Дополнительные материалы к курсу информатики 5-6, 7-9 классы на основе завершенной предметной линии учебников «Информатика» для 5–9 классов общеобразовательных учреждений Л.Л.Босовой, А.Ю.Босовой

Часть вторая (7-9 классы)

Время реализации модуля по робототехнике (включая программирование) – 20 час.

Авторы:

Воронина Вероника Вадимовна
учитель информатики МБОУ СШ №7 г.Павлово Нижегородской области

Воронин Игорь Вадимович
начальник отдела информационных технологий Института Проблем Лазерных и Информационных технологий Российской Академии Наук

Павлово, Москва

2016

Содержание

В современном мире цивилизованного человека на каждом шагу может ожидать встреча с роботами.

Так что же такое робот? Какие роботы бывают? Как ими управлять и что нужно знать, чтобы самому сделать робота или управлять им?

Об этом мы начали говорить еще в пятом классе, теперь же продолжим нашу работу...

§ 15 Эпизод пятнадцать. Такие разные контроллеры. 7 класс

Привязка к тематическому планированию по информатике: Компьютер, как универсальное устройство обработки информации. Основные компоненты персонального компьютера.

Наши УМКИ недаром называются Smart Car – Умные машинки. Раз они умные, значит могут думать, значит даже в игрушках должно быть устройство которое будет думать, и каким-то образом общаться с нами.

Каждый робот как мы уже это обсуждали тогда становится полезным, когда его можно запрограммировать на выполнение нужных нам функций.

Устройство, которое используется для управления, в электронике и вычислительной технике получило название «контроллер».

15.1 Различные контроллеры

Чем же отличаются роботы с которыми мы общались создавая различные программы, от простых механических или даже радиоуправляемых игрушек?
Наши УМКИ недаром называются Smart Car – Умные машинки. Раз они умные, значит могут думать, значит даже в игрушках должно быть устройство которое будет думать, и каким-то образом общаться с нами.
Такое устройство, которое используется для управления, в электронике и вычислительной технике получило название «контроллер»

Нас с вами будет интересовать непосредственно микроконтроллер – такая хитрая микросхема, внутри которой находится самый настоящий компьютер. В этом компьютере есть все, что необходимо для самостоятельной работы:

- процессор
- оперативная память - ОЗУ
- постоянная память - ПЗУ
- генератор тактовой частоты
- таймеры
- порты ввода/вывода
- последовательные интерфейсы
- и много чего еще...

15.2 Какие бывают контроллеры

Контроллеры могут применяться в самых разных областях техники

Игровой контроллер – нужен для игр в качестве устройства ввода информации. Контроллер такого вида обычно соединяется с игровым устройством (приставкой или ноутбуком). Другими словами, это пульт управления, джойстик. Он может быть в виде:

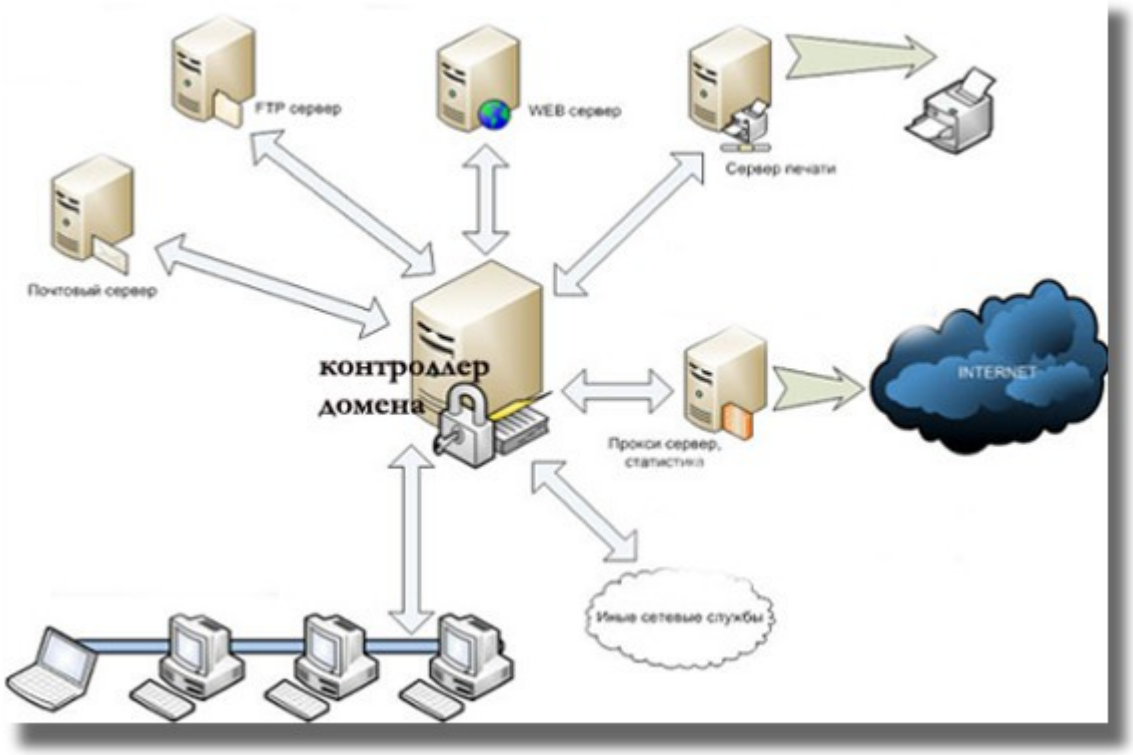
- руля;
- педалей;
- рычага;

- специальной клавиатуры;
- пистолета и т. п.

Промышленный контроллер – необходим для автоматизации техпроцессов. Такой контроллер должен быть высоконадежен, рассчитан на работы в тяжелой промышленности. Иногда этот тип контроллеров используется для наладки инженерно-технических установок зданий промышленных предприятий – отопление, освещение, вентиляция...

Контроллер прерываний – это микросхема или блок, встроенный в процессор, который отвечает за обработку запросов прерываний работы различных устройств.

Контроллер домена – устройство, которое контролирует компьютерную сеть – сервер. Такие контроллеры хранят каталоги данных, управляют взаимодействием пользователей в игре.



Микроконтроллер – (для управления электронными устройствами). В микроконтроллер закладывается программа, которая управляет различными электронными устройствами и осуществляет взаимодействие между ними. Микроконтроллеры можно встретить не только в компьютере, но даже в различных бытовых предметах, будь то телевизор, холодильник или кофемашина.

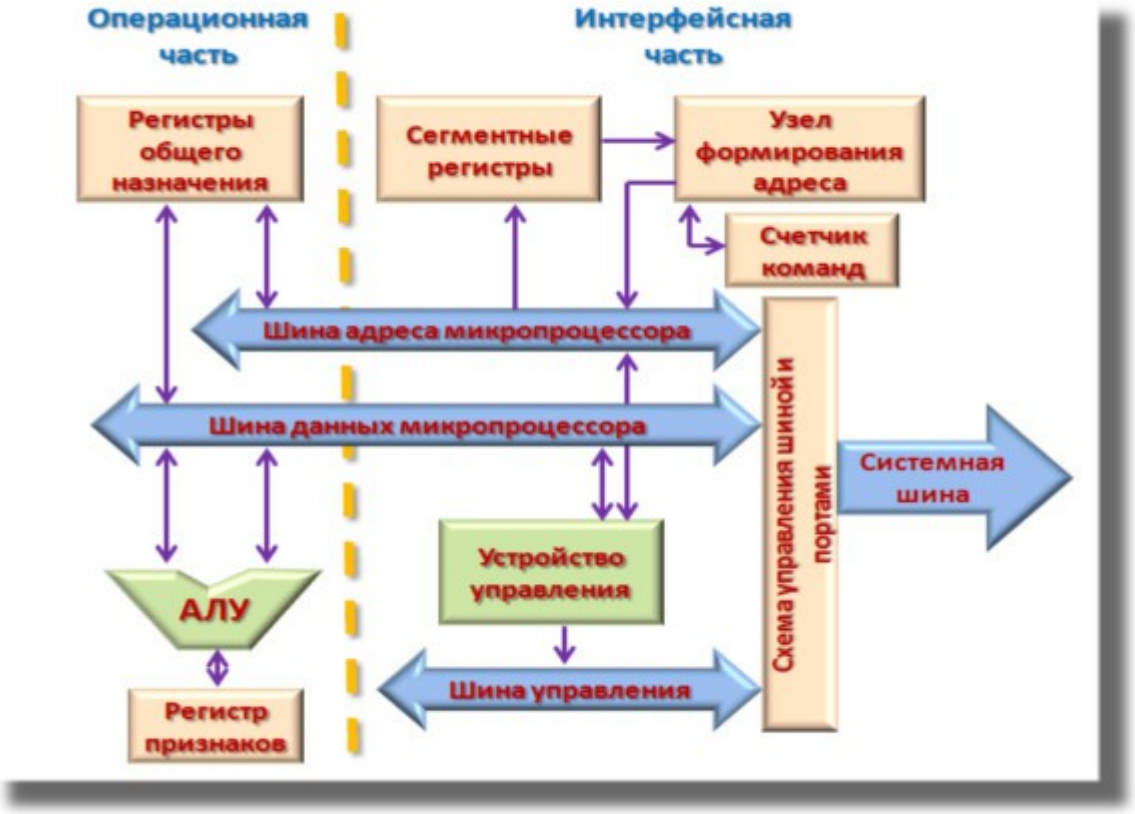
Системный контроллер – отвечает за организацию взаимодействия оперативной памяти и процессора.

Контроллер памяти – микросхема, которая отвечает за поток данных оперативной памяти.

Нас с вами интересует микроконтроллер, именно с микроконтроллерами мы будем иметь дело, называя их в дальнейшем, просто – контроллеры.

15.3 Как устроен контроллер

Самый главный элемент любого процессора – арифметико-логическое устройство , которое производит арифметические и логические операции с двоичными числами.



Чтобы микроконтроллер понимал, что мы от него хотим, в него должна быть загружена прошивка – последовательность действий, которую ему необходимо выполнить. Программируют микроконтроллеры обычно на языке Си или на ассемблере, а компилятор сам уже переводит программу с языка программирования в двоичные коды.

Однако же, программа, выполняющаяся внутри микросхемы и совершенно не связанная с внешним миром, была бы просто никому не нужна. Для полноценной работы, контроллеру необходимо обмениваться данными с внешним миром.

Для этого существуют порты ввода/вывода.

Порт – это набор однобитных каналов, каждый из которых может быть независимо настроен либо на ввод, либо на вывод. То есть каждый канал кодируется одной двоичной цифрой: либо нулем либо единицей. Любая программа для контроллера начинается именно с настройки портов. Мы должны определить, какие каналы будут работать на ввод, какие – на вывод.

Ну и конечно, при проектировании контроллеров, необходимо знать физику – **основной закон электротехники – Закон Ома**

Разные изготовители разрабатывают разные по функциональности и стоимости контроллеры.

15.4 Микроконтроллеры

Разные изготовители разрабатывают разные по функциональности и стоимости контроллеры.

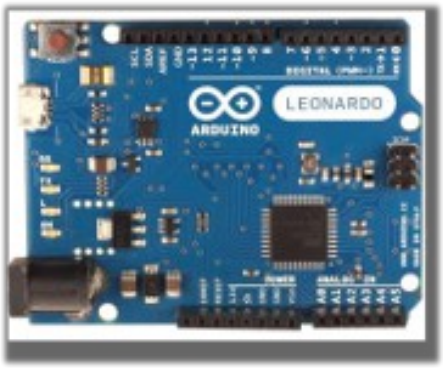
Контроллеры, могут быть такими:

SmartCar3

Arduino Nano

ТРИК

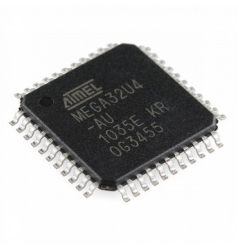
Arduino UNO



Arduino Leonardo

Чем же они отличаются?

Как мы уже говорили самая главная часть любого контроллера – микропроцессор. Отличия в количестве ножек и функциональных возможностей.



И соответственно по различным возможностями и внешнему виду отличаются сами контроллеры.

Например, в контроллере **ТРИК** все разъемы выведены на корпус, процессор ARM который используется в мобильниках и планшетах. Контроллер ТРИК разработан специально для робототехники и способен одновременно обрабатывать видео и звуковую информацию, стинтезировать речь, и управлять моторами, собирать показания с датчиков и обмениваться информацией по беспроводной связи. ТРИК оснащен цветным сенсорным дисплеем, программируемыми кнопками, у него есть поддержка WiFi и Bluetooth, при наличии адаптера ZigBee ТРИК может связываться с компьютером по радиоканалу.

Контроллер **SmartCar3** – опирается на процессор Xbee, который позволяет организовать распределенную сенсорную сеть, обеспечивающую взаимодействие множества агентов. Агенты обмениваются информацией между собой и используя доступную информацию,

получаемую от ближайших соседей, агенты-мобильные роботы сумеют, например сформировать определенный строй и двигаться к заданной цели, сохраняя его. Обмен информацией с компьютером осуществляется по беспроводному протоколу ZigBee.

Остальные контроллеры семейства Arduino на базе процессора AVR Atmega. В отличие от ТРИК, разъемы этих контроллеров составлены из штырьков на лицевой части платы. С компьютером все рассмотренные платы Arduino осуществляют связь по USB.

15.5 Кто может делать контроллеры

Каждый робот, как мы уже обсуждали, тогда становится полезным только тогда, когда его можно запрограммировать на выполнение нужных нам функций.

Чтобы программа работала по заданному алгоритму, она должна быть написана, отлажена и загружена в контроллер, для приведения в действие исполнительных механизмов — частей робота.

Так что же такое, контроллер? Из чего он состоит и как с ним обращаться? Поговорим об это подробнее...

Контроллеры бывают серийно выпускаемые, от ведущих компаний производителей — таких как напримерNational Instruments, Siemens, Philips, а бывают, и самодельные.

Самому сделать контроллер совсем несложно. Надо только обладать некоторыми специфическими знаниями — знаниями, дающими вам СИЛУ и набором нужных комплектующих. Вот и все...

Давайте разберем, как же нам сделать свой собственный контроллер.

15.6 Так чем же они, все-таки, отличаются?

Для начала, необходимо определиться, на базе какого микропроцессора (ЧИПа) будет сделан наш контроллер.

Чипы различаются по: количеству ножек — 8, 24, 50, 100 и более и по тактовой частоте. Чем больше тактов за секунду успевает сделать ЧИП — тем лучше. Сейчас, во встраиваемой электронике больше всего распространены контроллеры с частотой 1,6 ГГц, т.е. за одну секунду контроллер осуществит 1 600 000 000 (1 миллиард 600 миллионов) операций!

Но, с другой стороны — чем больше частота, тем больше расход энергии :(.

Если мы делаем автономного робота, то помнить о том, что энергия не бесконечна, а зависит от заряда батарей и расхода на силовые агрегаты — придется постоянно.

По архитектуре — наиболее сейчас распространены AVR микропроцессоры, бывают MSP микропроцессоры, но более производительными являются чипы с архитектурой ARM. Логично, что такие процессоры и будут более дорогими.

На всех планшетах, телефонах и других гаджетах встроены процессоры с архитектурой ARM.

И каждый производитель уверен, что только он выпускает самый лучший в мире процессор!

Теперь, достаньте свой телефон, запишите марку, и используя поисковые системы, определите, на базе какого микропроцессора разработан ваш телефон.

Определите микропроцессоры на базе которых разработаны наиболее популярные марки телефонов ваших родителей, телефоны ближайших друзей.

Чей телефон	Марка и модель телефона	Страна-производитель	Микропроцессор	Рабочая частота процессора	Цена новой модели на момент исследования	Отношение Цена/Частота процессора

Найдите в Интернете сколько стоит на момент исследования каждая модель телефона. Рассчитайте ориентировочную стоимость одного мегагерца, посчитав отношение Цены к Рабочей частоты процессора.

Сделайте вывод, насколько отличается цена у телефонов с одинаковыми характеристиками.

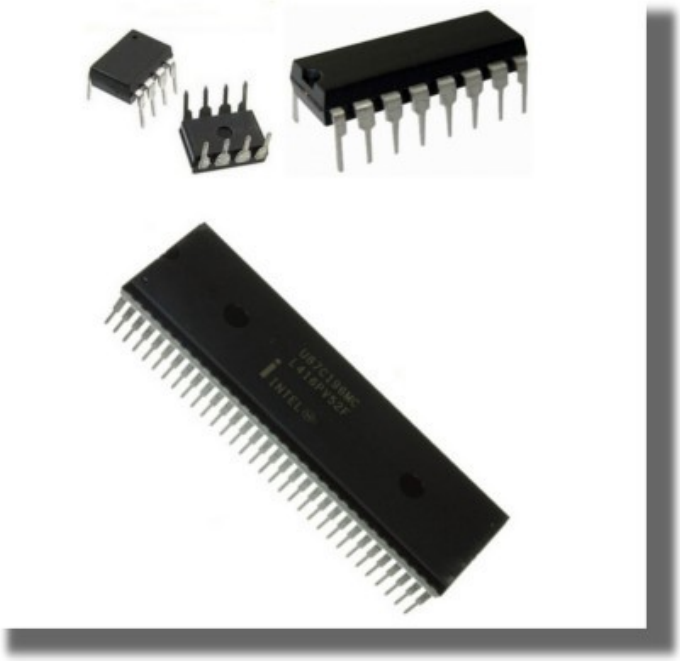
Сформулируйте, за что вы переплачиваете, делая дорогие покупки.

15.7 Сделай сам!

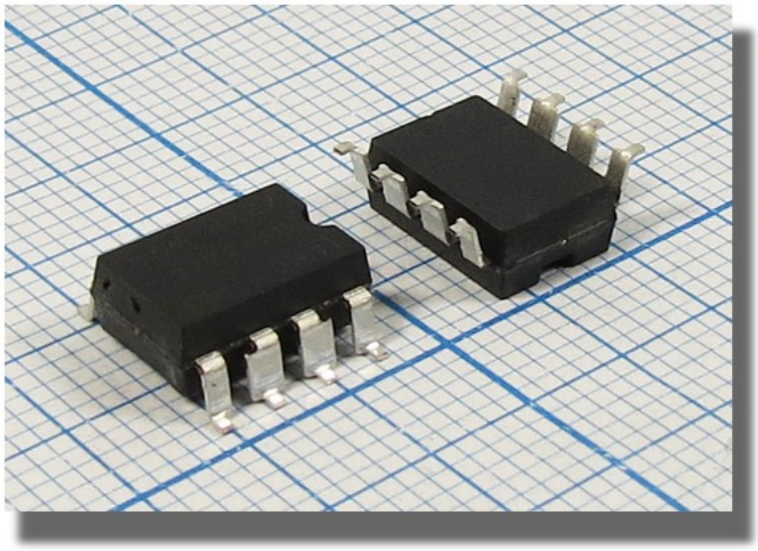
После того, как выбрали подходящий для нас по цене и по функциональным возможностям процессор, мы начинаем в специализированной программе разводить токопроводящие дорожки от каждой ножки процессора до нужного нам элемента на плате. При этом, конечно нам обязательно придется использовать ранее полученные знания по физике и, в частности, закон Ома.

Один и тот же процессор можно купить в разных корпусах. Корпуса бывают DIP либо SMD:

DIP-корпус можно без всякой пайки просто вставить в посадочные разъемы отладочной платы и перемычками соединить его ножки с исполнительными элементами, например, светодиодами, и подав питание на плату помигать этими светодиодами по программе,в которой команды управления в заданный момент времени идут на ножки процессора.



SMD корпус — гораздо меньше по размеру. Вот его-то уже обязательно надо припаивать к плате.



После того как мы продумали принципиальную схему контроллера, необходимо составить схему монтажную, и разработать спецификацию на закупаемые элементы. Потом надо будет изготовить плату и самому напаять на нее все нужные элементы, но конечно можно и попросить того, кто хорошо умеет паять помочь вам это сделать.

Так получаются самые первые простые контроллеры, которые со временем смогут развиться до масштабов управления космическими кораблями.

В чем такая популярность платы Arduino? Скорее всего в том, что ее разработчики первыми придумали при изготовлении платы предусмотреть схему программатора — на самой плате. И теперь, чтобы поменять код для исполнения в процессоре — достаточно просто соединить плату с компьютером и выполнить загрузку кода.

Для профессиональных же контроллеров, чтобы поменять программу надо использовать, как правило, специальную плату, которая одним шнуром соединяется с компьютером, а другим с контроллером — такие специальные платы и называют программаторами. Использование программаторов связано с тем, что на

промышленном контроллере борьба идет за каждый лишний квадратный сантиметр платы – для ее дешевизны. И, как правило, однажды «прошитые» на заводе-изготовителя платы контроллеров конечным потребителем уже больше не перепрограммируются, разве что, только в самых экстренных случаях.

А перепрограммирование платы Arduino – это нормальный рабочий процесс, потому что эта плата предназначена как раз для обучения. И для Arduino были разработаны специальные библиотеки о которых говорили когда обсуждали программирование.

§ 16 Эпизод шестнадцать. «Языки программирования для роботов». 7 класс

Привязка к тематическому планированию по информатике: Компьютер как универсальное устройство обработки информации. Программное обеспечение. Правовые нормы использования программного обеспечения.

16.1 Программирование на низком и высоком уровнях

Слышна музыка чир-данса –
Не мелодия романса.
Интенсивность, чёткий ритм –
Всех движений алгоритм

В предыдущих эпизодах мы с вами обсудили и пришли к соглашению, что роботом агрегат становится тогда, когда его можно программировать.

Программирование, как вы наверняка знаете, это запись алгоритма неких действий, на языке, понятном исполнителю.

Самой интеллектуальной частью робота является Контроллер, в котором установлен чип, прошитый[1] программным кодом. Согласно алгоритмам этого кода робот и действует, так, как от него добивается разработчик.

Программы бывают разного уровня. Та, что зашита в контроллере — это низкоуровневая программа. Условно говоря, она описывает как выполнить заданную ему команду по движению вперед. Какие моторы и в какую сторону начать крутить.

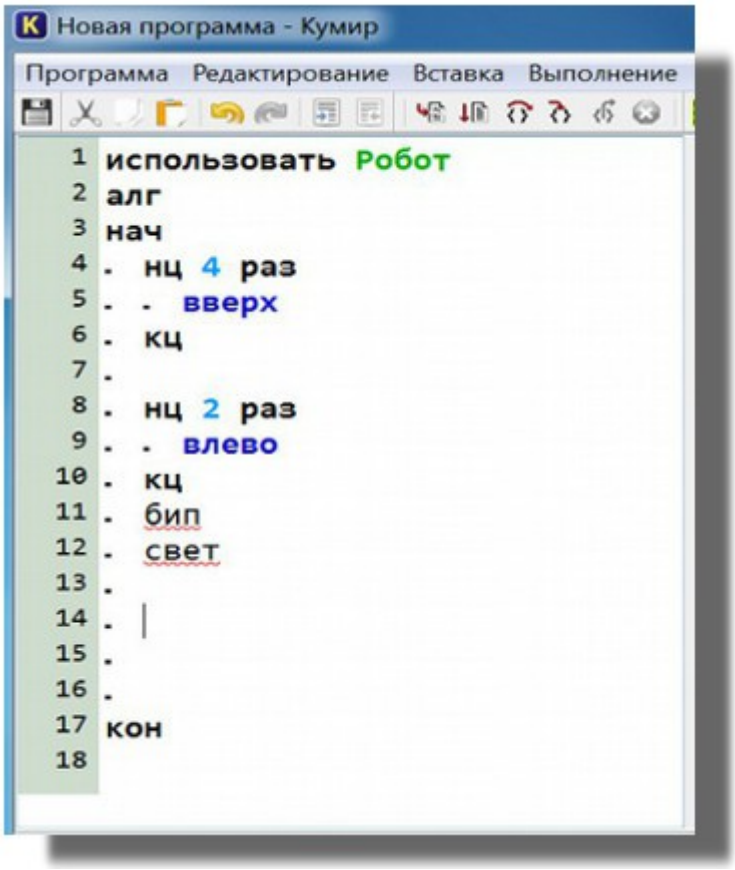
Например, программа может иметь такой вид:

```
void Vench::BPressLeft()
{
  SbrosDatKas();
   if(fl_play)return;
   if(fl_rec){
   b_tim_comm=GetTime();
            }
  if (shiftPress==0) SendCommLeft1();
  else SendCommExtrLeft1();
  PressLf=1;
}
```

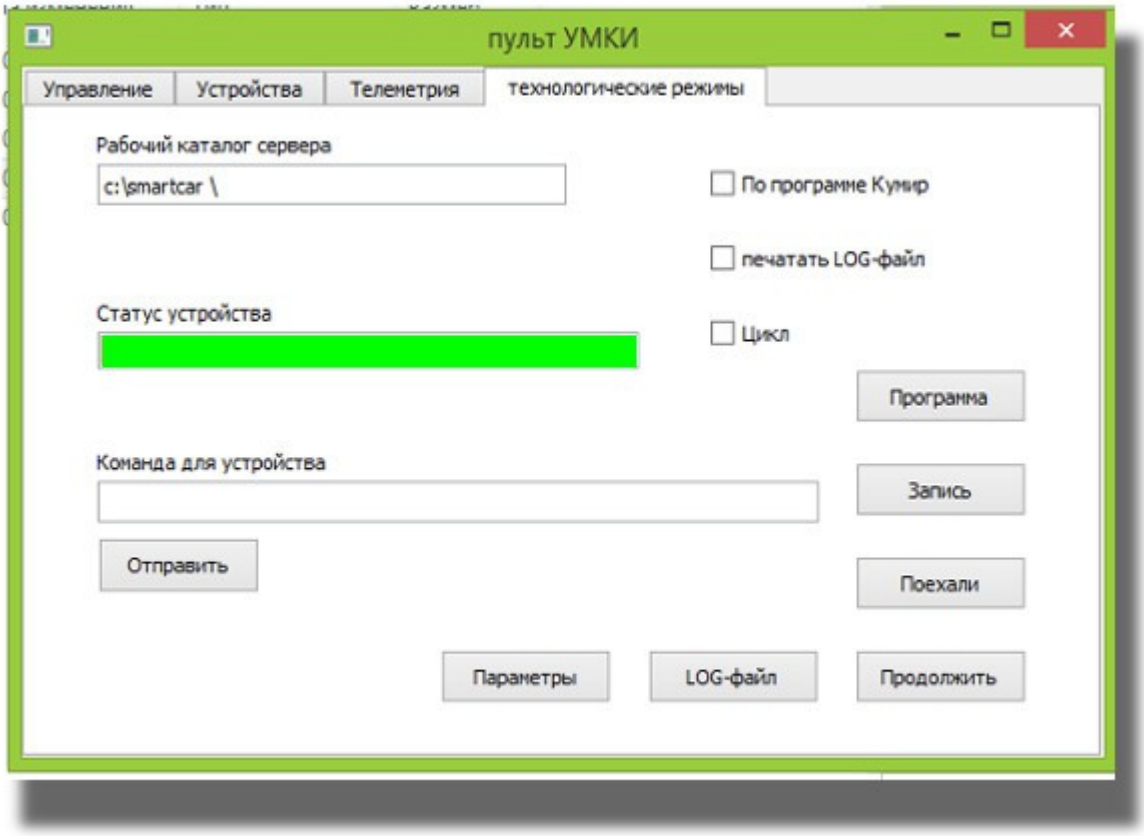
А то действие, которое вы задаете роботу в общем виде – куда поехать, что там сделать – это программа высокого уровня.

Например:

1 Прошивка – программа хранящаяся в постоянной памяти устройства. Эти программы могут меняться, поэтому время от времени возникает необходимость перепрошивать (обновлять) память компьютерных устройств.



Если для программ низкого уровня пишется и отлаживается сам код на языке исполнения, то для программ высокого уровня, кроме обработки действий, еще необходимо написать код интерфейса – связи человека с машиной. Это могут быть какие-то кнопочки, поля для ввода значений различных параметров, чекбоксы.

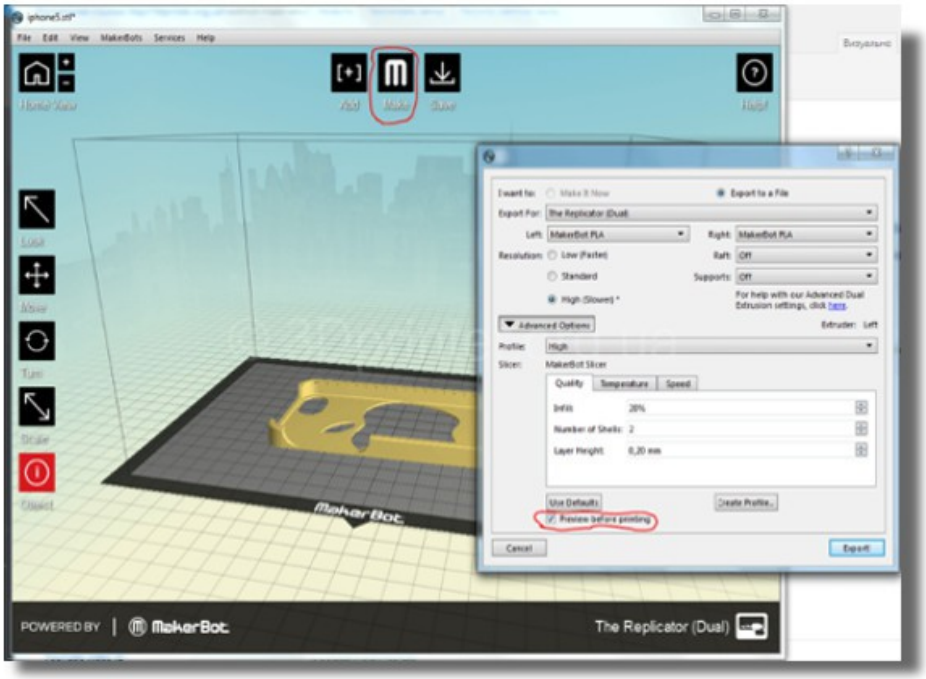


Программа высокого уровня может ожидать на входе скрипт действий, или набор команд в виде заранее написанного отладочного файла.

Так например, если вы положили рубашку в стиральную машину и выбрали программу быстрой стирки, то можно сказать, что вы задали через интерфейс, скрипт быстрой стирки, а программа внутри контроллера вашей стиральной машины, согласно этой директиве стала использовать ту часть своего кода, которая наливает поменьше воды, чем в обычном режиме, отжимает чуть быстрее, в-общем, экономит использование ресурсов.

В стиральных машинах, как вы знаете, бывают разные режимы работы – быстрый, бережный, льняные ткани, обычный режим. Использование каждого режима, запускает свой собственный скрипт для контроллера.

А если вы нарисовали 3D-модель, сгенерировали файл G-кодов и загрузили его на 3D-принтер, то можно сказать, что вы задали скрипт выполнения для вашего робота – принтера.



Как же так происходит, что написанные обычным человеком слова, наш робот понимает и выполняет с высокой точностью?

Такой случай впервые был описан в сказке Тысяча и одна ночь, когда Али-Баба сказал: «Сезам откройся», и дверь в пещеру растворилась. Пожалуй, «Сезам» это и был первый робот :).

16.2 Что понимает машина? Режим отладки

Сейчас роботы выполняют самые замысловатые действия, подчиняясь воли человека, если она правильно оформлена. Оформление вашего желания в программном коде, называется – программирование.

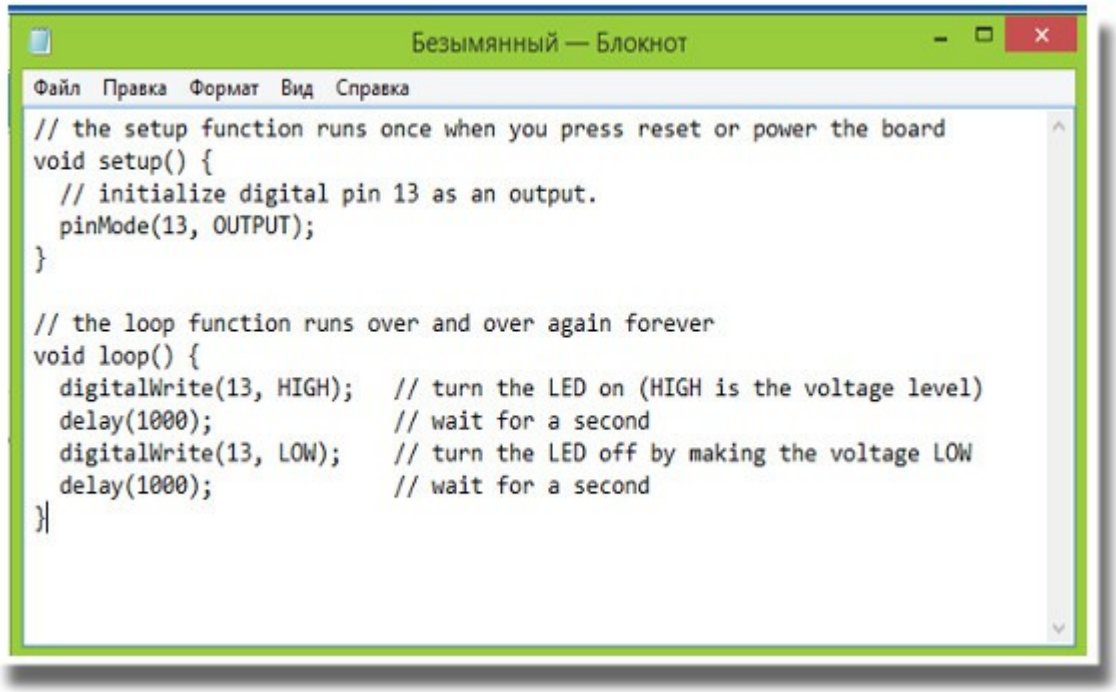
Программы пишутся на языках программирования. Языков программирования существует великое множество. Некоторые появляются, достигают своего расцвета популярности и потихоньку приходят в забвение.

Мы с вами — люди разговариваем на нашем человеческом языке – русском, английском, немецком... и программы также пишутся на языке понятном людям, но процессоры внутри контроллеров — не понимают человеческого языка. Они действуют согласно своей внутренней логике. Сейчас массово распространены микропроцессоры с двоичной логикой, на основе двоичной системы счисления. Раньше – на этапе создания первых ЭВМ был вариант развития троичной системы исчисления, который продвигали в СССР в 50-х годах двадцатого века. При помощи троичной системы счисления можно описать больше комбинаций при меньших ресурсах.

Но разработки советских ученых канули в лету, и мы сейчас пользуемся системой вычислений разработанной в американских лабораториях. Так же и языки программирования в подавляющем большинстве разрабатываются в Америке, и поэтому команды пишутся на английском языке.

После того как программа, состоящая из набора команд написана, она проверяется на синтаксические ошибки и если все в порядке, то переводится в машинные коды — понятные для исполнения контроллеру.

Вы можете написать текст программы для выполнения вашим роботом — допустим на arduino, в любом текстовом редакторе например она будет выглядеть так:



после того, как вы ее сохранили в виде файла, вам нужно ее проверить на наличие синтаксических ошибок — для этого запускается программа транслятор gcc и к ней на

вход подается имя с путем вашего файла. В случае ошибок, транслятор покажет в какой строке они присутствуют, их надо исправить и запустить транслятор еще раз и мы получаем объектный файл. После этого для перевода программы в машинные коды необходимо запустить компилятор, который соединит подготовленный вами код, с кодом указанных библиотек и создаст исполняемый файл.

Ура — победа! Теперь мы это файл загружаем в чип на плате контроллера, подаем на него питание и радостно смотрим как мерцает светодиод и куда поехала наша машинка.
Или увы не поехала... :(

Если контроллер ведет себя не так, как вы от него ожидали, то скорее всего, ваша программа написана правильно с точки зрения синтаксиса – здесь ошибок нет, но, с точки зрения логики действий что-то сделано не так... Не присвоено нужное значение какой-то переменной, или перепутан плюс с минусом, или еще что-то. Вариантов. к сожалению, бывает масса. Наступает самый увлекательный этап – поиск логических ошибок. Это называется отладка, или по английски — Debug, поиск жучков (программных ошибок).

Для этого используют инструмент отладки – программу gdb.

Вышеописанные операции можно выполнить в консольном режиме, запуская все эти программы из строки но для удобства разработки существует интегрированная среда, например, в Arduino она называется IDE Arduino. В ней, многие действия, которые мы с вами разобрали, скрыты от вас, и происходят автоматически – после нажатия нужной кнопки. О работе с этой интегрированной средой мы будем подробно говорить в курсе «Arduino. Первые шаги»

16.3 Библиотеки. Компиляторы, интерпретаторы, процедурные языки.

В arduino используется язык, очень похожий на язык Си. Т.е. если вы знакомы с языком Си, то вам будет легко понять команды, которые используются в arduino. И наоборот, если вы научитесь писать программы в arduino, то для вас не составит большого труда научится программировать в языке Си. Преимуществом arduino перед другими платформами является то, что для нее написано огромное количество библиотек и готовых программ с примерами, и все они выложены в открытый доступ в сети Интернет.

Каждая сложная программа состоит из набора простых команд, если вы пока не можете составить большую программу, то скорее всего вы сможете найти в Интернете нечто похожее, на то что нужно вам, слегка подправить готовый пример, и получить нужный результат.

Это вполне рабочий вариант – берется пример и слегка корректируется, но тут могут возникнуть подводные камни: скорее всего, в примере используются вызовы библиотек, которых может не оказаться у вас на компьютере. И транслятор на этапе синтаксической проверки выдаст сообщение, что вы вызываете функцию из библиотеки, которой он не видит. Для решения этой проблемы вам придется через поисковик найти файл с нужной библиотекой и положить ее в тот каталог на вашем компьютере, где ее пытается найти Arduino IDE и его транслятор. Благодаря тому, что arduino – открытый проект, найти нужную библиотеку, скорее всего, не составит труда.

Кроме того, вам надо знать, что все разработки выкладываются в общее хранилище, которое называется репозиторий. Один из самых распространенных — это github.com
В будущем, если вы научитесь писать такие программы, которые могут быть полезны другим людям и захотите, чтобы про вас узнали – вы сами тоже сможете зарегистрироваться и выкладывать туда плоды своих творений.

Мы с вами разобрали способ выполнения программы – компилятором, но этому есть альтернатива – интерпретатор. Для такого способа работы программы используются другие языки. Программист когда выбирает на каком языке ему писать программу, не всегда задумывается как она будет исполняться, но для работы со встраиваемой микроэлектронникой это бывает важно.

Языки компиляторы — это Си, Паскаль, и другие, компилятор который используется для их трансляции — это из свободно распространяемых gcc или minGW – для операционной системы Windows

Языки-интерпретаторы — это Basic, Кумир, Java, питон и другие. Причем языки Java и питон — вдобавок к основным своим преимуществам — еще и платформенно-независимые. Это значит, что однажды написанный код, неважно в какой операционной системе, могут точно так же работать и в любой другой: MacOS, Linux, Windows, Android, etc.

Еще бывают не алгоритмические, а процедурные языки, такие как Pl, SQL и другие, которые используются не для управления техникой, а для обработки и манипулирования большими объемами данных, для анализа поведения разных машин. Как мы уже разобрали ранее, бывают среды для разработки — это arduino IDE, geny и другие.

Так же очень удобно использовать фреймфорки: QT например или eclips. Они, в отличие от сред разработки, обладают гораздо большими возможностями и функционалом. В них, как правило, предусмотрены инструменты отладки кода. Во всех других случаях для поиска логических ошибок и просмотра состояния значений в переменных на работающей плате приходится использовать аппаратную часть JTag и программную часть – отладчик GDB.

16.4 Тук-тук, открыто!

Сравнительный анализ правового использования программного обеспечения на примерах языков программирования применяемых в образовательной робототехнике.

Так мы с вами рассмотрели шаги, которые необходимо совершить чтобы самому сделать свой контроллер. Но на первых порах вам, скорее всего, придется воспользоваться стандартными, доступными в коммерческом доступе контроллерами.

Конечно, хорошо создать такой контроллер который точно подходит под ваши задачи — например управляет освещением у вас в комнате, по хлопку в ладоши включает свет, а по свистку выключает, или сам, в автоматическом режиме насыпает корм и наполняет поилку вашему четвероногому другу.

Сделанный самостоятельно контроллер, если его запускать в серию для продажи — будет самым выгодным по цене изготовления.

Но, как мы разобрали раньше — проблема в том, что для самостоятельного изготовления контроллера, все-таки нужно обладать некоторыми профессиональными знаниями, которых у вас, скорее всего, пока еще не так много. Одним законом Ома тут, увы, не обойдешься :(.

Поэтому, для того чтобы попробовать, поэкспериментировать и научиться управлять ЧИПом, можно воспользоваться уже существующими на рынке, изготовленными до нас – универсальными контроллерами.

Их называют: средства разработки (по английски development kit , или starter kit). Если прямо такими словами искать в гугле-яндексе – устанете листать странички с разными ссылками на контроллеры от разных производителей, попробуйте сами.

Эти средства разработки существуют двух типов: с открытой технологией и с закрытой — проприетарной. Это значит, что для использования и продажи разработок на основе закрытой технологии необходимо покупать лицензию.

Таким образом, в Италии группа энтузиастов однажды разработала плату контроллера для изучения чипа Atmega, выложила ее схему и к ней разработанные библиотеки программ в открытый доступ, и им повезло — у них стали покупать эти платы в больших количествах.

Сейчас этот проект называется Arduino (Ардуино), и число решений реализованных на контроллере Arduino уже идет на миллионы. Благодаря тому, что схемы этого контроллера были опубликованы в открытом доступе, и что предложенное решение по архитектуре оказалось достаточно удачным— многие другие разработчики тоже стали самостоятельно придумывать всякие расширения и полезные модификации платы Arduino, и так появились их модификации — Arduino UNO, Arduino Nano, Arduino Leonardo и другие ветви этого клона.

§ 17 Эпизод семнадцать. Управление с помощью шестнадцатеричных кодов. 8 класс

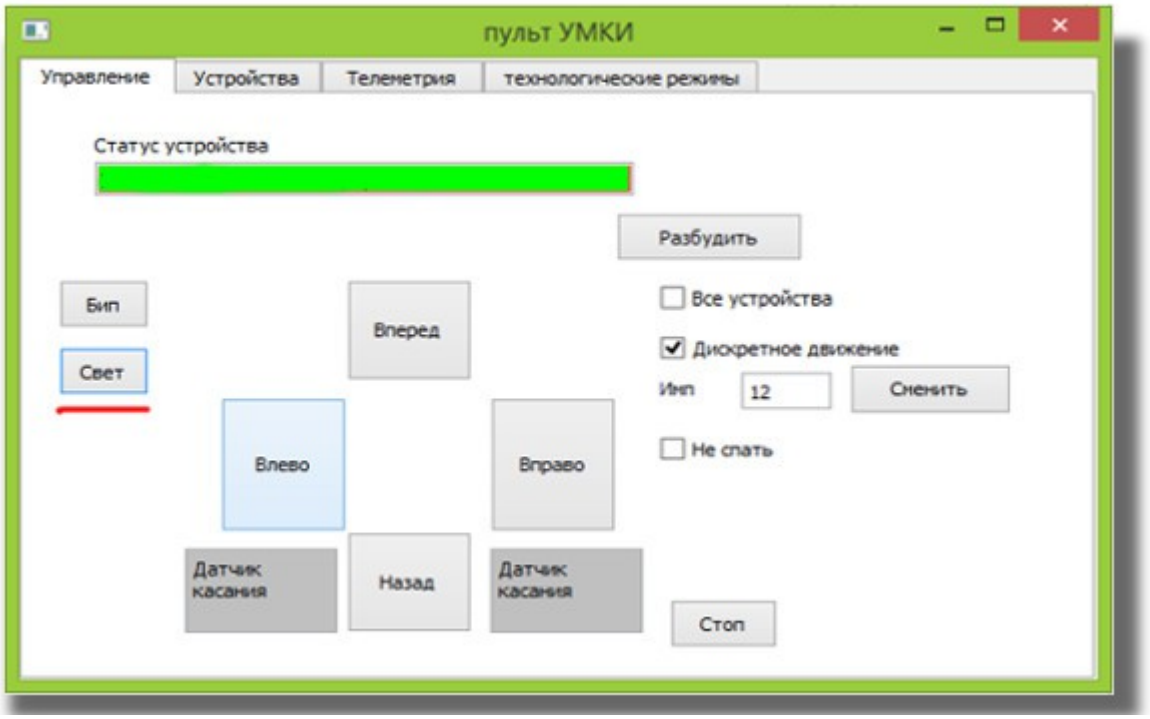
Привязка к тематическому планированию по информатике: Математические основы информатики. Знакомство с двоичной, восьмеричной и шестнадцатеричной системами счисления.

Рассмотрим подробнее управление платформой УМКИ SmatrCar с помощью пульта управления. Конечно это очень просто: нажимаешь кнопочку – машинка поехала веред, нажимаешь другую – замигал светодиод. Но ведь мы видим только конечный результат, а интересно каким же образом программисты смогли научить платформу выполнять конкретные действия при нажатии на кнопку.

17.1 Подробнее о ZigBee

Теперь давайте разберем по шагам, что происходит внутри нашего протокола, когда мы отправляем команду — «Зажечь светодиод».

Первое, например, мы нажимаем на программе пульта управления кнопку «свет».



Программа при этом распознает нажатую кнопку и формирует API фрейм из некоторого количества байтов.

Например такой:

7E 0 10 17 1 0 0 0 0 0 0 FF FF FF FE 2 44 37 5

Все байты указаны в шестнадцатеричной системе.

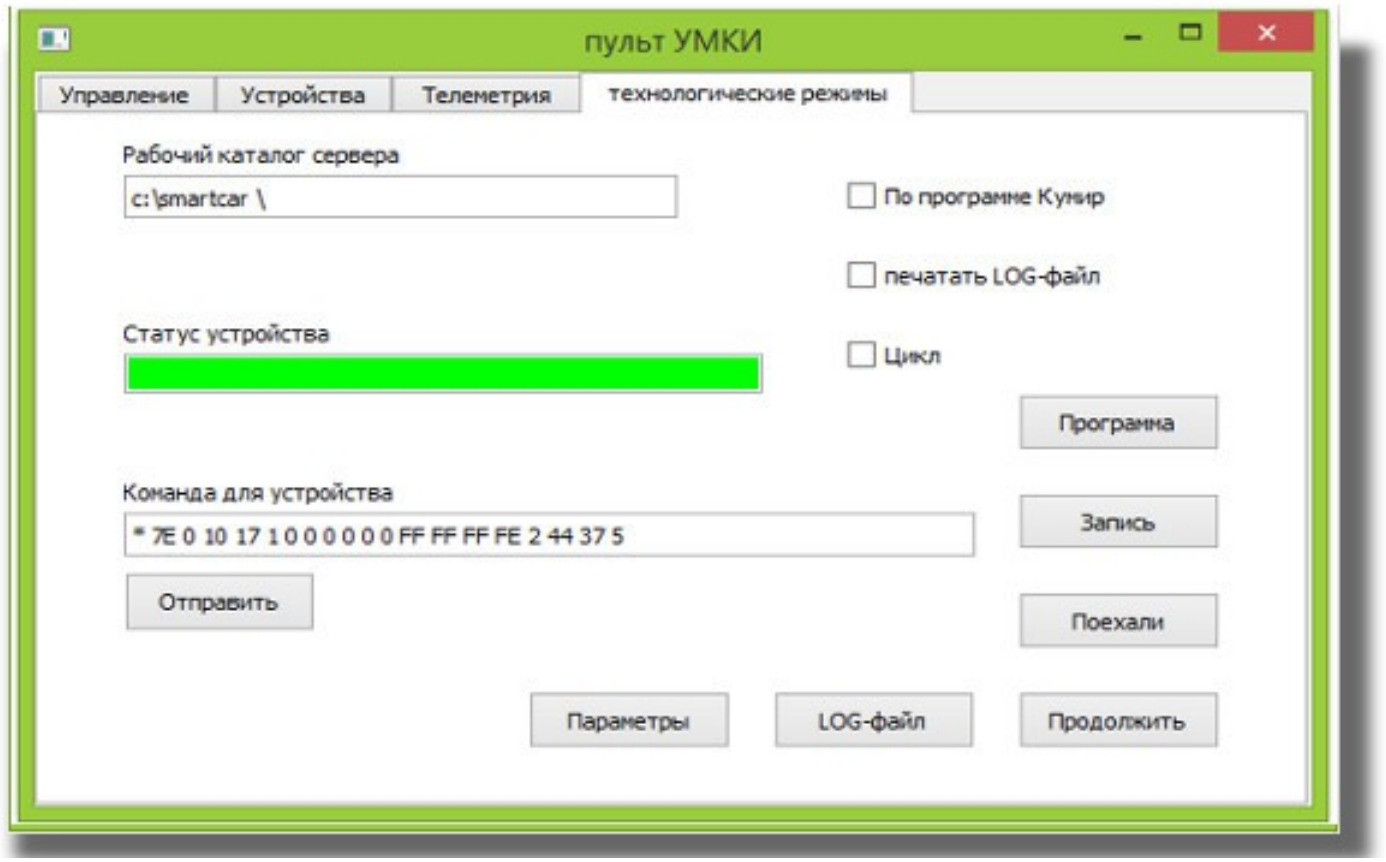
- Здесь первый байт 7E – заголовочный.
- Третий байт — длина: 10 шестнадцатеричных, или, иначе говоря, 16 байт если считать в привычной нам десятичной системе счисления.
- Следующий байт 17 — это номер API фрейма.

- Байты нули и FF указывают на то, что это широковещательный пакет, он обязателен к исполнению всем устройствам с данной сети. Если бы мы хотели адресовать его конкретному устройству SmartCar (иначе говоря моту сети), то вместо нулей мы бы указали конкретный, нужный нам МАК адрес.
- И завершают команду — три последних байта: 44, 37 — это значит свет, а 5 — значит включить.
- Каждый фрейм замыкается контрольной суммой — это тоже один байт. Здесь он не виден. Если вы хотите разобраться более подробно, какие команды бывают и что обозначает каждый байт в этом API фрейме, то вам следует обратится на сайт разработчика, и там почитать техническую документацию. Правда она на английском языке.

Наша машинка будет светить диодом либо до тех пор, пока не закончится заряд батареек, либо пока мы не пришлем устройству команду, где вместо последнего байта 5, будет указан байт 4.

Попробуйте выполнить на практике разобранные действия.

Подключите SmartCar УМКИ в режиме Super, перейдите на вкладку Технологические режимы



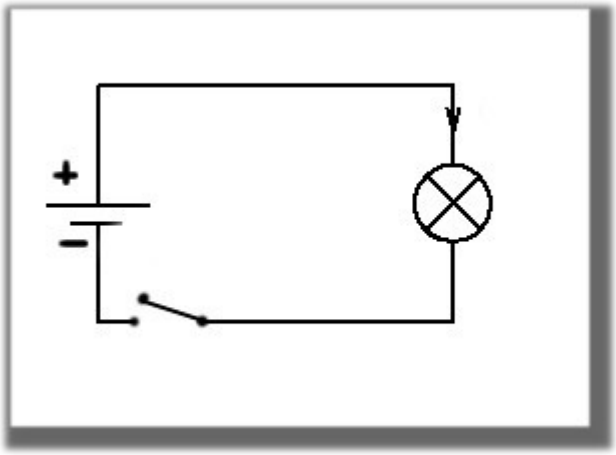
в поле Команда для устройства, введите разобранный на занятии API фрейм в виде шестнадцатеричных команд. Обратите внимание, что первым символом должна быть звездочка. Примерно так: * 7E 0 10 17 1 0 0 0 0 0 0 FF FF FF FE 2 44 37 5 – проверьте правильность выполнения команды визуально на устройстве – пронаблюдайте действительно ли идет сигнал на светодиод. Остановите выполнение данного действия, отправив, последовательность команд, выключающих светодиод. Внесите изменения в команду и пронаблюдайте результат.

§ 18 Эпизод восемнадцать. Базовые логические операции (Электронные конструкторы).

Привязка к тематическому планированию по информатике: Математические основы информатики. Логические операции - логическое отрицание, логическое умножение, логическое сложение.

Когда немецкий электротехник Георг Симон Ом положил на стол ректора Берлинского университета свою диссертацию, где впервые был сформулирован этот закон, без которого невозможен ни один электротехнический расчет, он получил весьма резкую резолюцию. В ней говорилось, что электричество не поддается никакому математическому описанию, так как электричество - это собственный гнев, собственное бушевание тела, его гневное Я, которое проявляется в каждом теле, когда его раздражают. Ректором Берлинского университета был в те годы Георг Вильгельм Фридрих Гегель.

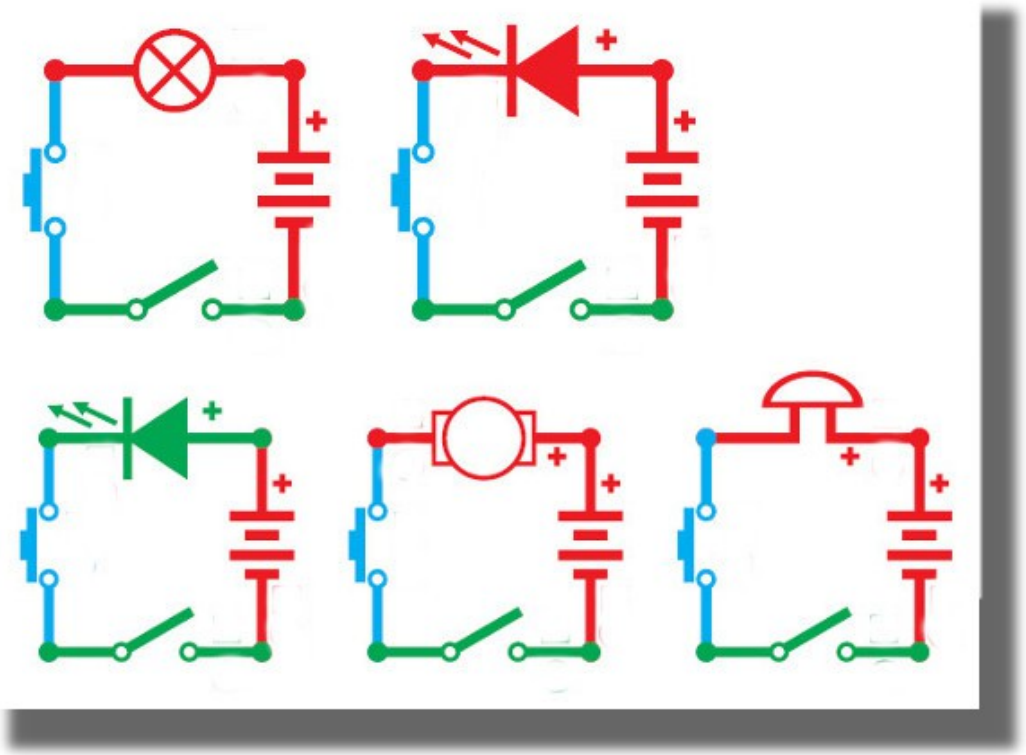
Соберем простую схему: батарейка и лампочка:



Замкнем выключатель – лампочка засветилась. По цепи протекает электрический ток, совершая работу – нагревая спираль лампочки, спираль начинает светиться. Таким образом, происходит преобразование электрической энергии в энергию тепловую и энергию света.

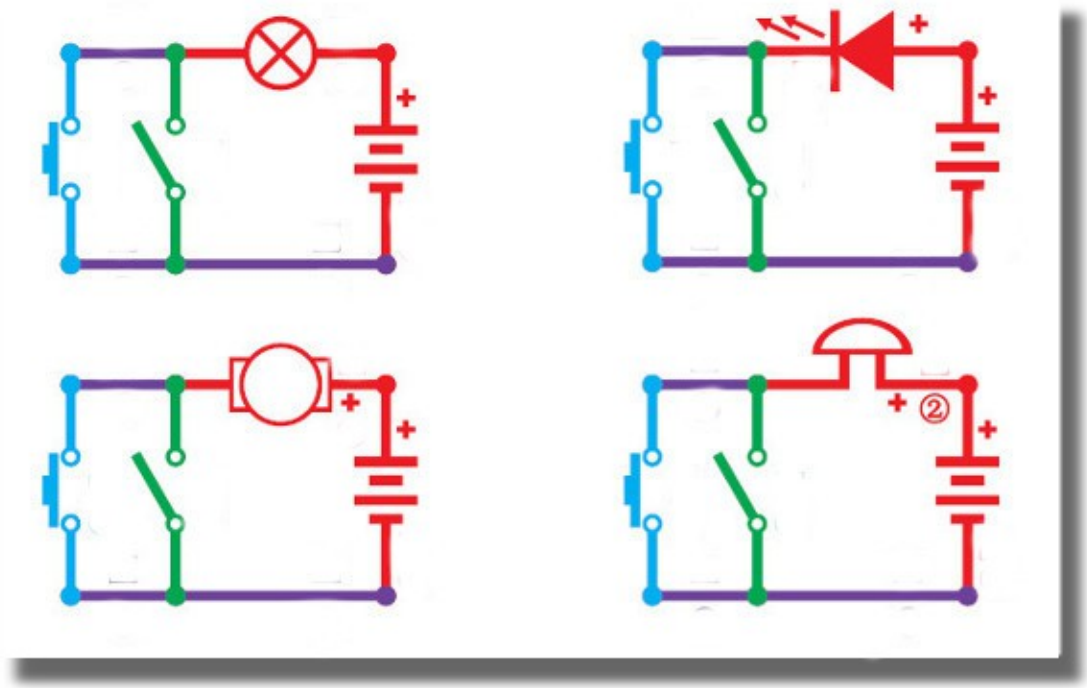
Попробуем используя имеющиеся элементы: источник питания, источник света (звука), пропеллер, кнопку и выключатель, проиллюстрировать работу логических элементов.

Логическое И - конъюнкция.



Лампа загорится, прозвучит звуковой сигнал, закрутится пропеллер, когда одновременно будет нажата кнопка **И** переключатель поставлен в режим ON.

Логическое ИЛИ — дизъюнкция.

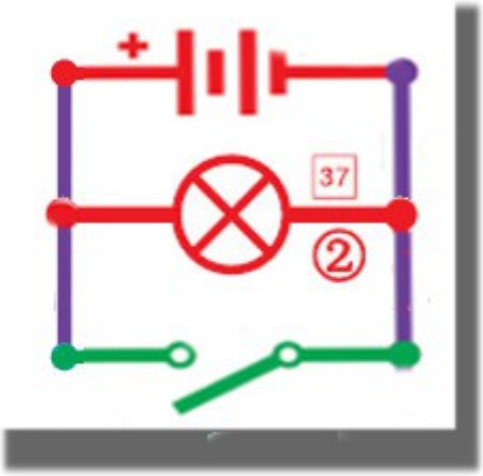


Лампа загорится, прозвучит звуковой сигнал или закрутится пропеллер, когда будет нажата кнопка **ИЛИ** замкнут переключатель.

Лампа загорится, прозвучит звуковой сигнал или закрутится пропеллер, когда будет нажата кнопка **ИЛИ** замкнут переключатель.

Создать модель операции логического отрицания можно по следующей схеме.

Логическое «НЕ» - инверсия.



Когда выключатель находится в положении «ON» – лампа не горит. Когда выключатель находится в положении «OFF» – лампа будет гореть.

§ 19 Эпизод девятнадцать. Функциональное разнообразие роботов. 8 класс.

Привязка к тематическому планированию по информатике: Основы алгоритмизации. Алгоритмы и исполнители.

Какие бывают роботы

Давайте разберем, какие вообще бывают роботы.

Рассмотрим две большие группы роботов: стационарные роботы на фундаменте и мобильные роботы, выполняющие работы всюду куда смогут добраться.

В какой же области деятельности могут использоваться роботы?

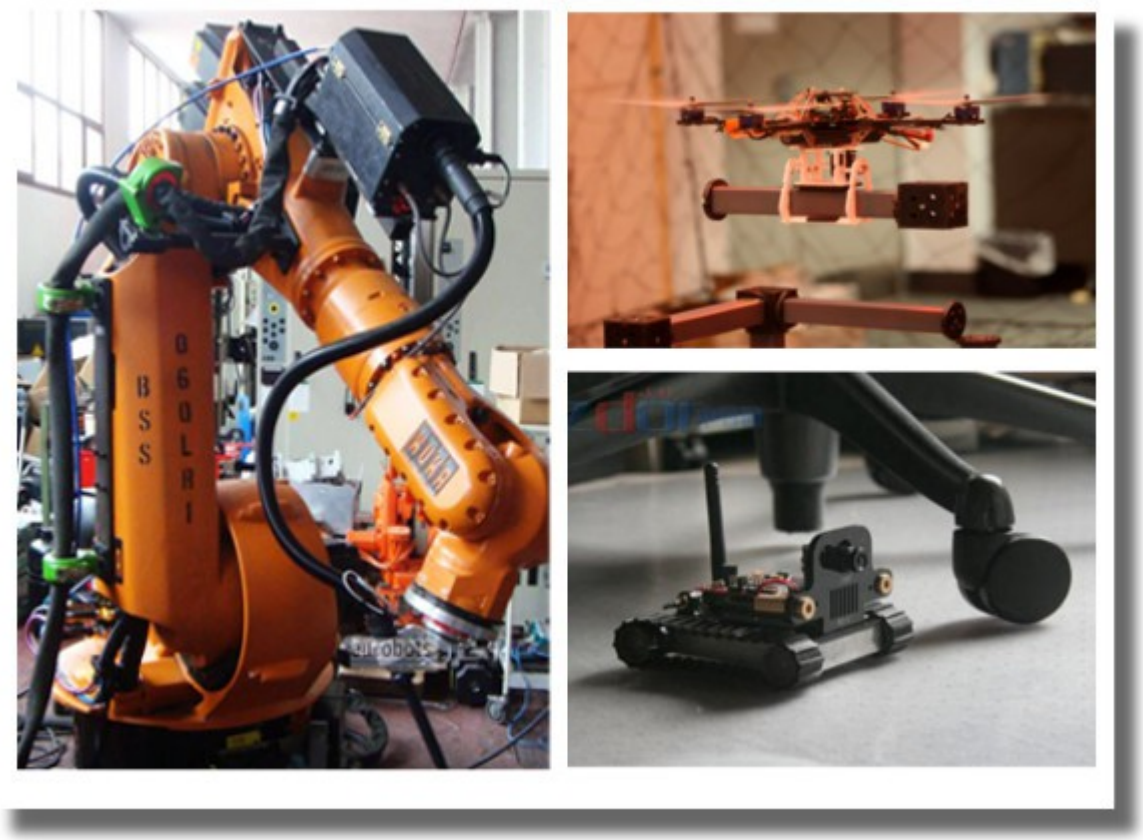
19.1 Стационарные и мобильные

Давайте разберем, какие вообще бывают роботы.

Наверное, имеет смысл разделить их на две большие группы.

- Это стационарные роботы на фундаменте — такие обычно используются в промышленности, на линиях сборки и сварки кузовных деталей автомобилей.
- И другая разновидность — это мобильные роботы, которые могут перемещаться как робот-пылесос всюду, и выполнять работу везде, куда смогут добраться.

В свою очередь, мобильные роботы можно разделить по способу передвижения: передвигающиеся на колесах или гусеницах, шагающие роботы, плавающие и летающие роботы.



19.2 Промышленные роботы

Это как правило мощные роботы-манипуляторы, установленные на неподвижный фундамент и способные выполнять действия в радиусе, куда достанет рука. Их на конвейере ставят в ряд сразу несколько штук, и каждый выполняет несколько операций: таких, как сварка, резка, диагностика события.

Для разработки таких роботов нужны знания в области механики — в институте такие дисциплины называются сопромат и теория машин и механизмов, а так же, познания в области силовой электрики, так как захваты деталей у них срабатывают от пневматического или гидравлического привода.

19.3 Медицинские роботы

Уже с 2001 первого года медицинский робот для хирургических операций используется в реальных условиях.

Такие роботы выполняют свое предназначение при помощи манипуляторов через систему дистанционного слежения, и позволяют не касаться тканей больного человека лишний раз, во время операции, чтобы не травмировать.

Так же, к разряду медицинских роботов можно отнести и экзоскелеты. Они помогают людям с проблемами в опорно-двигательной системе передвигаться по полу, шагая ногами.

Или могут быть полезны как протезы на руках — брать предметы, управлять такими электронными мышцами сейчас уже возможно снимая датчиками сигналы с мышц человеческого тела. Наука, занимающаяся подобными проблемами, называется бионика.

19.4 Подводные роботы

Используются в морском деле для проведения аварийно-спасательных работ, или различных исследований на дне моря.

Проблема в том, что невозможно управлять таким роботом дистанционно, по радио каналу. Как известно радио волны очень плохо распространяются в водной среде, поэтому таких автономных роботов надо очень тщательно программировать, чтоб он смог выполнить поставленную задачу и вернуться на базу самостоятельно.

19.5 Сельскохозяйственные роботы

Такие роботы могут быть очень актуальными в наших российских условиях. Поскольку в нашей стране много земли и мало людей на один квадратный километр, то для выращивания на больших площадях разных вкусных и полезных овощей, без автоматических помощников никак не обойтись.

Преимущество робота перед человеком в том, что он не устает при обработке большого количества насаждений и может работать хоть круглые сутки — пока не кончится заряд в батареях. Такие роботы могут с большой аккуратностью распылять и удобрять почву прямо под корнями растений, экономя при этом ресурсы и оберегая сами растения, не заставляя людей дышать вредными испарениями.

19.6 Военные роботы

К глубокому сожалению, это сейчас самое финансируемое направление в развитии робототехники, но, учитывая сильное напряжение в мировой обстановке, наверное, не стоит нам уделять таким роботам много внимания.

19.7 Строительные роботы

Для строительства жилых домов уже вовсю в мире, начинают использовать метод 3D- печати. Однако, если изготовить на 3D-принтере пластмассовую поделку не составит особого труда, то для строительства больших объектов — таких как дом, надо решить массу технических, технологических и инженерных проблем.

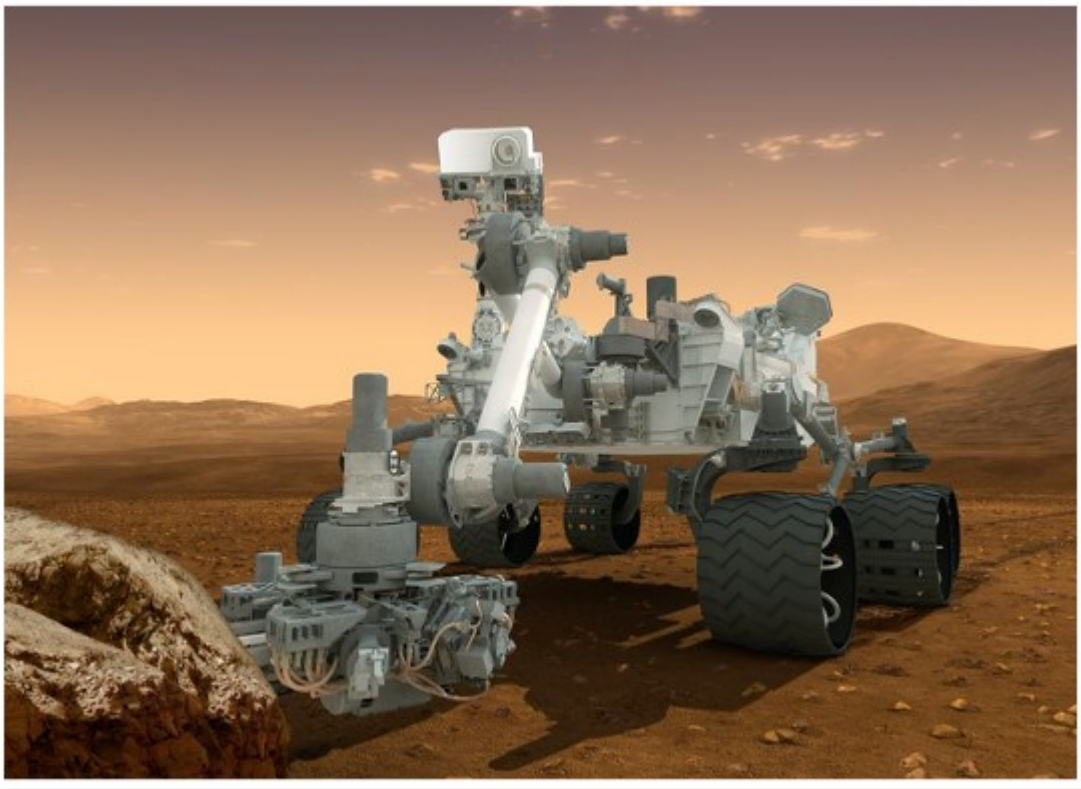
Одна из самых актуальных — это подбор строительного материала: он должен быть пластичным, чтобы подаваться в экструдер, и быстро сохнущим, чтобы не расползался, когда на первый слой будет наносится второй. Консистенция такого материала напоминает зубную пасту.
К тому же, дом из такого высохшего материала не должен разрушаться от внешних воздействий — ветра, солнца, дождя, снега, хорошо защищать от жары и холода.

Сейчас такие материалы придумываются и защищаются патентами, в скором времени мы, возможно, увидим бум строительства домов — особенно малоэтажной застройки по технологии 3D-прототипирования.

19.8 Космические роботы

Первым космическим роботом можно считать Луноход, отправленный на Луну, еще во времена СССР в далеком 1970 году.

Сейчас, широко известно работа на Марсе робота Кюриосити — что означает любознательный. При отправке и высадке его на Марс пришлось решать очень много технических и инженерных проблем с решением которых справились блестяще. Была разработана специальная платформа на реактивных двигателях, которая подлетев к поверхности Марса, как бы «зависла» и марсоход очень бережно на лебедке, сам с нее спустился на поверхность.



Также на комету Чурюмова-Герасименко высадился межпланетный агрегат — тоже полноценный робот, который проводит исследования поверхности в точки высадки, берет пробы грунта и анализирует их на месте, передавая на землю в цифровом виде данные о составе почвы и поверхности на которой он работает.

19.9 Сервисные роботы

Сервисные роботы помогают нам в быту, в окружающей нас действительности.

К сервисным роботам скорее всего можно отнести робот-пылесос, который помогает чистить пол в доме в отсутствие хозяев. Так же бывают сервисные роботы телеприсутствия, когда на выставках или в местах большого скопления людей они помогают сориентироваться, подсказывают куда сейчас стоит сходить и отвечают на вопросы посетителей.

В школе такие роботы телеприсутствия могут помочь ребятам, которые болеют и не могут придти в школу на занятия, почувствовать себя в коллективе, побродить на переменке по коридорам, пообщаться с друзьями, получить задания на дом.

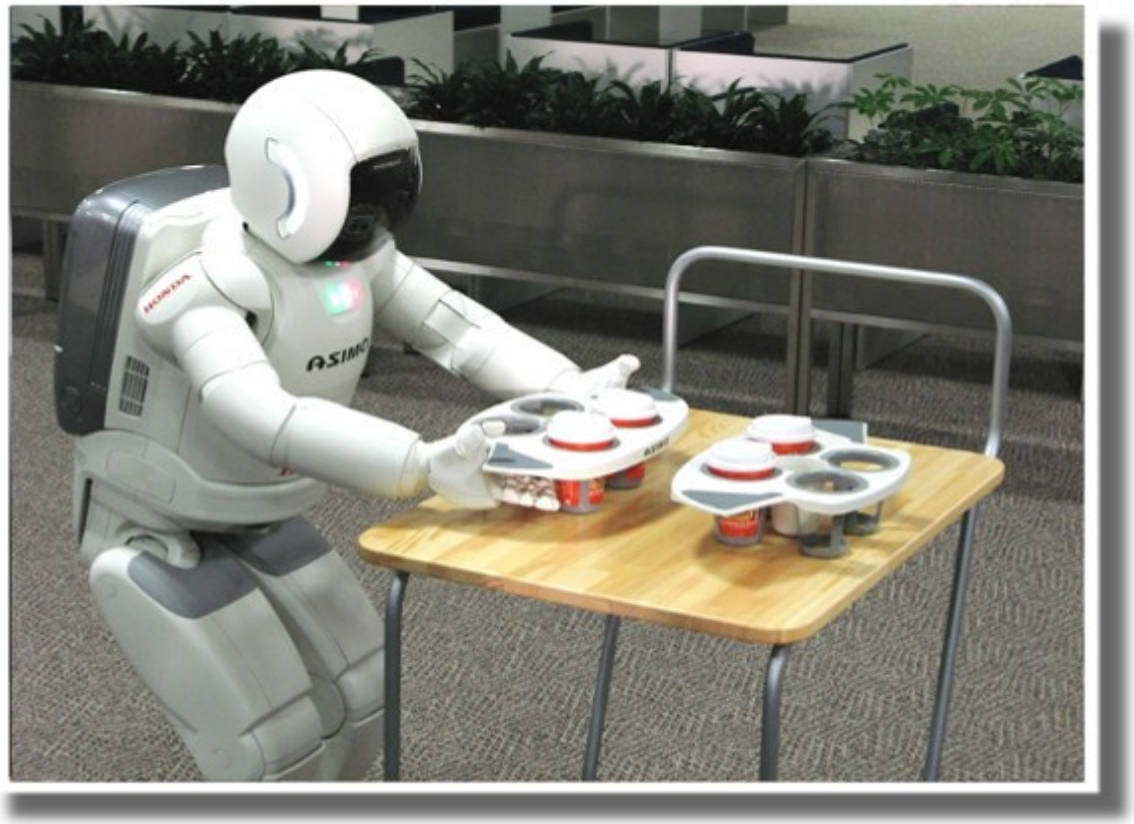
Бывают роботы музыканты. Они не играют на своих музыкальных инструментах под фонограмму, как это зачастую делают некоторые эстрадные выступающие, а реально извлекают звуки мелодий ударяя палочками по барабанам, дергая струны гитар и нажимая клавиши своих инструментов. Такое шоу роботов ездит по миру, бывает в России, и всегда пользуется огромной популярностью.

Еще один проект, который можно отнести к сервисным роботам — это обучающийся робот-колобок. Он может быть полезен не только как забавная игрушка в доме для ребенка, но и как способ совместного обучения для познания окружающей среды.

19.10 Шагающие роботы

По способу перемещения сервисных роботов можно разделить на колесных или гусенечных, и на шагающих роботов.

Самым известным пожалуй из шагающих роботов являются японский робот Асимо, такие роботы уже используются в некоторых кафе как подносчики заказанной еды.



И робот от компании БостонДинамик - шагающий мул. Эта компания продала свой разработку Гуглу за несколько миллиардов долларов. И теперь такой шагающий четырехногий друг стал используется для переноса тяжестей через пересеченную местность. Такой переносчик не устает даже после перехода в 70 км, достаточно только подзарядить батарейки.

19.11 Круиз-контроль

Так же к сервисным роботам можно отнести и систему круиз-контроля на автомобилях. Причем, если раньше такие системы только поддерживали определенную скорость при движении по трассе, то сейчас, появились значительно более интеллектуальные системы круиз-контроля, которые определяют еще и расстояние до идущего впереди автомобиля и позволяют двигаться со скоростью задаваемой лидером в колонне. И следующим этапом развития такой технологии, закономерно становится способ управления группой транспортных средств в езде за лидером. Таким образом, колонну автомобилей ведет водитель сидящий в первой машине, а водители в ведомых автомобилях могут заниматься своими делами, не следить внимательно за дорожной обстановкой: пить кофе, проверять почту, слушать музыку или читать книгу.

Такая система уже создана европейским сообществом, испытаны на треках компании Вольво и сейчас уже запускается в коммерческую эксплуатацию. Называется эта система по имени старинного философа — Сартр.

Конечно для разработки таких систем нужны знания по механике, по электрике, по гидравлике, по электронике, по программированию, по протоколам связи. Невозможно все это досконально знать одному человеку. Над решением задач такой сложности трудятся большие коллективы.

§ 20 Эпизод двадцать. Среда управления учебным исполнителем Робот – Кумир. 8 класс.

Привязка к тематическому планированию по информатике: Основы алгоритмизации. Понятие алгоритма как формального описания последовательности действий исполнителя при заданных начальных данных.

КуМир (Комплект Учебных МИРов) – свободно распространяемая кроссплатформенная русскоязычная система программирования, предназначенная для начального обучения основам алгоритмизации.

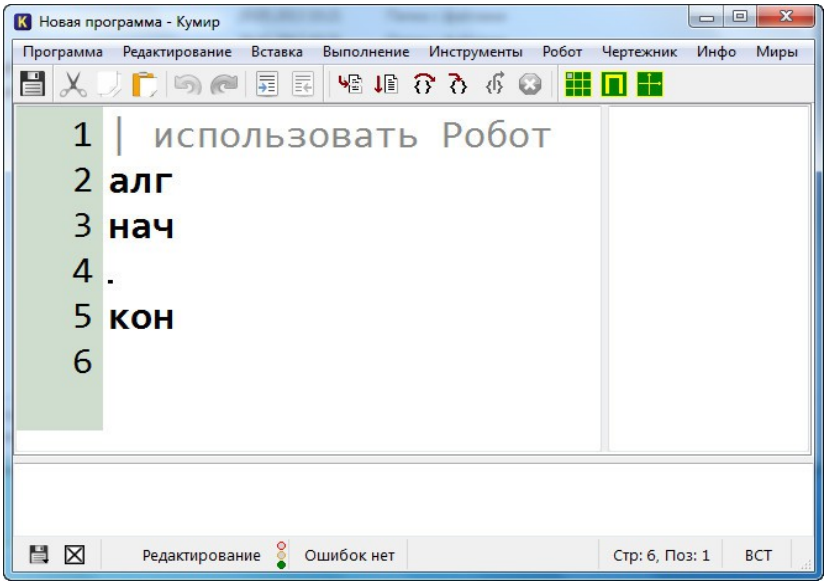
Система программирования КуМир основана на методике, разработанной под руководством академика Ершова А.П. В системе КуМир используется школьный алгоритмический язык к с русской лексикой и встроенными командами управления программными исполнителями (Робот, Чертёжник, Кузнечик, Черепаха).

Скачать программу можно на сайте разработчиков http://lpm.org.ru/kumir2/ , на официальном сайте проекта http://www.niisi.ru/kumir/, на сайте разработчиков: http://www.umkikit.ru/ .

Программа есть на DVD диске с программным обеспечение и материалами, который входит в комплект Цифровой лаборатории УМКИ .

Обучение работе в среде Кумир, может занимать отдельный курс, но для начального знакомства – освоить среду, чтобы заставить слушаться машинку-робота – достаточно нескольких занятий.

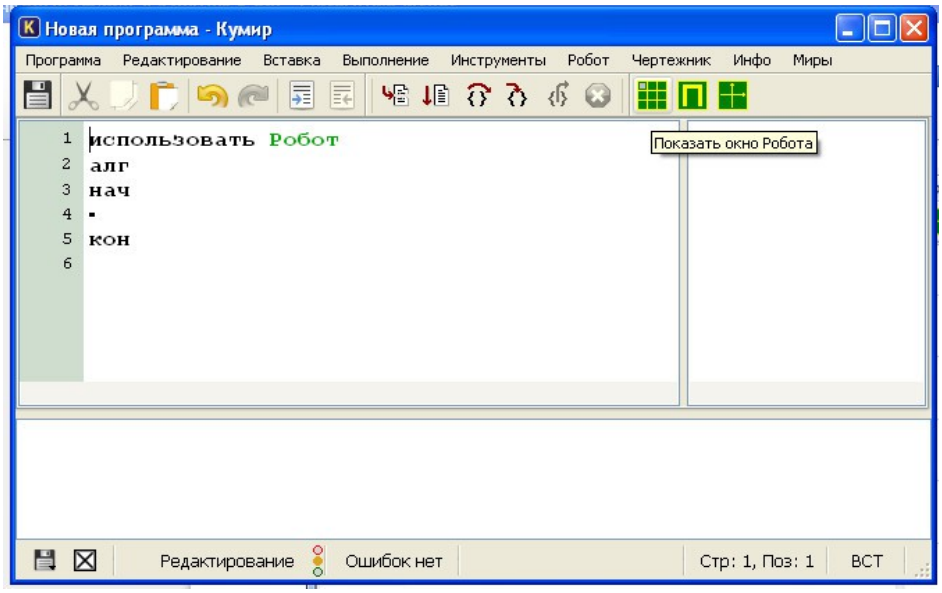
После загрузки среды Кумир на экране можно увидеть следующее окно:



Окно разбито на две основные области: рабочую область (вверху) и область ввода-вывода (внизу). В рабочей области располагается программа, с которой работает система Кумир.

Эта область делится на две части: область программы (слева) и область построчных сообщений (справа), в которую при подготовке программы выводятся сообщения об ошибках, найденных в каждой строке, а при выполнении — сведения о значениях переменных, присваиваемых в строке.

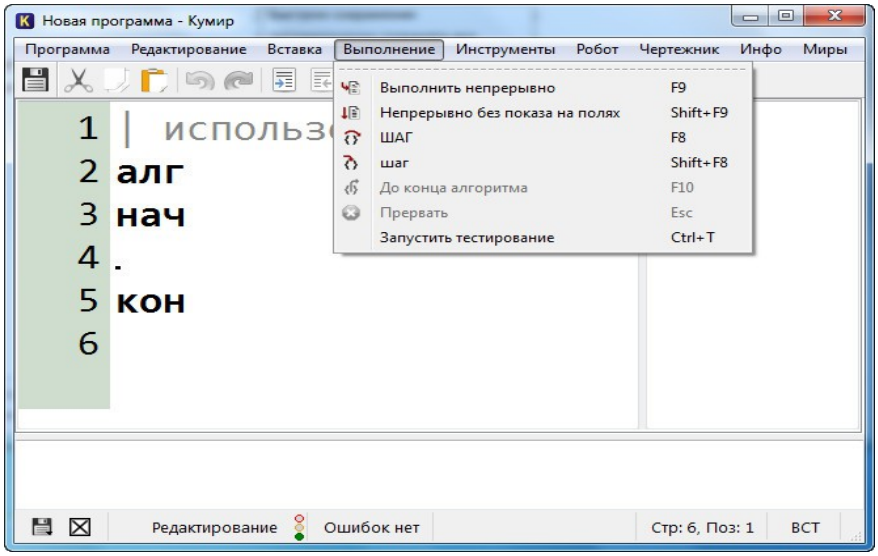
Для работы программы исполнителя Робот необходимо снять знак комментария перед командой выбора исполнителя



Исполнитель Робот перемещается в некоторой обстановке – прямоугольном поле, разбитом на клетки, между которыми могут располагаться стены. Робот может передвигаться по полю и закрашивать клетки.

Основные команды перемещения которые понимает Робот: вверх, вниз, влево, вправо, при выполнении каждой из которых, робот переместится на одну клетку соответственно: вверх, вниз, влево, вправо. Также робот умеет исполнять команду закрасить. По этой команде закрашивается клетка, в которой робот находится в настоящий момент.

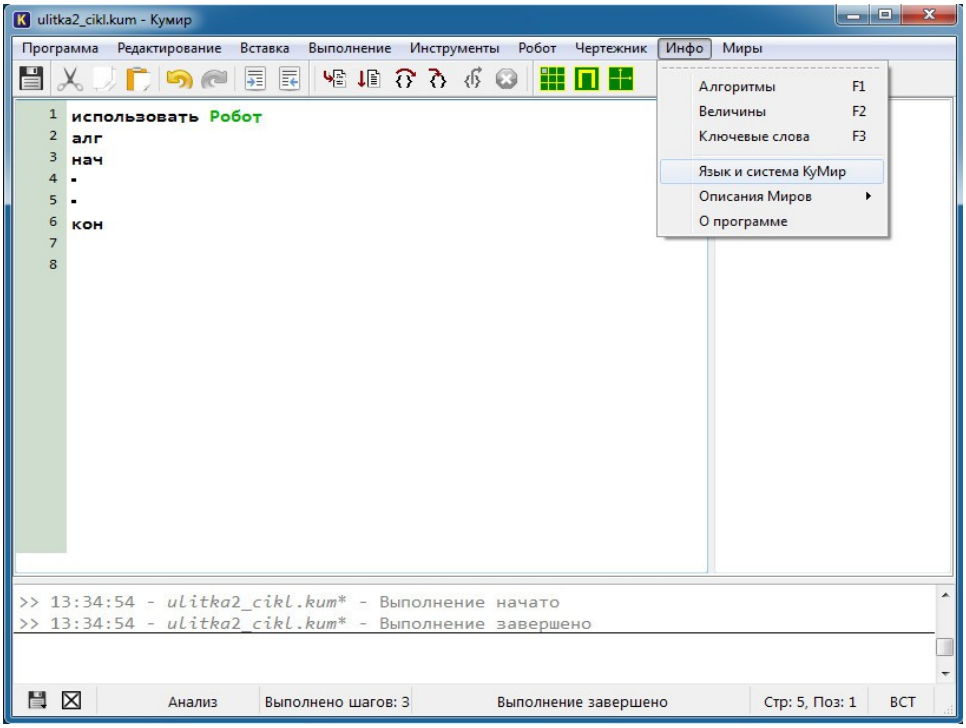
С помощью списка команд меню Выполнение, можно просматривать работу готовых алгоритмов в различных режимах.



«Выполнить непрерывно» – начинает непрерывное выполнение программы без остановок между шагами.

«ШАГ» – Выполняет один ШАГ программы и переходит в режим ПАУЗА. Для выполнения очередной команды, необходимо дать следующую команду ШАГ.

О работе в программе Кумир можно прочитать в справочной системе КуМир. На вкладке «Инфо» выберите «Язык и система КуМир»



В Интернете достаточно много информации о начале работы с данной средой, например

- на сайте разработчиков http://www.niisi.ru/kumir/meth.htm
- Электронный учебник Учимся программировать с КУМИР http://informatik.16mb.com/index.html
- Курс алгоритмизации с использованием исполнителей системы Кумир и автоматического тестирования Дениса Кириенко http://server.179.ru/wiki/?page=DenisKirienko/Kumir
- Курс по исполнителю Робот, размещенные на сайте К.Ю. Полякова «КуМир Что это такое?» http://kpolyakov.spb.ru/school/kumir.htm ,
- Презентация «Исполнитель Робот» http://kpolyakov.spb.ru/school/ppt.htm#robokum .
- Практикум для использования в среде КуМир http://kpolyakov.spb.ru/download/robokum.zip, который надо подключить к среде в разделе Инструменты

Параллельно с работой по управлению виртуальным роботом начинается работа по созданию программ для SmartCar в среде КуМир.

§ 21 Эпизод двадцать один. Программирование условных алгоритмов для платформы SmartCar. 8 класс.

Привязка к тематическому планированию по информатике: Основы алгоритмизации. Алгоритмические конструкции, связанные с проверкой условий

Учебный исполнитель Робот

Использование датчиков касания для прохождения лабиринта.

§ 22 Эпизод двадцать два. Калибровка энкодеров. 9 класс.

Привязка к тематическому планированию по информатике: Моделирование и формализация. Понятия натурной и информационной моделей.

Понятия натурной и информационной моделей на примере виртуального исполнителя Робот и платформы SmartCar.

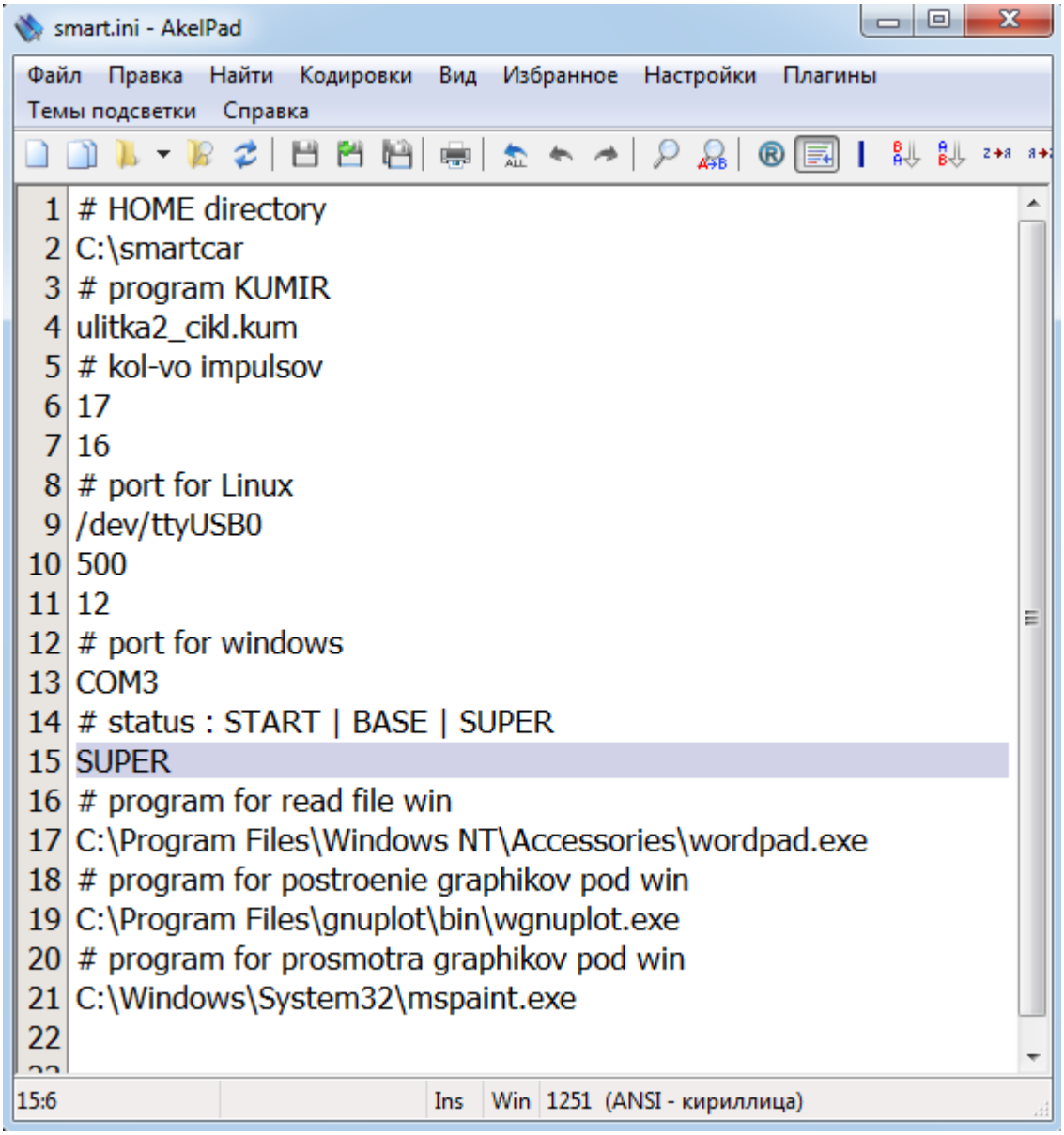
Как уже стало понятно ранее, при занятиях с платформами SmartCar – они не вполне корректно повторяют работу виртуального Робота программы Кумир. Если Робот из программы Кумир четко поворачивается на 90 градусов при повороте вправо или влево, аналогичный поворот механических платформ может достаточно отличаться от прямого угла. Это происходит потому, что длительность вращения каждого колеса определяется количеством импульсов подаваемых на это самое колесо, соответственно длина дискретного шага определяется этим количеством импульсов, и естественно поворот — правое колесо вращается, а левое нет.

А так как каждое колесо управляется своим собственным мотором, то тут и могут возникнуть проблемы – два одинаковых мотора при подаче на них одинакового количества импульсов могут поворачивать колеса таким образом, что они проходят чуть-чуть разное расстояние.

Для подсчета количества импульсов служит специальное устройство – энкодер.

И, чтобы повороты у все платформ были идентичны может потребоваться калибровка энкодеров – изменение количества импульсов, поступающих на мотор при однократной подаче команды движения.

Управление количеством импульсов в программе SmartCar возможно на уровне Super. Для запуска программы третьего уровня - SUPER потребуется изменить последнюю строку файла smart.ini. В строке комментариев изменяем BASE на SUPER → сохраняем изменения → закрываем окно.

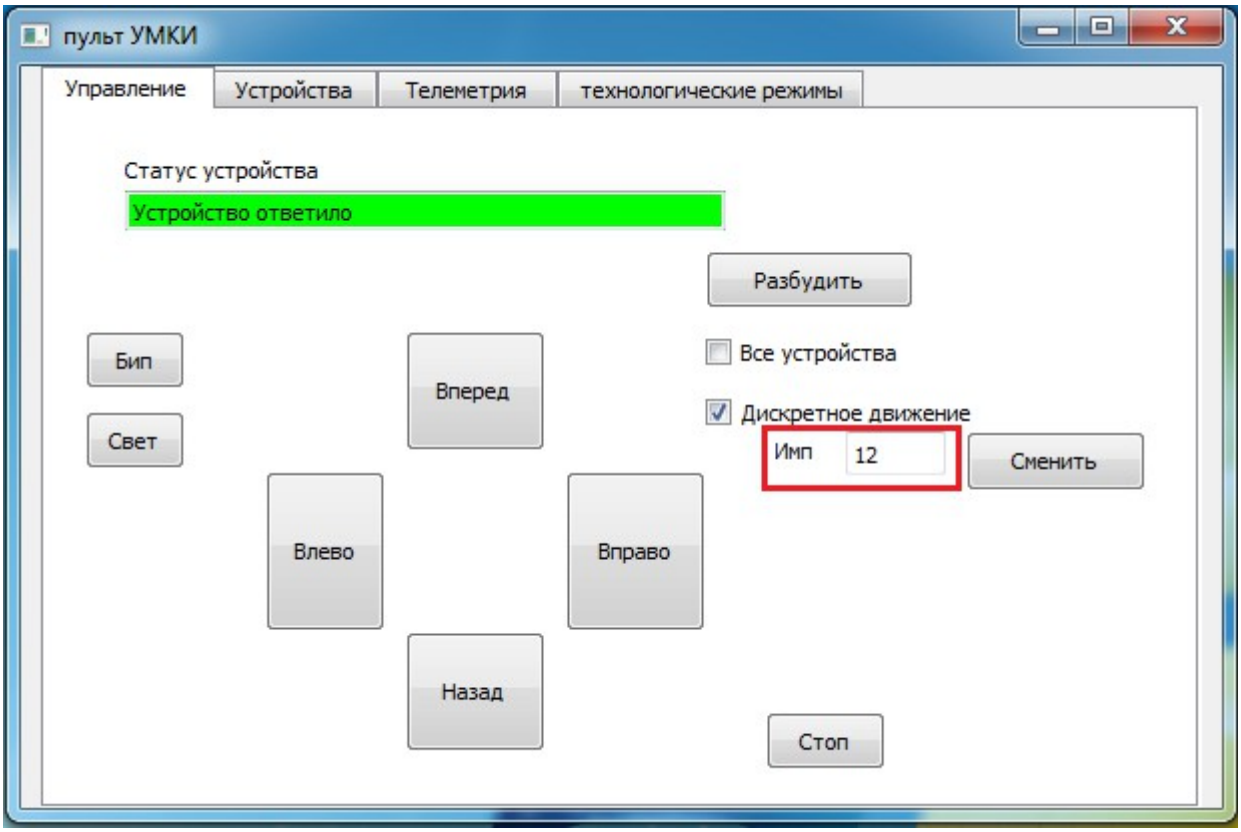


После запуска программы SmartCar, появляется рабочее окно с вкладками: «Управление», «Устройства», «Телеметрия», «Технологические режимы» – интерфейс управления изменился, по сравнению с предыдущими вариантами, показывая дополнительные возможности.

В нашем случае нас будет интересовать значение параметра «Импульсы»

В окне «Импульсы» можно задать любое значение. Именно этот прамет необходимо подкорректировать для получения точного поворота на 90 градусов или для увеличения шага движения.

Импульс – угол поворота колеса машинки. Если вы хотите, чтобы машинка ехала дольше в одном направлении, установите большее число импульсов, например, 50.


пульт УМКИ
Управление
Устройства
Телеметрия
технологические режимы
Статус устройства
Устройство ответило
Разбудить
Бип
Свет
Вперед
Влево
Вправо
Назад
Все устройства
Дискретное движение
Имп
12
Сменить
Стоп

§ 23 Эпизод двадцать три. Подробнее о коптерах. А вы любите летать? 9 класс

Привязка к тематическому планированию по информатике: Моделирование и формализация. Использование моделей в практической деятельности.

23.1 Беспилотные летательные аппараты

Прилетит вдруг волшебник
В голубом вертолете....

Благодаря развитию техники и технологии, появлению новых материалов, становится возможным разрабатывать доселе невиданные аппараты — например квадрокоптеры и другие беспилотные летательные аппараты.

Коптером мы будем называть управляемый летающий при помощи вращающихся пропеллеров аппарат. Самый распространенный — с четырмя винтами — это квадрокоптер. Бывает с шестью винтами — тогда называют гексакоптер, очень редкая модель — с тремя винтами. Классический вертолет имеет как правило один большой несущий винт и на хвосте маленький рулевой винт, для того чтобы весь корпус с пилотами и пассажирами не вращался сам по себе в противоположную сторону вращения несущего винта.

На некоторых моделях вертолетов ставят два винта на одной оси. И тогда они вращаются в противоположные стороны. При этом рулевой винт уже не так актуален.

Для того чтобы квадрокоптер полетел — необходимо изготовить раму, причем она должна быть легкой и прочной одновременно. Такой чтобы упав на землю с высоты не

треснула и не разлетелась на кусочки. Для этого используют композитный материал с такими свойствами – углеплатстик. Он известен еще с 60-х годов двадцатого века, но стал доступным по цене только недавно. Из углепластика же делают и несущие нагрузку весом самого квадрокоптера — винты. Моторы используются безколлекторные на неодимовых магнитах. Принципиального нового открытия тут нет, но это все стало доступным в коммерческом использовании не более чем 3-4 года назад.

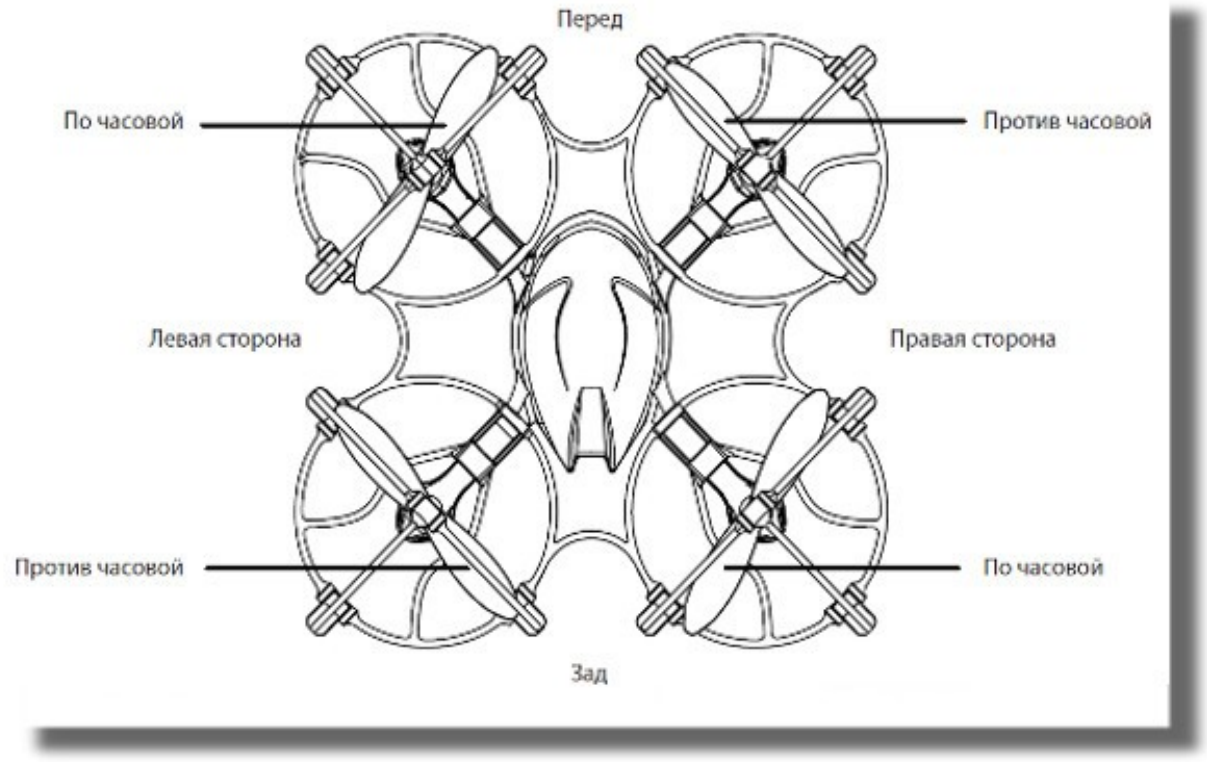
Ну и конечно без микроэлектроники невозможно разработать алгоритмы для самобалансировки коптера в пространстве.

Для управления такими квадрокоптерами используются различные протоколы связи.

Самый распространенный — это протокол WiFi, реже Bluetooth, но самым перспективным пожалуй является протокол ZigBee — в переводе означает пчела летающая зигзагом, но подробнее об этом дальше...

23.2 Управляем коптером

В нашем же случае, мы рассмотрим квадрокоптер, у которого четыре винта разнесены от центра на равные расстояния. При этом, для того, чтобы коптер не вращался абы как, два винта по диагонали должны крутится по часовой стрелке, а два – им навстречу, против часовой стрелки. Если обозначить винты цифрами по кругу 1,2,3,4; то в одну сторону крутятся 1 и 3-й , а 2 и 4-й в другую.



Как происходит управление движением коптера?

Для того чтобы коптер при старте с земли начал подниматься вверх — винты раскручиваются сначала медленно, потом все быстрее. Это называется — дать тягу или газ. Кроме вертикального подъема у коптеров и других летающих машин обязательно должны учитываться параметры — крен, тангаж, рысканье.

23.3 Мотор – первое дело

Чтобы винт начал вращаться нам необходим мотор, который жестко закреплен на корпусе коптера. Тяга у винта появляется в тот момент когда мотор набирает обороты. В принципе, винт может быть обычным плоским листом пластмассы, но чтобы тяга получалась максимальной, винт изготавливают после специальных аэрогидравлических расчетов и экспериментов. С начала 20-ого века разработана целая наука для расчета винтов, для разных высот полета и в разных условиях. В России этому обучают в академии имени Жуковского.

Чтобы коптер поднимался плавно вверх — скорость вращения винтов должна возрастать от нуля до максимума который может дать ваш мотор. Такие регуляторы скорости называют в английской транскрипции ESC. Бортовой микроконтроллер коптера, выдает нужную скорость вращения в зависимости от задания. Каждый ESC регулятор управляет одним мотором. Значит в нашем случае, на борту коптера имеется четыре ESC регулятора. Команды им поступают от полетного контроллера, в центральный микрочип которого прошита созданная разработчиками программа которая и управляет скоростью вращения всех моторов.

23.4 Плавно-плавно полетим

Все моторы, изготовленные на заводе пусть даже по одной номенклатуре имеют некоторые расхождения. У них могут быть чуть-чуть различные микрозазоры, например, в шестеренке редуктора. Внешне это никак не обнаружишь, но при работе когда полетный контроллер дает всем команду увеличить скорость вращения на 10%, в реальности один мотор может увеличить скорость на 11%, а другой только на 9%.

Что при этом произойдет? Разбаланс. И коптер вместо плавного подъема начнет рыскать и сваливаться в крен. Чтобы избежать таких неприятных моментов управления, на коптер устанавливают датчик угла наклона, который называется — Гироскоп. Еще может быть установлен — Акселерометр, измеритель скорости движения по трем осям.

Информация с этих датчиков заводится в полетный контроллер и встроенная программа оценивает изменение скорости и углы наклона. И в том случае, когда вместо плавного вертикального подъема по оси Z наш коптер начинает отклонятся от нуля по оси X или Y, тогда полетный контроллер понижает немного обороты на винте 1 и 3, и увеличивает на винте 4 например. Эту задачу выравнивания программа полетного контроллера решает постоянно — много раз за секунду.

Для движения вперед моторы 1 и 2 чуть снижают обороты, а 3 и 4 добавляют. Коптер при этом слегка наклоняется и у него кроме вертикальной появляется горизонтальная тяга и он летит вперед.

Скорость вращения винтов регулируется из программы при помощи управления широтно-импульсной модуляцией (ШИМ).

23.5 Управляем коптером

Программа движения для коптера задается либо дистанционно: когда мы двигаем рычаги пульта управления, или в специальном приложении мобильного телефона нажимаем нужную кнопку или с компьютера отправляем летающему устройсту нужную команду движения; либо такая программа может быть в него зашита заранее. И тогда, коптер следуя шагам программы перемещается по заданным точкам ориентируясь на спутники GPS или ГЛОНАС. Понятно, что при этом он должен быть оснащен нужным модулем связи с этими системами.

23.6 Программы для коптера

Если вы хотите научится разрабатывать собственные программы для управления коптерм, то вам нужно будет изучить алгоритмы балансировки и математику стабилизации, а также понять механизм работы пропорциональных регуляторов. В настоящее время достаточно много уже разработанных программ, которые выложены в открытый доступ. Вполне возможно вам будет проще начать изучение увлекательной технологии управления летающими машинами с уже апробированных программ (Ardupilot, MegapirateNG, MiltiWii, AeroQuad и т.п.).

Эти программы сейчас доступны для скачивания из общих репозиториев. Каждый, кто немного разбирается в языке Си — может ее скачать, модернизировать согласно своим требованиям к этому алгоритму и прошить на центральный процессор.

Наверняка со временем вы сможете их улучшить.

§ 24 Эпизод двадцать четыре. Датчики – органы чувств. 9 класс

Привязка к тематическому планированию по информатике: Алгоритмизация и программирование. Управление, управляющая и управляемая системы, прямая и обратная связь.

24.1 Глаза и уши

Эти глаза напротив –
Ярче и все теплей ....

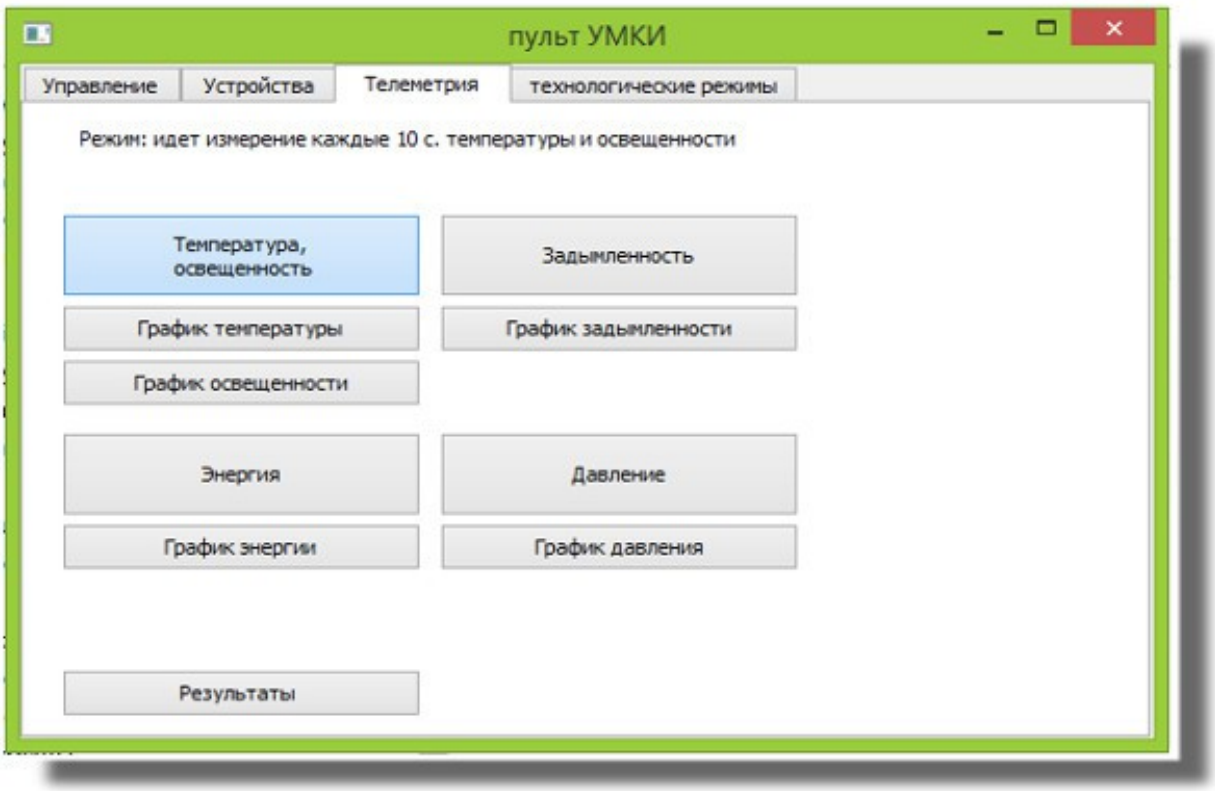
Всякий раз когда мы хотим понять что происходит вокруг нас — нам приходится обрабатывать информацию которую наш мозг получает от внешних раздражителей: от кожи, от глаз, ушей, носа и т. д. Так же и для робота. Чтобы центральный контроллер мог ориентироваться относительно внешнего мира — он регулярно получает информацию от установленных на борту датчиков. Получается, что датчики — это глаза и уши робота. На роботах УМКИ — модель CAR3 — при полной комплектации установлены следующие датчики:

- Датчик освещенности
- Датчик температуры
- Датчик атмосферного давления
- Датчик влажности
- Датчик задымленности
- Датчик касания (концевой выключатель, микрик)
- Датчик угла поворота колеса (энкодер) один на левом, другой на правом колесе.

По встроенной программе контроллер опрашивает состояние датчика в тот момент времени, когда приходит команда запроса, получает результат, обрабатывает его и отправляет информацию с показанием датчика обратно в сеть на координатор.

24.2 Производим замеры

Давайте разберем по шагам ситуацию, что происходит, когда мы нажимаем кнопку замерить Температуру.



1. При нажатии кнопки Температура или Освещенность на закладке Телеметрия, в программе smartcar, в порт устройства /dev/ttyUSB0 формируется и отправляется пакет данных – фрейм из набора шестнадцатеричных байтов.

Он характеризуется заголовочным байтом 0x7E, номером фрейма 0x17, мак-адресом устройства назначения, самой командой и контрольной суммой. (привести пример LOG)

2. После того, как наш робот получит адресованную ему команду, и произведет замер температуры, контроллер робота по заложенной в него программе сформирует пакет ответа и отправит его в сеть, на координатор.

3. Координатор сети, вставленный в USB порт ПК на котором запущена программа smartcar, примет фрейм ответа и выполнит следующие действия:

- Совершит разбор принятого фрейма.
- Вычленит из него нужный байт со значением температуры,
- Пересчитает по формуле, которую создал разработчик датчика температуры, значение из промежуточного — шестнадцатеричного в конечное — десятичное.
- Отобразит вычисленное значение на экране в указанном поле

24.3 Управляем командами – свет

Давайте попробуем отправить команду для выполнения некоторого действия — например включим светодиод платформы УМКИ — из программы smartcar. Для этого нам нужно получить текст самого фрейма. Его мы легко получить из последней строчки файла

LOG.txt . Чтобы прочитать этот файл, нажмем кнопку Технологические режимы>>LOG-файл.

Выберем нужную строчку и вставим ее в поле «технологические режимы>>Команда для устройства»

(Обратите внимание: перед самым первым байтом нужно поставить символ * и пробел.

Нажимаем кнопку Отправить.

Если все сделано правильно, у нас должен загореться светодиод.

Рассчитывать контрольную сумму всякий раз утомительно человеку. Поэтому, вы можете стереть последний байт. Программа сама рассчитает нужную величину контрольной суммы.

Для того чтобы выключить светящийся светодиод. Замените последний байт с 4 на 5 и отправьте снова команду на устройство.

24.4 Управляем командами – звук

Попробуем теперь вместо света включить, а затем выключить динамик.

Для этого, заменим предпоследний байт с 32 на 37. Управление включением — выключением динамика производится аналогично, как и в случае управления светом.

С полным набором команд вы можете ознакомится из документации. freim.pdf

Попробуем теперь дать команду не одну на конкретную машинку, а сразу на все доступные устройства.

Для этого, вместо первых шести байт МАК-адреса впишем нули, два последних байта заменим на FF FF, а сетевой адрес укажем, как FF FE.

Задание: Попробуйте дать указание машинке проехать 9 импульсов вперед.

§ 25 Эпизод двадцать пятый. Управление платформой Arduino. 9 класс

Привязка к тематическому планированию по информатике: Начала программирования. Язык программирования. Основные правила языка программирования.

25.1 Чуть подробнее про Arduino

Чтобы научится работать с платой Arduino— вам нужно:

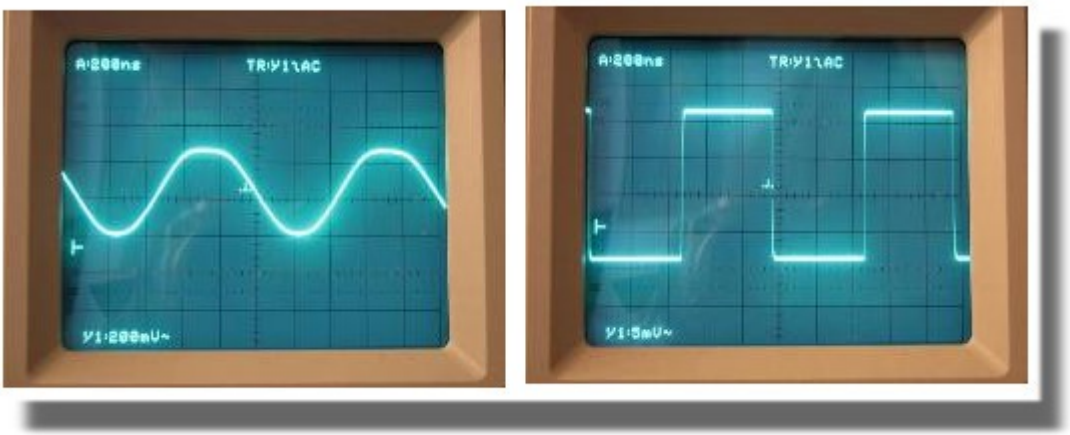
1. приобрести такую плату Arduino. Её можно заказать, например, в интернет-магазине, либо купить в ближайшем магазине радиодеталей, можно одолжить у друга, или записаться в школьный кружок, где есть секция робототехники;
2. скачать с сайта и установить на свой компьютер или ноутбук программу для работы с Arduino, которая называется Arduino IDE, (это аббревиатура английских слов Интегри́рованная Среда´ Разрабо́тки - IDE Integrated Development Environment);
3. соединить плату Arduino с компьютером при помощи шнура USB;
4. запустить программу Arduino IDE и сконфигурировать в ней параметры соединения), это, как правило, тип платы, порт, скорость общения);
5. написать программный код, который будет выполняться внутри вашей платы Arduino;
6. нажать кнопку компиляции и загрузить разработанный код в плату Arduino;
7. убедиться, что все работает так как было задумано с самого начала, при необходимости, подправить код в нужном месте и повторить его загрузку.

Что же можно делать с платой используя этот код? С программой идет набор примеров: можно начать знакомство с того, что позапускать эти примеры по-очереди — самое простое — это научиться управлять светодиодом, чтобы он начал мерцать с разным периодом. Потом позажигать несколько светодиодов. Потом поуправлять внешними контактами — подавать на них какую то информацию, либо считывать с ножек данные. Ну и конечно — покрутить моторчик. Какой же у нас это робот, который не умеет крутить моторчики. С разной скоростью и в разные стороны.

Контакты на ножках микрочипа контроллера бывают разные — бывают аналоговые, бывают цифровые. На аналоговых ножках сигнал может быть от нуля до какой-то предельной величины, а на цифровых — либо ноль, либо единица.

Датчики, которые покупаются отдельно и присоединяются к плате — тоже могут быть либо аналоговые, либо цифровые.

Если датчик с аналоговым выходом, то он на ножку даст сигнал в виде изменения напряжения (V) или сопротивления (Ом). И вы должны сами в рассмотреть в программе обработку этой величины, и по формулам, предложенным производителем этого датчика — пересчитать результаты, например, в силу падающего света, или в температуру.

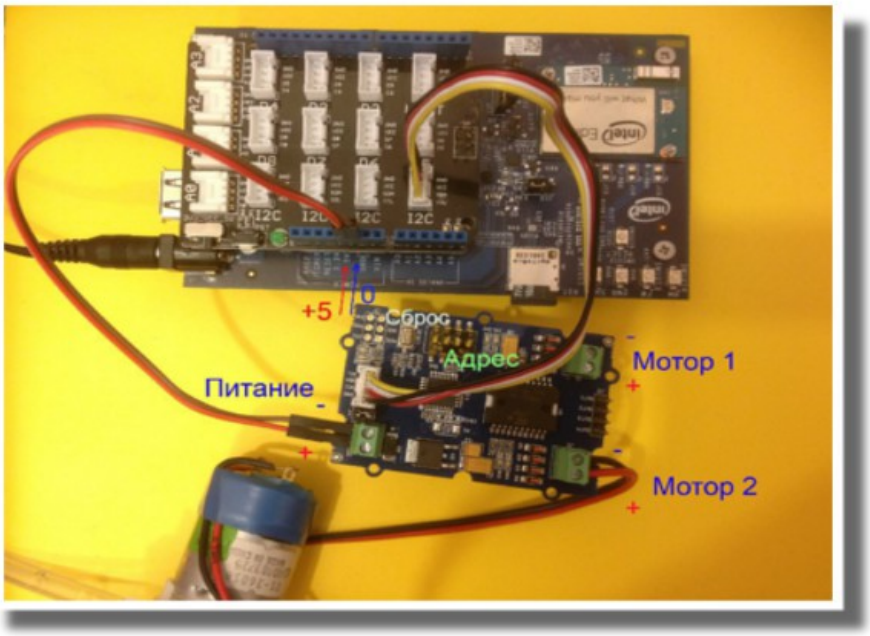


А если датчик цифровой, то он на ножку чипа дает информацию сразу в байтах — например какое расстояние до препятствия.

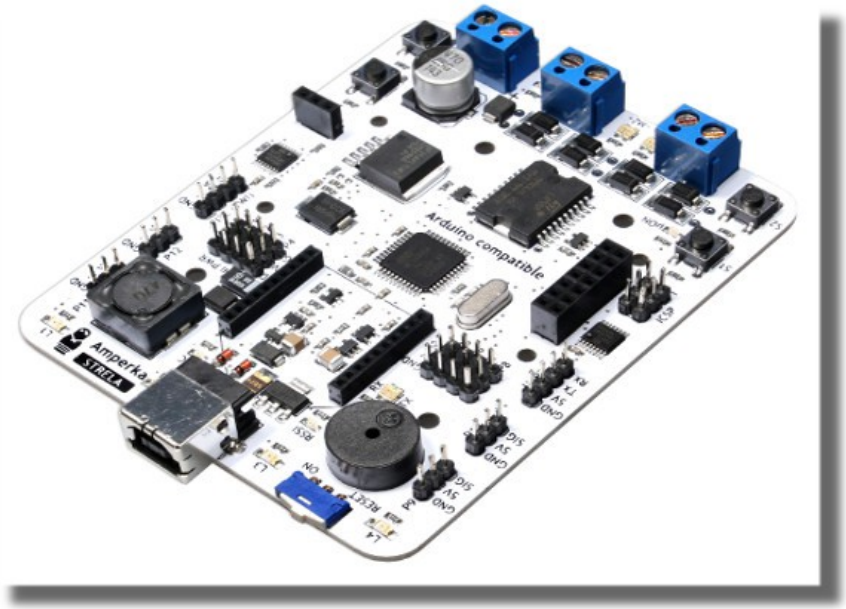
Иногда бывает удобно так — и иногда эдак, все зависит от поставленной задачи.

Следующим шагом, хотелось бы поэкспериментировать с управлением моторами.... Однако, напрямую к платеArduino подключить мотор, как правило, не получится. Дело в том, что мотор для своей работы потребляет очень много тока. Такой ток даже для простого моторчика от игрушечного автомобиля или самолетика выведет из строя микрочип контроллера, что уж говорить о мощных моторах, которые поднимают огромные тяжести. Чтобы через контроллер не проходили большие токи, используется силовая электрика. (опять без закона Ома при этих расчетах и проектировании — ни как не обойтись).

Поэтому моторы подключаются к Arduino при помощи специальных микросхем — ключей, или аппаратных драйверов. Не путайте это слово с драйверами — которые используются в ОС Windows. Если моторов, которыми вы хотите управлять много — четыре например или больше, то и драйверов должно быть несколько. Их выносят, как правило, на отдельную плату, которая называется шилд и присоединяется к Arduino через разъемы или шлейфом проводов.



Некоторые разработчики Arduino совмещают на одной плате и интеллектуальную часть и силовую, добавляют сюда и диоды, и кнопки, и динамик. Так поступает например компания Амперка со своей платой Стрела – на базе Arduino Leonardo для работы с двумя моторами.



С одной стороны это очень удобно, что приобретя такую плату сразу же можно поупражняться с разными примерами без присоединения других исполнительных элементов. Но если вы захотите что-то сделать отличающееся от того что задумали разработчики, например подключить третий мотор, то вынуждены будете все равно присоединять нужные компоненты. И если вдруг сгорит какой-то небольшой компонент на плате (резистор или диод), то скорее всего вся плата уже будет работать в нештатном режиме. И тогда не только вам, но и профессиональному разработчику встраиваемой электроники будет очень сложно понять, почему не горит нужный диод, или зажигается но не тот.

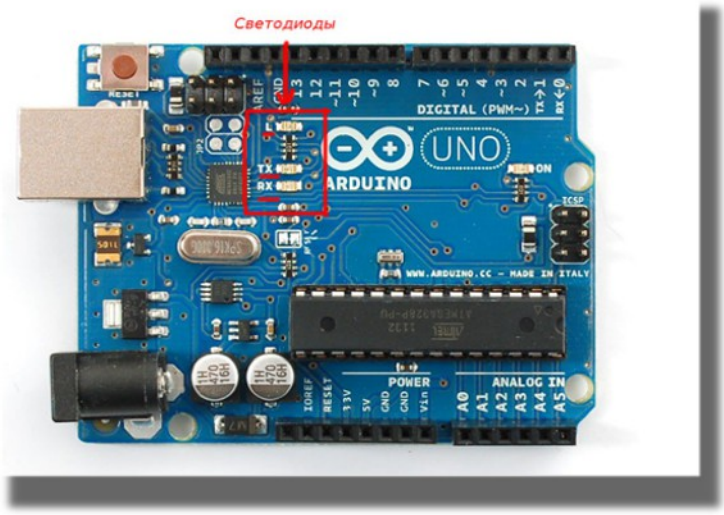
Когда вы подключите мотор и дадите команду на вращение мотора, то чтобы закрутить его в другую сторону, вам, скорее всего, придется изменить полярность подключения. Помните, у батареек бывает плюс и минус? А вот для управления скоростью вращения мотора, как правило, используется широтно-импульсная модуляция — ШИМ. На управлении изменением скорости вращения мотора базируется технология управления летающими коптерами.

Управление встроенным светодиодом:

25.2 Задачи эксперимента.

Подключение платы Arduino, запуск программы из библиотеки, проверка работы программы, модификация программы.

Первая программа, которую мы загрузим в плату Arduino – скетч, управляющий миганием светодиода.



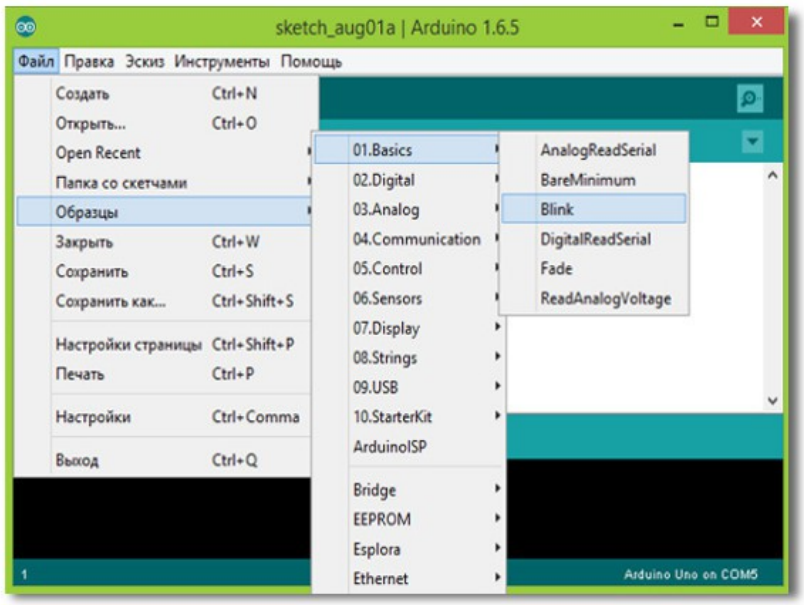
Рассмотрев внимательно схему платы, мы увидим четыре светодиода: ON, L, TX, RX.

Диод ON – сигнализирует о подключении питания к плате Arduino. Диод L – при подключении к USB разъему платы Arduino, будет сразу мигать, потому что в платы Arduino изначально загружена программа мигания светодиода.

Попробуем перепрограммировать Arduino внеся в скетч, изменения, влияющие на скорость, с которой мигает светодиод L.

Как мы уже упоминали, Arduino IDE включает в себя большую коллекцию примеров-скетчей, которые можно загрузить и использовать в своей работе. Скетч в котором запрограммирован мигающий светодиод называется «Blink».

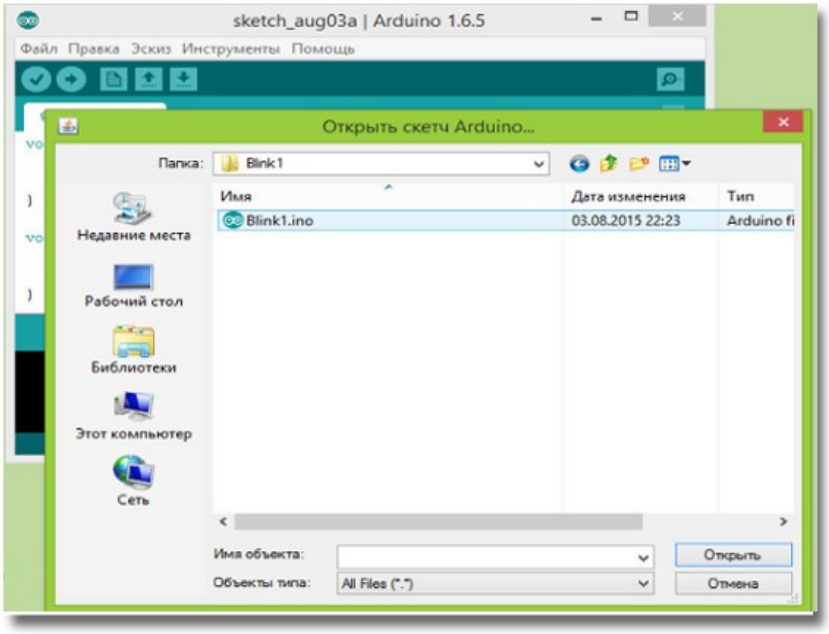
Давайте загрузим скетч «Blink», выполнив команду: «Файл» → «образцы» → «01.Basic» → «Blink»



После загрузки в окне IDE появится текст программы для мигания светодиода.

Но прежде, чем начать работы по изменению скорости мигания светодиода сохраним копию этого скетча. Для этого в меню Файл в IDE выбираем опцию «Сохранить как…». А затем, сохраним скетч, присвоив ему имя, например, Blink1.

Мы сохранили копию скетча «Blink» в своей папке. Это означает, что позже, можно будет просто открыть его с помощью опции «Файл» → «Открыть».



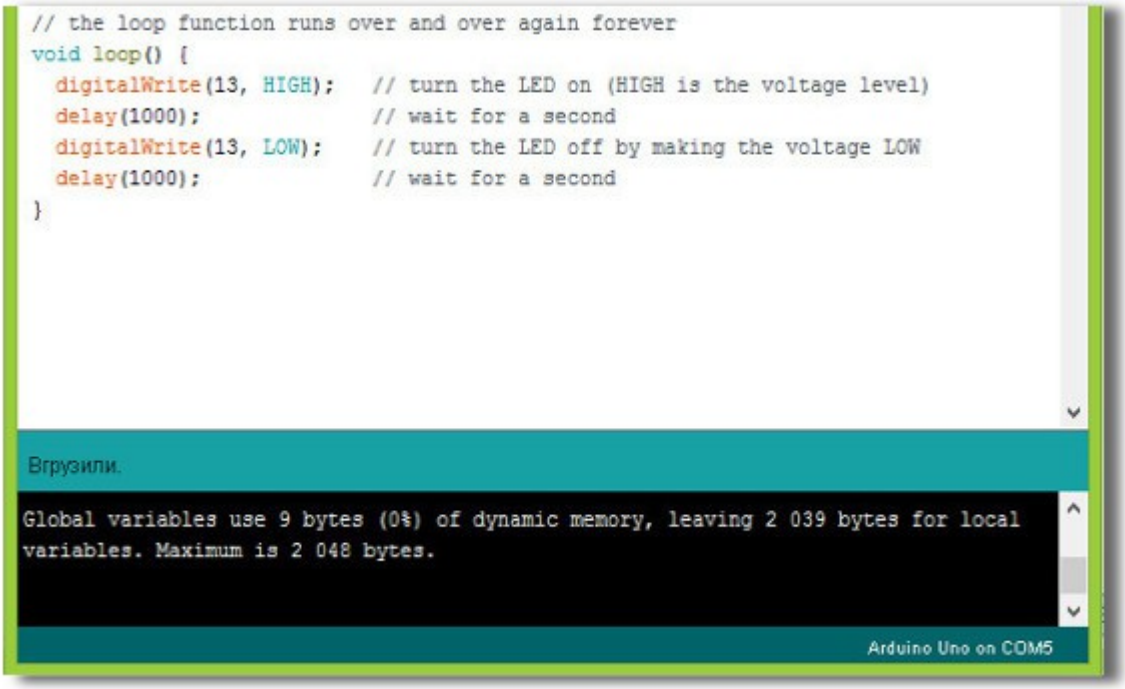
Итак, наш скетч имеет вид:

Для загрузки выбранного скетча в Arduino, нажмем на кнопку «Вгрузить». (Вторая кнопка слева на панели инструментов).

В нижней части IDE выводит ряд сообщений. Сначала выводится сообщение о компиляции скетча. То есть о преобразовании программы в формат, понятный для процессора.

Далее, статус процесса изменится на «Загружаем». На этот момент, светодиоды TX и RX на Arduino должны начать мерцать, указывая, что скетч загружается в плату.

Наконец, в строке состояния появится текст «Вгрузили».



Текст в информационном окне говорит нам, что скетч занимает 1084 байта из 30720 байт максимум.

Но, если мы забыли подключить плату, или драйверы установлены неправильно, или порт выбран неправильно, то в окне может появиться такая надпись, например:

В это случае нужно проверить подключение платы к компьютеру и еще раз загрузить скетч.

Код программы.

MyBlink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Обратите внимание, что текст программы состоит из непосредственно инструкций и комментариев этих инструкций. Комментарии не влияют на выполнение программы процессором. С помощью комментариев можно более подробно описывать работу программы.

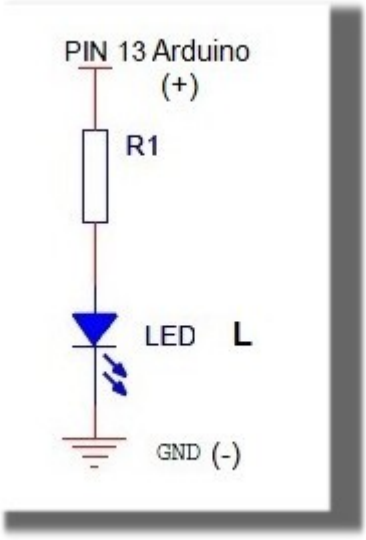
1. Многострочные комментарии выделяются символами/* ……*/ Текст в таких скобках не является какой-либо командой и не влияет на работу программы. В этих скобкам можно писать все, что угодно. Обычно в таких скобках пишут многострочные комментарии. Т.е. описание программы, дату создания и др.
2. //…. В случае с двумя наклонными палочками мы так же можем писать любой текст, и он не будет распознаваться программой. Но эти символы действуют только на одну строчку. Инструкции и комментарии в примерах, которые входят в IDE, написаны на английском языке. Но при самостоятельном написании скетча комментарии можно писать на русском языке.
3. Процедура setup().Эта процедура выполняется один раз в начале работы программы или при включении питания Arduino. И эта процедура должна присутствовать в каждом скетче. Последовательно выполняется каждая команда, заключенная между фигурными скобками этой процедуры. Каждая команда должна завершаться символом «;».

```
void setup() {
// initialize the digital pin as an output.
pinMode(13, OUTPUT);
}
```

В нашей программе процедура setup() содержит только одну команду pinMode(13,OUTPUT), которая сообщает Arduino то, что мы собираемся использовать вывод(пин) 13, как ВЫХОД. Это означает то, что на выводе 13 платы будут появляться электрические сигналы, которые создает сам контроллер. Ардуино.

Все эти действия обязательно вписываются между открывающей и закрывающей фигурными скобками { }.

Светодиод на плате Ардуино и на самом деле подсоединен к выводу 13.



4. Процедура loop().

После выполнения команд процедуры setup, процессор переходит к выполнению команд процедуры loop(). Особенность этой процедуры в том, что команды этой процедуры выполняются в бесконечном цикле. Т.е. после выполнения последней команды, процессор приступает к выполнению первой команды. И так, пока плата Ардуино подключена к питанию.

```
void loop() {
digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
```

Внутри цикла функции, размещена команда, которая переключает состояние пина в одно из двух возможных. HIGH или LOW. HIGH соответствует логической единице (она же в свою очередь соответствует напряжению питания микроконтроллера +5 Вольт.

Состояние вывода LOW соответствует логическому нулю, это примерно 0 Вольт.

Если пин 13 на плате Ардуино установить в состояние HIGH (5 Вольт), то светодиод начнет светиться, так как через него потечет электрический ток. Если пин 13 установить в состояние LOW (0 Вольт), то светодиод светиться не будет.

Следовательно, оператор digitalWrite(13, HIGH) формирует +5В на выходе 13 и зажигает таким образом светодиод L1.

```
delay(1000);
```

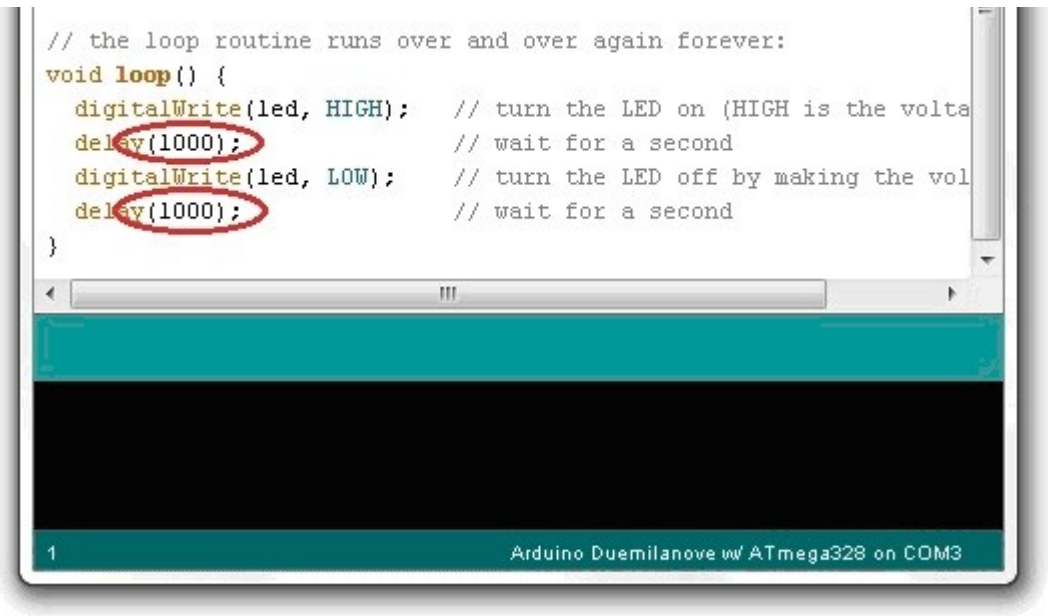
Эта команда заставляет микроконтроллер остановиться и ничего не делать целых 1000 миллисекунд. (Врезка 1000 мс = 1 секунда). Т.е. на выходе 13 в течении 1000 мс будет высокий уровень и светодиод будет светиться.

```
digitalWrite(13, LOW);
```

Эта команда устанавливает пин 13 в состояние LOW. Напряжение на выходе становится близким к нулю. И светодиод «гаснет». Затем снова выполняется команда паузы на 1000 миллисекунд.

На этом выполнение цикла заканчивается, но функция loop снова автоматически вызывается и запускает команду digitalWrite(13, HIGH). И так далее, пока подается питание на контроллер.

Немного модифицируем код, чтобы увидеть разницу с заводским миганием светодиода. Для этого надо изменить параметры команды delay.



Вместо строчки:

```
delay (1000);
```

напишем:

```
delay(100);
```

Теперь светодиод «L» должен загораться и гаснуть на десятую часть секунды, то есть, в 10 раз быстрее, чем в заводской версии.

Загрузим наш скетч в Arduino Uno и проверим, так ли это?

Если после загрузки светодиод начал мигать значительно быстрее, значит всё получилось. Теперь можно смело переходить к дальнейшим Экспериментам.

§ 26 Эпизод двадцать шесть. Чтение сигналов с порта Serial Port. 9 класс

Привязка к тематическому планированию по информатике: Начала программирования. Язык программирования. Разработка и выполнение программ в среде программирования.

В этом эпизоде рассмотрим управление контактами с порта Serial Port. Вспомним представление информации в двоичном виде.

26.1 Сигналы на выходе

Попробуем абстрагироваться от управления с помощью Arduino различных устройств, типа светодиодов, и просто посмотрим как изменения сигналы на выходах последовательного порта.

Простой пример управления этими контактами:

```
int P10 = 10;
int P9 = 9;

void setup() {
  PinMode(P9, OUTPUT);     // настраиваем P9 на выход
  PinMode(P10, OUTPUT);    // настраиваем P10 на выход
}
void loop() {
  DigitalWrite(P9, HIGH);     // подадим на P9 высокий уровень
  DigitalWrite(P10, HIGH);    // подадим на P10 высокий уровень
  delay(1000);
  DigitalWrite(P9, LOW);     // подадим на P9 низкий уровень
  DigitalWrite(P10, LOW);    // подадим на P10 низкий уровень
  delay(1000);
}
```

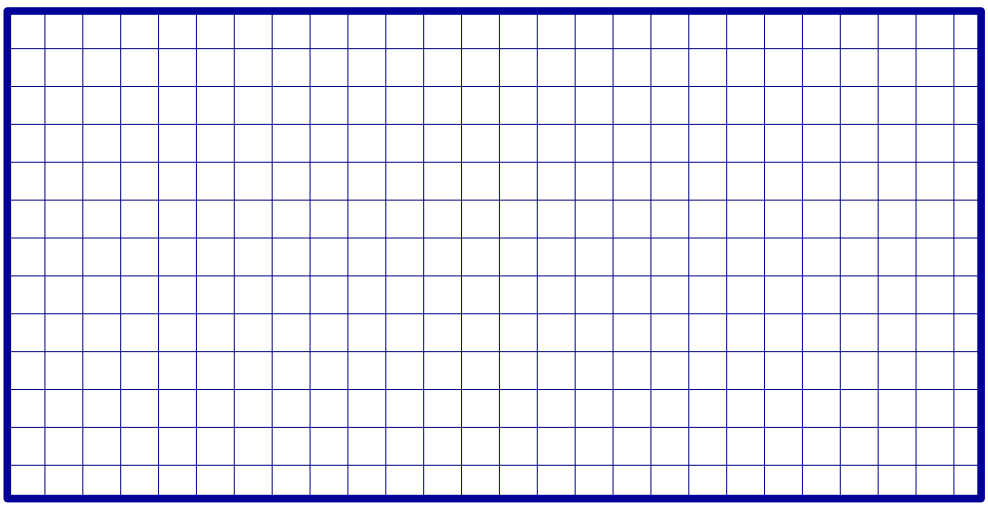
Давайте теперь изменим код программы не только для считывания уровня с ножек, но и для управления этими сигналами:

Каким будет начало программы, если те же самые контакты настроить на вход, а не на выход:

Задание:

Напишите предполагаемые команды в поле «Ваш ответ»

Ваш ответ



26.2 Сигналы на входе

Конечно вы сразу догадались, что все изменение заключается только в том, что команды OUTPUT необходимо заменить на команды INPUT,

но рассмотрим дальнейшие настройки последовательного порта:

```
int P10 = 10;
int P9 = 9;

void setup() {

PinMode(P9, INPUT);     // настраиваем P9 на вход

PinMode(P10, INPUT);   // настраиваем P10 на вход

Serial.begin(9600);       //Подготовим к работе Serial-порт

while (!Serial) {           // необходимо дождаться подключения виртуального Serial-порта
  }
}

void loop() {

Serial.print("P9 - ");    // Считаем значение с P9 и отправим его в последовательный порт

Serial.println(uDigitalRead(P9));

Serial.print("P10 - ");   //Считаем значение с P10 и отправим его в последовательный порт

Serial.println(uDigitalRead(P10));

delay(500);

}
```

§ 27 Эпизод двадцать семь. О контроллере R-5, Arduino Nano и о драйверах. 9 класс

Привязка к тематическому планированию по информатике: Моделирование и формализация. Использование компьютерных моделей при решении научно-технических задач.

Ранее мы проводили эксперименты с платой Arduino UNO.

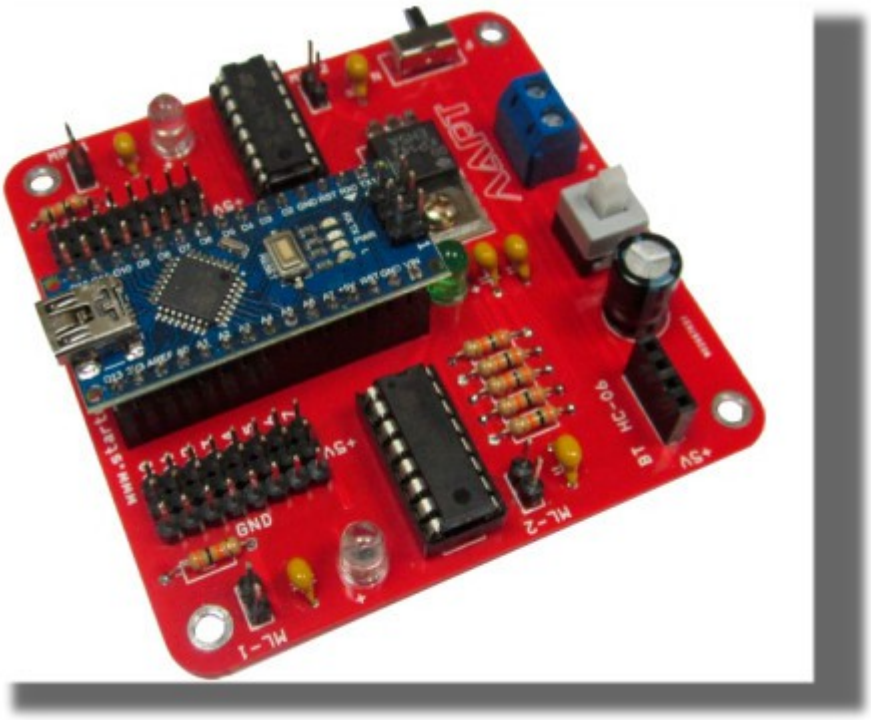
Но к сожалению, в нашей комплектации оборудования, поставить контроллер, например, вместо SmartCar 3, на платформу УМКИ у нас не получится :(. Умная машинка не поедет.

Чтобы управлять платформой с колесами нам нужно специальное устройство которое смогло бы преобразовывать электрические сигналы.

Такое устройство обеспечивающие преобразование электрических двоичных сигналов в воздействия, пригодные для непосредственного управления колесами называется **драйвером**.

Наеврное вы уже имеете представление о таком понятии - без драйвера-программы не будет работать видеокарта или сетевая карта в компьютере.

В нашем же случае драйвером будет служить специальная микросхема управляющая скоростью и направлением вращения электромотора.



Таким образом, нам будет гораздо удобнее работать, если мы воспользуемся специальным компактным модулем «Контроллер R-5»

В состав платы которого выходят:

•Контроллер Arduino NANO

•Стабилизатор питания 5 Вольт.

• Драйвер электромоторов (инвертор 74HC00 и L293D)

• Разъем для подключения модуля Bluetooth.

Электрическое питание может быть подано на плату через USB-порт от компьютера, или от источника питания. Источником питания может быть батарейка, аккумулятор или подключение через адаптер.

Самое главное не перепутать плюс (+) с минусом (–), а в диапазоне от 3-х до 12 вольт (не более 24 v.), платы Arduino будут вполне работоспособны.

27.1 Драйверы

Википедия дает следующие определения драйвера:

Драйвер — компьютерная программа, с помощью которой другие программы получают доступ к аппаратному обеспечению.
Драйвер — элемент задней втулки велосипеда, исполняющий роль храпового механизма.
Драйвер — наёмный водитель, который перегоняет автомобиль клиента из одного места в другое.
Драйвер — клюшка в гольфе.
Драйвер, сервоусилитель — в общем смысле устройство преобразования каких-либо сигналов до определённых параметров (например, драйвер RS-485). В узком смысле — источник высоких напряжений или токов, управляемый малым напряжением или током; такой драйвер применяется для управления электромотором (драйвер мотора), крупной светодиодной сборкой (драйвер светодиода) и т. д.

А в книге писателя Виктора Конецкого рассказывается: «Драйверами» называли когда-то самых отчаянных капитанов чайных клиперов. Драйверы и в ураганный ветер не спускали парусов и не брали рифов, а когда мачты уже готовы были улететь к чертовой матери, стреляли в парус из пистолета. Дырочку от пули ураганный ветер за десятые доли секунды превращал в огромные дыры, и парус обвисал лохмотьями. А мачты оставались на местах...

... слово «драйвер» работает у англичан в диапазоне от «гонщик» и «преследователь» до

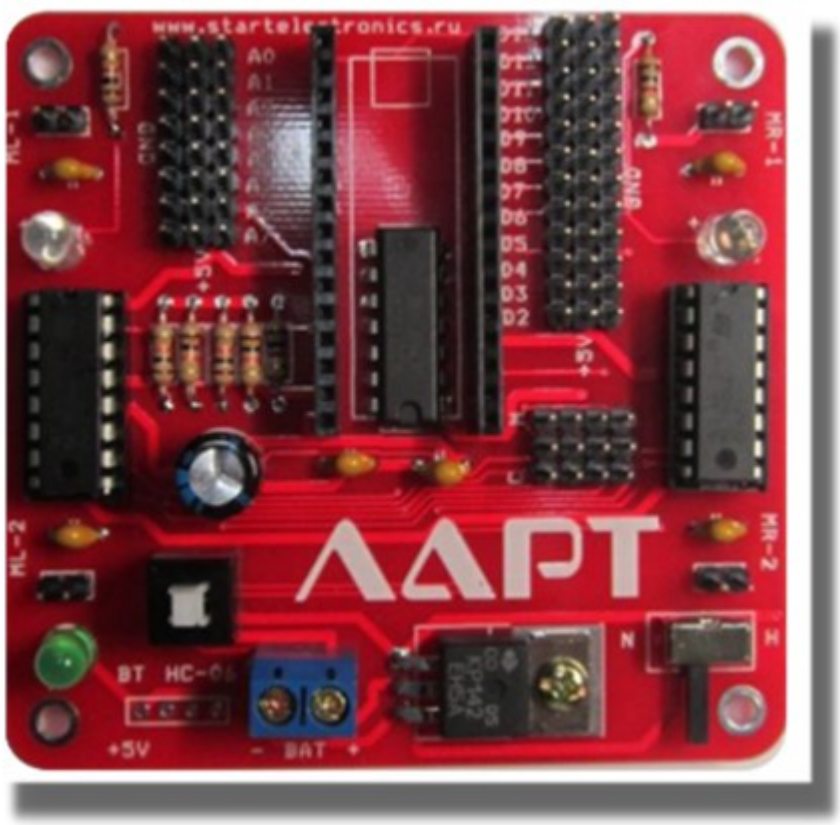
«надсмотрщик за рабами» и от мирного «кучер» до мрачного «доводящий до отчаяния». На американском сленге «драйв» — продажа товаров по дешевке с целью конкуренции, в горном деле «драйвер» — обыкновенный коногон, в сельском —погонщик скота, в медицинском «ту драйв мед» — сводить с ума. Еще это слово обозначает хозяина-эксплуататора, бизань-мачту, вождение автомобиля и… писательский труд («ту драйв э пен» — «гонять перо» в буквальном переводе).

Так что, вот какое интересное понятие — драйвер.

27.2 Блок управления роботом R-5

Итак в данном модуле рассматриваем «Мозг» робота контроллер – Arduino Nano.

Сопряжение, согласование, коммутация контроллера с датчиками, исполнительными устройствами и механизмами осуществляется при помощи блока управления R-5.

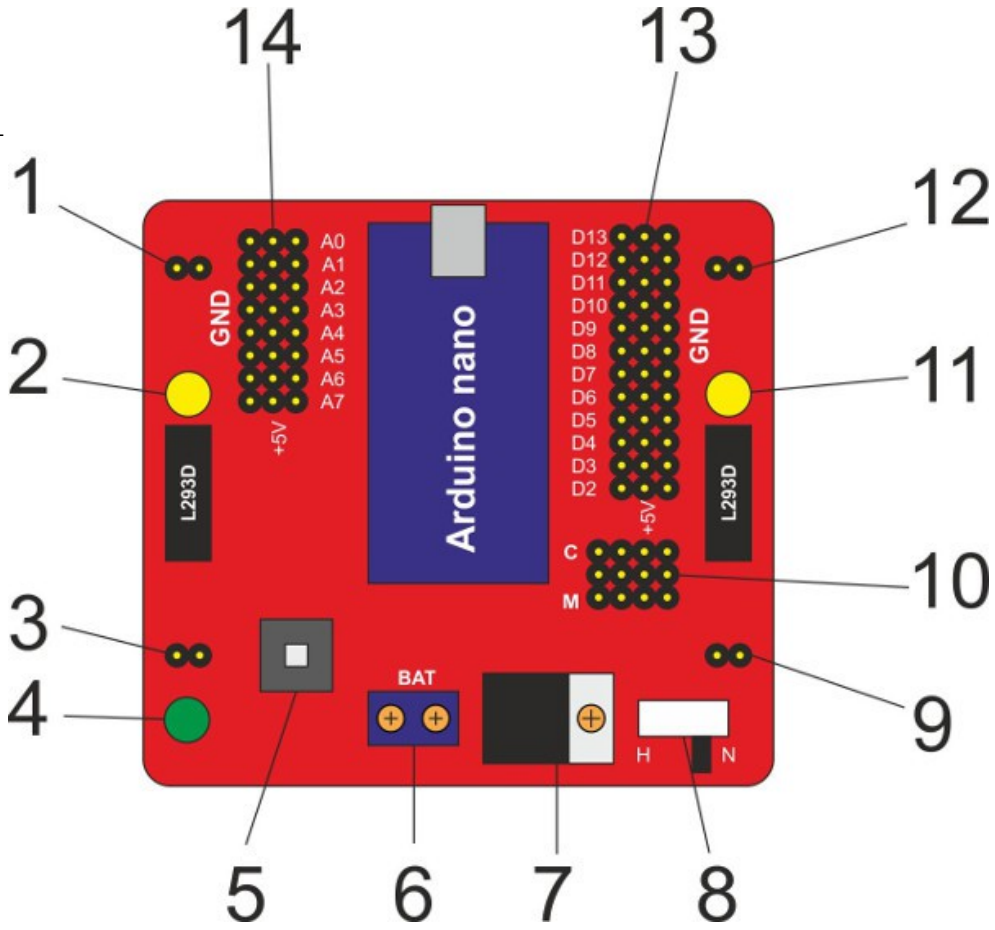


Для обеспечения удобства работы с внешними элементами на плате R-5 размещаются следующие компоненты:

- Колодки для установки Arduino Nano
- Драйвер моторов, построенный на инверторе 74HC00 и двух микросхемах L293D.
- Стабилизатор питания 5В на микросхеме 142ЕН5
- Переключатель питания электромоторов.

Переключатель подает на питание электромоторов или стабилизированное напряжение 5 Вольт, или напряжение питания батареи. Штыревые контакты, соответствующие выводам Arduino Nano.

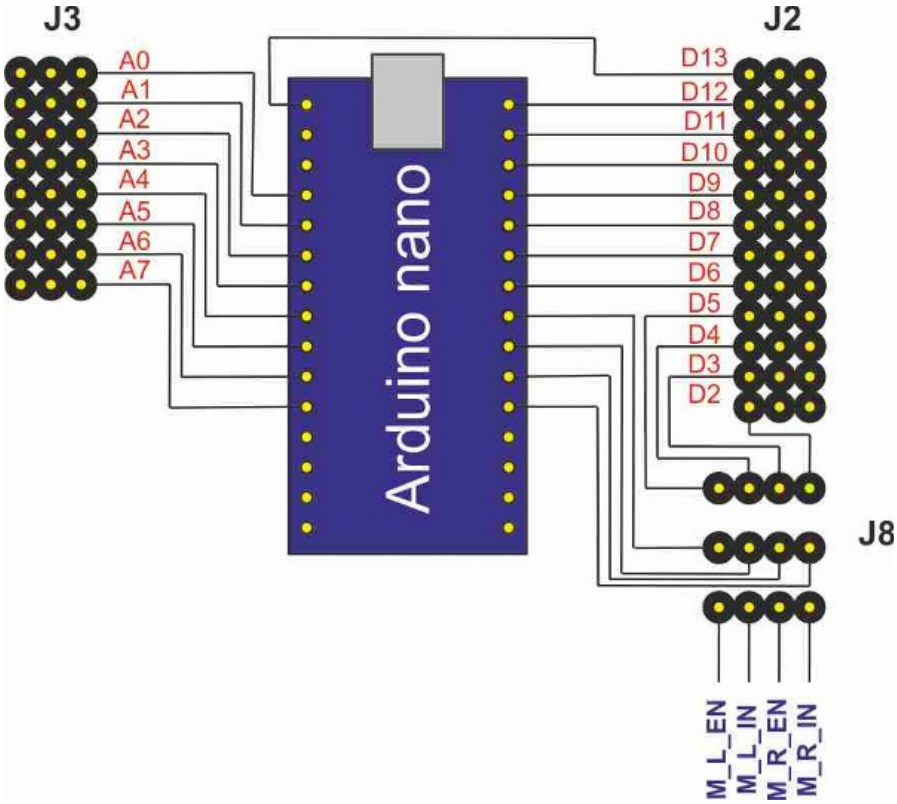
Назначение элементов блока R-5



подключения электромоторов.
2,11 – двунаправленные светодиоды, индицирующие направление вращения электромоторов.
4 – светодиод – индикатор электропитания.
5 – кнопка включения питания.
6 – клемма для подключения батареи питания.
7 – стабилизатор 5 Вольт.
8 – переключатель питания электромоторов. В положении N на драйвер электромоторов подается напряжение 5 вольт. В положении H на драйвер электромоторов подается питание непосредственно с батареи.
10 – разъем J8 для коммутации цифровых выходов Арудино D2, D3, D4, D5 на вход драйвера или на разъем J2.
13 – разъем J3. К этому разъему выведены соответствующие выходы Arduino.
14 – разъем J2. К этому разъему выведены аналоговые выходы Ардуино.

Разъемы J2, J3 имеют три ряда контактов. Внешний ряд подключен к минусу батареи (GND), средний ряд подключен к выходу стабилизатора 5 вольт. Внутренние ряды подключены к соответствующим контактам Arduino.

Схема подключения выходов Arduino



Цифровые и аналоговые контакты платы Arduino Nano выведены на соответствующие разъемы J2, J3.

Цифровые контакты D2, D3, D4, D5 с контроллера идут на разъем J8. В разъеме J8 джамперами выходы контроллера подключаются или к разъему J2 или к драйверу электромоторов.

Если блок управления R-5 используется для управления роботом, то джамперы необходимо установить в нижнее положение, что бы выходы Arduino соединились со входами драйвера.

Если блок R-5 используется для других экспериментов, то джамперы необходимо установить в верхнее положение.

Назначение входов драйвера электромоторов

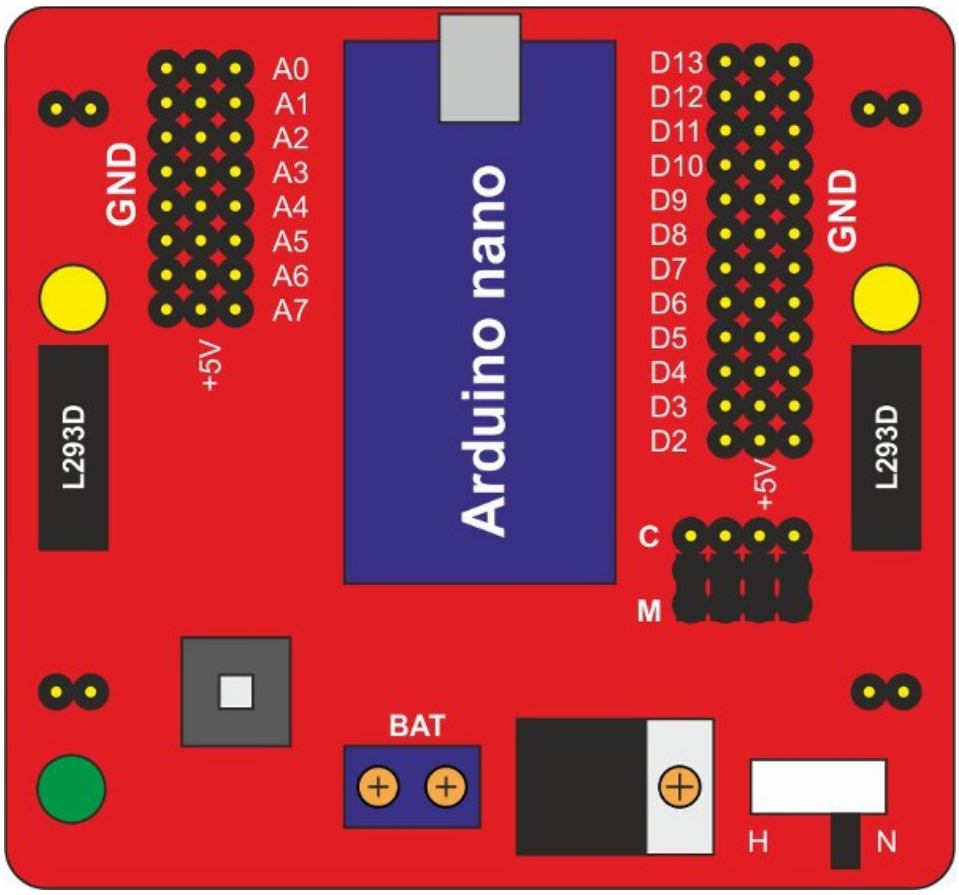
M_R_IN – вход драйвера, управляющий направлением вращения правого электромотора. Высокий уровень (HIGHT) – вращение вперед.

M_R_EN – вход драйвера, разрешающий вращение правого электромотора. Высокий уровень (HIGHT) разрешает вращение. При подаче на вход сигнала с широтно-импульсной модуляцией (ШИМ, PWM) вход управляет скоростью вращения.

M_L_IN – вход драйвера, управляющий направлением вращения левого электромотора. Высокий уровень (HIGHT) – вращение вперед.

M_L_IN – вход драйвера, разрешающий вращение левого электромотора. Высокий уровень (HIGHT) разрешает вращение. При подаче на вход сигнала с широтно-импульсной модуляцией (ШИМ, PWM) вход управляет скоростью вращения.

Т.е. исходя из вышесказанного для того, что бы мы могли управлять моторами робота мы должны соединить джамперами средние и нижние контакты разъема J8.



Если мы хотим использовать выходы D2, D3, D4, D5 в работе других устройств, то в этом случае джамперами необходимо соединить средние и верхние контакты разъема.

27.3 Управление роботом. Движение робота вперед, назад

Постановка задачи

Теперь уже приступаем непосредственно к программированию. Как вы уже поняли, программа для контроллера Ардуино обычно состоит из трех частей.

В первой части, как в обычной математической задаче описываются исходные данные. Мы присваиваем названия, назначение, функционал контактов контроллера. Записываем, какие библиотеки мы будем использовать в ходе выполнения программы.

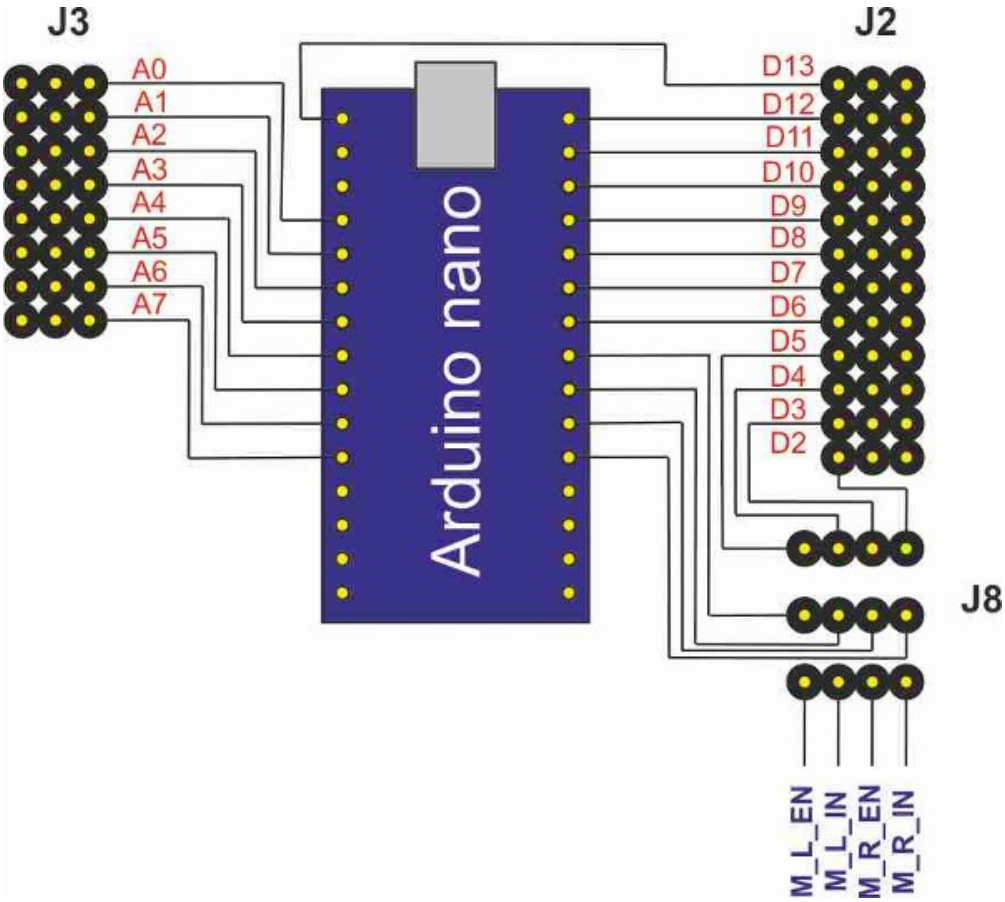
Во второй части с названием **Setup** пишем уже непосредственно код программы. Но этот код исполняется только один раз за время работы программы. Это необходимо для того, что бы запрограммировать выходы, входы контроллера на все время действия программы.

И третья часть программы **loop** представляет собой команды, которые исполняются процессором в течение всего периода работы процессора.

Анализируем плату

В первой части кода мы определяем, какими контактами платы Arduino мы будем управлять драйвером и соответственно электромоторами, придумаем название команд, что бы нам было понятно их назначение.

В блоке R-5 контакты Arduino жестко подключены к входам драйвера. Давайте посмотрим на рисунок ниже:



M_R_IN – вход драйвера, управляющий направлением вращения правого электромотора. Высокий уровень (HIGHT) – вращение вперед.

M_R_EN – вход драйвера, разрешающий вращение правого электромотора. Высокий

уровень (HIGHT) разрешает вращение. При подаче на вход сигнала с широтно-импульсной модуляцией (ШИМ, PWM) вход управляет скоростью вращения.
M_L_IN – вход драйвера, управляющий направлением вращения левого электромотора. Высокий уровень (HIGHT) – вращение вперед.
M_L_IN – вход драйвера, разрешающий вращение левого электромотора. Высокий уровень (HIGHT) разрешает вращение. При подаче на вход сигнала с широтно-импульсной модуляцией (ШИМ, PWM) вход управляет скоростью вращения.
И у нас получается следующее:
Контакт Arduino 2 (D2) – направление вращения правого мотора
Контакт Arduino 3 (D3)– разрешение вращения правого мотора. При работе ШИМ – регулировка скорости вращения.
Контакт Arduino 4 (D4)– направление вращения левого мотора.
Контакт Arduino 5 (D5)– разрешение вращения левого мотора. При работе ШИМ – регулировка скорости вращения.

Пишем код

```
#define DIR_R   2 //  управлять направлением вращения правого мотора будем с контакта 2
#define SPEED_R  3  // управлять разрешением вращения и скоростью вращения правого
//мотора  будем с контакта 3
#define DIR_L  4      //управлять направлением вращения левого мотора будем с контакта 4
#define SPEED_L 5 // управлять разрешением вращения и скоростью вращения левого
//мотора будем с контакта 5

//В этой части кода больше не будем задавать ни каких параметров

 // приступаем ко второй части программы. Мы знаем, что в этой части кода команды
//исполняются   только один раз

void setup()
{
 pinMode (DIR_R, OUTPUT); // Драйвер управляется выходными сигналами с Ардуино.
 //Поэтому мы определяем все контакты , как OUTPUT
 pinMode (SPEED_R, OUTPUT);
 pinMode (DIR_L, OUTPUT);
 pinMode (SPEED_L, OUTPUT);
}

 // И в третьей части кода мы уже пишем алгоритм работы. Т.е. то, что наш робот
должен выполнять
void loop()
{
 digitalWrite (DIR_R, HIGH); // Команда digitalWrite устанавливает на контакте 2 высокий
 //уровень.  Для драйвера моторов это означает то, что мотор будет вращаться вперед
 digitalWrite (SPEED_R, HIGH); //  высокий уровень на контакте 3 разрешает драйверу
 //вращать электромотор
 digitalWrite (DIR_L, HIGH);
 digitalWrite (SPEED_L, HIGH);
 delay(1000); // Вращаем 1 сек
```

```
 digitalWrite (DIR_R, HIGH);
 digitalWrite (SPEED_R, LOW); // Низкий уровень запрещает вращение моторов
 digitalWrite (DIR_L, HIGH);
 digitalWrite (SPEED_L, LOW);
 delay(1000);

 digitalWrite (DIR_R, LOW); // Включаем низкий уровень и мотор должен вращаться в
//обратную  сторону
 digitalWrite (SPEED_R, HIGH); //Разрешаем вращение мотора
 digitalWrite (DIR_L, LOW);
 digitalWrite (SPEED_L, HIGH);
 delay(1000);

 digitalWrite (DIR_R, LOW);
 digitalWrite (SPEED_R, LOW); // Вращение запрещено
 digitalWrite (DIR_L, LOW);
 digitalWrite (SPEED_L, LOW);
 delay(1000);
}
```

Скопируйте текст программы и вставьте его в Arduino IDE.

После копирования обязательно проверьте, что бы комментарии в каждой строчке начинались с двух символов //.

Загрузите код в контроллер. Включите питание робота.

Наш робот, исполняя написанный выше скетч, должен в течение одной секунды ехать вперед, затем на секунду остановиться и начать движение назад. И так как команда loop исполняется постоянно, то робот будет выполнять эти команды пока включено питание.

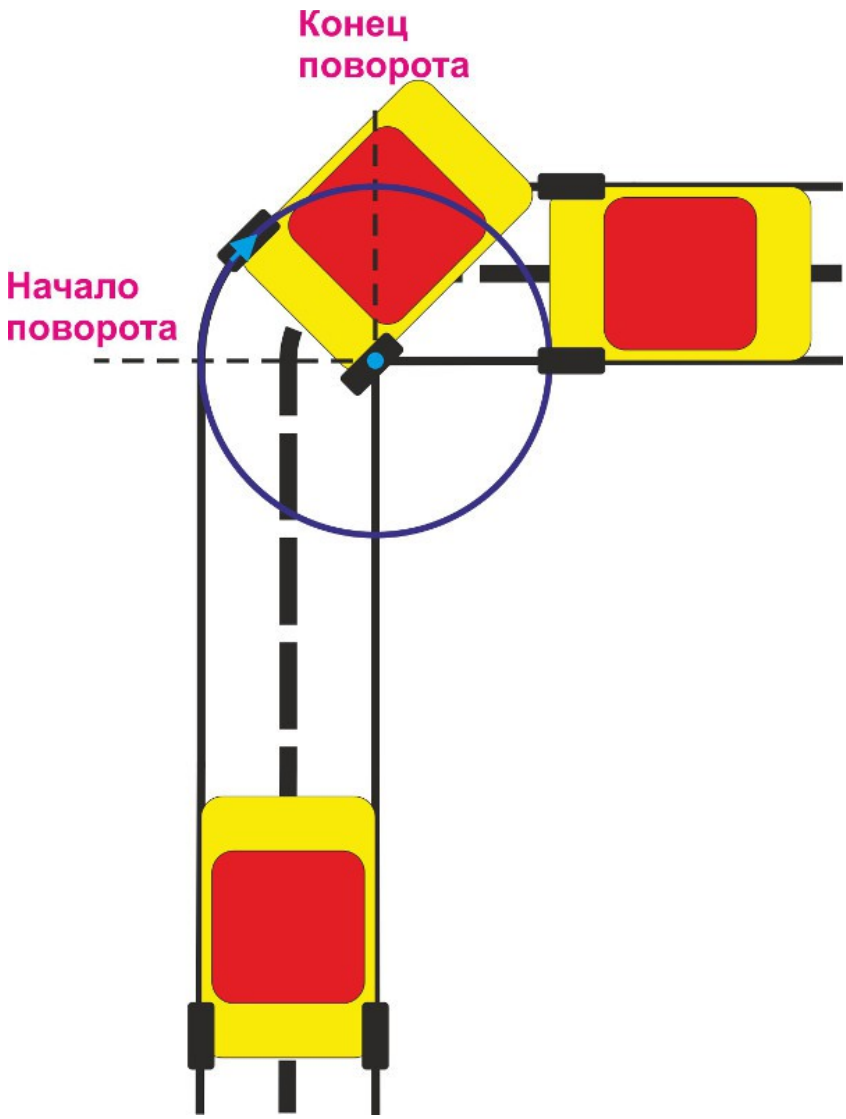
27.3 Повороты влево, вправо

1 Вариант поворота №1.

Мы будем рассматривать идеальный случай: колеса не проскальзывают по поверхности, скорость движения стабильная. При правильных расчетах робот точно выполняет все команды.

В первом случае поворот осуществляется левым колесом. Правое колесо остановлено и является центром радиуса поворота.

Прежде, чем писать программу поворота, попробуем графически изобразить движение робота при повороте вправо на 90 градусов.



Из рисунка следует, что радиус поворота равняется расстоянию между центрами колес. И длина хода левого колеса равна одной четверти длины окружности.
Измерив расстояние между центрами колес на нашем роботе мы получим радиус окружности. Зная радиус мы сможем найти путь, который должно пройти левое колесо для поворота направо на 90 градусов.

Радиус окружности –
Длина пути левого колеса –
Время вращения левого колеса -

2 Пишем код

```
        #define DIR_R   2
#define SPEED_R  3
#define DIR_L  4
#define SPEED_L 5

void setup()
{
  pinMode (DIR_R, OUTPUT);
  pinMode (SPEED_R, OUTPUT);
  pinMode (DIR_L, OUTPUT);
  pinMode (SPEED_L, OUTPUT);
```

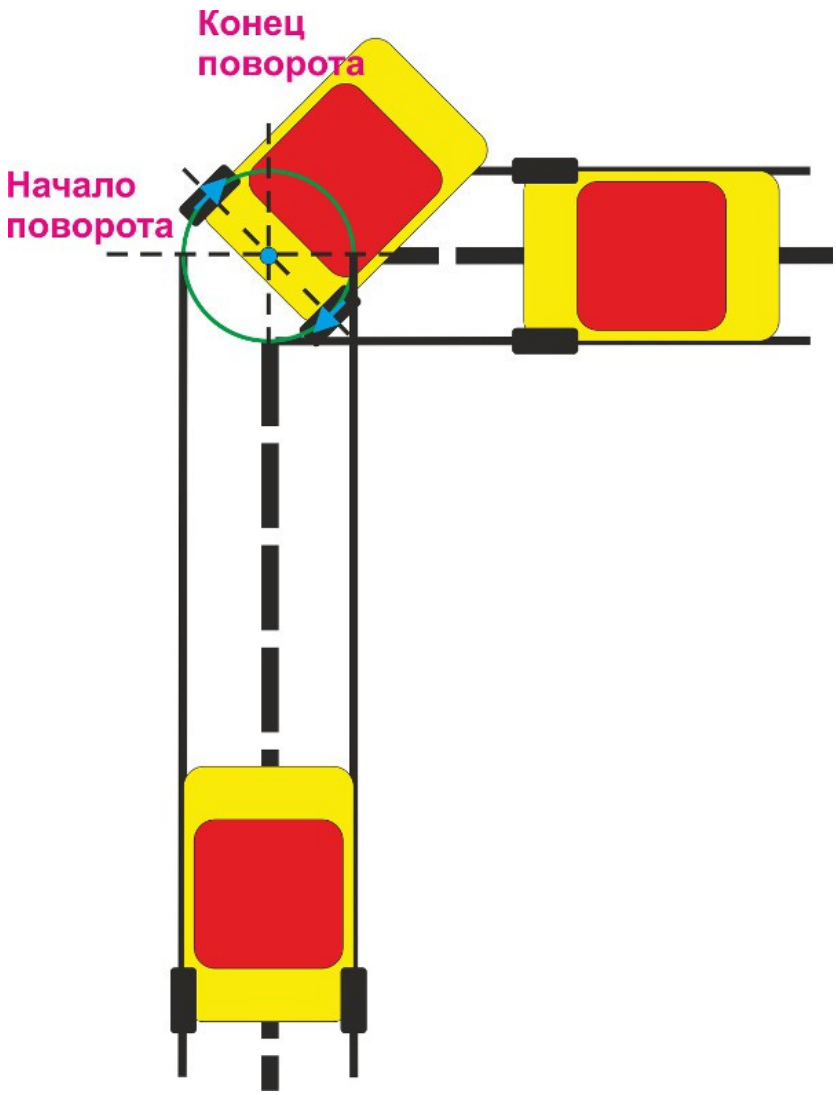
```
}

 void loop()
{
  digitalWrite (DIR_R, HIGH);
  digitalWrite (SPEED_R, HIGH);
  digitalWrite (DIR_L, HIGH);
  digitalWrite (SPEED_L, HIGH);
  delay(1000); // Вращаем 1 сек

  digitalWrite (DIR_R, HIGH);
  digitalWrite (SPEED_R, LOW); // Низкий уровень запрещает вращение моторов
  digitalWrite (DIR_L, HIGH);
  digitalWrite (SPEED_L, LOW);
  delay(2000); //останов 2 сек.

//далее левое колесо вращаем 1 сек., правое колесо остановлено
//после поворота направо робот сразу начинает движение вперед без паузы.
  digitalWrite (DIR_R, HIGH);
  digitalWrite (SPEED_R, HIGH);
  digitalWrite (DIR_L, HIGH);
  digitalWrite (SPEED_L, LOW);
  delay(1000); // Вращаем 1 сек

//вращаем оба колеса 1 сек.
  digitalWrite (DIR_R, HIGH);
  digitalWrite (SPEED_R, HIGH);
  digitalWrite (DIR_L, HIGH);
  digitalWrite (SPEED_L, HIGH);
  delay(1000); // Вращаем 1 сек

//вращаем правое колесо 1 сек., левое колесо остановлено
  digitalWrite (DIR_R, HIGH);
  digitalWrite (SPEED_R, LOW);
  digitalWrite (DIR_L, HIGH);
  digitalWrite (SPEED_L, HIGH);
  delay(1000); // Вращаем 1 сек.
  }
```

Скопируйте текст программы и вставьте его в Arduino IDE.
После проверки загрузите код в контроллер. Установите контроллер в блок R-5 и включите питание робота.

Во втором варианте для поворота мы будем вращать колеса в разные стороны. В этом случае значительно уменьшается радиус поворота. Робот разворачивается практически вокруг своей оси.

На рисунке видно, что радиус разворота равен половине расстояния между серединами колес.

Таким образом мы сможем посчитать время для поворота.
Половина расстояния между серединами колес (радиус поворота) –
Длина окружности поворота –
Четверть длины окружности –
Время прохождения участка поворота при средней измеренной скорости –



Получаем, что длина окружности нашего колеса составляет 132 мм. Умножаем 11 метров, пройденных роботом за 10 секунд на 6 (в минуте 60 сек.). Затем делим на длину окружности колеса. 11000*6/132 = 500 об./мин.

Так же нам может пригодиться такой параметр, как время, за которое колесо сделает один оборот.

Скорость вращения колес – 500 об/мин.
Длина окружности колеса – 132 мм.
Время, за которое колесо сделает один оборот (робот проедет 132 мм) – 120 миллисекунды.

Теперь, зная эти кинематические характеристики нашего робота, мы сможем строить различные траектории. Рассчитывать остановки, повороты и пр. действия робота.

4 Пишем код

```
        #define DIR_R   2
#define SPEED_R  3
#define DIR_L  4
#define SPEED_L 5

void setup()
{
  pinMode (DIR_R, OUTPUT);
  pinMode (SPEED_R, OUTPUT);
  pinMode (DIR_L, OUTPUT);
  pinMode (SPEED_L, OUTPUT);
```

```
}

 void loop()
{
  digitalWrite (DIR_R, HIGH);
  digitalWrite (SPEED_R, HIGH);
  digitalWrite (DIR_L, HIGH);
  digitalWrite (SPEED_L, HIGH);
  delay(1000); // Вращаем 1 сек

  digitalWrite (DIR_R, HIGH);
  digitalWrite (SPEED_R, LOW); // Низкий уровень запрещает вращение моторов
  digitalWrite (DIR_L, HIGH);
  digitalWrite (SPEED_L, LOW);
  delay(2000); //останов 2 сек.

//далее левое колесо вращаем 1 сек. вперед, правое колесо вращаем назад
//после поворота направо робот сразу начинает движение вперед без паузы.
  digitalWrite (DIR_R, HIGH);
  digitalWrite (SPEED_R, HIGH);
  digitalWrite (DIR_L, LOW);
  digitalWrite (SPEED_L, HIGH);
  delay(1000); // Вращаем 1 сек

//вращаем оба колеса 1 сек.
  digitalWrite (DIR_R, HIGH);
  digitalWrite (SPEED_R, HIGH);
  digitalWrite (DIR_L, HIGH);
  digitalWrite (SPEED_L, HIGH);
  delay(1000); // Вращаем 1 сек

//вращаем правое колесо 1 сек. назад, левое колесо вращаем вперед
  digitalWrite (DIR_R, LOW);
  digitalWrite (SPEED_R, HIGH);
  digitalWrite (DIR_L, HIGH);
  digitalWrite (SPEED_L, HIGH);
  delay(1000); // Вращаем 1 сек
  }
```

Скопируйте текст программы и вставьте его в Arduino IDE.
После проверки загрузите код в контроллер. Установите контроллер в блок R-5 и включите питание робота.

§ 28 Эпизод двадцать восемь. Управление роботом. Движение робота по линии. 9 класс

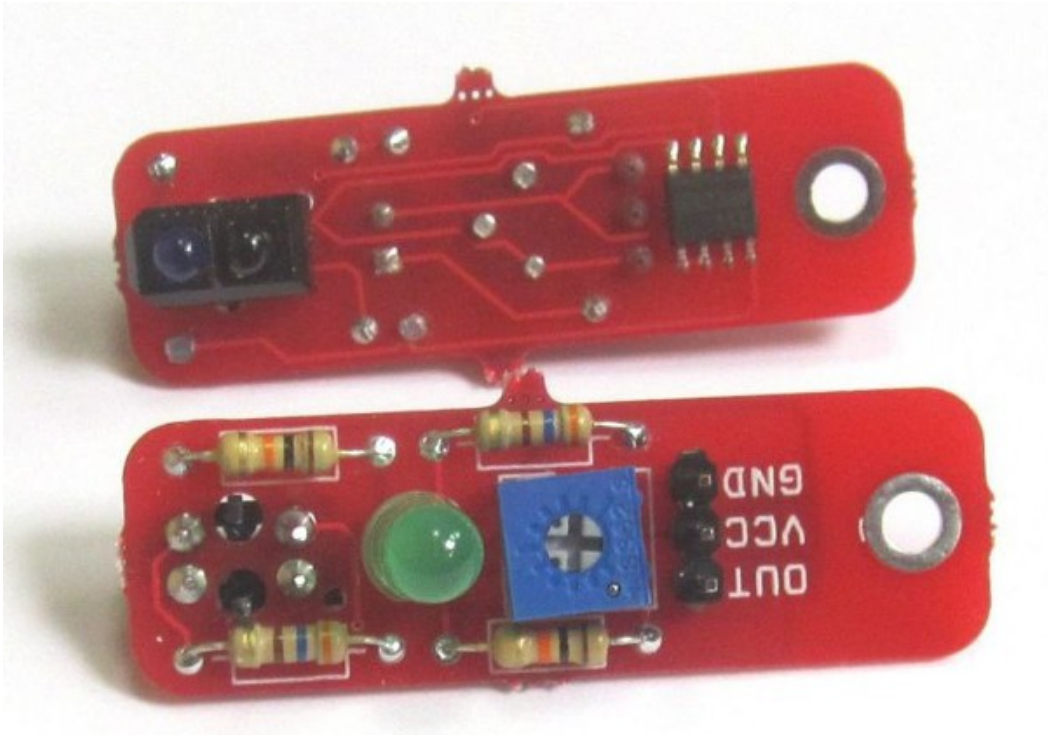
Привязка к тематическому планированию по информатике: Моделирование и формализация. Использование компьютерных моделей при решении научно-технических задач.

28.1 Датчики

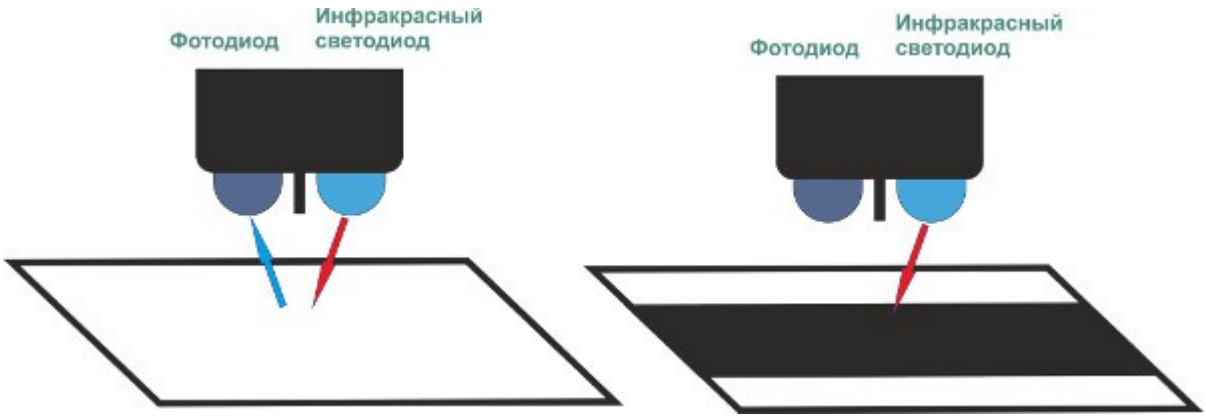
Движение робота по линии - самый популярный эксперимент начинающего роботостроителя.

Для того, что бы наш робот мог двигаться по полю с черной линией нам необходимы уже датчики, которые смогли бы различать белое и черное поля. Эти датчики называются датчиками линии. В их задачу входит определять наличие белого или черного поля и сообщать об этом контроллеру.

Внешний вид датчика



Для определения белого или черного поля используется электронный компонент, состоящий из инфракрасного светодиода(ИК) и инфракрасного фотодиода. ИК светодиод излучает свет в инфракрасном диапазоне с длиной волны порядка 940 нанометра. Такой диапазон света используется для защиты от помех, создаваемых осветительными лампами, солнечным светом.



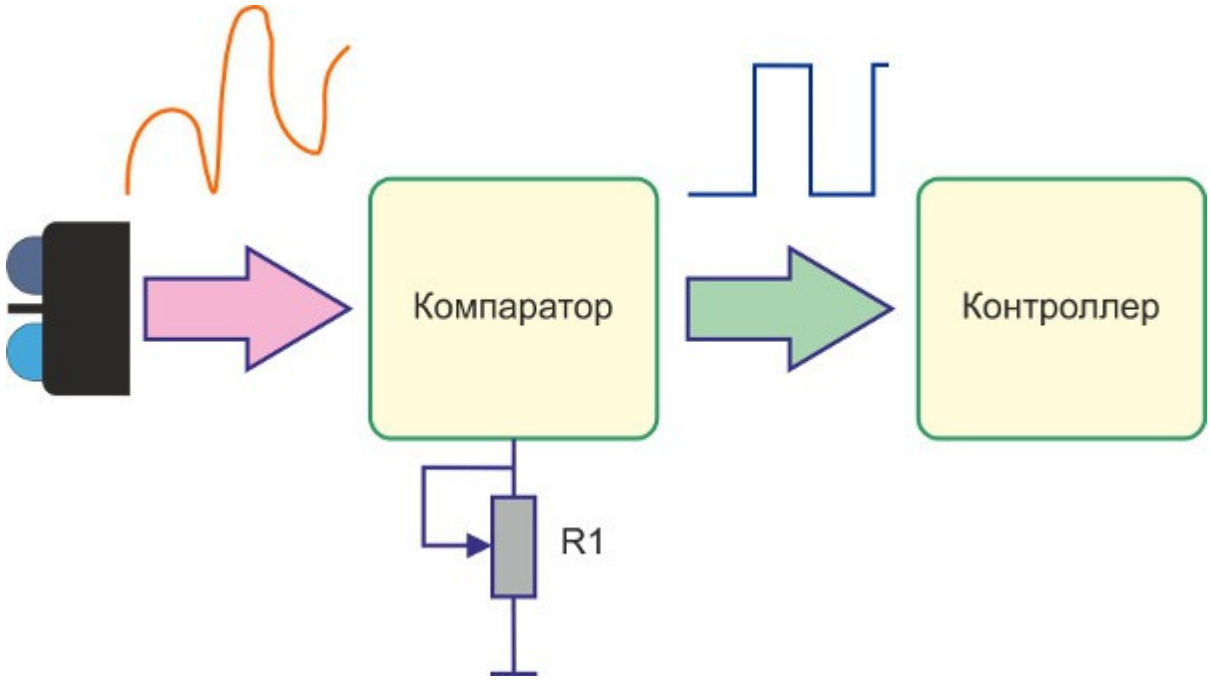
Рассмотрим два варианта возможных событий.

1.Датчик находится над белым полем.

Отраженный от белого поля свет попадает на фотодиод. Далее уже оцифрованный компаратором электрический сигнал поступает на вход контроллера. Контроллер исходя из написанной программы воспринимает этот сигнал именно, как белое поле.

2.Датчик находится над черным полем.

Как мы знаем их физики, черное поле поглащает свет. Отражения не происходит. Такое состояние датчика контроллером воспринимается, как наличие черного поля. На выходе фотодиода формируется хаотичный аналоговый сигнал. Для повышения помехозащищенности в схему датчика включен компаратор, который отсеивает помехи, ложные отражения и др. Порог срабатывания компаратора устанавливается резистором R1. С выхода компаратора мы получаем сигнал в цифровом виде.



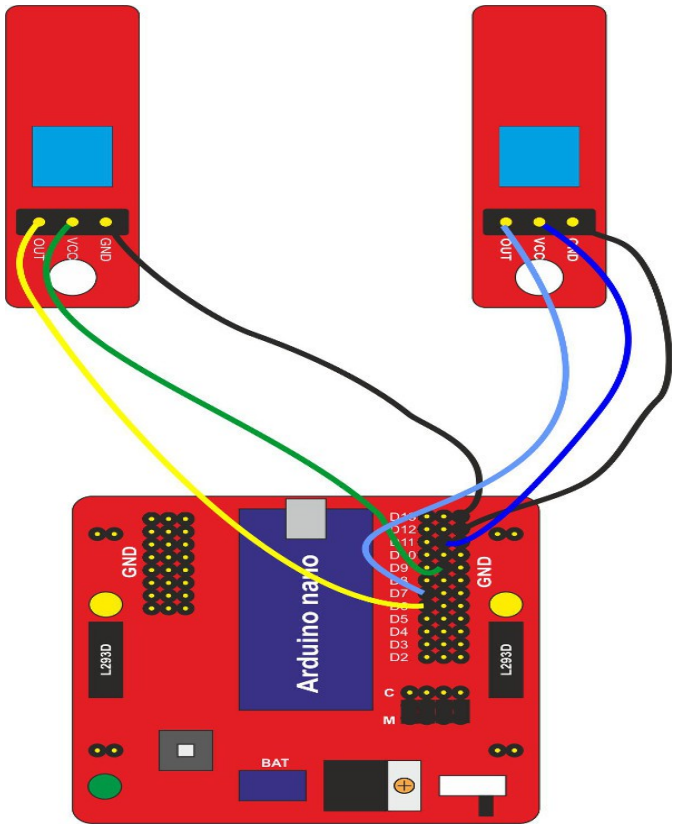
28.2 Подключение датчиков

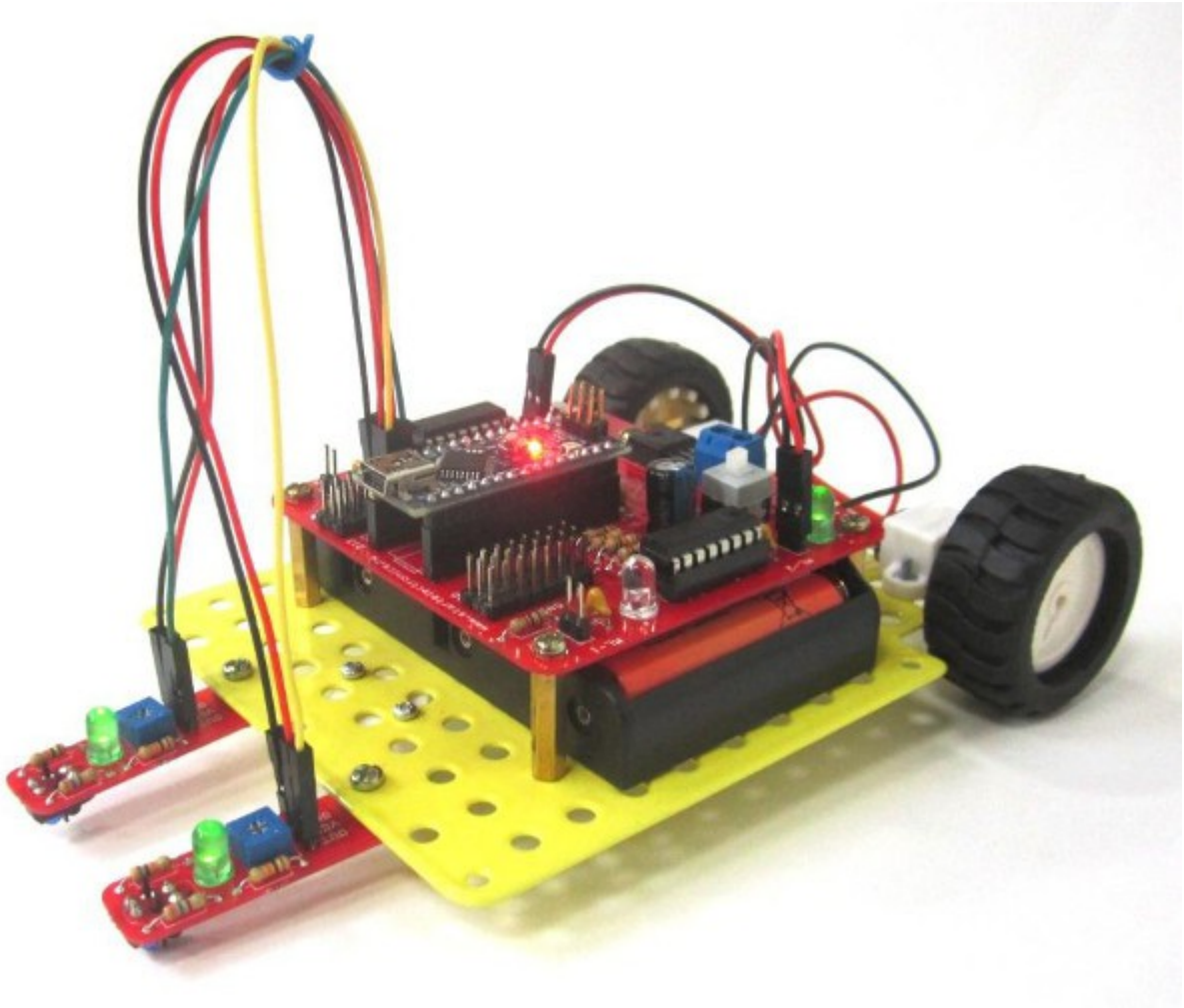
Устанавливаем датчики линии и подключаем их к блоку управления R-5. Контакты GND на датчиках соединяем проводками с контактами GND на блоке R-5(в разъеме J3 правый ряд контактов). Контакт VCC на датчиках соединяем с контактами +5V на блоке R-5 (средний ряд контактов). Контакты OUT левого и правого датчиков подключаем к контактам разъема J3. К каким контактам подключим выходы датчиков, такие и надо будет нам указывать в нашей программе.

Помним, что контакты D2-D5 у нас заняты уже драйвером моторов. Подключим датчики следующим образом.

Левый датчик – D6

Правый датчик – D7





27.3 Пишем код

Перед написанием кода рассмотрим на рисунке ситуации, которые могут возникать при движении робота.

ШАГ1. Датчики робота «видят» белое поле. Контроллер принимает решение включить оба мотора и двигаться прямо.

ШАГ2. Робот правым датчиком наезжает на черную линию. Для выравнивания положения необходимо притормозить правое колесо.

ШАГ3. Робот наезжает левым датчиком на черную линию. Для выравнивания положения необходимо притормозить вращение левого колеса.

В этом эксперименте поворот в сторону линии сделаем следующим образом.
При наезде правым датчиком на черную линию мы останавливаем правый мотор.
Выравнивание хода осуществляется левым мотором.
При наезде левым датчиком на черную линию мы останавливаем левый мотор
Выравнивание хода осуществляется правым мотором.

Если оба датчика оказываются над черной линией, то робот продолжает движение в прямом направлении

```
#define LS 6      //  левый датчик
#define RS 7      // правый датчик

#define DIR_R   2
#define SPEED_R  3
#define DIR_L  4
#define SPEED_L  5

void setup()
{
  pinMode(LS, INPUT);
  pinMode(RS, INPUT);
  pinMode(DIR_R , OUTPUT);
  pinMode(SPEED_R , OUTPUT);
  pinMode(DIR_L , OUTPUT);
  pinMode(SPEED_L, OUTPUT);
}
void loop()
{
  if(digitalRead(LS==HIGH) && digitalRead(RS==HIGH))     // Движение вперед
  {
    digitalWrite(DIR_R , HIGH);
    digitalWrite(SPEED_R ,HIGH);
    digitalWrite(DIR_L , HIGH);
    digitalWrite(SPEED_L ,HIGH);
  }

  if((digitalRead(LS==LOW)) && digitalRead(RS==HIGH))     // Поворот направо
  {
   digitalWrite(DIR_R , HIGH);
    digitalWrite(SPEED_R ,LOW);
    digitalWrite(DIR_L , HIGH);
    digitalWrite(SPEED_L ,HIGH);

  }

  if(digitalRead(LS==HIGH) && (digitalRead(RS==LOW))     // Поворот влево
  {
    digitalWrite(DIR_R , HIGH);
    digitalWrite(SPEED_R ,HIGH);
    digitalWrite(DIR_L , HIGH);
    digitalWrite(SPEED_L ,LOW);
 }
    if(digitalRead(LS==LOW) && (digitalRead(RS==LOW))     // остановка
  { digitalWrite(DIR_R , HIGH);
    digitalWrite(SPEED_R ,HIGH);
    digitalWrite(DIR_L , HIGH);
    digitalWrite(SPEED_L ,HIGH);
 }
}
```

§ 29 Эпизод двадцать девять. Протокол связи – настоящее и будущее. Мультиагентное управление устройствами. 9 класс

Привязка к тематическому планированию по информатике: Коммуникационные технологии. Передача информации в современных системах связи.

Понятно, что роботы не должны бродить тоскливыми одиночками. Нормальный робот должен уметь общаться с другими роботами, или, на крайний случай, человек, должен составить для робота программу и каким-то образом и передать ее роботу (прошить устройство).

Т.е. для роботов крайне необходима возможность связи.

Способ научить разные устройства общаться друг с другом называется протоколом связи**.** Иначе говоря, протокол – это язык общения электронных устройств.

29.1 Как мы общаемся. Обмен информацией – люди, звери и роботы.

Скажи мне кто твой друг
И я скажу, кто ты.

Мы, люди, так или иначе, все общаемся между собой. Как это происходит? Пишем друг-другу сообщения по почте, шлем SMS-ки, оставляем записи на стене, ну и конечно, разговариваем по телефону и при личной встрече. Что мы каждый раз используем, чтобы понимать собеседника?

Совершенно верно — язык!

Язык это средство общения между людьми.

Роботы для обмена информацией между собой тоже общаются. Язык общения роботов – это протокол связи.

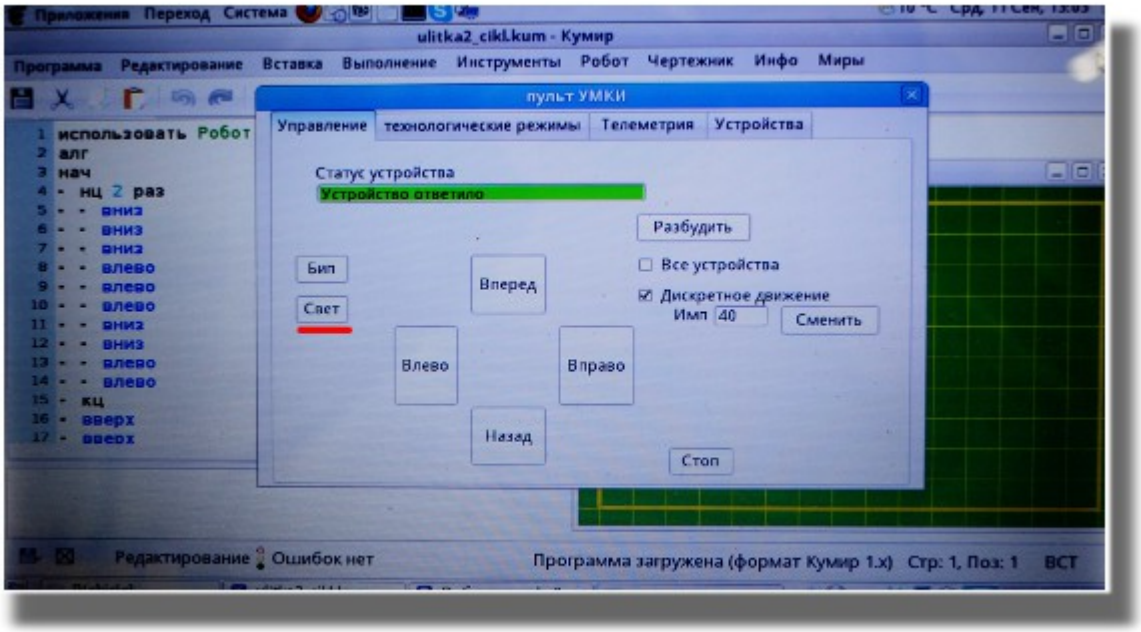
Люди, когда растут и начинают учиться говорить, узнают, что стол – звучит как «стол», а стул – звучит как «стул», так и роботам, чтобы понимать друг друга надо пользоваться заранее согласованными способами. Эти правила и называются протоколом связи.

Например, вы хотите, чтобы ваш друг-щенок по команде «Сидеть!» останавливался и садился в том месте, где пробегает, виляя хвостиком.

Для этого вы подаете команду – издаете горлом и языком определенные колебания воздуха, эти звуковые волны разносятся по всей округе, в зависимости от громкости поданной команды, и когда ваш умный щенок, своими ушами услышит эту команду, то скорее всего поймет, что ему надо выполнить некоторое действие: остановиться и сеть. За это он получит кусочек лакомства, причем, услышав вашу команду может сесть и другой пробегающий мимо, соседский пес, конечно, если он приучен к выполнению служебных команд.

Так же и у роботов. Допустим, вы хотите, чтобы уехавший в другой угол комнаты робот пискнул или подмигнул диодом. Для этого, вы должны со своего компьютера подать на него нужную команду, а робот должен понять и отреагировать на поданную команду. Таким образом, источник сигнала (ваш компьютер) и приемник сигнала (робот в углу) должны общаться по одному протоколу.

Этот сигнал вы формируете определенным образом у себя на ноутбуке. Допустим, в программе управления роботом вы нажимаете кнопку «свет».



При этом, ваша программа понимает, что в случае нажатия кнопки «свет» надо сформировать определенный пакет данных и отправить его в порт[2] на устройство. Какие устройства есть у вашего ноутбука или планшета? Это скорее всего встроенные Bluetooth, Wi-Fi или присоединенный по USB порту ZigBee модуль. Вот и получается, что сформированный программой пакет данных через ваше устройство улетает в эфир, в виде электромагнитных волн.

А в другом углу комнаты ваш робот находится в ожидании приема команд. И как только по радиоэфиру он получает соответствующую команду, то по своей программе, выполняет указанное действие.

Причем, если вы своего щенка приучали к команде «сидеть», которую говорили по русски, то вряд ли он будет вас слушать, если вы будете ему эту команду говорить на

2 Порты ввода-вывода являются основным средством связи микроконтроллеров с окружающим миром. Через эти порты микроконтроллер получает сигналы (аналоговые или цифровые) от внешних устройств; управляет или обменивается данными с внешними устройствами; сигнализирует о проделанной работе и т.д.

английском или на немецком языке. Так же и робот, если он укомплектован Bluetooth модулем, то никак не отреагирует на вашу команду отправленную по Wi-Fi. Способ отправки и приемки сообщений таким образом, чтобы он был распознан в обеих точках — мы договорились называть протоколом сообщений.

29.2 Разные уровни

Каждый протокол имеет несколько уровней. Уровни разделяются на низкие и высокие. А вместе это все называется – стек протокола. Так команда «сидеть» разделяется на низкий, физический уровень, это колебания воздуха, и высокий – логический это сама команда, по которой щенок должен именно сесть, а не лечь и не подать голос.

Бывает еще средний уровень — сетевой, в котором мы адресуем команду. Например, для конкретного щенка команда может прозвучать так: «Дружок, сидеть!».

Низкий физический уровень — это колебания электромагнитного поля. В России для ZigBee, Bluetooth и Wi-Fi, колебание происходит на частоте 2.4 гигагерца, получается что для этих протоколов — физический уровень одинаковый.

Различия возникают уже на сетевом уровне — когда устройства идентифицируются по МАК адресам. Этот уровень нам нужен чтобы адресовать нашу команду либо ко всем устройствам в сети, либо на какое-то конкретное устройство.

Самый высокий – это уровень приложений. Стандарт протокола обозначается буквами IEEE и цифрами, например, для протокола ZigBee – это 802.15.4

29.3 Зачем нужен стандарт

Он необходим для того, чтобы устройства изготовленные разными разработчиками в рамках одного протокола всегда могли связаться друг с другом и понимать команды отправляемые и принимаемые от других. Когда вы со своего мобильника присоединяетесь к точке Wi-Fi, то совершено не задумываетесь, каким производителем изготовлена эта самая точка-роутер. Обычному пользователю это и не должно быть актуальным. Но если вы хотите сами разрабатывать свои устройства, то без понимания на каком стандартном протоколе оно будет общаться с другими устройствами, запустить в серию вы его не сможете.

Иначе говоря, чтобы ваш робот помигал светодиодом, или попищал динамиком, вам надо отправить ему команду по соответствующему протоколу.

Команды могут быть краткими – АТ команды, или более расширеннми — это API фреймы. АТ команды в основном используются для протокола Bluetooth, а API фреймы для протоколов Wi-Fi и ZigBee.

Чем же различаются эти протоколы? Почему нельзя пользоваться каким то одним?

Дело в том, что исторически каждый протокол создавался под свои конкретные нужды.

Bluetooth необходим, чтобы связывать друг с другом на коротком расстоянии устройства. Например два мобильника для передачи файлов, либо наушники и планшет.

Протокол Wi-Fi гораздо более производительный, он с большей скоростью позволяет обмениваться данными между планшетом и ноутбуком, но обязательно использует для этого точку доступа.

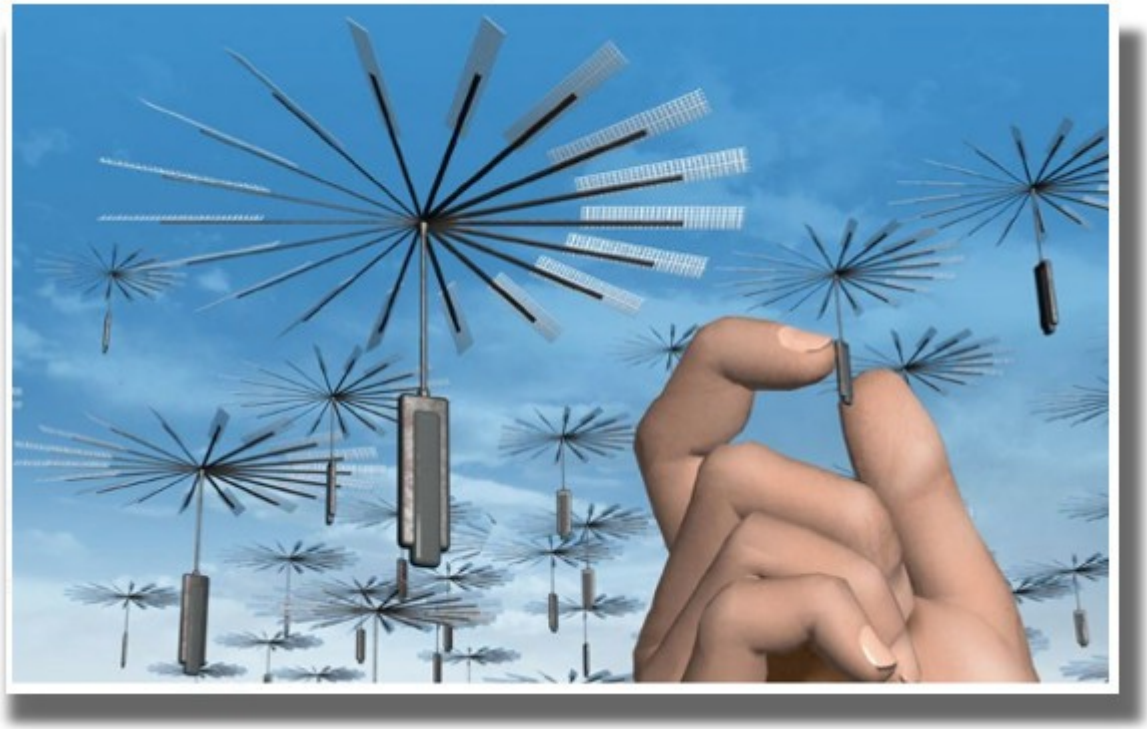
И самое главное – оба этих протокола очень сильно потребляют энергию даже в том режиме, когда ничего не делают, просто за счет накладных расходов :(

Архитектура связи у них у обоих — звезда. Это значит, что есть какая-то центральная точка доступа и все конечные устройства присоединяются к ней в зоне прямой видимости. Wi-Fi может построить сеть еще в архитектуре Дерево, когда есть возможность выстраивать сеть через роутеры — промежуточные точки доступа.

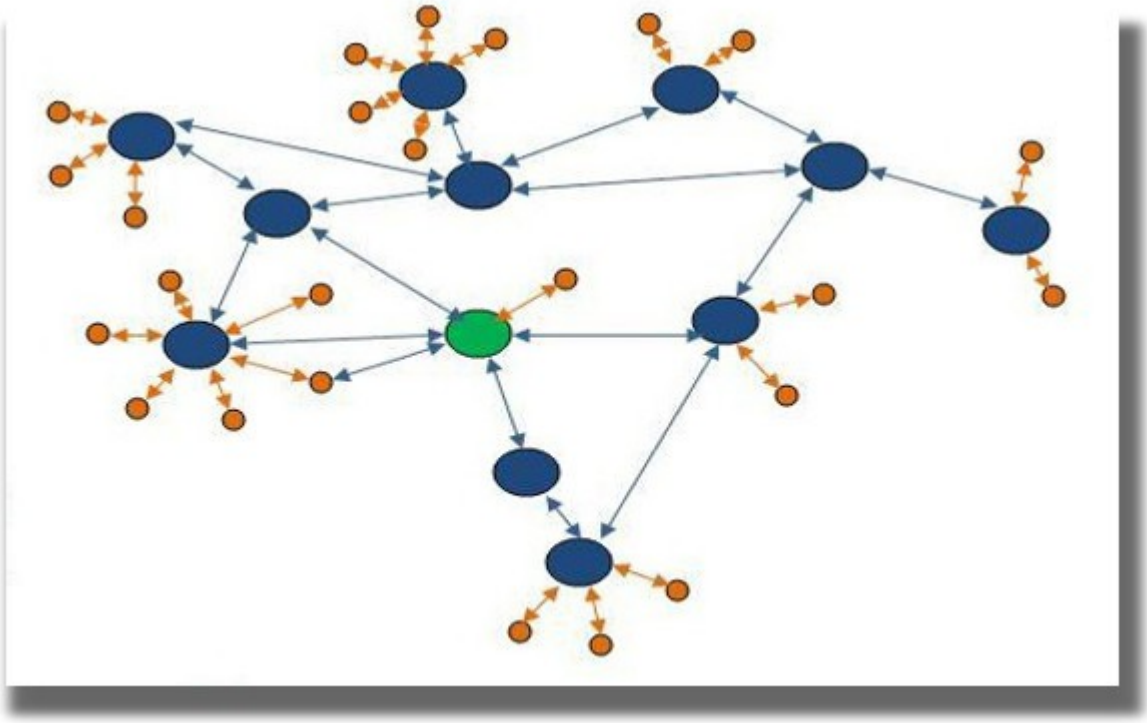
29.4 Все-таки они отличаются...

Третий протокол, который мы здесь рассматриваем — ZigBee, отличается тем, что он был изначально задуман, как протокол для малого энергопотребления, например, для небольших устройств с автономными источниками питания — батарейками.

Основной целью использования этого протокола было то, чтобы устройства, разбросанные случайным образом, могли сами соорганизоваться в сеть и просуществовать как можно дольше — собирая и передавая на центральную точку сбора информацию о событиях происходящих на исследуемой территории. Например, о замере температуры, или об освещенности, для осуществления так называемого экологического мониторинга. Большее время жизни для устройств в сети ZigBee достигается за счет того, что пока не происходит замер параметра, или пока не произошло какое-то событие, каждое устройство в сети спит и практически не потребляет энергию. Таких устройств в сети, иначе говоря Мотов, может быть очень много, они небольшие по размерам и иногда их называют пылинками (есть даже такой термин Smart Dust – умная пыль).



Еще одной отличительной особенностью сети ZigBee, является то, что моты могут организовываться в единую структуру не только по архитектуре Звезда или Дерево, как мы уже разобрали, но и в так называемую Меш-структуру (MESH), когда каждый мот общается не со своим старшим в группе, а с равноценным участником, которого он видит в зоне прямого доступа. И пакет данных внутри такой MESH-сети переходит от мота к моту распространяясь там лавинообразно. Это бывает очень удобно. Если у какого-то мота заканчивается энергия в батарейках и он выпадает из сети, то поток информации, который шел через него не рвется, а продолжает перетекать просто по другим узлам.



Но конечно такая уникальная возможность, проявлять необычно устойчивую «живучесть» всей сети, на основе прокола связи ZigBee, накладывает и некоторые весьма существенные ограничения. Так например, нужно разработать необходимые алгоритмы маршрутизации потоков данных в сети с учетом того, что некоторые узлы-моты могут спать или даже полностью выпадать из сети.

Конечно, некоторые протоколы уже разработаны и запатентованы, но работы еще предстоит много сделать в этом направлении. Так компания Гугл анонсировала в 2014 году новый способ связи мобильных устройств на основе MESH технологии, что позволит выходить в интернет без использования услуг провайдеров в сегодняшнем понимании, а только за счет того, что вы сможете отправлять и получать данные в глобальную сеть при помощи устройств, которые вас окружают: холодильников, кофеварок или мобильников других людей.

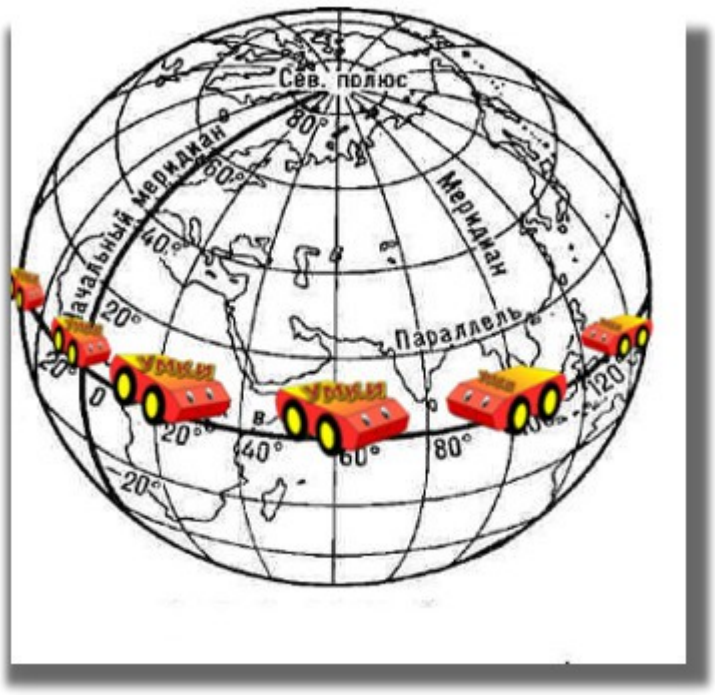
Иначе говоря, у протокола ZigBee — очень большие перспективы!

29.5 ZigBee и SmartCar

Протокол ZigBee был выбран в качестве базового для связи в УМКИ. На каждой умной машинке SmartCar — входящей в комплект лаборатории УМКИ, установлен модуль связи Xbee, который позволяет по протоколу ZigBee связывать множество машинок в одну единую сеть.

Дальность действия этого модуля может составлять один километр. Например, в качестве эксперимента, если машинки smartcar расставить на поверхности земли через один километр, и поставить их в ряд 40 тысяч штук, то последняя машинка окажется на расстоянии одного километра от первой.

Догадались почему?



Земля круглая, а длину окружности Земли по периметру, можно считать равной 40 тыс км.

И тогда, сидя в школе, вы можете заставлять мигать диодами или пищать динамиком наши SmartCar'ы все сразу, или по-очереди, либо по какому-то алгоритму, удивляя местных аборигенов.

А что же дальше? Резюме

Теперь, наверное, у вас огромное желание подробнее узнать то, о чем мы говорили в эпизодах, посвященных непосредственно робототехнике, при изучении информатики.

Наверняка, вам очень хочется самим конструировать роботы, которых можно учить двигаться, увидеть работу протокола ZigBee, управляющего роем робототехнических устройств, разработать собственную программу на одном из языков программирования.

Поэтому, мы идем к новым эпизодам... На очереди у нас модули подробнее рассматривающие работу с Arduino UNO, ArduinoNano и распределенными сенсорными сетями на основе процессораXbee, модуль знакомства с программированием для на языке Си++, знакомство с основами электроники и управлением контроллера.

Итак, вперед! Робототехника рулит!....