МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ»

«Робототехника для кадетских классов (10-11 класс)»

Москва 2023

ОГЛАВЛЕНИЕ

ТЕМА 1. ВВОДНОЕ ЗАНЯТИЕ. ИНСТРУКТАЖ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ

Занятие №1. Вводное занятие и инструктаж.

Интерес к робототехническим обучающим комплексам в образовательных учреждениях, клубах и домах детского творчества неуклонно с каждым годом растет. Работая с интерактивным оборудованием, кадеты учатся управлять виртуальными системами, вовлекаются в исследовательские проекты и творческие занятия, чтобы научиться изобретать, понимать и осваивать новое, выражая собственные мысли, принимать решения и помогать друг другу, формулировать интересы и осознавать возможности. Цель данного курса - обеспечение получения кадетами первичных знаний, навыков и умений по основам программирования и конструирования робототехнических комплексов, связанных с направлением предпрофессиональной подготовки ПВО СВ (Войска противовоздушной обороны Сухопутных войск) Вооруженных Сил Российской Федерации.

Но прежде, чем переходить к решению комплексной задачи, необходимо сначала разобраться, что это за наука такая – робототехника, какие знания, навыки и умения необходимы, чтобы стать специалистом по разработке роботов и их обслуживанию, и как робототехника сможет помочь в освоении профессии военно-прикладной направленности.

Войска ПВО СВ – род войск Сухопутных войск, предназначенный для прикрытия войск и объектов от действий средств воздушного нападения противника. Одна из основных задач, возлагаемых на войска ПВО СВ – это ведение разведки воздушного противника радиолокационными методами. Эти методы используются для определения расстояния до воздушного объекта и его скорости движения.

С помощью робототехнического комплекса, созданного на базе платформы Arduino появилась возможность объединить демонстрационный эксперимент и лабораторную работу для имитации радиолокационных методов по определению дальности и скорости объектов с помощью ультразвуковых датчиков. Этот демонстрационный эксперимент в виде лабораторной работы по робототехнике приближает понимание, как предназначения, так и устройства радиолокационных средств, используемых в войсках ПВО СВ.

Повышение компетентности в таких сферах как: программирование, конструирование, робототехника и работа радиолокационных приборов,

способствует в дальнейшем осознанному выбору профессии государственного служащего и поступления в высшее учебное заведение, ориентированное на подготовку специалистов подразделений войсковой ПВО ВС РФ.

Работа с Arduino может быть интересным и познавательным опытом для школьников старших классов. Однако, важно обеспечить безопасность при работе с этой технологией. Вот несколько мер безопасности, которые следует соблюдать при работе с Arduino:

1. Работайте с Arduino под руководством опытного учителя или инструктора. Это поможет вам получить необходимую поддержку и руководство в случае возникновения проблем или вопросов.
2. Используйте правильные компоненты и материалы. Убедитесь, что вы используете подходящие компоненты, соответствующие вашему проекту. Неправильное подключение или использование некачественных компонентов может привести к нестабильной работе или даже повреждению Arduino.
3. Правильно подключайте и отключайте Arduino. Перед подключением или отключением компонентов, убедитесь, что питание Arduino выключено. Это поможет избежать возможности повреждения компонентов или Arduino при неправильном подключении.
4. Питание и электрическая безопасность. При работе с Arduino используйте правильные и надежные источники питания, соответствующие требованиям Arduino и подключаемых компонентов. Не используйте поврежденные кабели или адаптеры. Также избегайте работать с Arduino во время грозы или во влажных условиях, чтобы предотвратить возможность возникновения короткого замыкания или поражения электрическим током.
5. Защита от перегрева. У Arduino есть пределы по току, напряжению и теплу, которые следует учитывать. При работе с высокими токами или интенсивной нагрузке, обратите внимание на нагрев Arduino и его окружающих компонентов. Убедитесь, что устройство достаточно охлаждается и избегайте покрытия Arduino материалами, которые могут удерживать тепло.
6. Защита от короткого замыкания. Правильно подключайте провода и компоненты, чтобы избежать короткого замыкания. Убедитесь, что провода не перекрещиваются и не соприкасаются друг с другом без

необходимости. Используйте изоляционную ленту или проводной кожух, чтобы защитить подключения.

ТЕМА 2. ВВЕДЕНИЕ В РОБОТОТЕХНИКУ. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ В ОБЛАСТИ РОБОТОТЕХНИЧЕСКИХ КОМПЛЕКСОВ.

Занятие №2. Основы электроники.

Электроника (от греч. Ηλεκτρόνιο - электрон) — область науки и техники, занимающаяся созданием и практическим использованием различных устройств и приборов, работа которых основана на изменении концентрации и перемещении заряженных частиц (электронов) в вакууме, газе или твердых кристаллических телах, и других физических явлениях.

В робототехнике знания по электронике необходимы, чтобы понимать функционирование:

- электронных компонентов;
- аналоговых схем;
- цифровой логики;
- микроконтроллеров.

Электроника – это то, без чего в современном мире при производстве робототехнического устройства практически невозможно обойтись, если только не планируется создать полностью механического робота или использовать пневматику для управления.

При создании робототехнического изделия, как правило, работа ведется с цепями постоянного тока. Напомним основные физические величины, которыми оперирует электроника и схемотехника.

Заряд q [Кл] — это физическая скалярная величина, определяющая способность тел быть источником электромагнитных полей и принимать участие в электромагнитном взаимодействии.

Сила тока I [A] — отношение протекающего через проводник заряда q [Кл] ко времени t [c].

Напряжение U [В] — скалярная физическая величина, значение которой равно работе эффективного (включающего сторонние заряды) электрического поля, совершаемой при переносе единичного пробного электрического заряда из одной точки в другую.

Сопротивление R [Ом] — физическая величина, которая показывает свойство проводника противодействовать прохождению через него электрического тока.

Закон Ома для участка цепи связывает три этих параметра. Помимо силы тока, сопротивления и напряжения, напомним про емкость, заряд и индуктивность.

Электрическая емкость C [Ф] — величина, характеризующая способность тела накапливать электрический заряд.

Индуктивность L [Гн] — это физическая величина, характеризующая магнитные свойства электрической цепи. В некоторых источниках её называют коэффициентом самоиндукции, так как она зависит от текущего в замкнутом контуре тока и создаваемого им магнитного потока.

Базовые электронные компоненты.

Переходим к изучению базовых электронных компонентов. Рассмотрим группы базовых электронных компонентов. Деление на группы осуществляется исходя из принципов применения и строения. Сначала рассмотрим блок, включающий в себя все возможные переключатели и кнопки.

Кнопка — компонент, в основе того лежит принцип размыкания и замыкания электрической цепи при ее нажатии. Это обеспечивается за счет наличия двух не соединенных между собой проводников. При нажатии на кнопку третий проводник замыкает цепь с первыми двумя. После нажатия кнопки он сам поднимается с помощью пружины в исходное положение.

Выключатель отличается от кнопки главным образом тем, что при одном нажатии фиксирует свое положение, а не возвращается в исходное. Замыкание происходит в результате сдвига проводника, замыкающего эти контакты.

Переключатель может находиться сразу в трех положениях. Работает так же, как выключатель, но имеет больше неподвижных контактов, и свободный проводник может замыкать поочередно обе группы, либо находиться в разомкнутом положении.

Перейдем к рассмотрению *резистивных компонентов*. Их главная задача – контроль электрического тока, протекающего в цепи. К резистивным компонентам относятся и датчики, сопротивление которых каким-либо образом зависит от параметров окружающей среды.

Резистор – это элемент электрической цепи, имеющий определенную постоянную величину сопротивления, предназначенный для регулировки параметров цепи согласно законам Ома.

Потенциометр – компонент на базе резистора, который позволяет менять напряжение на участке цепи из-за изменения параметров внутреннего проводника. Если определения недостаточно для успешной работы с данным компонентом, то для полного понимания существует много упрощенных аналогий. Например, к этой же категории компонентов относится и *реостат*, который по сути является составной частью потенциометра. Он имеет два контакта в отличие от потенциометра, и сопротивление между ними меняется при передвижении ползунка. *Фоторезистор* меняет сопротивление в зависимости от яркости падающего на чувствительный элемент света, *термистор*, соответственно, от температуры.

Разберем методы световой индикации. Раньше для световой индикации чаще всего использовались простые лампы накаливания. Внутри корпуса лампы расположена нить накала, при пропускании тока через которую она раскаляется и начинает излучать свет. В настоящее время такой источник освещения популярности не имеет.

На сегодняшний день гораздо чаще используются светодиоды. *Светодиод* – полупроводниковый элемент, в котором при прохождении электрического тока возникает электромагнитное излучение видимого спектра. Он обладает полярностью, то есть, порядок подключения проводов питания и земли в данном случае важен. Как правило, у светодиода разные длины контактов, более длинный подключается к “плюсу”. Для ограничения величины тока, который будет протекать через светодиод (это обусловлено его строением и параметрами) необходимо рассчитать необходимое сопротивление ограничивающего резистора и подключить его.

RGB-светодиод — это популярная разновидность светодиода, небольшое устройство, которое объединяет в себе три диода базовых цветов: синий, красный и зеленый. Светодиод имеет четыре контакта, которые позволяют управлять цветом свечения. Контакты бывают различных форм и размеров в зависимости от модели светодиода, но обычно их можно различить по цвету.

Например, подавая высокий уровень напряжения на все управляющие контакты, получаем белый цвет.

Один контакт управляет красным цветом, другой - зеленым, а третий контакт - синим цветом. Черный контакт - это контакт земли или общий контакт, который обычно подключается к нулевому потенциалу и является общей точкой для всех других контактов. Если на светодиод подать напряжение через контакты питания и заземлить контакт “земли”, светодиод начнет светиться. А если на контакты управления цветом подать правильную комбинацию напряжений, то свечение будет соответствовать выбранному цвету.

Еще один индикаторный элемент – семисегментный индикатор. Его достаточно удобно использовать для вывода информации в числовом формате: показания датчиков, время, количество итераций и так далее.

Семисегментный индикатор состоит из 8 отделенных друг от друга областей, они используются для поэтапного отображения цифр, дополнительно к ним идет значок точки. Каждый сегмент – это отдельный диод, который можно программировать самостоятельно и подключать независимо. Компонент имеет десять выходов, как и RGB-светодиод, общую “землю” и отдельные подводы питания для каждого из сегментов. Отличие в том, что выходов для “земли” два - по одному на четыре контакта питания.

Помимо перечисленного набора элементов существует большое количество других вариантов индикации, но, тем не менее, основные перечислены выше.

Продолжим пошаговое изучение базовых электронным компонентов – рассмотрим транзистор. *Транзистор* – электронный ключ, позволяющий с помощью подачи небольших токов на один из трех контактов разрешать протекание тока и управлять более большими величине токами на остальных компонентах схемы. Например, такое устройство необходимо для работы с мощными светодиодами, которые управляются с помощью микроконтроллера. Контакты транзистора имеют названия: база, коллектор и эмиттер. При подаче управляющего сигнала на контакт базы между коллектором и эмиттером начинает протекать ток. Транзистор, как базовое устройство, может быть модифицирован в зависимости от специфики использования.

Конденсатор – устройство, способное накапливать электрический заряд и мгновенно его отдавать. Используется для построения цепей с частотно зависимыми свойствами (в таких цепях или компонентах электрическое поведение, такое как сопротивление, емкость или индуктивность, может

изменяться с изменением частоты сигнала) фильтров, цепей обратной связи, колебательных контуров и проч. Возможность быстрого разряда конденсатора позволяет получить импульс большой мощности, который применяется, например, в лазерной технике. Кроме того, в электронных устройствах помимо функции хранения электроэнергии конденсаторы используют как элементы памяти. В промышленной электротехнике конденсаторы используются для компенсации реактивной мощности. Это означает, что конденсаторы помогают уравновесить реактивную мощность в электрических системах, улучшая эффективность использования электроэнергии и обеспечивая стабильную работу системы. Конденсаторы также используются в фильтрах высших гармоник. Гармоники - это нежелательные периодические колебания или искажения в электрической системе. Фильтры высших гармоник с помощью конденсаторов удаляют или снижают влияние этих нежелательных гармоник, обеспечивая более чистое электрическое питание для других устройств и оборудования. Кроме того, конденсаторы могут использоваться как датчики малых перемещений. Это означает, что они могут реагировать на изменение расстояния между своими обкладками. Даже небольшие изменения в этом расстоянии могут существенно изменить емкость конденсатора. Это свойство конденсаторов используется в различных приложениях, например, для измерения давления, деформации или других физических величин.

Основные компоненты электрических цепей перечислены. Рассмотрим основные связки электронных компонентов, которые достаточно часто встречаются в робототехнике.

Схемы с резистором.

Основное применение резистора – ограничение тока в цепи. Но у резистора гораздо больше применений. Самый яркий пример – подключение перед входом светодиода, чтобы ограничить проходящий ток и не повредить элемент.

Подтягивающий резистор – это резистор, который подключается между сигнальной линией и источником питания (обычно напряжением +5 В), чтобы обеспечить определенное начальное состояние сигнала. Он используется в электронике, особенно при работе с цифровыми сигналами. Подтягивающий резистор "подтягивает" сигнальную линию к источнику питания, обеспечивая устойчивое напряжение на линии в отсутствие других активных сигналов. Зачем он нужен? Подтягивающие резисторы помогают предотвратить "плавающее" состояние сигнала, когда сигнальная линия не активна или не связана с каким-либо устройством. Без подтягивающего резистора, сигнальная линия может быть

неопределенной и легко подвержена помехам. В случае нахождения между проводником и “землей” говорят, что *резистор стягивающий*. Он нужен для обеспечения низкого уровня на логическом входе в изложенных выше случаях.

Любой логический вход имеет ёмкость относительно земли. Если сигнал формируется на открытом выводе ключевого элемента, то чем выше сопротивление подтягивающего резистора, тем больше время нарастания или спада сигнала при размыкании ключевого элемента. Время спада или нарастания — это время между размыканием ключа и достижением сигнала порогового напряжения.

Пороговое напряжение – это напряжение, при достижении которого логическим входом фиксируется изменение логического состояния. При проектировании логических схем приходится рассчитывать сопротивление подтягивающего резистора, при этом известны ёмкость входа и пороговое напряжение. Время спада или нарастания пропорционально сопротивлению подтягивающего резистора, то есть, например, при увеличении сопротивления вдвое – время спада или нарастания также увеличится вдвое.

Следующая базовая схема совместной работы устройств – *матричное подключение*. Это могут быть матрицы светодиодов или любая клавиатура. Под таким соединением принято понимать расположение компонентов в порядке строк и столбцов, где, например, ряд элементов имеет один набор общих контактов, а столбец – другой. Этот способ позволяет существенно сократить необходимое количество контактов для подключения многоэлементных систем.

На данном занятии были даны основные термины и определения электроники, рассмотрены базовые электронные компоненты и наиболее часто используемые “связки” электронных компонентов. Также изученная тема позволила получить необходимую вводную информацию для более детального разбора области робототехники.

Занятие №3. Основы робототехники

Мы уже познакомились с основными понятиями в робототехнике и узнали, какие основные разделы она в себя включает. Для дальнейшего обучения полезно ознакомиться с основными программами, которые можно использовать как для обучения, так и для дальнейшей работы в сфере робототехники.

Робототехника основывается на четырех основных направлениях - электроника, программирование, инженерия и 3D-моделирование. Мы рассмотрим программы для каждого из перечисленных направлений.

Предложенное программное обеспечение (ПО) можно либо бесплатно установить, либо использовать бесплатные учебные или сокращенные версии.

Tinkercad.

Для базового обучения электронике, программированию (и, при желании, 3D моделированию) очень удобно использовать такой сервис как Tinkercad. Данная программа является бесплатной и позволяет быстро начать обучение, так как не требует установки. Всё, что нужно, — это пройти регистрацию на сайте https://www.tinkercad.com. Tinkercad обладает хорошим функционалом и встроенными уроками по электронике и программированию.

По данным урокам и заранее сделанным проектам можно самостоятельно начать осваивать азы электроники и программирования. В Tinkercad существует несколько видов программирования. Во-первых, это обычный текстовый формат кода на Си-подобном языке. Во-вторых, это блочное программирование, которое позволяет создавать базовые программы, не прибегая к написанию текстового кода. Это дает возможность на начальных этапах быстро получать готовые схемы, устройства без необходимости разбираться в синтаксисе, что значительно увеличивает скорость работы.

Однако, набор блоков, из которых можно написать программу, недостаточно большой. Поэтому нет возможности создать программы для подключения некоторых датчиков, экранов и т.п., но для начальных этапов обучения данного ПО вполне достаточно. Помимо этого, есть промежуточный вариант, в котором программа представлена сразу в виде блоков и текста. Этот формат позволяет сделать плавный переход от начального обучения написанию программ в виде блоков к формату, который применим в реальных разработках, при этом понимание структуры программы, работы циклов, логические операции не нужно будет осмысливать заново.

Tinkercad позволяет не только заниматься самостоятельно, но и работать в составе класса с преподавателем. Помимо этого, существует еще два раздела, которые позволяют строить 3D модели, однако, с точки зрения обучения проектированию и 3D моделированию, не рекомендуется использовать данное ПО, так как принципы построения модели отличаются от большинства общепринятых и используемых в других, более профессиональных системах проектирования. Для обучения 3D моделированию рекомендуется использовать иные программы, которые будут приведены далее.

Arduino IDE.

Основное ПО, которое используется для программирования микроконтроллера Arduino - Arduino IDE. Данное ПО является и первым ПО, которое появилось для этих целей, и, несомненно, самым популярным. Его можно использовать как для обучения, так ив профессиональной деятельности. Arduino IDE специально разработана для написания программ для Arduino и однотипные контроллеры.

Язык разработки является Си-подобным. Данная программа обладает широким спектром специализированных библиотек для подключения различных модулей, что позволяет достаточно просто начать писать программы для сложных компонентов, глубоко не вникая в принцип их работы.

Достоинством использования данного ПО является то, что существует множество уроков, примеров написания программ в Arduino IDE. Также присутствуют встроенные образцы с подробными комментариями разработчиков. Обычно данные комментарии написаны на английском языке, поэтому для работы с Arduino, и, как правило, для программирования с использования других платформ, является значительным преимуществом, а иногда, и необходимостью - знание английского языка или уверенное пользование переводчиком.

Кроме перечисленных функций, существует ряд программ, которые позволяют писать код уже под более серьезные проекты, основанные на контроллерах семейства STM 32. Нужно отметить, что данные контроллеры являются более сложными для освоения и имеют достаточно высокие требования к знаниям разработчиков. Поэтому не рекомендуется приступать к их изучению пока нет понимания Arduino на достаточно высоком уровне.

В данном курсе приведена лишь краткая информация о работе с подобными контроллерами, с целью получения общего представления об их возможностях и о перспективах дальнейшего, более глубокого, их изучения.

Для работы с STM можно использовать различные виды ПО и их комбинации. Например, CubeMX и Keil, либо же CubeMX и Cude IDE, а также и Keil, Cube IDE по-отдельности.

На первом этапе изучения необходимо разобраться с самим принципом работы программы. В отличии от Arduino, при работе с STM у разработчика имеется значительно больше возможностей по настройке и работе с тем или иным контроллером. Это, соответственно, усложняет процесс работы. В случае

с Arduino, все “пины” (входы платы, на которые можно подключать устройства) заранее разделены на группы: цифровые, аналоговые и поддерживающие ШИМ (широтно-импульсная модуляция). В случае с STM, инициализация, то есть определение типа входов-выходов, производится каждый раз во время написания кода. Следующие программы могут существенно упростить процесс инициализации для разработчиков.

CubeMX.

Позволяет с помощью графического интерфейса произвести инициализацию. После того, как собран нужный вариант “распиновки” (то есть, сочетания нужных входов платы), программа автоматически формирует основу, в которой уже созданы все “пины” в соответствии со схемой. После этого написание кода становится сопоставимым с написание программы для Arduino.

Для написания программ под микроконтроллеры STM чаще всего используется язык Си. Однако, в отличие от Arduino, существует и множество других достаточно популярных языков программирования, на которых можно писать код под контроллеры серии STM. Можно выбрать язык, который будет более удобен в использовании.

Рассмотрим программы, которые используются для разработки электроники, составления электронных схем. Для базового освоения удобно использовать упомянутый ранее Tinkercad. Данный сервис позволяет составлять несложные электронные схемы, как с использованием микроконтроллеров, так и без. В этом случае, основным достоинством является простота работы. В отличие от других программ, которые будут рассмотрены далее, все электронные компоненты представлены в виде рисунков самих компонентов, а не условных обозначений. Это упрощает разработку. Данную программу рекомендуется использовать именно для начального обучения. Сервис Tinkercad содержит заранее созданные схемы, которые расположены в специальной последовательности и позволяют пройти мини курс обучения уже в самой программе.

Интересной практикой является проектирование и создание собственных плат. Для того, чтобы создавать платы и их принципиальные схемы, можно использовать различные программы. Рассмотрим некоторые из них.

Fritzing.

Одной из наиболее удобных программ для создания своих собственных плат и схем является Fritzing. В ней достаточно просто реализован переход от работы

с платами, как с изображениями самих компонентов, к условным обозначениям, принятым в электронике. В программе Fritzing можно собирать схемы по аналогии со схемами в Tinkercad, однако разнообразие компонентов, позволяет создавать более сложные схемы.

После этапа составления схемы возможно перейти к отображению этой схемы в виде условных обозначений. Помимо этого, существует вариант отображения в виде специальных файлов, которые можно использовать для создания печатной платы. При этом все три вида объединены в один проект, что позволяет производить редактирование схемы в любом из видов и получать результат сразу во всех отображениях. Данное ПО отлично подходит как для обучения, так и выполнения проектов любого уровня.

EasyEDA.

EasyEDA является бесплатной программой, обладает большим набором компонентов. Кроме того, существует возможность перенести в EasyEDA проект из другой программы для создания принципиальных схем и отредактировать его. Этот подход позволяет проводить совместную разработку схемы, даже с использованием различного ПО. Несомненным достоинством является то, что программа является облачной, то есть, разработку можно вести браузере. Программа содержит сеть встроенных уроков, которые позволяют быстро ее освоить и создавать свои собственные проекты.

Помимо программ, которые перечислены выше, есть еще множество ПО, позволяющих создавать те или иные разработки. В данном разделе в качестве примеров приведены те программные продукты, которые подходят прежде всего для обучения. Помимо них можно рассматривать Altium designer, TinyCad, ZenitPCB и другие. В целом, не так важен выбор программного обеспечения, как знание основных принципов работы в исследуемой области

Для создания механики изделия используются такие программы, как САПРы (системы автоматизированного проектирования). САПР — это программа, которая позволяет создать 3D модель, сборку и чертеж заданного изделия. Данное ПО также позволяет выполнить расчеты (например, расчет прочности изделия). Помимо этого, некоторые программы обладают специальными модулями для подготовки управляющей программы для фрезерного, токарного станка с ЧПУ (числовое программное управление), либо же для 3D принтера.

Перечислим основные программы, использующиеся для 3D моделирования.

Tinkercad также имеет модули и для создания 3D моделей. В программе есть два модуля для этих целей. Первый — “Блоки кода’’, где разработка 3D моделей производится за счет соединения различных блоков. В качестве блоков представлены геометрические фигуры (параллелепипед, конус, пирамида), а также блоки по изменению геометрии (объединение, разделение, перемещение). Навыки работы с данным модулем не пригодятся в дальнейшем при работе с более профессиональным ПО, но удобны для написания программ базового уровня.

Структура построения модели аналогична структуре написания программы. В Tinkercad есть модуль, позволяющий работать с 3D моделями с помощью базовых форм и различных операций над ними. Однако, процесс их создания сильно отличается от процесса создания моделей с помощью первого модуля. Данный модуль можно использовать для начального обучения и для быстрого создания несложным форм.

Компас-3D.

Программа позволяет выполнять весь цикл разработки какой-либо конструкции — от создания детали, сборки, до чертежей и спецификаций. Компас-3D — это программа, созданная в России, поэтому она адаптирована под нормы ЕСКД (единая система конструкторской документации — объединение нормативных документов, регулирующих порядок разработки различных изделий) и позволяет создавать чертежи и спецификации намного удобнее, чем аналогичное ПО от иностранных производителей.

Хотя существуют недостатки. Например, при работе со сложными элементами, имеющими множество поверхностей, либо же со сборками, состоящими из большого количества компонентов, программа может перегружаться и экстренно завершать работу. Следует отметить, что данная ситуация характерна для многих ресурсоемких программных продуктов.

Программа имеет встроенные уроки Азбука Компас, что позволяет достаточно быстро научиться ей пользоваться. Разработкой ПО занимается отечественная компания Аскон, что означает наличие понятного русскоязычного интерфейса и множества статей по его использованию.

Autodesk Inventor.

Рассмотрим программу со схожими параметрами. Она, как и Компас 3D, позволяет создавать детали, сборки, оформлять чертежи и спецификации.

Программа иностранная, но при этом существуют версии, позволяющие делать оформление по ЕСКД, хотя и не совсем эталонное.

Все недостатки при необходимости можно исправить вручную. Преимуществом данной программы является ее оптимизация, возникает меньше проблем при создании сложных объектов.

Данное ПО обладает тем же функционалом, что и Компас 3D, однако, есть еще и ряд дополнительных модулей, которые позволяют дополнить создание модели еще и расчетами.

Есть и другой способ решения проблемы с оформлением чертежей по ЕСКД. Для моделирования используют Autodesk Inventor либо SolidWorks (еще одна CAD-система), а для оформления чертежей и остальной документации применяют Компас 3D.

Ardublok.

Для начального обучения программированию возможно использование Ardublok. Данная программа представлена в двух версиях. Как и в случае с Tinkercad, существует версия, которая позволяет работать онлайн. Также, при необходимости, ее можно установить на компьютер.

Данное ПО дает возможность использовать простое программирование в виде блоков кода для построения программы, которая далее будет переведена в формате обычного текстового кода. Текстовый код, в свою очередь, можно загрузить в обычную схему или в виртуальную в Tinkercad.

Несмотря на наличие некоторых проблемных аспектов, удобство использования Ardublok и обилие блоков для различных специализированных модулей позволяют создавать любые, даже достаточно сложные программы. Такой подход даёт возможность быстро обучиться структурировано писать программный код, понять основные принципы и закономерности работы программ, не отвлекаясь на написание текста.

Fusion 360.

Данная программа также является продуктом компании Autodesk. Она позволяет создавать модели, сборки и чертежи, но принцип построения отличается от многих аналогов.

Принцип построения модели более прост и понятен, поэтому данная программа подойдет для базового обучения 3D моделированию. При этом

многие продолжают использовать ее и при дальнейшей работе, так как программа предназначена и для профессионалов.

Вышеперечисленные программы имеют сходные принципы создания и обработки моделей, поэтому решение по поводу того, что именно использовать в работе, остается за пользователем. Для наглядности многие примеры в этом пособии будут разобраны на базе данных программ.

Представление о различных вариантах использования программного обеспечения позволит получать дальнейшие знания в области робототехники.

Занятие № 4. Знакомство с платформой Arduino

Arduino - это открытая платформа для разработки электронных устройств и прототипирования, которая позволяет создавать проекты, которые могут быть контролируемы и запрограммированы с помощью программного обеспечения Arduino IDE (интегрированная среда разработки). Платформа состоит из микроконтроллера, набора ввода/вывода (например, цифровых и аналоговых входов и выходов), а также средства подключения к компьютеру для загрузки программы.

Arduino позволяет создавать различные устройства - от простых светодиодных “мигалок” до сложных систем автоматизации и управления роботами. Она предоставляет множество возможностей для создания электронных устройств, включая подключение различных датчиков и модулей, управление моторами и сервоприводами, работу с LCD-экранами, Ethernet-соединениями и беспроводными модулями связи, такими как Bluetooth и Wi-Fi. На рисунке 6.1. изображены различные типы плат компании Arduino.

Arduino также имеет большое сообщество пользователей и разработчиков, которые делятся своими знаниями и опытом, что делает платформу еще более доступной и легко используемой. Кроме того, Arduino имеет открытый исходный код, что позволяет любому желающему внести свой вклад в развитие платформы. С помощью Arduino можно создавать не только прототипы, но и коммерческие продукты, что делает ее незаменимой для электронных разработчиков и инженеров.

Существуют множество проектов, которые создаются с помощью Arduino, и многие из них доступны для свободного скачивания и использования.

Кроме того, Arduino имеет открытый исходный код, что позволяет любому желающему внести свой вклад в развитие платформы. С помощью Arduino

можно создавать не только прототипы, но и коммерческие продукты, что делает ее незаменимой для электронных разработчиков и инженеров.

На рисунке 6.1 представлены все печатные платы, выпущенные компанией Arduino.

Рис. 6.1. Различные типы плат компании Arduino

Открытый исходный код Arduino позволяет любому желающему внести свой вклад в развитие платформы. С помощью Arduino можно создавать не только прототипы, но и коммерческие продукты, что делает ее незаменимой для электронных разработчиков и инженеров.

Платформа Arduino относительно недорогая и доступна для широкого круга людей, что делает ее популярной среди любителей и профессионалов в области электроники.

Несмотря на все преимущества, у платформы Arduino есть и некоторые ограничения. Например, она не подходит для создания крупных и сложных проектов, требующих большого объема памяти или процессорной мощности. Однако для многих начинающих разработчиков Arduino является отличным выбором.

На этом занятии более подробно была представлена платформа Ардуино, что даст возможность создавать программы управления для роботов.

Занятие №5. Сборка электрических схем

Сборка электрических схем - это процесс соединения различных компонентов и элементов электрической схемы в единое целое, чтобы она могла работать в соответствии с заданными параметрами.

На этом занятии мы рассмотрим процесс сборки электрических схем, включая выбор компонентов, размещение их на плате, проводку и проверку работоспособности.

Выбор компонентов.

Первый шаг в сборке электрической схемы - выбор компонентов. Компоненты могут быть различных типов, например, резисторы, конденсаторы, диоды, транзисторы, интегральные схемы и т.д. При выборе компонентов необходимо учитывать требования к их параметрам, таким как номинальное значение, допустимое отклонение, мощность, рабочее напряжение, частотный диапазон и т.д. Также важно учитывать доступность компонентов и их стоимость.

Размещение компонентов на плате.

После выбора компонентов необходимо разместить их на плате. При этом следует учитывать следующие факторы:

- расположение компонентов на плате должно быть оптимальным для обеспечения лучшей работоспособности электрической схемы;
- компоненты должны быть размещены на достаточном расстоянии друг от друга, чтобы избежать их взаимных помех;
- должно быть обеспечено правильное расположение выводов компонентов для удобства их соединения проводами.

Проводка.

После размещения компонентов на плате следует выполнить проводку, т.е. соединение компонентов между собой проводами. При проводке необходимо учитывать следующие факторы:

- провода должны быть выполнены из материалов с низким сопротивлением и высокой электрической изоляцией;
- должно быть обеспечено правильное соединение компонентов между собой и с источником питания

- провода должны быть размещены на достаточном расстоянии друг от друга, чтобы избежать помех.

Проверка работоспособности.

После завершения сборки электрической схемы следует проверить ее работоспособность. Это включает в себя проверку на соответствие электрической схемы заданным параметрам и функциональным требованиям.

Для этого могут быть использованы различные методы, например, измерение напряжения и тока на ключевых точках схемы, использование осциллографа для анализа формы сигнала, проверка времени задержки и многие другие.

Если в процессе проверки выявляются ошибки или несоответствия, то требуется внести корректировки в схему, поправить проводку или заменить компоненты. После этого проверка работоспособности должна быть повторена.

Оформление.

После успешной проверки работоспособности электрической схемы следует её оформление. Это включает в себя фиксацию положения компонентов на плате, закрепление проводов, установку клеммных колодок или разъемов для подключения внешних устройств и т.д.

Документация.

Наконец, важным этапом в процессе сборки электрических схем является составление документации, которая включает в себя схему подключения, список используемых компонентов, спецификацию их параметров и другую необходимую информацию. Это позволяет упростить процесс обслуживания и ремонта схемы в будущем.

Существует множество типов электрических схем, которые используются для различных целей и задач. Рассмотрим некоторые из наиболее распространенных типов электрических схем.

● Однополюсник - это наиболее простая электрическая схема, которая состоит из одного элемента и используется для соединения источника питания с потребителем. Например, это может быть лампочка, резистор или конденсатор. Однополюсники могут быть использованы для создания простых электрических устройств, таких как фонарик или мигалка.

● Двуполюсник - это электрическая схема, которая состоит из двух элементов, соединенных последовательно или параллельно. Например, это

может быть пара резисторов, светодиодов или конденсаторов. Двуполюсники могут использоваться для создания простых фильтров, усилителей и других устройств.

- Мостовая схема - это электрическая схема, которая используется для измерения переменного тока или переменного напряжения. Она состоит из четырех диодов, соединенных в определенной конфигурации. Мостовые схемы могут использоваться для создания диодных мостов, которые используются для выпрямления переменного тока.

- Усилитель - это электрическая схема, которая увеличивает амплитуду сигнала. Она состоит из нескольких элементов, таких как транзисторы, операционные усилители, конденсаторы и резисторы. Усилители могут использоваться для усиления звука, сигналов радиовещания и других видов сигналов.

- Импульсная схема - это электрическая схема, которая генерирует импульсный сигнал. Она используется в электронике для управления моторами, светодиодами и другими устройствами. Импульсные схемы могут быть использованы для создания плавного управления скоростью вентиляторов или для управления светодиодами в RGB-подсветке.

- Логическая схема - это электрическая схема, которая используется для обработки логических операций. Она состоит из логических элементов, таких как И-НЕ, ИЛИ-НЕ и других, которые выполняют определенные операции над входными сигналами. Логические схемы могут быть использованы для создания компьютеров и других устройств, которые основаны на цифровых сигналах.

- Регулятор - это электрическая схема, которая используется для поддержания постоянного значения величины, такой как напряжение или ток. Она состоит из нескольких элементов, таких как операционный усилитель, резисторы и конденсаторы. Регуляторы могут быть использованы для стабилизации напряжения питания, управления яркостью светодиодов и других устройств.

- Преобразователь - это электрическая схема, которая используется для преобразования электрической энергии из одной формы в другую. Например, это может быть преобразование переменного тока в постоянный ток или преобразование высокого напряжения в низкое напряжение. Преобразователи

могут быть использованы во многих устройствах, таких как инверторы, зарядные устройства и другие.

Выбор конкретного типа электрической схемы зависит от задачи, которую необходимо решить. При сборке электрических схем необходимо учитывать множество факторов, таких как электрические параметры элементов, соединения, допустимые токи и напряжения и другие. Правильный выбор и сборка электрической схемы существенно влияют на ее работоспособность и надежность.

В заключении можно отметить, что сборка электрических схем является важным этапом в процессе создания электронных устройств и систем. Для успешной сборки требуется не только технические знания и навыки, но и аккуратность, внимательность и терпение. Кроме того, необходимо иметь доступ к качественным компонентам и инструментам, чтобы обеспечить оптимальную работу электрической схемы.

ТЕМА 3. ОБЗОР РАЗЛИЧНЫХ КЛАССОВ РОБОТОВ. ВЫБОР ПРОЕКТНОГО РОБОТА.

Занятие №6. Обзор классов роботов

Робот - это устройство, которое способно выполнять различные задачи без прямого участия человека. Кроме того, роботы могут быть созданы для выполнения определенных целей, которые обычно определяются задачами, на которые направлено их применение. В основе любого робототехнического изделия лежит сочетание механических компонентов и управляющего центра, который позволяет программировать робота на выполнение нужных задач.

Сегодня роботы используются во многих сферах жизни, например, в производстве, медицине, образовании, науке и даже в военной сфере. Роботы помогают людям в работе, улучшают качество жизни, повышают эффективность и экономическую выгоду.

Робототехника – это наука о создании роботов. В настоящее время, робототехника является одной из самых быстроразвивающихся отраслей техники, которая представляет огромный потенциал не только в автоматизации, но и в других сферах жизни. Основы робототехники включают в себя знания из различных научных областей, таких как электроника, механика, программирование и другие. Роботы создаются для того, чтобы выполнять определенные задачи, их действия программируются заранее и контролируются специальным оборудованием.

Создание робота начинается с задания определенных параметров и требований к его функциональности. Затем инженеры и конструкторы приступают к проектированию механической части робота, включая опоры, манипуляторы, шарниры, датчики и другие компоненты. Они также создают управляющую систему, которая позволяет программировать робота на выполнение задач.

Ранее уже рассматривались основные понятия робототехники и основные разделы, которые она в себя включает.

Раздел программирования микроконтроллеров. является важным и обширным, так как он посвящен созданию управляющих программ для будущего устройства, их отладке, а также методам оптимизации ресурсов системы. В данном учебном курсе основное внимание уделяется языку программирования Си, который является наиболее распространенным в робототехнике. В последнее

время все большую популярность приобретает язык Python, который легче освоить для многих пользователей.

Раздел схемотехника занимается электрическими платами и цепями. Использование схемотехники в робототехнике необходимо для создания самостоятельных оптимизированных устройств. Успех любого робототехнического проекта зависит от уверенного знания принципов построения электрических цепей, строения электронных плат и умения обращаться с ними. Существуют специализированные программные средства, которые позволяют проводить инженерные расчеты максимально удобно, а также проектировать и отлаживать электрические схемы и платы. Этот курс включает как теоретические основы, так и полезные прикладные навыки, включая методику создания плат. Таким образом, знание светотехники является важным элементом успеха в робототехнических проектах.

Раздел 3D моделирования занимается созданием трехмерных визуальных моделей объектов и систем. В робототехнике использование 3D моделирования является неотъемлемой частью разработки и проектирования роботов. Оно позволяет инженерам и дизайнерам создавать и визуализировать реалистичные трехмерные модели роботов, компонентов и окружающей среды. 3D моделирование дает возможность рассмотреть и оценить различные аспекты робототехнического проекта, такие как форма, размеры, функциональность и взаимодействие компонентов. С помощью специализированного программного обеспечения, инженеры могут создавать, модифицировать и анализировать 3D модели роботов перед их физической реализацией.

Раздел механика служит для создания элементов устройства. В данном разделе используются знания в области механики и инженерии. Она охватывает основные принципы создания механизмов и способы соединения различных деталей. В этом разделе рассматриваются часто используемые решения и области их применения. Главная цель механики заключается в создании конкурентоспособных и логичных инженерных решений для механической части робототехнической системы.

Часто термины робот и робототехника связывают с мехатроникой, так как они также включают в себя механические и программные компоненты. Мехатроника включает в себя технологии, которые объединяет точную механику с электроникой, электротехникой и компьютерными компонентами для создания новых механизмов, машин и систем. Они могут быть управляемыми и обладают интеллектуальными возможностями. Область мехатроники позволяет создавать

сложные системы, которые могут выполнять различные функции, используя точную механику и электронику, что значительно повышает их эффективность в самых разных областях.

Еще раз обозначим основные программами, на которые будет ориентирован данный курс изучения робототехники. Указанное ПО можно использовать как для обучения, так и для дальнейшего развития и работы в сфере робототехники.

● **Arduino IDE.** Основным программным обеспечением для программирования микроконтроллера Arduino является Arduino IDE. Оно является самым первым и популярным ПО для этих целей и может использоваться как для обучения, так и для реальной работы. Arduino IDE была специально разработана для написания программ под Arduino и схожие контроллеры, и использует язык программирования, похожий на С. Благодаря широкому спектру специализированных библиотек для подключения различных модулей, начать программирование сложных компонентов может быть довольно просто, даже если не глубоко понимать их принципы работы. В этом курсе мы не будем создавать свою управляющую систему с нуля, а познакомимся с уже готовыми решениями на базе платформы Arduino.

● **Tinkercad**. Данный сервис позволяет составлять несложные электронные схемы как с использованием микроконтроллеров, так и без. В этом случае основным достоинством является простота работы. В отличие от других программ, которые будут рассмотрены далее, все электронные компоненты представлены в виде рисунков самих компонентов, а не условных обозначений. Это упрощает разработку. Данную программу рекомендуется использовать именно для начального обучения. В Tinkercad имеются заранее созданные схемы, которые расположены в специальной последовательности и позволяют пройти базовый курс обучения уже в самой программе.

● **Fritzing.** Одним из наиболее удобных инструментов для создания собственных плат и схем является программа Fritzing. В этой программе очень просто перейти от работы с изображениями компонентов к использованию универсальных обозначений, принятых в электронике. Fritzing позволяет создавать схемы, используя аналогию с Tinkercad, но при этом предлагает более широкий спектр компонентов, что делает возможным создание более сложных схем.

● **EveryCircuit** - представляет собой электронный онлайн эмулятор с хорошими графиками. Компоненты схемы имитируются с минимальными параметрами. Очень проста в использовании, имеет отличный дизайн.

● **CircuitLab** - бесплатное веб-приложение CircuitLab является свободным, веб-базированным инструментом, который может упростить процесс проектирования: резисторы, конденсаторы, источники тока и остальные детали схемы размещают на чертежной доске и соединяют все элементы. Если в процессе проектирования возникли проблемы, можно обратиться к сообществу за помощью, расширив ваши схемы отметкой “публичные”. Кроме того, так как приложение веб-базированное, значит оно должно хорошо работать на всех платформах.

На этом занятии были освещены с основы робототехники, что позволяет продолжить более углубленное изучение робототехники. Далее более подробно изучим платформу Arduino.

Занятие №7. Выбор класса роботов под задачи

В современном мире в связи с применением роботов во многих сферах жизни, классификация роботов постоянно дополняется. Рассмотрим основные классы роботов.

<u>В зависимости от рода выполняемой роботами промышленные роботы можно разделить на классы.</u>

● Литейные - предназначены для отливки изделий расплавленным материалом, в том числе и 3D принтеры. Главной технологической сложностью при разработке являются высокие температуры при плавлении.

● Роботы для механических обработок - используются при обработке изделий с помощью механического воздействия с применением режущего инструмента, кузнечных работах, а также прессовки и штамповки.

● Сборочные - в большинстве случаев это манипуляторы использующие различные инструменты как для механического соединения, так и для пайки электронных компонентов.

● Окрасочные - используются при автоматическом нанесении лакокрасочного покрытия, а также последующей полировки изделия.

● Строительные - предназначены для автоматизации строительства, а также добычи ресурсов, сюда входят и роботизированные средства доставки строительных материалов и машины для постройки различных объектов.

● Фасовочно-сортировальные - используются для проверки качества продукта, его сортировки и фасовки в упаковку, в большинстве случаев это

последний этап автоматизации на конвейерах, не считая средств доставки изделий потребителям.

- Транспортные - к этому классу относятся любые роботизированные средства доставки грузов, наиболее распространенными среди них являются конвейерные.

- Сельскохозяйственные - это роботы, основной задачей которых является автоматизация сельскохозяйственного производства, например, оросители, комбайны, трактора и др.

- Исследовательские роботы - это устройства для проведения различных исследований, в том числе и возможностей использования роботов для выполнения различных функций.

- Космические - используются для проведения исследований в условиях космоса, к ним можно отнести различные исследовательские спутники.

- Боевые роботы - это многофункциональное техническое устройство с антропоморфным (человекоподобным) поведением, частично или полностью выполняющее функции человека при решении определенных боевых задач. Позволяет заменить человека при выполнении боевых задач, сохранить ему жизнь, а также выполнить задачи, несовместимые с его возможностями.

По средам использования различают.

- Воздушные БПЛА (Беспилотный летательный аппарат) - предназначены для выполнения воздушных миссий, таких как наблюдение и разведка, координация нанесения ударов по противнику, создание рядом с собой беспроводных сетей связи.

- Сухопутные - к этому классу относятся наземные боевые машины, это беспилотные военные автомобили, системы разведки, охранные системы, роботы-сапёры, а также полноценные боевые комплексы.

- Морские - этот класс объединяет роботизированных устройств надводного и подводного типа, основными задачами которых является разведка, сопровождение, патрулирование и поиск мин.

Роботы предназначены для выполнения различных операций, и в зависимости от назначения роботизированных изделий можно выделить стационарный и мобильный тип устройства.

По способу передвижения различают.

● Колесный способ - наиболее распространенный способ передвижения, который в зависимости от числа используемых колес можно разделить на подклассы. Преимуществом использования малого (от 1 до 2) количества колес может служить простота конструкции и отличная маневренность, с другой стороны, увеличение числа колес расширяет площадь контакта с поверхностью, что способствует значительному улучшению проходимости.

● Гусеничный способ - чаще всего применяется в боевых роботах, так как использование гусениц значительно повышает проходимость на пересечённой местности.

● Шагающий способ - использование для передвижения аналоги ног повышает сложность проектирования, вместе с тем современные технологии не позволяют достичь устойчивости, приближенной к человеческой.

● Передвижение по воздуху - к нему относятся так называемые БПЛА, ракеты, а также самолеты и вертолеты, оснащенные автопилотом.

● Плавающий способ - использующий для передвижения гребные винты или силы ветра, способные передвигаться над и под водой, к этому способу относятся БППА (беспилотный плавающий аппарат) а также корабли, оснащенные автопилотом.

В зависимости от рода выполняемой роботами обслуживающие роботы можно разделить на классы.

● Транспортные роботы - используются для перевозки пассажиров и грузов в автоматическом режиме.

● Умный дом - интеллектуальная, роботизированная система главной задачей которой является автоматизация и согласование всех систем жизнеобеспечения и безопасности.

● Робот-помощник - универсальный класс роботов способных на физическую и интеллектуальную помощь хозяину.

● Робот-домохозяйка - класс роботов, выполняющих повседневную работу в доме, к нему относятся роботы-повара, пылесосы, мойщики окон, посудомойки, очистители воздуха, автокормушки, уборщики бассейнов и др.

● Социальный робот - робот, способный в автономном или полуавтономном режиме взаимодействовать и общаться с людьми в общественных местах или домах.

● Роботы члены семьи - устройства, способные практически полностью “влиться” в состав семьи, способны передвигаться по дому, взаимодействовать с окружающими.

● Роботы-животные - устройства, заменяющие домашних животных, способны копировать их движения и звуки.

● Роботы-игрушки - средства развлечения детей, способствующие их обучению различным навыкам и знаниям.

● Медицинский робот - робот, созданный для выполнения медицинских манипуляций под управлением человека. Существуют роботы-хирурги, способные выполнять высокоточные операции, роботы-фармацевты, разносящие медицинские препараты пациентам в больницах, а также большое количество узкоспециализированных роботов.

● Роботизированные протезы - предназначены для замены утраченных или необратимо повреждённых частей тела искусственными роботизированными устройствами.

● Роботизированные трансплантаты - используются для замены поврежденных или не функционирующих органов и тканей на роботизированные устройства, способными действенно их заменить.

● Роботы-сиделки - способны заменить работников младшего медицинского персонала при уходе за больными.

● Роботизированные симуляторы пациентов - предназначены для практического обучения и отработки навыков медицинских специалистов.

● Роботы-диагносты - способны на основе данных анамнеза поставить диагноз и назначить лечение.

Современная жизнь постоянно предлагает все новые возможности по применению роботов, что ставит перед разработчиками проблему выбора класса роботов под ту или иную задачу.

Каждый класс роботов уникален и предназначен для выполнения определенных задач. Для понимания, рассмотрим более подробно отдельные классы роботов.

Манипуляционные роботы состоят из манипулятора и перепрограммируемого устройства управления, которое формирует управляющие воздействия, задающие требуемые движения исполнительных органов манипулятора. Эти роботы значительно автоматизируют процесс

конвейерного производства, что позволяет увеличить производительность труда, уменьшить издержки производства и снизить влияние человеческого фактора, что, в свою очередь, повысит конкурентоспособность.

Социальные роботы - это современные автоматические устройства, которые могут быть запрограммированы для общения с людьми в различных общественных местах и домах. Эти роботы могут выполнять множество функций, таких как обеспечение безопасности, оказание помощи в повседневных задачах, обучение и развлечение. Социальные роботы используются в широком спектре сфер, включая здравоохранение, образование, туризм, развлечения и многие другие. Они могут быть полезны для старшего поколения, людей с ограниченными возможностями, а также для детей и молодежи. Они могут быть использованы как персональные помощники, гиды, преподаватели, врачи и даже друзья. Кроме того, социальные роботы могут улучшить качество жизни людей, уменьшить социальную изоляцию и повысить уровень коммуникации и взаимодействия в обществе.

Исследовательские роботы — это роботы, используемые для научных исследований, могут варьироваться от простых роботов-исследователей, до более сложных и специализированных систем. Они используются во многих областях, таких как геология, астрономия, метеорология, океанография и биология.

На этом занятии были перечислены различные классы роботов, что позволяет определиться с тематикой робота для собственного проекта.

ТЕМА 4. ИЗУЧЕНИЕ ОСНОВ ЯЗЫКА СИ. СИНТАКСИС, ПРОСТЕЙШИЕ ПОНЯТИЯ

Занятие №8. Основы языка программирования Си

Язык программирования Си был создан в начале 1970-х годов в Bell Labs Деннисом Ритчи и Кеном Томпсоном и быстро стал очень популярным среди программистов, благодаря своей эффективности и универсальности.

Рассмотрим основные аспекты языка программирования Си.

- Синтаксис. Си является языком со строгой синтаксической структурой. Основные элементы языка - это переменные, операторы, функции, массивы, структуры и указатели.

● Типы данных. Си поддерживает несколько типов данных, таких как целочисленные (int), символьные (char), вещественные (float, double) и логические (bool).

● Управляющие конструкции. В Си используются управляющие конструкции, такие как условные операторы (if-else), циклы (for, while, do-while), операторы перехода (break, continue, return) и операторы выбора (switch-case).

● Функции. Функции в Си позволяют разделять код на отдельные логические блоки. Они могут быть определены внутри других функций и возвращать значения.

● Указатели. Указатели в Си являются мощным инструментом для работы с памятью. Они позволяют получить доступ к адресу ячейки памяти и изменять значения, хранящиеся по этому адресу.

● Библиотеки. Си имеет множество стандартных библиотек, которые содержат множество полезных функций, таких как работы со строками, математические операции, ввод/вывод и т.д.

● Препроцессор. Си имеет мощный препроцессор, который позволяет использовать директивы препроцессора для управления компиляцией кода и определения констант.

В целом, Си является мощным и эффективным языком программирования, который используется во многих областях, таких как разработка операционных систем, системное программирование, научные и инженерные вычисления, разработка приложений и игр.

Для создания проектов для Arduino на языке программирования Си необходимо знать основы языка Си. Как минимум, следует иметь представления о следующих понятиях.

1. Синтаксис: основные элементы синтаксиса языка Си, такие как переменные, функции, операторы, условные операторы, циклы и т.д.

2. Типы данных: различные типы данных, такие как целочисленные, вещественные, символьные, логические, указатели и т.д.

3. Массивы: работа с массивами, объявление, инициализация, доступ к элементам массива и циклический доступ.

4. Функции: объявление, вызов и использование функций в коде.

5. Указатели: понимание указателей и работы с ними, в том числе указателей на функции.

6. Структуры и объединения: объявление, инициализация, доступ к полям и использование структур и объединений в коде.

7. Директивы препроцессора: знание директив препроцессора, таких как #include, #define, #ifdef и т.д.

8. Библиотеки: использование стандартных библиотек языка Си, таких как stdio.h, stdlib.h, math.h и т.д.

9. Операции ввода-вывода: понимание операций ввода-вывода и использование функций ввода-вывода, таких как printf и scanf.

В этом занятии представлены основные понятия языка программирования Си, что позволяет научиться создавать программы, в том числе и для управления роботами. Кроме того, для работы с Arduino необходимо знание основ микроконтроллеров и понимание принципов работы платформы.

Занятие №9. Основы языка программирования Си

Рассмотрим простые примеры, которые дают представление о программировании на языке Си.

Пример 1: Вывод текста на экран.

Для вывода текста на экран в С используется функция printf(). Эта функция определена в заголовочном файле stdio.h.

```
#include <stdio.h> // Подключаем заголовочный файл для работы с вводом и выводом
int main() {
        printf("Привет, мир!\n"); // printf - функция вывода строки на экран
return 0; // выход из функции main, что означает выход из программы
}
```

Вывод:

```
Привет, мир!
```

Пример 2: Основные операции с числами.

В Си существует несколько основных операций с числами, таких как сложение (+), вычитание (-), умножение (*) и деление (/). Вот пример использования этих операций:

```
#include <stdio.h> // Подключаем заголовочный файл для работы с вводом и выводом
int main() {
  // Объявляем целочисленные переменные
    int a = 10, b = 5; // Начальные значения a и b
    int sum = a + b; // считаем сложение a и b
    int diff = a - b; // считаем вычитание a и b
    int prod = a * b; // считаем умножение a и b
    int quot = a / b; // считаем деление a и b
    // Используем функцию printf для вывода на экран
    printf("Сумма: %d\n", sum);
 printf("Разность:%d\n", diff);
    printf("Произведение: %d\n", prod);
    printf("Частное: %d\n", quot);
    return 0; // выход из программы
}
```

Вывод:

```
Сумма: 15
Разность: 5
Произведение: 50
Частное: 2
```

Пример 3: Операторы ветвления.

Операторы ветвления используются в Си для выполнения различных действий в зависимости от условия. Вот пример использования операторов ветвления:

```
#include <stdio.h> // Подключаем заголовочный файл для работы с вводом и выводом
int main() {
```

```
    // объявляем целочисленные переменные
    int a = 10, b = 5; // задаем начальные значения a и
b
      if (a > b) {
        printf("a больше, чем b\n"); // вывод на экран
      }
      else if (a == b) {
        printf("a равно b\n"); // вывод на экран
      }
      else {
        printf("a меньше, чем b\n"); // вывод на экран
      }
      return 0; // выход из программы
}
```

Вывод:

```
a больше, чем b
```

Пример 4: Циклы.

Циклы используются в Си для повторения одного и того же блока кода несколько раз. Вот пример использования цикла:

```
#include <stdio.h> // Подключаем заголовочный файл для
работы с вводом и выводом
int main() {
    int i; // объявляем целочисленную переменную, но
 не задаем начальное значение
    for (i = 1; i <= 10; i++) {
      printf("%d\n", i); // вывод на экран
    }
    return 0; // выход из программы
}
```

Вывод:

```
1
2
3
4
5
6
7
8
9
10
```

Пример 5: Массивы.

Массивы позволяют хранить несколько значений одного типа в одной переменной. В Си массивы задаются так:

```
#include <stdio.h> // Подключаем заголовочный файл для работы с вводом и выводом
int main() {
    int arr[5] = {1, 2, 3, 4, 5}; // объявляем массив целочисленных данных размером 5
    printf("%d\n", arr[0]); // Выводит первый элемент массива
    printf("%d\n", arr[2]); // Выводит 3й элемент массива
    return 0; // выход из программы
}
```

Вывод:

```
1
3
```

Пример 6: Функции.

Функции позволяют разбить программу на более мелкие и удобные блоки, каждый из которых выполняет определенную задачу. Вот пример использования функции:

```
#include <stdio.h> // Подключаем заголовочный файл для работы с вводом и выводом
// Объявили функцию сложения двух целых чисел
int sum(int a, int b) {
return a + b; // возвращаем сумму двух входных параметров функции
}
int main() {
int a = 10, b = 5; // объявили начальные значения переменных
int s = sum(a, b); // вызываем функцию сложения и кладем в переменную s сумму
printf("Сумма: %d\n", s);// вывод на экран
return 0;
}
```

Вывод:

```
Сумма: 15
```

Пример 7: Указатели.

Указатели позволяют работать с адресами памяти в программе. Вот пример использования указателей:

```
#include <stdio.h> // Подключаем заголовочный файл для работы с вводом и выводом
int main() {
int a = 10;
int *p = &a; // Определение указателя
printf("%d\n", a); // Выводит 10 printf("%d\n",*p); // Выводит 10
*p = 20; // Изменение значения, на которое указывает указатель
```

```
printf("%d\n", a); // Выводит 20
return 0;
}
```

Вывод:

```
10
10
20
```

Пример 8: Структуры.

Структуры позволяют группировать несколько переменных в одну логическую единицу. Вот пример использования структур:

```
#include <stdio.h> // Подключаем заголовочный файл для работы с вводом и выводом
struct Person {
char name[50];
int age;
};
int main() {
struct Person person1 = {"John Doe", 30}; // создаем переменную типа Person с name = “John Doe” и age = 30
printf("Имя: %s\n", person1.name); // Выводит "John Doe"
printf("Возраст: %d\n", person1.age); // Выводит 30
return 0; // выход из программы
}
```

Вывод:

```
Имя: John Doe
Возраст: 30
```

Пример 9: Динамическая память.

В Си можно выделять и освобождать память вручную с помощью функций malloc и free. Вот пример использования динамической памяти:

```
#include <stdio.h> // Подключаем заголовочный файл для работы с вводом и выводом
#include <stdlib.h> // Подключаем заголовочный файл для работы с функцией объявления памяти malloc
int main() {
int *p = (int*)malloc(sizeof(int)); // Выделение памяти
if (p == NULL) {
// Проверка, выделена ли память успешно
printf("Ошибка выделения памяти!\n");
exit(1); // аварийный выход из программы с кодом 1
}
*p = 10; // Присваивание значения
printf("%d\n", *p); // Выводит 10
free(p); // Освобождение памяти
return 0;
}
```

Вывод:

```
10
```

Пример 10: Работа со строками.

Строки в Си представляются как массивы символов. Вот пример работы со строками:

```
#include <stdio.h> // Подключаем заголовочный файл для работы с вводом и выводом
#include <string.h> // Подключаем заголовочный файл для работы со строками
int main() {
char str1[50] = "Hello, "; // создаем массив символов
размером 50 и кладем в него строку "Hello, "
```

```
char str2[50] = "world!"; // создаем массив символов
размером 50 и кладем в него строку “world!“

strcat(str1, str2); // Функция склеивания строк

printf("%s\n", str1); // Выводит "Hello, world!"

return 0;

}
```

Вывод:

```
Hello, world!
```

Пример 11: Файлы.

В Си можно работать с файлами, используя стандартные библиотеки ввода-вывода. Вот пример чтения и записи в файл:

```
#include <stdio.h> // Подключаем заголовочный файл для
работы с вводом и выводом


int main() {
FILE *fp;
char str[100];
// Запись в файл
fp = fopen("file.txt", "w");
fprintf(fp, "Это строка, которую нужно записать
в файл!\n");
fclose(fp);
// Чтение из файла fp = fopen("file.txt", "r");
fgets(str, 100, fp);
printf("%s", str);
fclose(fp);
return 0;
}
```

Вывод:

```
Это строка, которую нужно записать в файл!
```

На этом занятии представлены примеры, написанными на языке Си. Но это всего лишь несколько примеров того, что можно сделать с помощью этого языка программирования. Язык Си является основным языком программирования в работе с микроконтроллерами Arduino.

Занятие №10. Интерфейс работы с Arduino

Одним из ключевых компонентов работы с Arduino является интерфейс программирования и управления.

Интерфейс программирования и управления для работы с Arduino включает в себя несколько основных компонентов.

- IDE (Integrated Development Environment) - интегрированная среда разработки, которая предоставляет инструменты для написания, отладки и загрузки программ на платформу Arduino.

- Компилятор - программа, которая преобразует код, написанный на языке Arduino, в машинный код, который может быть загружен на платформу.

- Библиотеки - наборы функций и методов, которые могут быть использованы для работы с различными компонентами, такими как сенсоры, моторы и другие устройства.

- USB-порт - используется для загрузки программного кода (скетчей) на платформу и для подключения платформы к компьютеру для управления и мониторинга.

- Серийный монитор - программа, которая позволяет мониторить данные, передаваемые между платформой и компьютером через серийный порт.

- Язык программирования Arduino - базируется на языке Си, но имеет несколько специфических для Arduino функций и библиотек.

Интерфейс работы с Arduino является простым и удобным для начинающих. Интегрированная среда разработки Arduino IDE содержит все необходимые инструменты для создания и загрузки скетчей на платформу. Кроме того, множество библиотек и документации доступны в Интернете, что позволяет ускорить процесс разработки. Шилды и модули расширения - это дополнительные устройства, которые можно подключать к плате Arduino для расширения ее функциональности. Они включают в себя различные сенсоры,

модули связи, дисплеи, устройства управления моторами и многое другое. Интерфейс работы с Arduino также позволяет пользователю использовать различные протоколы связи, такие как USB, Bluetooth и Wi-Fi, для управления платой Arduino. Это позволяет пользователю контролировать плату Arduino из удаленной локации или с помощью мобильного устройства. Кроме того, в интерфейсе работы с Arduino есть возможность мониторинга и отладки программного кода, а также взаимодействия с внешними приложениями, такими как Processing или Max/MSP.

В целом, интерфейс работы с Arduino предоставляет различные инструменты и ресурсы для создания электронных проектов, что делает его популярным средством для обучения, прототипирования (процесс создания макета) и разработки IoT-устройств (интернет вещей - это множество физических объектов, подключенных к интернету и обменивающихся данными).

На этом занятии был представлен интерфейс программирования и управления Arduino. Знакомство с интерфейсом дает нам возможность загружать программы на Arduino, а также производить взаимодействие между компьютером и платой.

Занятие №11. Алгоритмы в робототехнике

Робототехника - это область, в которой роботы используются для автоматизации и упрощения различных задач. Важной частью процесса создания робота является разработка алгоритмов, которые позволяют ему выполнить нужные действия.

Алгоритм - это последовательность шагов, которые выполняются для достижения определенной цели. В робототехнике алгоритмы используются для управления движениями робота, обработки сенсорных данных и принятия решений.

Одним из наиболее распространенных типов алгоритмов в робототехнике являются алгоритмы управления движением. Они используются для контроля за движением робота и его положением в пространстве. Такие алгоритмы обычно основаны на информации, получаемой от сенсоров, которые устанавливают положение и скорость робота. На рисунке 11.1 представлен один из вариантов алгоритмов управления движения.

Другой тип алгоритмов, используемых в робототехнике, это алгоритмы обработки данных. Они обрабатывают данные, полученные от сенсоров, и вычисляют, какие действия необходимо выполнить. Например, алгоритм

обработки данных может анализировать изображение, полученное от камеры, и принимать решение о том, какую команду отправить роботу.

Еще один тип алгоритмов, используемых в робототехнике, это алгоритмы принятия решений. Они анализируют данные, полученные от сенсоров, и принимают решение о том, какое действие должен выполнить робот. Например, алгоритм принятия решений может анализировать данные о расстоянии до препятствия и решить, нужно ли роботу остановиться или изменить направление движения.

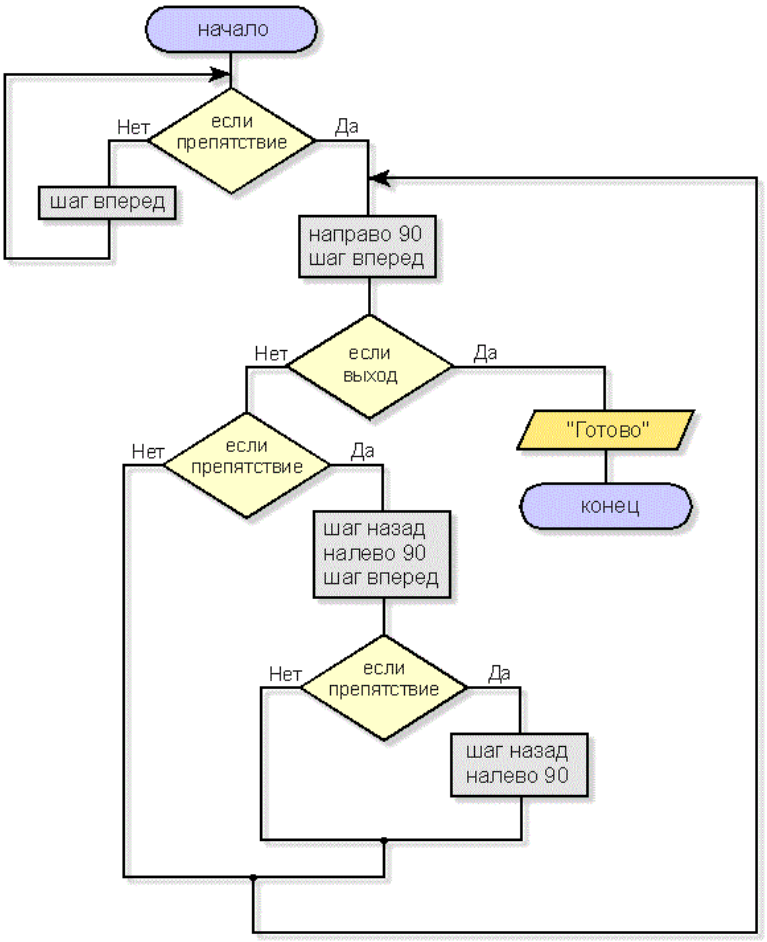

Рис. 11.1 Пример алгоритма управления движением

Наконец, алгоритмы машинного обучения становятся все более популярными в робототехнике. Они используются для обучения робота на основе полученных данных. Например, робот может обучаться распознавать объекты на изображениях, используя алгоритмы машинного обучения.

Рассмотрим более подробно один из самых популярных базовых алгоритмов.

Алгоритмы поиска пути используются в робототехнике для определения пути, который должен пройти робот от точки А к точке В. Эти алгоритмы могут быть применены для навигации в лабиринтах, автономного вождения автомобилей и даже для обнаружения и исправления ошибок в системах управления производственными процессами. Некоторые из наиболее распространенных алгоритмов поиска пути включают следующие варианты последовательности шагов.

- Алгоритм Дейкстры: используется для поиска кратчайшего пути в графе(Граф — математическая абстракция реальной системы любой природы, объекты которой обладают парными связями. Граф как математический объект есть совокупность двух множеств — множества самих объектов, называемого множеством вершин, и множества их парных связей, называемого множеством рёбер. Элемент множества рёбер есть пара элементов множества вершин. В качестве простейшего примера из жизни можно привести схему перелетов определенной авиакомпании, которая моделируется графом, где вершинами графа являются города, а рёбрами — рейсы, соединяющие пары городов. Дерево каталогов в компьютере также является графом: диски, папки и файлы являются вершинами, а ребра показывают вложенность файлов и папок в папки и диски.) с неотрицательными весами ребер. Этот алгоритм работает путем просмотра каждой вершины графа и нахождения кратчайшего пути от начальной вершины до каждой другой вершины в графе.

- Алгоритм A*: используется для поиска кратчайшего пути в графе с ненулевыми весами ребер. Этот алгоритм является более эффективным, чем алгоритм Дейкстры, так как он учитывает не только расстояние от начальной вершины до текущей вершины, но и эвристическую оценку(метод проверки удобства интерфейса) расстояния до конечной вершины.

- Алгоритм Джонсона: используется для поиска кратчайшего пути в графе с произвольными весами ребер. Этот алгоритм является более эффективным, чем алгоритмы Дейкстры и A*, так как он сначала пересчитывает веса ребер, чтобы сделать их неотрицательными, а затем использует алгоритм Дейкстры для поиска кратчайшего пути.

На этом занятии было дано понятие алгоритма и перечислены наиболее распространенные из них. Также были изучили специфические для робототехники алгоритмы. Стоит отметить, что разработка алгоритма функционирования является одной из важной частью процесса проектирования и программирования роботов.

Занятие №12. Создание библиотек

Библиотеки в программировании - сборник подпрограмм или объектов, используемых для разработки программного обеспечения.

Существует два типа библиотек: статические библиотеки и динамические библиотеки.

Статическая библиотека (иногда называемая archive, “архив”) состоит из подпрограмм, которые скомпилированы и линкуются непосредственно с вашей программой. Когда вы компилируете программу, использующую статическую библиотеку, все функции статической библиотеки, которые использует ваша программа, становятся частью вашего исполняемого файла. В Windows статические библиотеки обычно имеют расширение .lib, а в Linux – расширение .a (archive, архив). Одним из преимуществ статических библиотек является то, что вам нужно распространять только исполняемый файл, чтобы пользователи могли запускать вашу программу. Поскольку библиотека становится частью вашей программы, это гарантирует, что с вашей программой всегда будет использоваться правильная версия библиотеки. Кроме того, поскольку статические библиотеки становятся частью вашей программы, вы можете использовать их так же, как функции, которые вы написали для своей программы. С другой стороны, поскольку копия библиотеки становится частью каждого исполняемого файла, который ее использует, это может привести к потере большого количества места. Статические библиотеки также не могут быть легко обновлены – для обновления библиотеки необходимо заменить весь исполняемый файл.

Динамическая библиотека (также называемая shared library, “общая библиотека”) состоит из подпрограмм, которые загружаются в ваше приложение во время выполнения. Когда вы компилируете программу, использующую динамическую библиотеку, библиотека не становится частью вашего исполняемого файла – она остается отдельной единицей. В Windows динамические библиотеки обычно имеют расширение .dll (dynamic link library, библиотека динамической компоновки), а в Linux – расширение .so (shared object, общий объект). Одним из преимуществ динамических библиотек является то, что многие программы могут совместно использовать одну копию библиотеки, что экономит место. Возможно, большим преимуществом является то, что динамическую библиотеку можно обновить до более новой версии без замены всех исполняемых файлов, которые ее используют.

Поскольку динамические библиотеки не связаны с вашей программой, программы, использующие динамические библиотеки, должны явно загружать и взаимодействовать с динамической библиотекой. Этот механизм может сбивать с толку и затруднять взаимодействие с динамической библиотекой. Чтобы упростить использование динамических библиотек, можно использовать библиотеку импорта.

Библиотека в языке Си - это коллекция функций, которые могут быть использованы в разных программах. Создание библиотеки может быть полезным для повторного использования кода в различных проектах, а также для упрощения поддержки кода.

Создание библиотеки в языке Си включает в себя несколько шагов.

1. Написание функций: необходимо написать функции, которые будут включены в библиотеку. Все функции должны быть определены в отдельных файлах.

2. Создание заголовочного файла: заголовочный файл содержит прототипы функций и другие необходимые объявления. Он должен быть подключен в любой программе, которая будет использовать функции из библиотеки.

3. Создание статической или динамической библиотеки: в зависимости от потребностей проекта можно создать статическую или динамическую библиотеку. Статическая библиотека включает объектные файлы функций непосредственно в исполняемый файл, в то время как динамическая библиотека загружается в память только при необходимости.

4. Компиляция и сборка: после написания функций и создания заголовочного файла и библиотеки необходимо скомпилировать и собрать все компоненты. Это может быть выполнено с помощью утилиты ar (для создания статической библиотеки) или ld (для создания динамической библиотеки).

5. Тестирование: перед использованием библиотеки необходимо ее протестировать, чтобы убедиться в ее правильной работе.

6. Использование: заголовочный файл и скомпилированная библиотека могут быть подключены к другим проектам и использоваться для вызова функций, определенных в библиотеке.

Рассмотрим каждый шаг более подробно.

<u>Написание функций.</u>

Создание библиотеки начинается с написания функций, которые необходимо включить в нее. Каждая функция должна быть определена в отдельном файле с расширением ".c". Например, если вы создаете функцию для работы с матрицами, то вы можете создать файл "matrix.c", который содержит определение функций для работы с матрицами.

Создание заголовочного файла.

Заголовочный файл содержит прототипы функций и другие объявления, которые необходимы для использования функций в библиотеке. Заголовочный файл имеет расширение ".h". Например, для библиотеки матрицы можно создать заголовочный файл "matrix.h", который содержит объявления функций, определенных в "matrix.c".

Создание статической или динамической библиотеки.

Создание статической библиотеки может быть выполнено с помощью утилиты "ar". Для создания динамической библиотеки можно использовать утилиту "gcc". *Утилита* - вспомогательная компьютерная программа в составе общего программного обеспечения для выполнения специализированных типовых задач, связанных с работой оборудования и операционной системы. Разница между статической и динамической библиотекой заключается в том, как функции включаются в программу. Статическая библиотека включает объектные файлы функций непосредственно в исполняемый файл, поэтому размер исполняемого файла может быть большим. Динамическая библиотека загружается в память только при необходимости, поэтому она может быть использована несколькими программами.

Компиляция и сборка.

После написания функций и создания заголовочного файла и библиотеки необходимо скомпилировать и собрать все компоненты. Для этого мы используем компилятор Си и утилиты ar или gcc. Статическая библиотека может быть создана с помощью следующей команды:

```
ar rcs libmatrix.a matrix.o
```

Здесь "libmatrix.a" - имя создаваемой библиотеки, а "matrix.o" объектный файл, содержащий функции.

Для создания динамической библиотеки можно использовать следующую команду:

```
gcc -shared -o libmatrix.so matrix.o
```

Здесь "libmatrix.so" - имя создаваемой динамической библиотеки.

Тестирование

Перед использованием библиотеки необходимо библиотеку протестировать, чтобы убедиться в ее правильной работе. Для этого можно написать несколько простых тестов для каждой функции, чтобы проверить работу библиотеки. Обычно для тестирования используются специальные библиотеки, такие как "CUnit" или "Check".

Использование библиотеки.

После тестирования библиотека готова к использованию. Для использования библиотеки в программе нужно подключить заголовочный файл и скомпилировать программу, используя созданную библиотеку.

Для использования статической библиотеки в программе нужно скомпилировать ее с помощью команды:

```
gcc -o program program.c -L. -lmatrix
```

Здесь "program.c" - файл программы, которая использует функции из библиотеки, а "-L." и "-lmatrix" - параметры компилятора, которые указывают на местоположение библиотеки и ее имя соответственно.

Для использования динамической библиотеки в программе нужно скомпилировать ее с помощью команды:

```
gcc -o program program.c -L. -lmatrix
```

Здесь параметры "-L." и "-lmatrix" указывают на местоположение и имя динамической библиотеки соответственно.

На этом занятии мы изучили создание библиотек в языке C - это довольно простой процесс, который включает в себя написание функций, создание заголовочного файла, создание библиотеки, компиляцию и сборку, тестирование и использование. Создание библиотек позволяет повторно использовать код и упрощает разработку программного обеспечения.

Знание основ языка Си является ценным для программирования Arduino, так как обеспечивает более глубокое понимание внутренней структуры и функционирования платформы. Это помогает разработчикам более эффективно использовать ресурсы Arduino, создавать оптимизированный и надежный код, а также лучше понимать взаимодействие с аппаратурой и разрабатывать сложные проекты, используя всю мощь Arduino. Знание основ языка Си дает

программисту больше гибкости и контроля над кодом, позволяет реализовывать более сложные алгоритмы и эффективно использовать ресурсы микроконтроллера.

ТЕМА 5. ЗНАКОМСТВО С МИКРОКОНТРОЛЛЕРОМ ARDUINO

Занятие №13. Знакомство с микроконтроллером Arduino

На предыдущих занятиях уже рассматривались основные аспекты платформы Arduino, сейчас познакомимся с микроконтроллером Arduino.

Arduino-платы - это миниатюрные компьютеры, состоящие из микроконтроллера семейства AVR (Advanced Virtual RISC) и программы, записанной в память. Для расширения функциональности платы используются шилды - дополнительные платы, которые подключаются по типу “бутерброда” и имеют конкретные элементы, такие как дисплей, контроллер, драйвер двигателя, датчик.

Использование шилдов в проектах значительно упрощает разработку и сборку сложных систем, экономит место и время. Кроме того, на рынке существует множество аксессуаров для Arduino, таких как датчики температуры, влажности, вибрации, переменного тока, препятствий, устройства вывода и ввода, адаптеры, макетные платы, соединительные перемычки и многое другое.

Оригинальные платы Arduino доступны для покупки на официальном сайте компании производителя, однако существует множество плат-клонов, производимых с использованием документации, которая есть на официальном сайте Arduino в открытом доступе. При выборе клонов следует учитывать их качество и соответствие стандартам. Arduino-платы широко используются в различных областях, таких как автоматизация дома, IoT-проекты, робототехника, медицинская техника и многие другие. Их простота и универсальность делает их доступными для любого уровня пользователей, от начинающих до профессионалов.

Ниже приведены несколько типов плат Arduino, которые различаются форм-фактором, характеристиками микроконтроллера, количеством портов и функционалом.

- Arduino UNO (рис. 13.1). Самый распространенный размер с 20 входами-выходами (14 цифровых, 6 аналоговых). Построен на основе микроконтроллеров ATmega168, ATmega328. Совмещается со всеми шилдами и периферийными устройствами. Варианты – Uno, Leonardo, Extreme, NG, Diecimila, Duemilanove.

Рис. 13.1 Arduino UNO

● Arduino Mega (рис. 13.2) – увеличенный размер, расширенный набор интерфейсов, максимальная мощность. Работает на основе микроконтроллера ATmega2560, 70 входов-выходов (54 цифровых, 16 аналоговых). Совмещается только с определенными шилдами. Варианты – Mega, Mega2560 и Arduino ADK.

Рис. 13.2. Arduino MEGA

● Arduino Nano (рис.13.3) – аналогичная Arduino xxx, но уменьшенный размер платы, 22 входа-выхода, не совместима с шилдами. Для компактных устройств.

● Arduino Micro (Рис. 13.4) – встроенная поддержка USB-соединения. Применяется как HID-устройство (клавиатура, мышь, MIDI-устройство).

Рассмотрим одну из плат подробнее. На рисунке 13.5 изображены основные компоненты Arduino UNO, где:

- микроконтроллер аналог микропроцессора в обычном ПК;

- кнопка сброса осуществляет сброс микроконтроллера и повторный запуск программы;
- порт USB обеспечивает связь с ПК и питание устройства;
- светодиод №13 светодиод, соединенный с цифровым выходом №13;
- питание +9В дополнительное питание от внешнего источника (батарея, блок питания).

Рис. 13.3. Arduino NANO

Рис. 13.4. Arduino MICRO

Для работы с Arduino Uno потребуется персональный компьютер со свободным USB портом. В случае подключения Arduino к компьютеру, внешнее питание устройству не требуется. Для включения микроконтроллера, достаточно подать на него питание либо от ПК через USB кабель, либо напрямую через специальный разъем внешнего питания. При этом, напряжение внешнего питания может варьироваться от +7В до +12В.

Рис. 13.5. Компоненты Arduino UNO

На этом занятии были представлены различные платы Arduino. На основании полученных знаний можно правильно выбрать плату для своего собственного робота.

ТЕМА 6. СХЕМОТЕХНИКА. ЗНАКОМСТВО С МОДУЛЯМИ, РАБОТАЮЩИМИ НА БАЗЕ ARDUINO

Занятие №14.Знакомство со светодиодом

Светодиод (LED) - это электронный компонент, который излучает свет при прохождении электрического тока через него. Светодиоды разных цветов изображены на рисунке 14.1. В контексте Arduino, светодиоды используются как выходные устройства для отображения информации в виде световых сигналов.

Для работы светодиода с Arduino необходимо подключить его к плате. Светодиод имеет две “ноги” - длинную (анод) и короткую (катод). Длинная “нога” должна быть подключена к цифровому выводу платы через резистор, а короткая “нога” должна быть подключена к “земле”. Когда цифровой вывод включен, ток проходит через светодиод, и он начинает излучать свет.

Светодиоды могут быть разных цветов, таких как красный, зеленый, синий, желтый и т.д. Также существуют светодиоды с разными уровнями яркости и углами обзора.

Светодиоды на Arduino могут использоваться для различных целей, таких как индикация статуса программы, отображение данных с датчиков, сигнализация о событиях и т.д. Кроме того, светодиоды можно использовать вместе с другими компонентами, такими как кнопки, датчики, дисплеи и т.д. для создания интерактивных проект.

Рис. 14.1. Светодиоды разных цветов

Важно помнить, что светодиод нельзя заменить на резистор в цепи, потому что он обладает нелинейными характеристиками. Также стоит учитывать, что светодиод полярен, то есть при неправильном подключении он не будет светиться. Кроме того, каждый светодиод имеет свой максимальный ток, который он может выдержать при работе. Для большинства обычных 3 и 5 мм светодиодов это значение составляет 20 мА. Еще одной важной характеристикой светодиода является падение напряжения (Forward Voltage), которое зависит от цвета свечения. Кристалл светодиода определяет его цвет, а значит и падение напряжения. Например, у красных светодиодов падение напряжения составляет около 2.5 В, а у синих, зеленых и белых около 3.5 В. Если необходимо получить более точную информацию о светодиоде, можно обратиться к его документации или использовать таблицу минимальных значений.

Мигать светодиодом с помощью Arduino – очень простая задача: для начала необходимо подключить катод светодиода к контакту GND, а анод – к одному из “пинов” GPIO. Многие начинающие пользователи Arduino считают, что аналоговые “пины” на плате являются исключительно аналоговыми, но это не совсем верно: на самом деле они являются обычными цифровыми “пинами” с возможностью оцифровки аналогового сигнала. На плате Nano, например, “пины” A0-A5 могут использоваться как цифровые, так и аналоговые, в то время как “пины” A6 и A7 работают исключительно в режиме чтения аналогового

сигнала. Поэтому, если подключиться к “пину” A1, настроим его на режим выхода и начнем мигать светодиодом, то все должно работать. На рисунке 14.2 изображена схема подключения лампочки с резистором.

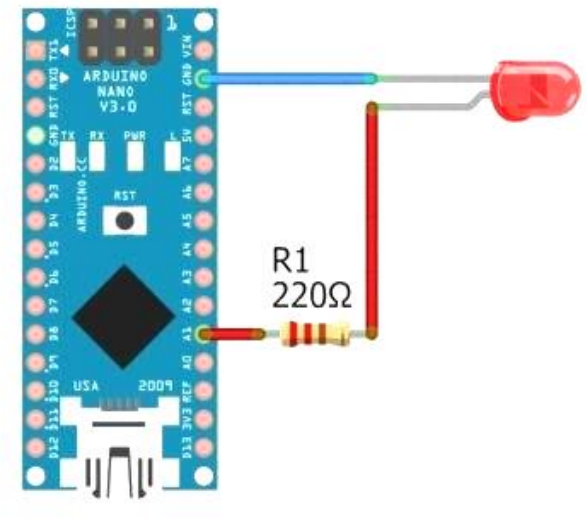

Рис. 14.2. Схема подключения лампочки с резистором

Рассмотрим программный код, который предназначен для управления светодиодом с помощью микроконтроллера Arduino. В функции setup() устанавливается “пин” A1 в качестве выходного “пина”, что означает, что он будет использоваться для управления другими устройствами.

В функции loop() происходит основной цикл программы. Сначала устанавливаем “пин” A1 в состояние HIGH (высокий уровень), что включает подключенное к нему устройство. Затем мы ждем 500 миллисекунд с помощью функции delay(). После этого устанавливаем “пин” A1 в состояние LOW (низкий уровень), что выключает устройство, и снова ждем 500 миллисекунд. Этот процесс повторяется в цикле бесконечно. Таким образом, данный код будет мигать подключенным к “пину” A1 устройством с интервалом 500 миллисекунд (0,5 секунды) включения и 500 миллисекунд выключения.

```
// функция setup вызывается 1 раз и производит
настройку
void setup() {
    pinMode(A1, OUTPUT); // устанавливаем “пин” A1 в
 режим вывода
}
```

```
// функция цикла программы, она бесконечно работает, пока контроллер включен
void loop() {
    digitalWrite(A1, HIGH); // Устанавливаем “пин” A1 в высокий уровень
    delay(500); // функция ожидания 500 миллисекунд
    digitalWrite(A1, LOW); // Устанавливаем “пин” A1 в низкий уровень
    delay(500); // функция ожидания 500 миллисекунд
}
```

Для регулировки яркости светодиода используется ШИМ сигнал. Для этого светодиод подключается к одному из ШИМ “пинов” (на плате Nano это D3, D5, D6, D9, D10, D11), настроим “пин” как выход и будем управлять яркостью при помощи ШИМ-сигнала.

В функции setup() устанавливаем “пин” 3 в качестве выходного “пина”, что означает, что мы будем использовать его для управления другими устройствами.

В функции loop() происходит основной цикл программы. Мы используем функцию analogWrite() для установки аналогового сигнала на “пин” 3 с определенным значением. Затем мы ждем 500 миллисекунд с помощью функции delay(). После этого мы повторяем этот процесс, устанавливая на “пин” 3 другие значения аналогового сигнала (100, 150, 200 и 255) и снова ждем 500 миллисекунд.

Таким образом, данный код будет периодически изменять яркость устройства, подключенного к “пину” 3, с использованием аналоговых значений. Каждые 500 миллисекунд значения аналогового сигнала будут меняться от самого низкого (10) до самого высокого (255). Этот процесс будет повторяться бесконечно в цикле loop().

```
// функция setup вызывается 1 раз и производит настройку
void setup() {
pinMode(3, OUTPUT); // устанавливаем “пин” 3 в режим вывода
}
```

```
// функция цикла программы, она бесконечно работает, пока контроллер включен
void loop() {
analogWrite(3, 10);  // устанавливаем “пин” 3 с аналоговым значением 10
delay(500); // функция ожидания 500 миллисекунд
analogWrite(3, 100); // устанавливаем “пин” 3 с аналоговым значением 100
delay(500);
analogWrite(3, 150); // устанавливаем “пин” 3 с аналоговым значением 150
delay(500);
analogWrite(3, 200); // устанавливаем “пин” 3 с аналоговым значением 200
delay(500);
analogWrite(3, 255); // устанавливаем “пин” 3 с аналоговым значением 255
delay(500);
}
```

Рассмотрим схему подключения с потенциометром, которая представлена на рисунке 14.3. Подключим потенциометр на A0 и попробуем регулировать яркость с его помощью. В функции setup() устанавливаем “пин” 3 “в качестве выходного “пина”, что означает, что он будет использоваться для управления другими устройствами.

В функции loop() происходит основной цикл программы. Мы используем функцию analogRead() для чтения аналогового значения с “пина” 0 (A0). Это значение находится в диапазоне от 0 до 1023, делим его на 4, чтобы получить значение в диапазоне от 0 до 255, который является диапазоном значений для функции analogWrite(). Затем устанавливаем полученное аналоговое значение на “пин” 3 с помощью analogWrite(). После этого мы ждем 100 миллисекунд с помощью функции delay().

Таким образом, данный код будет считывать аналоговое значение с “пина” A0, делить его на 4 и устанавливать полученное значение на “пин” 3. Это позволяет управлять яркостью устройства, подключенного к “пину” 3, на основе

аналогового значения с “пина” A0. Этот процесс будет повторяться бесконечно в цикле loop().

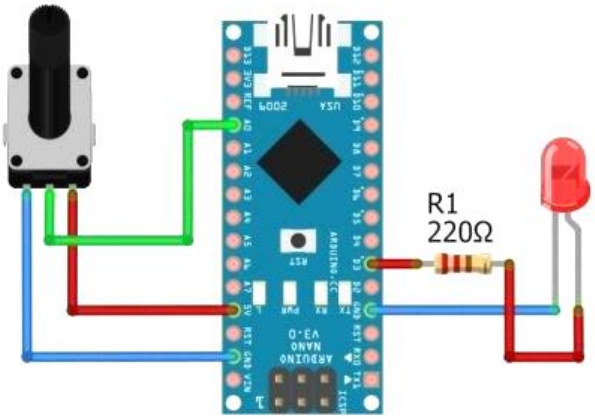

Рис. 14.3. Схема подключения потенциометра

```
void setup() {
    pinMode(3, OUTPUT); // устанавливаем “пин” 3 в
 режим вывода
}
void loop() {
    // читаем аналоговое значение “пина” 0, делим на 4
 и отправляем его на “пин” 3
    analogWrite(3, analogRead(0) / 4);
    delay(100); // задержка 100 миллисекунд
}
```

В данном занятии представлены основы управления светодиодом с использованием Arduino. Было показано, как подключить светодиод к плате Arduino, установить правильный режим “пина” и управлять его состоянием.

Занятие №15. Знакомство с кнопкой и пьезодинамиком

Кнопка и пьезодинамик - это два устройства, используемые в электронных системах для взаимодействия с пользователем. *Кнопка*, также известная как переключатель, представляет собой устройство, которое при нажатии замыкает электрическую цепь. Обычно она имеет два состояния - "включено" и "выключено". Кнопки могут использоваться для включения и выключения устройств, выбора опций в меню или для выполнения других задач, связанных с управлением устройством.

Пьезодинамик - это устройство, которое преобразует электрический сигнал в звуковые волны. Он состоит из керамического элемента, который при подаче электрического сигнала начинает вибрировать, создавая звуковые волны. Пьезодинамик может использоваться для создания звуковых сигналов, уведомлений или для проигрывания музыки.

Оба устройства могут использоваться вместе для создания интерактивных электронных систем, таких как игровые устройства или контроллеры управления. Например, кнопка может использоваться для выбора опции в меню, а затем пьезодинамик может использоваться для проигрывания звукового эффекта для подтверждения выбора.

Кнопки и пьезодинамики могут использоваться в различных типах электронных систем. В дополнение к игровым устройствам и контроллерам, кнопки и пьезодинамики могут использоваться в автомобилях для управления различными функциями, такими как управление системой зажигания или радио, или в бытовой технике, такой как стиральные машины и микроволновые печи.

Существует много различных типов кнопок и пьезодинамиков, которые могут быть использованы в электронных системах. Некоторые кнопки могут быть механическими, где контакты внутри устройства замыкаются при нажатии на кнопку. Другие могут быть капацитивными, где электрический сигнал обнаруживает изменение емкости в устройстве, когда палец приближается к поверхности кнопки. Аналогично, существуют много типов пьезодинамиков, которые могут варьироваться по размеру, частоте, мощности и другим параметрам.

Важно понимать, как правильно подключать кнопки и пьезодинамики к электронной системе, чтобы они работали правильно и не вызывали повреждения других компонентов. Обычно кнопки и пьезодинамики могут быть подключены к микроконтроллерам, Arduino или другим контроллерам через специальные контакты или порты, которые позволяют управлять входными и выходными сигналами.

Кроме того, кнопки и пьезодинамики могут управляться программно с помощью языков программирования, таких как Си, Python, или JavaScript. Например, при нажатии кнопки можно запустить определенную функцию, а при необходимости пьезодинамики могут проигрывать звуковые эффекты, создавая интерактивный опыт для пользователя.

Для управления пьезодинамиком могут использоваться различные алгоритмы и методы, которые определяют, какой звуковой сигнал должен быть проигран. Например, для проигрывания музыки на пьезодинамике можно использовать алгоритм преобразования частоты, который переводит ноты музыкальной композиции в соответствующие частоты для проигрывания на пьезодинамике.

Некоторые кнопки и пьезодинамики могут быть управляемы с помощью дополнительных элементов управления, таких как потенциометры, которые позволяют изменять частоту звукового сигнала или уровень громкости.

В целом, кнопки и пьезодинамики являются важными компонентами многих электронных систем, которые обеспечивают управление и обратную связь между системой и пользователем. Их правильное использование может повысить эффективность и удобство использования системы, а неправильное использование может привести к неисправностям или повреждениям.

Кроме простых кнопок и пьезодинамиков, существуют и более сложные типы этих компонентов. Например, сенсорные кнопки позволяют обнаруживать касание без физического нажатия на кнопку. Они используются в различных электронных устройствах, таких как смартфоны, планшеты, сенсорные экраны и другие.

Кроме того, существуют и другие типы пьезоэлектрических устройств, таких как пьезокерамика и пьезомоторы. Пьезокерамика используется для создания колебаний внутри электронных устройств и имеет широкий спектр применений, от медицинских устройств до промышленных процессов. Пьезомоторы используются для управления движением механизмов, их малый размер и высокая точность позволяют использовать их во многих областях, таких как оптические устройства и микромеханизмы.

Наконец, важно помнить о безопасности при использовании кнопок и пьезодинамиков. Некоторые типы кнопок могут быть подключены к сети переменного тока, поэтому важно следить за правильным подключением, чтобы избежать возможности поражения электрическим током. Кроме того, громкий звук, создаваемый пьезодинамиком, может повредить слух, поэтому важно контролировать уровень громкости звуковых эффектов и использовать защитные наушники при работе с громкими звуками. На рисунке 15.1 изображены сенсорная кнопка, пьезодинамик и тактовая кнопка.

Рис. 15.1. Сенсорная кнопка, пьезодинамик, тактовая кнопка

Ниже представлен пример программы функцией tone()и noTone().

```
//"Пин", к которому подключен пьезодинамик.
int piezoPin = 3;
void setup() {
  pinMode(piezoPin, OUTPUT); // установили "пин" 3 в режим вывода
}
void loop() {
  /*Функция принимает три аргумента
    1) Номер "пина"
    2) Частоту в герцах, определяющую высоту звука
    3) Длительность в миллисекундах.
  */
  tone(piezoPin, 1000, 500); // Звук прекратится через 500 мс, о программа останавливаться не будет!

  /* Вариант без установленной длительности   */
  tone(piezoPin, 2000); // Запустили звучание
  delay(500); // задержка 500 миллисекунд
  noTone(); // Остановили звучание
}
```

Схема подключения пъезодинамика для примера выглядит следующим образом:

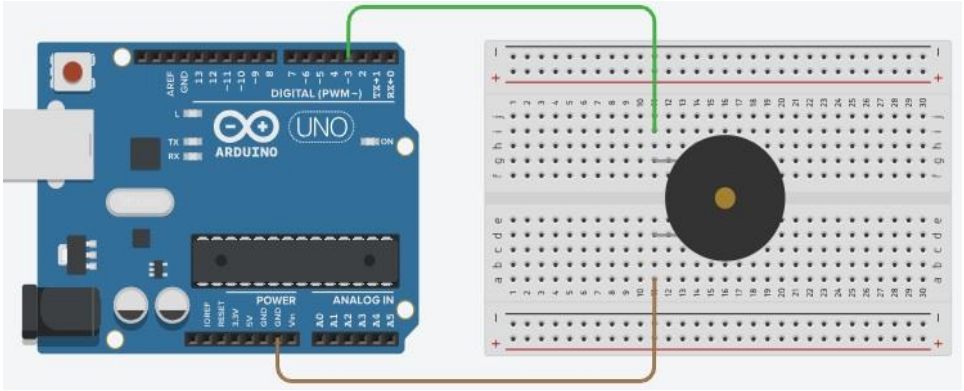

Рис 15.2. Подключение пъезодинамика к 3 “пину” Arduino

На рисунке 15.3 представлена схема подключения кнопки на “пин” D3 (и GND):

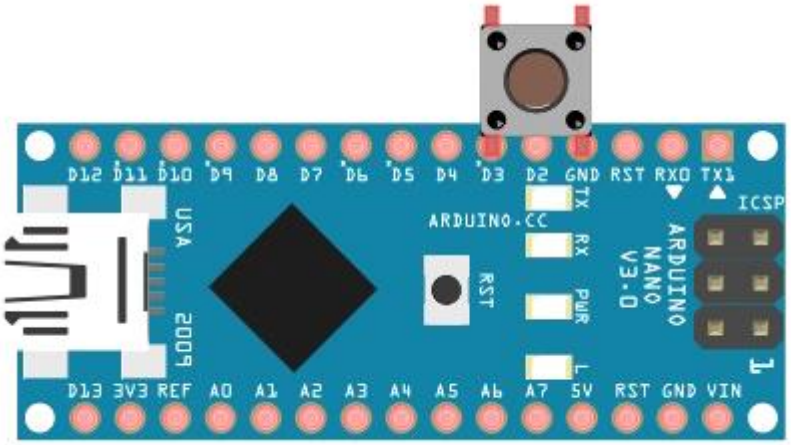

Рис 15.3. Подключение кнопки к 3 “пину” Arduino

```
void setup() {
    Serial.begin(9600);  //  Включаем  ввод\вывод  со скорость 9600 бит в сек
    pinMode(3, INPUT_PULLUP); // устанавливаем “пин” 3 в режим ввода с подтягивающим резистором
}
void loop() {
 // выведет 0, если кнопка нажата
 // и 1, если нет
```

```
    Serial.println(digitalRead(3));
    delay(10);
}
```

На данном занятии перечислены базовые аспекты управления кнопкой и пьезодинамиком с использованием Arduino. Были представлены варианты подключения их к плате Arduino, установки режима работы “пинов”.

В заключение следует отметить, что кнопки и пьезодинамики являются важными компонентами электронных систем, обеспечивающими управление и обратную связь между системой и пользователем. Существуют различные типы кнопок и пьезодинамиков, включая сенсорные кнопки и пьезокерамику, которые находят широкое применение в различных областях. Правильное использование этих компонентов может повысить эффективность и удобство использования системы, а неправильное использование может привести к неисправностям или повреждениям. Поэтому важно следовать инструкциям по установке и использованию этих компонентов и соблюдать меры безопасности, чтобы избежать возможных проблем.

Занятие №16. Знакомство с потенциометром

Потенциометр - это электронный компонент, который позволяет изменять сопротивление в цепи. Проще говоря, он позволяет управлять тем, сколько электрического тока пропускается через цепь. Потенциометры широко применяются в электронике для регулирования яркости света, громкости звука, настройки температуры и других параметров. Они также используются для контроля положения двигателей и других механических устройств.

Например, если у вас есть светодиод, который светится с определенной яркостью, вы можете использовать потенциометр, чтобы изменить яркость света. Когда вы вращаете ручку потенциометра, вы меняете сопротивление, что влияет на количество электричества, проходящего через цепь и соответственно на яркость света светодиода.

Потенциометры бывают разных типов: линейные и логарифмические. Линейные потенциометры имеют равномерное изменение сопротивления при вращении ручки, а логарифмические - изменяют сопротивление нелинейно, что позволяет более точно настраивать сопротивление в заданном диапазоне значений.

Линейные и логарифмические потенциометры - это два типа потенциометров, которые используются для регулировки сопротивления в

электрических цепях. Они отличаются формой изменения сопротивления в зависимости от положения ручки потенциометра.

Линейные потенциометры имеют линейную зависимость изменения сопротивления от положения ручки. Это означает, что если ручка перемещается на 50% длины хода потенциометра, то значение сопротивления изменится на 50% от максимального значения сопротивления. Таким образом, изменение сопротивления линейного потенциометра пропорционально изменению положения ручки. Линейные потенциометры могут использоваться для регулирования яркости светодиодов, скорости двигателей и других параметров, которые имеют линейную зависимость.

Логарифмические потенциометры, напротив, имеют логарифмическую зависимость изменения сопротивления от положения ручки. Это означает, что изменение сопротивления логарифмического потенциометра пропорционально логарифму изменения положения ручки. В результате при перемещении ручки с одной крайности на середину, изменение сопротивления будет небольшим, а при дальнейшем перемещении к другой крайности, изменение сопротивления будет значительным. Логарифмические потенциометры обычно используются для регулирования громкости в звуковых устройствах, так как человеческое восприятие громкости изменяется логарифмически в зависимости от мощности звука.

На практике, логарифмические потенциометры могут иметь маркировку "A" или "Audio", а линейные - "B" или "Linear". Однако, это не является универсальным правилом, и всегда лучше проверять документацию производителя или измерять сопротивление потенциометра в различных положениях ручки для определения типа.

Для работы с Arduino и потенциометром необходимо подключить потенциометр к аналоговому входу микроконтроллера. Аналоговые входы позволяют измерять изменение напряжения в заданном диапазоне значений. Например, если вы подключите потенциометр к аналоговому входу Arduino, то сможете измерить изменение сопротивления при вращении ручки. Для этого в Arduino есть функция analogRead(), которая позволяет считывать значения с аналоговых входов.

В качестве примера представлена программа для работы с потенциометром на Arduino:

```
int potPin = A0; // определяем аналоговый вход для потенциометра
int ledPin = 13; // определяем цифровой выход для светодиода
void setup() {
    pinMode(ledPin, OUTPUT); // устанавливаем выход
для светодиода как выход
}
void loop() {
    int val = analogRead(potPin); // считываем
    значение с аналогового входа
    val = map(val, 0, 1023, 0, 255); // масштабируем значение до диапазона яркости светодиода
    analogWrite(ledPin, val); // устанавливаем яркость светодиода
}
```

В этом примере потенциометр подключается к аналоговому входу A0 и светодиод к цифровому выходу 13. В функции loop() мы считываем значение с потенциометра, масштабируем его до диапазона яркости светодиода и устанавливаем яркость светодиода с помощью функции analogWrite(). Этот код позволяет изменять яркость светодиода, в зависимости от положения ручки потенциометра. Также, потенциометры можно использовать для управления сервоприводами, которые используются в робототехнике и автоматических системах управления. Сервоприводы позволяют управлять положением объектов, например, “роботических рук” или моделей самолетов.

Для управления сервоприводами с помощью потенциометра на Arduino можно использовать библиотеку Servo. Ниже представлен пример кода для управления сервоприводом с помощью потенциометра.

```
#include <Servo.h>
int potPin = A0; // определяем аналоговый вход для потенциометра
int servoPin = 9; // определяем “пин” для сервопривода
```

```
Servo myservo; // создаем объект для управления
сервоприводом
void setup() {
    myservo.attach(servoPin); // прикрепляем
 сервопривод к “пину” 9
}
void loop() {
    int val = analogRead(potPin); // считываем
    значение с потенциометра
    val = map(val, 0, 1023, 0, 180); // масштабируем
    значение до диапазона угла поворота сервопривода
    myservo.write(val); // устанавливаем угол
    поворота сервопривода
    delay(15); // задержка для стабилизации
    сервопривода
 }
```

В этом примере потенциометр подключается к аналоговому входу A0 и сервопривод к “пину” 9. В функции loop() мы считываем значение с потенциометра, масштабируем его до диапазона угла поворота сервопривода и устанавливаем угол поворота сервопривода с помощью функции write() объекта myservo. Этот код позволяет изменять угол поворота сервопривода в зависимости от положения ручки потенциометра.

Таким образом, потенциометры являются полезными компонентами в электронике и могут использоваться для регулирования различных параметров, управления механическими устройствами и многих других задач. Программирование с использованием потенциометров на Arduino может быть интересным способом изучения основ электроники и робототехники.

В данном занятии были перечислены основы работы потенциометра с использованием Arduino. Рассказано, как подключить потенциометр к плате Arduino, установить правильный режим “пина” и управлять его состоянием с помощью вращения ручки потенциометра.

Занятие №17. Знакомство с датчиком расстояния

Во многих роботизированных изделиях применяют *датчики расстояния*, которые являются важной составляющей функционирования устройства. В качестве датчика расстояния можно использовать ультразвуковой дальномер модуль HC-SR04 (рис. 17.1).

Рис. 17.1. Ультразвуковой дальномер модуль HC-SR04

Ультразвуковой дальномер модуль HC-SR04 — это помещенные на одну плату приемник и передатчик ультразвукового сигнала.

HC-SR04 использует принцип эхолокации для измерения расстояния. Датчик генерирует ультразвуковой сигнал, который отражается от объекта и затем считывается приемником. Используя знания скорости распространения ультразвуковых волн и время задержки между сигналами, можно точно определить расстояние до объекта. На рисунке 17.2 представлен принцип работы дальномера.

Данный датчик имеет ряд преимуществ перед инфракрасными дальномерами, так как не подвержен влиянию источников света или цвета объекта, но может испытывать сложности при измерении расстояний до тонких или пушистых объектов. Ультразвуковые датчики находят применение в различных областях, таких как машиностроение, сельское хозяйство и автоматизированные производства, где требуется точное измерение расстояния до объекта.

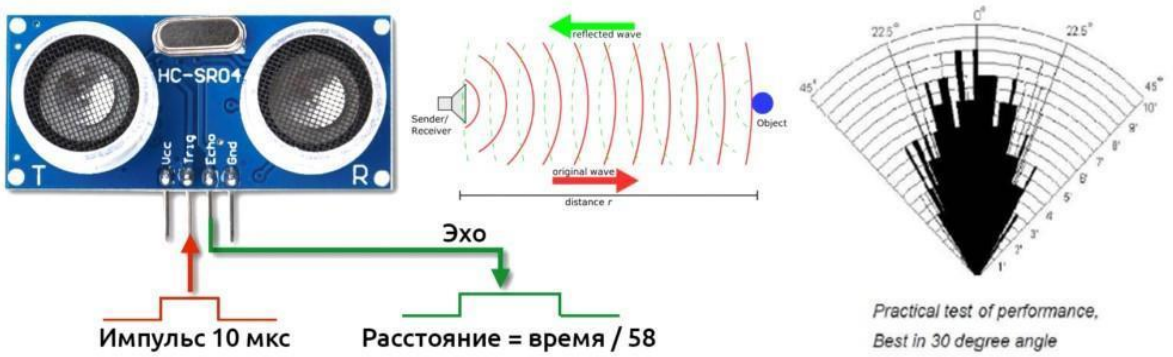

Рис. 17.2. Принцип работы дальномера

Характеристики модуля HC-SR04:

- питание: 5V;
- рабочий ток: 15 мА;
- звуковая частота: 40 кГц;
- угол измерения: 15 градусов;
- диапазон измерения: 2 см … 4 м.
- Точность: ~1 мм при грамотной фильтрации.

На рисунке 17.3 представлена схема подключения дальномера. Подключать дальномер к питанию можно к цифровым “пинам”.

С датчиком можно работать без библиотек, стандартными средствами Arduino. Но есть и библиотеки:

HC-SR04 – можно установить по названию HC-SR04 из менеджера библиотек (автор Dirk Sarodnick);

NewPing – можно установить по названию NewPing из менеджера библиотек.

В данном коде используется датчик расстояния HC-SR04 для измерения расстояния и вывода его в монитор последовательной связи.

С помощью директив #define определены константы HC_TRIG и HC_ECHO, которые указывают “пины”, на которых подключены соответствующие выводы датчика.

В функции setup() устанавливаются режимы “пинов”: “пин” HC_TRIG устанавливается в режим вывода (OUTPUT), а “пин” HC_ECHO - в режим ввода (INPUT). Также инициализируется последовательная связь с компьютером через

функцию Serial.begin(), чтобы можно было выводить данные в монитор последовательной связи.

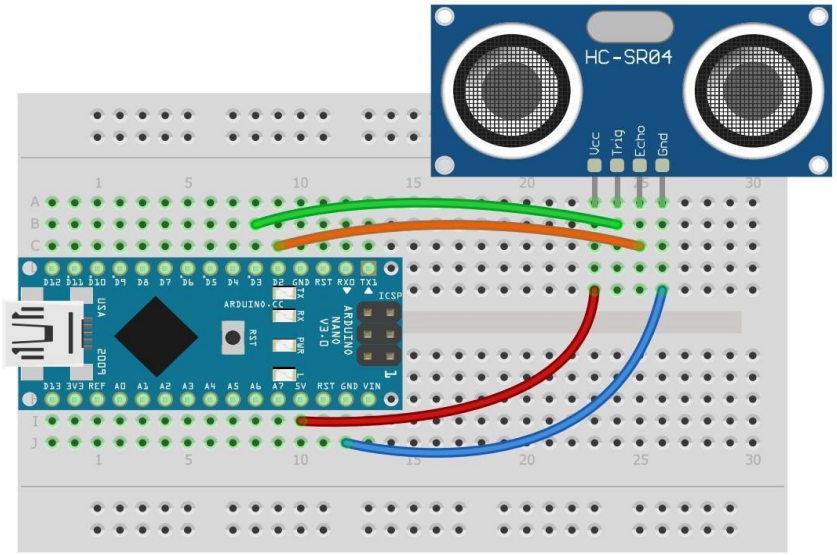

Рис. 17.3. Схема подключения дальномера

В функции loop() происходит основной цикл программы. Сначала вызывается функция getDist(), которая возвращает расстояние в виде значения типа float. Затем это значение выводится в монитор последовательной связи с помощью функции Serial.println(). После этого происходит задержка в 50 миллисекунд с помощью функции delay().

В конце кода определена функция getDist(), которая выполняет измерение расстояния с помощью датчика HC-SR04. Сначала генерируется короткий импульс на “пине” HC_TRIG с помощью функций digitalWrite() и delayMicroseconds(). Затем измеряется время ответного импульса на “пине” HC_ECHO с помощью функции pulseIn(). После этого расстояние вычисляется на основе измеренного времени и возвращается из функции.

```
// “пины”
#define HC_TRIG 3
#define HC_ECHO 2
void setup() {
    Serial.begin(9600);// для связи
    pinMode(HC_TRIG, OUTPUT); // trig выход
    pinMode(HC_ECHO, INPUT); // echo вход
}
```

```
void loop() {
    float dist = getDist(); // получаем расстояние
    Serial.println(dist); // выводим delay(50);
}
// сделаем функцию для удобства
float getDist() {
    // импульс 10 мкс
    digitalWrite(HC_TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(HC_TRIG, LOW);
    // измеряем время ответного импульса
    uint32_t us = pulseIn(HC_ECHO, HIGH);
    // считаем расстояние и возвращаем
    return (us / 58.2);
}
```

Чтобы датчик не ловил “эхо” от самого себя – его не рекомендуется опрашивать чаще чем 1 раз в 30 мс.

Можно протестировать датчик передвигая перед ним руку или любой предмет. На рисунке 17.4а представлен график зависимости расстояния от времени с результатами первого теста, его можно открыть в среде разработки по следующему пути: (Инструменты -> Порт: “COM6”).

Повторим тот же самый тест, но отсеивая данные при помощи простого экспоненциального фильтра. Для этого добавим в программный код аргумент distFilt и изменим функцию loop()

```
float distFilt = 0;
void loop() {
    float dist = getDist(); // получаем расстояние
    distFilt += (dist - distFilt) * 0.2; // фильтруем
    Serial.println(distFilt); // выводим delay(50);
}
```

На рисунке 17.4б представлена график зависимости расстояния от времени с результатами второго теста.

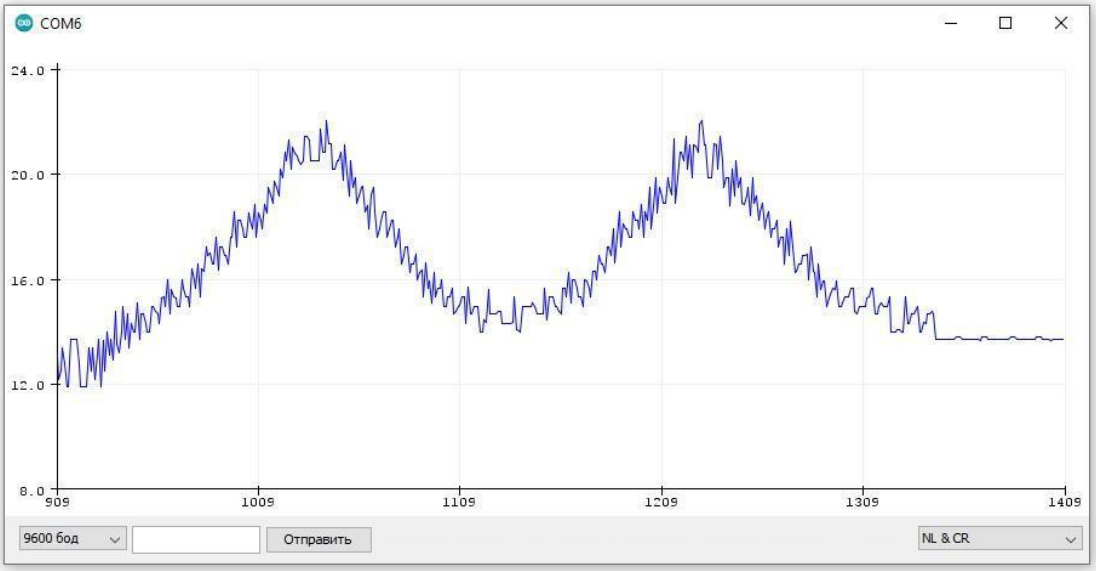

Рис. 17.4а. Результат первого теста

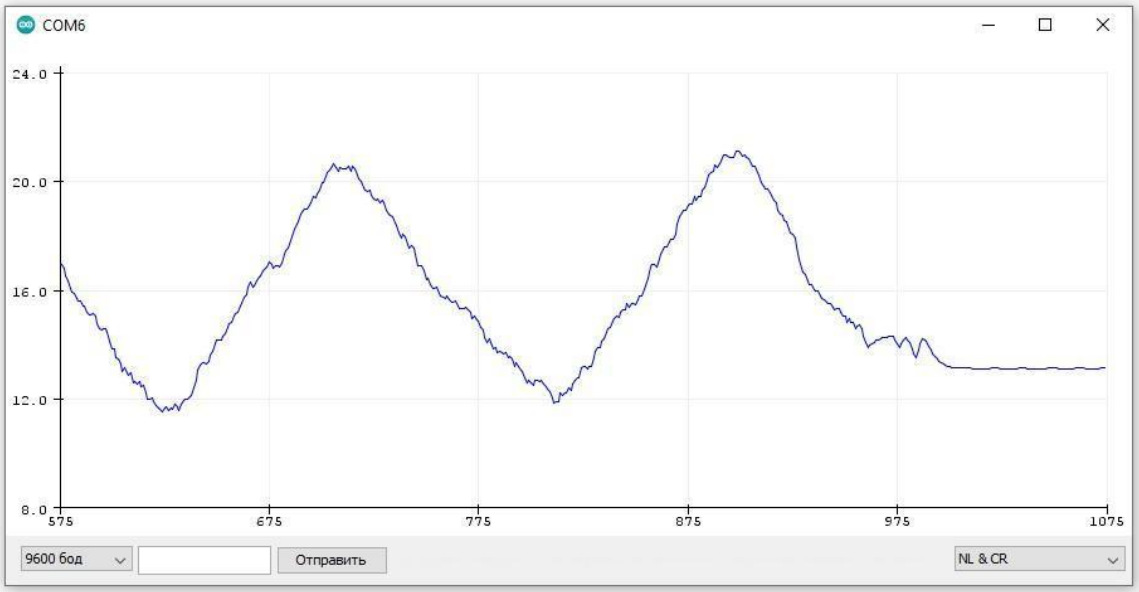

Рис. 17.4б. Результат второго теста

Наглядно видно, что качество получаемых данных после фильтрации улучшилось. Чтобы вывести график на экран можно воспользоваться комбинацией клавиш Ctrl+Shift+L, или выбрать вкладку Tools - > Serial Plotter.

Датчики расстояния бывают разные, но принцип работы остается неизменным. На рисунке 17.5 представлен инфракрасный датчик расстояния Sharp GP2Y0A21YK0F.

Sharp GP2Y0A21YK0F это модуль ближней инфракрасной датчиковой системы, который широко используется в робототехнике и автоматизации. Он способен обнаруживать объекты в своем поле зрения, используя инфракрасное излучение, и предоставляет аналоговый выходной сигнал, соответствующий расстоянию до обнаруженного объекта.

GP2Y0A21YK0F имеет диапазон обнаружения от 10 см до 80 см, а также угол обнаружения в 23 градуса. Аналоговый выходной сигнал сенсора пропорционален расстоянию до обнаруженного объекта, с диапазоном напряжения от 0,4 В до 2,75 В. Модуль относительно прост в использовании, имеет простой 3-контактный интерфейс и может питаться от 4,5 В до 5,5 В постоянного тока. Один из факторов, на который следует обратить внимание при использовании датчика GP2Y0A21YK0F, это то, что его выходной сигнал зависит от отражательных свойств обнаруженного объекта. Сенсор наиболее точен при обнаружении объектов с плоской, нерефлективной поверхностью, а его точность может снижаться при обнаружении объектов с высокой отражательной способностью или нерегулярными поверхностями. В целом, Sharp GP2Y0A21YK0F - это популярный и надежный датчик.

Рис. 17.5. Инфракрасный датчик расстояния Sharp GP2Y0A21YK0F

На данном занятии было рассказано об основах работы датчика расстояния с использованием Arduino, как подключить датчик расстояния к плате Arduino, получить с него дистанцию до объекта.

Занятие №18. Знакомство с сервоприводом

Сервопривод - это устройство, которое используется для управления движением объекта с высокой точностью и контролем позиции. Он состоит из

двигателя, электроники управления и обратной связи, которая сообщает системе точную позицию объекта. На рисунке 18.1 изображено устройство сервопривода.

Сервоприводы широко используются в робототехнике, автоматизации и других областях, где требуется точное управление движением. Они могут быть использованы для поворота и наклона камеры, перемещения робота или другого объекта, регулирования угла поворота колеса и т.д. Для работы с сервоприводом необходимо подключить его к источнику питания и управляющему устройству, например, микроконтроллеру.

Управление сервоприводом происходит путем отправки ему сигналов управления в виде импульсов. Ширина импульса определяет угол поворота сервопривода. Для управления сервоприводом можно использовать специальные библиотеки программного обеспечения, которые позволяют задавать точное положение сервопривода, скорость и ускорение движения.

Рис. 18.1. Устройство сервопривода

При использовании сервопривода необходимо учитывать его технические характеристики, например, максимальный угол поворота, максимальное ускорение и ток потребления. Некоторые сервоприводы могут также иметь ограничения по скорости движения и точности позиционирования. Кроме того, при работе с сервоприводом необходимо обеспечить достаточное охлаждение, чтобы избежать перегрева и повреждения устройства.

При выборе сервопривода необходимо учитывать требования к точности позиционирования и скорости движения объекта, который будет управляться. Также необходимо учитывать максимальный угол поворота и нагрузку, которую может выдерживать сервопривод.

Существуют различные типы сервоприводов, включая аналоговые и цифровые. Аналоговые сервоприводы работают с постоянным током и имеют меньшую точность и скорость, чем цифровые. Цифровые сервоприводы работают с переменным током и имеют более высокую точность и скорость, а также возможность настройки различных параметров управления.

Для управления сервоприводами также могут использоваться различные интерфейсы, например, PWM (широтно-импульсная модуляция) или сигналы RS-232. Интерфейс PWM позволяет управлять сервоприводом путем изменения ширины импульса, а сигналы RS-232 используются для передачи данных между устройствами.

Сервоприводы также могут использоваться в комбинации с другими устройствами, например, с датчиками, чтобы обеспечить точное позиционирование объекта. Кроме того, существуют специальные системы управления сервоприводами, которые позволяют управлять несколькими сервоприводами одновременно.

Для управления сервоприводом существует несколько основных методов управления: управление положением, управление скоростью и управление моментом.

Управление положением осуществляется путем задания угла поворота сервопривода. Обычно для этого используется широтно-импульсная модуляция (PWM). Ширина импульса определяет угол поворота сервопривода. Ширина импульса может быть изменена в диапазоне от 1 мс до 2 мс, где 1 мс соответствует минимальному углу поворота, а 2 мс максимальному углу поворота.

Управление скоростью осуществляется путем изменения скорости вращения сервопривода. Для этого используется модуляция широты импульса (PPM). Ширина импульса соответствует скорости вращения сервопривода.

Управление моментом осуществляется путем задания момента, который необходимо развить сервоприводом. Для этого используется управляющий сигнал, который задает момент, который необходимо развить.

Для управления сервоприводом можно использовать микроконтроллеры или специальные устройства управления сервоприводами. Для управления несколькими сервоприводами можно использовать мультиплексоры или декодеры.

При использовании сервопривода необходимо учитывать также его энергопотребление и тепловыделение. При больших нагрузках или длительной работе сервопривод может нагреться и выйти из строя. Также для обеспечения более точного позиционирования объекта необходимо использовать обратную связь. Обратная связь позволяет контролировать положение объекта и корректировать его положение при необходимости.

Одной из важных технических характеристик сервопривода является его максимальный угол поворота. Обычно этот угол составляет от 90 до 180 градусов. Также необходимо учитывать максимальную скорость и максимальный момент, которые может развивать сервопривод. Для выбора подходящего сервопривода необходимо также учитывать характеристики управляемого объекта и требования к точности позиционирования. Например, для управления легким объектом, таким как модельный самолет, может быть достаточно использовать сервопривод малой мощности, а для управления большим промышленным роботом потребуется более мощный сервопривод. Для обеспечения более точного управления и уменьшения ошибок позиционирования можно использовать различные методы обратной связи. Например, можно использовать энкодеры (измерительный преобразователь угла поворота вращающегося объекта в цифровые или аналоговые сигналы, которые позволяют определить угол его поворота) для измерения угла поворота сервопривода и обратной связи с микроконтроллером для корректировки позиции объекта. Также можно использовать гироскопы (устройства, используемые для измерения или поддержания ориентации и угловой скорости) и акселерометры (прибор для измерения ускорения, который работает как датчик изменения положения устройства в пространстве) для определения положения объекта и его ориентации в пространстве.

Некоторые модели сервоприводов также имеют дополнительные функции, такие как программирование пути движения или сохранение позиции после отключения питания.

Важно также учитывать электрические характеристики сервопривода, такие как напряжение питания и потребляемый ток. Например, для использования

сервопривода с микроконтроллером может потребоваться дополнительное питание или стабилизатор напряжения для обеспечения стабильности работы.

Еще одной важной характеристикой сервопривода является его управляющий сигнал. Обычно сервоприводы используют сигнал ШИМ (PWM - Pulse Width Modulation), который представляет собой сигнал с изменяющейся шириной импульса. Частота и ширина импульсов определяют угол поворота сервопривода и его скорость. Например, для поворота сервопривода на угол 90 градусов может быть использован импульс длиной 1,5 мс, а для поворота на угол 180 градусов - импульс длиной 2 мс.

Важно также учитывать ограничения на максимальную длину импульса, которую поддерживает сервопривод, а также на минимальную длину периода между импульсами. Некоторые сервоприводы могут иметь дополнительные ограничения, например, на максимальную длительность импульса или на частоту обновления сигнала управления. Также важно учитывать противодействие механической нагрузки на сервопривод. Если объект, который управляется сервоприводом, имеет высокую инерцию или сопротивление, то это может привести к перегрузке сервопривода и повреждению его механизма. Для уменьшения такого риска можно использовать редукторы или другие механические устройства, которые снижают нагрузку на сервопривод. Наконец, для использования сервопривода необходимо иметь управляющую электронику, которая обеспечивает генерацию сигнала ШИМ и корректировку его параметров для управления позицией и скоростью сервопривода. Также может потребоваться дополнительная электроника для обеспечения питания сервопривода и защиты от перегрузок и коротких замыканий.

Данное занятие посвящено основам работы сервопривода с использованием Arduino. Рассказано, как подключить сервопривод к плате Arduino и управлять состоянием сервопривода. В целом, использование сервоприводов требует достаточно серьезного подхода и учета многих технических характеристик и условий эксплуатации. Однако благодаря своей универсальности и широкому спектру применения, сервоприводы остаются незаменимым элементом в автоматизации.

Занятие №19. Знакомство с мотором

Мотор - устройство, преобразующее какой-либо вид энергии в механическую работу. Моторы, работающие на базе Arduino, представляют собой важный элемент многих проектов, особенно в области робототехники. Arduino это открытая платформа с открытым исходным кодом,

которая позволяет создавать различные электронные проекты, включая проекты, связанные с управлением моторами. На данном занятие рассмотрим, как моторы работают с Arduino, а также представим несколько примеров кода для управления моторами.

Arduino может управлять моторами посредством использования различных методов, таких как управление постоянным током (DC), управление шаговым двигателем (Stepper) и управление сервомоторами. Каждый из этих методов управления имеет свои особенности и требует разных подходов к программированию.

Управление моторами постоянного тока является наиболее простым и распространенным методом управления моторами с использованием Arduino. Для этого используется схема мост H (H-bridge), который позволяет изменять направление движения и скорость мотора. Эта схема состоит из четырех транзисторов, которые могут управлять током, который идет через мотор. Для управления с помощью Arduino необходимо подключить четыре вывода: два для управления направлением и два управления скоростью. Например, в драйвере L298N, выводы управления направлением называются IN1, IN2, а выводы управления скоростью EN1, EN2.

Пример кода для управления мотором постоянного тока с помощью мостового драйвера моторов L298N:

```
// объявляем переменные для пинов
int enA = 9; int in1 = 8; int in2 = 7;
void setup() {
      pinMode(enA, OUTPUT);
      pinMode(in1, OUTPUT);
      pinMode(in2, OUTPUT);
}
void loop() {
     digitalWrite(in1, HIGH);
     digitalWrite(in2, LOW);
     analogWrite(enA, 200);
}
```

В этом примере используется мост H для управления мотором. При выполнении программы мотор будет вращаться в одном направлении со

скоростью 200. Чтобы изменить направление вращения, нужно изменить значения в переменных in1 и in2.

Другим способом управления моторами с Arduino является управление шаговым двигателем (Stepper). Шаговый двигатель может быть управляемым или неуправляемым. Неуправляемый шаговый двигатель не может изменять скорость вращения, а управляемый двигатель может контролировать положение и скорость вращения. Управление шаговым двигателем осуществляется посредством последовательного управления фазами двигателя. Чтобы управлять шаговым двигателем с помощью Arduino, необходимо подключить его к микроконтроллеру, соединяя “пины” Arduino с входами драйвера и использовать специальную библиотеку, такую как Stepper.h. Пример кода для управления шаговым двигателем с помощью драйвера ULN2003. “Пины” 8, 9, 10, 11 соединяются со входами IN1-IN4 драйвера, а выходы соединяются с обмотками шагового двигателя:

```
#include <Stepper.h> // добавляем заголовочный файл
для работы с шаговым двигателем

const int stepsPerRevolution = 200;

Stepper myStepper(stepsPerRevolution, 8, 9, 10,

11);

void setup() {

    myStepper.setSpeed(60); // устанавливаем угловую
 скорость вращения

}

void loop() {

    myStepper.step(stepsPerRevolution); // выполнить
 движение двигателем на 200 шагов по часовой

    delay(500); // задержка 500 миллисекунд
```

```
    myStepper.step(-stepsPerRevolution); // выполнить
 движение двигателем на 200 шагов против часовой
 стрелки
    delay(500);
}
```

В этом примере используется библиотека Stepper.h, которая позволяет управлять шаговым двигателем. Метод setSpeed() устанавливает скорость вращения двигателя. В цикле программы используется функция step(), который заставляет двигатель вращаться определенное количество шагов. В данном случае двигатель сначала вращается на 200 шагов в одном направлении, затем на те же 200 шагов в обратном направлении.

Третий способ управления моторами с Arduino - это управление сервомоторами. Сервомоторы используются для точного позиционирования и имеют ограниченный угол вращения. Управление сервомоторами происходит путем отправки импульса с заданным временем длительности.

Пример кода для управления сервомотором:

```
#include <Servo.h>
Servo myservo;
void setup() {
    myservo.attach(9);
}
void loop() {
    myservo.write(0);
    delay(1000);
    myservo.write(90);
    delay(1000);
    myservo.write(180);
    delay(1000);
}
```

На данном занятии были озвучены основы работы мотора с использованием Arduino. Было определено, как подключить мотор к плате Arduino, установить правильный режим “пинов” и управлять его мощностью.

При работе с моторами необходимо учитывать их мощность и электрические характеристики, чтобы избежать повреждения мотора или других компонентов системы. Однако, с использованием платформы Arduino, управление моторами становится более простым и гибким, что позволяет создавать разнообразные проекты на базе Arduino.

Занятие №20. Знакомство с датчиком цвета

Датчик цвета - это электронное устройство, которое используется для измерения цветовых характеристик объектов. Он может быть полезен во многих областях, включая промышленность, робототехнику, медицину, пищевую промышленность, дизайн и даже искусство. На рисунке 20.1 представлен датчик цвета TCS3200

Рис. 20.1. Датчик цвета TCS3200

Принцип работы датчика цвета основан на измерении спектрального отражения или поглощения света объектом. Обычно датчики цвета используются для определения RGB-значений (красный, зеленый и синий) объекта, которые затем преобразуются в другие цветовые модели, такие как CMYK (циан, маджента, желтый и черный) или HSV (оттенок, насыщенность и яркость).

Существует несколько типов датчиков цвета, включая оптические, радиационные и химические. Оптические датчики работают на основе дифракции света и определения цветовой гаммы объекта. Радиационные датчики измеряют уровень излучения объекта в различных длинах волн. Химические датчики используются для измерения концентрации определенных химических соединений, которые изменяют свой цвет при взаимодействии с объектом.

Датчики цвета могут быть использованы для различных задач, таких как определение цвета материала для сопоставления и различения, определение степени зрелости фруктов и овощей, анализ качества пищевых продуктов, сортировка объектов по цвету и многое другое. В робототехнике датчики цвета часто используются для определения цвета объектов и принятия решений на основе этой информации. Например, робот может использовать датчик цвета для различения разных цветных маркеров на листе бумаги и создания рисунков.

Один из примеров использования датчика цвета в проекте на Arduino - это создание системы автоматического управления освещением в комнате. Датчик цвета может измерять цветовую температуру освещения в комнате, и если она не соответствует заданному диапазону, то Arduino может управлять освещением, чтобы увеличить или уменьшить интенсивность света, чтобы достичь необходимой температуры цвета.

Для реализации такого проекта необходимо подключить датчик цвета к плате Arduino и написать соответствующий код программы. Пример кода для работы с датчиком цвета TCS3200 в Arduino IDE может выглядеть следующим образом:

```
#include <Wire.h>
#include <Adafruit_TCS34725.h>
// Инициализация датчика цвета TCS3200
Adafruit_TCS34725 tcs =
Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_4X); void setup() {
Serial.begin(9600); // Инициализация датчика цвета if
(!tcs.begin()) {
Serial.println("Ошибка инициализации TCS34725"); while
(1);
}
// Настройка интервала измерения
tcs.setInterrupt(false);
} void loop() { uint16_t r, g, b, c; float colorTemp;
```

```
// Чтение данных с датчика цвета tcs.getRawData(&r,
&g, &b, &c);

// Вычисление температуры цвета по данным с датчика
colorTemp = 1000000.0 / ((float)r * 0.299 + (float)g *
0.587 + (float)b * 0.114);
      // Отправка данных в последовательный порт
Serial.print("R: "); Serial.print(r); Serial.print("
G: "); Serial.print(g);
Serial.print(" B: "); Serial.print(b);
Serial.print(" C: "); Serial.print(c);
Serial.print(" Color Temperature: ");
Serial.print(colorTemp, 0); // Управление освещением в
зависимости от температуры цвета
Serial.println(" K");
if (colorTemp < 2700) { // Увеличить яркость света
analogWrite(LED_PIN, 255); // пример,
увеличивает яркость света на максимум
} else if (colorTemp > 6500) { // Уменьшить яркость
света
        analogWrite(LED_PIN, 0); // пример, выключает
свет
} else {
// Оставить яркость света без изменений
} delay(500);

}
```

В этом примере программа считывает данные с датчика цвета TCS3200, вычисляет температуру цвета и управляет яркостью света в зависимости от температуры. Этот проект можно доработать, добавив дополнительные функции, такие как:

- добавление дополнительных условий для управления освещением, например, управление цветом света или переход в ночной режим;

- использование дополнительных датчиков, таких как датчик движения, чтобы автоматически включать и выключать свет в комнате;

- создание интерфейса управления освещением через веб-страницу или мобильное приложение;

- использование беспроводных модулей для управления освещением в других комнатах или на другом конце дома.

На данном занятии приведено лишь несколько примеров того, как можно использовать датчик цвета в проекте на Arduino. Реализация проекта с использованием датчика цвета зависит от того, какие функции и возможности заданы для конкретного проекта.

Занятие №21. Знакомство с датчиком ИК

Датчик ИК (инфракрасный датчик) - это устройство, которое использует инфракрасные лучи для обнаружения объектов вокруг себя. Они широко используются в различных приложениях, таких как безопасность дома, автоматическое освещение, удаленный контроль и т. д. Чтобы понять, как работает датчик ИК, нужно понимать, что инфракрасный свет находится за пределами видимого диапазона света и имеет длину волны от 700 нм до 1 мм.

Датчик ИК состоит из инфракрасного излучателя и фотоприемника, которые работают в паре. Инфракрасный излучатель генерирует инфракрасный свет, который направлен на объект. Если объект находится в поле зрения датчика, инфракрасные лучи отражаются от объекта и попадают на фотоприемник. Фотоприемник определяет наличие объекта, опираясь на количество отраженного инфракрасного света. На рисунке 21.1 изображен Датчик ИК.

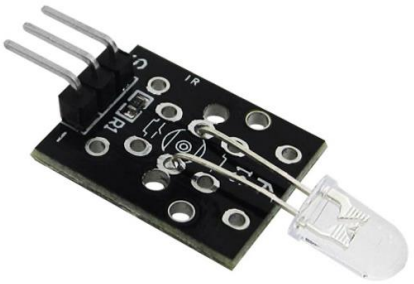

Рис.21.1. Датчик ИК

Датчики ИК могут быть использованы как для обнаружения движения объектов, так и для измерения расстояния до объектов. Некоторые датчики ИК

могут также быть управляемыми, что позволяет им быть более гибкими в использовании.

Для начала работы с датчиком ИК вам нужно ознакомиться с документацией и спецификациями конкретного датчика. Затем необходимо подключить датчик к Arduino, обычно используя “пины” GPIO. После подключения можно использовать библиотеки или написать код на языке программирования для управления датчиком ИК и чтения сигналов, полученных с фотоприемника.

Для работы датчиком ИК можно использовать Arduino или Raspberry Pi (миниатюрный одноплатный компьютер). Существует множество ресурсов и примеров кода, доступных в Интернете, которые помогут начать работу с датчиком ИК и использовать его в различных приложениях.

```
#include <IRremote.h> // Подключение библиотеки для работы с инфракрасным датчиком
const int IR_PIN = 2; // Определение пинов для подключения датчика
IRrecv irrecv(IR_PIN); // Инициализация объекта для работы с инфракрасным датчиком
decode_results results; // Создание объекта для хранения принятых сигналов


void setup()
{
  Serial.begin(9600);    // Инициализация сериального порта
  irrecv.enableIRIn(); // Включение датчика
}
void loop()
{
  if (irrecv.decode(&results))   // Проверка наличия принятого сигнала
  {
    Serial.println(results.value, HEX);  // Вывод кода принятого сигнала в сериальный порт
    irrecv.resume();  // Сброс буфера датчика
```

```
    }
}
```

Приведенный выше код использует библиотеку IRremote для работы с инфракрасным датчиком. Он инициализирует объект IRrecv для получения данных от датчика и устанавливает «пин» для подключения датчика (в данном случае – «пин» 2).

В функции setup() код инициализирует сериальный порт для вывода результатов и включает датчик с помощью метода enableIRIn().

В основном цикле loop() код проверяет наличие принятого сигнала с помощью метода decode(). Если сигнал принят, код сигнала выводится в сериальный порт с помощью Serial.println(). Затем буфер датчика сбрасывается с помощью метода resume().

Следует помнить о необходимости подключения датчика ИК к правильному «пину» на плате Arduino и об установке соответствующей библиотеки IRremote перед компиляцией и загрузкой кода на Arduino.

Примеры применения датчика ИК.

● Системы безопасности: датчики ИК могут использоваться для обнаружения движения внутри помещения. Это может быть полезно для систем безопасности дома или офиса, чтобы сигнализировать о возможной угрозе.

● Автоматическое освещение: датчики ИК могут использоваться для автоматического включения света, когда кто-то входит в комнату. Это может помочь сэкономить энергию и деньги, уменьшив время, когда свет горит в пустой комнате.

● Удаленный контроль: датчики ИК могут применяться для управления устройствами на расстоянии. Например, вы можете использовать датчик ИК для управления телевизором, ресивером, кондиционером и другими устройствами.

● Измерение расстояния: некоторые датчики ИК могут использоваться для измерения расстояния до объектов. Это может быть полезно в робототехнике или других приложениях, где необходимо измерить расстояние до объектов.

● Игрушки и игровые устройства: датчики ИК могут применяться в игрушках и игровых устройствах. Например, датчик ИК может использоваться в пистолете, который стреляет лучами света, и игроки должны попасть в цели, которые также оснащены датчиками ИК.

Кроме того, стоит отметить, что в зависимости от модели датчика ИК могут быть определенные особенности и ограничения. Например, некоторые датчики

ИК могут иметь ограниченный дальности обнаружения, чувствительность к окружающей среде или некоторые другие факторы, которые могут повлиять на их работу. Поэтому важно изучить документацию перед началом работы с датчиком ИК.

Для работы с датчиком ИК необходимо знать некоторые основные характеристики и параметры.

- Дальность обнаружения - это максимальное расстояние, на котором датчик ИК может обнаружить объект. Обычно дальность обнаружения указывается в документации к датчику.

- Частота - это частота сигнала ИК-излучения, которым работает датчик ИК. Обычно датчики ИК работают на частотах 38 кГц или 40 кГц.

- Угол обзора - это угол, на который датчик ИК может "видеть" объекты. Обычно угол обзора указывается в документации к датчику.

- Чувствительность - это способность датчика ИК обнаруживать объекты в зависимости от их размера и материала. Чувствительность датчика ИК может зависеть от окружающей среды, например, от солнечного света или электрических полей.

- Для работы с датчиком ИК также необходимо знать, какой контроллер или микроконтроллер поддерживает работу с датчиком ИК. Некоторые контроллеры имеют встроенную поддержку датчиков ИК, в то время как другие могут требовать использования библиотек или дополнительных компонентов.

Важно также помнить, что ИК-излучение может быть плохо видимо глазом человека, поэтому для проверки работы датчика ИК может потребоваться использование камеры на мобильном телефоне или других устройствах.

Для работы с датчиком ИК необходимы компоненты.

- Датчик ИК - это основной компонент, который используется для обнаружения ИК-излучения и преобразования его в электрический сигнал.

- Резисторы - резисторы используются для ограничения тока, который поступает на датчик ИК. Это может помочь защитить датчик от повреждений.

- Конденсаторы - конденсаторы используются для сглаживания сигнала и устранения помех.

● Транзисторы - транзисторы используются для управления током, который поступает на датчик ИК. Это может помочь снизить потребление энергии и увеличить длительность работы устройства.

● Микроконтроллеры - микроконтроллеры используются для управления датчиком ИК и обработки сигналов, которые он получает. Микроконтроллеры могут также использоваться для управления другими компонентами устройства.

Кроме того, для работы с датчиком ИК может потребоваться использование специальных инструментов, таких как осциллографы или логические анализаторы, которые могут помочь в тестировании и отладке устройства.

Для работы с датчиком ИК важно иметь хорошее понимание принципов его работы и соответствующих технологий, чтобы избежать ошибок и повреждений компонентов устройства.

Датчики ИК используются во многих приложениях, включая удаленное управление, системы безопасности, термометры и т.д. Некоторые области применений датчиков ИК.

● Удаленное управление - датчики ИК используются в устройствах удаленного управления, таких как телевизоры, DVD-плееры, кондиционеры и т.д. Они позволяют управлять устройством из дальнего расстояния с помощью ИК-сигналов.

● Безопасность - датчики ИК могут использоваться в системах безопасности для обнаружения движения и/или температурных изменений. Они могут использоваться, например, в системах тревоги или в дверных замках.

● Медицина - датчики ИК могут использоваться в медицинских приложениях, таких как термометры, которые измеряют температуру тела пациента. Также они могут использоваться для измерения температуры пищи, например, в промышленности или в ресторанах.

● Промышленность - датчики ИК могут использоваться для контроля температуры и измерения уровня жидкостей в различных промышленных приложениях.

● Робототехника - датчики ИК могут использоваться для управления движением роботов и дронов.

На данном занятии были перечислены основы работы датчика ИК с использованием Arduino. Определено, как подключить этот датчик к плате Arduino и считывать его состояние. Датчики ИК являются важным компонентом

многих устройств, и их использование позволяет повысить эффективность и точность работы этих устройств.

Занятие №22. Знакомство с Bluetooth модулем

Bluetooth-модуль на базе Arduino - это важный элемент для создания беспроводной связи между Arduino и другими устройствами, использующими Bluetooth-технологию. На данном будет рассмотрено, как используется Bluetooth-модуль с Arduino, а также представим несколько примеров кода для управления устройствами по Bluetooth. У Bluetooth-модуля под Arduino, есть несколько преимуществ, перед стандартными дополнениями под другие микроконтроллеры.

- Инженеру нет необходимости изучать технологию протокола Bluetooth, чтобы написать софт или начать использовать уже готовые библиотеки.

- Простота использования. Нет необходимости паять отдельную плату под распределение мощностей, просто подсоедините устройство к уже готовому микроконтроллеру через “пины”.

- Обширный выбор библиотек. Так как Arduino имеет низкий порог вхождения, под все его модули можно найти большое количество библиотек, разного назначения. Но стоит отметить, что во многих ситуациях, просто модифицировать чужой софт – не лучшее решение, и значительно проще написать самостоятельно.

На рынке существует множество различных Bluetooth-модулей, которые могут быть использованы в сочетании с Arduino.

- HC-05: Это один из самых популярных Bluetooth-модулей на базе Arduino, который используется для создания беспроводных связей между Arduino и другими устройствами с Bluetooth-поддержкой. Модуль HC-05 использует стандартный протокол Bluetooth SPP (Serial Port Profile) и может работать на расстоянии до 10 метров.

- HC-06: Этот Bluetooth-модуль также широко используется в проектах Arduino. Он поддерживает тот же протокол SPP, что и HC-05, но имеет более ограниченный диапазон действия до 6 метров.

- HM-10: Это Bluetooth-модуль на базе Arduino, который поддерживает более новый протокол Bluetooth Low Energy (BLE). HM-10 может работать на расстоянии до 30 метров и обеспечивает более низкое энергопотребление, что делает его идеальным для проектов, где важно сберегать заряд батареи.

● Bluefruit EZ-Link: Это Bluetooth-модуль, который был разработан специально для работы с Arduino. Он может работать как в режиме SPP, так и в режиме HID (Human Interface Device) и позволяет создавать беспроводные связи между Arduino и компьютером или смартфоном.

● RN-42: Это еще один Bluetooth-модуль на базе Arduino, который поддерживает протокол SPP. RN-42 обеспечивает стабильную работу на расстоянии до 10 метров и обладает высокой скоростью передачи данных.

Конечный выбор Bluetooth-модуля для проекта будет зависеть от требуемых характеристик, таких как расстояние действия, скорость передачи данных и энергопотребление.

Bluetooth-модуль для Arduino имеет набор выводов, включая RX и TX, а также питание и землю. Модуль подключается к Arduino через эти выводы и может использоваться для передачи и приема данных через Bluetooth. Чтобы использовать Bluetooth-модуль с Arduino, сначала нужно установить соответствующую библиотеку Bluetooth в Arduino IDE.

Пример кода для управления светодиодом через Bluetooth-модуль.

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(10, 11); // RX | TX
int LED = 13; void setup() {
pinMode(LED, OUTPUT); Serial.begin(9600);
BTserial.begin(38400);
} void loop() {
if (BTserial.available()) { char c = BTserial.read();
if (c == '1') {
digitalWrite(LED, HIGH);
} else if (c == '0') {
digitalWrite(LED, LOW);
}
}
}
```

В этом примере кода используется Bluetooth-модуль для управления светодиодом на Arduino. В коде устанавливается вывод 13 как выходной и

подключается его к светодиоду. Затем проверяется, есть ли доступные данные через Bluetooth-модуль. Если данные доступны, то они считываются и происходит проверка - является ли символ "1". Если это так, светодиод включается, а если символ "0", то выключается. Этот пример показывает, как можно использовать Bluetooth-модуль для управления устройствами на Arduino.

Еще один пример кода, использующий Bluetooth-модуль на базе Arduino, показывает, как считывать данные с датчика и отправлять их через Bluetooth-модуль.

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(10, 11); // RX | TX int
sensorPin = A0; void setup() {
Serial.begin(9600);
BTserial.begin(38400);
} void loop() {
int sensorValue = analogRead(sensorPin);
BTserial.println(sensorValue); delay(100);
}
```

В этом примере кода используется Bluetooth-модуль для передачи данных с датчика на Arduino. Данные считываются с аналогового “пина” A0 и отправляются через Bluetooth-модуль в виде строки. Между каждой передачей данных включается задержка в 100 миллисекунд, чтобы не перегружать Bluetooth-модуль.

В данном занятии мы познакомились с основами работы Bluetooth-модуля с использованием Arduino. Мы узнали, как подключить модуль к плате Arduino и производить общение по протоколу Bluetooth. Стоит отметить, что Bluetooth-модуль на базе Arduino - это полезный элемент для создания беспроводной связи между Arduino и другими устройствами, использующими Bluetooth-технологию. На этом занятие показано, как использовать Bluetooth-модуль с Arduino и представлено несколько примеров кода для управления устройствами и передачи данных через Bluetooth-модуль. Опираясь на эти примеры, можно создать свои собственные проекты с использованием Bluetooth-модуля и Arduino.

Занятие №23. Знакомство с Wi-Fi модулем

Wi-Fi модуль для Arduino – это устройство, которое позволяет Arduino подключаться к беспроводным сетям Wi-Fi и обмениваться данными через

интернет. Этот модуль можно использовать для создания IoT-приложений, удаленного управления устройствами, получения данных с Интернета и многого другого. На рисунке 23.1 представлен модуль Wi-Fi ESP8266 (ESP-01).

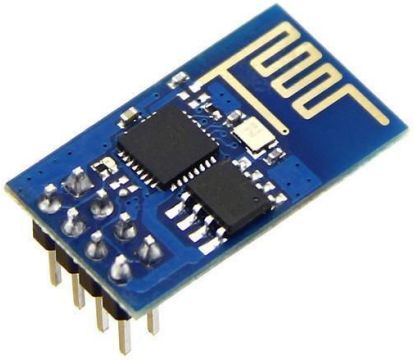

Рис. 23.1. Модуль Wi-Fi ESP8266 (ESP-01)

Одним из популярных Wi-Fi модулей для Arduino является ESP8266, который можно легко подключить к плате Arduino. Он имеет встроенный Wi-Fi-адаптер и может работать как точка доступа или в режиме станции. Библиотеки для этого модуля доступны в Arduino IDE, что упрощает процесс программирования.

В этом примере используется библиотека SoftwareSerial, которая позволяет использовать “пины”, отличные от стандартных “пинов” RX и TX для общения с модулем ESP8266 (ESP-01).

В функции setup() мы инициализируем последовательную связь с монитором и модулем ESP8266. Затем отправляем команду AT для проверки связи с модулем.

В функции loop() мы проверяем наличие данных от ESP8266 и отправляем их в монитор последовательной связи. Если есть данные от монитора последовательной связи, мы отправляем их в ESP8266.

Функция sendCommand() используется для отправки команды в ESP8266. В этом примере мы просто отправляем команду и ждем 1 секунду перед отправкой следующей команды.

Этот пример кода позволяет установить связь с модулем ESP8266 и передавать данные между модулем и монитором последовательной связи. Вы можете дополнить его дальнейшей логикой и командами для работы с Wi-Fi и другими функциями ESP8266.

```
#include <SoftwareSerial.h>
```

```
SoftwareSerial esp8266(2, 3); // Указываем “пины” для соединения с модулем ESP8266 (RX, TX)
void setup() {
  Serial.begin(9600); // Инициализация последовательной связи с монитором
  esp8266.begin(9600); // Инициализация последовательной связи с ESP8266
  delay(1000); // Ждем 1 секунду, чтобы модуль успел запуститься
  sendCommand("AT"); // Отправляем команду AT для проверки связи с модулем
  delay(1000); // Ждем 1 секунду
}
 void loop() {
  if (esp8266.available()) { // Если есть данные от ESP8266
    Serial.write(esp8266.read()); // Отправляем данные в монитор последовательной связи
  }
  if (Serial.available()) { // Если есть данные от монитора последовательной связи
    esp8266.write(Serial.read()); // Отправляем данные в ESP8266
  }
}
 void sendCommand(String command) {
  esp8266.println(command); // Отправляем команду в ESP8266
  delay(1000); // Ждем 1 секунду
}
```

Один из примеров использования Wi-Fi модуля в проекте на Arduino - это создание “умного” дома или “умной” системы автоматизации. Например, можно

использовать Wi-Fi модуль, чтобы подключить проектированную систему к интернету и контролировать ее удаленно через смартфон или компьютер.

Конкретный пример проекта - это создание “умного” выключателя. Можно использовать Arduino, Wi-Fi модуль и реле, чтобы создать устройство, которое может включать или выключать свет в комнате через Интернет. В результате можно управлять выключателем через приложение на смартфоне или компьютере, а также настроить расписание включения и выключения света в разное время дня.

Кроме того, Wi-Fi модуль может использоваться для получения данных из Интернета, например, для получения погодных прогнозов или новостей, которые могут быть отображены на дисплее или воспроизведены голосом через динамик.

Также, Wi-Fi модуль может использоваться для создания сетевых устройств, таких как датчики и мониторы, которые могут собирать данные о температуре, влажности, уровне шума и других параметрах и отправлять эти данные на удаленный сервер для анализа или хранения.

В данном занятии мы познакомились с основами работы Wi-Fi модуля с использованием Arduino. Мы узнали, как подключить модуль к плате Arduino и производить коммуникацию по протоколу Wi-Fi. В целом, возможности использования Wi-Fi модуля в Arduino проектах довольно обширны и, какие зависят от того задачи и цели ставятся в проекте.

Занятие №24. Знакомство с терморезистором

Терморезистор (термистор) - это электронный элемент, который изменяет свое электрическое сопротивление в зависимости от температуры окружающей среды. Он используется для измерения температуры в различных приложениях, таких как термометры, термостаты, термокомпенсаторы и другие системы контроля температуры.

Терморезисторы могут быть сделаны из различных материалов, но наиболее распространенными являются металлы, такие как платина, никель или медь, а также полупроводники, такие как кремний и германий.

Сопротивление терморезистора имеет зависимость от темепературы $R = R_0(1 + \alpha\Delta T)$, где R_0 - сопротивление при комнатной температуре, α температурный коэффициент, а ΔT - изменение температуры. На рисунке 24.1 представлено условно-графическое обозначение терморезистора.

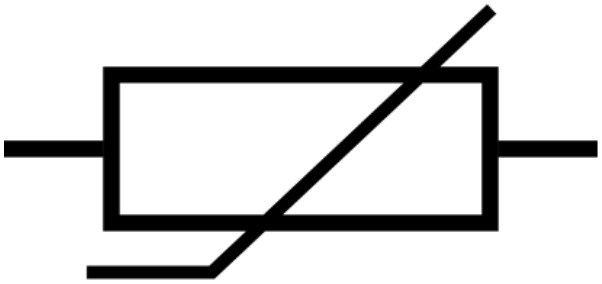

Рис. 24.1. Условно-графическое обозначение терморезистора

Для измерения температуры с помощью терморезистора необходимо измерить изменение его сопротивления и затем использовать калибровочную кривую, чтобы определить соответствующую температуру.

Терморезисторы имеют множество преимуществ перед другими типами термометров, включая высокую точность, широкий диапазон измерений, быстрый отклик и долгий срок службы. Они также могут быть очень компактными и удобными для использования в автоматизированных системах контроля температуры.

Однако, также следует отметить, что терморезисторы имеют свои ограничения, такие как чувствительность к воздействию других факторов, таких как влажность и механические воздействия, а также более высокую стоимость по сравнению с другими типами термометров.

Другое важное свойство терморезисторов - это их температурный диапазон. Некоторые терморезисторы могут работать при очень высоких температурах до 1000 градусов Цельсия, в то время как другие могут быть ограничены только до нескольких десятков градусов. Поэтому важно выбрать правильный терморезистор в зависимости от конкретных потребностей и требований приложения.

Существуют различные типы терморезисторов, такие как RTD (Resistance Temperature Detector) или PT100, которые являются наиболее распространенными, а также NTC (Negative Temperature Coefficient) терморезисторы, которые имеют отрицательный температурный коэффициент. Каждый тип имеет свои преимущества и недостатки, и выбор определенного типа зависит от конкретных потребностей и требований приложения.

Для работы с терморезисторами могут использоваться специальные измерительные приборы, такие как термометры сопротивления, которые способны измерять сопротивление терморезистора и преобразовывать его в температуру. Также могут использоваться различные методы компенсации ошибок, такие как двухпроводная, трехпроводная и четырехпроводная схемы подключения терморезистора.

Терморезисторы могут использоваться в различных областях применения, включая промышленность, медицину, науку, электронику и многое другое. Например, они могут использоваться для измерения температуры в промышленных процессах, таких как плавка металла или кристаллизация стекла, где точность и надежность очень важны. В медицинских приложениях они могут использоваться для измерения температуры тела пациента, а в науке - для измерения температуры окружающей среды или температуры образцов в лаборатории. Терморезисторы могут также использоваться в различных устройствах для управления температурой, например, в холодильниках, кондиционерах, водонагревателях и термостатах. Они могут использоваться в сочетании с другими датчиками для создания автоматических систем управления, которые могут регулировать температуру и поддерживать ее на определенном уровне.

Важно понимать, что для каждого конкретного приложения необходимо выбирать терморезистор с соответствующими характеристиками и параметрами, такими как температурный диапазон, точность, надежность и другие факторы. При правильном выборе и использовании терморезисторы могут быть очень полезными инструментами для контроля и измерения температуры в различных условиях.

Одним из важных аспектов терморезисторов является их точность. Точность терморезистора зависит от его характеристик и параметров, таких как диапазон рабочих температур, стабильность, длина провода и другие факторы. В целом, терморезисторы могут иметь точность до нескольких десятых градуса Цельсия.

Точность терморезистора может быть улучшена путем компенсации ошибок, которые могут возникнуть из-за сопротивления проводов или контактов. Один из способов компенсации ошибок - это использование трехпроводной или четырехпроводной схемы подключения, которые позволяют измерить сопротивление проводов и компенсировать его при измерении температуры.

Также существуют калибровочные процедуры, которые позволяют повысить точность терморезистора. Калибровка может быть проведена путем сравнения измерений терморезистора с известной точностью измерений другого термометра или термометра-стандарта. Наконец, терморезисторы могут быть изготовлены из различных материалов, таких как платина, никель, медь и другие. Каждый материал имеет свои уникальные характеристики и параметры, которые могут быть подходящими для конкретных приложений. Например, платиновые терморезисторы обычно имеют высокую точность и надежность, но также являются более дорогими по сравнению с терморезисторами из других материалов.

Более того, важным аспектом терморезисторов является их чувствительность к изменению температуры. Чувствительность терморезистора определяет, насколько быстро он реагирует на изменение температуры и какой диапазон температур он может измерить.

Коэффициент температурного сопротивления (TCR) является мерой чувствительности терморезистора к изменению температуры. Коэффициент TCR обычно выражается в процентах относительно сопротивления при определенной температуре. Например, для платинового терморезистора с коэффициентом TCR 0,00392 / градус Цельсия, изменение температуры на 1 градус Цельсия приведет к изменению сопротивления на 0,392%.

Также следует учитывать, что разные материалы имеют разную чувствительность к изменению температуры и могут быть более или менее подходящими для конкретных приложений.

Еще одним важным фактором, который следует учитывать при использовании терморезисторов, является их сопротивление. Сопротивление терморезистора может варьироваться в зависимости от температуры и может быть значительно ниже или выше сопротивления других типов датчиков температуры, таких как термопары. Наконец, при использовании терморезисторов следует учитывать их температурную инерцию. Терморезисторы могут иметь определенное время реакции на изменение температуры, которое зависит от их конструкции и характеристик. В некоторых приложениях это может быть важным фактором, особенно если требуется быстрый и точный контроль температуры.

Терморезисторы могут использоваться в различных приложениях, включая контроль температуры в промышленных процессах, системах отопления и кондиционирования, электронике и медицинском оборудовании.

Для использования терморезисторов в различных приложениях необходимо правильно подобрать тип терморезистора, который соответствует требуемому диапазону измерения температуры и другим характеристикам. Например, платиновые терморезисторы могут иметь более высокую точность и стабильность сопротивления в широком диапазоне температур, чем терморезисторы из других материалов, но они также могут быть более дорогими.

Для работы с терморезисторами обычно используются специализированные схемы измерения, такие как мостовые схемы или усилители с дифференциальным входом. Эти схемы позволяют уменьшить влияние изменения напряжения питания и других помех на измерение сопротивления терморезистора.

Подключим термистор к Arduino как это показано на рисунке 24.2:

Подключим один контакт резистора на 10 КОм к контакту 5 В, второй контакт резистора 10 КОм 1% - к одному контакту термистора. Второй контакт термистора подключается к земле. 'Центр' двух резисторов подключим к контакту Analog 0 на Arduino.

Теперь запустим следующий скетч (программа для Arduino):

```
// значение 'другого' резистора
#define SERIESRESISTOR 10000
// к какому “пину” подключается термистор
#define THERMISTORPIN A0
void setup(void) {
Serial.begin(9600);
}
void loop(void) {
float reading;
reading = analogRead(THERMISTORPIN);
Serial.print("Analog reading ");
Serial.println(reading);
// преобразуем полученные значения в сопротивление
reading = (1023 / reading) - 1;
```

```
reading = SERIESRESISTOR / reading;
Serial.print("Thermistor resistance ");
Serial.println(reading);
delay(1000);
}
```

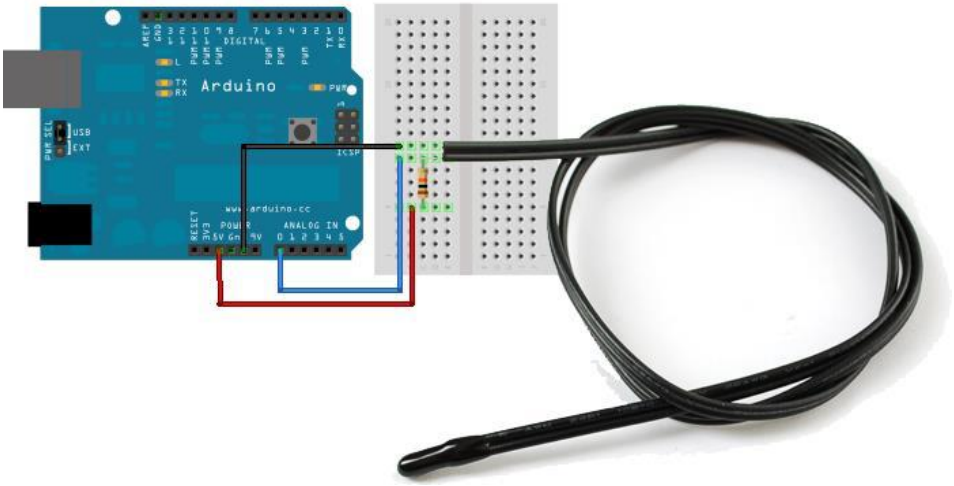

Рис. 24.2. Подключение термиста к Arduino

В результате вы должны получить значения, которые соответствуют измеренным с помощью мультиметра.

В данном занятии определены основы работы терморезистора с использованием Arduino. Рассказано, как подключить терморезистор к плате Arduino и считывать его состояние. В заключение можно отметить, что терморезисторы являются удобными и точными датчиками температуры, которые могут быть использованы во многих приложениях. Правильный выбор типа терморезистора, а также специализированной схемы измерения, помогут достичь максимальной точности и стабильности измерения температуры.

Занятие №25. Знакомство с датчиком освещенности

Датчик освещенности - это электронный компонент, который используется для измерения уровня освещенности в окружающей среде. В проектах на базе Arduino, датчик освещенности может быть использован для автоматического управления освещением, создания регуляторов освещенности и других задач, связанных с измерением уровня освещенности.

Перечислим несколько примеров проектов, которые могут использовать датчик освещенности на базе Arduino.

● Автоматический датчик освещения. Этот проект использует датчик освещенности для автоматического управления включением и выключением света. Если уровень освещенности падает ниже определенного порога, Arduino может управлять реле для включения света.

● Мониторинг света в комнате. Этот проект использует датчик освещенности для мониторинга уровня освещенности в комнате. Arduino может отправлять данные о уровне освещенности в облако, где их можно отслеживать и анализировать на удаленном компьютере.

● Система контроля растительности. Этот проект использует датчик освещенности для контроля уровня освещенности в комнате с растениями. Arduino может использоваться для управления источниками света, такими как лампы, чтобы обеспечить растениям необходимый уровень освещенности.

● Робот-пылесос: Этот проект использует датчик освещенности для определения уровня освещенности в комнате, а затем использует эту информацию для навигации по комнате во время уборки.

В зависимости от характеристик проекта, необходимо выбрать подходящий датчик освещенности на базе Arduino. Некоторые из наиболее популярных датчиков включают в себя фоторезисторы, фотодиоды и фототранзисторы.

Принцип работы датчика освещенности на базе Arduino заключается в том, что датчик определяет уровень освещенности, используя фоторезистор (также известный как LDR) или фототранзистор или фотодиоды. Фоторезистор изменяет свою сопротивляемость в зависимости от уровня света, на который он подвергается, и измеряет эту сопротивляемость.

Существует несколько типов датчиков освещенности, которые могут быть использованы с Arduino.

● Фоторезисторы. Это наиболее распространенный тип датчика освещенности, который работает на основе изменения сопротивления в зависимости от уровня освещенности. Фоторезисторы могут быть подключены к Arduino с помощью аналогового входа.

● Фотодиоды. Эти датчики работают на основе эффекта фотоэлектрической эмиссии, когда свет попадает на поверхность диода и вызывает поток электронов. Фотодиоды могут быть подключены к Arduino через аналоговый вход.

● Фототранзисторы. Эти датчики работают на основе изменения тока при изменении уровня освещенности. Фототранзисторы могут быть подключены к Arduino через аналоговый вход.

● Цифровые датчики. Это датчик освещенности, который имеет высокую точность и может измерять уровень освещенности в широком диапазоне, от очень темного до очень яркого освещения.

Выбор датчика освещенности для Arduino зависит от конкретных потребностей проекта. Если необходима высокая точность измерения, то лучше использовать цифровой датчик (например: TSL2561 или BH1750). Если же требуется более простой датчик, то фоторезисторы, фотодиоды или фототранзисторы могут быть лучшим выбором.

Для работы с датчиком освещенности на базе Arduino необходимо подключить датчик к аналоговому входу платы Arduino, затем считывать значение, которое он измеряет. Код для чтения значения датчика освещенности выглядит следующим образом.

```
const int LDR_PIN = A0; // “Пин” для подключения LDR
int LDR_VALUE; // переменная для хранения значения LDR
void setup() {
Serial.begin(9600); // начало последовательного порта }
void loop() {
LDR_VALUE = analogRead(LDR_PIN); // считывание
значения с LDR_PIN Serial.println(LDR_VALUE); delay(500);
// пауза 500 мс
}
```

Этот код считывает значение датчика освещенности с помощью функции analogRead(), затем отправляет его в последовательный порт для вывода на компьютер. Пауза в 500 мс между считываниями значения позволяет избежать чрезмерного использования процессорного времени.

На данном занятии представлены основы работы датчика освещенности с использованием Arduino. Определено, как подключить датчик к плате Arduino и считывать его состояние. В заключение можно сделать вывод, что датчик освещенности на базе Arduino - это удобный и простой способ измерения уровня освещенности в окружающей среде. Он может быть использован во многих различных проектах, связанных с контролем освещения, а также в проектах, связанных с ботаникой, робототехникой и многими другими областями.

Занятие №26. Знакомство с датчиком хола

Датчик Холла - это электронное устройство, которое использует явление Холла для измерения магнитных полей. Он состоит из магнитного датчика и схемы обработки сигнала. Датчик Холла использует эффект Холла, который заключается в появлении электрического напряжения в проводнике, помещенном в магнитное поле, перпендикулярное направлению тока. На рисунке 26.1 представлен миниатюрный аналоговый датчик Холла SS49E.

Рис. 26.1. Миниатюрный аналоговый датчик Холла SS49E

Датчики Холла могут быть использованы для измерения постоянных магнитных полей, а также для измерения изменяющихся магнитных полей, например, в случае использования их в датчиках положения или скорости. Они могут быть использованы в различных системах, включая электронику автомобилей для измерения скорости вращения колес и распознавания местоположения дверей, а также в медицинском оборудовании для измерения магнитных полей внутри тела человека.

Датчики Холла также используются для контроля положения и перемещения магнитов, таких как магнитные ленты и диски, используемые для записи и хранения данных. Они также используются в электрических двигателях для измерения положения ротора и контроля скорости вращения.

Датчики Холла имеют ряд преимуществ перед другими типами датчиков, таких как датчики сопротивления, датчики емкости и датчики индуктивности. Они более точные, не чувствительны к температурным изменениям и механическим воздействиям, и могут работать в широком диапазоне температур и влажности. Они также обладают высокой чувствительностью и быстрым откликом.

“Шаги” использования датчика Холла в проекте на Arduino:

1. Подключите датчик Холла к Arduino. Для этого подключите “пин” VCC к 5V, “пин” GND к “земле” и “пин” OUT к любому цифровому “пину” на Arduino, например, 2.

2. Напишите программу в Arduino IDE, чтобы считывать данные с датчика Холла.

3. Загрузите программу в Arduino и откройте монитор порта в Arduino IDE. Вы должны увидеть значения 0 или 1, которые соответствуют магнитному полю, обнаруженному датчиком Холла.

4. Чтобы узнать, какие значения соответствуют магнитному полю, вы можете использовать магнит или другой магнитный объект и приближать его к датчику Холла, пока не увидите изменение значений в мониторе порта.

5. Используйте полученные данные для управления проектом. Например, можно использовать датчик Холла, чтобы контролировать скорость двигателя в зависимости от магнитного поля.

В данном коде используется датчик Холла, подключенный к “пину” hallPin.

В функции setup() происходит инициализация последовательной связи с помощью Serial.begin() для вывода данных.

В функции loop() происходит основной цикл программы. Считывается значение с датчика Холла с помощью digitalRead(). Затем полученное значение выводится в монитор последовательной связи с помощью Serial.println(). Между измерениями осуществляется задержка в 100 миллисекунд с помощью delay().

```
const int hallPin = 2; void setup() {

Serial.begin(9600);
}
void loop() { int reading = digitalRead(hallPin);
Serial.println(reading); delay(100);

}
```

На данном занятии определена работа датчика Холла с использованием Arduino. Датчик Холла представляет собой устройство, которое позволяет обнаруживать магнитные поля и преобразовывать их в электрический сигнал. Мы узнали, как подключить датчик Холла к плате Arduino, указав соответствующий “пин” для считывания сигнала. С помощью функции

digitalRead() мы можем считывать состояние “пина”, где подключен датчик, и определить наличие или отсутствие магнитного поля. С использованием последовательной связи и функции Serial.println() мы выводили считанные значения в монитор последовательной связи. Это позволяло нам наблюдать изменения состояния датчика и проверять его работу.

Занятие №27. Знакомство с LCD дисплеем

LCD (Liquid Crystal Display) дисплей - это тип экрана, который использует жидкие кристаллы для создания изображения. Они наиболее распространены в портативных устройствах, таких как мобильные телефоны, планшеты, цифровые фотоаппараты и ноутбуки.

Для подключения LCD дисплея к устройству может потребоваться контроллер LCD. Контроллеры LCD управляют поведением жидкокристаллической матрицы и определяют, какие пиксели должны быть включены или выключены.

Кроме того, может потребоваться библиотека или драйвер, которые позволят вам легко взаимодействовать с LCD дисплеем через микроконтроллер или компьютер.

Чтобы начать работу с LCD дисплеем, необходимо изучить документацию по конкретному дисплею и контроллеру, чтобы понять, как подключить его к устройству и как использовать его функции. Для начала можно попробовать простые примеры, которые поставляются с библиотекой или драйвером для вашего контроллера LCD, чтобы понять, как отображать текст и графику на экране. На рисунке 27.1 представлен пример модуля с LCD дисплеем.

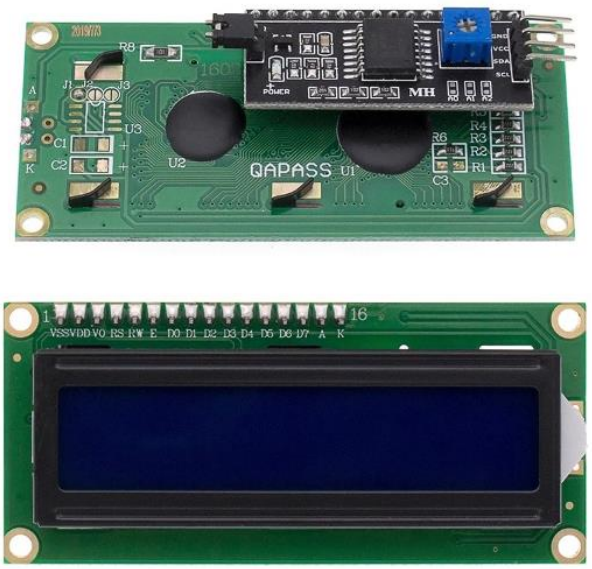

Рис. 27.1. Пример модуля с LCD дисплеем

Кроме отображения текста и графики, LCD дисплеи могут иметь дополнительные функции, такие как подсветка, сенсорные экраны или возможность отображения изображений высокого разрешения. Чтобы использовать подсветку LCD дисплея, может потребоваться подключить дополнительный контроллер для управления яркостью светодиодов. Сенсорные экраны могут использоваться для взаимодействия с пользователем, что позволяет им вводить данные или управлять устройством, касаясь экрана.

Если есть необходимость отображать изображения высокого разрешения, то может потребоваться использовать более сложные техники, такие как интерполяция пикселей или сглаживание изображений.

Одной из важных вещей при работе с LCD дисплеями является оптимизация скорости обновления экрана. Быстрое обновление экрана может повысить плавность работы вашего устройства и улучшить пользовательский опыт.

Наконец, необходимо помнить, что каждый LCD дисплей и контроллер управления им может иметь свои особенности и требования, поэтому важно внимательно изучить документацию и примеры кода для конкретного устройства перед началом работы с ним.

Часто LCD дисплеи имеют ограниченную память для хранения графических объектов, поэтому необходимо управлять ее использованием, чтобы избежать переполнения памяти. Существуют специальные библиотеки, которые помогают

управлять памятью и упрощают работу с графическими объектами, такими как линии, прямоугольники, круги и т.д.

Существует несколько интерфейсов, таких как SPI, I2C и параллельный интерфейс, каждый из которых имеет свои преимущества и недостатки. SPI и I2C - это последовательные интерфейсы, которые позволяют передавать данные с небольшой скоростью, но с меньшим количеством контактов. Параллельный интерфейс позволяет передавать данные на более высоких скоростях, но требует большего количества контактов. При выборе интерфейса важно учитывать не только требования к скорости и количеству контактов, но также и удобство использования, и доступность драйверов и библиотек для работы с конкретным интерфейсом.

Также при работе с LCD дисплеями важно учитывать их размер и разрешение. Чем больше разрешение, тем более детально можно отображать информацию, но также и требования к памяти и скорости передачи данных будут выше.

Калибровка сенсорного экрана. Если LCD дисплей оснащен сенсорным экраном, то необходимо произвести калибровку, чтобы убедиться в правильной работе всех точек сенсорного экрана.

Калибровка сенсорного экрана может быть произведена с помощью специальных инструментов и библиотек, которые позволяют определить координаты всех точек сенсорного экрана. Это позволяет обеспечить точное и надежное управление сенсорным экраном при работе с устройством.

Также важно учитывать, что работа с LCD дисплеями может потребовать использования дополнительных компонентов и устройств, таких как контроллеры сенсорного экрана, драйверы подсветки, усилители сигнала и т.д.

Выбор дополнительных компонентов и устройств зависит от конкретного проекта и требований к работе устройства. При выборе дополнительных компонентов необходимо учитывать совместимость с LCD дисплеем, а также требования к их работе и настройке.

Управление и обновление содержимого дисплея. Для этого используются специальные библиотеки и программное обеспечение, которые позволяют управлять всеми аспектами работы с LCD дисплеем, включая отображение графики, текста, анимации и других элементов интерфейса.

В некоторых случаях может потребоваться создание собственных элементов интерфейса, таких как кнопки, ползунки, текстовые поля и т.д. Для этого

используются различные графические библиотеки, которые позволяют создавать и управлять различными элементами интерфейса на LCD дисплее.

При работе с LCD дисплеями необходимо учитывать потребляемую мощность, чтобы обеспечить достаточное время автономной работы устройства. Для управления энергопотреблением используются различные методы, такие как динамическое изменение яркости подсветки дисплея, использование спящего режима при отсутствии ввода пользователя и другие методы.

Давайте подключим LCD-дисплей к Arduino:

Дисплей подключается по шине I2C, выведенной на “пины”:

Arduino: SDA – A4, SCL – A5

Wemos: SDA – D2, SCL – D1

Питание: 5V в обоих случаях

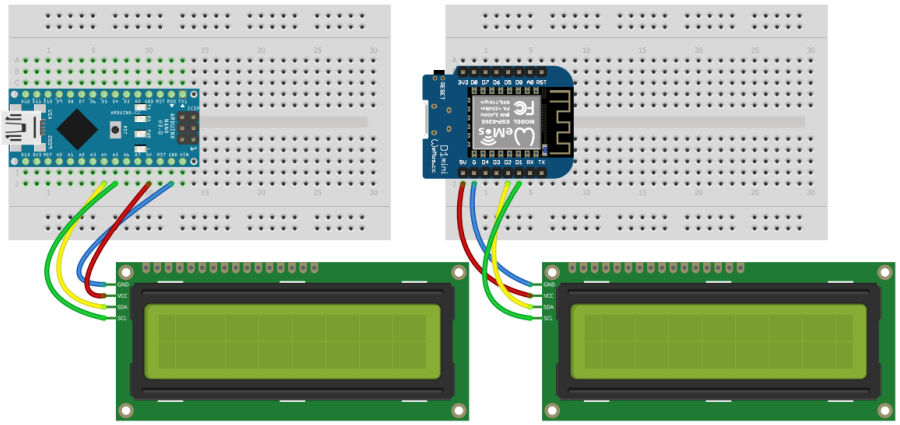

Рис. 27.2. Подключение LCD-дисплея к Arduino

Для этого дисплея существует несколько библиотек, рекомендуем LiquidCrystal_I2C от Frank de Brabander. Библиотека идёт в архиве к набору, а также её можно скачать через менеджера библиотек по названию LiquidCrystal_I2C и имени автора.

```
print(data);            // вывести (любой тип данных)
setCursor(x, y);        // курсор на (столбец, строка)
clear();                // очистить дисплей
home();                 // аналогично setCursor(0, 0)
noDisplay();            // отключить отображение
display();              // включить отображение
```

```
blink();                // мигать курсором на его текущей позиции
noBlink();              // не мигать
cursor();               // отобразить курсор
noCursor();             // скрыть курсор
scrollDisplayLeft();    // подвинуть экран влево на 1 столбец
scrollDisplayRight();   // подвинуть экран вправо на 1 столбец
backlight();            // включить подсветку
noBacklight();          // выключить подсветку
createChar(uint8_t, uint8_t[]);                    // создать символ
createChar(uint8_t location, const char *charmap);  // создать символ
```

При первой работе с дисплеем нужно настроить контраст и определиться с адресом. Это можно сделать следующим образом:

- “прошить” пример “Демо”;
- если дисплей показывает чёрные прямоугольники или пустой экран – нужно крутить контраст;
- если кроме чёрных прямоугольников и пустого экрана ничего не видно – необходимо сменить адрес в программе: 0x26, 0x27 и 0x3F;
- снова крутим контраст, должно заработать;
- если не работает – проверить подключение и повторить сначала.

Примечание: в наборе должны идти дисплеи с адресом 0x27, но адрес может зависеть от партии.

```
#include <LiquidCrystal_I2C.h>      // подключаем библиотеку
// адрес дисплея 0x3f или 0x27
// размер дисплея 16x2 (поддерживаются и другие, например 20x4)
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);  // адрес, столбцов, строк
void setup() {
 lcd.init();            // инициализация
 lcd.backlight();       // включить подсветку
 lcd.setCursor(1, 0);   // столбец 1 строка 0
 lcd.print("Hello, world!");
 lcd.setCursor(4, 1);   // столбец 4 строка 1
 lcd.print("GyverKIT");
}
void loop() {
}
```

С помощью этого кода можно осуществить базовый инициализацию и вывод текста на дисплей.

На данном занятии рассказано об основах работы LCD дисплея с использованием Arduino. Определено, как подключить экран к плате Arduino и выводить на него информацию. В целом, работа с LCD дисплеями является важной и необходимой частью разработки устройств, которые требуют удобного и интуитивно понятного пользовательского интерфейса. Она требует изучения множества аспектов, начиная от выбора подходящего дисплея и интерфейса, и заканчивая созданием элементов интерфейса и управлением энергопотреблением.

Занятие №28. Знакомство с семисегментным индикатором

Семисегментный индикатор — устройство отображения цифровой информации. Это - наиболее простая реализация индикатора, который может отображать арабские цифры. Для отображения букв используются более сложные многосегментные и матричные индикаторы.

Одноразрядный семисегментный индикатор подключить к Arduino можно через макетную плату, самое главное знать “распиновку” (цоколевку индикатора), чтобы управлять сегментами от Arduino Uno. Рассмотрим, как подключить семисегментный индикатор к Arduino и сделать простую программу

с таймером. Будем управлять индикатором напрямую от микроконтроллера, используя тактовую кнопку.

Модуль представляет из себя небольшой led индикатор в котором находится семь светодиодов (благодаря этому индикатор и получил свое название) и восьмой светодиод в виде точки. Включая светодиоды в разной последовательности от Arduino Uno, можно выводить различные цифры.

Обратите внимание, что панель не имеет резисторов, поэтому при подключении светодиодов используйте внешние резисторы. Если цоколевка семисегментного индикатора с общим анодом, вызывает трудности для понимания», то можно опытным путем установить “распиновку”, подключая питание к разным выводам. При неправильном включении ничего страшного не произойдет, а вот без резистора светодиоды могут сгореть.

Светодиоды всех элементов соединяются одноименными выводами между собой или анодами, или катодами. Поэтому разделяют семисегментные индикаторы с общим анодом (подключаем питание) и общим катодом (подключаем на “землю”). Дальше в примерах будем рассматривать семисегментные индикаторы общим катодом.

Вне зависимости от количества разрядов и размеров цифр каждый сегмент имеет название в виде одной из первых букв английского алфавита: a, b, c, d, e, f, g. Точка обозначается dp.

Чтобы отобразить на индикаторе цифру или букву следует засветить несколько сегментов. Например, для отображения единицы 1 задействуются сегменты b и c. При отображении восьмерки 8 задействуются все символы от a до g. Пятерка получается из таких символов: a, c, d, f, g.

Чтобы подключить семисегментный индикатор к плате понадобиться:

- Arduino Uno / Arduino Nano / Arduino Mega;
- одноразрядный семисегментный индикатор 5161as / hdsp 7503;
- тактовая кнопка;
- резисторы 220 Ом;
- макетная плата (breadboard);
- провода “папа-папа”.

В первом примере поочередно включаются (мигают) светодиоды для индикации на панели различных чисел.

Все светодиоды подключаются к выводам микроконтроллера через отдельные резисторы сопротивлением 220…330 Ом.

Не стоит экономить на резисторах и подключать все элементы через один общий резистор. Поскольку в таком случае с изменением числа задействованных сегментов будет изменяться величина тока, протекающего через них. Поэтому цифра 1 будет светиться ярче, чем 8.

Программный код получается большим, поэтому ограничимся тремя числами. Вывод других чисел на семисегментный индикатор Arduino не составит труда.

```
#define A 8
#define B 7
#define C 6
#define D 5
#define E 4
#define F 3
#define G 2
void setup() {
 pinMode(A, OUTPUT);  // Устанавливаем пин A как выход
 pinMode(B, OUTPUT);  // Устанавливаем пин B как выход
     pinMode(C, OUTPUT);  // Устанавливаем пин C как
выход
 pinMode(D, OUTPUT);  // Устанавливаем пин D как выход
 pinMode(E, OUTPUT);  // Устанавливаем пин E как выход
 pinMode(F, OUTPUT);  // Устанавливаем пин F как выход
 pinMode(G, OUTPUT);  // Устанавливаем пин G как выход
}
void loop() {
 digitalWrite(A, LOW); //цифра один
 digitalWrite(B, HIGH);  // Устанавливаем пин B в
состояние HIGH (единица)
 digitalWrite(C, HIGH);
 digitalWrite(D, LOW);  // Устанавливаем пин D в
состояние LOW (ноль)
 digitalWrite(E, LOW);
 digitalWrite(F, LOW);
 digitalWrite(G, LOW);
 delay(1000);  // Задержка в 1 секунду
```

```
 digitalWrite(A, HIGH); //цифра два
 digitalWrite(B, HIGH);
 digitalWrite(C, LOW);
 digitalWrite(D, HIGH);
 digitalWrite(E, HIGH);
 digitalWrite(F, LOW);
 digitalWrite(G, HIGH);
 delay(1000);  // Задержка в 1 секунду
 digitalWrite(A, HIGH); //цифра три
 digitalWrite(B, HIGH);
 digitalWrite(C, HIGH);
 digitalWrite(D, HIGH);
 digitalWrite(E, LOW);
 digitalWrite(F, LOW);
 digitalWrite(G, HIGH);
 delay(1000);  // Задержка в 1 секунду
}
```

В следующем примере переключение чисел на индикаторе будет происходить только при нажатии тактовой кнопки. Дойдя до числа 3, таймер вновь обнулится и будет ожидать повторного нажатия на кнопку.

```
   #define A 8
#define B 7
#define C 6
#define D 5
#define E 4
#define F 3
#define G 2
#define BUTTON 12
byte v = 0;
void setup() {
 pinMode(A, OUTPUT);  // Устанавливаем пин A как выход
 pinMode(B, OUTPUT);  // Устанавливаем пин B как выход
 pinMode(C, OUTPUT);  // Устанавливаем пин C как выход
 pinMode(D, OUTPUT);  // Устанавливаем пин D как выход
 pinMode(E, OUTPUT);  // Устанавливаем пин E как выход
 pinMode(F, OUTPUT);  // Устанавливаем пин F как выход
 pinMode(G, OUTPUT);  // Устанавливаем пин G как выход
```

```
 pinMode(BUTTON, INPUT);  // Устанавливаем пин BUTTON как
вход
 }
   void                    loop()                    {
 digitalWrite(A,      HIGH);            //цифра      нуль
 digitalWrite(B,  HIGH);      //  Устанавливаем  пин  B  в
состояние                 HIGH                 (единица)
 digitalWrite(C,                                  HIGH);
 digitalWrite(D,     HIGH);     digitalWrite(E,     HIGH);
 digitalWrite(F,                                  HIGH);
 digitalWrite(G, LOW);  // Устанавливаем пин G в состояние
LOW                                                (ноль)
 if (digitalRead(BUTTON) == HIGH) { // Если кнопка нажата
     delay(500);       //   Задержка   в   0.5   секунду
     v = 1; // Устанавливаем переменную v в значение 1
     }
     while          (v          ==          1)          {
         digitalWrite(A,      LOW);      //цифра      один
         digitalWrite(B,                            HIGH);
         digitalWrite(C,                            HIGH);
         digitalWrite(D,                             LOW);
         digitalWrite(E,                             LOW);
         digitalWrite(F, LOW); digitalWrite(G, LOW);
         if (digitalRead(BUTTON) == HIGH) {
             delay(500);
             v  =  2;   //  Устанавливаем  переменную  v  в
            значение 2
           }
   }
   while          (v          ==          2)          {
         digitalWrite(A,      HIGH);      //цифра      два
         digitalWrite(B,                            HIGH);
         digitalWrite(C,                             LOW);
         digitalWrite(D,  HIGH);  digitalWrite(E,  HIGH);
         digitalWrite(F,                             LOW);
```

```
        digitalWrite(G,                          HIGH);
        if (digitalRead(BUTTON) == HIGH) { // Если кнопка
нажата
            delay(500);    //  Задержка  в  0.5  секунду
            v  =  3;   //  Устанавливаем  переменную  v  в
значение                                                 3
            }
        }
        while         (v         ==         3)         {
            digitalWrite(A,    HIGH);    //цифра    три
            digitalWrite(B,                       HIGH);
            digitalWrite(C,                       HIGH);
            digitalWrite(D,                       HIGH);
            digitalWrite(E,                        LOW);
            digitalWrite(F,                        LOW);
            digitalWrite(G,                       HIGH);
            if (digitalRead(BUTTON) == HIGH) { // Если
кнопка                                           нажата
                delay(500); // Задержка в 0.5 секунду
                v = 0;  // Устанавливаем переменную v в
значение                                                 0
                }
        }
}
```

Переменная byte v = 0; используется в программе для перехода одного цикла while к другому. При нажатии на кнопку значение переменной v меняется. В программе поставлена небольшая задержка в каждом условии для защиты от быстрого перехода от одного цикла while в другой.

На данном занятии перечислены основы работы семисегментного индикатора с использованием Arduino. Рассказано, как подключить семисегментный индикатор к плате Arduino и считывать его состояние. В заключение можно отметить, что семисегментные индикаторы являются удобными средствами отображения информации, которые могут быть использованы во многих приложениях.

В целом, использование микроконтроллера Arduino и его модулей в механике и инженерии, открывает широкий широкий спектр возможностей для

создания инновационных и функциональных решений. Arduino обладает простым и понятным программным интерфейсом, что делает его доступным даже для начинающих разработчиков и инженеров.

ТЕМА 7. МЕХАНИКА И ИНЖЕНЕРИЯ. ПОДВИЖНЫЕ И НЕПОДВИЖНЫЕ СОЕДИНЕНИЯ ДЕТАЛЕЙ ДЛЯ РАЗЛИЧНЫХ УЗЛОВ, ДЕМОНСТРИРУЮЩИХ РАБОТУ ТЕХНИКИ ВОЙСК ПВО

Занятие №29. Основные понятия и соединения

Каждая машина состоит из деталей, число которых зависит от сложности и размеров машины. Так автомобиль содержит около 16 000 деталей (включая двигатель), крупный карусельный станок имеет более 20 000 деталей и т.д.

Чтобы выполнять свои функции в машине детали соединяются между собой определенным образом, образуя подвижные и неподвижные соединения. Например, соединение коленчатого вала двигателя с шатуном, поршня с гильзой цилиндра (подвижные соединения). Соединение штока гидроцилиндра с поршнем, крышки разъемного подшипника с корпусом (неподвижное соединение).

Подвижные соединения определяют кинематику (область физики, описывающая движение точек, тел (объектов) и систем тел (групп объектов) без учета сил, заставляющих их двигаться) машины, а неподвижные – позволяют расчленить машину на отдельные блоки, элементы, детали.

Соединения состоят из соединительных деталей и прилегающих частей соединяемых деталей, форма которых подчинена задаче соединения. В отдельных конструкциях специальные соединительные детали могут отсутствовать.

С точки зрения общности расчетов все соединения делят на две большие группы: неразъемные и разъемные соединения.

Неразъемными называют *соединения*, которые невозможно разобрать без разрушения или повреждения деталей. К ним относятся заклепочные (клепаные), сварные, клеевые соединения, а также соединения с гарантированным натягом. Неразъемные соединения осуществляются силами молекулярного сцепления (сварка, пайка, склеивание) или механическими средствами (клепка, вальцевание, прессование). На рисунке 29.1 представлены примеры неразъемных соединений.

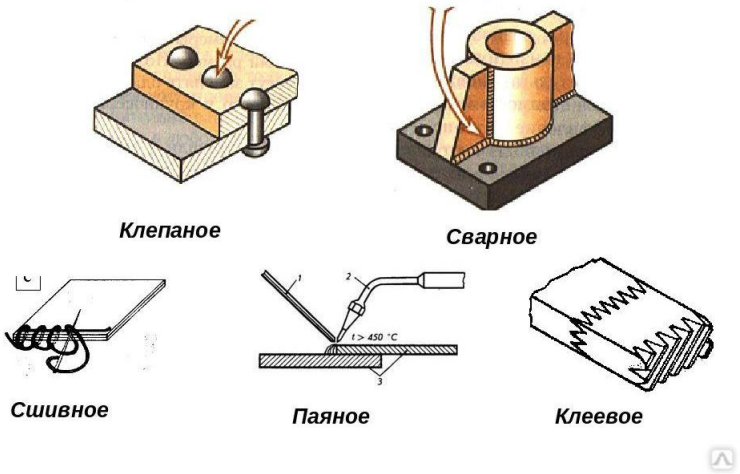

Рис. 29.1. Неразъемные соединения

Разъемными называют *соединения*, которые можно многократно собирать и разбирать без повреждения деталей. К разъемным относятся резьбовые, шпоночные и шлицевые соединения, штифтовые и клиновые соединения. На рисунке 29.2. представлены примеры разъемных соединений.

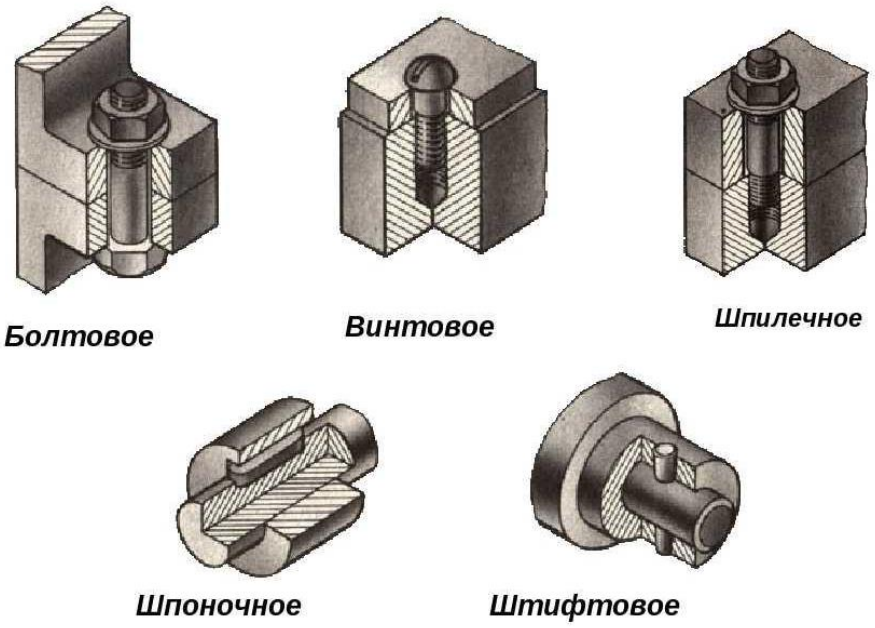

Рис. 29.2. Разъемные соединения

По форме сопрягаемых поверхностей соединения делят на плоское, цилиндрическое, коническое, сферическое, винтовое и т.д.

Выбор типа и вида соединения определяется условиями взаимодействия деталей, требованиями к прочности соединения, условиями работы, требованиями к надежности, долговечности и др.

Подвижные и неподвижные соединения деталей машин для различных узлов, агрегатов и механизмов подбираются с учетом наибольшей целесообразности прочностных характеристик, особенностей монтажа, экономичности (стоимости изготовления и эксплуатации) и т. д.

Сварные соединения применяются обычно для соединения деталей, испытывающих значительные по мощности, но постоянные по направлению нагрузки. Получают сварные соединения при помощи сварочных аппаратов различных типов (электродуговая сварка, газосварка и т.д.). Сварные швы могут быть сплошными, прерывистыми, круговыми. Бывает так же точечная сварка; применяются "электрозаклепки", представляющие собой сварные швы, уложенные внутри отверстия одной из соединяемых деталей на поверхность другой детали.

Пайка, в общем, по технологии и характеристикам сходна со сваркой, но отличается тем, что для пайки применяются специальные составы (припои), как правило на основе олова, свинца и флюсовых добавок. Наиболее широко пайка применяется в радиотехнике, электронике, при соединении деталей гидравлических систем (пайка трубок и штуцеров) и т.д.

Заклепочное (клепаное) соединение применяется в случаях, когда соединяемые детали испытывают знакопеременные нагрузки малой и средней мощности (в том числе вибрации), или знакопеременные нагрузки большой мощности, исключающие работу на срез. Пример: рамы, корпуса, крепление несъемных облицовок и т.п.

Резьбовые соединения применяются повсеместно и являются наиболее распространенным видом соединения в технике. Суть резьбового соединения в применении пары дополнительных деталей, соединяющихся посредством вкручивания одной детали в другую по резьбе, и тем самым соединяющих основные детали. Надежность резьбового соединения обеспечивается за счет силы трения в витках резьбы. Коэффициент трения в правильно соединенных деталях должен превышать коэффициент сдвига основных деталей. Величина коэффициента трения зависит от момента затяжки резьбового соединения, размеров и свойств резьбовой пары. Наиболее распространенными элементами резьбовых соединений являются болты, винты, шпильки, гайки.

Шпоночные и шлицевые соединения применяются при соединении деталей совместного вращения. Чаще всего это валы и зубчатые колеса, валы и шкивы, валы и муфты, а также валы и всевозможные рукоятки, толкатели и т.п.

Шлицевое соединение обеспечивает передачу значительно большего момента, чем шпоночное и применяется в более нагруженных узлах.

Штифтовое соединение обеспечивает неподвижность и точную ориентацию деталей относительно друг друга и применяется, например, для обеспечения соосности отверстий в деталях разъемных корпусов (корпуса редукторов, коробок перемены передач и т.д.).

Проектирование соединений является очень ответственной задачей, поскольку большинство разрушений в машинах происходит именно в местах соединений.

К соединениям в зависимости от их назначения предъявляются требования прочности, плотности (герметичности) и жесткости.

При оценке прочности соединения стремятся приблизить его прочность к прочности соединяемых элементов, т. е. стремятся обеспечить равнопрочность конструкции.

Требование плотности является основным для сосудов и аппаратов, работающих под давлением. Уплотнение разъемного соединения достигается за счет:

- сильного сжатия достаточно качественно обработанных поверхностей;
- введения прокладок из легко деформируемого материала.

При этом рабочее удельное давление q в плоскости стыка должно лежать в пределах $q = (1,5...4)p$, где: p – внутреннее давление жидкости в сосуде.

Экспериментальные исследования показали, что жесткость соединения во много раз меньше жесткости соединяемых элементов, а поскольку жесткость системы всегда меньше жесткости наименее жесткого элемента, то именно жесткость соединения определяет жесткость системы.

Рассмотрим более подробно некоторые виды соединений.

Сварные соединения.

Сваркой называют процесс соединения металлических и пластмассовых деталей путем установления межатомных связей между соединяемыми частями при местном нагреве, пластической деформации или одновременном действии того и другого.

Различают термическую, термомеханическую и механическую сварки. Наиболее распространенными видами сварки являются электродуговая,

электронно-лучевая, газовая (термические); контактная и термокомпрессионная (термомеханические); трением, холодная и ультразвуковая (механические).

При электродуговой сварке электрической дугой в месте контакта электрода и соединяемых деталей расплавляется металл деталей и электрода и образуется прочный шов. Защитная обмазка металлического электрода образует при сварке большое количество шлака и газа, которые обеспечивают устойчивое горение дуги и защищают расплавленный металл от окисления. В месте сварки сильно окисляющихся при нагреве алюминиевых и магниевых сплавов, сплавов титана, высоколегированных сталей электрическую дугу окружают слоем инертного газа, аргона или гелия, что сильно удорожает сварку.

При газовой сварке для нагрева и плавления металлов используют теплоту газового пламени при сжигании ацетилена в кислороде. Такую сварку часто применяют для тонкостенных и легко окисляющихся деталей из металлов, обладающих различными температурами плавления, в частности, для сварки деталей из конструкционных сталей толщиной до 2 мм, меди – до 4 мм. Газовая сварка вызывает небольшие деформации и структурные изменения.

Электронно-лучевую (лазерную) сварку производят потоком электронов (частиц света) большой энергии. Этим способом обычно сваривают тугоплавкие и сильно окисляющиеся металлы и сплавы. Сварку производят в вакууме или в атмосфере аргона.

Контактная сварка – самый производительный способ сварки в массовом производстве. Различают точечную, стыковую и роликовую (шовную) контактные сварки.

При точечной сварке тонкостенные детали соединяют внахлестку. Под действием давления электродов, проводящих ток к месту сварки, образуются точечные сварные соединения. Так как высокие температуры действуют на небольших участках (точках), отсутствует коробление соединяемых деталей. Точечную сварку используют при изготовлении кожухов, панелей, шасси, стоек и других деталей.

При стыковой сварке соединяемые детали сжимают и в зоне контакта при прохождении электрического тока выделяется большое количество теплоты. Стыковой сваркой соединяют детали различных форм и сечений (круг, квадрат, труба, уголок и т.д.).

Шовную сварку осуществляют вращающимися дисковыми электродами. При этом получается непрерывный сварной шов, обеспечивающий герметичное соединение тонкостенных деталей.

Разновидностью контактной сварки является конденсаторная сварка – ток к месту сварки подается в виде короткого импульса при разряде конденсаторов. Контактная сварка позволяет сваривать разнородные материалы, детали малых толщин и сечений (сварка в “шарик” монтажных приводов) и детали различных сечений.

Термокомпрессионная сварка – это сварка под давлением с местным нагревом участка соединения за счет теплопередачи от нагретого электрода. Применяется для присоединения металлических проводников толщиной в десятки микрон к полупроводниковым кристаллам, к напыленным пленкам, т.е. при монтаже элементов микросхем.

При сварке трением нагрев в месте соединения осуществляется за счет теплоты, выделяемой в месте контакта прижатых друг к другу и вращающихся по отношению друг к другу деталей.

Холодная сварка осуществляется без нагрева соединяемых деталей за счет их сжатия с помощью механических и гидравлических прессов до появления пластических деформаций. Холодной сваркой сваривают металлы с хорошими пластическими свойствами – алюминий и его сплавы, медь и ее некоторые сплавы; никель; олово; серебро; разнородные металлы, например, алюминий и медь. Для получения прочных и плотных швов необходимо предварительно очистить поверхности контакта от окислов. Прочность соединения при точечной холодной сварке может быть выше, чем при точечной контактной сварке, но при этом значительно хуже внешний вид соединения из-за вмятин и пластической деформации.

Ультразвуковая сварка основана на создании в месте соединения деталей переменных напряжений сдвига с частотой ультразвуковых генераторов, преобразующих колебания электрических величин в механические колебания. Ультразвуковая сварка позволяет сваривать металлы с различными, в том числе неметаллическими покрытиями, пластмассы.

В зависимости от выбранного вида сварки и требований, предъявляемых к соединению, применяют различные виды соединений.

В зависимости от взаимного расположения соединяемых элементов различают следующие виды сварных соединений: стыковые, нахлесточные, угловые и тавровые.

В зависимости от расположения по отношению к направлению нагрузки сварные швы делят на лобовые – шов перпендикулярен направлению нагрузки, фланговые – шов параллелен направлению нагрузки, косые и комбинированные.

Достоинствами сварных соединений являются высокая производительность, равнопрочность, герметичность, возможность соединения различных материалов и деталей разных форм.

Недостатки сварных соединений: появление остаточных напряжений в местах сварки за счет локального нагрева, что может привести к деформации свариваемых деталей; недостаточная вибрационная и ударная прочность; необходимость проведения термической обработки для снятия остаточных напряжений; сложность контроля дефектов и качества соединения.

Из неметаллических материалов сварке подвергаются только термопластические пластмассы (полиэтилен, полистирол, полипропилен и др.), при этом кромки деталей разогреваются до пластического вязкотекучего состояния, а затем подвергаются сжатию. Известны следующие способы сварки пластмасс: ультразвуком, токами высокой частоты, трением, газовыми теплоносителями и нагретыми инструментами.

Соединения пайкой.

Пайкой называют процесс соединения металлических или металлизированных деталей с помощью дополнительного связующего материала – припоя, температура плавления которого ниже температуры плавления материала соединяемых деталей.

Хорошее соединение пайкой можно получить только при чистых поверхностях спаиваемых деталей, свободных от окислов и загрязнений и при заполнении зазора между деталями припоем. Для очистки и защиты соединяемых поверхностей и припоя от окисления, улучшения смачиваемости и лучшего растекания припоя применяют флюсы. Они способствуют очищению поверхностей от загрязнений, растворяют окисные пленки, улучшают смачиваемость поверхностей припоем, обеспечивают лучшее затекание припоя в зазоры между спаиваемыми деталями.

Достоинствами пайки являются простота и дешевизна технологического процесса, широкие возможности его механизации и автоматизации, возможность

соединения всех металлов и разнородных материалов (металл с керамикой, стеклом, резиной), малые остаточные температурные напряжения и деформации, малое электросопротивление мест соединения.

Так как непосредственная пайка при соединении металлов с неметаллами невозможна, то на поверхности неметаллических материалов создают промежуточный слой из меди, никеля, серебра, который хорошо сцепляется с поверхностью этих материалов и обеспечивает качественную пайку с металлом.

Недостатком соединений пайкой является их невысокая механическая и термическая прочность.

Заклепочные (клепаные) соединения.

Заклепочные (клепаные) соединения выполняют с помощью специальных крепежных деталей – заклепок или непосредственным расклепыванием цапф деталей. Форма и размеры заклепок стандартизированы. Стержень заклепки может быть сплошным или полым; головки по форме бывают полукруглые, потайные, полупотайные, плоские.

Заклепки изготавливают из пластичных материалов: низкоуглеродистых сталей (Ст2, Ст3, 08, 10), меди (М1), латуни (Л62), алюминиевых сплавов.

Достоинствами заклепочных соединений являются возможность соединения различных материалов, хорошая сопротивляемость вибрационным и ударным нагрузкам, удобство и надежность контроля качества соединения.

К недостаткам относятся трудоемкость (разметка, сверление отверстий, закладка и клепка заклепок) и высокая стоимость; ослабление соединяемых деталей отверстиями; дополнительный расход материала на накладки.

Клеевые соединения.

Склеиванием называют соединение деталей тонким слоем быстротвердеющего раствора – клея.

Процесс склеивания состоит из подготовки соединяемых поверхностей деталей, нанесения клея, соединения деталей и выдержки при определенных давлении и температуре.

Клеевые соединения применяют для скрепления деталей из различных металлических и неметаллических (стекло, керамика, пластмасса) материалов в любом их сочетании. К клеевым соединениям не предъявляют требований высокой прочности, но они должны хорошо сопротивляться вибрациям, воздействию влаги, колебаниям температур.

Соединения бывают чисто клеевые и клеемеханические, для повышения герметичности (клеерезьбовые, клеесварные).

Клеевые соединения улучшают герметизацию, снижают стоимость изделия и позволяют проще решать задачи миниатюризации конструкций. Их часто применяют в тех случаях, когда невозможно механическое крепление соединяемых деталей, например, склеивание оптического стекла с помощью прозрачных и неокрашенных клеев, крепление полупроводникового кристалла с кристаллодержателем.

Прочность клеевого соединения зависит от способа подготовки поверхностей. Желательно, чтобы они были шероховатые. Для этого применяют механическую (абразивную) и химическую (травление в растворах) обработку.

Соединения заформовкой.

Заформовка заключается в соединении металлических элементов (арматуры) со стеклом, пластмассами, резиной, легкоплавкими цинковыми, алюминиевыми и магниевыми сплавами путем погружения этих элементов в формуемый материал, находящийся в вязкотекучем пластичном или жидком состоянии. После застывания формующего материала образуется неразъемное соединение. Таким способом получают различные рукоятки, крышки, сайлент-блоки, клеммовые держатели, детали для электроизмерительных, оптико-механических и электронных приборов.

Заформовка является единственным способом получения газонепроницаемого соединения металлических электродов со стеклянными баллонами электровакуумных устройств.

Соединения заформовкой имеют следующие достоинства: не требуются высокие точность и чистота обработки погружаемых частей арматуры; можно получить необходимые, часто несовместимые местные свойства элементов узла – электро- и теплопроводность арматуры при сохранении изоляционных свойств узла; уменьшаются масса изделий и расход металла, стоимость.

При формовке практически отсутствует сцепление арматуры с формуемым материалом. Прочность и плотность соединений обеспечивают выбором соответствующих форм погружаемой арматуры в виде кольцевых проточек, впадин, уступов, уширений, загибов, увеличивающих поверхности контакта и препятствующих ее выдергиванию.

Соединения запрессовкой.

Соединения *запрессовкой* получают путем создания гарантированного натяга между охватываемой и охватывающей поверхностями при сборке. После сборки вследствие упругих и пластических деформаций на поверхности контакта возникает удельное давление и соответствующие ему силы трения, препятствующие взаимному смещению деталей.

Сборка при соединении запрессовкой может осуществляться одним из трех способов: прессование без нагрева, с нагревом втулки или с охлаждением вала. Наиболее распространены соединения запрессовкой по цилиндрическим поверхностям. Они применяются для соединения зубчатых колес на валиках, при соединении зубчатого венца червячного колеса со ступицей.

Достоинствами соединений запрессовкой являются: отсутствие дополнительных креплений, простота конструкции, хорошая центровка сопрягаемых деталей, возможность передачи значительных осевых усилий и крутящих моментов.

К недостаткам соединений запрессовкой относятся: высокие точность и стоимость изготовления соединяемых деталей, сложность сборки, влияние величины натяга, коэффициента трения и рабочих температур на прочность соединения.

В заключении можно отметить, что правильный выбор соединения обеспечивает высокую надежность и долговечность конструкции.

На данном занятии рассказано об основных понятиях и соединениях в механике и инженерии, что дает нам понимание как создать надежную конструкцию для собственного робота.

Занятие №30. Передача кинематических цепей

Механическая энергия, используемая для приведения в движение “машины-орудия”, представляет собой энергию вращательного движения вала двигателя. Вращательное движение получило наибольшее распространение в механизмах и машинах, так как обладает следующими достоинствами: обеспечивает непрерывное и равномерное движение при небольших потерях на трение; позволяет иметь простую и компактную конструкцию передаточного механизма.

Все современные двигатели для уменьшения габаритов и стоимости выполняют быстроходными с весьма узким диапазоном изменения угловых скоростей. Непосредственно быстроходный вал двигателя соединяют с валом

машины редко (вентиляторы и т. п.). В абсолютном большинстве случаев режим работы “машины-орудия” не совпадает с режимом работы двигателя, поэтому передача механической энергии от двигателя к рабочему органу машины осуществляется с помощью различных передач.

Передачей будем называть устройство, предназначенное для передачи энергии из одной точки пространства в другую, расположенную на некотором расстоянии от первой.

В современном машиностроении в зависимости от вида передаваемой энергии применяют механические, пневматические, гидравлические и электрические передачи. В данном курсе рассматриваются только наиболее распространенные механические передачи.

Механическими передачами, или просто передачами, называют механизмы для передачи энергии от машины-двигателя к машине-орудию, как правило, с преобразованием скоростей, моментов, а иногда - с преобразованием видов (например, вращательное в поступательное) и законов движения.

Передача (в механике) соединяет вал источника энергии двигателя и валы потребителей энергии - рабочих органов машины, таких, например, как ведущие колеса гусеничного движителя или автомобиля.

Механические передачи известны со времен зарождения техники, прошли вместе с ней длительный путь развития и совершенствования и имеют сейчас очень широкое распространение. Грамотная эксплуатация механических передач требует знания основ и особенностей их проектирования и методов расчетов.

При проектировании к механическим передачам предъявляются следующие требования:

- высокие нагрузочные способности при ограниченных габаритных размерах, весе, стоимости;
- постоянство передаточного отношения или закона его изменения;
- обеспечение определенного взаимного расположения осей ведущего и ведомого валов, в частности, межосевого расстояния aw;
- малые потери при передаче мощности (высокий кпд) и, как следствие, ограниченный нагрев и износ;
- плавная и бесшумная работа;
- прочность, долговечность, надёжность.

Передачи имеют широкое распространение в машиностроении по следующим причинам:

- энергию целесообразно передавать при больших частотах вращения;

- требуемые скорости движения рабочих органов машин, как правило, не совпадают с оптимальными скоростями двигателя; обычно ниже, а создание тихоходных двигателей вызывает увеличение габаритов и стоимости;

- скорость исполнительного органа в процессе работы машины-орудия необходимо изменять (например, у автомобиля, грузоподъемного крана, токарного станка), а скорость машины-двигателя чаще постоянна (например, у электродвигателей);

- нередко от одного двигателя необходимо приводить в движение несколько механизмов с различными скоростями;

- в отдельные периоды работы исполнительному органу машины требуется передать вращающие моменты, превышающие моменты на валу машины-двигателя, а это возможно выполнить за счет уменьшения угловой скорости вала машины-орудия;

- двигатели обычно выполняют для равномерного вращательного движения, а в машинах часто оказывается необходимым поступательное движение с определенным законом;

- двигатели не всегда могут быть непосредственно соединены с исполнительными механизмами из-за габаритов машины, условий техники безопасности и удобства обслуживания;

- распределять работу двигателя между несколькими исполнительными органами машины.

Как правило, угловые скорости валов большинства используемых в настоящее время в технике двигателей (поршневых двигателей внутреннего сгорания, газотурбинных, электрических, гидравлических и пневматических двигателей) значительно превышают угловые скорости валов исполнительных или рабочих органов машин, порой на 2-3 порядка. Поэтому доставка (передача) энергии двигателя с помощью передачи любого типа, в том числе и механической, происходит, как правило, совместно с одновременным преобразованием моментов и угловых скоростей (в сторону повышения первых и понижения последних).

При этом необходимо отметить, что конструктивное обеспечение функции транспортного характера – чисто передачи энергии иной раз вступает в логическое противоречие с направлением задачи конечного преобразования силовых и скоростных параметров этой энергии. Например, в трансмиссиях многих транспортных машин (особенно высокой проходимости) входной редуктор сначала повышает частоту вращения, понижение ее до требуемых пределов производят бортовые или колесные редукторы.

Этот прием позволяет снизить габаритно-весовые показатели промежуточных элементов трансмиссии (коробок перемены передач, карданных валов) – размеры валов и шестерен пропорциональны величине передаваемого крутящего момента в степени 1/3.

Аналогичный принцип используется при передаче электроэнергии – повышение напряжения перед ЛЭП позволяет значительно снизить тепловые потери, определяемые в основном силой тока в проводах, а заодно уменьшить сечение этих проводов.

Иногда передача механической энергии двигателя сопровождается также преобразованием вида движения (например, поступательного движения во вращательное или наоборот) или законов движения (например, равномерного движения в неравномерное).

Широко известными образцами таких передач являются кривошипно-шатунный механизм и кулачковый привод механизма газораспределения.

Классификация механических передач, применяемых в машиностроении, представлена ниже.

По энергетической характеристике механические передачи делятся на: кинематические (передаваемая мощность $P<0,1$ кВт); силовые (передаваемая мощность $P\geq0,1$ кВт).

По принципу передачи движения: передачи трением; действующие за счет сил трения, создаваемых между элементами передач.

Фрикционные передачи подразделяют на: фрикционные передачи с жесткими звеньями (с различного рода катками, дисками); фрикционные передачи с гибким звеном (ременные, канатные); работающие в результате возникновения давления между зубьями, кулачками или другими специальными выступами на деталях.

Как фрикционные, так и зубчатые передачи могут быть выполнены с непосредственным контактом ведущего и ведомого звеньев или посредством гибкой связи – ремня, цепи.

По способу соединения деталей: передачи с непосредственным контактом тел вращения; передачи с гибкой связью.

По характеру изменения скорости: повышающие (мультипликаторы); понижающие (редукторы); регулируемые (со ступенчатым регулированием и бесступенчатым (плавным) регулированием); нерегулируемые

По взаимному расположению валов в пространстве: с параллельными валами - зубчатые с цилиндрическими колесами, фрикционные с цилиндрическими роликами, цепные; с пересекающимися валами - зубчатые и фрикционные конические, фрикционные лобовые; с перекрещивающимися валами - зубчатые - винтовые и коноидные, червячные, лобовые фрикционные со смещением ролика; с соосными валами.

По характеру изменения передаточного отношения (числа): передачи с постоянным (неизменным) передаточным отношением; передачи с переменным (изменяемым или по величине, или по направлению или и то и другое вместе) передаточным отношением.

По характеру движения валов: простые передачи, в которых валы вращаются лишь вокруг своих осей, а оси валов и сопряженные с ними детали остаются в пространстве неподвижными; планетарные передачи, в которых оси и сопряженные с ними детали (сателлиты) перемещаются в пространстве, разновидностью планетарных передач являются волновые передачи.

По подвижности осей и валов: передачи с неподвижными осями валов - рядовые (коробки скоростей, редукторы); передачи с подвижными осями валов (планетарные передачи, вариаторы с поворотными роликами).

По числу ступеней (т.е. отдельных передач, взаимно связанных и одновременно участвующих в передаче и преобразовании движения): одноступенчатые; многоступенчатые.

По конструктивному оформлению: открытые (не имеют общего закрывающего их корпуса); полузакрытые, смонтированные в легкий защитный кожух, который не выполняет силовых функций; закрытые, заключенные в общий прочный и жесткий корпус, объединяющий все подшипниковые узлы и выполняющий герметизацию, и постоянную смазку передачи.

Передача, в которой энергия с входного на выходное звено передается через несколько параллельно расположенных механизмов, называется многопоточной передачей. К таким передачам относятся также разветвленные передачи – приводы от одного двигателя нескольких исполнительных механизмов. Многопоточными являются волновые зубчатые и планетарные передачи, так называемые передачи с многопарным зацеплением. Многопарное зацепление – это такое зацепление, в котором одновременно находятся две и большее число пар зубьев. В многопоточной передаче, благодаря распределению нагрузки между параллельно работающими механизмами, кинематическими цепями или кинематическими парами, уменьшены габаритные размеры и масса.

Зубчатые передачи.

Зубчатой передачей называется трехзвенный механизм, в котором два подвижных звена являются зубчатыми колесами, или колесо и рейка с зубьями, образующими с неподвижным звеном (корпусом) вращательную или поступательную пару.

Зубчатая передача состоит из двух колес, посредством которых они сцепляются между собой. Зубчатое колесо с меньшим числом зубьев называют шестерней, с большим числом зубьев – колесом. Термин “зубчатое колесо” является общим. Параметрам шестерни приписывают индекс 1, а параметрам колеса – 2.

Основными преимуществами зубчатых передач являются:

- постоянство передаточного числа (отсутствие проскальзывания);
- компактность по сравнению с фрикционными и ременными передачами;
- высокий КПД (до 0,97…0,98 в одной ступени);
- большая долговечность и надежность в работе (например, для редукторов общего применения установлен ресурс ~ 30 000 ч);
- возможность применения в широком диапазоне скоростей (до 150 м/с), мощностей (до десятков тысяч кВт).

Недостатки зубчатых передач:

- шум при высоких скоростях;
- невозможность бесступенчатого изменения передаточного числа;
- необходимость высокой точности изготовления и монтажа;
- незащищенность от перегрузок;

- наличие вибраций, которые возникают в результате неточного изготовления и неточной сборки передач.

Червячные передачи.

Червячные передачи применяют для передачи движения между перекрещивающимися осями, угол между которыми, как правило, составляет 90°. Движение в червячных передачах передается по принципу винтовой пары. В отличие от большинства разновидностей зубчатых в червячной передаче окружные скорости на червяке и на колесе не совпадают. Они направлены под углом и отличаются по значению. При относительном движении начальные цилиндры скользят. Большое скольжение является причиной низкого КПД, повышенного износа и заедания. Для снижения износа применяют специальные антифрикционные пары материалов: червяк – сталь, венец червячного колеса – бронза (реже – латунь, чугун).

Достоинства червячных передач:

- большие передаточные отношения;
- плавность и бесшумность работы;
- высокая кинематическая точность;
- самоторможение.

Недостатки червячных передач:

- низкий КПД;
- высокий износ, заедание;
- использование дорогих материалов;
- высокие требования к точности сборки.

Цепные передачи.

Цепная передача состоит из двух колес с зубьями (звездочек) и охватывающей их цепи. Наиболее распространены передачи с втулочно-роликовой цепью и зубчатой цепью. Цепные передачи применяются для передачи средних мощностей (не более 150 кВт) между параллельными валами в случаях, когда межосевые расстояния велики для зубчатых передач.

Преимуществами цепных передач являются:

- отсутствие проскальзывания;
- достаточная быстроходность (20-30 м/с);

- сравнительно большое передаточное число (7 и более);
- высокий КПД;
- возможность передачи движения от одной цепи нескольким звездочкам;
- небольшая нагрузка на валы, т.к. цепная передача не нуждается в предварительном натяжении цепи необходимом для ременной передачи.

Недостатками цепных передач являются:

- вытяжка цепей вследствие износа шарниров;
- более высокая стоимость передачи по сравнению с ременной;
- необходимость регулярной смазки;
- значительный шум.

По назначению цепи подразделяют на приводные, используемые в приводах машин; тяговые, применяемые в качестве тягового органа в конвейерах, и грузовые, используемые в грузоподъемных машинах для подъема грузов. Цепные передачи применяются, например, для управления рулем направления самолета, для привода механизма отклонения триммера руля высоты.

Ременные передачи.

Сравнивая ременную передачу с зубчатой, можно отметить следующие преимущества:

- возможность передачи движения на значительное расстояние (до 15 м и более);
- плавность и бесшумность работы, обусловленные эластичностью ремня и позволяющие работать при высоких скоростях;
- способность выдерживать перегрузки (до 300 %) благодаря увеличению скольжения ремня;
- невысокая стоимость;
- простота обслуживания и ремонта.

Основными недостатками ременной передачи являются:

● непостоянство передаточного отношения из-за скольжения ремня на шкивах;

- значительные габаритные размеры при больших мощностях (для одинаковых условий диаметры шкивов примерно в 5 раз больше диаметров зубчатых колес);
- большое давление на шкивы в результате натяжения ремня;
- низкая долговечность ремней (от 1000 до 5000 ч).

Ременные передачи применяют преимущественно в тех случаях, когда по условиям конструкции валы расположены на значительных расстояниях. Мощность современных передач не превышает 50 кВт.

В многоступенчатых приводах ременную передачу применяют обычно в качестве быстроходной ступени, устанавливая ведущий шкив на валу двигателя. В таком случае габариты и масса передачи будут наименьшими.

Фрикционные передачи.

Передачи, работа которых основана на использовании сил трения, возникающих между рабочими поверхностями двух прижатых друг к другу тел вращения, называют *фрикционными передачами*. Область применения. Фрикционные передачи с постоянным передаточным отношением применяют сравнительно редко. Их область ограничивается преимущественно кинематическими цепями приборов, от которых требуется плавность движения, бесшумность работы, безударное включение на ходу и т.п.

Фрикционные вариаторы применяют достаточно широко для обеспечения бесступенчатого регулирования скорости в станкостроении, текстильных, бумагоделательных и других машинах и приборах. В авиастроении фрикционные передачи не применяются. Диапазон передаваемых мощностей обычно находится в пределах до 10 кВт, так как при больших мощностях трудно обеспечить необходимое усилие прижатия катков.

Различные варианты соединений применяются в зависимости от конкретных задач, и могут быть использованы для самых сложных элементов конструкции.

На данном занятии перечислены варианты передачи энергии в механических узлах. Эти знания необходимы, чтобы грамотно проектировать подвижные конструкции, которых большое количество в современных роботах.

ТЕМА 8. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ ПО СОЗДАНИЮ ВЫБРАННОГО РОБОТА. ДЕМОНСТРАЦИЯ ВОЗМОЖНОСТЕЙ

Занятие №31. Постановка основных задач проекта

Любой проект требует ясно сформулированной цели, к которой должен стремиться исполнитель. Это может быть, проведение научного исследования, создание программного обеспечения или проектирования сложного устройства. Правильное определение цели устанавливает определенное направление для последующей работы над ее достижением и придает ей смысл.

Обычно, конечная цель объемная и содержит в себе много работы, поэтому необходимо разбить ее на задачи - подцели, связанные с разными направлениями деятельности, необходимыми для выполнения работы. Например, при создании робота задачами могут быть проектирование, изготовление деталей, подбор компонентов, написание программного обеспечения, сборка, тестирование и испытания. Если проект многопрофильный, рекомендуется разбить задачи по каждому направлению для упрощенного понимания конечного результата.

При постановке целей и задач следует проявлять особую ответственность, так как они могут меняться в ходе работы, что усложняет совместную работу. Поэтому чем больше времени и размышлений потрачено на их осмысление, тем проще будет организовать свою работу или работу команды. В командной работе необходимо разделить обязанности и назначить координатора, который будет контролировать выполнение задач и подзадач.

Не стоит описывать всю работу подробно - слишком детальный план может привести к тому, что выполнение одной задачи потребует корректировки всех остальных. Действия должны быть четко обозначены, но без прямых указаний на их выполнение.

Перед началом работы над проектом по робототехнике необходимо определить основные задачи, которые должен выполнять созданный робот. Например, если это робот-пылесос, то его задачей будет очищение пола от пыли и мусора. Если это робот-манипулятор, то его задачей будет подъем и перемещение различных предметов.

Рассмотрим конкретный пример - проект по созданию робота и покажем, какие задачи необходимо поставить на каждом этапе работы.

<u>Пример составления плана работ по проекту.</u>

1. Планирование проекта.

1.1. Определение цели проекта и основных требований к роботу.

1.2. Составление графика работ и распределение задач между участниками проекта.

1.3. Определение бюджета проекта и планирование затрат.

2. Проектирование.

2.1. Создание эскиза робота с удаленным управлением и определение необходимых компонентов.

2.2. Выбор материалов для изготовления робота и его частей.

2.3. Разработка электрической схемы и выбор необходимых электронных компонентов.

2.4. Создание 3D-модели робота для дальнейшего производства.

3. Изготовление.

3.1. Изготовление основной конструкции робота и его отдельных частей.

3.2. Сборка компонентов и подключение их к электрической схеме.

3.3. Программирование микроконтроллера для управления роботом.

3.4. Тестирование работоспособности робота.

4. Доработка.

4.1. Исправление ошибок в программном обеспечении и аппаратуре.

4.2. Устранение дефектов в конструкции и компонентах робота.

5. Тестирование и испытание.

5.1. Проведение тестового контроля работы робота.

5.2. Испытания на прочность и надежность конструкции робота.

5.3. Тестирование управления роботом в различных условиях.

6. Внедрение в эксплуатацию.

6.1. Подготовка робота к работе в реальных условиях.

6.2. Проведение финальных испытаний и контроль работоспособности робота.

Для успешного выполнения проекта необходимо ясно сформулировать цель. Также важно определить бюджет проекта и планировать затраты.

Данное занятие посвящено постановке основных задач проекта, что является важным этапом проектирования, определению целей и временных ограничений. Эта информация позволяет достичь успеха в реализации проекта.

Занятие №32. Реализация проекта

Используя знания, полученные ранее, поставленные задачи и имеющийся план работ команда может приступать к реализации собственных робототехнических проектов.

Первый шаг – необходимо задать назначение робота, какой функционал он будет иметь, в каких условиях будет работать. Далее необходимо определить его внешний вид, форму и размеры, мощность, скорость и надежность.

Для выбора компонентов необходимо учитывать следующие факторы:

- совместимость компонентов между собой;
- надежность и качество компонентов;
- цена компонентов.

В нашем проекте робот будет передвигаться перед ультразвуковым датчиком. Необходимый функционал - прямолинейное движение вперёд и назад на различных скоростях. Движение будет осуществляться по плоскости с рельсом. Длина пути робота составляет 2 метра.

Вторым шагом на этапе определения задач робота является совместное продумывание внешнего вида и расположения модулей робота. На рисунке 32.1 представлен эскиз ультразвукового датчика в корпусе, а на рисунке 32.2. изображен эскиз макета для передвижения робота и базирования датчика.

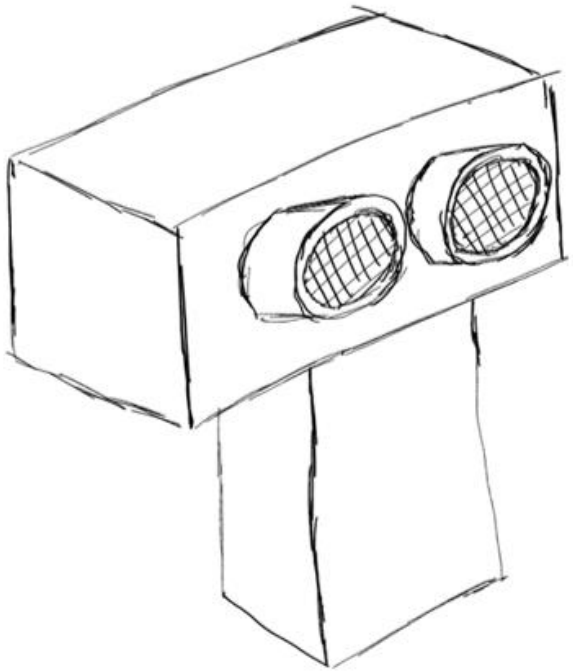

Рис 32.1. Эскиз ультразвукового датчика в корпусе

Третий шаг – создание макета робота (робототехнического комплекса). На основе концепции робота, которую обсуждали на предыдущих этапах, следует сделать прототип, который подтверждает возможность корректной работы такого устройства.

Четвертый шаг – участники команды занимаются подбором необходимых комплектующих, что является одним из самых важных этапов в создании робота. Учитывая ранее обсуждаемую специфику и все факторы, влияющие на конечный результат, участник команды работает над поиском, выбором и совместимостью между компонентами. Для поиска комплектующих можно использовать различные источники информации, такие как специализированные каталоги компонентов, интернет-магазины, форумы и сообщества робототехников, также можно обратиться за помощью к опытным преподавателям или специалистам в области робототехники. После выбора компонентов необходимо проверить их совместимость между собой и с макетом робота. Если компоненты не совместимы, то необходимо выбрать другие компоненты или изменить дизайн макета робота.

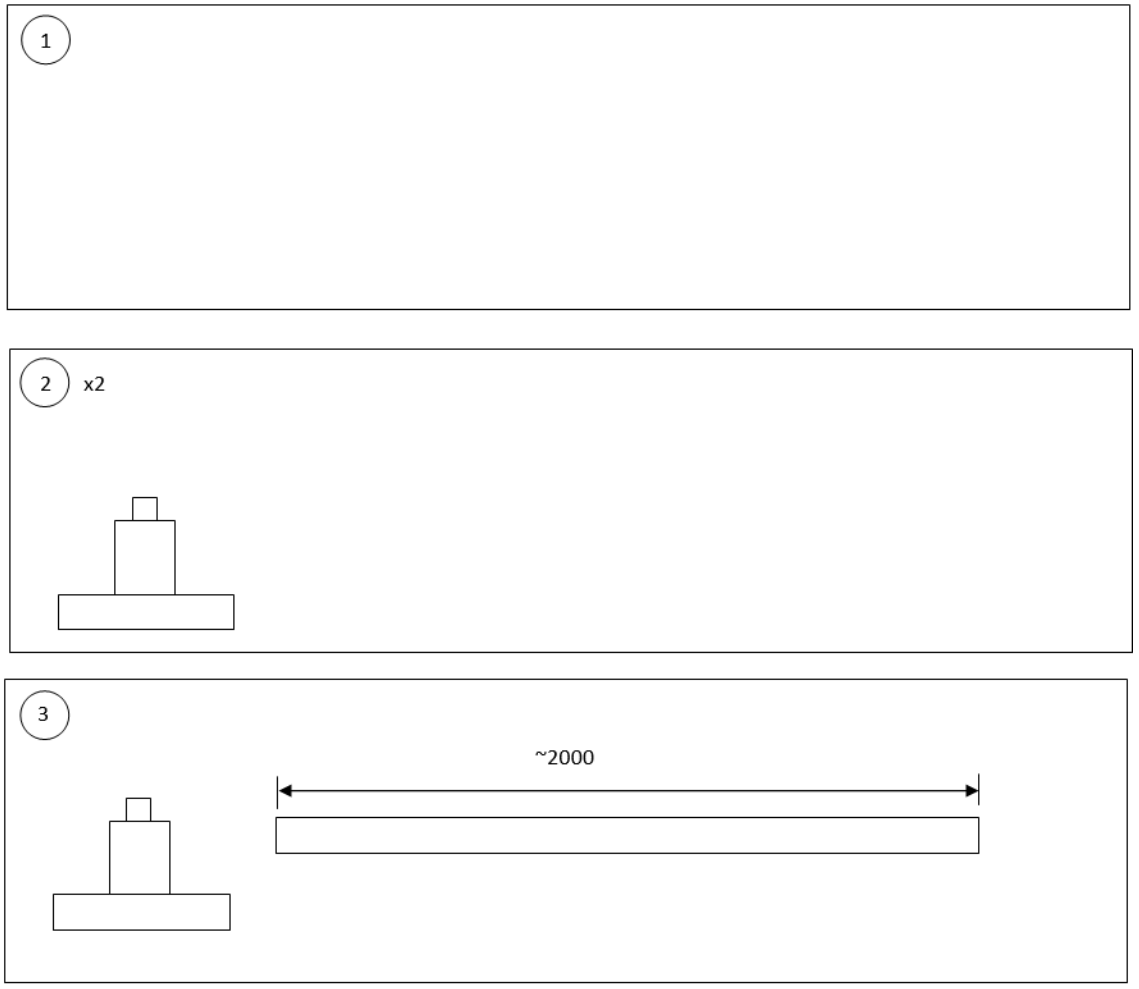

Рис.32.2. Эскиз макета для передвижения робота и базирования датчика

Необходимые компоненты для реализации:

- ультразвуковой дальномер;
- плата Arduino UNO;
- провода мама-папа, папа-папа;
- макетная плата;
- ПО Arduino IDE.

Основной принцип работы установки заключается в определении расстояния до объекта в каждый момент времени. Ультразвуковой датчик генерирует отдельные звуковые импульсы ультразвукового диапазона, то есть такие, которые человеку его ухом не слышно.

И поскольку данные импульсы распространяются через воздух, то движутся они со скоростью звука. Как только этот звук достигает ближайшей границы объекта напротив, он отражается от нее по принципу возникновения

эхо, и тогда датчик, принимая отраженный сигнал, вычисляет расстояние до объекта, от которого произошло отражение.

Сначала фиксируется время, которое прошло между отправкой сигнала и моментом его прихода назад, затем оно умножается на скорость звука, а после - делится на два. Так как расстояние до объекта определяется здесь временем распространения и возврата звуковой волны, точность измерений, выполняемых ультразвуковым датчиком не зависит от помех.

В принципе, любой предмет, отражающий звук, может быть обнаружен независимо от его цвета и освещенности. Это может быть деревянный забор или стеклянное окно, кусок отделки из нержавеющей стали или поликарбонат. Не важно, есть ли на пути ультразвука туман, или мембрана сенсора датчика имеет легкие загрязнения. На функционировании датчика это не скажется.

Передатчик расположен на плате рядом с приемником, так что он способен воспринимать ультразвуковые волны, испущенные приемником и отраженные от объекта, находящегося перед датчиком, если между датчиком и объектом от которого происходит отражение находится воздух.

Когда в зону действия ультразвукового луча попадает какое-нибудь препятствие, схема рассчитывает время, которое проходит с момента отправки ультразвукового сигнала до момента его прихода обратно — в приемник.

Это осуществить легко, тем более электронике, ведь скорость звука в воздухе известна, она равна 343,2 метра в секунду, следовательно, умножив время на данную скорость - получим длину прямолинейной траектории на пути ультразвука от приемника до места отражения и обратно.

Разделив на два — получим расстояние до поверхности отражения, независимо от того, твердая она или мягкая, цветная или прозрачная, плоская или какой-нибудь причудливой формы. А несколько таких датчиков, расположенных под правильными углами, позволят определить и размеры объектов.

Скорость приближающегося объекта находится как отношение изменения координаты за какой то промежуток времени к этому промежутку.

На этом занятии мы научились подбирать комплектующие, создавать макет робота под задачи нашего проекта. В следующем занятии мы продолжим реализовывать наш проект.

Занятие №33. Реализация проекта

Пятый шаг – разработка программного обеспечения для платформы Arduino. Разработка программного обеспечения включает в себя написание кода для управления роботом и выполнения его задач. Для создания программного обеспечения можно использовать различные языки программирования, такие как Python и Си. В случае разработки для платформы Arduino используется язык Си.

Последовательность процесса разработки программного обеспечения

- понять задачи, которые должен выполнять робот;
- написать код для управления моторами, датчиками и другими компонентами для достижения ожидаемого результата;
- проверить работу программного обеспечения на макете робота;
- отладить код и исправить ошибки.

Например, с помощью представленного ниже кода, мы можем получить расстояние до цели с помощью ультразвукового датчика расстояния.

```
uint32_t get_measurement()
{
  uint32_t duration = 0;
  uint32_t cm = 0;
  // Сначала генерируем короткий импульс длительностью 5
микросекунд.
  digitalWrite(PIN_SENSOR_TRIG, LOW);
  delayMicroseconds(5);
  digitalWrite(PIN_SENSOR_TRIG, HIGH);
  // Выставив высокий уровень сигнала, ждем около 10
микросекунд. В этот момент датчик будет посылать сигналы
с частотой 40 КГц.
  delayMicroseconds(10);
  digitalWrite(PIN_SENSOR_TRIG, LOW);
  // Время задержки акустического сигнала на эхолокаторе.
  duration = pulseIn(PIN_SENSOR_ECHO, HIGH);
  // Теперь осталось преобразовать время в расстояние
  cm = (duration / 2) / 29.1;
  return cm;
}
```

Шестой шаг – после того, как каждый член команды выполнил свою задачу, необходимо объединить результаты для получения желаемого робота. На этом этапе команда должна перейти к изготовлению первого прототипа робота. Это означает, что необходимо начать сборку всех модулей и соединение их между собой. Например, колеса должны быть подключены к моторам, а моторы - к плате управления, на которую уже должно быть загружено соответствующее программное обеспечение. На рисунке 32.3 представлен первый прототип робота.

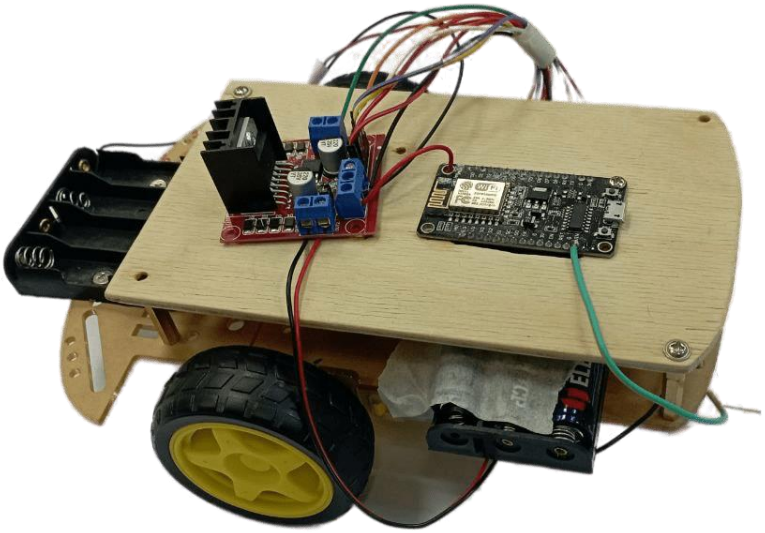

Рис.32.3. Первый прототип робота

Седьмой шаг – после изготовления первого рабочего прототип робота, начинается следующий важный этап - проверка функционала и постепенное внедрение оставшейся функциональности. Эта работа требует совместных усилий всей команды, так как каждый модуль должен быть проверен на работоспособность и соединен с другими модулями. Каждый модуль должен быть тщательно протестирован, чтобы обеспечить эффективность работы робота в дальнейшем.

Восьмой шаг – когда все функции были добавлены и проверены на работоспособность, команда переходит к тестированию робота по заданию, которое было составлено изначально. Это позволяет проверить работу робота в различных условиях и убедиться в том, что он выполняет все необходимые функции. Таким образом, изготовление первого прототипа робота - это только первый большой шаг в разработке. Далее требуется тщательная проверка и

доработка функционала, чтобы обеспечить эффективность работы робота в дальнейшем. Важно не забывать о том, что каждый этап разработки требует совместных усилий всей команды и планомерного подхода к решению возникающих проблем.

Девятый шаг – реализация проекта. На данном этапе необходимо учитывать возможные изменения и дополнения в функционале робота, которые могут возникнуть в процессе работы.

В этом занятии мы научились разрабатывать ПО, а также разработали первый прототип.

Занятие №34. Доработка и демонстрация

На данном этапе участникам команды необходимо предоставить результаты работы преподавателю, друзьям, родителям, сообществу робототехники.

Часто демонстрация проекта - это процесс, в котором учащиеся представляют свой проект робота на публичном мероприятии или конкурсе. Этот этап является важным, так как позволяет ученикам продемонстрировать свои знания и навыки в области робототехники, а также получить обратную связь от экспертов и публики.

Перед демонстрацией проекта необходимо провести подготовительную работу. Команда должна убедиться в том, что робот полностью функционирует и соответствует всем требованиям задания. Также необходимо разработать план демонстрации, который будет включать в себя описание проекта, его основные характеристики и возможности. Важным аспектом демонстрации проекта является коммуникация с публикой. Участники проекта должны быть готовы к ответам на вопросы и объяснению работы робота. Для этого необходимо провести обсуждения и подготовиться к возможным вопросам.

Также необходимо учитывать возможные непредвиденные ситуации, которые могут возникнуть во время демонстрации. Команда должна быть готова к быстрому реагированию и решению проблем, очень важно иметь запасные комплектующие, различные расходные материалы, например, батарейки или аккумуляторы.

Важным аспектом демонстрации проекта является оценка экспертов. Все должны быть готовы к получению конструктивной критики и использованию ее для улучшения проекта в будущем.

Желательно подготовить небольшую презентацию, если будет техническая возможность ее отобразить. Кратко описать, почему вы выбрали этот проект, какие задачи может выполнять данный робот, с наглядной демонстрацией работы.

Таким образом, благодаря полученным знаниям можно разработать проект робота на основе платформы Arduino, демонстрирующий работу радиолокационных средств войск ПВО СВ.

ЗАКЛЮЧЕНИЕ

Обучение образовательной робототехнике является важным элементом школьного образования. Оно позволяет ученикам углубить знания по основным школьным предметам, развивает их практические навыки и помогает определить выбор будущей профессии. Образовательная робототехника тесно связана с другими школьными предметами и может решать задачи обучения, развития и воспитания, учащихся на более высоком уровне. Ее использование позволяет решать сложные проблемы современности с помощью комплексного подхода.

Кроме того, обучение образовательной робототехнике развивает у учащихся навыки программирования, проектирования, использования электронных устройств и решения сложных логических задач. Эти навыки являются важными для будущей карьеры, особенно в сфере IT и науки.

Демонстрация работы радиолокационных методов по определению расстояния до движущегося объекта и его скорости с использованием микроконтроллера Arduino позволяет создавать эффективные тренажеры, симуляторы и прототипы. Это помогает улучшить обучение и тренировки военного персонала, а также тестирует новые идеи и концепции перед их внедрением на практике.

Вышеизложенное указывает на значимость и перспективы использования микроконтроллера Arduino и механики в области инженерии и разработки робототехнического комплекса, способного решить комплексную задачу по имитации методов радиолокации для определения расстояния до движущегося объекта и его скорости. Это позволяет создавать инновационные решения, обеспечивать надежность и эффективность устройств, а также улучшать обучение и тренировки военного персонала.

Arduino и его модули предоставляют мощную и гибкую платформу для разработки электронных проектов. Они имеют широкий набор встроенных функций и библиотек, которые упрощают процесс программирования и управления различными устройствами.

Использование Arduino и его модулей в механике и инженерии позволяет создавать подвижные и неподвижные соединения деталей для различных узлов. Arduino может контролировать двигатели, актуаторы и другие устройства, обеспечивая управление и координацию движений в механизмах.

Разработка проектов, демонстрирующих работу радиолокационных средств войск ПВО СВ, с использованием Arduino и его модулей, открывает возможности для создания прототипов и симуляций военной техники. Это может

включать в себя моделирование движений танков, боевых дронов, систем навигации и многое другое.

Использование Arduino и механических соединений позволяет создавать эффективные и надежные конструкции для различных узлов в технике войск. Например, это может быть механизм крепления оружия, система подъема и опускания оборудования или механизмы управления роботизированными системами.

Arduino обеспечивает программную гибкость и возможность настройки, что позволяет адаптировать системы для различных задач, в том числе и войскового применения. Это позволяет усовершенствовать функциональность и повышать эффективность предпрофессиональной подготовки специалистов для войск ПВО СВ.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Колпаков, С. Г. Классификация роботов по использованию, передвижению и компонентам / С. Г. Колпаков, А. Д. Мячиков. — Текст : непосредственный // Молодой ученый. — 2017. — № 3 (137). — С. 241-244. — URL: https://moluch.ru/archive/137/36438/ (дата обращения: 23.05.2023)

2. Arduino для начинающих: самый простой пошаговый самоучитель / Стюарт Ярнольд; [пер. с англ. М. Райтман]. - Москва: Эксмо, 2017. - 256 с. - (Электроника для начинающих) ISBN 978-5-699-98944-7

3. С. Монк Программируем Arduino. Профессиональная работа со скетчами. - СПБ.: Питер, 2017. ISBN: 978-5-496-02385-6

4. Ардуино [Электронный ресурс]. – Режим доступа: http://arduino.ru/, свободный. – Загл. с экрана.

5. Всероссийский учебно-методический центр образовательной робототехники [Электронный ресурс]. – Режим доступа http://фгосигра.рф/, свободный. – Загл. с экран

6. Копосов Д.Г. Первый шаг в робототехнику: практикум для 5–6 классов / Д.Г. Копосов. – М: БИНОМ. Лаборатория знаний, 2012. – 286 с. ISBN: 978-5-9963-1695-3

7. Ловин Д. Создаем робота-андроида своими руками / Д. Ловин; пер. с англ. Г. Мельникова. – М.: Издат. дом «ДМК Пресс», 2007. – 312 с. ISBN: 5-9706-0032-6

8. Накано Э. Введение в робототехнику / Э. Накано; пер. с яп. А.И. Логинов, А.М. Филатов. – М.: Мир, 1998. – 334 с. ISBN: 5-03-000396-7

9. Образовательная робототехника в начальной школе: учебно-методич. пособие / В.Н. Халамов (рук.) [и др.]. – Челябинск: Взгляд, 2011. – 152 с. ISBN 978-5-93946-191-7

10. Предко М. 123 эксперимента по робототехнике / М. Предко; пер. с англ. В.П. Попова. – М.: НТ Пресс, 2007. – 544 с. ISBN 5-477-00216-6

11. Федеральный государственный образовательный стандарт [Электронный ресурс]. – Режим доступа http://standart.edu.ru, свободный. – Загл. с экран

12. Филиппов С.А. Робототехника для детей и родителей / С.А. Филиппов. – СПб.: Наука, 2013. – 319 с. ISBN 978-5-02-038-200-8

13. Шимов И.В. Применение робототехнических устройств в обучении программированию школьников / И.В. Шимов // Педагогическое образование в России, 2013. – № 1. – С. 185–188. ГСНТИ 14.25.09