умножением, что заставит МК
выделить дополнительную память
long val;
val = 2000000000 + 6000000;          // посчитает корректно (т.к. сложение)
val = 25 * 1000;                     // посчитает корректно (умножение, меньше 32'768)
val = 35 * 1000;                     // посчитает НЕКОРРЕКТНО! (умножение, больше
32'768)
val = (long)35 * 1000;               // посчитает корректно (выделяем память (long) )
val = 1000 + 35 * 10 * 100;          // посчитает НЕКОРРЕКТНО! (в умножении больше
32'768)
val = 1000 + 35 * 10 * 100L;         // посчитает корректно! (модификатор L)
val = (long)35 * 1000 + 35 * 1000; // посчитает НЕКОРРЕКТНО! Второе умножение всё
портит
val = (long)35 * 1000 + (long)35 * 1000; // посчитает корректно (выделяем память
(long) )

// === ВЫЧИСЛЕНИЯ FLOAT ===
// если при вычислении двух целочисленных нужен дробный результат - пишем (float)
float val;
val = 100 / 3;          // посчитает НЕПРАВИЛЬНО (результат 3.0)
val = (float)100 / 3;   // посчитает правильно (указываем (float))
val = 100.0 / 3;        // посчитает правильно (есть число float)

// при присваивании float числа целочисленному типу данных дробная часть отсекается
```

```
int val;
val = 3.25;         // val принимает 3
val = 3.92;         // val принимает 3
val = round(3.25); // val принимает 3
val = round(3.92);  // val принимает 4

// === МАТЕМАТИЧЕСКИЕ ФУНКЦИИ ===
// Ограничить диапазон числа val между low и high
val = constrain(val, low, high);

// Перевести диапазон числа val (от inMin до inMax) в новый диапазон (от outMin до
outMax)
val = map(val, inMin, inMax, outMin, outMax);

min(a, b);     // Возвращает меньшее из чисел a и b
max(a, b);     // Возвращает большее из чисел
abs(x);        // Модуль числа
round(x);      // Математическое округление
radians(deg); // Перевод градусов в радианы
degrees(rad); // Перевод радиан в градусы
sq(x);         // Квадрат числа
cos(x)         // Косинус (радианы)
sin(x)         // Синус (радианы)
tan(x)         // Тангенс (радианы)
fabs(x)        // Модуль для float чисел
fmod(x, y)     // Остаток деления x на y для float
sqrt(x)        // Корень квадратный
sqrtf(x)       // Корень квадратный для float чисел
cbrt(x)        // Кубический корень
hypot(x, y)    // Гипотенуза ( корень(x*x + y*y) )
square(x)      // Квадрат ( x*x )
floor(x)       // Округление до целого вниз
ceil(x)        // Округление до целого вверх
exp(x)         // Экспонента (e^x)
cosh(x)        // Косинус гиперболический (радианы)
sinh(x)        // Синус гиперболический (радианы)
tanh(x)        // Тангенс гиперболический (радианы)
acos(x)        // Арккосинус (радианы)
asin(x)        // Арксинус (радианы)
atan(x)        // Арктангенс (радианы)
atan2(y, x)    // Арктангенс (y / x) (позволяет найти квадрант, в котором находится
точка)
log(x)         // Натуральный логарифм x ( ln(x) )
log10(x)       // Десятичный логарифм x ( log_10 x)
pow(x, y)      // Степень ( x^y )
fma(x, y, z)  // Возвращает x*y + z
fmax(x, y)     // Возвращает большее из чисел
fmin(x, y)     // Возвращает меньшее из чисел
trunc(x)       // Возвращает целую часть числа с дробной точкой
round(x)       // Математическое округление


// === КОНСТАНТЫ ===
F_CPU          // частота тактирования в Гц (16000000 для 16 МГц)
INT8_MAX       // 127 Максимальное значение для char, int8_t
UINT8_MAX      // 255 Максимальное значение для byte, uint8_t
INT16_MAX      // 32767 Максимальное значение для int, int16_t
UINT16_MAX     // 65535 Максимальное значение для unsigned int, uint16_t
INT32_MAX      // 2147483647  Максимальное значение для long, int32_t
UINT32_MAX     // 4294967295 Максимальное значение для unsigned long, uint32_t
M_E            // 2.718281828 Число e
M_LOG2E        // 1.442695041 log_2 e
M_LOG10E       // 0.434294482 log_10 e
M_LN2          // 0.693147181 log_e 2
M_LN10         // 2.302585093 log_e 10
```

```
M_PI          // 3.141592654 pi
M_PI_2        // 1.570796327 pi/2
M_PI_4        // 0.785398163 pi/4
M_1_PI        // 0.318309886 1/pi
M_2_PI        // 0.636619772 2/pi
M_2_SQRTPI    // 1.128379167 2/корень(pi)
M_SQRT2       // 1.414213562 корень(2)
M_SQRT1_2     // 0.707106781 1/корень(2)
PI            // 3.141592654 Пи
HALF_PI       // 1.570796326 пол Пи
TWO_PI        // 6.283185307 два Пи
EULER         // 2.718281828 Число Эйлера e
DEG_TO_RAD    // 0.01745329  Константа перевода град в рад
RAD_TO_DEG    // 57.2957786
```

Функции

```
// полный урок тут: https://alexgyver.ru/lessons/functions/

// Функция, которая ничего не принимает и ничего не возвращает. Пример - сумма
void sumFunction() {
  c = a + b;
}

// Функция, которая ничего не принимает и возвращает результат. Пример - сумма
int sumFunction() {
  return (a + b);
}

// Функция, которая принимает параметры и возвращает результат. Пример - сумма
int sumFunction(byte paramA, byte paramB) {
  return (paramA + paramB);
}

// оператор return завершает выполнение функции и возвращает результат
// в void функции он вернёт void, всё верно
```

Входы/выходы

Цифровые

```
// урок: https://alexgyver.ru/lessons/digital/

pinMode(pin, mode);
// Устанавливает режим работы пина pin (ATmega 328: D0-D13, A0-A5) на режим mode:
// INPUT - вход (все пины сконфигурированы так по умолчанию)
// OUTPUT - выход (при использовании analogWrite ставится автоматически)
// INPUT_PULLUP - подтяжка к питанию (например для обработки кнопок)

digitalRead(pin);
// Читает состояние пина pin и возвращает :
// 0 или LOW - на пине 0 Вольт (точнее 0-2.5В)
// 1 или HIGH - на пине 5 Вольт (точнее 2.5-опорное В)

digitalWrite(pin, value);
// Подаёт на пин pin сигнал value:
// 0 или LOW - 0 Вольт (GND)
// 1 или HIGH - 5 Вольт (точнее, напряжение питания)
```

Аналоговые

```
// урок: https://alexgyver.ru/lessons/analog-pins/

analogRead(pin);
// Читает и возвращает оцифрованное напряжение с пина pin. 0-1023
// Перевести значение в напряжение:
```

```
float volt = (float)(analogRead(pin) * 5.0) / 1024;
// именно /1024, потому что АЦП сам отнимает 1 бит при вычислении

analogReference(mode);
// Устанавливает режим работы АЦП согласно mode:
// DEFAULT: опорное напряжение равно напряжению питания МК
// INTERNAL: встроенный источник опорного на 1.1V для ATmega168 или ATmega328P и 2.56V
на ATmega8
// INTERNAL1V1: встроенный источник опорного на 1.1V (только для Arduino Mega)
// INTERNAL2V56: встроенный источник опорного на 2.56V (только для Arduino Mega)
// EXTERNAL: опорным будет считаться напряжение, поданное на пин AREF
```

ШИМ

```
// урок: https://alexgyver.ru/lessons/pwm-signal/

analogWrite(pin, value);
// Запускает генерацию ШИМ сигнала на пине pin со значением value.
// Для стандартного 8-ми битного режима это значение 0-255, соответствует скважности 0-
100%.
// ШИМ пины:
// ATmega 328/168 (Nano, UNO, Mini): D3, D5, D6, D9, D10, D11
// ATmega 32U4 (Leonardo, Micro): D3, D5, D6, D9, D10, D11, D13
// ATmega 2560 (Mega): D2 – D13, D44 – D46
```

Прерывания

```
// полный урок тут: https://alexgyver.ru/lessons/interrupts/

attachInterrupt(pin, ISR, mode);
// Подключить прерывание на номер прерывания pin,
// назначить функцию ISR как обработчик и
// установить режим прерывания mode:
// LOW – срабатывает при сигнале LOW на пине
// RISING – срабатывает при изменении сигнала на пине с LOW на HIGH
// FALLING – срабатывает при изменении сигнала на пине с HIGH на LOW
// CHANGE – срабатывает при изменении сигнала (с LOW на HIGH и наоборот)

volatile int counter = 0; // переменная-счётчик
void setup() {
  Serial.begin(9600); // открыли порт для связи
  // подключили кнопку на D2 и GND
  pinMode(2, INPUT_PULLUP);

  // D2 это прерывание 0
  // обработчик - функция buttonTick
  // FALLING - при нажатии на кнопку будет сигнал 0, его и ловим
  attachInterrupt(0, buttonTick, FALLING);
}
void buttonTick() {
  counter++;  // + нажатие
}
void  loop()  {
  Serial.println(counter); // выводим
  delay(1000);              // ждём
}
```

Случайные числа

```
// полный урок тут: https://alexgyver.ru/lessons/random/

random(max);         // возвращает случайное число в диапазоне от 0 до (max – 1)
random(min, max);    // возвращает случайное число в диапазоне от min до (max – 1)
randomSeed(value);  // дать генератору случайных чисел новую опорную точку для счёта
```

Функции времени

```
// полный урок тут: https://alexgyver.ru/lessons/time/

delay(period);
// Приостанавливает" выполнение кода на time миллисекунд.
// Дальше функции delay выполнение кода не идёт, за исключением прерываний.

delayMicroseconds(period);
// Аналог delay(), но в микросекундах

millis(); // Возвращает количество миллисекунд, прошедших со старта программы
micros(); // Возвращает количество микросекунд, прошедших со старта программы
```

Структуры

```
// полный урок тут: https://alexgyver.ru/lessons/variables-types/

struct myStruct { // создаём ярлык myStruct
  boolean a;
  byte b;
  int c;
  long d;
  byte e[5];
} kek;            // и сразу создаём структуру kek

// создаём массив структур cheburek типа myStruct
myStruct cheburek[3];

void setup() {
  // присвоим членам структуры значения вручную
  kek.a = true;
  kek.b = 10;
  kek.c = 1200;
  kek.d = 789456;
  kek.e[0] = 10;    // e у нас массив!
  kek.e[1] = 20;
  kek.e[2] = 30;

  // присвоим структуру kek структуре cheburek номер 0
  cheburek[0] = kek;

  // присвоим элемент массива из структуры kek
  // структуре cheburek номер 1
  cheburek[0].e[1] = kek.e[1];

  // забьём данными структуру cheburek номер 2
  cheburek[2] = (myStruct) {
    false, 30, 3200, 321654, {1, 2, 3, 4, 5}
  };
}
```

Перечисления

```
// полный урок тут: https://alexgyver.ru/lessons/variables-types/

// создаём перечисление modes, не создавая ярлык
enum {
  NORMAL,
  WAITING,
  SETTINGS_1,
  SETTINGS_2,
  CALIBRATION,
  ERROR_MODE,
} modes;
```

```
void setup() {
  Serial.begin(9600);   // для отладки
  modes = CALIBRATION;  // присваивание значения
  // можем сравнивать
  if (modes == CALIBRATION) {
    Serial.println("calibr");
  } else if (modes == ERROR_MODE) {
    Serial.println("error");
  }

  // присваиваем числом
  modes = 3;  // по нашему порядку это будет SETTINGS_2
}
```

Битовые операции

```
// полный урок тут: https://alexgyver.ru/lessons/bitmath/

//  &   - битовое И
//  <<  - битовый сдвиг влево
//  >>  - битовый сдвиг вправо
//  ^   - битовое исключающее ИЛИ (аналогичный оператор - xor)
//  |   - битовое ИЛИ
//  ~   - битовое НЕ

bit(val);           // возвращает 2 в степени val (0 будет 1, 1 будет 2, 2 будет 4, 3 будет 8 и т.д.)
bitClear(x, n);     // устанавливает на 0 бит, находящийся в числе x под номером n
bitSet(x, n);       // устанавливает на 1 бит, находящийся в числе x под номером n
bitWrite(x, n, b);  // устанавливает на значение b (0 или 1) бит , находящийся в числе x под номером n
bitRead(x, n);      // возвращает значение бита (0 или 1), находящегося в числе x под номером n
highByte(x);        // извлекает и возвращает старший (крайний левый) байт переменной типа word (либо второй младший байт переменной, если ее тип занимает больше двух байт).
lowByte(x);         // извлекает и возвращает младший (крайний правый) байт переменной (например, типа word).

// ====== Битовое И ======
// 0 & 0 == 0
// 0 & 1 == 0
// 1 & 0 == 0
// 1 & 1 == 1
myByte = 0b11001100;
myBits = myByte & 0b10000111;
// myBits теперь равен 0b10000100

// ====== Битовое ИЛИ ======
// 0 | 0 == 0
// 0 | 1 == 1
// 1 | 0 == 1
// 1 | 1 == 1
myByte = 0b11001100;
myBits = myByte | 0b00000001; // ставим бит №0
// myBits теперь равен 0b11001101

// ====== Битовое НЕ ======
~0 == 1
~1 == 0
myByte = 0b11001100;
myByte = ~myByte; // инвертируем
// myByte теперь 00110011
```

```
// ====== Битовое исключающее ИЛИ ======
// 0 ^ 0 == 0
// 0 ^ 1 == 1
// 1 ^ 0 == 1
// 1 ^ 1 == 0
myByte =
0b11001100;
myByte ^= 0b10000000; // инвертируем 7-ой бит
// myByte теперь 01001100

// ====== Битовый сдвиг
======myByte =
0b00011100;
myByte = myByte << 3; // двигаем на 3 влево
// myByte теперь 0b11100000

myByte >>= 5;
// myByte теперь 0b00000111

myByte >>= 2;
// myByte теперь 0b00000001
// остальные биты потеряны!
```

Указатели и ссылки

```
// полный урок тут: https://alexgyver.ru/lessons/pointers/

// & - возвращает адрес данных в памяти (адрес первого блока данных)
// * - управляет значением по указанному адресу

// === указатели ===
// управление переменной через
указатель byte b;      // просто
переменная типа byteb = 10; // b
теперь 10
byte* ptr; // ptr - переменная "указатель на объект типа
byte"ptr = &b;  // указатель ptr хранит адрес переменной
b
*ptr = 24; // b теперь равна 24 (записываем по
адресу &b)byte s;      // переменная s
s = *ptr;   // s теперь тоже равна 24 (читаем по адресу &b)

// === ссылки ===
// управление переменной через
ссылку byte b;   // просто переменная
типа byteb = 10;// b теперь 10
byte &link = b; // link - переменная "ссылка на объект типа
byte"link = 24;  // b теперь равна 24 (записываем через
ссылку)
byte s;      // переменная s
s = link;   // s теперь тоже равна 24 (читаем по ссылке)
```

Приложение 6

Программное обеспечение
необходимое для реализации программы

Fritzing:https://github.com/fritzing/fritzing-app/releases/download/CD-548/fritzing-3d61c58421bdb63ca903bb5d11310a257f1ec0ed-develop-548.windows.64.zip

Модули из GyverKIT для Fritzing:
https://github.com/AlexGyver/GyverKIT/archive/main.zip

Среда автоматизации проектирования электроники EasyEDA: https://easyeda.com/editor

ScratchDuino: https://appnapc.com/2184/

ArduBlock: http://ardublock.ru/ru/

mBlock: https://mblock.makeblock.com/en-us/download/

Онлайн сервис «Цепи» в Tinkercad: https://www.tinkercad.com

Bluino Loader
https://play.google.com/store/apps/details?id=com.bluino.bluinoloader&hl=ru&gl=US

Android RemoteXY:
https://play.google.com/store/apps/details?id=com.shevauto.remotexy.free

Virtuino:
https://play.google.com/store/apps/details?id=com.virtuino_automations.virtuino

Blynk: https://play.google.com/store/apps/details?id=cc.blynk

IoT Wi-Fi контроллер:
https://play.google.com/store/apps/details?id=com.nelso.iotremotecontroller

Справочник по Arduino:
https://play.google.com/store/apps/details?id=com.arduino_hb.Arduino_HandBook_FREE&hl=ru&gl=US

Приложение 7

Критерии оценки защиты творческого проекта

Автор(ы) проекта_____

Наименование проекта_____

Критерии оценивания:

- отметка «**5**»: проект выполнен полностью и правильно;

- отметка «**4**»: работа выполнена правильно с учетом 2-3 несущественных ошибок;

- отметка «**3**»: работа выполнена правильно не менее чем на половину или допущены существенные ошибки;

- отметка «**2**»: работа не выполнена.

Раздел	Критерий	Обоснование критерия	Баллы
ПРОЕКТ	Оригинальность и качество решения	Проект уникален и продемонстрировал творческое мышление участников. Проект хорошо продуман и имеет реалистичное решение / дизайн / концепцию.	
	Практическая значимость		
	Эффективность устройства	Устройство выполняет эффективную работу	
	Исследование	Команда продемонстрировала высокую изученности проекта, сумела четко и ясно сформулировать результаты исследования.	
	Уникальность проекта		
	Общий вид устройства		
	Техническая сложность		
	Оправданность применения тех или иных компонентов		
	Зрелищность	Проект имел восторженные отзывы, смог заинтересовать на его дальнейшее изучение.	
ПРОГРАММ ИРОВАНИЕ	Автоматизация	Проект работает автономно, с небольшим вмешательством человека. Роботы принимают решения на основе данных, полученных с датчиков.	
	Логика	Программа написана грамотно, выполнение	

		происходит логично на основе ввода данных с датчиков.	
ПРЕЗЕНТАЦИЯ	Успешная демонстрация	Проект работает так, как и предполагаюсь, с высокой степенью воспроизводимости.	
	Логичность представления	Качество выступления: • грамотная речь; • оформление презентации; • доступность. • артистичность • логика	
	Навыки общения и аргументация	Участники смогли рассказать, о чем их проект, иобъяснить, как он работает и ПОЧЕМУ они решили его сделать.	
КОМАНДНАЯ РАБОТА (приработе вкоманде)	Уровень понимания проекта	Участники продемонстрировали, что все члены команды имеют одинаковый уровень знаний о проекте.	
	Сплоченность коллектива	Команда продемонстрировала, что все участникиколлектива сыграли важную роль в создании и презентации проекта.	
Максимальное количество баллов			

Приложение 8

Диагностическая карта освоения программы

Дата заполнения «_____» _____ 20 __ год

Педагог дополнительного образования _____

№ п/п	Фамилия, имя обучающегося	Параметры			
		Основы робототехники, программирования, конструирования (теория)	Развитие личностных качеств, творческих способностей, общей культуры	Метапредметные навыки	Творческий проект (практика
		Высокий	Средний	Низкий	

Приложение 9

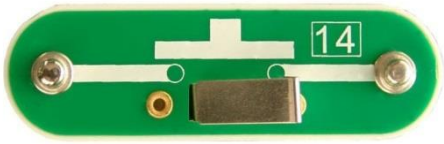
Конкурс задач «Знаток»

1. **Напишите название элемента**

Ответ_____

2. **Напишите название элемента**

Ответ_____



3. **Укажите номинал резистора**

А) 10 Ом

Б) 10000 Ом

В) 100 Вт

Г) 1000 Ом

4. **Какой из представленных элементов не является «транзистором»**

А)

Б)

В)

5. Соотнесите названия элементов

Диод

Катушка индуктивности

Светодиод

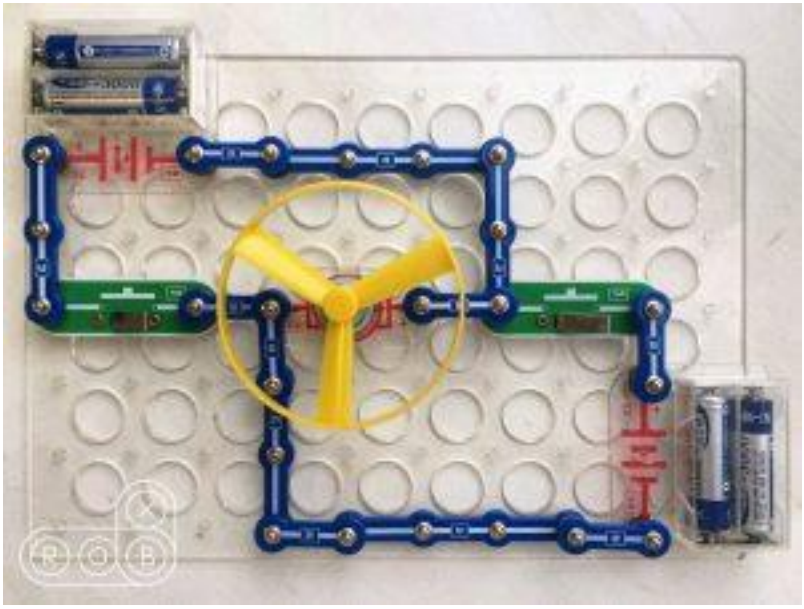
Конденсатор

Геркон

Ключ

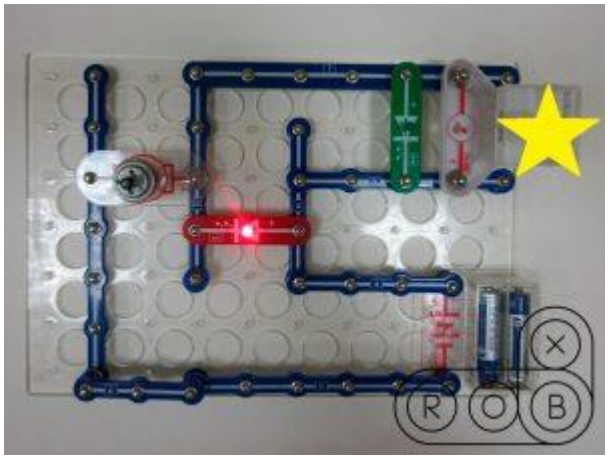
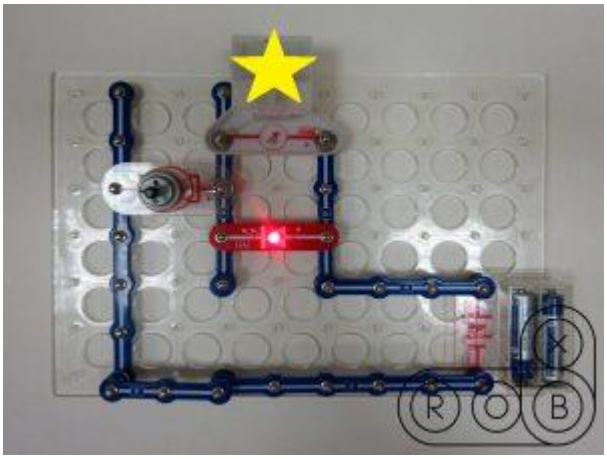
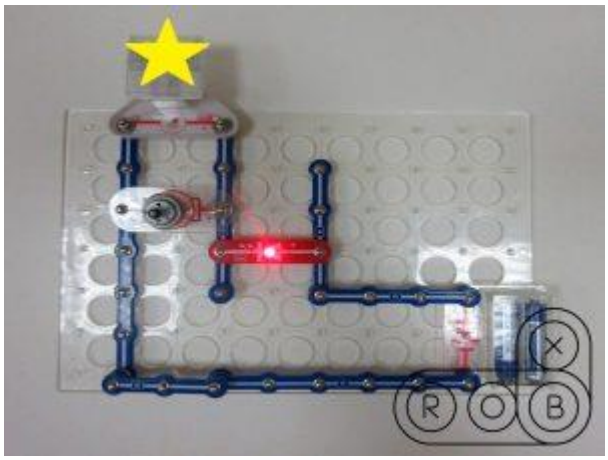
6. Необходимо заставить пропеллер взлететь. Укажите какой элемент в схеме стоит неправильно, и поменяйте его положение.

- почему в неисправленной схеме, когда включали две кнопки сразу, мотор начинал крутиться медленнее?

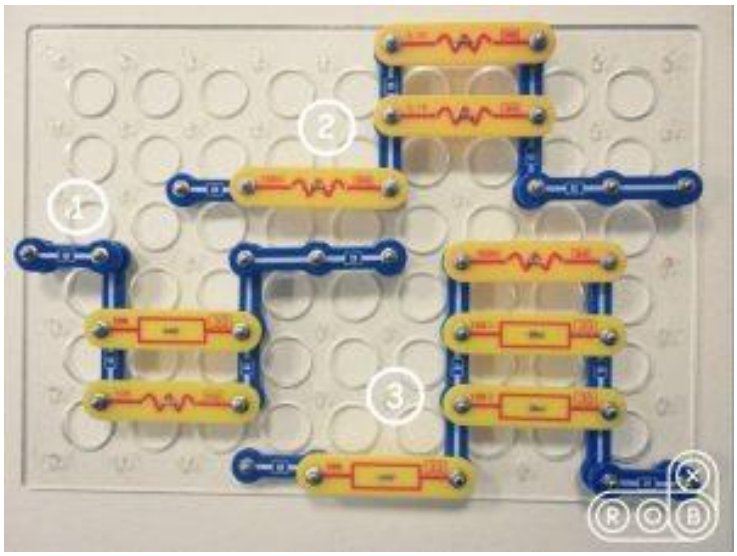


7. Соберите электрические цепи на конструкторе «Знаток», как показано на картинках. Внимательно рассмотрите их, и сравните показатели включенных в них гальванометров.

- Подсказка: Во всех случаях моторы не вращаются.
- Задание-бонус для самых умных: Уберите гальванометры и добавьте к схеме всего один провод так, чтобы мотор начал вращаться.

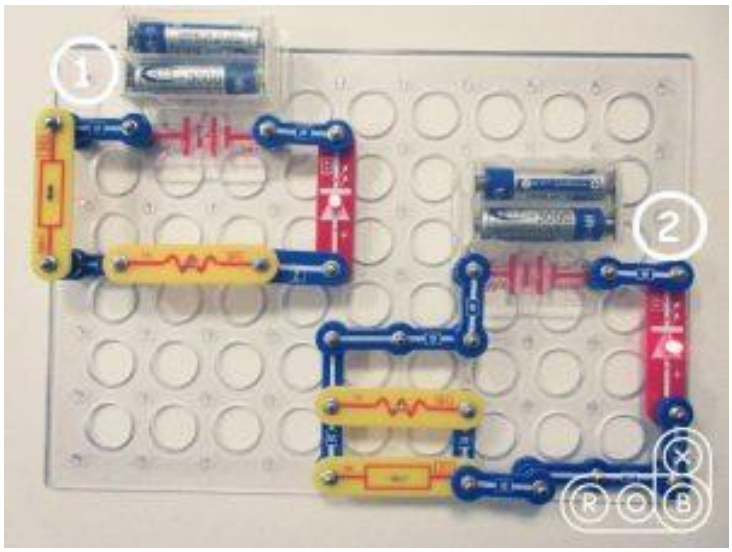




8. Задание: расположите в порядке возрастания по общему сопротивлению три фрагмента схем и объясните своё решение.



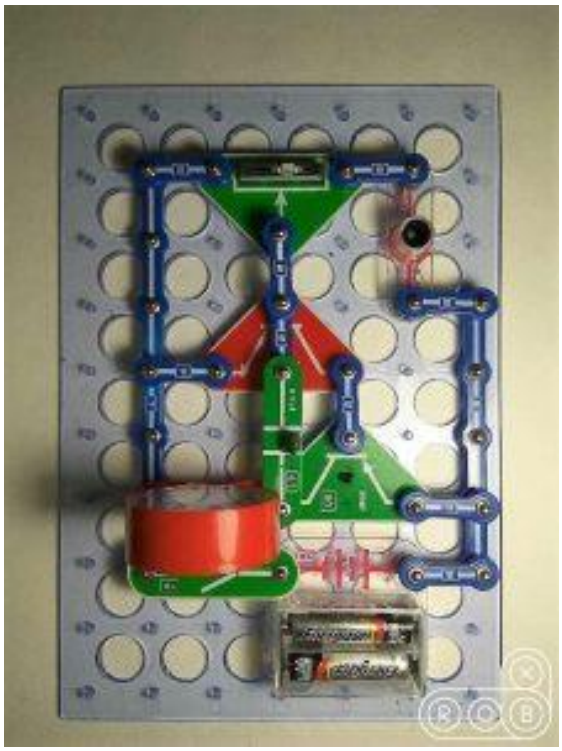
9. Соберите две схемы на конструкторе «Знаток», как показано на картинке. В этот раз мы предлагаем вам решить заковыристую задачку с формулой и ответить на вопросы:

- Какое общее сопротивление двух резисторов в каждой из схем (R).
- В какой схеме светодиод горит ярче и почему?



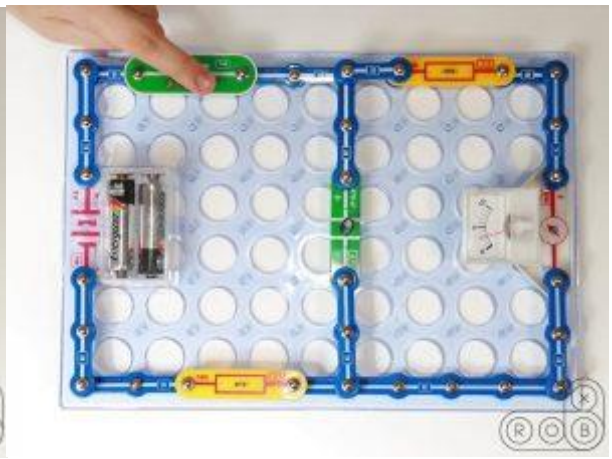
10. Соберите схему на конструкторе “Знаток”, как показано на картинке. Сыграйте любую мелодию на собранной схеме и попробуйте ответить на вопрос:

- Как заставить динамик поменять тембр звучания?



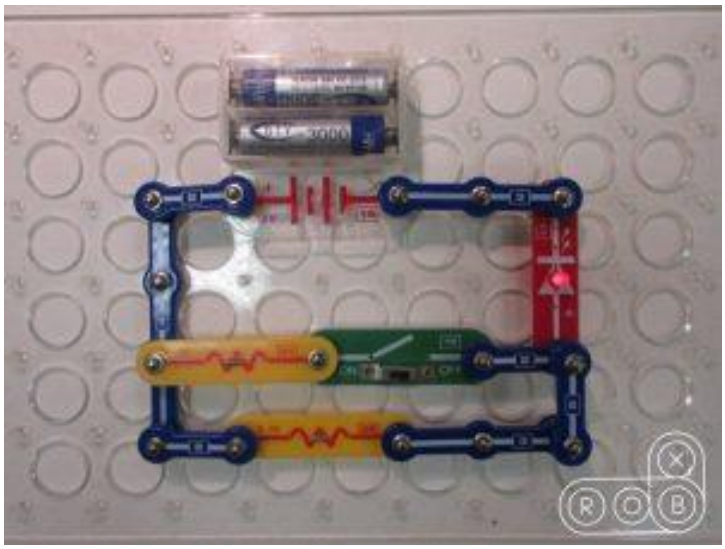
11. Соберите RC цепь на конструкторе “Знаток”, как показано на наших картинках, и посмотрите, как будет себя вести гальванометр, ответив на следующие вопросы:

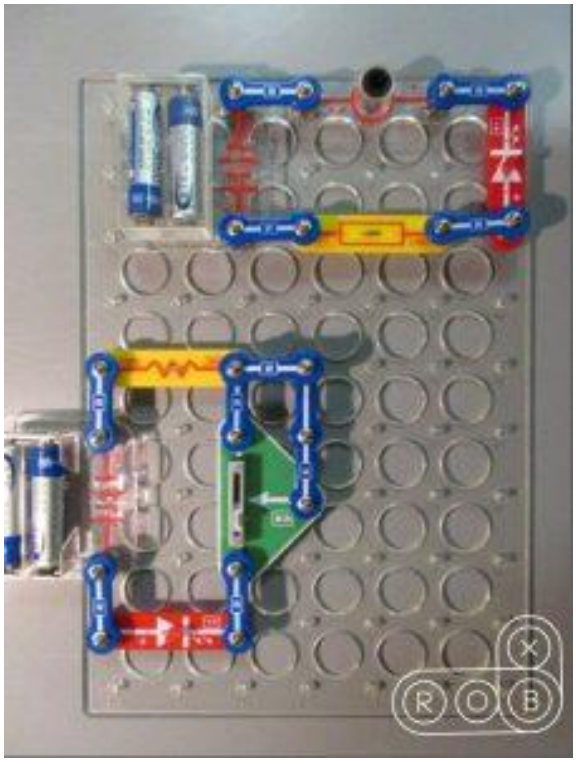
- Что происходит, если нажать и удерживать кнопку?
- До какого максимального значения дотягивается стрелка гальванометра?
- Что надо сделать, чтобы стрелка прибора удерживалась на значении “3”?
- Какому напряжению соответствует одно деление гальванометра?
- Как изменить схему, чтобы стрелка отклонялась не медленно и плавно, а резко и быстро?



12. Соберите три схемы, на конструкторе «Знаток».

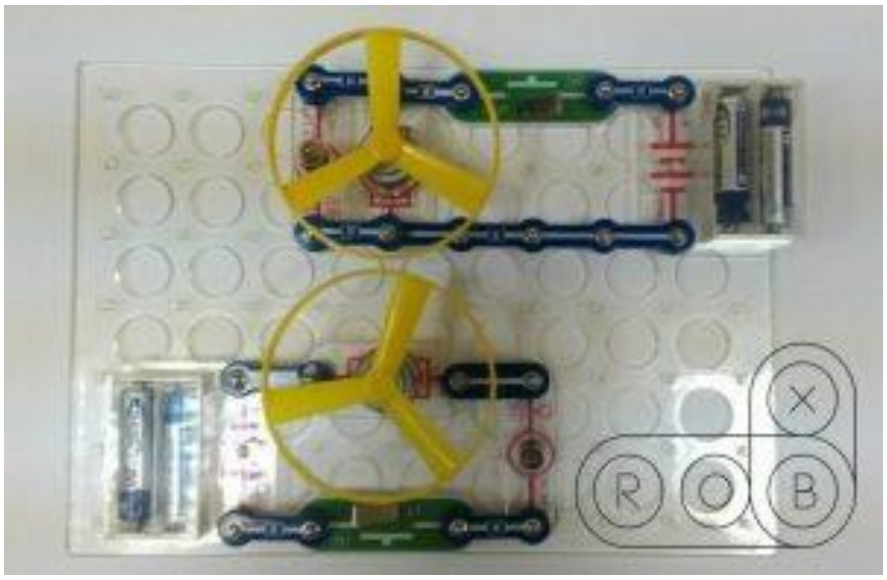
- Что общего у этих схем, и чем они отличаются.



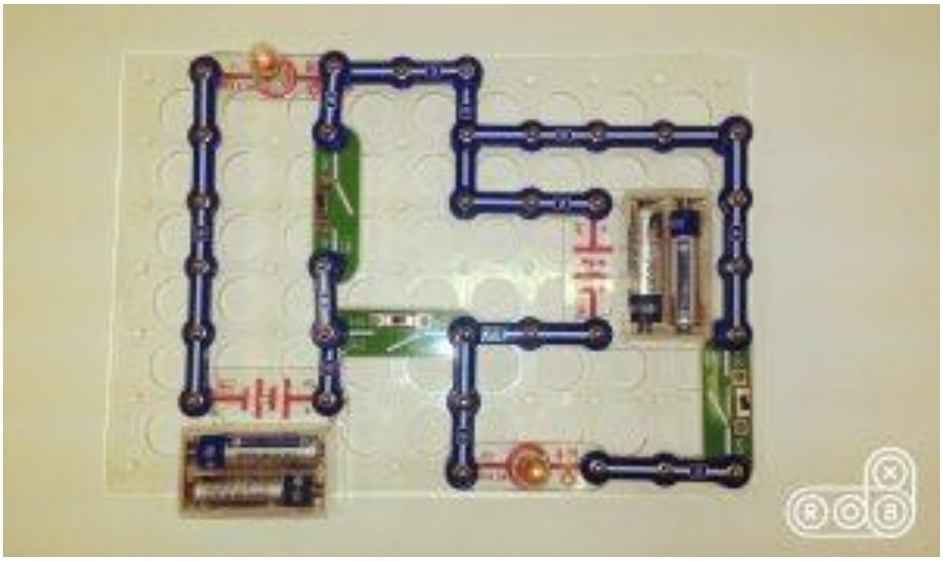


13. Необходимо сравнить две схемы подключения лампочки и мотора.

- В какой схеме моторчик будет вращаться быстрее и почему?
- Как будут светиться лампочки (ярче или тусклее), если снять пропеллер с моторчиков?



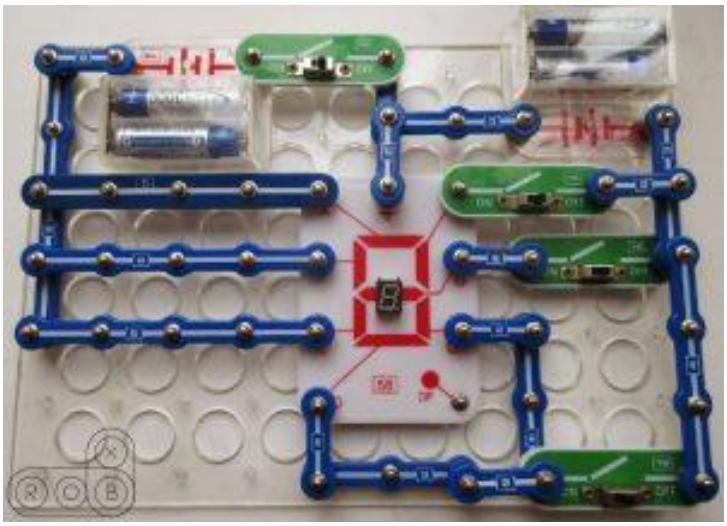
14. Какие переключатели нужно включить, а какие выключить, чтобы лампочка в левом верхнем углу загорелась максимально ярко?

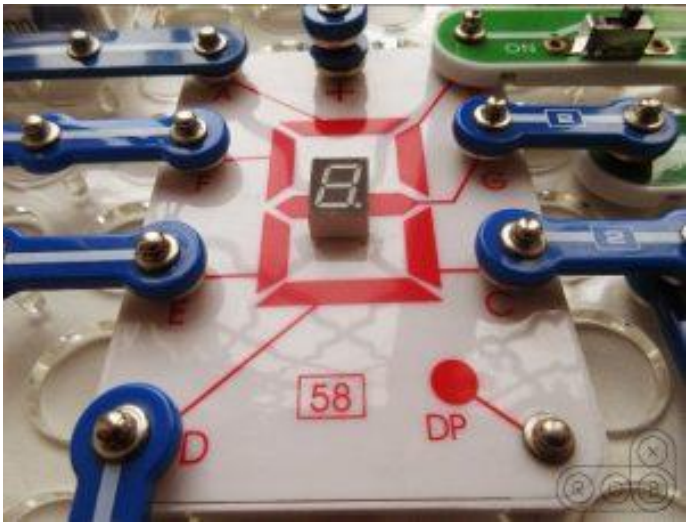


15. Что нужно сделать, чтобы красный светодиод зажёгся? И что нужно сделать, чтобы он горел то более ярко, то более тускло?

16. Найдите все русские буквы, которые можно зажечь на этой схеме.

- Сколько их?
  Составьте с помощью этих букв как можно больше слов.
- В слове буква должна встречаться только один раз.



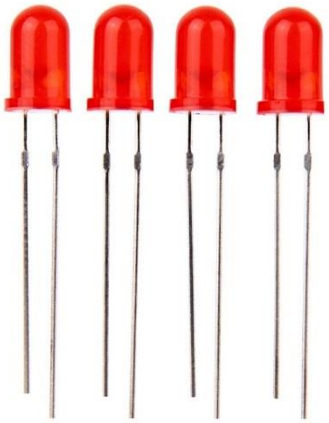


Приложение 10

Итоговое тестирование раздела

«Начало работы с платформой Arduino»

1. Какова правильная полярность подключения светодиода?



а Длинная ножка (анод) к «минусу» питания, короткая ножка (катод) – к «плюсу»
б Длинная ножка (катод) к «плюсу» питания, короткая ножка (анод) – к «минусу»
в Длинная ножка (анод) к «плюсу» питания, короткая ножка (катод) – к «минусу»

2. В чем необходимо обязательно убедиться перед загрузкой программы в контроллер Arduino?

а Выбран тип платы
б В коде созданы макроопределения
в Плата физически подключена к компьютеру
г Выбран порт, к которому подключена плата

3. Для назначения режима работы пинов Arduino используется:

а директива #define
б функция pinMode()
в функция digitalWrite()
г функция digitalRead()

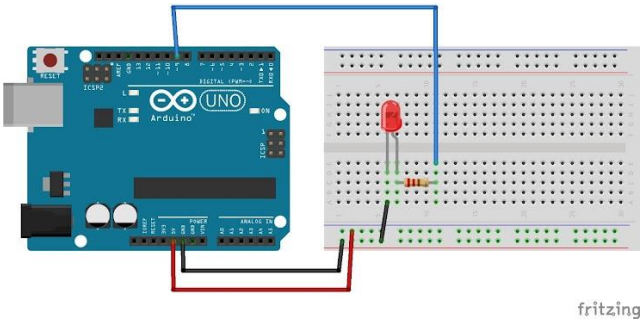
4. Процедура void setup() выполняется

а только один раз
б один раз при включении платы Arduino
в все время, пока включена плата Arduino

5. Как работает «=»?

а Это оператор сравнения
б Это оператор присваивания, он помещает значение, расположенное справа от него, в переменную, стоящую слева
в Это оператор присваивания, он делает оба операнда равными большему из них

6. Для какой цели в данной схеме используется резистор, последовательно соединенный со светодиодом?



а Для уменьшения силы тока, текущего через светодиод
б Для увеличения яркости свечения светодиодов
в Для увеличения силы тока, текущего через светодиод
г Для подавления шума на выводе кнопки

7. Функция delay()

а останавливает выполнение программы на заданное количество миллисекунд
б останавливает мигание светодиода на заданное количество миллисекунд
в останавливает выполнение программы на заданное количество секунд

8. Для считывания значений с аналогового входа используется команда

а digitalRead();
б delay();
в pinMode();
г analogWrite();
д analogRead();
е digitalWrite();

9. Для считывания значений с цифрового входа используется команда

а digitalRead();
б delay();
в pinMode();
г analogWrite();
д analogRead();
е digitalWrite();

10. В какой строчке не допущена ошибка?

а if (push==1) digitalWrite(13,HIGH);
б if (push>1); digitalWrite(13,HIGH);
в if (push>=1) digitalRead(13,1);
г if (push>=1) analogRead(13,500);

11. Что верно в отношении функции pinMode()?

а В эту функцию можно не передавать параметры
б Принимает параметром направление работы порта (вход или выход)
в Принимает параметром номер пина, который конфигурируется
г Эта функция нужна для конфигурации направления работы порта

12. Что следует помнить при создании переменной?

а Ей нужно задать тип
б Ей нужно выбрать имя
в Ей можно присвоить значение
г Имя состоит из латинских букв (обязательно начинается с нее), цифр и символов «_»
д Имя переменной нужно давать уникальное и осмысленное

е Это инструкция, должна заканчиваться «;»

ж Значение переменной нельзя будет изменить

13. Что верно в отношении функции digitalWrite()?

а В эту функцию можно не передавать параметры

б Уровень напряжения можно задать константами HIGH (напряжение питания, 5В для Arduino UNO) и LOW (0В)

в Принимает параметром уровень напряжения (высокий или низкий), который необходимо выставить на контакте

г В качестве выставляемого напряжения можно указать любое напряжение в диапазоне 0—5В

д Эта функция позволяет включать или выключать напряжение на определенном пине

е Принимает параметром номер пина, которым нужно управлять

14. Что означает появившаяся после компиляции программы ошибка «'PIN_1' was not declared in this scope»

а не закрыта скобка или нет точки запятой после PIN_1

б в функции pinMode() не использовано имя порта PIN_1

в в скетче не объявлена переменная PIN_1

15. Какие утверждения относятся к условному оператору if?

а условием может быть логическое выражение

б else позволяет определить действия, которые выполнятся, если условие ложно

в с помощью него можно задать условие, в зависимости от которого определенные действия будут или не будут выполнены

г внутри if нельзя использовать другой if

16. Каким образом измеряется сила тока с помощью мультиметра?

а Нужно включить мультиметр в режим прозвонки

б Нужно установить щуп в разъем мультиметра, соответствующий предполагаемым токам

в Нужно извлечь из мультиметра батарею

г Нужно включиться щупами в цепь последовательно

Нужно выбрать диапазон измерений (предполагаемую верхнюю границу)

17. К чему приведет выполнение следующего кода?

```
void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);

  digitalWrite(2, LOW);
  digitalWrite(3, LOW);

}

void loop() {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
}
```

а Напряжение на 2 и 3 пине будет включаться и выключаться

б Будет включено напряжение на 2 пине, затем оно будет выключено и включено на 3, на следующей итерации loop() напряжение выключится на 3 пине и вновь включится на 2

в Будет включено напряжение на 2 пине, затем на 3 пине

г Будет включено напряжение на 2 пине, затем оно будет выключено и включено на 3

18. Укажите диапазон входящего напряжения питания платы Arduino NANO

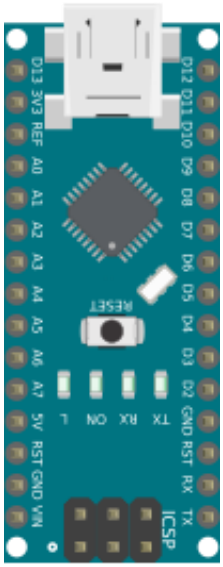
а 3,3 до 24 вольт

б 7 до 12 вольт

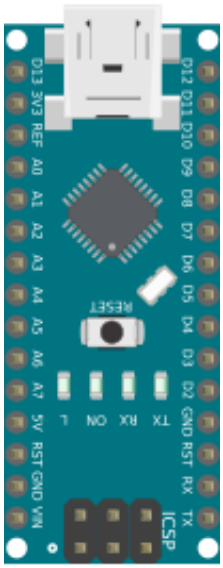
в 5 до 12 вольт

г 7 до 24 вольт

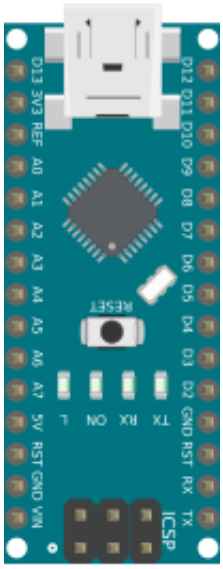
19. **Укажите следующие пины питания:**
    - Выходной пин 5V
    - Входной пин от 7 до 12 вольт
    - Выходной пин 3.3V
    - Выводы земли



20. Укажите цифровые и АЦП пины:



21. Укажите ШИМ пины:



22. Какую функция используется для выключения светодиода:

а digitalWrite(ledPin, LOW);
б digitalRead(ledPin, HIGH);
в digitalRead (ledPin, LOW);
г digitalWrite (ledPin, HIGH);

23. Какой функцией в программе можно назначить выводу порт ввода:

а pinMode(pin, INPUT);
б Serial.begin(9600);
в void loop (){}
г val = Serial.read ();