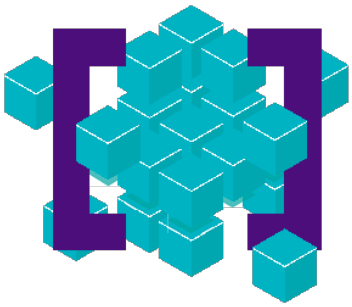


BIT EDUCATION
КАДРЫ ДЛЯ ЦИФРОВОЙ ЭКОНОМИКИ

Цифра в регионы

Применение робототехники при изучении предмета «Технология» в средней школе

Проект реализуется
в рамках федерального проекта
«Кадры для цифровой экономики»
национальной программы
«Цифровая экономика Российской Федерации»
государственной программы Российской Федерации
«Развитие образования»

2020

Благодарности

В данном пособии использованы результаты в области учебной робототехники, полученные специалистами компании «Научные развлечения» Дмитрием Илюшиным, Русланом Матвейчуком, членом делового Совета МФТИ и выпускником ФМХФ МФТИ Андреем Новиковым и преподавателем МФТИ Вячеславом Обуханом. Ссылки даны везде, где цитируются или упоминаются их работы.

Предисловие Директора «Физтех-лицея»

Наш “Физтех-лицей” готовит будущих специалистов в области естественных наук и передовых технологий. Мы решаем эту задачу в сотрудничестве с преподавателями МФТИ, а также других известных ВУЗов и школ России.

Конечно, мы стараемся обеспечивать нашим учащимся доступ к самому современному оборудованию и технологиям.

Одним из вызовов для технологического образования в средней школе является необходимость приобщения учащихся к современным технологиям. Не секрет, что существующие учебники по предмету «Технология» предоставляют прекрасную методическую базу для ремесленных профессий. В то же время предстоит ещё много сделать для создания адекватной методической базы для изучения современных цифровых технологий.

Данное пособие является попыткой обобщения нашего опыта преподавания робототехники в формате кружка для школьников и в рамках курсов повышения квалификации для учителей предмета “Технология”. Мы, таким образом, видим в учебной робототехнике один из возможных ответов на задачи, поставленные в Распоряжении Минпросвещения России от 1 ноября 2019 года за номером Р-109 «Об утверждении методических рекомендаций для органов исполнительной власти субъектов Российской Федерации и общеобразовательных организаций по реализации Концепции преподавания предметной области «Технология».

Действительно, робототехника позволяет в предметной форме познакомить учащихся и помочь им приобрести базовые навыки работы

с современным технологическим оборудованием, освоить современные сквозные цифровые технологии и обеспечить преемственность переходом от общего образования к среднему профессиональному.

Очень важно при этом не подменять самую суть и методику обучения технологиям другим видом обучения – а именно, как пользоваться техническими средствами обучения (ТСО). ТСО нужно уметь пользоваться, и они сами по себе представляют большое поле для деятельности, которые требуют развития и подготовки собственных специалистов. Однако одно лишь умение использовать сложные установки для, например, физических экспериментов без понимания физической сути явлений, без определенной культуры физического эксперимента не достигает главной цели: подготовки всесторонне развитого специалиста.

Робототехника представляет собой область, в которой можно осознанно ставить такие прикладные задачи, как создание измерительных комплексов для физических экспериментов, прототипов автоматизированных рабочих мест (АРМ) для высокотехнологических отраслей и многие другие. При этом естественным образом создаются условия для метапредметного обучения и постановки перед учащимися междисциплинарных задач как учебного, так и проектного характера.

Мы надеемся на продолжение сотрудничества с нашим партнерами, в особенности с учителями школ России, и надеемся, что обратная связь от них поможет улучшить данное пособие.

Машкова М.Г.

Предисловие от «Научных развлечений»

Данное методическое пособие появилось в результате нашего сотрудничества с АНОО “Физтех-лицей” по двум направлениям:
- организация курсов повышения квалификации для преподавателей предмета “Технология” в 2020 году;
- организация кружка робототехники для школьников.

Решения, которые мы предложили в ходе этого сотрудничества,

используют как нашу экспертизу, так и достижения нашего партнера, компании ArTec Co., Ltd. (ArTec) - крупнейшего игрока на рынке учебного оборудования Японии.

Уже около 30 лет инженеры и методисты нашей компании разрабатывают и производят в России учебное оборудование для средних школ и вузов. Наш каталог включает широкую линейку датчиков для различных измерений, а также оборудование для изучения естественных наук в вузах, школах и детских садах.

Одно из наших решений - цифровая лаборатория для дошкольников и младших классов - вызвало интерес не только в России, но и в таких странах, как Франция и Япония. Благодаря этому началось наше сотрудничество с японской компанией ArTec, которая в том числе разрабатывает и производит наборы для сборки учебных роботов. Именно они стали основой учебной базы для вышеупомянутых кружка и курсов на базе “Физтех-лицея”.

Этот выбор объясняется несколькими причинами:
- Наборы ArTec используют плату Studuino®, которая является усовершенствованным вариантом широко распространенной платы Arduino®.
- Важным достоинством продуктов ArTec является очень развитая методическая база с объяснениями, инструкциями и упражнениями, которую мы перевели на русский язык и адаптировали для использования в российских школах и детских садах.
- В основе наборов - надежная и гибкая конструктивная среда из блоков-кубиков, моторов и датчиков. Она позволяет реализовывать большое количество оригинальных устройств от простейших роботов для дошкольников до действующих прототипов сложных установок для обучения старшеклассников и студентов вузов.

Благодаря этим качествам наборов ArTec нашей компании удалось разработать и реализовать на их основе ряд таких оригинальных устройств, как:
- установка для изучения аэродинамики и принципов управления летающим квадрокоптером;
- рабочее место оператора АЭС;
- теплица с управлением через среду IoT;
- установка для автоматического химического анализа жидкостей;

- установок, моделирующих прецизионные механические движения и других.

Эти решения оказались достаточно новаторскими. Мы полагаем, что они могут быть востребованы при обучении в средней школе по предметам “Технология” и “Информатика”.

Наш опыт показал, что сборный, модульный характер наборов ArTec позволяет создавать действующие прототипы реальных технологических устройств и решать учебные задачи их эксплуатации и программирования. Немаловажно и то, что такие установки для использования в школах просты, долговечны, легко ремонтируются и достаточно экономичны.

Олег Поваляев,
к. ф-м. н.
Лауреат Государственной премии РФ
в области образования, 2015 г.
Генеральный директор ООО «Научные развлечения»

Содержание

1 Робототехника в курсе предмета технология для 4-6 классов. 13

1.1 Занятие 1. 13

1.2 Занятие 2. 15

1.3 Занятие 3. 17

1.4 Занятие 4. 18

1.5 Занятие 5. 19

1.6 Занятие 6. 22

1.7 Занятие 7. 23

1.8 Занятие 8. 25

1.9 Занятие 9-10. 28

1.10 Занятие 11-12. 32

1.11 Занятие 13. 35

2 Практические работы с технологическими установками. 38

2.1 Занятие 1-2. 39

2.2 Занятие 3-4. 48

2.3 Занятие 5-6. 55

2.4 Занятие 7-8. 64

2.5 Занятие 9. 73

2.6 Краткое содержание урока 75

2.7 Занятие 10. 76

2.8 Занятие 11. 78

2.9 Занятие 12. 80
2.10 Занятие 13-14. 84
2.11 Занятие 15-16. 89

Введение

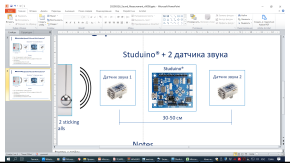
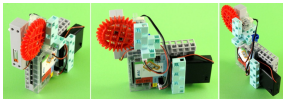
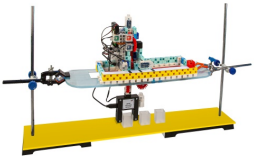
Робототехника в данной брошюре рассматривается как инструмент для преподавания предмета «Технология» в средней школе. Сама по себе технология робототехники является безусловно востребованной составляющей и того, что называется передовым краем технического прогресса, и того, что называется современным производством. Самые разнообразные роботы находят все большее применение в транспорте, автоматической сборке, медицине, дистанционном сборе информации и так далее. Расширяются их функциональные возможности, включая, в том числе, возможности принятия самостоятельных решений.

В 1-й части данного пособия будет предложена программа изучения азов робототехники из 16 уроков, с примерами конструкций роботизированных устройств на базе робот-наборов АрТек. Данное пособие не ставит целью заменить все специализированные методические пособия, специально созданные для этих наборов. Вместо этого предлагается проверенная на практике программа занятий, с кратким описанием основных тем, учебных задач и достигаемых целей.

Уроки 1-й части предназначены для начинающих знакомиться с робототехникой и могут быть предложены ученикам 5-7 класса, особенно не имеющим опыта программирования.

Во второй части пособия предлагается программа из 16 уроков для углубленного изучения отдельных вопросов, связанных с созданием высокотехнологических устройств.

Хочется подчеркнуть, что по-настоящему гибкие наборы для сборки роботов предоставляют хорошую базу для ряда важных направлений:

№	Направление	Примеры	Комментарии
1	Создание измерительных стендов для изучения различных физических явлений		(*) Летающий квадрокоптер в комплекте с кардановом подвесе. Предназначен для изучения элементов аэродинамики и теории управления, азов программирования дронов
2			(*) Стенд для измерения скорости звука в воздухе. Отработка техники физического эксперимента
3	Элементы прецизионных устройств		(*) Стенд для отработки точных движений роботов в осложненных условиях работы
4	Прототипирование рабочих мест операторов высокотехнологического производства		Рабочее место оператора АЭС
5	Установки для междисциплинарных занятий		Теплица с дистанционным управлением
6			(*) Установка для поточного химического анализа жидкостей.

Во 2-й части настоящего пособия будут предложены уроки, где учащиеся сами могут собрать установки отмеченные (*), провести с ними опыты, получить навыки программирования. Эта часть программы рассчитана на возрастную группу 14-17 лет.

Эти установки представляют собой оригинальные изделия, не имеющие аналогов в настоящее время. Для работы с ними созданы специальные методические материалы, выдержки из которых использованы в данном пособии.

Почеркнем, что наборы АрТек для создания учебных роботов оказались достаточно гибкими, чтобы на их базе построить все это разнообразие автоматизированных, программируемых устройств.

В конце второй части будут также представлены примеры менее «гибких», более специализированных наборов, предназначенных исключительно для отработки движений человеческой фигуры. Мы полагаем, что такие задачи обладают определенной самостоятельной ценностью. Это наборы (1) - Двуногий робот Gekko «Ходок» и (2) KONDO KHR-3HV и «Робокит» на базе KHR-3HV.

В 3-й части пособия будут рассмотрены методы анализа и сравнения учебных проектов. Включение этого материала вызвано тем, что робототехника и «Технология» как предмет в целом, как правило, являются хорошим источником различных учебных проектов. Для того, чтобы с ними работать, часто требуются управленческие методы анализа и принятия решений, не достаточно представленные в имеющейся учебной литературе для школ.

Обзор имеющейся методической литературы для работы с роботами ArTec

Всего имеется три серии пособий для роботов ArTec, рассчитанные на разный возраст учащихся. Все пособия включают очень подробные инструкции по сборке, а где необходимо – по программированию с пояснением теоретических моментов.

1. Серия «Азбука робототехники» включает:

- Три книги, посвященные теме «Конструирование роботов», где ставятся учебные задачи по созданию роботоподобных механизмов без управляющей платы. Эти задания позволяют освоить все механизмы движения роботов от источника тока, а также многие базовые конструкции и соединения. При этом в схеме отсутствует управляющая плата, а запуск-остановка робота происходит при включении/выключении электропитания. В общей сложности 16 разных видов конструкций.
- Три книги, посвященные теме «Пиктограммное программирование», где на примере 14 различных конструкций ученики познакомятся с основами программирования в пиктограммной среде для самых начинающих.

2. Серия «Основы программирования роботов» также состоит из трех книг, посвященных программированию сервомоторов, датчиковых систем и роботов с обычными двигателями. Программирование осуществляется как в пиктограммах, так и в среде блочного программирования, сходной со Scratch.

3. Серия «Искусство программирования роботов» состоит из 4 книг для учащихся и одного руководства для учителя. Эта самая продвинутая серия книг рассказывает о 24 интересных и сложных робот-конструкциях.

Литература

1. Академия Наураши: Азбука робототехники. Конструирование роботов: Учебное пособие для детей от 6 лет. Ч. 1-3 / С. И. Мусиенко, Х. Дайчи, О. Казухей, К. Масаки, У. Аири. — М. : Де‘Либри, 2019.

2. Академия Наураши: Азбука робототехники. Пиктограммное программирование: Учебное пособие для детей от 6 лет. Ч. 1-3 / С. И. Мусиенко, Х. Дайчи, О. Казухей, К. Масаки, У. Аири. — М. : Де‘Либри, 2019.

3. А. Цуцких, К. Охаси. "Основы программирования роботов"в 3 ч. Рабочие тетради для детей 10-12 лет / Перевод Цуцких А. Ю.

— М. : Де‘Либри, 2019.

4. "Искусство программирования роботов". Ч. 1-4. Учебное пособие. / Перевод А. Ю. Цуцких. — М. : Де‘Либри, 2019.

1 Робототехника в курсе предмета технология для 4-6 классов.

1.1 Занятие 1.

Знакомство с набором АрТек для конструирования роботов. Примеры простейших роботоподобных конструкций без необходимости программирования (пропедевтика робототехники).

Литература: [1], Ч.1, стр. 5-60.

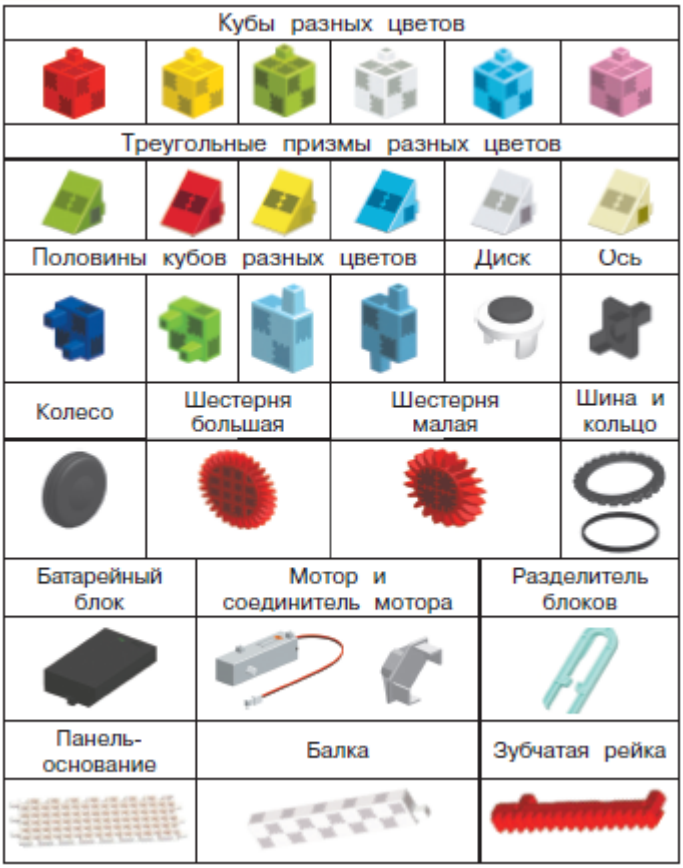


Рис. 1: Основные строительные элементы

Цели занятия

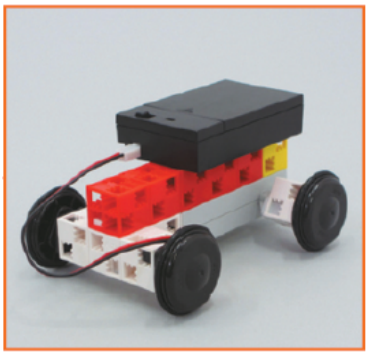
Введение в создание простейших конструкций с целью ознакомления со строительными элементами, применением принципа рычага, шатунной и зубчатой передач.

Тренируемые навыки и приобретаемые знания

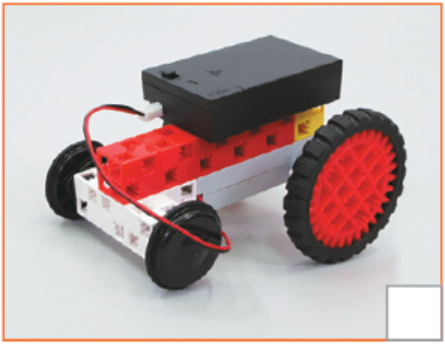
- Сборка конструкций согласно предлагаемой схеме за заданной время с последующей разборкой.
- Свойства зубчатых передач, понятие о передаточных соотношениях.
- Понятие о движении по маршруту (время движения между остановочными пунктами, скорость движения).
- Зависимость скорости от размера ведущего колеса.

Краткое содержание урока

Сборка основных конструкций согласно предложенным схемам:



Управляемая машина



Машина с укрупненными колесами

Робот-сумоист

Механический дятел

Отметим, что механический дятел является заготовкой для музыкальной шкатулки – программируемой конструкции, которая может быть предложена опытным учащимся старшего возраста.

1.2 Занятие 2.

Продолжение знакомства с набором АрТек для конструирования роботов. Примеры роботоподобных конструкций без необходимости программирования (пропедевтика робототехники).

Литература: [1], Ч.1, стр. 61-80.

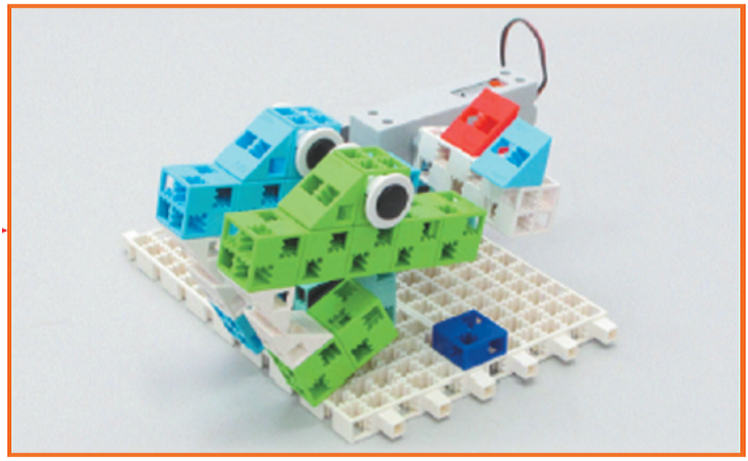
Цели занятия

Сборка модели «Кусачего крокодила» с целью ознакомления с принципами работы музыкальной шкатулки. Применение штифтов.

Тренируемые навыки и приобретаемые знания

- Сборка конструкций согласно предлагаемой схеме за заданной время с последующей разборкой.
- Задание ритма движения с помощью мотора.
- Способы запоминания информации в виде валика музыкальной шкатулки.
- Различные конструкции штифтов.

Краткое содержание урока



На этом занятии собирается модель «Кусачий крокодил», приводимый в действие мотором постоянного тока. Ритм открывания челюстей модели задается сменяемыми штифтами разной формы наподобие зубцов валика музыкальной шкатулки. См. пример штифта в виде синей и красной призмы в верхнем правом углу фото модели. Учащимся предлагается поэкспериментировать со штифтами собственной конструкции.

1.3 Занятие 3.

Продолжение знакомства с набором АрТек для конструирования роботов. Примеры роботоподобных конструкций без необходимости программирования (пропедевтика робототехники).

Литература: [1], Ч.2, стр. 6-22.

Цели занятия

Создание установки для запуска бумажных самолетиков с помощью резинки. Обсуждение принципов накопления энергии в упругих предметах.

Тренируемые навыки и приобретаемые знания

- Сборка конструкций согласно предлагаемой схеме за заданной время с последующей разборкой.
- Энергия упругих предметов (резиновый аккумулятор).
- Прицеливание и прицельная стрельба.
- Понятие и важность симметрии (сборка самолетиков).

Краткое содержание урока

Данная конструкция является самой простой установкой для запуска бумажных самолетиков. Она является первым шагом для учащихся в приобретении опыта с такими изделиями. Всего же предлагается 4 варианта такой установки нарастающей сложности. Наиболее сложный вариант содержит плату, несколько двигателей для прицеливания по наклону и азимуту и джостик с акселерометром. Джостик также собирается учащимся.

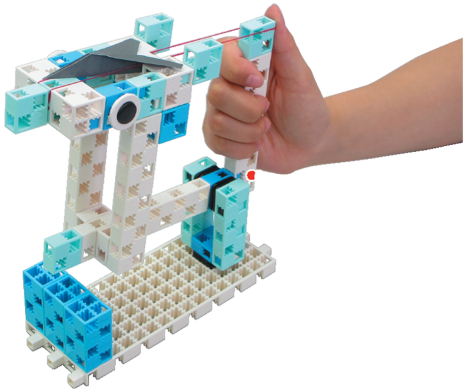


Рис. 2: Сборка машинки для запуска самолетиков согласно предлагаемой схеме

1.4 Занятие 4.

Завершение вводных занятий с набором АрТек для конструирования роботов. Примеры конструкций для транспортировки.

Литература: [1], Ч.3, стр. 5-29, 44-54.

Цели занятия

Создание установок: (1) на колесах с шатунным приводом; (2) на колесах с зубчатым приводом (3) шагающей машины.

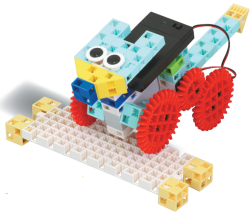
Тренируемые навыки и приобретаемые знания

- Сборка конструкций согласно предлагаемой схеме за заданной время с последующей разборкой.
- Сравнительные характеристики шагающих и колесных транспортных средств.
- Знакомство с шатунным и зубчатым механизмами для передачи движения.

Краткое содержание урока

Сборка трех моделей устройств с разными принципами приведения в движение. Сравнение их свойств.

Пегас с шатунным приводом



Обезьянка с зубчатой передачей

Шагающая утка

Во время урока учащиеся могут заполнить таблицу для сравнения трех моделей по параметрам скорости, устойчивости, преодоления препятствий и другим.

1.5 Занятие 5.

Знакомство с платой Studuino®. Начало работы со средой программирования в пиктограммах.

Литература: [2], Ч.1, стр. 5-40.

Цели занятия

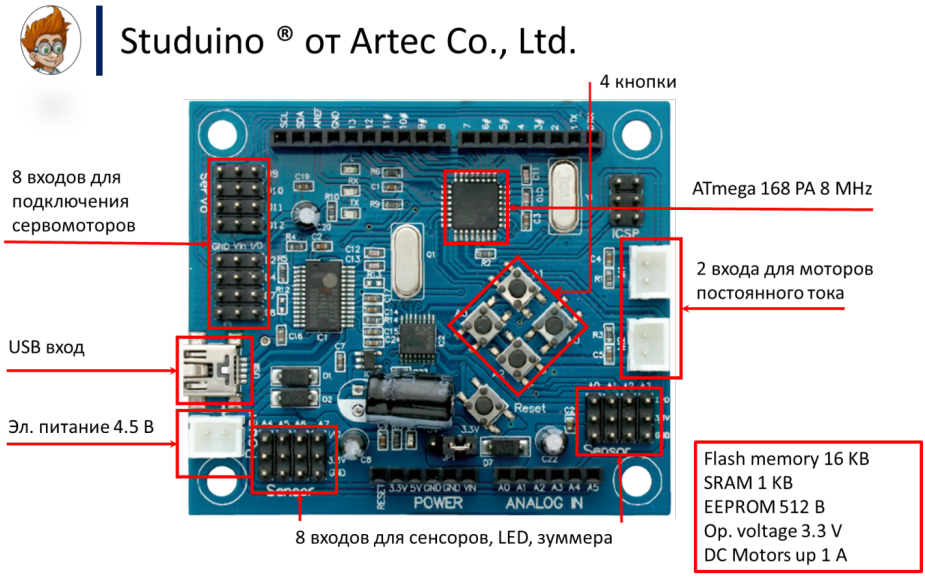
Основной функционал платы Studuino®. Знакомство с интерфейсом программной среды. Написание простых программ с использованием датчика звука.

Тренируемые навыки и приобретаемые знания

- Основные характеристики платы Studuino®. Плата – это маленький компьютер.

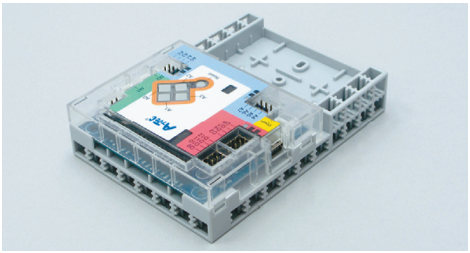
- Программирование – способ дать задание компьютеру.
- Правила подключения датчика к плате.
- Запись линейной программы без ветвлений.
- Запись условия на включение датчика (реакция на хлопок ладоней).

Краткое содержание урока

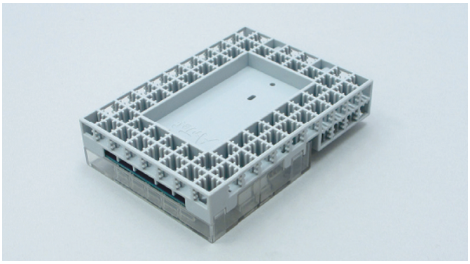


Плата Studuino® является развитием широко распространенной платы Arduino®. В руки учащимся она выдается внутри специального корпуса, удобного для присоединения конструкционных элементов.

Вид сверху



Вид снизу

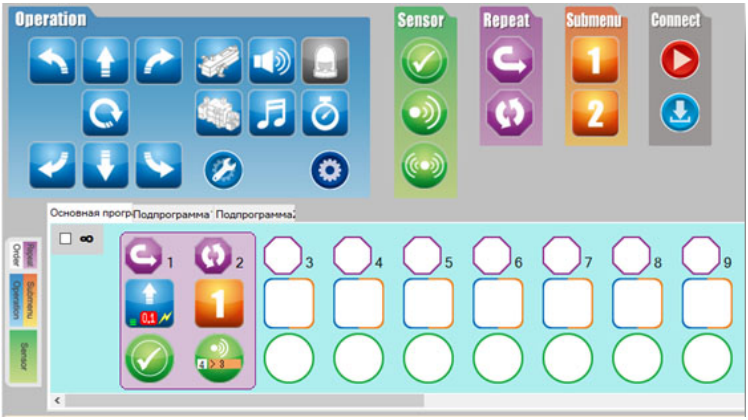


По сравнению с обычной Arduino® добавлен драйвер управления DC двигателями, выведена линейка под разъемы на 3 контакта (стандартный BLS-3). На каждом таком разъеме присутствует питание: плюс и минус, а также один из выводов платы Arduino. Все разъемы подписаны. В качестве исполнительных механизмов производитель предлагает два DC мотора с редукторами и до 8-ми сервоприводов собственной конструкции (их количество может меняться в зависимости от комплектации набора). Моторы подключаются через специальные разъемы, которые есть на плате.

Датчики – «глаза и уши» любого робота – подключаются к центральному блоку с помощью кабелей, которые идут в комплекте с набором. Для закрепления датчиков в модели робота на корпусах у них есть выступы, которыми они могут крепиться к конструкции.

Пиктограммное программирование осуществляется через удобный, интуитивно понятный интерфейс:

Группа пиктограмм «Операторы» дает простой способ задания движения, позволяет делать задержки, включать и выключать двигатели, управлять зуммером. На экране показана программа из двух операторов: (1) робот едет вперед в течение 0,1 сек и (2) запускается подпрограмма номер 1 при условии, что датчик звука регистрирует значение выше порога 1. Эти два действия повторяются заданное количество раз.



1.6 Занятие 6.

Сборка и программирование самоходной машины с одним двигателем для движения по одномерному маршруту с заданными остановками.

Литература: [2], Ч.1, стр. 41-61.

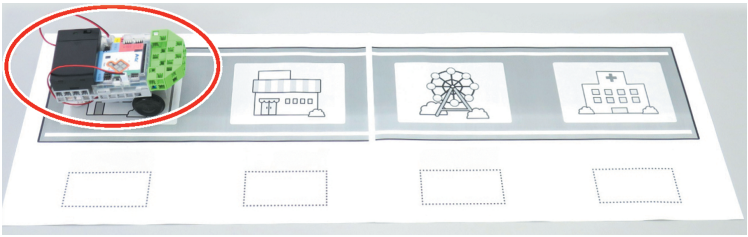
Цели занятия

Разработка и изображение линейного маршрута. Сборка самоходной машинки. Программирование и подбор времени движения между остановочными пунктами.

Тренируемые навыки и приобретаемые знания

- Подбор программных параметров (времени и скорости движения)
- Проведение повторных экспериментов по движению по маршруту.
- Ведение записи по результатам экспериментов.
- Сохранение программы в файле.

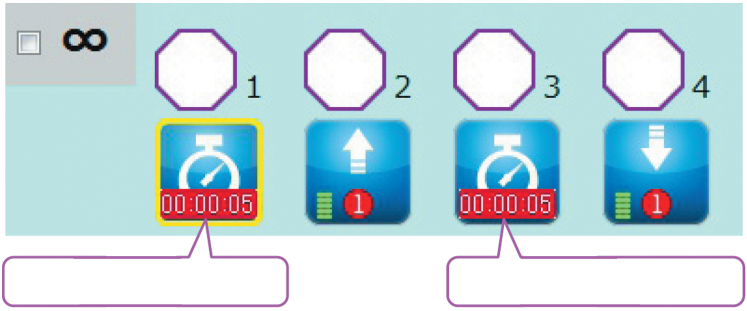
Краткое содержание урока



Учащиеся самостоятельно разрабатывают маршрут движения, остановочные пункты и интервалы движения. Для удобства эта информация может быть организована в таблицы вида:

Шаг	Остановка	Время стоянки, с
1	Дом	3
2	Супермаркет	5
3	Аттракционы и т.д.	10

Затем эта таблица переводится на язык, понятный компьютеру в виде последовательности команд:



1.7 Занятие 7.

Сборка и программирование самоходной машины с двумя двигателями для движения по двумерному маршруту с заданными остановками.

Литература: [2], Ч.1, стр. 62-73.

Цели занятия

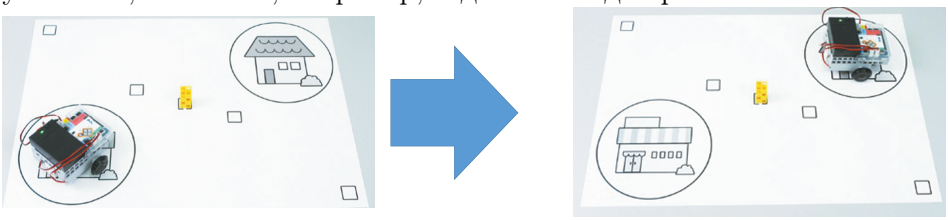
Разработка и изображение двумерного маршрута. Сборка самоходной машинки с двумя двигателями для осуществления поворотов. Программирование и подбор времени движения между остановочными пунктами.

Тренируемые навыки и приобретаемые знания

- Подбор программных параметров (времени и скорости движения)
- Проведение повторных экспериментов по движению по маршруту.
- Ведение записи по результатам экспериментов.

Краткое содержание урока

Данное занятие в основном повторяет Занятие 6 с более сложной схемой движения. Новыми видами задач, которые можно предложить ученикам, являются, например, задачи объезда препятствий.



Пример задания

Доберитесь на машине от супермаркета до дома, не задев блок, расположенный на пути её следования.

1.8 Занятие 8.

Сборка и программирование самоходной машины с двумя двигателями для рисования. Знакомство с понятием цикла.

Литература: [2], Ч.1, стр. 74-83.

Цели занятия

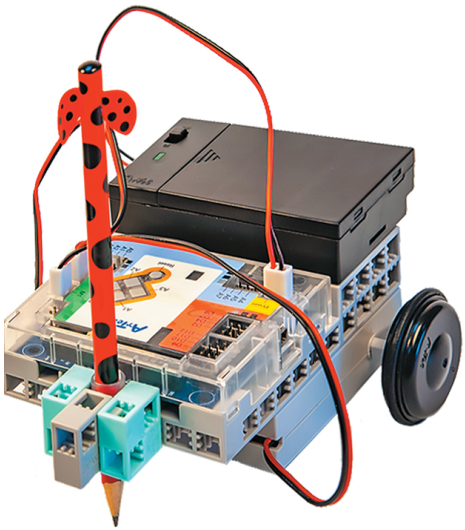
Сборка самоходной машинки с двумя двигателями для рисования. Программирование и подбор команд для получения желаемых рисунков.

Тренируемые навыки и приобретаемые знания

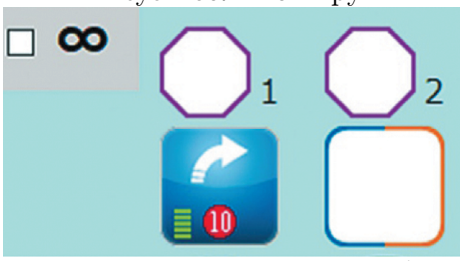
- Подбор программных параметров (времени и скорости движения) для сложных задач движения.
- Понятие о работе плоттеров и других устройств для автоматического воспроизведения изображений.
- Ведение записи по результатам экспериментов.
- Освоение альтернативной записи элементов рисунка через команды-пиктограммы. Тренировка символьной записи элементов рисунка.

Краткое содержание урока

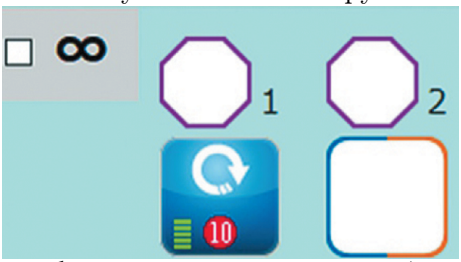
Собирается машинка с присоединенным пером (фломастером):
На этом занятии ученики продолжают осваивать возможности, заложенные в робот-наборы. После того, как учащиеся освоят простые команды для рисования правильных фигур, можно поставить задачу создать словарь, в котором разным элементам ставится в соответствие набор команд.



Рисуем большой круг



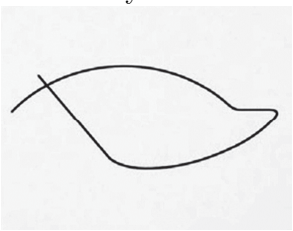
Рисуем маленький круг



Так, лепесток цветка может быть изображен с помощью всего 4 команд:

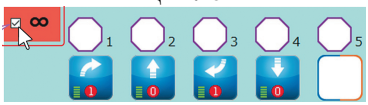
Программа из 4 команд

Результат

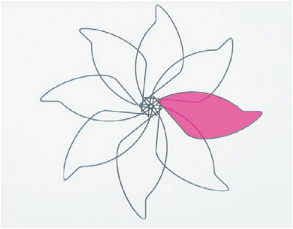


Первое движение выполняется 1,7 секунды, второе 0,2 секунды, третье 1,2 секунды, а четвертое 0,7 секунды. Далее естественно поместить эти 4 команды в цикл. Результатом выполнения такой программы будет цветок:

Программа из 4 команд в цикле



Результат

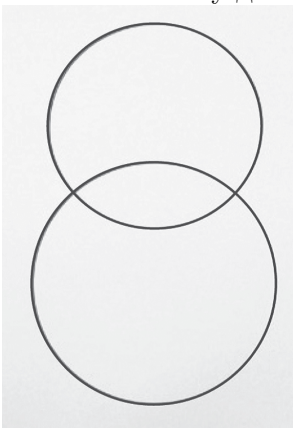


Обратите внимание: чтобы получить бесконечный цикл, нужно поставить галочку рядом с горизонтальной восьмеркой – математическим символом бесконечности. Для конечного цикла применяются значки начала цикла и конца – см. пример в Занятии 5:

Первая команда конечного цикла

Последняя команда конечного цикла

Количество повторений можно будет задать, кликнув по телу цикла правой кнопкой мыши. В рамках этого занятия можно предложить создать словарь графических элементов, из которых состоит какой-либо объект, а потом перевести этот словарь в язык команд. В качестве примера можно привести снеговика, и предложить нарисовать с помощью машинки заготовку для его фигуры:



Занятия 9-12 проводятся в рамках изучения программирования светодиодов LED, зуммеров и датчиков. Программирование осуществляется в среде блочного программирования, напоминающего Scratch. На занятиях собираются модели светофоров с различными функциона-

лами, а также модели светомузыкальных устройств. Занятие 9 -10 и Занятие 11-12 являются сдвоенным и состоят из 2-х уроков по 45 мин. При необходимости они могут, конечно, проводится отдельно.

1.9 Занятие 9-10.

Изучение работы современного светофора и его представление в виде блок-схемы. Устройство светодиода. Звуковые сигналы светофора. Кнопочное управление уличным светофором. Другие устройства сигнализации в современной городской среде. Программирование пешеходного светофора с дополнительными функциями (кнопка, звуковой сигнал).

Литература: [3], Часть «Программирование датчиков» стр. 3-50.

Цели занятия

Изучение работы светофора. Создание блок-схемы для его работы. Роль программирования в управлении городскими сигнальными устройствами и сервисными устройствами в целом (например, автоматами по продаже транспортных билетов, напитков). Связь блок-схемы и программирования.

Тренируемые навыки и приобретаемые знания

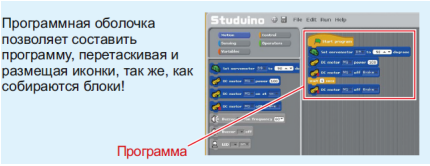
- Создание списка робото-технических устройств вокруг нас (стиральная машина, смартфон, микроволновка и т.д.)
- Изучение работы уличных сигнальных и обслуживающих устройств в современной городской среде.
- Функции компьютера как устройства для выполнения повторяющихся действий в быту, на улице, на производстве
- Ведение записи по результатам экспериментов

- Понятие о настройки портов платы.
- Знакомство с условными операторами.

Краткое содержание урока

Начало программирование в блочной среде, отчасти напоминающей Scratch.

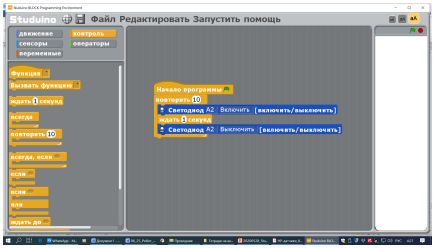
1. Вызов программы



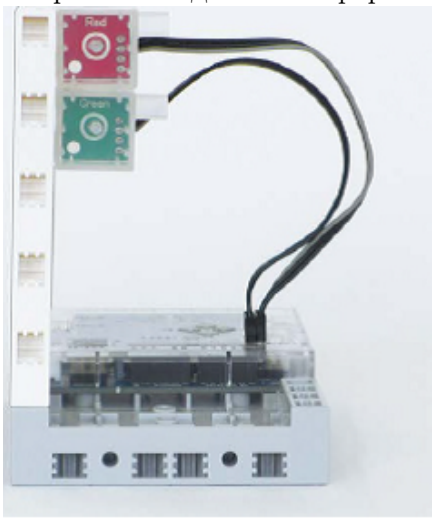
2. Нужно помнить, что плата сама по себе – «не знает», что к ней подсоединяются те, или иные устройства. Мы должны ей об этом «сказать». Это делается через Меню > Редактировать > Настройка портов.

Мы только что «сказали» плате, что к его порту (выходу) A1 подключен датчик освещенности. Точно также мы «сообщаем» ей о подключении моторов, сервомоторов, и других датчиков.

3. Обучение выполнению простейших команд проще всего осуществлять на коротких «живых» примерах.
Пример включения светодиода (выход A2) на 1 секунду. Далее он выключается. Все действия повторяются 10 раз.



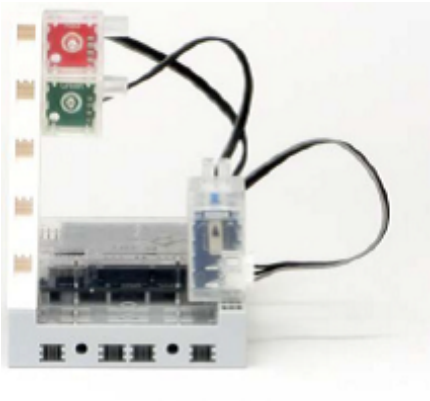
4. Сборка 1-й модели светофора.



5. Создание блок схемы для последующего программирования

Обратите внимание на использование цикла «Всегда». Светофор всегда выполняет этот цикл. Он работает непрерывно.

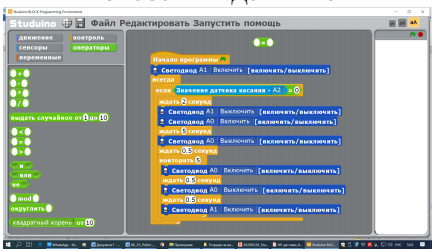
6. Усовершенствованная модель светофора №2. Светофор для пешеходов с кнопкой.



7. Блок-схема для светофора с кнопкой.

8. Готовая программа. На входе A0 – красный светодиод, на входе A1 – зеленый.

9. Отметим, что мы используем условный оператор, которые проверяет нажатие датчика касания. Когда датчик нажат, он посылает в плату сигнал 0. Справа и слева равенства могут быть выражения и показания датчиков.



10. Далее можно поставить задачу создания светофора с разрешающим звуковым сигналом для движения пешеходов (подключение зуммера). Заметим, что аналогичная тема разбирается в Занятиях 11-12.

1.10 Занятие 11-12.

Создание светомузыкальных устройств. Включение от датчиков света. Сравнение с другими устройствами – цифровой камерой, жидкокристаллическим телевизором, уличным освещением. Важность регулирования функционирования различных устройств от датчиков (от двери лифта до запирания дверцы стиральной машины).

Литература: [3], Часть «Программирование датчиков» стр. 51-88.

Цели занятия

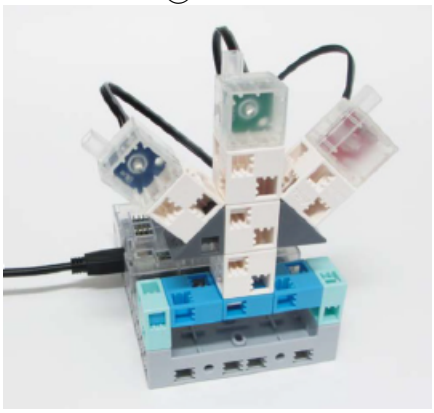
Изучение работы светомузыкального устройства. Создание блок-схемы для его работы. Роль программирования в управлении такими и аналогичными устройствами. Разработка блок-схемы. Написание программы.

Тренируемые навыки и приобретаемые знания

- Обсуждение значимости датчиков состояния окружающей среды и собственных датчиков различных устройств для их собственного функционирования
- Примеры: датчики закрывания дверей для лифтов, стиральных машин. Датчики загрязнения окружающей среды.
- Продолжение знакомства с условными операторами.
- Логические составные операторы.
- Некоторые встроенные функции.

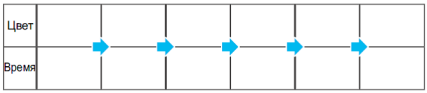
Краткое содержание урока

1. Прежде всего, создается конструкция из трех светодиодов и платы Studuino®.



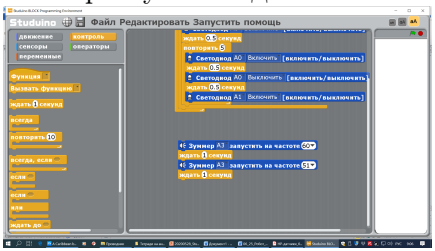
2. Блок-схема простейшего варианта свето-музыкальной установки – «Световое шоу» (звуковых функций ещё нет).

3. Усовершенствование работы светодиодов. Ученики сами заполняют таблицу зажигания светодиодов.



Затем эти пожелания записываются в виде блочной программы аналогично программе для светофора из Занятий 9-10.

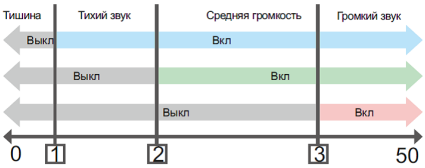
4. К светодиодам добавляется зуммер. Для зуммера можно написать простую мелодию.



5. В конструкцию можно добавить датчик света для её включения при падении уровня освещенности. Условие создается из шаблона условного оператора при значении порога освещенности 20 (при шкале от 0 до 100) и значении собственно светового датчика.

6. Итоговая программа собирается из частей, написанных в пунктах 3-5.

7. В качестве дополнительного задания можно предложить добавить датчик звука в конструкцию и написать программу включения светодиодов в зависимости от уровня звука. Например, при любых звуках загорается синий, при средних и громких – зеленый, при громких – красный. Пороговые значения 1, 2 и 3 на этой схеме предлагается задать самостоятельно.



1.11 Занятие 13.

Автоматическая дверь. Варианты реализации с разными датчиками. Изучение взаимодействия датчиковых систем и сервомоторов на примере конструкции автоматической двери.

Литература: [3], Часть «Программирование датчиков» стр. 3-55.

Цели занятия

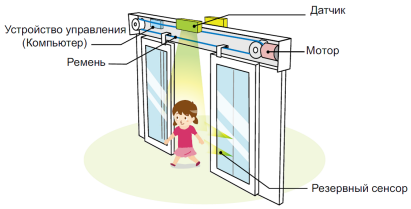
Обсуждение важности датчиков для работы систем, обсуживающих человека. Разработка блок-схемы. Написание программы.

Тренируемые навыки и приобретаемые знания

- Понимание различий между сервомоторами и обычными моторами.
- Калибровка сервомоторов.
- Принципиальная конструкция ИК-датчика.
- Альтернативные способы открывания двери: ИК-датчик, датчик света, «звонок» в дверь.
- Обеспечение безопасности при пользовании автоматическими дверями и другие похожие задачи (на примере ИК-датчиков движения).

Краткое содержание урока

1. Обсуждение принципа автоматической двери. Требования к функционированию, в т.ч. по безопасности (зачем нужен резервный сенсор).



2. Сервомоторы могут быть запрограммированы на поворот на любой угол от 0 до 180 градусов . Они делают это очень точно. В этом их отличие от обычных электрических моторов, которые могут делать много оборотов, но при этом не обеспечивают точный угол поворота.

3. Новые сервомоторы могут немного отличаться по своим настройкам друг от друга. Поэтому перед началом работы их нужно откалибровать. Это делается из Меню >Редактировать>Калибровка моторов.

4. Датчик ИК (инфракрасного излучения) на самом деле улавливает не наше с вами тепловое излучение. У него есть свой маленький излучатель ИК лучей, которые отражаются от нас и других предметов, и частично возвращаются в маленький приемник, который также является частью этого датчика:

ИК-датчик. Излучатель показан синей стрелкой, приемник – красной.

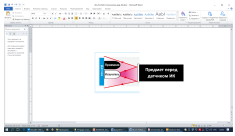
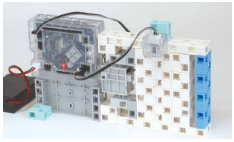
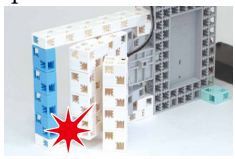


Схема работы ИК-датчика.

5. Сборка модели автоматической двери.

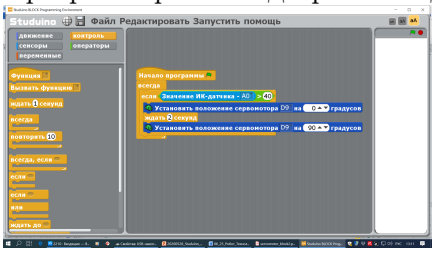


Дверь закрыта.



Дверь открывается. Синий столбик изображает человека перед дверью.

6. Программирование двери с ИК датчиком.



7. Альтернативные программы открывания дверей с помощью датчика света или кнопок на плате могут быть предложены в качестве дополнительных упражнений.

2 Практические работы с технологическими установками.

Следующие четыре сдвоенные занятия (4 занятия по 90 мин.) посвящены вопросам использования цифровых и аналоговых датчиков для решения практических задач.

Робототехника – область технологии, использующая симбиоз многих наук и практик, для создания устройств, которые призваны облегчать жизнь человеку, заменять его при работе в условиях вредных или опасных воздействий, а также выполнять тяжелые рутинные операции. В качестве наук, которые входят в этот симбиоз можно рассматривать такие как: механика и мехатроника, информатика и программирование, электроника и технология, а также многие другие.

Следующие четыре занятия позволяют получить общие понятия о таких устройствах, освоить принципы их построения и функционирования, а также научиться составлять программы, работая по которым робототехническое устройство будет выполнять задачу, поставленную пользователем.

Занятия рассчитаны на слушателей, которые уже имеют базовые знания о языке программирования Си, и могут написать простую программу с использованием переменных, блоков ветвления и циклов. В курсе используется набор образовательной робототехники NauRobo «Искусство программирования роботов» (базовый блок ArTec; DC моторы; шестерни; зубчатая рейка; датчики: касания, IR; набор базовых кубиков и блоков для построения конструкции), персональный компьютер или ноутбук с установленной средой Arduino IDE для программирования робототехнической конструкции.

После освоения курса, слушатели будут знать базовые особенности работы с цифровыми и аналоговыми датчиками; смогут выбирать необходимый тип датчика для управления движением робототехнической конструкции, управлять DC-двигателями на платформе Studuino®; получат знаниями для проектирования базовых алгоритмов контроля и перемещения робототехнических устройств в пространстве.

2.1 Занятие 1-2.

Управление DC-двигателями на платформе Studuino®. Специфика и особенности этих двигателей. Техническое задание для проектирования робототехнического устройства.

Цели занятия

Познакомиться с внутренним устройством платформы Studuino® в части управления DC-двигателями. Научиться составлять программу для управления работой двигателя (запуск, остановка, изменение скорости и направления вращения) в среде Arduino IDE. Определить причину возникновения разных результатов работы двигателя при одинаковой программе. Поставить задачу и сформулировать техническое задание на проектирование робототехнического устройства.

Краткое содержание занятия

Знакомство с платформой Studuino® и ее разъемами для подключения разных устройств. Назначения драйвера двигателя, режимы его работы и их задание. Программа для управления двигателями. Влияние нагрузки, напряжения элементов питания на работу двигателя как доказательство необходимости контролировать работу двигателя, чтобы получать одинаковые результаты (чтобы, например, платформа все время приезжала в одну точку), несмотря на то, что программа не изменяется. Формулирование задачи на создание робота, работающего на складе. **Ход занятия** Конструктор НАУРОБО «Искусство программирования роботов» базируется на платформе Studuino® от компании ArTec. В качестве строительной основы в конструкторе используется всего лишь несколько базовых деталей, из которых можно собрать достаточно сложные конструкции.

Рисунок 1. Базовые блоки конструктора ArTec

Рисунок 2. Пример моделей из деталей конструктора

Для начинки основного управляющего блока компания ArTec сделала свою версию платы Arduino, доработав ее, и назвав Studuino®. Компания добавила в нее драйвер управления DC-двигателями и вывела линейку под разъемы на 3 контакта (стандартный BLS-3, такие ставят на сервоприводы). На каждом таком разъеме присутствует питание: плюс и минус, а также один из пинов Arduino платы. Все разъемы подписаны, так что ошибиться при подключении к ним тяжело.

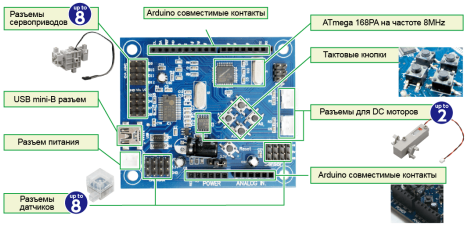


Рисунок 3. Платформа Studuino®

Для совместимости со стандартными шилдами Arduino на штатных местах присутствуют такие же разъемы, как и на Arduino.

Основным отличием от платы Arduino является то, что напряжение питания контроллера составляет 3,3В. Объясняется это тем, что питание центрального управляющего модуля осуществляется через поставляемый в комплекте батарейный отсек на 3 батарейки AA (в сумме дают 4,5В).

Плата находится в пластиковом корпусе, к которому можно крепить базовые детали конструктора и другие элементы.

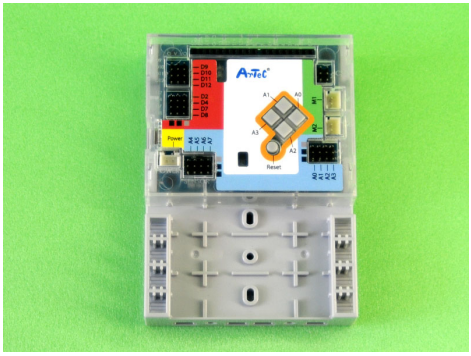


Рисунок 4. Центральный блок ArTec

В качестве исполнительных механизмов можно подключить два DC-мотора с редукторами и до 8-ми сервоприводов. Моторы подключаются через специальные разъемы, которые есть на плате.

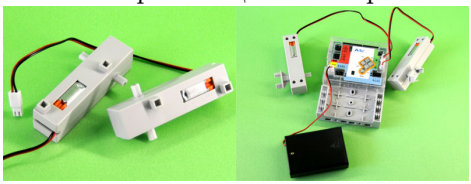


Рисунок 5. DC-моторы с редукторами и их подключение

Можно собрать достаточно легко и быстро самую простую конструкцию, которая приходит на ум: робота-машинку с двумя двигателями, управляя которыми можно заставить ее двигаться в нужном направлении.

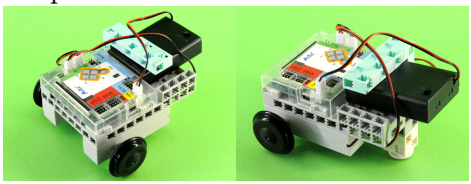


Рисунок 6. Простой робот-машинка с двумя DC-двигателями

Соберем такого робота-машинку для того, чтобы научиться управлять DC-двигателями, через написание программы на языке Си в среде Arduino IDE.

Двигатели являются исполнительными устройствами, для работы которых требуется большой выходной ток (больше, чем микроконтроллер способен пропустить через свои выводы). Для того чтобы

не повредить микроконтроллер, подключение и управление работой DC-двигателей производится с помощью специальной микросхемы – драйвера двигателей, которая уже встроена в платформу Studuino® и обеспечивает одновременное подключение и управление до двух двигателей, подключаемых в разъемы M1 и M2.

Управление двигателями осуществляется через задание логических состояний выводов D2, D4, D3 для двигателя, который подключен в разъем M1 и D7, D8, D5 для двигателя подключенного к разъему M2 на плате Studuino®. Именно к этим выводам разработчики платы Studuino® параллельно подключили микросхему-драйвер управления двигателями. Использовать эти выводы, подключив туда свою схему, и при этом одновременно использовать двигатели, не получится.

Рассмотрим как управлять двигателем, который подключен к разъему M1. Устанавливая разное логическое состояние выводов D2 и D4, можно добиться вращения двигателя в нужную сторону. Например: при D2 – 1 и D4 – 0 двигатель вращается по часовой стрелке, а если D2 – 0 и D4 – 1, то двигатель вращается против часовой стрелки. Также еще необходимо задать скорость вращения двигателя, установив значение ШИМ на выводе D3 – оно может быть в диапазоне от 0 до 255. При значении равным 0 двигатель не будет вращаться, а при значении 255 получим максимальную скорость вращения. Драйвер двигателя также допускает мгновенное изменение направления вращения двигателя и поддерживает режим торможения мотора. Для торможения двигателя следует задать на выводах D2 и D4 одинаковое логическое состояние равное 1.

Кстати, выводов D3 и D5 нет в разъемах, которые имеют по 3 контакта (GND, VCC, SIG) на плате Studuino®, но они есть на стандартном разъеме Arduino, который разработчики оставили для сохранения совместимости платформ.

Аналогичным образом осуществляется управление двигателем, который подключен к разъему M2. Для него направление вращения задается через состояние выводов D7 и D8, а скорость вращения выводом D5.

Напишем программу для управления одним двигателем в среде

Arduino IDE.

```
#define   M1_A    2         // вывод 1 для мотора M1
#define   M1_B    4         // вывод 2 для мотора M2
#define   M1_PWM  3         // вывод для контроля
//скорости вращения мотора M1

void setup() {
  pinMode(M1_A, OUTPUT); // установка выводов мотора
  //для задания им режима
  pinMode(M1_B, OUTPUT); // работы на «вывод», то есть
  //будем в них подавать значения
  analogWrite(M1_PWM, 100); // задание скорости вращения двигателя
  //(допустимы значения от 0 до 255)
  digitalWrite(M1_A, HIGH); // установим один из выводов мотора
  //в состояние 1,
  digitalWrite(M1_B, LOW); // а второй установим в 0
  //это заставит мотор вращаться в одну сторону
}

void loop() {
}
```

Программа 1. Бесконечная работа одного двигателя

При загрузке программы в робота-машинку произойдет следующее: робот будет оставаться на месте, вращая одним колесом и поворачиваясь вокруг оси второго колеса, которое будет неподвижно.

Изменим программу так, чтобы робот-машинка вращал колесом не бесконечно, а только определенный период времени и останавливался.

```
#define   M1_A    2      // вывод 1 для мотора M1
#define   M1_B    4      // вывод 2 для мотора M2
#define   M1_PWM  3      // вывод для контроля
//скорости вращения мотора M1
```

```
void setup() {
  pinMode(M1_A, OUTPUT);          // установка выводов мотора
  //для задания им режима
  pinMode(M1_B, OUTPUT);          // работы на «вывод», то есть
  //будем в них подавать значения
  analogWrite(M1_PWM, 100);       // задание скорости вращения двигателя
                                  // (допустимы значения от 0 до 255)
  digitalWrite(M1_A, HIGH);         // установим один из выводов
  //мотора в состояние 1,
  digitalWrite(M1_B, LOW);        // а второй установим в 0
  //это заставит мотор вращаться в одну сторону
  delay(10000);                   // установка задержки перед
  //выполнением следующей команды в 10000 миллисекунд
  digitalWrite(M1_A, LOW);        // установка вывода
  //мотора в состояние 0, чтобы он остановился
}

void loop() {
}
```

Программа 2. Работа одного двигателя в течение 10 секунд

Теперь требуется самостоятельно изменить программу таким образом, чтобы было задействовано два двигателя одновременно, и робот двигался вперед в течение 10 секунд и останавливался.

Пример такой программы приведен ниже. Если робот двигается не вперед, а назад, то нужно изменить или конструкцию робота-машинки, развернув двигатели на 180 градусов относительно текущей установки, что обеспечит вращение колес в другую сторону. Также можно воспользоваться вторым вариантом решения этой проблемы, изменив программу, чтобы моторы вращались в другую сторону. Для этого требуется изменить названия выводов в программе в тех командах, где им задается логическая единица (HIGH) или ноль (LOW):

$$\mathbf{M1\_A} \rightarrow \mathbf{M1\_B}$$

$$\mathbf{M2\_A} \rightarrow \mathbf{M2\_B}$$

```
#define   M1_A    2     // вывод 1 для мотора M1
#define   M1_B    4     // вывод 2 для мотора M2
```

```
#define   M1_PWM  3     // вывод для контроля скорости вращения
                        // мотора M1

#define   M2_A    7     // вывод 1 для мотора M2
#define   M2_B    8     // вывод 2 для мотора M2
#define   M2_PWM  5     // вывод для контроля скорости вращения
                        // мотора M2

void setup() {
  pinMode(M1_A, OUTPUT);     // установка выводов мотора 1
                             // для задания им режима
  pinMode(M1_B, OUTPUT);     // работы на «вывод», то есть будем
                             // в них подавать значения
  analogWrite(M1_PWM, 100); // задание скорости вращения двигателя 1
                             // (допустимы значения от 0 до 255)

  pinMode(M2_A, OUTPUT);     // установка выводов мотора 2
                             // для задания им режима
  pinMode(M2_B, OUTPUT);     // работы на «вывод», то есть
                             // будем в них подавать значения
  analogWrite(M2_PWM, 100); // задание скорости вращения двигателя 2
                             // (допустимы значения от 0 до 255)

  digitalWrite(M1_A, HIGH);         // установим один из выводов
                             // мотора 1 в состояние 1,
  digitalWrite(M1_B, LOW);       // а второй установим в 0
// это заставит мотор вращаться в одну сторону

  digitalWrite(M2_A, HIGH);         // установим один из выводов
                             // мотора 2 в состояние 1,
  digitalWrite(M2_B, LOW);       // а второй установим в 0
// это заставит мотор вращаться в одну сторону

  delay(10000);              // установка задержки перед выполнением
  // следующей команды в 10 сек
  digitalWrite(M1_A, LOW);  // установка вывода мотора 1 в состояние 0,
                             // чтобы он остановился

  digitalWrite(M2_A, LOW);  // установка вывода мотора 1 в состояние 0,
                             // чтобы он остановился
}
```

```
void loop() {
}
```

Программа 3. Движение робота-машинки вперед в течение 10 секунд

Предлагается провести небольшой эксперимент. Если в группе обучающихся есть несколько моделей собранных роботов-машинок, то загрузить в них одну и ту же программу и, запустив их с одной точки, отметить точку остановки. После этого положить сверху на робота-машинку какой-нибудь небольшой груз (это может быть сотовый телефон или какой-то небольшой предмет), снова запустить с исходной точки и отметить точку останова.

Можно увидеть после проведения такого эксперимента, что точка остановки будет отличаться. Более того, в ряде случаев можно увидеть, что одинаковые внешне роботы-машинки, собранные из одинаковых деталей, обладая одинаковой программой, проезжают разное расстояние за одинаковое время.

Объяснение этому явлению достаточно простое: драйвер двигателя, при управлении им программой просто коммутирует напряжение питания от элементов питания на двигатели. Скорость и мощность DC-двигателей напрямую зависит от уровня их заряда. На разных роботов заряд элементов питания может немного отличаться. Более того, одинаковые внешне DC-двигатели могут иметь немного разные характеристики работы и таким образом за одинаковое время работы программы (задержка между стартом и остановкой в 10 секунд), мы можем увидеть разный результат выполнения одной и той же программы. А опыт, проведенный с роботом, который двигается с нагрузкой, еще раз показывает, что для достижения требуемой точки при перемещении робота необходим контроль робота, который не может быть осуществлен таким простым методом как установка задержки между запуском и остановкой двигателей.

А теперь попробуем сформулировать небольшую практическую задачу. Представим, что нам нужно спроектировать робота, который должен перемещаться по складу вперед и назад и нам нужно заставить его останавливаться в заданных местах. Если не контролировать перемещение робота, то может произойти аварийная ситуация.

Перемещение робота вперед и назад можно сделать с помощью DC-двигателей или с помощью сервоприводов. Сервоприводы решают вопрос, как остановить робота в нужном месте (можно точно задавать угол поворота сервопривода). Однако у этого решения есть ограничение (будем использовать сервопривод из набора с закрепленным на нем колесом) – сервоприводы не могут поворачиваться на угол более чем 180 градусов и таким образом перемещение нашего робота будет ограничено половиной длины оборота колеса на сервоприводе, а обычно требуется перемещение на большее расстояние.

Таким образом, нам остается использовать DC-мотор с редуктором из набора для перемещения вперед и назад. У этих моторов нет обратной связи. После запуска двигателя мы не можем сказать, какое расстояние проехал робот. Можно засечь время, за которое при работе двигателя робот проходит требуемое расстояние и эти задержки использовать в программе для остановки робота в нужном месте. Наши исследования показали наличие у этого метода одного существенного недостатка – скорость вращения двигателя напрямую зависит от напряжения, которое на него подается и от требуемого усилия. Так как в роботе используются батареи, которые через некоторое время немного разрядятся и их напряжение станет меньше, то за то же самое время робот начнет проходить меньшее расстояние, и нужно будет опять подбирать временную задержку.

Таким образом, нужно каким-то способом контролировать передвижение робота и останавливать его не по временному интервалу, а другим методом. Решение этой проблемы есть – установить на робота «глаза» и «уши», которые позволят ему получать данные о том, как работают его исполнительные механизмы и о том, что происходит в пространстве вокруг него. Эти данные помогут роботу ориентироваться в окружающей его обстановке и принимать решения об остановке или продолжения движения для выполнения поставленной задачи.

В нашем случае «глазами» и «ушами» робота будут датчики, которые входят в набор НАУРОБО «Искусство программирования роботов». О том, как использовать эти датчики и получать с них данные, будет рассказано на следующем занятии.

2.2 Занятие 3-4.

Сборка модели робота для перемещения по складу. Цифровые датчики и получение данных с них. Датчик касания.

Цели занятия

Разработать конструкцию робота для перемещения по складу (перемещение по одному измерению: X, Y или Z). Научиться подключать цифровой датчик касания и получать данные с него. Разработать и реализовать алгоритм контроля передвижения робота с помощью датчика касания.

Краткое содержание занятия

Демонстрация зависимости движения робота-машинки от типа поверхности. Выбор зубчатой рейки и шестерни в качестве основного элемента передвижения робота по складу (зубчатая рейка – основание, шестерня – колесо, которое «катится» по рейке). Сборка робота для перемещения вдоль оси X. Ось X – имитация перемещения робота вдоль полок на складе. Принцип работы датчика касания. Функция получения данных с цифровых датчиков. Внесение изменения в конструкцию складского робота для использования в нем датчика касания. Алгоритм и программа, использующие датчик касания для ограничения перемещения складского робота.

Ход занятия

На прошлом занятии была рассмотрена модель робота-машинки, обладающей функцией «ехать вперед» по программе, которую мы же сами и разработали. Опыты с этой моделью показали, что дистанция перемещения может быть различная в зависимости от нагрузки на робота.

А если провести еще один эксперимент? Положим по ходу движения робота небольшой листок бумаги (из блока для заметок), кото-

рый может достаточно легко проскальзывать на поверхности, таким образом, чтобы робот при движении наехал на него одним колесом. Запустим робота и посмотрим, что произойдет. С очень большей вероятностью при наезде робота на лист одним колесом произойдет проскальзывание колеса вместе с листом бумаги, и робот изменит свою траекторию. А это значит, что у нас есть еще один фактор, который необходимо учитывать или избежать при разработке конструкции складского робота. Этот фактор называется коэффициентом сцепления робота с поверхностью. То есть необходимо контролировать не только вращение двигателя, но еще и оценивать поверхность, по которой робот передвигается, и нет ли проскальзывания колес.

Для исключения этого фактора многие разработчики робототехнических конструкций проектируют их таким образом, чтобы исключить возможность проскальзывания колеса по поверхности. Например, это можно сделать, если поменять колесо на шестерню, а поверхность, по которой эта шестерня будет перемещаться – на зубчатую рейку. Таким образом, зубцы шестерни, совпадая с зубцами рейки, исключат возможность проскальзывания при перемещении. Учитывая этот фактор, из деталей набора НАУРОБО «Искусство программирования роботов» можно собрать следующую модель складского робота.

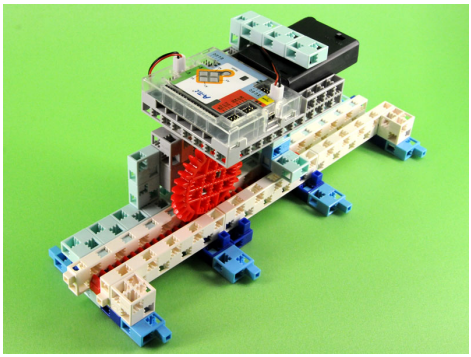


Рисунок 7. Модель складского робота для перемещения вдоль зубчатой рейки.

Для сборки робота используется один DC-двигатель с надетой на его вал шестерней, которая исполняет роль колеса и перемещается по направляющей зубчатой рейке. Это позволяет верхней подвижной части робота перемещаться без проскальзывания. Если внимательно

посмотреть на рисунок с моделью робота, то можно заметить, что направляющая расположена чуть ниже основной плоскости, по которой скользит подвижная часть и шестерня частично находиться внутри паза, где расположена зубчатая рейка. Это позволяет конструкции самостоятельно выравниваться во время перемещения робота влево и вправо и не съезжать с заданной траектории.

Учащимся предлагается собрать аналогичную конструкцию, используя детали из набора НАУРОБО «Искусство программирования роботов». На этой модели складского робота будет происходить демонстрация и отработка алгоритмов контроля перемещения робота.

После сборки этой конструкции предлагается вспомнить программу, которая запустит DC-двигатель робота для перемещения подвижной части робота в одном направлении (программа №1), загрузить ее в робота, запустить и понаблюдать, что с ним произойдет.

Как и следовало ожидать, программа и конструкция не предусматривает какой-либо реакции робота на то, что зубчатая рейка во время передвижения подвижной части может закончиться, и движение робота завершится аварийно. Пример такой аварии показан в этом видео:

Видео 1. https://youtu.be/a-q58XA4iKs

Попробуем сделать первую модификацию конструкции и алгорит-

ма перемещения робота так, чтобы было взаимодействие с окружающей обстановкой. Так как у нас модель робота для работы на складе, то логично предположить, что склад – это помещение со стенами, и робот, двигаясь вдоль стеллажа в какую-либо сторону, может упереться в стену. Для этого необходимо в конструкцию робота заложить датчик, который будет сигнализировать об этом.

В любом робототехническом наборе помимо исполнительных механизмов должны присутствовать датчики, которые являются «глазами» и «ушами» любого робота. В набор НАУРОБО «Искусство программирования роботов» входят следующие датчики: датчик освещения, датчик звука, датчик ускорения, два датчика ИК, датчик касания. Также в наборе есть светодиоды (зеленый, красный, синий, белый) и пьезодинамик.

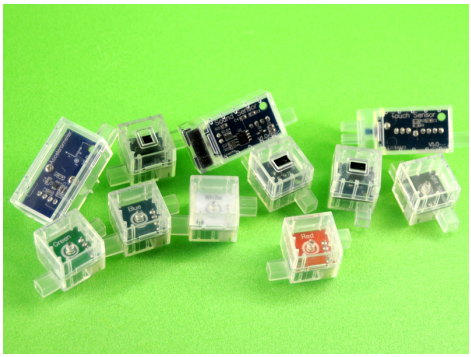


Рисунок 8. Датчики в наборе

Датчики подключаются к центральному блоку с помощью кабелей, которые идут в комплекте с набором. Для закрепления датчиков в модели робота на корпусах у них есть шипы, которыми они могут крепиться к конструкции.

Датчик касания представляет собой кубик с выступающим шипом, который может нажиматься при встрече с препятствием. Если поставить два таких датчика по краям нашего робота, то доезжая до стены слева или справа, складской робот обязательно коснется ими стены склада. Ожидание этого события можно запрограммировать в программе робота и обработать таким образом, чтобы робот останавливался или начинал двигаться в противоположную сторону.

Добавим в модель складского робота датчики касания, а также столбики слева и справа, имитирующие стены склада, в которые будут упираться датчики во время перемещения робота.

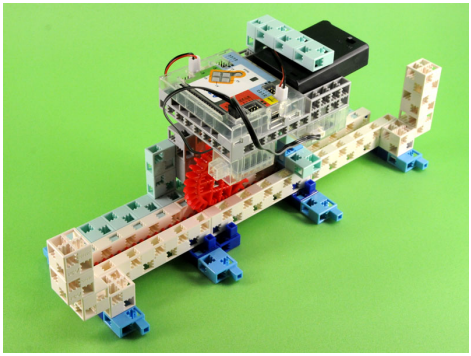


Рисунок 9. Модель складского робота с датчиками касания

Существуют датчики цифровые и аналоговые. Цифровые датчики имеют фиксированное число положений и переходят из одного положения в другое резко и без промежуточных значений. В основном используются датчики с двумя положениями. Датчики касания является наиболее ярким примером цифрового датчика: пока датчик не нажат он выдает логическое значение 1, а когда нажат – логическое значение равное 0 (в этом конструкторе и с этими датчиками используется именно такая схема, но бывают схемы, где это сделано наоборот: нажат – 1, не нажат – 0).

Для считывания показаний с цифровых датчиков подходят контакты микроконтроллера, которые имеют обозначение с префиксом «D» в начале: «D» - digital (цифровой). Главное помнить при их подключении, что при использовании DC-двигателей в конструкции, номера контактов, куда подключаются датчики, не пересекались с контактами управления DC-двигателями.

В этой конструкции датчики подключены к контактам с номерами D9 и D10. Для того чтобы считать показания с датчиков в программе управления роботом необходимо выполнить следующие операции: настроить эти контакты в режим чтения данных из них (режим ввода), а после этого специальной функцией можно прочитать состояние этого контакта (датчик нажат или нет) в тот момент, когда это необходимо.

Программа, которая запускает робота на перемещение и отслеживает нажатие датчиков касания, представлена ниже:

```
#define   M1_A    2     // вывод 1 для мотора M1
#define   M1_B    4     // вывод 2 для мотора M2
#define   M1_PWM  3     // вывод для контроля скорости вращения
                        // мотора M1

#define   SW1     9     // вывод для подключения датчика касания 1
#define   SW2     10    // вывод для подключения датчика касания 2

void setup() {
  pinMode(M1_A, OUTPUT);       // установка выводов мотора 1
                              // для задания им режима
  pinMode(M1_B, OUTPUT);       // работы на «вывод», то есть
                              //будем в них подавать значения

  analogWrite(M1_PWM, 100);    // задание скорости вращения двигателя 1
                               // (допустимы значения от 0 до 255)

  pinMode(SW1, INPUT_PULLUP); // установка выводов в режим
                             // чтения данных из них
  pinMode(SW2, INPUT_PULLUP); // для первого и второго датчиков касания
}

void loop() {
    digitalWrite(M1_A, HIGH); // запустить вращение
    digitalWrite(M1_B, LOW);  // двигателя в одну сторону

    while (digitalRead(SW1) != 0); // ждать пока не будет нажат
                                   // датчик касания 1

    digitalWrite(M1_A, LOW);  // запустить вращение
    digitalWrite(M1_B, HIGH); // двигателя в другую сторону

    while (digitalRead(SW2) != 0); // ждать пока не будет нажат
                                   //датчик касания 2
}
```

Программа 4. Движение складского робота с контролем датчиками касания

Следует помнить, что программа, разработанная в среде Arduino IDE для платформы Studuino®, всегда имеет две функции: «setup» и «loop». Если посмотреть на программы, которые были рассмотрены на предыдущем занятии, то в них блок «loop» был пустой (наличие этого блока даже пустого – обязательно, в противном случае возникнут ошибки при компиляции программы). Блок «setup» при запуске платформы Studuino® (при подаче питания на нее) выполняется один раз, а блок «loop» повторяется многократно в бесконечном цикле, пока на платформу подается питание.

В этой программе установка параметров контактов моторов и датчиков происходит в части «setup», так как это необходимо сделать только один раз. В ходе работы программы складского робота эти параметры остаются неизменными. В блоке «loop» размещены команды, которые запускают вращение двигателя в ту или иную сторону и ожидают срабатывания датчиков касания. Так как этот блок повторяется многократно, то модель робота будет перемещаться влево и вправо до касания стен склада датчиками касания бесконечно долго пока на платформу Studuino® подается питание. Работа этой программы показана в этом видео:

Видео 2. https://youtu.be/HA3jLus8px8

На видео видно, что робот доезжает до конца склада и касается датчиком стены, потом едет назад до противоположной стены склада

и там тоже касается стены другим датчиком касания. После этого процесс повторяется. Таким образом, здесь есть две точки, которые робот точно «знает» – это точки, когда происходит срабатывание датчика касания.

Иногда этого достаточно, чтобы решить поставленную задачу. Можно внести изменения в конструкцию робота, поставить дополнительные датчики касания, которые будут сигнализировать о достижении роботом требуемой точки в пространстве. Однако, использование датчиков касания не всегда оправдано, а иногда технически сложно и невыполнимо для отслеживания сложной траектории передвижения робота или точек, где робот должен остановиться. Для решения этой проблемы используются другие датчики, которые позволяют определить положение робота в пространстве бесконтактным методом. Реализация этого решения будет описана в следующем занятии.

2.3 Занятие 5-6.

Аналоговые датчики и получение данных с них. Модель складского робота с бесконтактным определением положения.

Цели занятия

Познакомиться с аналоговыми датчиками набора НАУРОБО «Искусство программирования» и научиться получать информацию с них. Изучить принцип работы ИК-датчика. Изменить модель складского робота и его программу для бесконтактного определения положения робота.

Краткое содержание занятия

Отличие аналоговых датчиков от цифровых. Понятие механизма аналого-цифрового преобразования. Функция для получения данных от аналоговых датчиков на языке программирования Си для платформы Studuino® в среде Arduino IDE. Калибровка аналоговых датчиков

и оценивание возвращаемых значений датчика по отношению к реальным данным. Принцип работы ИК-датчика. Внесение изменений в конструкцию складского робота и его программы для бесконтактного определения его положения.

Ход занятия

В предыдущем занятии рассматривалась модель складского робота с возможностью его контроля в двух точках склада, а именно в крайних положениях, около стен. Для этого использовались датчики касания, которые срабатывали, когда робот подъезжал близко к стене. Такое решение часто неудобно в практическом применении, потому что требует определенной конструкции не только от управляемого устройства, но и накладывает определенные требования на окружающую обстановку (наличие специальных точек, где будет происходить срабатывание датчиков касания). Эти факторы делают робототехническое устройство плохо приспособленным для смены окружающей обстановке (изменения позиций точек остановки). Именно поэтому очень часто применяются датчики, которые могут получать данные об окружающей среде робота бесконтактным способом, а на практике используется комбинация контактных и бесконтактных датчиков, чтобы улучшить уровень контроля робота и продублировать работу его систем.

Обратимся к набору датчиков, которые входят в состав набора НАУРОБО «Искусство программирование роботов» и рассмотрим те из них, показания значений которых можно будет использовать для оценивания обстановки вокруг робота и контроля его перемещения. Под эти условия наиболее подходят датчик освещения и ИК-датчики, входящие в набор робототехнического конструктора. Эти датчики являются аналоговыми и могут получать данные об окружающем их мире бесконтактным методом. В чем же состоит основное отличие аналогового датчика от цифрового?

Если цифровые датчики имеют фиксированное число положений и переходят из одного положения в другое резко и без промежуточных значений, то аналоговые датчики описывают некоторую неразрывную связь сигнала и напряжения или тока на выходе и не имеют фиксиро-

ванных значений. Это определение говорит нам о том, что на выходе у аналогового датчика получается некоторое вещественное значение. Однако данные с датчика будут обрабатываться на цифровой платформе, поэтому прежде чем с ними можно начать работать, их необходимо «оцифровать» или, говоря другим языком, провести процедуру аналого-цифрового преобразования. Для этого в микропроцессорном контроллере платформы Studuino® есть специальный блок АЦП, который осуществляет такой механизм преобразования «прозрачно» для пользователя. Пользователю остается лишь правильно подключить такие датчики в специальные контакты платформы, которые имеют префикс «A» - analog (аналоговый). Эти контакты могут работать также и в цифровом режиме, но если пользователь даст специальную команду чтения данных с этих контактов, то перед тем как данные попадут к пользователю, они пройдут процедуру преобразования при помощи блока АЦП.

Платформа Studuino® построена на базе микроконтроллера Atmega168. Разрядность его блока АЦП составляет 10 бит, что дает диапазон преобразования аналогового сигнала от 0 до 1023. Возникает законный вопрос: будет ли число, полученное после преобразования показания датчика освещенности равное 500 соответствовать освещению в 500 люкс? Ответом на этот вопрос будет: нет, не будет. Для каждого датчика есть свои характеристики и диапазон возвращаемых значений, которые описаны в документации на датчик от производителя. Для того чтобы привести полученное число к общепринятым в Международной системе измерения (СИ), необходимо выполнить его приведение по таблице, которая дается в сопроводительной документации на датчик. В ряде случаев достаточно формулы линейного преобразования для приведения относительных показаний датчика к реальным значениям, с которыми мы привыкли иметь дело.

В нашем случае мы работаем с образовательным набором и для решения нашей задачи нам не нужны абсолютные значения показаний аналоговых датчиков. Мы можем оперировать относительными значениями без приведения их к системе СИ. Для этого нам необходимо выполнить процедуру калибровки датчика. То есть выбрать какое-то показание датчика, относительно которого мы будем оценивать его текущее показание. Например, датчик освещенности может

выдавать значение в диапазоне 700-800 при обычном дневном свете и значения 20-50 при отсутствии освещения. Если требуется, чтобы робототехническая конструкция срабатывала при отсутствии освещения, можно выбрать в качестве опорного значение 700 и сравнивать текущие показания датчика с ним. В качестве примера можно посмотреть на простую модель установки, реагирующую на отсутствие дневного освещения, подавая питание на светодиод (модель автоматического управления освещением).

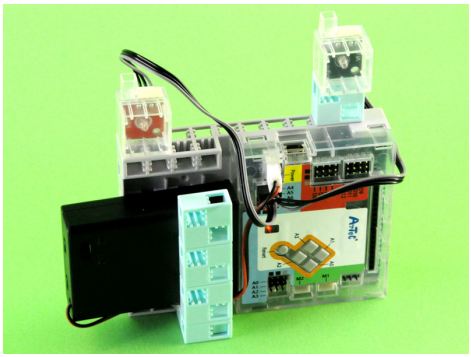


Рисунок 10. Модель автоматического управления освещением

Слева на модели расположен светодиод, который включается, когда уровень освещения становится меньше заданного порога. Текущий уровень освещенности измеряется датчиком света, который расположен справа на модели.

На видео показана работа модели:

Видео 3. https://youtu.be/IporhRPSjf4

Программа, которая загружена в модель, представлена ниже.

```
#define LUX_SENS  A4  // контакт, к которому подключен датчик света
#define LED_PIN   A5  // контакт, к которому подключен светодиод

#define MIN_LUX   700 // порог срабатывания устройства

void setup() {
  pinMode(LED_PIN, OUTPUT); // выставить режим "вывода"
  // для контакта светодиода
}

void loop() {
  if (analogRead(LUX_SENS) < MIN_LUX) { // если аналоговое значения
  // с датчика света меньше порогового значения
    digitalWrite(LED_PIN, HIGH);        // включить светодиод
  } else {                              // иначе
    digitalWrite(LED_PIN, LOW);         // выключить светодиод
  }
}
```

Программа 5. Контроль уровня освещения

Вернемся к нашей задаче о роботе на складе. Датчик освещения не очень подходит для этого устройства, так как реагирует на доста-

точно широкий диапазон световой волны, а также этот датчик достаточно инертный при измерении быстроизменяющихся показаний. Инертность этого датчика – это главная причина отказа от его использования в нашей конструкции. Попробуем использовать ИК-датчик. ИК-датчик работает по принципу улавливания отраженного сигнала. На рисунке можно четко увидеть два элемента: ИК-светодиод и ИК-фотоприемник.

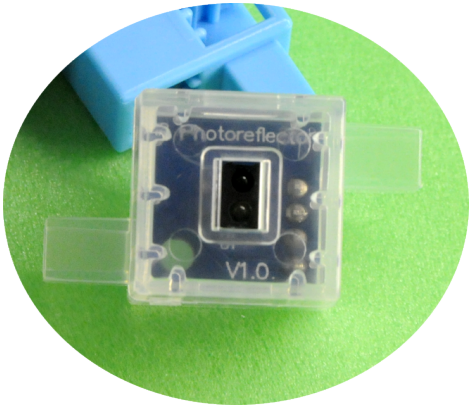


Рисунок 11. ИК-датчик

Если перед датчиком поставить какое-либо препятствие, от которого будут отражаться ИК лучи, то чем лучше они будут отражаться от препятствия (в зависимости от расстояния до препятствия или типа препятствия), тем больше их уловит ИК-фотоприемник и датчик вернет большее значение пользователю при его опросе.

После изменений в конструкции и программе робота, можно увидеть результат на видео:

Видео 4. https://youtu.be/VFrxd$_L$Qzo0

Рассмотрим более подробно, как изменились возможности робота. Очевидно, что двух точек остановки недостаточно и требуется больше. Для этого используется датчик ИК. Закрепив его на движущейся платформе, под ним снизу расположили полоску бумаги с предварительно нарисованными маркером черными линиями, на которых робот должен будет останавливаться. У робота также убрали один из датчиков касания (с правой стороны) и добавили светодиод красного цвета для подачи различных сигналов (на рисунке светодиод расположен слева, ИК-датчик – справа).

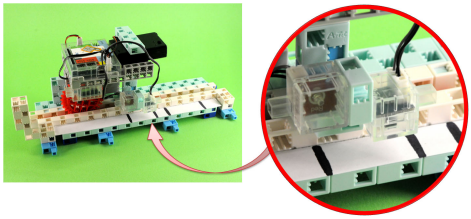


Рисунок 12. Робот с ИК-датчиком

В результате получилась классическая схема устройства одной из осей 3D-принтера или станка с ЧПУ. При включении принтер не знает, где он находится и едет до упора в одну сторону (до касания концевым выключателем), а после этого считает эту точку нулем и начинает от нее отсчитывать свое положение по этой оси.

В этой конструкции отсчет происходит по черным полосам на бумаге. Количество этих полос заранее известно, поэтому по достижении последней полосы можно вернуться в нулевую точку отсчета. Программа на языке Си для платформы Studuino® представлена ниже.

```
#define   M1_A    2     // вывод 1 для мотора M1
#define   M1_B    4     // вывод 2 для мотора M2
#define   M1_PWM  3     // вывод для контроля скорости вращения
                        // мотора M1

#define   SW1     A4    // контакт для подключения датчика касания
#define   IR      A7    // контакт для подключения ИК-датчика
#define   LED     A5    // контакт для подключения светодиода

#define   STOP    4     // количество точек остановки

#define   MIN_IR  400   // минимальное значение:
                        // началась черная линия
#define   MAX_IR  500   // максимальное значение:
                        // закончилась черная линия

void setup() {
  pinMode(M1_A, OUTPUT);       // установка выводов мотора 1
                               // для задания им режима
  pinMode(M1_B, OUTPUT);       // работы на "вывод", то есть
                               // будем в них подавать значения
  analogWrite(M1_PWM, 100);    // задание скорости вращения двигателя 1
                               // (допустимы значения от 0 до 255)
  pinMode(SW1, INPUT_PULLUP); // установка вывода в режим чтения
                               // данных из него
  pinMode(LED, OUTPUT);        // установка вывода в режим
                               // "вывод" данных
}

/* функция, которая моргает светодиодом n раз */
void blink(int n) {
  for (int i = 0; i < n; i++) { // цикл со счетчиком,
                               // повторения задаются параметром "n"
    digitalWrite(LED, HIGH);   // включить светодиод
    delay(200);                // подождать 200 мс
    digitalWrite(LED, LOW);    // выключить светодиод
```

```
    delay(200);                    // подождать 200 мс
  }                                // повторить цикл, если выполняется
                                   // условие "i < n"
}

void loop() {
  digitalWrite(M1_A, HIGH);            // включить двигатель,
                                      // чтобы робот поехал влево
  digitalWrite(M1_B, LOW);             //
  while (digitalRead(SW1) != 0);       // ждать пока не коснется
                                   // датчиком касания стены
  digitalWrite(M1_A, LOW);             // включить двигатель,
                                   // чтобы робот поехал вправо
  digitalWrite(M1_B, HIGH);            //
  for (int i = 0; i < STOP; i++) {     // заранее известно количество
                                   // линий остановок, поэтому цикл,
                                   // пока не проедем все линии
    while (analogRead(IR) > MIN_IR); // ждем пока не начнется
                                   // черная линия
    digitalWrite(M1_B, LOW);           // останавливаем робота
    blink(i + 1);                      // моргнем светодиодом то
                                   // количество раз, на какой линии
                                   // сейчас остановились
    digitalWrite(M1_B, HIGH);          // включаем двигатель,
                                   // чтобы робот поехал вправо
    while(analogRead(IR) < MAX_IR);  // ждем пока не кончится
                                   // черная линия
  }                                    // если не проехали все
                                   // черные линии "i < STOP"
                                   // то продолжаем выполнять цикл for
}
```

Программа 6. Робот, использующий ИК-датчик для контроля перемещения по складу

Цифры 400 и 500 подобраны опытным путем. Находясь над белой поверхностью, датчик показывал значения примерно 650-750. Над черной поверхностью датчик возвращал значения в диапазоне 180-250. Цифра 400 – это момент, когда датчик начинает переходить с белой поверхности на черную, а цифра 500 – момент перехода с черной поверхности на белую. Эти цифры взяты с небольшим запасом, чтобы

перекрыть погрешность измерения датчика. При разработке реальной конструкции, необходимо учитывать условия, в которых будут сниматься показания с датчика (внешнее освещение, расположение датчика и т.д.), потому что все эти факторы будут оказывать влияние на показания датчика. Возможно придется разработать алгоритм постоянной корректировки данных значений в зависимости от внешних условий. ИК-датчик выдает аналоговый сигнал освещенности, который может зависеть от разных факторов, таких как внешняя засветка, напряжение батареи питания и даже температуры окружающей среды.

Подведем небольшой итог. Было рассмотрено два возможных варианта решения задачи для перемещения робота по складу. Оба этих варианта требовали дополнительных внешних «меток», по которым ориентировался робот. А можно ли как-нибудь обойтись без них? Чтобы робот, например, знал, на какой угол повернулся вал двигателя и, в зависимости от значения угла, принял решение об остановке или движении далее. Такое решение существует, и оно будет рассмотрено на занятии 7-8.

2.4 Занятие 7-8.

Определения угла поворота выходного вала редуктора как метод контроля перемещения робототехнической конструкции. Модель энкодера и алгоритм работы с ним.

Цели занятия

Получить основные сведения и понятия о датчиках угла поворота и принципах обработки данных с них. Разработка конструкции датчика угла поворота, используя стандартные детали конструктора НАУРОБО «Искусство программирования роботов». Разработка алгоритма контроля вращения вала двигателя с помощью энкодера.

Краткое содержание занятия

Назначение и устройство датчиков угла поворота. Обработка информации с энкодеров. Разработка модели энкодера на базе ИК-датчика из элементов набора НАУРОБО «Искусство программирования роботов». Разработка алгоритма и программы контроля вращения вала DC-двигателя, используя энкодер.

Ход занятия

На предыдущих занятиях рассматривались возможности робототехнического набора НАУРОБО «Искусство программирования роботов», которые позволяли управлять DC-двигателями, входящими в комплект. Также были рассмотрены принципы получения данных и обработки информации с аналоговых и цифровых датчиков. Была построена модель складского робота, перемещение которой можно было контролировать в определенных точках, задаваемых пользователем. Однако все рассмотренные методы и алгоритмы в предыдущих занятиях для контроля перемещения устройства требовали дополнительных элементов в окружающей робота обстановке. Для датчиков касания требовалось установить точки касания. Для ИК-датчика нанести специальную разметку, которая будет использоваться для определения положения робота в пространстве. Такие дополнительные элементы всегда ограничивают гибкость использования разработанной конструкции, так как в случае перемещения исполнительного механизма в другую среду (например, перемещение складского робота для работы в другое помещение), необходимо также заново разместить дополнительные элементы для его нормального функционирования. В ряде случаев это дополнительные временные затраты, а иногда это сделать невозможно в связи с особенностями нового места для размещения робототехнической конструкции.

Для решения этой проблемы разработчики стараются при проектировании решения не использовать такие датчики, требующих установки внешних «маяков», по которым происходит их правильное функционирование и в конечном итоге, определения положения в пространстве, получение каких-то иных данных и т.д. Например, для склад-

ского робота можно использовать специальный датчик угла поворота, установленный на вал выходного редуктора, который обеспечивает перемещение робота вдоль зубчатой направляющей. Так как проскальзывание отсутствует, то можно получить достаточно точные данные о том, в каком месте сейчас находится робот, зная диаметр шестерни и угол поворота выходного вала редуктора.

Итак, что же такое датчик угла поворота или как его еще часто называют – энкодер?

Датчик угла поворота (энкодер) – устройство, предназначенное для преобразования угла поворота вращающегося объекта (вала) в цифровые или аналоговые сигналы, позволяющие определить угол его поворота.

Энкодеры подразделяются: по способу выдачи информации на накапливающиеся (инкрементные) и абсолютные (позиционные); по принципу действия на оптические, резистивные, магнитные, индуктивные, механические; по допустимому углу поворота вала с ограниченным диапазоном работы и с неограниченным диапазоном работы.

Накапливающие энкодеры на выходе формируют импульсы, по которым принимающее устройство определяет текущее положение вала путем подсчета импульсов счетчиком. Сразу же после включения энкодера положения вала неизвестно и нужно привязать начало системы отсчета к какой-то заранее известной точке (на занятии 3 робот на складе после его включения не знал, где он находится и считал стартовой точкой момент срабатывания датчика касания, расположенного слева).

Абсолютные энкодеры выдают на выходе сигналы, которые можно однозначно интерпретировать как угол поворота вала датчика угла. Датчики угла этого типа не требуют привязки системы отсчета к какому-либо нулевому положению.

В наборе НАУРОБО «Искусство программирования роботов» есть DC-двигатели с редуктором, но они не имеют энкодера. А что если попробовать сделать простой оптический накапливающий энкодер из деталей конструктора, тем более что шестерни из набора имеют достаточно крупный модуль (размер зуба)?

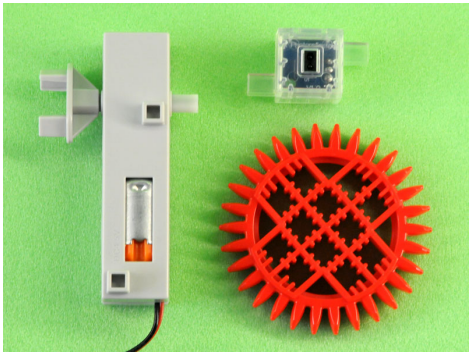


Рисунок 13. Основные элементы энкодера

Основной проблемой для такой конструкции является подбор расположения и закрепления ИК-датчик таким образом, чтобы его световой поток пересекался зубом шестерни при ее вращении. На рисунке ниже представлена конструкция, где это условие выполняется.

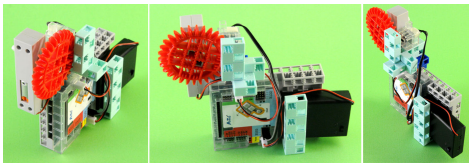


Рисунок 14. Тестовая модель энкодера

Как видно на изображении выше, ИК-датчик закреплен так, чтобы шестерня при вращении пересекала его рабочую область измерения своими зубьями. Напротив (с другой стороны шестерни) ИК-датчика дополнительно установлено препятствие, работающее на отражение ИК-лучей, чтобы получить более корректные данные с датчика. Когда шестерня будет вращаться и ИК-датчик измерять отражение сигнала, то будут большие значения, когда перед датчиком будет зуб и меньшие, когда – «дырка» между зубами шестерни.

Для проверки работоспособности этой идеи разработаем простую программу, которая запустит двигатель на вращение с постоянной скоростью и будет непрерывно выводить значения с ИК-датчика на отладочную консоль.

```
#define M1_A        2       // Управляющая ножка 1 для двигателя 1
#define M1_B        4       // Управляющая ножка 2 для двигателя 1
```

```
#define M1_PWM      3        // Управляющая ножка для задания
                             // скорости двигателя 1

#define SENSOR_PIN  A4       // Ножка на которую подключен IR-sensor

void setup() {
  Serial.begin(9600);        // Инициализация отладочной консоли
                             // для вывода туда данных

  pinMode(M1_A, OUTPUT);     // Установка управляющих пинов контроллера
  pinMode(M1_B, OUTPUT);     // в режим "вывода данных"
  analogWrite(M1_PWM, 100); // Задание скорости вращения двигателя

  digitalWrite(M1_A, HIGH); // Запуск двигателя на вращение
  digitalWrite(M1_B, LOW);

  for (int i=0; i < 2000; i++) {     // В цикле 2000 раз опросить
                                     // значение ИК-датчика
    Serial.println(analogRead(SENSOR_PIN));    // Опросить и вывести
                                               // значения в консоль
  }
  digitalWrite(M1_A, LOW);  // Остановить вращение двигателя
}

void loop() {
}
```

Программа 7. Вывод данных с ИК-датчика в отладочную консоль

По данным, которая программа вывела в консоль, можно построить следующий график:

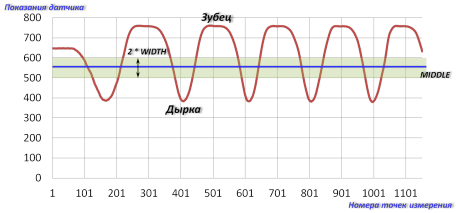


Рисунок 15. График изменения значений ИК-датчика при вращении шестерни

Характер графика напоминает форму зубцов у шестерни, что говорит о том, что действительно такие данные можно использовать для контроля вращения двигателя, используя зубчатую шестерню из набора конструктора как диск энкодера. Для исключения «дребезга» используется гистерезис, который реализован следующим образом (обозначения на графике): MIDDLE – это среднее значение между максимальной и минимальной величиной показаний ИК-датчика, WIDTH – это отклонение от MIDDLE к большему или меньшему значению для создания определенной «полосы погрешности» измерений сигнала (общая ширина этой полосы равняется 2*WIDTH). MIDDLE и WIDTH используются в алгоритме работы контроля вращения двигателя. Алгоритм подсчета зубцов в шестерне во время вращения двигателя можно представить следующим образом:

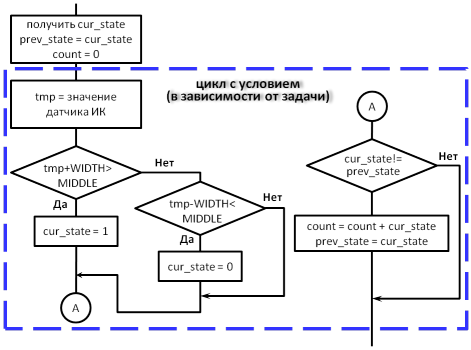


Рисунок 16. Алгоритм подсчета зубцов в шестерне во время вращения двигателя

В алгоритме используются следующие обозначения:
prev_state – предыдущее состояние шестерни;
cur_state – текущее состояние шестерни;
count – количество подсчитанных зубцов шестерни;
tmp – показания ИК-датчика.

Принцип подсчета зубцов шестерни в этом алгоритме основан на постоянном считывания показания с ИК-датчика и изменения значения переменной cur_state при переходе уровня сигнала за верхнюю или нижнюю линию «полосы погрешности». Если значение переходит верхнюю границу, то переменная cur_state становится равна 1, что означает зубец шестерни, а при переходе за нижнюю границу, пере-

менная cur_state становится равна 0, что означает провал между зубьями шестерни. Прибавление в переменной count происходит только при изменении состояния переменной cur_state.

Программа, реализующая данный алгоритм, представлена ниже. В ней описана подпрограмма ожидающая, пока двигатель не повернет шестерню на заданное количество зубов и после этого возвращает управление в основную программу.

```
#define M1_A         2       // Управляющая ножка 1 для двигателя 1
#define M1_B         4       // Управляющая ножка 2 для двигателя 1
#define M1_PWM       3       // Управляющая ножка
                             // для задания скорости двигателя 1
#define SENSOR_PIN   A4      // Ножка на которую подключен IR-sensor

#define MIDDLE       550     // Серединное значение диапазона,
                     // по которому определяется зубец или "дырка"
#define WIDTH        50      // Значение ширины погрешности
                             // от середины диапазона, чтобы исключить
                             // ложный подсчет значений из-за
                             // погрешности считывания
                             // с аналогового датчика

int enc_tooth = 0;           // Переменная, содержащая количество
                             // зубцов шестеренки
int cur_state = 0;           // Текущее состояние шестеренки:
                             // зубец (1) или "дырка" (0)
int prev_state = 0;          // Предыдущее состояние шестеренки
int tmp;                     // Переменная для хранения временных
                             // значений вычислений

void init_enc() {
  enc_tooth = 0;
  cur_state = 0;
  prev_state = 0;

  // Прочитать текущее состояние шестеренки и инициализировать
  // значение переменной cur_state
  // то есть если зубец, то переменная = 1, а иначе там "дырка"
  // и переменная не меняет свое значение из 0 как она и была присвоена из
  if (analogRead(SENSOR_PIN)>MIDDLE) {
```

```
    prev_state = 1;
    cur_state = 1;
    enc_tooth++;
  }
}

void wait_by_count(int count) {
  // Выполнить цикл пока не будет достигнуто count, в котором производить
  // измерения с датчика IR и в зависимости от его значения увеличивать
  // количество зубцов или "дырок"
  while (enc_tooth <= count) {
    // Считываем показания с IR-датчика во временную переменную
    tmp = analogRead(SENSOR_PIN);

    // Если полученное значение лежит выше середины + ширина погрешности,
    // то значит там зубец
    if ((tmp + WIDTH) > MIDDLE) {
      cur_state = 1;
    } else {
    // Если полученное значение лежит ниже середины - ширина погрешности,
    // то значит там "дырка"
      if ((tmp - WIDTH) < MIDDLE) {
        cur_state = 0;
      }
    }

    // Если текущее значение не совпадает с предыдущим, то значит шестере
    // повернулась уже настолько, что вместо зуба появилась "дырка" или
    // наоборот - вместо "дырки" появился зуб
    if (cur_state != prev_state) {
    // Увеличиваем количество зубов, если прибавить 0 к какому-то значени
    // то ее значение не поменяется
      enc_tooth += cur_state;

    // Изменяем предыдущее состояние для сравнения при новом измерении
      prev_state = cur_state;
    }
  }
}

void setup() {
  // Инициализировать монитор порта для последующего вывода данных туда
```

```
  Serial.begin(115200);

  // установить ножки для управления двигателем в состояние "выход"
  pinMode(M1_A, OUTPUT);
  pinMode(M1_B, OUTPUT);

  // обнулить значения переменных
  init_enc();

  // установить скорость вращения двигателя
  analogWrite(M1_PWM, 100);

  // запустить двигатель на вращение
  digitalWrite(M1_A, HIGH);
  digitalWrite(M1_B, LOW);

  // ждать, пока не пройдут 30 зубцов шестерни
  wait_by_count(30);

  // Остановить вращение двигателя торможением
  digitalWrite(M1_B, HIGH);

  Serial.print("Количество зубцов = ");
  Serial.println(enc_tooth);

}

void loop() {
}
```

Программа 7. Вывод данных с ИК-датчика в отладочную консоль

Программа запускает двигатель и ждет, пока шестерня не повернется на 30 зубцов – это общее количество зубцов в шестерне, и после этого останавливает мотор. Ниже представлено видео, которое демонстрирует работу программы.

По белому кусочку бумаги, который приклеен к шестерне, можно отследить ее оборот на 360 градусов.

Видео 5. https://youtu.be/wS0eODG7zzc

На этом можно завершить данное занятие. В качестве самостоятельной работы можно попробовать реализовать энкодер для робота на складе и отслеживать перемещение с помощью него.

Подводя итоги цикла из четырех занятий, можно сказать, что объема этого материала пока недостаточно для проектирования полноценных робототехнических устройств, используемых в реальной жизни. Однако этот материал может стать отправной точкой, которая содержит базовые знания о принципах функционирования таких устройств, о том, как такие устройства «общаются» с окружающим миром и понимают его. Алгоритмы, предложенные в этом пособии можно развивать и модернизировать в зависимости от типа решаемых задач.

Следующие 4 занятия посвящены созданию рабочего стенда для изучения управления квадрокоптером. Стенд состоит из летющего квадрокоптере и карданного подвеса.

2.5 Занятие 9.

Сборка карданного подвеса для квадрокоптера. Практическое знакомство с понятием степени свободы механической системы, управление числом степеней свободы путем выборочной блокировки осей вращения. Изучение показаний телеметрии летательных аппаратов и графи-

ческое представление данных, передаваемых беспроводным образом от квадрокоптера на компьютер. Описание положения твёрдого тела в трёхмерной системе координат.

Цели занятия

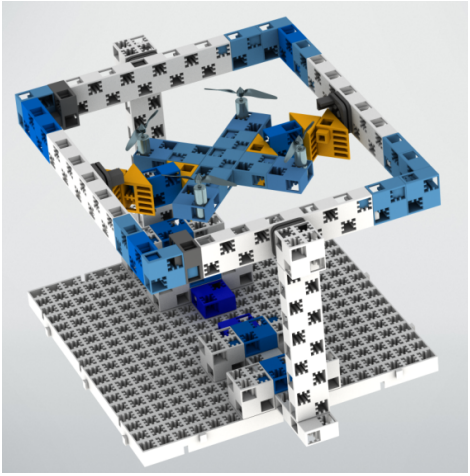
Сборка из конструктора и применение карданного подвеса для позиционирования квадрокоптера в пространстве. Практическое знакомство с понятием физической величины, зависящей от времени. Угол и его производная по времени. Изучение графиков, строящихся в реальном времени, на примере графиков углов наклона борта и их производных. Наглядная демонстрация физического смысла угловой скорости (производной угла по времени) с помощью системы беспроводной телеметрии. Демонстрация достаточности трёх независимых вещественных величин для описания положения твёрдого тела в трёхмерном пространстве.

Тренируемые навыки и приобретаемые знания

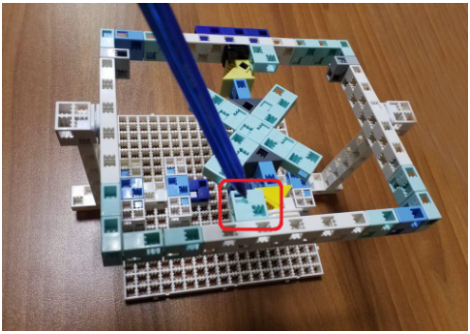
- Обсуждение измерительных систем квадрокоптера.
- Обсуждение исполнительных систем квадрокоптера.
- Формирования навыка математического описания физических свойств системы.
- Навык визуального анализа графической информации.

2.6 Краткое содержание урока

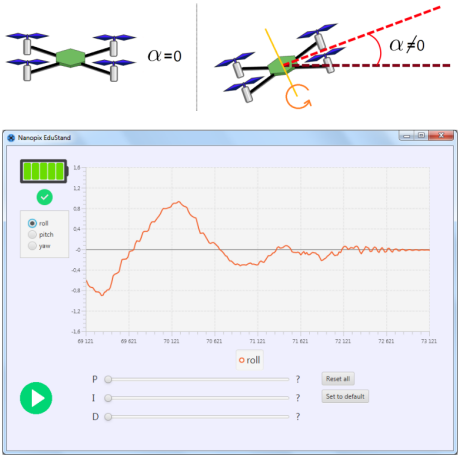
1. Прежде всего, создается конструкция карданного подвеса.



2. Выборочная блокировка оси вращения.



3. Беспроводная передача телеметрии с борта на ПК:



2.7 Занятие 10.

Настройка системы автоматической стабилизации квадрокоптера. Практическое изучение понятия зависимости угла наклона борта относительно горизонтали от времени и его производная по времени с помощью цифровой измерительной системы на базе гироскопа и акселерометра. Изучение показаний телеметрии летательных аппаратов и графическое представление данных, передаваемых беспроводным образом от квадрокоптера на компьютер. Практическое знакомство с понятием системы управления с обратной связью. Настройка пропорционального коэффициента. Настройка дифференциального коэффициента.

Цели занятия

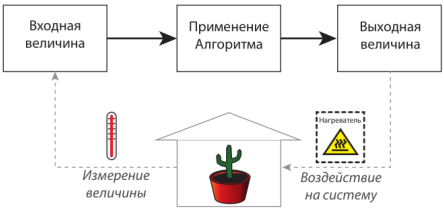
Изучение систем автоматического управления на примере квадрокоптера. Знакомство с входными (измеряемыми) и выходными (управляемыми) физическими параметрами. Подбор коэффициентов системы автоматического управления. Практическое знакомство с понятием производной угла по времени. Знакомство с характерными ошибками настройки систем управления: перерегулирование и недорегулирование.

Тренируемые навыки и приобретаемые знания

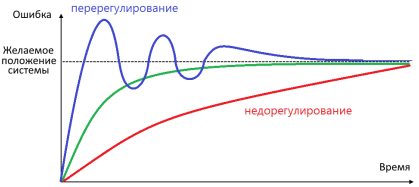
- Обсуждение измерительных систем квадрокоптера.
- Обсуждение исполнительных систем квадрокоптера.
- Знакомство с системой управления с обратной связью.
- Качественное описание характера поведения системы управления в зависимости от недостаточности или избыточности управляющих коэффициентов.
- Навык подбора коэффициентов PID-регулятора.

Краткое содержание урока

1. Блок-схема системы управления с обратной связью



2. 2. Анализ поведения механической системы:



3. 3. Подбор коэффициентов системы управления исходя из наблюдаемой реакции

2.8 Занятие 11.

Изучение тяги винтомоторной группы квадрокоптера. Изложение теоретических аспектов возникновения подъемной силы крыла. Представление пропеллера как пары крыльев. Описание экран-эффекта. Конструирование экспериментальной установки из конструктора. Проведение практического эксперимента с помощью цифровой лаборатории. Изучение зависимости подъемной силы от потребляемого мотором тока.

Цели занятия

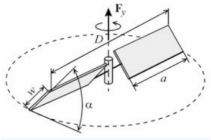
Изучение силовой установки квадрокоптера. Знакомство с физическими принципами, лежащими в основе винтомоторной установки. Практическое измерение подъемной силы одного мотора с пропеллером. Конструирование управляемой лабораторной системы питания электромотора с помощью модульной электрической схемы. Демонстрация ослабления тяги при возникновении экран-эффекта.

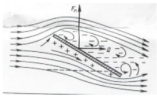
Тренируемые навыки и приобретаемые знания

- Обсуждение устройства пропеллера и электромотора.
- Знакомство с простейшими эффектами аэродинамики.
- Конструирование управляющей электрической цепи.
- Общий навык проведения эксперимента с обработкой и визуализацией результата.

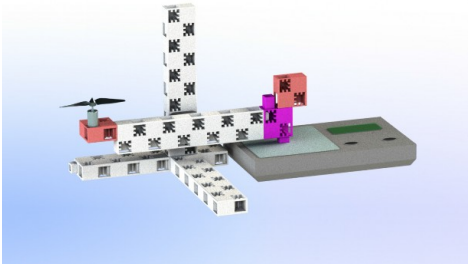
Краткое содержание урока

1. Иллюстрации теоретических аспектов возникновения подъемной силы и представления пропеллера в виде простейшей модели.

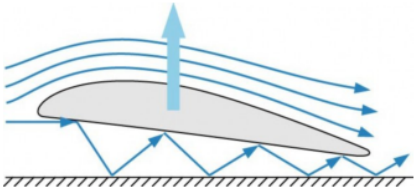




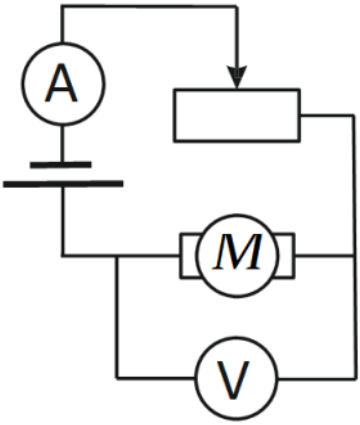
2. Схема экспериментальной конструкции с электронными весами и электромотором.

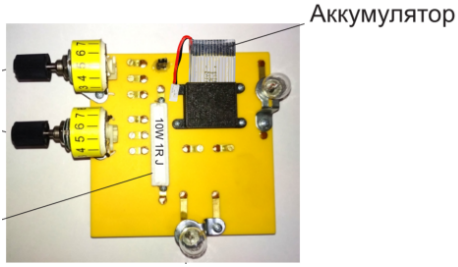


3. Иллюстрация экран-эффекта



4. Иллюстрация экран-эффекта





5. Проведение измерений, запись результатов в таблицу.
6. Построение графиков зависимостей (в цифровой лаборатории)

2.9 Занятие 12.

Конструирование измерительного прибора – термоанемометра. Ламинарные и турбулентные потоки. Обсуждение нелинейных особенностей течения газов и жидкостей. Обсуждение конструкции термоанемометра, его чувствительного элемента и обслуживающей электрической цепи. Настройка прибора, меры предосторожности для предотвращения выхода элемента из строя.

Цели занятия

Обсуждение важности датчиков и измерительных систем. Обсуждение идей конструирования приборов, возникающих конструкционных

альтернатив и их влияние на качество измерений и сложность прибора. Конструирование измерительной схемы с помощью модульной электроплаты. Калибровка и применение термоанемометра, проведение эксперимента.

Тренируемые навыки и приобретаемые знания

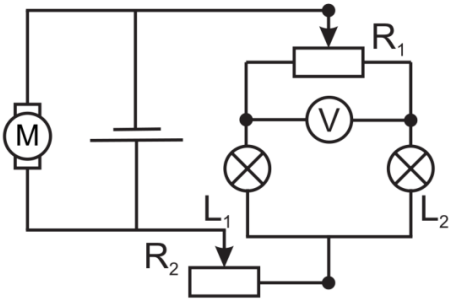
- Понимание различий между типами анемометров: термо и механических.
- Калибровка электрической схемы термоанемометра.
- Общие принципы построения измерительных приборов.
- Альтернативные способы реализации анемометра.
- Обеспечение безопасности при пользовании измерительными приборами с открытым чувствительным элементов.

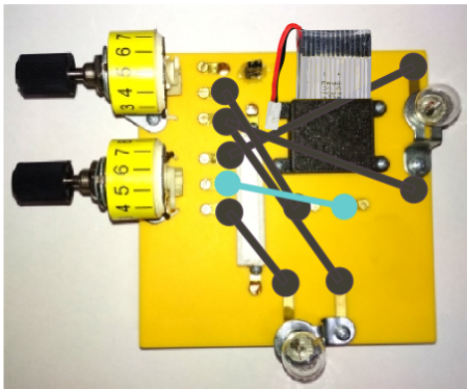
Краткое содержание урока

1. Обсуждение задачи измерения потоков воздуха. Неоднородность скорости движения воздуха.
2. Простейшие анемометры.

3. Механические анемометры.

4. Конструирование и настройка термоанемометра.





5. Установка чувствительного элемента термоанемометра:

6. Проведение эксперимента и запись результатов в таблицу, построение графика.

Следующие сдвоенные занятия 13-14 и 15-16 посвящены подготовке к сборке и сборке «Двуногий робот Gekko Ходок».

Этот сегмент программы предоставляет возможность ознакомиться с электромеханической частью шагающего робота (в т.ч, сервомоторами), а также с его программным управлением.

2.10 Занятие 13-14.

Предварительная подготовка сервоприводов для сборки робота Gekko «Ходок»

Цели занятия

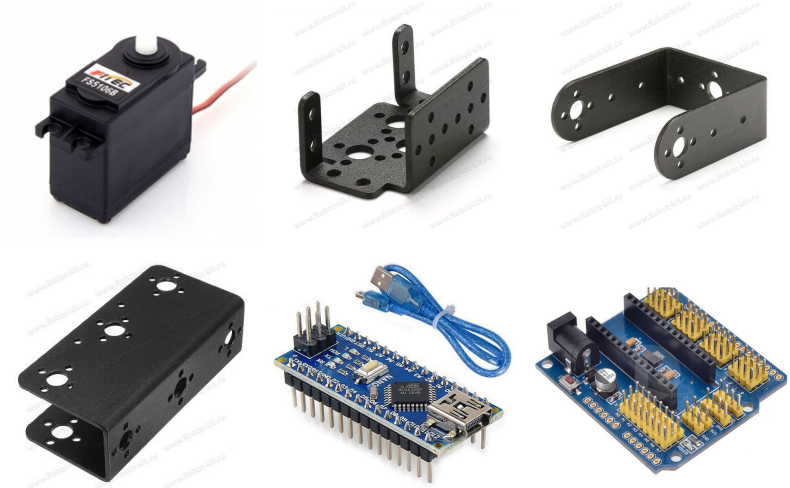
Ознакомление с деталями набора «Двуногий робот Gekko Ходок». Ознакомление с платой управления. Сборка схемы управления сервоприводами. Ознакомление со скетчем «sl_Servo_test.ino». Ознакомление с библиотекой Servo <Servo.h>. Ознакомление с функцией map(value, fromLow, fromHigh, toLow, toHigh). Получение для каждого из сервоприводов предельных величин длительностей импульса, которые переводят сервопривод из одного крайнего положения в другое. Заполнение таблицы крайних и средних положений сервоприводов.

Тренируемые навыки и приобретаемые знания

- Подключение сервоприводов к Arduino Nano с платой расширения.
- Управление сервоприводами с помощью программы Arduino IDE.
- Работа с монитором порта, библиотекой Servo <Servo.h> и функцией map.
- Работа и настройка сервомоторов.

Краткое содержание урока

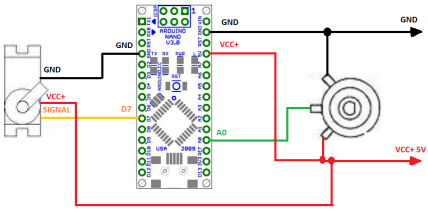
1. **Ознакомление с деталями набора «Двуногий робот Gekko Ходок»**
   Сервомоторы FS5106 компании Feetech
   Набор алюминиевых кронштейнов для сервоприводов Arduino Nano
   Плата расширения для Arduino Nano



2. **Ознакомление с платой управления**
   Arduino Nano с платой расширения
   Обозначение выводов Arduino Nano

pictures2/pic46.png	pictures2/pic47.png

3. **Сборка схемы управления сервоприводами** К плате управления к выводам GND и 5v подключаются крайние выводы переменного резистора (1 - 10 кОм), средний вывод подсоединяем к 0.



Сервопривод, которым мы будем управлять с помощью переменного резистора, подключаем к выводу 7 (D7).

4. **Ознакомление со скетчем «sl_Servo_test.ino»** Загружаем скетч «sl_Servo_test.ino»

```
#include <Servo.h>        // Подключение библиотеки Servo
int PtnPin = 0;           // Подключаем потенциометр к пину A0
int SrvPin = 7;           // Подключаем сервопривод к пину D7
int MinTiming = 350;      // Переменная для нижней
                          // границы длительности
int MaxTiming = 2800;     // Переменная для верхней
                          // границы длительности
int i = 0;                // Переменная для текущих значений
                          // потенциометра (0 - 1023)
int j = 0;                // Переменная для текущих значений
                          // TestServo (MinTiming - MaxTiming)
Servo TestServo;          //создаём объект "TestServo" типа Servo
                          // для управления сервоприводом

void setup(){
    pinMode(PtnPin,INPUT);  // Назначаем PtnPin как вход
    TestServo.attach(SrvPin,MinTiming,MaxTiming);
                            // Подключаем TestServo к пину SrvPin
    Serial.begin(9600);     // Инициализация Serial-соединения
}

void loop(){
    i = analogRead(PtnPin); // Считываем текущее значение
                            // сигнала от потенциометра
```

```
    j = map(i,0,1023,MinTiming,MaxTiming);
                            // Преобразование сигнала из
                            // 0 - 1023 в MinTiming - MaxTiming
    TestServo.writeMicroseconds(j);
                        // Перемещение сервопривода в
                        // положение соответствующее сигналу
    Serial.println(j);  // Отправка текущего значение
                        // в монитор порта
    delay(100);         // Задержка на выполнение перемещения
}
```

Вращая переменный резистор, мы управляем вращением сервопривода и на мониторе порта видим текущее значение угла поворота в миллисекундах от MinTiming до MaxTiming.

5. **Ознакомление с библиотекой Servo <Servo.h>.**
Библиотека Servo позволяет осуществлять программное управление сервоприводами. С помощью нее можно достаточно просто задать положение сервопривода. Причем положение можно задавать как в градусах, так и в микросекундах. Для этого заводится переменная типа Servo. Управление осуществляется следующими функциями:
attach() — присоединяет переменную к конкретному пину. Возможны два варианта синтаксиса для этой функции:
servo.attach(pin) и servo.attach(pin, min, max). При этом pin — номер пина, к которому присоединяют сервопривод, min и max — длины импульсов в микросекундах, отвечающих за углы поворота 0° и 180°. По умолчанию выставляются равными 544 мкс и 2400 мкс соответственно.
write() — отдаёт команду сервоприводу принять некоторое значение параметра. Синтаксис следующий:
servo.write(angle), где angle — угол, на который должен повернуться сервопривод.
writeMicroseconds() — отдаёт команду послать на сервоприводимульс определённой длины, является низкоуровневым аналогом предыдущей команды. Синтаксис следующий: servo.writeMicroseconds(uS
где uS — длина импульса в микросекундах.
read() — читает текущее значение угла, в котором находится сервопривод. Синтаксис следующий:

Номер сервы	Минимум	Максимум	Середина
1	500	2 500	1 500
2	500	2 500	1 500
3	500	2 500	1 500
4	500	2 500	1 500
5	460	2 570	1 510
6	500	2 500	1 500

servo.read(), возвращается целое значение от 0 до 180.

6. **Ознакомление с функцией map(value, fromLow, fromHigh, toLow, toHigh)**
В наборе команд Ардуино в разделе Математические вычисления есть функция map(value, fromLow, fromHigh, toLow, toHigh) map преобразовывает значение переменной value , равное fromLow, в число toLow, а значение fromHigh — в toHigh. Все промежуточные значения value масштабируются относительного нового диапазона [toLow; toHigh].

   В нашем случае:
   j = map(i, 0,1023, MinTiming, MaxTiming); //Преобразование сигнала от переменного резистора от 0 до 1023 в MinTiming - MaxTiming

7. **Получение для каждого из сервоприводов предельных величин длительностей импульса.**
Вращая переменный резистор, мы находим крайние значения вращения сервопривода и на мониторе порта видим текущее значение угла поворота в миллисекундах Заносим эти Min и Max значения в таблицу и затем определяем середину (берем средне арифметические значения)

   Заполнив таблицу крайних и средних положений сервоприводов, мы осуществим предварительную подготовку сервоприводов для сборки робота.

2.11 Занятие 15-16.

Сборка и подготовка к эксплуатации двуногого робота Gekko «Ходок».

Цели занятия

Собрать корпус (скелет) робота «Ходок». Собрать и закрепить сервоприводы Закрепить плату управления.

Тренируемые навыки и приобретаемые знания

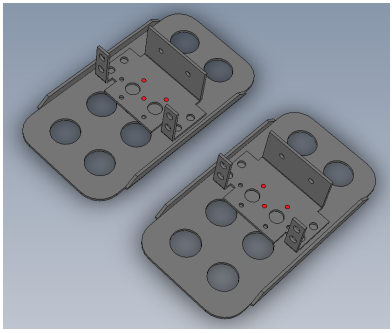
- Механическая сборка деталей набора робота при помощи отвертки, круглогубцев, пинцета или рожковых ключей. Работа с подшипниками.
- Сборка сложной пространственной конструкции.
- Работа с сервоприводами – установка «качалок», крепление к корпусу, определение свободного хода суставов робота. Крепление платы управления.

Краткое содержание урока

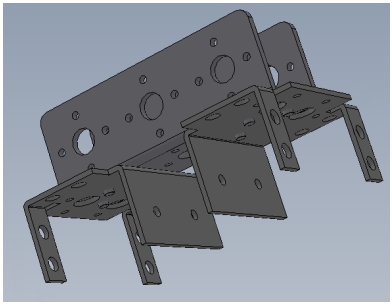
Для сборки робота нам понадобятся:
- Детали набора «Двуногий робот Gekko Ходок» с которым мы познакомились на прошлом уроке.
- Малая и средняя крестообразные отвертки с намагниченными наконечниками.
- Круглогубцы малые, пинцет или рожковые ключи на 3 и 5,5мм.
- Винты DIN965 M2x5.
- Винты DIN7985 M2x5.
- Винты DIN7985 M2x16.
- Гайки DIN934 M2.
- Винты DIN7985 M3x8.
- Винты DIN7985 M3x10.
- Гайки DIN934 M3.

- Гровер DIN127 M3.
- Шайбы DIN125 M3.
- Шурупы DIN7981 2.2x6.5.
- Подшипники 8x4.

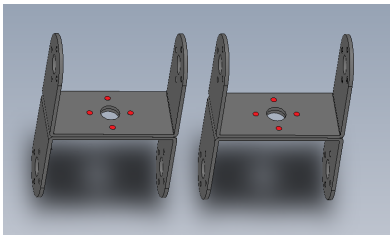
1. **Сборка «Стопы»**
   Скрепляем элементы вместе как показано на картинке. При этом винты DIN965 M2x5 (заподлицо) вставляем через отверстия указанные на рисунке красным со стороны пола, гайки DIN934 M2 затягиваем ключом или круглогубцами с той стороны, которая видна на картинке как верхняя.



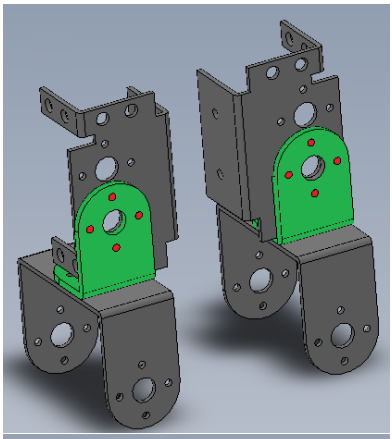
2. **Сборка «Нижней части туловища»**
   Соединяем винтами DIN7985 M2x5 и гайками DIN934 M2 как на картинке

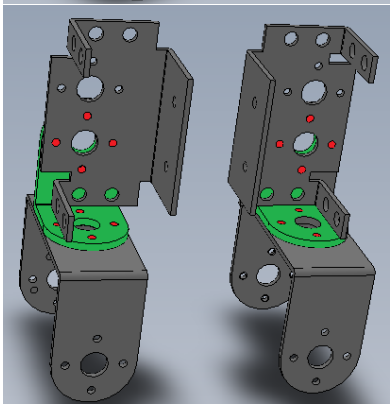


3. **Сборка «Бедра»**
   Соединяем винтами DIN7985 M2x5 и гайками DIN934 M2 как на картинке



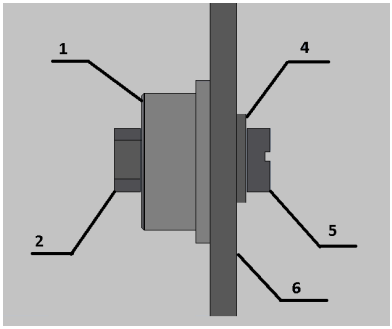
4. **Сборка «Голени»**
   Соединяем винтами DIN7985 M2x5 и гайками DIN934 M2 как на картинке



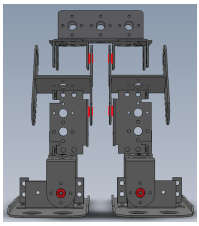


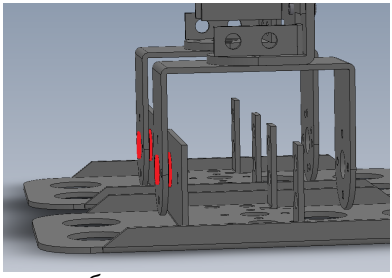
5. **Установка шарикоподшипников с фланцем Gekko ABB-**

03 и сборка скелета робота. Соединяем элементы вместе как показано на картинке
1 – Подшипник 8*4; 2 – гайка М3; 4 – гравер М3; 5 – винт М3-10; 6 – Кронштейн Многоцелевой для сервоприводов

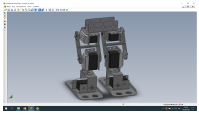


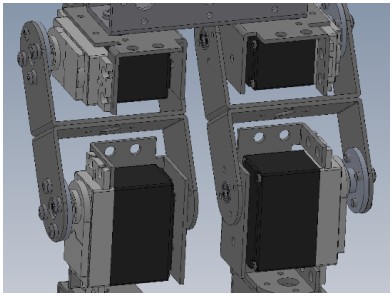
Подшипники устанавливаем по схеме



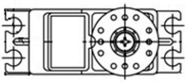


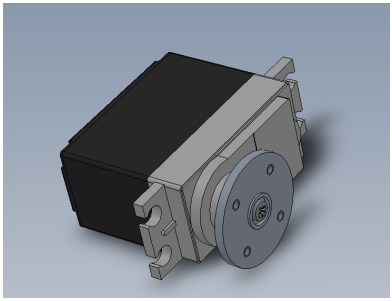
По это же схеме собираем скелет робота.
Чтобы лучше разобраться, запускаем исполнительный файл << *Hodok_3D.easm* >>, в котором, в среде SolidWorks eDrawings Viewer, сделана 3D модель этого робота.
Модель можно вращать, перемещать или менять масштаб.



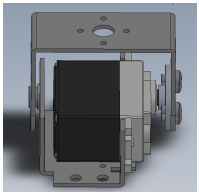


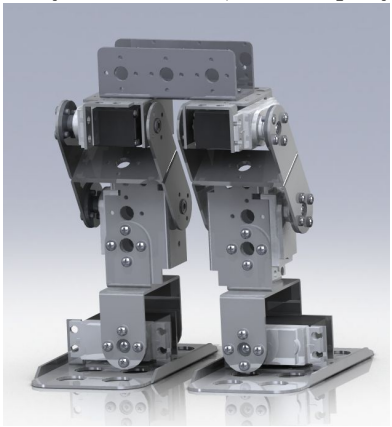
6. **Присоединение «качалок»**
   Круглые качалки с отверстиями для монтажа одеть на выходной вал (шестеренку) сервопривода и закрепить винтом DIN7985 M3x8 как на рисунке





7. **Крепление сервоприводов**
   Необходимо вставить все сервоприводы на свои места, в Кронштейны Многоцелевые и прикрутить их винтами DIN7985 M3x8 с гайками DIN934 M3 через шайбу DIN125 M3, как на рисунках.





   Что бы лучше разобраться смотрите 3D модель этого робота
   При сборке необходимо провести провода от сервоприводов так, чтобы они находились на своих местах, и в последствии не пришлось снимать из-за них сервоприводы.
   После установки сервоприводов нужно шурупами DIN7981 2.2x6.5 прикрутить «качалки» к элементам робота. Нужно обратить внимание что бы у подвижных частей робота сохранилась нормальная степень свободы. Для бедренного и сустава стопы в этом

особой проблемы не будет т.к. эти суставы конструктивно имеют свободу ниже, чем у сервопривода, поэтому нужно разместить сервомоторы так что бы суставы были полностью свободны по всей траектории. Для коленного сустава свободы сервоприводов не хватает, и поэтому к установке его сервомотора подойти с особой тщательностью, что бы обеспечить на обеих ногах робота одинаковую свободу.

8. **Крепление платы управления** Arduino Nano с платой расширения, винтами DIN7985 M2x16 и гайками DIN934 M2, через пластиковые столбики, крепим к «Нижней части туловища» как на картинке.