Департамент образования Вологодской области

бюджетное профессиональное образовательное учреждение
Вологодской области «Череповецкий металлургический колледж
имени академика И.П.Бардина»

РОБОТОТЕХНИКА. ПРОГРАММИРОВАНИЕ на ARDUINO

Методические рекомендации по проведению учебных занятий по дополнительной общеобразовательной общеразвивающей программе технической направленности «Робототехника»

Составитель: Егорушкин О.И.,
преподаватель БПОУ ВО «ЧМК»

2020

Робототехника. Прикладное программирование. Методические рекомендации по проведению учебных занятий по дополнительной общеобразовательной общеразвивающей программе технической направленности «Робототехника». /Составитель Егорушкин О.И./ - Череповец: Череповецкий металлургический колледж имени академика И.П.Бардина, 2020 –164 с.

Данная методическая разработка рассмотрена на заседании цикловой комиссии «Информационные технологии и вычислительная техника» и рекомендована к применению.

Председатель: / / Молоткова Л.Н.

«__»_____ 2020 г.

Протокол №__

Содержание

Введение

Применение микропроцессорных систем практически во всех электрических устройствах – важнейшая черта технической инфраструктуры современного общества. Робототехника, электроэнергетика, промышленность, транспорт, системы связи существенно зависят от компьютерных систем управления. Микропроцессорные системы встраиваются в измерительные приборы, электрические аппараты, осветительные установки и другие устройства. Целью лабораторного практикума является формирование у обучающихся знаний общей методологии, а также применения конкретных методов проектирования основных разновидностей современных микропроцессорных средств. Ардуино – один из популярнейших микроконтроллеров для создания разнообразных автоматизированных систем.

Представленная методическая разработка нацелена на освоение учащимися навыков программирования микроконтроллеров, создание программ, тестирования, определения параметров, отладки микроконтроллеров, а также на изучение основных этапов разработки и проектирования автоматических устройств.

Данная методическая разработка представляет собой серию лабораторных работ для обучающихся проведению учебных занятий по дополнительной общеобразовательной общеразвивающей программе технической направленности «Робототехника».

В процессе выполнения лабораторных работ обучающиеся отрабатывают первичные навыки работы с платформой Arduino, возможностями платформы, настраивают среду разработки Arduino IDE, разрабатывают коды программы (скетчи), базовых элементов ProcessingC/C++, отрабатывают порядок запуска и компиляции готовых скетчей с учетом особенностей беспаечной макетной платы. Работы выполняются с соблюдением правил использования, правил подготовки платы к работе и электронных компонентов, используемых при работе с платой. Время выполнения каждой лабораторной работы – 90 мин. Каждая лабораторная работа содержит:

- краткий теоретический материал по изученной теме;
- основные цели и задачи, которые должны быть достигнуты по окончанию лабораторной работы;
- четко сформулированные задания по вариантам, подкрепленные иллюстрациями, пояснениями и примерами.

Оценка образовательных достижений обучающихся по дополнительной общеобразовательной общеразвивающей программе технической направленности «Роботехника» осуществляется в ходе текущего контроля с использованием тестирования и экспертной оценки выполнения лабораторной работы. Лабораторная работа входит в состав оценочных средств и предназначена для текущей аттестации и оценки знаний и умений аттестуемых, соответствующих основным показателям оценки результатов подготовки по Программе.

Правила техники безопасности

При проведении лабораторных работ необходимо соблюдать правила техники безопасности, которые прописаны в инструкциях по ТБ данного кабинета, разъяснения даютс преподавателем перед проведением лабораторных работ. После ознакомления с инструкциями по ТБ, проводится инструктаж с записью в журнале по ТБ.

1.Платформа Arduino

1.1 Плата Arduino Uno

Arduino — это электронный конструктор и удобная платформа быстрой разработки электронных устройств для новичков и профессионалов. Платформа пользуется огромной популярностью во всем мире благодаря удобству и простоте языка программирования, а также открытой архитектуре и программному коду. Устройство программируется через USB без использования программаторов.

Микроконтроллер на плате программируется при помощи языка Arduino (основан на языках Си и C++) и собственной среды разработки, которая доступна для бесплатного скачивания. Проекты устройств, основанные на Arduino, могут работать самостоятельно, либо же взаимодействовать с программным обеспечением на компьютере. Большинство плат Arduino построены на 8-битных микроконтроллерах фирмы Atmel. Процессоры работают на тактовой частоте 16 MHz. Платы содержат все необходимые для работы микроконтроллера компоненты, включая схемы питания и кварцевый резонатор. Рабочее напряжение в большинстве случаев 5 вольт. Программирование микроконтроллера можно осуществлять, просто подключив его к компьютеру через порт USB.

На плате есть разъемы, позволяющие подключать внешние схемы к большинству выходов микроконтроллера. Эти разъемы позволяют использовать цифровые и аналоговые входы и выходы, ШИМ генераторы, различные цифровые интерфейсы. Для Arduino выпускается множество плат расширения, которые позволяют использовать плату как основу для управления роботами, подключаться к компьютерным сетям через Ethernet или Wi-Fi.Так как среда Arduino очень популярна, то многие разработчики микроконтроллеров делают свои платы, совместимыми с ней. Это позволит применить навыки, полученные при работе с Arduino, с другими, в том числе гораздо более мощными микроконтроллерами, как представлено на рисунке 1.

Рисунок 1- Плата Arduino Uno

Плата Arduino Uno, изображенная на Рис. 1.2, построена на базе микроконтроллера ATmega328. Платформа имеет 14 цифровых входов/выходов (6 из которых могут

использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, разъем питания, разъем для программатора (ICSP) и кнопку перезагрузки. Для того, чтобы она заработала, ее необходимо подключить к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи.

1.2 Среда разработки Arduino

В этом разделе мы познакомимся со средой разработки Arduino IDE (Integrated Development Environment – интегрированная среда разработки). Эту среду нужно использовать, если у вас есть реальная плата Arduino.

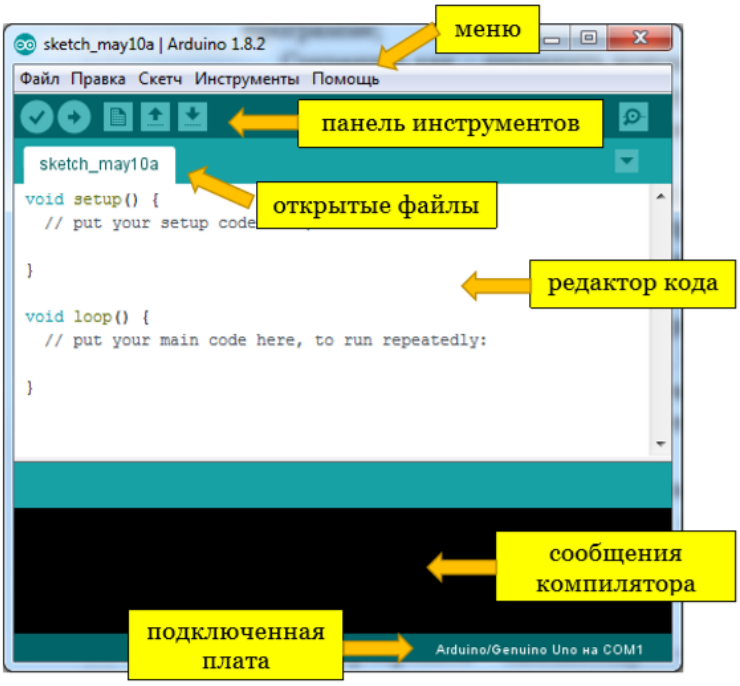


Рисунок 2-Области окна среды разработки Arduino

Строка меню редактора, как представлено на рисунке 2, включает в себя следующие главные элементы: файл, правка, скетч, сервис и справка. Рассмотрим подробнее каждый из них.

В пункте Файл можно найти команды, отвечающие за создание новой программы, загрузки с диска уже существующей, сохранения ее изменений, а также команды для загрузки программы на микроконтроллер:

– «Создать» – создать новую программу (в данной среде программы называются скетчами);

– «Открыть» – открыть существующую программу;

– «Папка со скетчами» – открыть программу из заданной папки;

– «Примеры» – открыть пример программы;

– «Закрыть» – закрыть текущее окно;

– «Сохранить» – сохранить изменения;

– «Сохранить как» – сохранить программу в новом файле;

– «Загрузить» – загрузить программу в Arduino;

– «Загрузить с помощью программатора» – загрузить программу посредством программатора;

– «Настройка печати» – настройка принтера;

– «Печать» – вывод на печать кода программы;

– «Настройки» – настройки редактора;

– «Выход» – выход из Arduino IDE.

Пункт меню Правка содержит команды, связанные с редактирование текста программы, включающие в себя копирование, вставку, настройку отступов и поиск.

В разделе Скетч размещаются команды для управления компиляцией программы:

– «Проверить/Компилировать» – компилировать программу;

– «Показать папку скетчей» – открыть системную папку с программами;

– «Добавить файл» – добавить к проекту файл с данными или программой;

– «Импортировать библиотеку» – подключить к программе библиотеку из списка установленных.

Пункт меню Сервис включает в себя вспомогательные функции для работы с самим микроконтроллером:

– «Автоформатирование» – автоматическая расстановка отступов, переносов строк и т.п.;

– «Архивировать скетч» – архивация папки с программой, и сохранение архива в указанное место;

– «Монитор порта» – открыть окно для обмена данными с микроконтроллером;

– «Плата» – выбор текущей платы (в нашем случае Arduino Uno);

– «Последовательный порт» – выбор порта, к которому подключено устройство;

– «Программатор» – выбор программатора (нами в учебном пособии не используется);

– «Записать загрузчик» – запись программы загрузчика в микроконтроллер (также нами не используется).

Наконец, меню Справка содержит подробное описание всех функций самого редактора Arduino IDE, а также всевозможные команды и приемы работы с платформой. На панель инструментов вынесены в виде кнопок наиболее часто используемые функции среды. Их назначение описано в таблице 1.

Таблица 1- Кнопки на панели инструментов

Кнопка	Описание
	Проверить/Компилировать программу
	Загрузить программу в Arduino
	Создать новую программу
	Открыть существующую программу
	Сохранить программу
	Монитор последовательного порта

Непосредственно текст программы создается и редактируется в окне редактора кода. Это окно представляет собой типичный текстовый редактор с подсветкой синтаксиса программы. В нижней части окна Arduino IDE имеется область, служащая для вывода уведомлений и сообщений об ошибках, возникающих в процессе компиляции программы или во время загрузки программы в микроконтроллер.

1.2.1 Ошибки кода при его компиляции для Arduino Uno

Как несложно догадаться, компиляция – приведение кода на языке Си к виду машинного (двоичного) и преобразование множественных функций в простые операции, чтобы те смогли выполняться через встроенные операнды контроллера. Выглядит всё достаточно просто, но сам процесс компиляции происходит значительно сложнее, и поэтому ошибка во время проведения оной может возникать по десяткам причин. Компилятор действует схожим образом, за исключением того, что причины ошибок указываются далеко не всегда. Именно поэтому рекомендуется протестировать код сначала в среде разработки, и лишь затем уже приступать к его компиляции в исполняемые файлы под Ардуино.Разберёмся, как происходит процесс компиляции:

1. Первое, что делает компилятор – подгружает все инклуднутые файлы, а также меняет объявленные дефайны на значения, которое для них указано. Это необходимо затем, чтобы не нужно было по нескольку раз проходиться синтаксическим парсером в пределах одного кода. Также, в зависимости от среды, компилятор может подставлять функции на место их объявления или делать это уже после прохода синтаксическим парсером. Далее он проверяет первичный синтаксис. Этот процесс проводится в изначальном компилируемом файле, и своеобразный парсер ищет, были ли описаны приведенные функции ранее, подключены ли необходимые библиотеки и прочее. Также проверяется правильность приведения типов данных к определенным значениям. Не стоит забывать, что в C99 используется строгая явная типизация, и вы не можете засунуть в строку, объявленную integer, какие-то буквенные значения. Если такое замечается, сразу вылетает ошибка.
2. В зависимости от среды разработки, иногда предоставляется возможность последний раз протестировать код, который сейчас будет компилироваться, с запуском интерпретатора соответственно.

3. Последним идет стек из различных действий приведения функций, базовых операнд и прочего к двоичному коду, что может занять какое-то время. Также вся структура файлов переносится в исполняемые exe-шники, а затем происходит завершение компиляции.

Как можно увидеть, процесс не так прост, как его рисуют, и на любом этапе может возникнуть какая-то ошибка, которая приведет к остановке компиляции. Проблема в том, что, в отличие от первых трех этапов, баги на последнем – зачастую неявные, но всё ещё не связанные с алгоритмом и логикой программы. Соответственно, их исправление и зачистка занимают значительно больше времени.

Возможны следующие ошибки при компиляции программы и краткие рекомендации по их устранению:

- **ошибка: avrdude: stk500_recv(): programmer is not responding;**

Что делать в этом случае? Первым делом обратите внимание какую плату вы используете и к какому порту она подключена (смотри на скриншоте в правом нижнем углу). Необходимо сообщить Arduino IDE, какая плата используется и к какому порту она подключена. Если вы загружаете скетч в Ардуино Nano V3, но при этом в настройках указана плата Uno или Mega 2560, то вы увидите ошибку, как на скриншоте ниже, как представлено на рисунке 3.

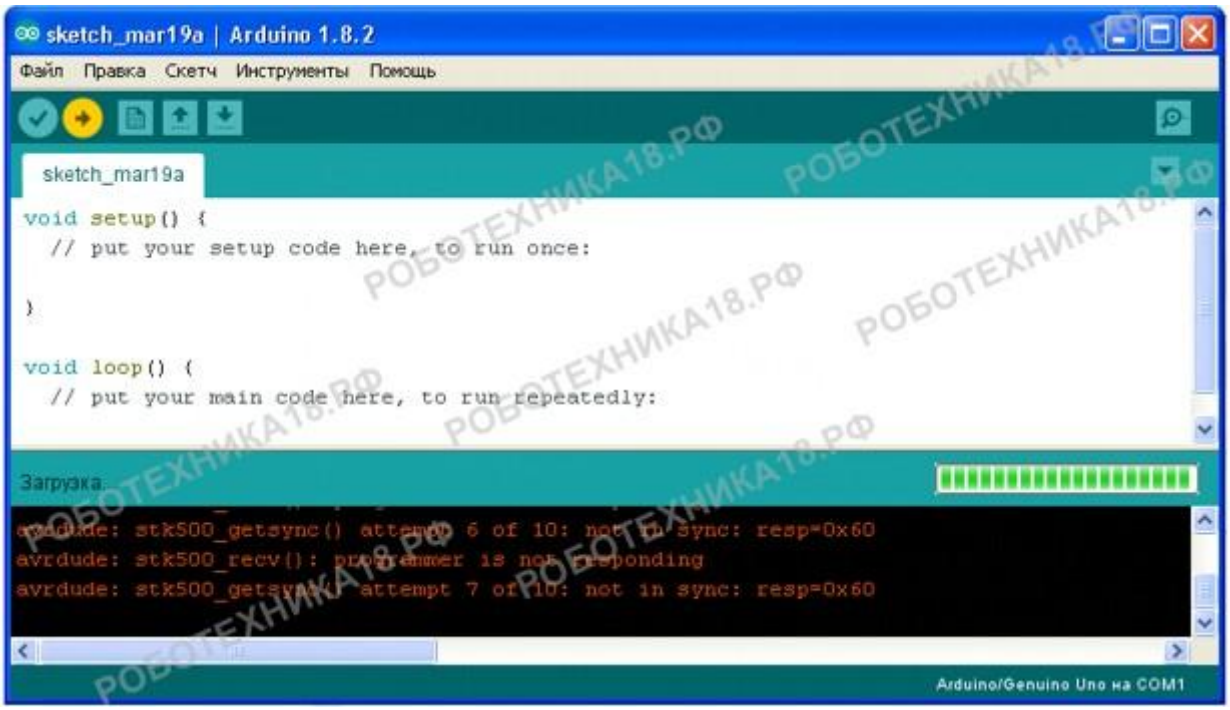


Рисунок 3-Ошибка Ардуино: programmer is not responding

Такая же ошибка будет возникать, если вы не укажите порт к которому подключена плата (это может быть любой COM-порт, кроме COM1). В обоих случаях вы получите сообщение — плата не отвечает (programmer is not responding). Для исправления ошибки надо на панели инструментов Arduino IDE в меню «Сервис» выбрать нужную плату и там же, через «Сервис» → «Последовательный порт» выбрать порт «COM7».

- **ошибка: a function-definition is not allowed here before ‘{‘ token;**

Это значит, что в скетче вы забыли где-то закрыть фигурную скобку. Синтаксические ошибки IDE тоже распространены и связаны они просто с невнимательностью. Такие проблемы легко решаются, так как Arduino IDE даст вам подсказку, стараясь отметить номер строки, где обнаружена ошибка. На скриншоте видно, что строка с ошибкой подсвечена, а в нижнем левом углу приложения указан номер строки, как представлено на рисунке 4.

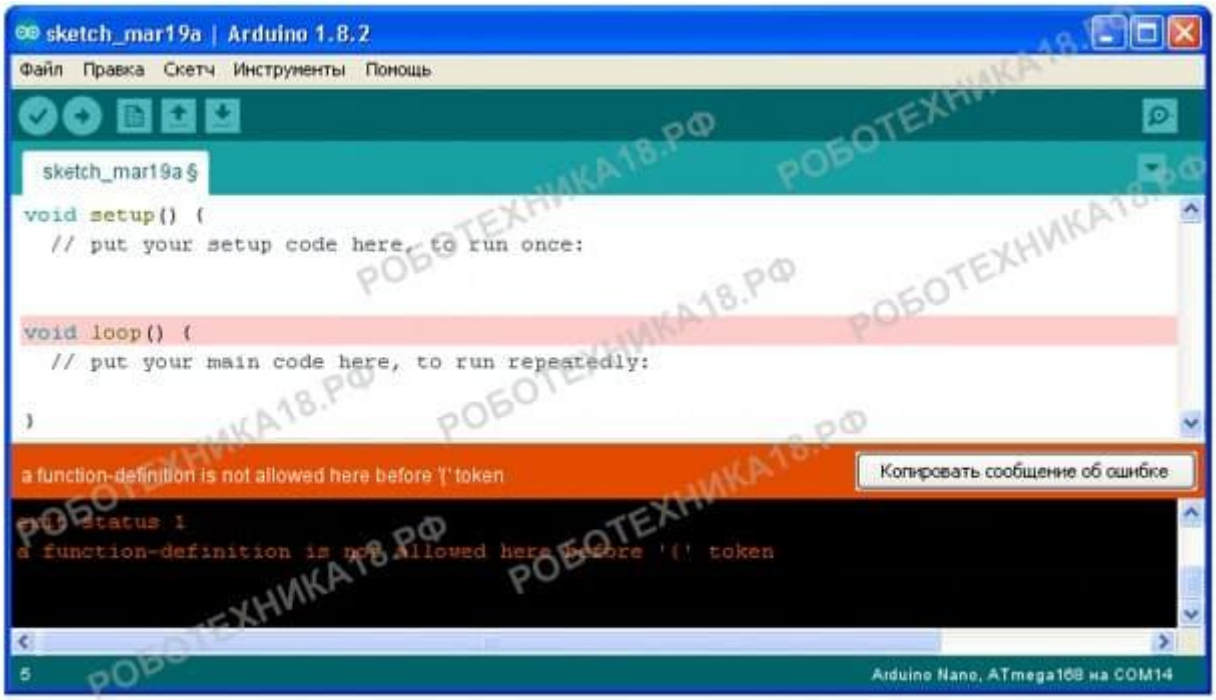


Рисунок 4- Ошибка: a function-definition is not allowed here before ‘{‘ token

- **ошибка: expected initializer before ‘}’ token / expected ‘;’ before ‘}’ token;**

Сообщение expected initializer before ‘}’ token говорит о том, что вы, наоборот где-то забыли открыть фигурную скобку. Arduino IDE даст вам подсказку, но если скетч довольно большой, то вам придется набраться терпения, чтобы найти неточность в коде. Ошибка при компиляции программы: expected ‘;’ before ‘}’ token говорит о том, что вы забыли поставить точку с запятой в конце командной строки.

- **ошибка: ‘ ‘ was not declared in this scope;**

Что за ошибка? Arduino IDE обнаружила в скетче слова, не являющиеся служебными или не были объявлены, как переменные. Например, вы забыли продекларировать переменную или задали переменную ‘DATA’, а затем по невнимательности используете ‘DAT’, которая не была продекларирована. Ошибка was not declared in this scope возникает при появлении в скетче случайных или лишних символов, как представлено на рисунке 5.

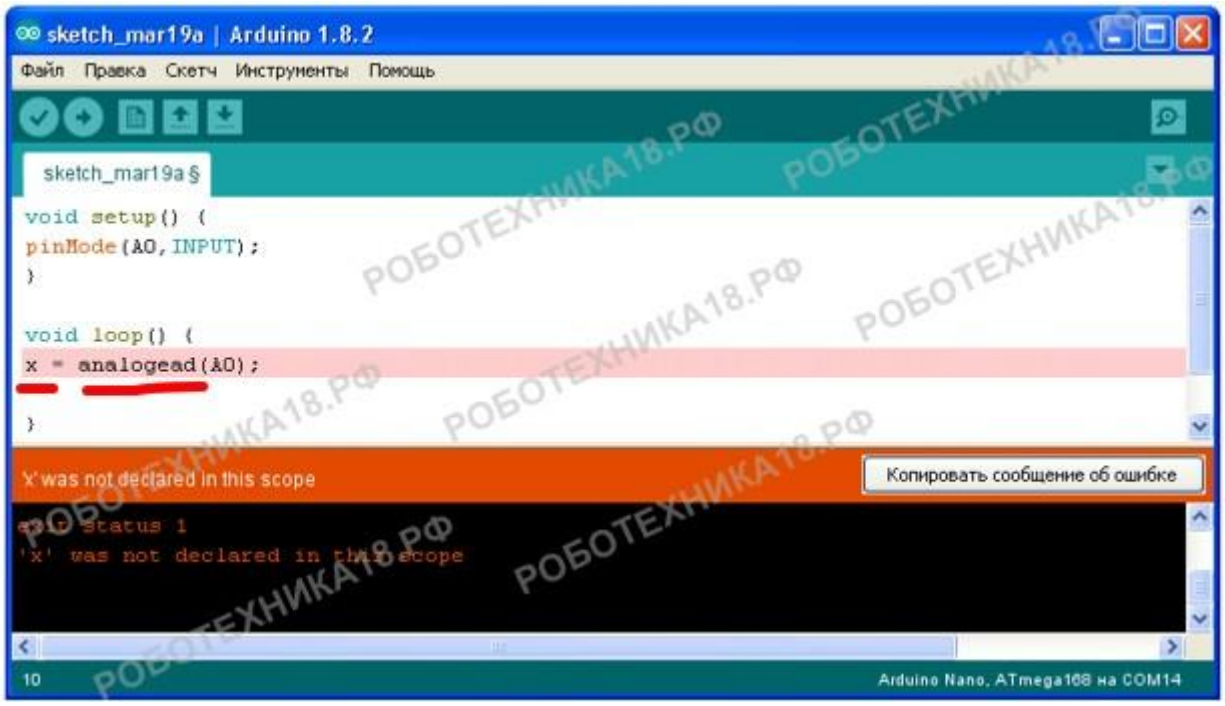


Рисунок 5-Ошибка Ардуино: was not declared in this scope

Например, на скриншоте выделено, что программист забыл продекларировать переменную ‘x’, а также неправильно написал функцию ‘analogRead’. Такая ошибка может возникнуть, если вы забудете поставить комментарий, написали функцию с ошибкой и т.д. Все ошибки также будут подсвечены, а при нескольких ошибках в скетче, сначала будет предложено исправить первую ошибку, расположенную выше.

- **ошибка: No such file or directory / exit status 1;**

Данная ошибка возникает, если вы подключаете в скетче библиотеку, которую не установили в папку libraries. Например, не установлена библиотека ИК приемника Ардуино: fatal error: IRremote.h: No such file or directory. Как исправить ошибку? Скачайте нужную библиотеку и распакуйте архив в папку C:\Program Files\Arduino\libraries. Если библиотека установлена, то попробуйте скачать и заменить библиотеку на новую, как представлено на рисунке 6.

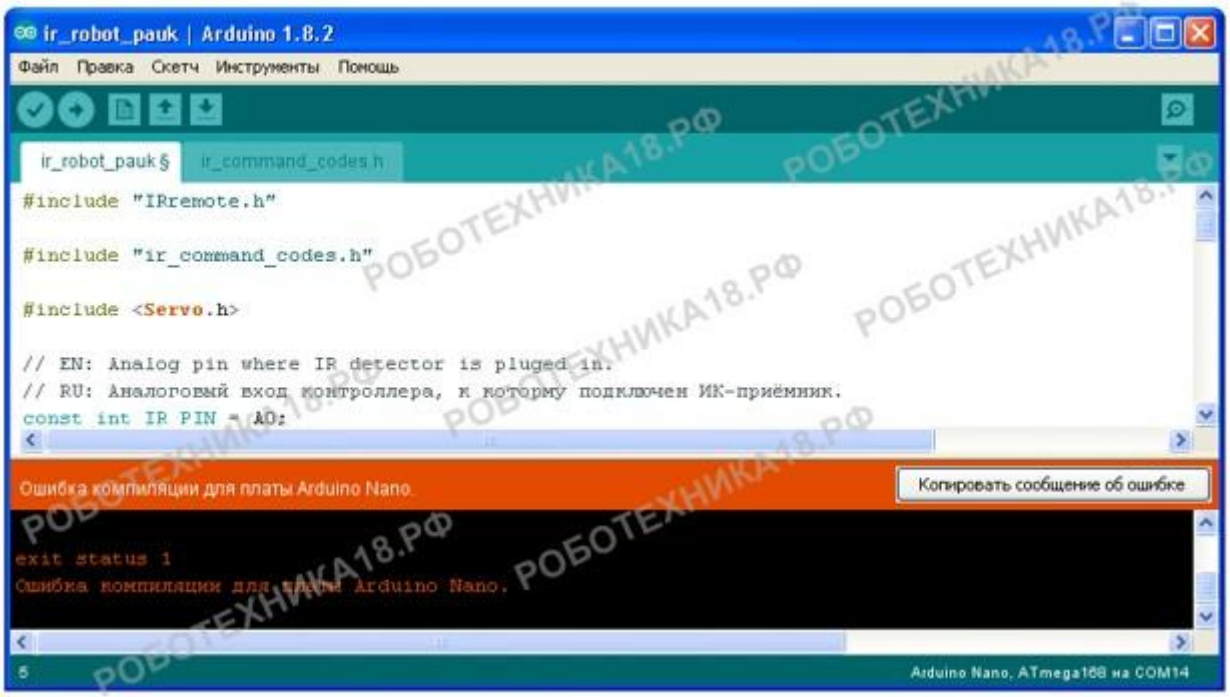


Рисунок 6- Exit status 1 Ошибка компиляции для платы Arduino Nano

Довольно часто у новичков выходит exit status 1 ошибка компиляции для платы arduino/genuino uno. Причин данного сообщения при загрузке скетча в плату Arduino Mega или Uno может быть огромное множество. Но все их легко исправить, достаточно внимательно перепроверить код программы.

- **ошибки типа redefinision of void setup' ;**

Возникают, если в подключаемой библиотеке автор зачем то уже объявил функции, которые в используете в своем скетче. Нужно переименовать методы (у себя или в бибилиотеке).

- **ошибка undefined reference to `loop'.**

Возникает, если вы случайно удалили или переименовали функцию loop. Ардуино не сможет запустить скетч без команду, указанных в этом блоке. Если вы напортачили с функцией setup, то ошибка будет выглядеть соответственно: undefined reference to `setup'. Выход в обоих случаях один – вернуть loop или setup на свое место в скетче.

1.3 Программирование на языке Си

Программирование в среде Arduino осуществляется на языке Си с минимальным набором элементов C++. В дальнейшем будем, для краткости, называть этот язык программирования языком Си. Рассмотрим основные особенности синтаксиса этого языка, знание которых необходимо для написания программ.Каждое выражение заканчивается символом точки с запятой, например:

a = b + c;

Тело функций и составных операторов (if, else, for, while) выделяются фигурными скобками (аналогично BeginEnd в языке Pascal), например:

if(a > 0) {

```
    b = a+1;
}
```

Строки выделяются двойными кавычками: **lcd.print("some text")**, а символы выделяются одинарными кавычками: **symbol = 'a'**.

Подключение библиотек осуществляется с помощью конструкции: **#include <math.h>**.

Комментарии в программе начинаются двумя прямыми слешами: // **это комментарий** .

Объявление переменной в языке Си осуществляется с помощью конструкции вида:

тип_переменной имя_переменной.

Пример:

int x, y; // объявлены переменные x и y, имеющие целый тип

В программах для Arduino можно использовать следующие типы для числовых данных:

– byte – целое число от 0 до 255;

– int – целое число от -32768 до +32767;

– word – целое число от 0 до 65535;

– long – целое число от -2147483648 до 2147483647;

– float и double – числа с плавающей точкой, соответствуют типу float для обычных компьютеров, тип double в настоящее время эквивалентен float.

Массивы в языке Си объявляются конструкцией вида:

- тип_элемента имя_массива[размер];

Пример:

int values[10]; // **массив из десяти целых чисел**

В языке Си предусмотрен тип для хранения символов и строк, это тип **char.** Строки представляют собой массивы типа **char,** в которых последний символ имеет код 0, что обозначает конец строки.

Пример:

char my_str[10]; // **строка не более девяти символов длиной**

Тип boolean используется для представления логических значений true (истина) и false (ложь). Кроме того, любое целое значение может использоваться как логическое, при этом 0 означает ложь, а любое ненулевое значение – истину.

Последний тип данных, который мы рассмотрим – тип **void.** Это «пустой» тип, который используется при объявлении функций, не принимающих аргументов или не

возвращающих результат (аналогично процедурам в языке Pascal). Указание слова **const** перед объявлением переменной говорит о том, что ее значение не может быть изменено.

В языке Си используется стандартный набор математических операторов, включающий +, -, * и /. Для вычисления остатка от деления используется %.

Кроме того, есть их специальные комбинации с оператором присваивания, позволяющие сократить запись:

x += 4; // **Означает x = x + 4.**

Такие формы есть для всех операторов. Для увеличения и уменьшения числа на единицу есть инкременты и декременты:

x++; // **Означает x = x + 1**

y--; // **Означает y = y – 1**

Набор операторов сравнения тоже стандартен и включает операторы <, >, <=, >=. Равенство записывается с помощью двух знаков равно (==), а неравенство записывается как !=. Логические выражения можно вычислять с помощью операторов «и» – &&, «или» – ||. Логическое отрицание записывается с помощью восклицательного знака !.

Условный оператор имеет следующий синтаксис: **if(условие) код,** выполняемый, если условие истинно.

Если в случае истинности условия необходимо выполнить несколько операторов, то они объединяются в один блок с помощью фигурных скобок: **if(условие){код,** выполняемый, если условие истинно}.

Полная форма условного оператора включает блок **else**, код в котором выполняется, если условие ложно:

```
if(условие) {

    код, выполняемый, если условие истинно

}

else {

    код, выполняемый, если условие ложно

}
```

Оператор switch позволяет осуществить выбор одной из веток кода в зависимости от значения числовой переменной, например:

```
switch(mode) {

    default:

    case 0:

        код для режима 0
```

```
        break;
    case 1:
        код для режима 1
        break;
    case 2:
        код для режима 2
        break;
}
```

При этом значение переменной **mode** определяет, в какую точку перейдет исполнитель программы внутри тела оператора. Дальше будут выполняться все действия, если не встретится **break**, который прервет дальнейшее исполнение кода внутри **switch.**

Если значение выражения не соответствует ни одной из меток, то выполняется код, начиная с метки **default**. В данном примере из-за отсутствия **break** после **default** управление попадет в «действия для режима 0».Цикл **while** позволяет выполнять код до тех пор, пока условие верно. Он имеет синтаксис:

```
while( условие )
    код, выполняемый в цикле
```

Более сложным является цикл for:

```
for(инициализация; условие; шаг)
    код, выполняемый в цикле
```

В этом цикле выражение, указанное в блоке «инициализация», выполняется один раз перед началом цикла. Обычно в нем задают начальные значения для переменных цикла. Условие проверяется каждый раз в самом начале очередной итерации цикла и если оно станет ложно, то цикл останавливается. «Шаг» выполняется после выполнения тела цикла и обычно увеличивает (или уменьшает) переменные цикла. В блоках «инициализация» и «шаг» можно изменить несколько переменных, если написать соответствующие выражения через запятую. Пример цикла, вычисляющего сумму арифметической прогрессии:

```
int i, sum;
for(i = 0, sum = 0; i <= 10; i++)
    sum += i;
```

Код программы на языке Си должен быть обязательно разбит на функции. Для объявления функции используется следующий синтаксис:

```
тип_функции имя_функции( параметры ) {
```

```
    код, выполняемый в рамках функции

    return результат_функции;

}
```

Тип функции сообщает тип значения, которое будет возвращать функция. Например, стандартная функция cos, вычисляющая косинус, возвращает значение типа float. Имя функции может быть любой строкой, начинающейся с буквы, и содержащей только буквы, цифры и символы подчеркивания.

Параметры – это перечень переменных, которые принимают значения, передаваемые в функцию. Переменные указываются через запятую, сначала пишут тип параметра, а затем – его название.

Чтобы вернуть значение из функции используется оператор return. Он немедленно прерывает выполнение функции, а ее результатом становится значение указанного после return выражения. Если функция ничего не возвращает, то тип_функции указывается как void. Если функции не передается никаких аргументов, то можно ничего не писать внутри круглых скобок или можно написать там слово void. Ниже приведен пример функции, суммирующей два числа типа int:

```
int sum(int a, int b) {

    int result;

    result = a + b;

    return result;

}
```

Для вызова функции необходимо написать ее имя и в круглых скобках указать через запятую значения аргументов, например:

```
r = sum(3, 10);
```

В программах на языке Си можно объявить переменные за пределами любых функций. Такие переменные называются глобальными. Их можно использовать в любых функциях, расположенных после объявления переменной. Например, в этой программе переменная **led** используется в функциях **setup()** и **loop().**

```
const int led = 2;

void setup() {

    pinMode(led, OUTPUT);

}

void loop() {

    digitalWrite(led, HIGH);

    delay(1000);
```

```
        digitalWrite(led, LOW);
        delay(1000);
}
```

Вопросы для самоконтроля

1.Какие из следующих строк не соответствуют синтаксису языка Си:

a) int x;

b) int x, char u

c) int x, y;

d) float int x;

2.Какое значение примет переменная b после выполнения этого кода:

```
int a = 2;
char b = ‘ ‘;
switch(a) {
        default:
        case 0:
                b = ‘a’;
                break;
        case 1:
                b = ‘b’;
                break;
        case 2:
                b = ‘c’;
                break;
}
```

3.Чему будет равно значение переменной k в функции setup() в результате выполнения следующего блока кода:

```
int func (int b){
int a = 0;
if (b % 2 == 0 || a > 0)
```

```
    return a;
else
    return (a+1);
}
void setup(){
    int k = func(5);
}
```

4.Сколько раз выполнится тело следующего цикла:

```
int i = 4;
while(i >= 0) {
        i--;
}
```

1.4 Структура скетча

В предыдущем разделе мы говорили о языке программирования, который используется в Arduino. Этот язык очень похож на язык Си, но есть и некоторые отличия. Во-первых, когда пишут программу для Arduino, она обязательно должна быть написана в одном файле, который называют скетч. Вы можете часто встретить такое сочетание слов «скетч для Arduino», это значит программа для Arduino.

Есть отличие в структуре скетча от структуры обычной программы на Си, и вызвано оно различием между работой программы на обычном компьютере и во встроенных системах. Когда выполняют программу на обычном компьютере, то она запускается, выполняет какие-то действия, а потом завершается. Если программу пишут на языке Си, то за ее выполнение отвечает функция main().

При запуске программы запускается именно эта функция. Из нее программист уже может вызвать остальные функции программы в том порядке, в котором они должны выполняться. Когда функция main() возвращает управление с помощью return, программа завершается. Но для программы, управляющей микроконтроллером, такое поведение не имеет смысла. Пока на устройство подано питание, микроконтроллер должен управлять устройством, а значит, он должен выполнять свою программу. Эта программа никогда не завершается сама, она перестает работать, только если выключить микроконтроллер.

Поэтому структура программы для Arduino отличается от структуры обычной программы на языке Си. В программе для Arduino не должно быть функции main(), вместо нее должны быть две функции: setup() и loop().

Первая из этих функций, функция setup(), вызывается один раз в самом начале выполнения программы сразу после того, как микроконтроллер включили. Она должна подготовить микроконтроллер и все устройства к работе. Обычно в этой функции настраивают микроконтроллер, например, задают, какие его выводы будут использоваться

как входы, а какие – как выходы и так далее. В этой функции можно вывести на экран приветствие для пользователя (если у устройства есть экран). Или каким-то другим образом приготовить устройство к работе. Может быть, нужно что-то включить, или наоборот, убедиться, что ничего лишнего пока не включено.

Вторая функция, функция loop(), выполняется все время, пока работает наше устройство. Если она завершится, то среда Arduino запустит ее заново. В этой функции как раз и нужно реализовать алгоритм работы устройства. Обычно в ней опрашивают подключенные к микроконтроллеру сенсоры, или сами входы микроконтроллера, анализируют полученную информацию и выдают команды исполнительным устройствам.
Можно считать, что где-то в библиотеке Arduino уже есть функция main(), которая устроена примерно так:

```
void main() {
    setup();
    while(true)
    loop();
}
```

Такая функция там действительно есть, скорее всего она устроена более сложно, она может делать еще какие-то действия по управлению микроконтроллером, но общая структура именно такова. А нам нужно написать две функции setup() и loop().

Как и в обычных программах, мы можем создавать и любые другие свои функции и вызывать их из setup() и loop(). Если вы делаете что-то простое, то можно написать весь код просто внутри setup() и loop(). Так стоит сделать, если весь их код помещается в несколько строчек. Если делать что-то сложное, то стоит разбить код на отдельные функции.

В программе Arduino можно вызывать стандартные функции, которые есть в библиотеке. Для этого не нужно добавлять директив #include. В обычном языке Си это нужно делать даже для стандартных функции. Одна из таковых, которая будет часто использоваться, это функция delay(время), делающая задержку на указанное время. Время задается в миллисекундах, поэтому, чтобы подождать 1 секунду, надо написать:
delay(1000);

Есть еще несколько стандартных функций, которые мы будем использовать, и которые можно сразу писать в программе. Эти функции управляют базовыми возможностями микроконтроллера, и мы их рассмотрим, когда они нам понадобятся.

Большим преимуществом среды Arduino является ее большая популярность. Многие производители выпускают платы расширения, которые можно подключать к Arduino и использовать вместе с ней. Для того чтобы такие платы было удобно использовать, для них пишут библиотеки, содержащие удобные функции для работы с этими устройствами. Пишут такие библиотеки и просто для отдельных компонентов, которые можно подключить к плате Arduino. Мы будем использовать, например, библиотеку для управления жидкокристаллическим экраном. Чтобы использовать библиотеку, нужно в начале программы поместить директиву #include:
#include < имя_библиотеки.h >

Например, библиотека для работы с дисплеем называется LiquidCrystall, и чтобы ее использовать, надо сначала написать:

```
#include <LiquidCrystal.h>
```

В языке Arduino для работы с библиотеками используют элементы языка C++. Фактически, сами библиотеки пишут на C++, но в скетче можно использовать только очень простые возможности этого языка. Язык C++ позволяет создавать классы. Классы – это пользовательские типы данных, описывающие набор полей и операции, которые с ними можно делать. С помощью этого механизма в C++ можно сделать, например, класс «дисплей» и переменную такого типа. Такие переменные называют объектами. Потом можно работать с этими «сложными» переменными. Чтобы сделать такую переменную, ее надо объявить, как объявляют обычные переменные. Например, для работы с дисплеем надо сделать переменную типа LiquidCrystall:

```
LiquidCrystal lcd(4, 5, 6, 7, 8, 9);
```

Тут LiquidCrystall – это название типа, его пишут в документации на библиотеку. lcd – это название нашей переменной, можно использовать любое имя, как удобно. Цифры – это параметры, которые передаются объекту при создании. О том, какие параметры надо указать, тоже пишут в документации. Для дисплея это, на самом деле, номера выводов микроконтроллера, к которым он подключен. Позже мы рассмотрим работу с дисплеями более подробно.

После того, как переменная создана, мы можем с ней работать. Для этого используют специальные функции, которые доступны только для этой переменной. Такие функции еще называют «методами класса». Вызывают их следующим способом:

```
имя_переменной.имя_функции( параметры );
```

Например, текст на дисплей выводит функция print, и ее можно вызвать следующим образом:

```
lcd.print("Hello!");
```

Так как мы указываем сначала имя переменной, то функция print вызывается именно для этой переменной. Можно подключить несколько дисплеев и сделать для каждого из них свою переменную.

1.5 Цифровой ввод-вывод

1.5.1 Цифровые выходы. Работа со светодиодом

Одно из простейших действий, которое может выполнить микроконтроллер, это переключение состояния контактов цифровых входов-выходов. Любой из цифровых выходов микроконтроллера можно программным способом переключить между высоким уровнем (+5 V) и низким (0 V). С помощью такого переключения происходит управление простыми устройствами.

Переключение осуществляется функцией digitalWrite, вызов которой имеет вид:

digitalWrite(номер_контакта, уровень_сигнала). Параметр уровень_сигнала может принимать два значения: HIGH (высокий, +5 V) или LOW (низкий). Параметр номер_контакта – это число, соответствующее номеру контакта на плате Arduino.

Обратите внимание, что одни и те же выводы микроконтроллера могут выступать как в роли цифровых выходов, так и в роли цифровых входов. Поэтому перед их использованием необходимо указать, в какой роли будет использоваться контакт: входа или выхода. Это делается с помощью функции pinMode: pinMode(номер_контакта, режим_контакта). Ее аргумент режим_контакта может принимать значения: **OUTPUT** (выход) и **INPUT** (вход). Таким образом, чтобы установить на выходе №2 высокий уровень сигнала, достаточно выполнить следующую программу:

```
const int pin = 2;

void setup() {

    pinMode(pin, OUTPUT);

    digitalWrite(pin, HIGH);

}

 void loop() {

}
```

Одно из наиболее простых для подключения исполнительных устройств, которым можно управлять с помощью цифрового выхода, это светодиод. Это устройство представляет собой полупроводниковый прибор, способный излучать свет при пропускании через него электрического тока в прямом направлении (от анода к катоду). На рисунке 7 показан внешний вид светодиода и его обозначение на электрической схеме.

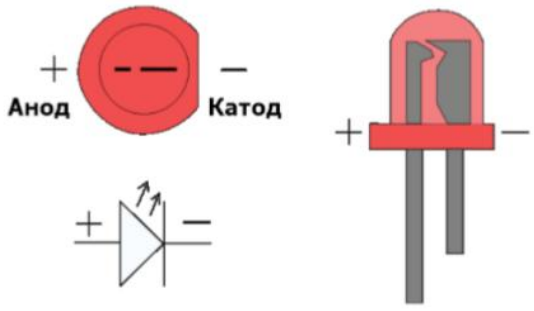


Рисунок 7- Внешний вид светодиода и его обозначение на электрической схеме

Для того чтобы правильно включить светодиод в электрическую цепь, необходимо отличать катод от анода. Сделать это можно по трем признакам:

1) анод светодиода имеет более длинный проводник;

2) со стороны катода корпус светодиода немного срезан;

3) внутри светодиода катод шире, чем анод.

В современной микроэлектронике применяются миниатюрные светодиоды для поверхностного монтажа. Такие индикаторы, например, имеются на Arduino Uno для информирования пользователя о состоянии системы. Важно отметить, что рабочее напряжение питания светодиода варьируется от 1.85 до 2.5 вольт при рекомендуемой силе тока не более 20мА. Чтобы сила тока не превысила это значение, при подключении светодиода к выходу микроконтроллера, который выдает напряжение 5 V, в цепь следует добавить последовательный ограничивающий резистор сопротивлением от 200 до 500 Ом.

В противном случае ток через светодиод будет слишком большим, что может повредить как диод, так и вывод микроконтроллера. На Рис. 6.2 представлена электрическая схема подключения светодиода к Arduino Uno, а также модель собранного устройства. Резисторы, в особенности малой мощности — мелкие детали, резистор мощностью 0,125 Вт имеет длину несколько миллиметров и диаметр порядка миллиметра. Прочитать на такой детали номинал с десятичной запятой трудно, поэтому при указании номинала вместо десятичной точки пишут букву, соответствующую единицам измерения (К — для килоомов; М — для мегаомов; Е, R или без указания единиц — для единиц Ом). Кроме того, любой номинал отображается максимум тремя символами. Например, 4K7 обозначает резистор сопротивлением 4,7 кОм, 1R0 — 1 Ом, M12 — 120 кОм (0,12 МОм) и т. д. Однако в таком виде наносить номиналы на маленькие резисторы сложно, и для них применяют маркировку цветными полосами.

Для резисторов с точностью 20 % используют маркировку с тремя полосками, для резисторов с точностью 10 % и 5 % — маркировку с четырьмя полосками, для более точных резисторов — с пятью или шестью полосками. Первые две полоски всегда означают первые два знака номинала. Если полосок 3 или 4, третья полоска означает десятичный множитель, то есть степень десятки, которая умножается на число, состоящее из двух цифр, указанное первыми двумя полосками. Если полосок 4, последняя указывает точность резистора. Если полосок 5, третья означает третий знак сопротивления, четвёртая — десятичный множитель, пятая — точность. Шестая полоска, если она есть, указывает температурный коэффициент сопротивления (ТКС). Если эта полоска в 1,5 раза шире остальных, то она указывает надёжность резистора (% отказов на 1000 часов работы), как показано на рисунке 4.

Следует отметить, что иногда встречаются резисторы с 5 полосами, но стандартной (5 или 10 %) точностью. В этом случае первые две полосы задают первые знаки номинала, третья — множитель, четвёртая — точность, а пятая — температурный коэффициент.

Пример определения номинала резистора: допустим, на резисторе имеются четыре полосы: коричневая, чёрная, красная и золотая. Первые две полоски дают 1 0, третья 100, четвёртая даёт точность 5 %, итого — резистор сопротивлением 10·100 Ом = 1 кОм, с точностью ±5 %, как представлено на рисунке 8.

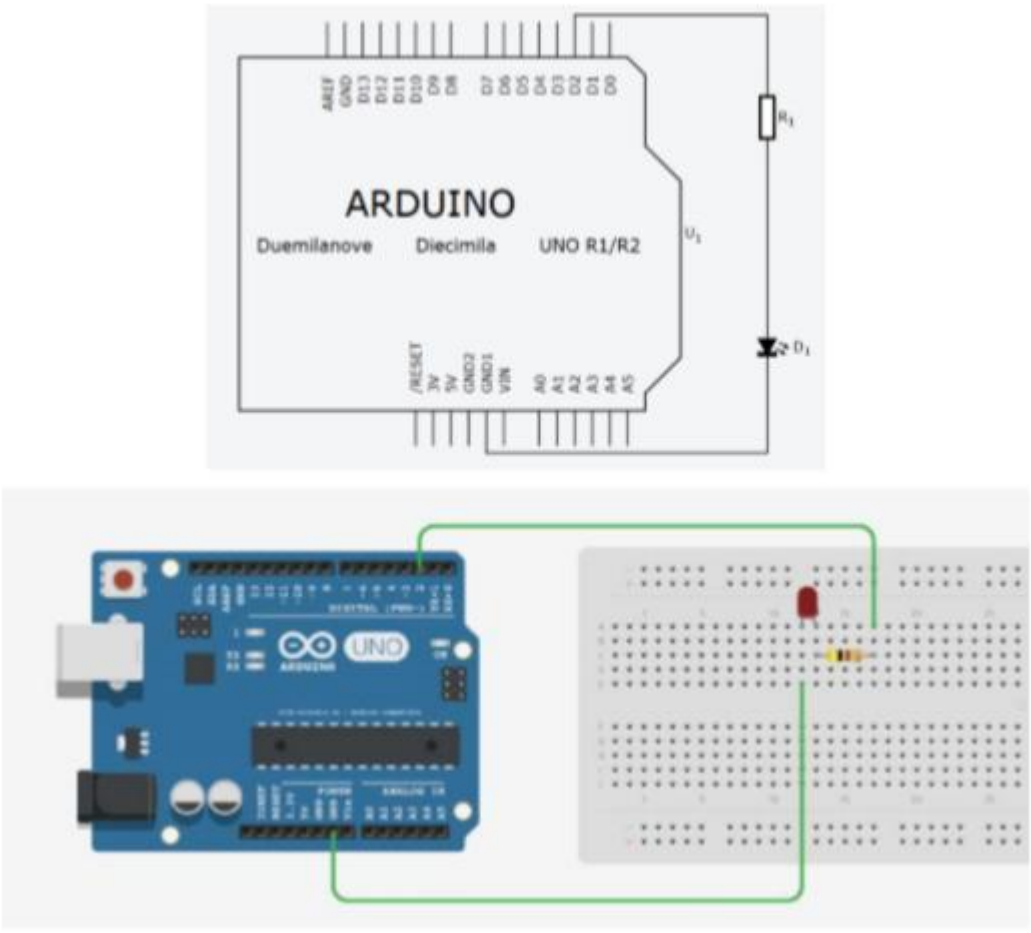


Рисунок 8- Электрическая схема подключения светодиода и модель собранного устройства

Попробуем «помигать» светодиодом. Для этого мы будем последовательно зажигать его, передавая на ногу №2 сигнал HIGH, а затем гасить с помощью сигнала LOW. Между включением и выключением светодиода обязательно нужно поставить задержку в несколько сотен миллисекунд, иначе мы не заметим мигания. Вспомним, что контроллер Arduino работает на частоте 16MHz, поэтому он может включать и выключать светодиод тысячи раз в секунду. Получаем следующую программу:

```
int led = 2;

void setup() {

        pinMode(led, OUTPUT);

}

void loop() {

        digitalWrite(led, HIGH);

        delay(1000);

        digitalWrite(led, LOW);

        delay(1000);
```

}

Аналогичным образом можно подключить еще один светодиод, как показано на рисунке 9. Для подключения сразу двух светодиодов к земле используем шину питания на макетной плате.

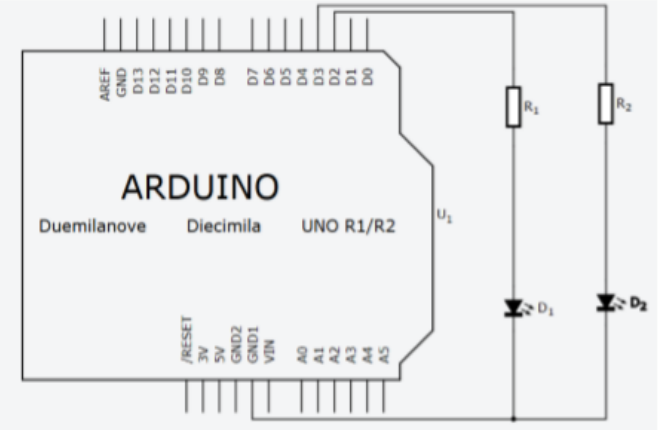


Рисунок 9- Схема подключения двух светодиодов

Имеем следующий код:

```
int led_r = 2;

int led_g = 3;

void setup() {

        pinMode(led_r, OUTPUT);

        pinMode(led_g, OUTPUT);

}

void loop() {

        digitalWrite(led_r, HIGH);

        digitalWrite(led_g, HIGH);

        delay(1000);

        digitalWrite(led_r, LOW);

        digitalWrite(led_g, LOW);

        delay(1000); }
```

1.5.2 Вывод информации через последовательный порт

Arduino может передавать текстовые сообщения на персональный компьютер. Для того чтобы этим воспользоваться, необходимо в функции setup инициализировать последовательный порт устройства:

Serial.begin(9600);

Значение 9600 означает скорость передачи данных в битах в секунду. 9600 – это одна из стандартных скоростей последовательного порта. Непосредственно для вывода короткого сообщения используется функция print:

Serial.print(“text”);

Пример программы:

```
void setup() {
        Serial.begin(9600);
        Serial.print("Hello!");
}
void loop() {
}
```

Чтобы увидеть эти сообщения в Autodesk CIRCUITS включите Serial Monitor на панели управления над редактором кода. Если вы используете настоящую плату Arduino и редактор Arduino IDE, используйте имеющееся в нем окно «Монитор порта».

1.5.3 Самостоятельная работа. Напишите программу, которая будет каждую секунду передавать через последовательный порт увеличивающееся на единицу число.

1.5.4 Подключение RGB- светодиода к Ардуино UNO

Рассмотрим использование цифровых и аналоговых выходов с «широтно импульсной модуляцией» на плате Arduino UNO для включения RGB-светодиода с различными оттенками. Расскажем про устройство и распиновку полноцветного RGB-светодиода и рассмотрим директиву **#define** в языке программирования C++.

Для отображения всей палитры оттенков вполне достаточно три цвета, используя RGB синтез (Red — красный, Green — зеленый, Blue — синий). RGB палитра используется не только в графических редакторах. Смешивая красный, зеленый и синий цвет в разной пропорции можно получить практически любой цвет.

RGB-светодиоды объединяют три кристалла разных цветов в одном корпусе. Использование RGB-светодиодов и RGB LED ленты позволяет создать осветительный прибор или освещение интерьера с любым оттенком цвета. Преимущества RGB-светодиодов в простоте конструкции и высоком КПД светоотдачи.RGB LED имеет 4 вывода — один общий (анод или катод имеет самый длинный вывод) и три цветовых вывода. К каждому цветовому выходу следует подключать резистор.

Рисунок 10- Распиновка RGB светодиода и модуль с RGB светодиодом для Ардуино

Распиновка RGB-светодиода указана на рисунке 10. Заметим также, что для многих полноцветных светодиодов необходимы светорассеиватели, иначе будут видны составляющие цвета. Далее подключим RGB-светодиод к Ардуино и заставим его светится всеми цветами радуги с помощью «широтно импульсной модуляции».

1.5.5 Управление RGB- светодиодом на Ардуино

Аналоговые выходы на Ардуино используют «широтно-импульсную модуляцию» для получения различной силы тока. Мы можем подавать на все три цветовых входа на светодиоде различное значение ШИМ-сигнала в диапазоне от 0 до 255, что позволит нам получить на RGB LED Arduino практически любой оттенок света.Для проведения работ с RGB-светодиодом нам понадобятся следующие детали:

- плата Arduino Uno;
- макетная плата;
- USB-кабель;
- RGB светодиод;
- 3 резистора 220 Ом;
- провода «папка-мамка».

Модуль «RGB-светодиод» можно подключить напрямую к плате, без проводов и макетной платы. Подключите модуль с полноцветным RGB светодиодом к следующим пинам: **Минус** — GND, **B** — Pin13, **G** — Pin12, **R** — Pin11 (как указано на рисунке 23). Если вы используете RGB LED (Light Emitting Diode), то подключите его по схеме, как показано на рисунке 11.

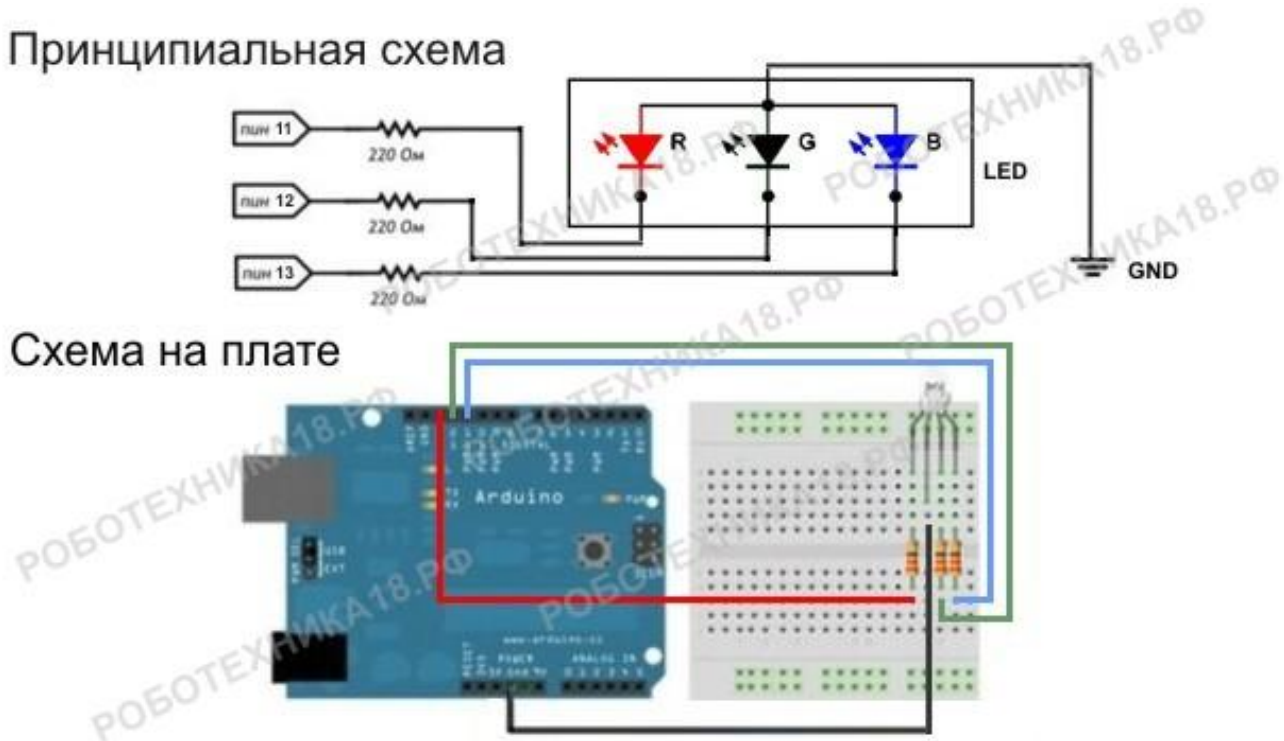


Рисунок 11- Схема подключения RGB- LED к Ардуино на макетной плате

После подключения модуля или сборки схемы загрузите следующий скетч:

готовый скетч для примера:

```
#define RED 11 // Присваиваем имя RED для пина 11
#define GREEN 12 // Присваиваем имя GREEN для пина 12
#define BLUE 13 // Присваиваем имя BLUE для пина 13

void setup()
{
 pinMode(RED, OUTPUT); // Используем Pin11 для вывода
 pinMode(GREEN, OUTPUT); // Используем Pin12 для вывода
 pinMode(BLUE, OUTPUT); // Используем Pin13 для вывода
}

void loop()
{

 digitalWrite(RED, HIGH); // Включаем красный свет
 digitalWrite(GREEN, LOW);
 digitalWrite(BLUE, LOW);

 delay(1000); // Устанавливаем паузу для эффекта

 digitalWrite(RED, LOW);
 digitalWrite(GREEN, HIGH); // Включаем зеленый свет
 digitalWrite(BLUE, LOW);

 delay(1000); // Устанавливаем паузу для эффекта

 digitalWrite(RED, LOW);
 digitalWrite(GREEN, LOW);
 digitalWrite(BLUE, HIGH); // Включаем синий свет
```

```
 delay(1000); // Устанавливаем паузу для эффекта
}
```

1.5.6 Пояснения к коду:

1. С помощью директивы #define мы заменили номер пинов 11, 12 и 13 на соответствующие имена RED, GREEN и BLUE. Это сделано для удобства, чтобы не запутаться в скетче и понимать какой цвет мы включаем.
2. В процедуре void setup() мы назначили пины 11, 12 и 13, как выходы.
3. В процедуре void loop() мы поочередно включаем все три цвета на RGB LED.

На что обратить внимание:

- пины 11, 12 и 13 используются, как цифровые выходы digitalWrite.

1.5.7 Плавное управление включением RGB-светодиодом

Управление RGB-светодиодом на Arduino можно сделать плавным, используя аналоговые выходы с «широтно импульсной модуляцией». Для этого цветовые входы на светодиоде необходимо подключить к аналоговым выходам, например, к пинам 11, 10 и 9. И подавать на них различные значения ШИМ (PWM) для различных оттенков.После подключения модуля с помощью проводов «папа-мама» загрузите скетч:

Готовый скетч для примера

```
#define RED 9 // Присваиваем имя RED для пина 9
#define GREEN 10 // Присваиваем имя GREEN для пина 10
#define BLUE 11 // Присваиваем имя BLUE для пина 11
 void setup()
{
  pinMode(RED, OUTPUT); // Используем Pin9 для вывода
  pinMode(GREEN, OUTPUT); // Используем Pin10 для вывода
  pinMode(BLUE, OUTPUT); // Используем Pin11 для вывода
}
void loop()
{
  analogWrite(RED, 50); // Включаем красный свет
  analogWrite(GREEN, 250); // Включаем зеленый свет
  analogWrite(BLUE, 150); // Включаем синий свет
}
```

1.5.8 Пояснения к коду

1. С помощью директивы #define мы заменили номер пинов 9, 10 и 11 на соответствующие имена RED, GREEN и BLUE. Это сделано для удобства, чтобы не запутаться в скетче и понимать какой цвет мы включаем.
2. В процедуре void setup() мы назначили пины 9, 10 и 11, как выходы.
3. В процедуре void loop() мы включаем все три цвета на RGB светодиоде.

На что обратить внимание:

1. Пины 11, 12 и 13 используются, как аналоговые выходы analogWrite.

1.5.9 Самостоятельная работа. Напишите скетч, чтобы на полноцветном RGB-светодиоде включались различные цвета.

1.5.10 Управление LED-светодиодом, установленным на плате Arduino UNO.

Для выдержки паузы между вкл/выкл LED в 1 сек. может использована функция **delay()**. В это время контроллер не может выполнять другие команды в главной функции **loop()**

```
/* Мигание LED
 * ------------
 *
 * Включает и выключает светодиод (LED) подсоединенный
 * к выходу 13, с интервалом в 2 секунды
 *
 */

int ledPin = 13;                // LED подсоединен к выводу 13
void setup()
{
  pinMode(ledPin, OUTPUT);      // устанавливаем вывод 13 как выход
}
void loop()
{
  digitalWrite(ledPin, HIGH);   // включаем LED
  delay(1000);                  // пауза 1 секунда
  digitalWrite(ledPin, LOW);    // выключаем LED
  delay(1000);                  // пауза 1 секунда
}
```

Внесем корректировки в функцию **loop()**. Чтобы сделать код более компактным и изящным, заменим 2 пары строчек на одну пару. Вместо установки значения в HIGH, а затем обратно в LOW, мы получим текущее значение **ledPin** и проинвертируем его. Т.е. если оно было HIGH, то станет LOW и наоборот.

```
void loop()
{
  digitalWrite(ledPin, !digitalRead(ledPin));   // включаем/выключаем LED
  delay(1000);                  // задержка 1 сек.
}
```

Теперь мы усовершенствуем функцию **delay()**. Взамен, мы будем использовать функцию **millis()**. Данная функция возвращает количество миллисекунд, прошедшее с момента запуска текущей программы. Функция переполнится (вернется в нуль) приблизительно через 50 суток работы программы.

Альтернативной функцией является **micros()**, которая возвращает количество микросекунд, прошедшее с момента запуска текущей программы. Функция переполнится (вернется в нуль) приблизительно через 70 минут работы программы.

В нашем примере мы будем использовать функцию **millis()**:

```
/* Мигание LED Версия 2
 * ----------------------
 * Включает и выключает светодиод (LED) подсоединенный
 * к выходу 13, с интервалом в 2 секунды используя функцию millis()
 *
 */
int ledPin = 13;                // LED подсоединен к выводу 13
unsigned long currentTime;
unsigned long loopTime;

void setup()
{
  pinMode(ledPin, OUTPUT);      // устанавливаем вывод 13 как выход
  currentTime = millis();                       // считываем время, прошедшее с момента запуска программы
  loopTime = currentTime;
}

void loop()
{
  currentTime = millis();                                                                    // считываем
время, прошедшее с момента запуска программы
  if(currentTime >= (loopTime + 1000)){                                                     // сравниваем текущий таймер с
переменной loopTime + 1 секунда
    digitalWrite(ledPin, !digitalRead(ledPin));             // включаем/выключаем LED
    loopTime = currentTime;                                                                 //       в       loopTime
записываем новое значение
  }
  // Здесь могут быть другие команды
}
```

В данном примере мы ввели две дополнительные переменные currentTime и loopTime. В функции **setup()** обе переменные имеют одно и тоже значение. В функции **loop()**, переменная currentTime каждый раз обновляется в цикле. Когда currentTime больше чем loopTime на 1 секунду (loopTime + 1000), то LED меняет свое состояние, а переменной loopTime присваивается текущее значение currentTime.Обратите внимание, что в данном примере мы не использовали функцию **delay()** и процессор может выполнять другие операции.

1.6 Аналоговые входы платы Ардуино

Плата Arduino UNO содержит 6 аналоговых входов предназначенных для измерения напряжения сигналов. Правильнее сказать, что 6 выводов платы могут работать в режиме, как дискретных выводов, так и аналоговых входов.Эти выводы имеют номера от 14 до 19. Изначально они настроены как аналоговые входы, и обращение к ним можно производить через имена A0-A5. В любой момент их можно настроить на режим дискретных выходов.

```
pinMode(A3, OUTPUT); // установка режима дискретного вывода для A3
digitalWrite(A3, LOW);  // установка низкого состояния на выходе A3
```

Чтобы вернуть в режим аналогового входа:

```
pinMode(A3, INPUT);  // установка режима аналогового входа для A3
```

1.6.1 Аналоговые входы и подтягивающие резисторы

К выводам аналоговых входов, так же как и к дискретным выводам, подключены подтягивающие резисторы. Включение этих резисторов производится командой:

digitalWrite(A3, HIGH); // включить подтягивающий резистор к входу A3

Команду необходимо применять к выводу настроенному в режиме входа.Надо помнить, что резистор может оказать влияние на уровень входного аналогового сигнала. Ток от источника питания 5 В, через подтягивающий резистор, вызовет падение напряжения на внутреннем сопротивлении источника сигнала. Так что лучше резистор отключать.

1.6.2 Аналого-цифровой преобразователь платы Ардуино

Собственно измерение напряжение на входах производится аналого-цифровым преобразователем (АЦП) с коммутатором на 6 каналов. АЦП имеет разрешение 10 бит, что соответствует коду на выходе преобразователя 0…1023. Погрешность измерения не более 2 единиц младшего разряда.Для сохранения максимальной точности (10 разрядов) необходимо, чтобы внутреннее сопротивление источника сигнала не превышало 10 кОм. Это требование особенно важно при использовании резисторных делителей, подключенных к аналоговым входам платы. Сопротивление резисторов делителей не может быть слишком большим.

1.6.3 Программные функции аналогового ввода(int analogRead(port))

Считывает значение напряжения на указанном аналоговом входе. Входное напряжение диапазона от 0 до уровня источника опорного напряжения (часто 5 В) преобразовывает в код от 0 до 1023.При опорном напряжении равном 5 В разрешающая способность составляет 5 В / 1024 = 4,88 мВ. Занимает на преобразование время примерно 100 мкс.

```
int inputCod;          // код входного напряжения
float inputVoltage; // входное напряжение в В
inputCod= analogRead(A3); // чтение напряжения на входе A3
inputVoltage= ( (float)inputCod * 5. / 1024. ); // пересчет кода в напряжение (В)
```

1.6.4 Опорное напряжение для АЦП

Void analogReference(type) задает опорное напряжение для АЦП. Оно определяет максимальное значение напряжения на аналоговом входе, которое АЦП может корректно преобразовать. Величина опорного напряжения также определяет коэффициент пересчета кода в напряжение:
Напряжение на входе = код АЦП * опорное напряжение / 1024. Аргумент type может принимать следующие значения:

- DEFAULT – опорное напряжение равно напряжению питания контроллера (5 В или 3,3 В). Для Arduino UNO R3 – 5 В.
- INTERNAL – внутреннее опорное напряжение 1,1 В для плат с контроллерами ATmega168 и ATmega328, для ATmega8 – 2,56 В.
- INTERNAL1V1 – внутреннее опорное напряжение 1,1 В для контроллеров Arduino Mega.
- INTERNAL2V56 – внутреннее опорное напряжение 2,56 В для контроллеров Arduino Mega.

- EXTERNAL – внешний источник опорного напряжения, подключается к входу AREF.

analogReference(INTERNAL); // опорное напряжение равно 1,1 В

Рекомендуется внешний источник опорного напряжения подключать через токоограничительный резистор 5 кОм.

1.6.5 Подключение кнопки

Кнопка – это прибор, который позволяет замыкать и размыкать электрическую цепь. Смена состояния выключателя может происходить разными способами, в зависимости от типа устройства.Механические выключатели используются непосредственно человеком. К такому типу относятся различные тумблеры, кнопки, клавиши, как представлено на рисунке 12.

Рисунок 12-Внешний вид кнопки

Электромагнитные и электронные, напротив, применяются в автоматических системах и управляются при помощи электрических сигналов. Самым известным электромагнитным выключателем является реле (relay). Примером электронного выключателя может служить транзистор (transistor). В лабораторной работе мы будем использовать простой механический выключатель (pushbutton), который замыкает два своих контакта при нажатии. Для удобства монтажа их часто делают с 4-мя контактами, у которых выводы соединены попарно. Для чтения цифрового сигнала с одного из контактов Arduino, необходимо воспользоваться функцией digitalRead:
результат = digitalRead(номер_контакта)/

После вызова этой функции переменная результат будет хранить уровень цифрового сигнала, детектируемый на соответствующем контакте. Чтобы читать состояние кнопки, ее нужно подключить таким образом, чтобы при ее нажатии на входе был высокий уровень напряжения (5 V), а при отпускании – низкий (0 V).

1.6.5.1 Подключение кнопки с подтягивающим резистором

Приведем схему подключения кнопки с резистором:

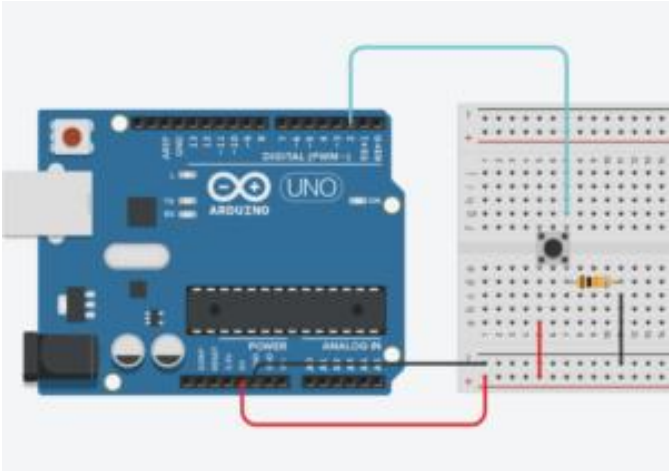


Рисунок 13- Схема подключения кнопки

Для этого подключим последовательно между шиной питания и землей кнопку и резистор, а вход микроконтроллера подключим посередине между ними, как показано на рисунке 13. Тогда, если кнопка не нажата, то выход микроконтроллера будет соединен через резистор с землей и будет иметь низкий уровень, а если кнопку нажать, то получится, что вход соединен с питанием и на нем будет высокий уровень. При такой схеме подключения через резистор потечет ток, который зависит от его сопротивления. Чтобы схема не потребляла много лишнего электричества, возьмем резистор побольше, например, 10 kOm. В скетче мы будем отслеживать факт нажатия и выводить сообщение в монитор порта. Скетч для кнопки Ардуино с подтягивающим резистором:

```
/*
Пример использования тактовой кнопки в ардуино.
Кнопка подключена к пину 2.
*/

const int PIN_BUTTON = 2;

void setup() {
Serial.begin(9600);
pinMode(PIN_LED, OUTPUT);
}

void loop() {
// Получаем состояние кнопки и выводим в мониторе порта
int buttonState = digitalRead(PIN_BUTTON);
Serial.println(buttonState);
delay(50);
}
```

1.6.5.3 Дребезг кнопки Ардуино

Работа с тактовыми кнопками на Ардуино

Главная проблема использования кнопок для управления Arduino заключается в «дребезге контактов». Дело в том, что механические контакты в тактовых кнопках никогда не замыкаются и размыкаются мгновенно. В течении нескольких миллисекунд происходит многократное замыкание и размыкание контактов — в итоге на

микроконтроллер поступает не единичный сигнал, а серия импульсов, как представлено на рисунке 14.

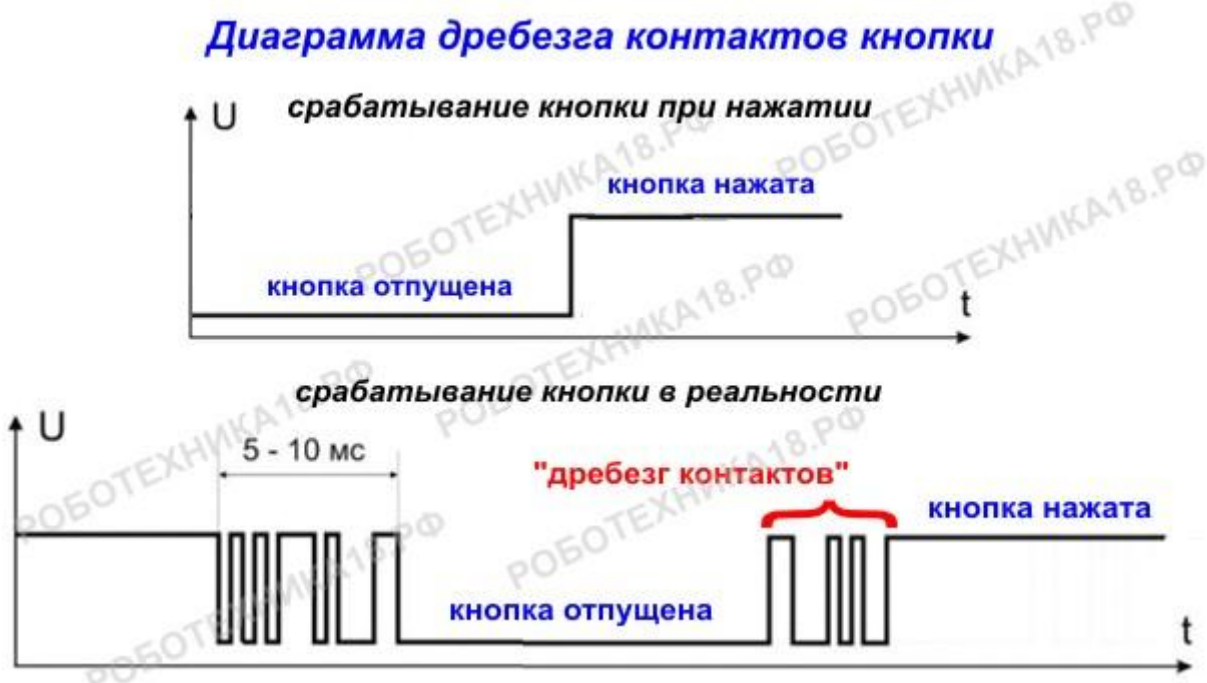


Рисунок 14-Подключение кнопки к Ардуино и дребезг

Для того, чтобы исключить на микроконтроллере Arduino дребезг кнопки используют различные электрические схемы с триггерами и конденсаторами. Но намного удобнее и проще использовать программный способ борьбы с возможным дребезгом тактовой кнопки — для этого применяют задержку на несколько миллисекунд или используют библиотеку Bounce2.h для борьбы с дребезгом контактов для Arduino. В процессе работы с кнопками мы можем столкнуться с очень неприятным явлением, называемым дребезгом кнопки. Как следует из самого названия, явление это обуславливается дребезгом контактов внутри кнопочного переключателя. Металлические пластины соприкасаются друг с другом не мгновенно (хоть и очень быстро для наших глаз), поэтому на короткое время в зоне контакта возникают скачки и провалы напряжения. Для устранения дребезга используют программные и аппаратные решения. В двух словах лишь упомянем основные методы подавления дребезга:

- добавляем в скетче паузу 10-50 миллисекунд между полкучением значений с пина ардуино;
- если мы используем прерывания, то программный метд использоваться не может и мы формируем аппаратную защиту. Простейшая из них – RC фильтр с конденсатором и сопротивлением;
- для более точного подавления дребезга используется аппаратный фильтр с использованием триггера шмидта. Этот вариант позволит получить на входе в ардуино сигнал практически идеальной формы.

Выполним следующий проект-будем переключать состояние светодиода (включен/выключен) при каждом нажатии кнопки. Разработаем скетч:

```
const int LED=10; // Контакт 10 для подключения светодиода
```

```
const int BUTTON=2; // Контакт 2 для подключения кнопки
int tekButton = LOW; // Переменная для сохранения текущего состояния кнопки
int prevButton = LOW; // Переменная для сохранения предыдущего состояния
// к нопки
boolean ledOn = false; // Текущее состояние светодиода (включен/выключен)
void setup()
{
// Сконфигурировать контакт светодиода как выход
pinMode (LED, OUTPUT);
// Сконфигурировать контакт кнопки как вход
pinMode (BUTTON, INPUT);
}
void loop()
{
tekButton=digitalRead(BUTTON);
if (tekButton == HIGH && prevButton == LOW)
{
// нажатие кнопки – изменить состояние светодиода
ledOn=!ledOn;
digitalWrite(LED, ledOn);
}
prevButton=tekButton;
}
```

При нажатии кнопки светодиод должен изменять свое состояние. Но это будет происходить не всегда. Виной тому – дребезг кнопок.Кнопки представляют из себя механические устройства с системой пружинного контакта. Когда вы нажимаете на кнопку вниз, сигнал не просто меняется от низкого до высокого, он в течение нескольких миллисекунд меняет значение от одного до другого, прежде чем контакты плотно соприкоснутся и установится значение HIGH.Микроконтроллер зафиксирует все эти нажатия, потому что дребезг неотличим от настоящего нажатия на кнопку. Устранить влияние дребезга можно программно. Алгоритм следующий:

1. Сохраняем предыдущее состояние кнопки и текущее состояние кнопки (при инициализации LOW).
2. Считываем текущее состояние кнопки.
3. Если текущее состояние кнопки отличается от предыдущего состояния кнопки, ждем 5 мс, потому что кнопка, возможно, изменила состояние.
4. После 5 мс считываем состояние кнопки и используем его в качестве текущего

состояния кнопки.
5. Если предыдущее состояние кнопки было LOW, а текущее состояние кнопки HIGH, переключаем состояние светодиода.
6. Устанавливаем предыдущее состояние кнопки для текущего состояния кнопки.
7. Возврат к шагу 2. Добавляем к нашему скетчу подпрограмму устранения дребезга. Получаем код, показанный ниже.

```
const int LED=10; // Контакт 10 для подключения светодиода

const int BUTTON=2; // Контакт 2 для подключения кнопки

int tekButton = LOW; // Переменная для сохранения текущего состояния кнопки

int prevButton = LOW; // Переменная для сохранения предыдущего состояния

// к нопки

boolean ledOn = false; // Текущее состояние светодиода (включен/выключен)

void setup()

{

// Сконфигурировать контакт светодиода как выход

pinMode (LED, OUTPUT);

// Сконфигурировать контакт кнопки как вход

pinMode (BUTTON, INPUT);

}

// Функция сглаживания дребезга. Принимает в качестве

// аргумента предыдущее состояние кнопки и выдает фактическое.

boolean debounce(boolean last)

{

boolean current = digitalRead(BUTTON); // Считать состояние кнопки,

if (last != current) // если изменилось...

{

d elay(5); // ж дем 5 м с

current = digitalRead(BUTTON); // считываем состояние кнопки

return current; // возвращаем состояние кнопки

}

}

void loop()

{

tekButton = debounce(prevButton);

if (prevButton == LOW && tekButton == HIGH) // если нажатие...
```

```
{
ledOn = !ledOn; // инвертировать значение состояния светодиода
}
prevButton = tekButton;
digitalWrite(LED, ledOn); // изменить статус состояния светодиода
}
```

Загружаем скетч в плату Arduino и проверяем работу. Теперь все работает нормально, каждое нажатие кнопки приводит к изменению состояния светодиода.

1.6.6 Аналоговый датчик температуры

Распространенным вариантом датчиков с аналоговым выходом являются датчики температуры. Мы будем использовать датчик TMP36, который изображен на рисунке 76.

Назначение его контактов следующее:

– Vs – питание 2.7 - 5.5В (в нашем случае +5В);

– Vout – выходное напряжение, зависящее от температуры;

– Gnd – земля.

Пример подключения датчика изображен на рисунке 15.

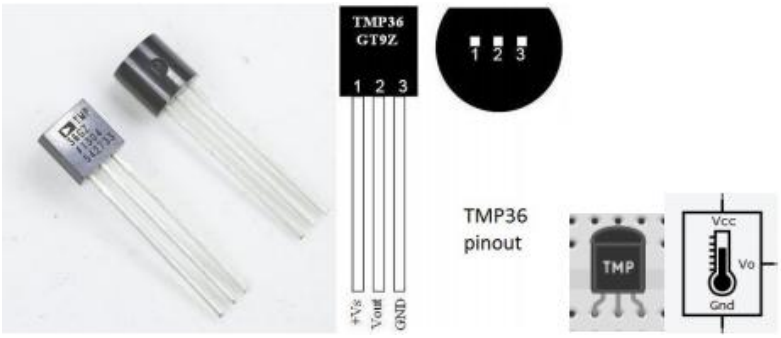


Рисунок 15- Аналоговый датчик температуры TMP36

Выходное напряжение сенсора зависит только от температуры и не зависит от напряжения питания. Для того чтобы рассчитать температуру, нужно воспользоваться формулой:

$$T = \frac{V_{out} - 500}{10},$$

где V_{out} – выходное напряжение датчика в милливольтах, $\boldsymbol{T}$ – температура в градусах Цельсия. Учитывая, что напряжению 5 V соответствует значение 1023, получаем следующую формулу:

$$T = \frac{Reading * \frac{5000}{1023} - 500}{10},$$

где Reading – значение, полученное в результате вызова функции analogRead.

Пример подключения аналогового датчика температуры приведен на рисунке 16.

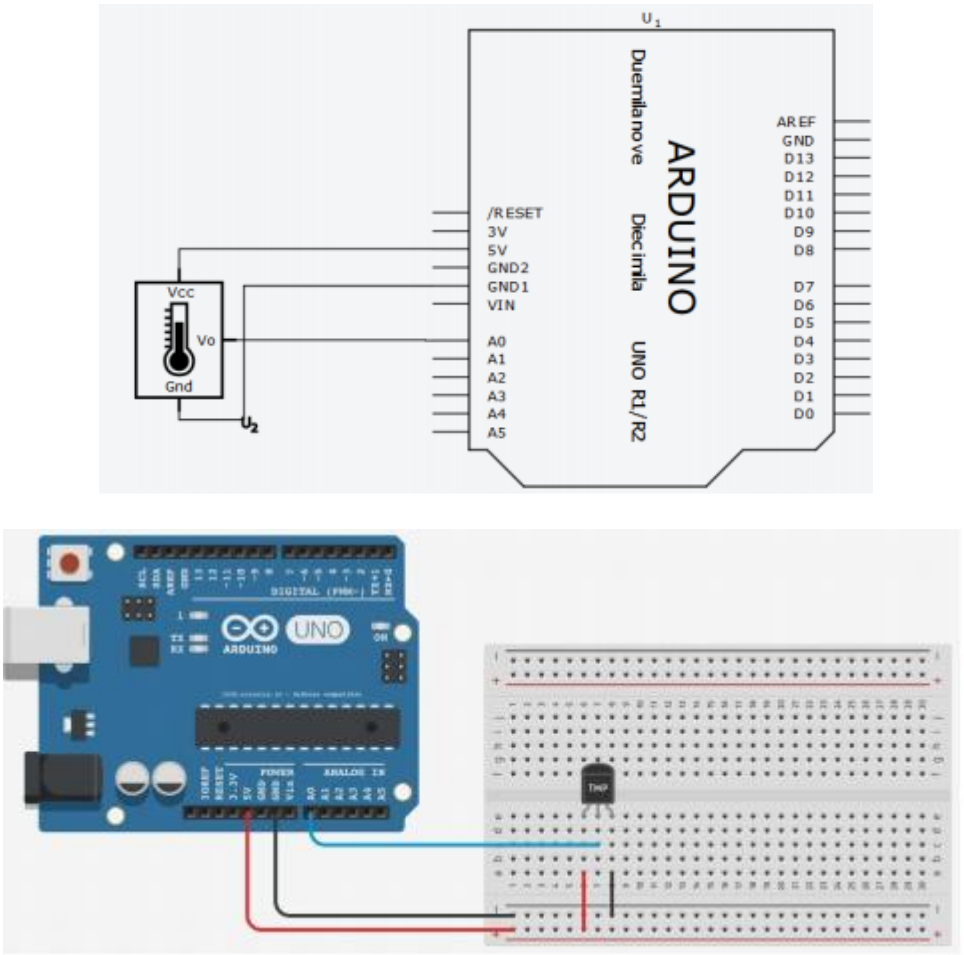


Рисунок 16- Пример подключения аналогового датчика температуры

1.6.7 Датчик температуры DTH11 Ардуино

В работе мы будем использовать датчик DHT11, смонтированный на плате. DHT11 — это цифровой датчик, состоящий из термистора и емкостного датчика влажности. Наряду с невысокой стоимостью DHT11 имеет следующие характеристики: питание осуществляется от 3,5-5V, определение температуры от 0 до 50 градусов с точностью 2 град, определение влажности от 20% до 95% с 5% точностью, как представлено на рисунке 17.

Рисунок 17-Устройство датчика температуры и влажности (dht11) для Ардуино

Модуль DHT11 оборудован трех пиновым разъемом и подключается по схеме:

G — подключается к выводу GND;
V — подключается к выводу +5V;
S — подключается к цифровому выводу (Pin2).

Подключение датчика температуры и влажности DHT11 как представлено на рисунке 18.

Термистор — это термический резистор, сопротивление которого изменяется с температурой, т.е. увеличение температуры приводит к падению его сопротивления. По сути термистор — это термометр сопротивления, изготовленный на основе смешанных оксидов переходных металлов. Относится к измерительной технике и может быть использован для автоматического измерения температуры в различных средах.

Емкостной датчик влажности — это конденсатор с переменной емкостью, который содержит токопроводящие обкладки из медной фольги на текстолите. Этот конденсатор заключен в герметичный чехол, поверх которого расположен влагопоглощающий слой. При попадании частиц воды на этот слой, меняется его диэлектрическая проницаемость, что приводит к изменению емкости конденсатора.

Рисунок 18-Подключение датчика температуры и влажности DHT11

Для подключения датчика DHT11 необходимо установить библиотеку. Для этого необходимо скачать архив по ссылке, извлечь папку «**DHT11**» и переместить ее в раздел

«**C:\Program Files\Arduino\libraries**» на своем компьютере. При использовании датчика DHT11, необходимо подключать библиотеку в скетче. Загрузите следующую программу после подключения датчика температуры воздуха DHT11 к Ардуино.Скетч термодатчика DHT11 для Ардуино:

```
#include <DHT.h>      // подключаем библиотеку для датчика
DHT dht(2, DHT11);  // сообщаем на каком порту будет датчик

void setup() {
  dht.begin();              // запускаем датчик DHT11
  Serial.begin(9600);   // подключаем монитор порта
}

void loop() {
  // считываем температуру (t) и влажность (h)
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // выводим температуру (t) и влажность (h) на монитор порта
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print("Temperature: ");
  Serial.print(t);
}
```

Пояснения к коду:

1. переменные «h» и «t» являются типом данных float, которая служит для хранения чисел с десятичным разделителем;
2. команда Serial.print()выводит информацию на порт без переноса строки,
   команда Serial.println()выводит информацию на порт с переносом строки.

Пример программного кода вывода значений датчика температуры и влажности DHT11 на COM-порт ПК:

```
#include <dht11.h>

dht11 DHT;

#define DHT11_PIN 3

void setup(){

  Serial.begin(9600);

  Serial.println("tHumidity (%),\tTemperature (C)");

  }

  void loop(){

  int chk;

  chk = DHT.read(DHT11_PIN);    // READ DATA

  Serial.print(DHT.humidity,1);

  Serial.print(",\t");
```

```
 Serial.println(DHT.temperature,1);

  delay(1000);

 }
```

Открываем монитор порта. В него будут выводиться значения влажности и температуры, как представлено на рисунке 19.

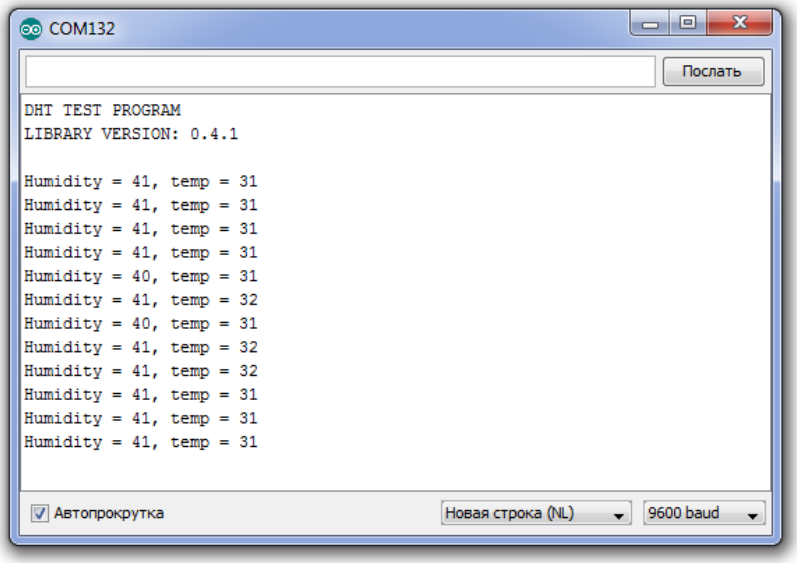


Рисунок 19- Вывод значений датчика температуры и влажности DHT11 на COM-порт ПК.

1.6.7.1 Самостоятельная работа. Создайте программу, которая каждую секунду выводит в последовательный порт температуру, измеренную аналоговым датчиком. Температура должна выводиться в градусах Цельсия. Рекомендация: для расчетов используйте значения с плавающей точкой (float). Для этого следует задавать константы в виде «5000.0».

1.6.8 Датчик освещенности

Рассмотрим еще один аналоговый датчик – датчик света, который называют фоторезистором, как представлено на рисунке 20. Датчики освещенности (освещения), построенные на базе фоторезисторов, довольно часто используются в реальных ардуино проектах. Они относительно просты, не дороги, их легко найти и купить в любом интернет-магазине. Фоторезистор ардуино позволяет контролировать уровень освещенности и реагировать на его изменение. В этой лабораторной работе мы рассмотрим, что такое фоторезистор, как работает датчик освещенности на его основе, как правильно подключить датчик в платам Arduino.Фоторезисторы достаточно активно применяются в самых разнообразных системах. Самый распространенный вариант применения — фонари уличного освещения. Если на город опускается ночь или стало пасмурно, то огни включаются автоматически. Можно сделать из фоторезистора экономную лампочку для дома, включающуюся не по расписанию, а в зависимости от освещения.

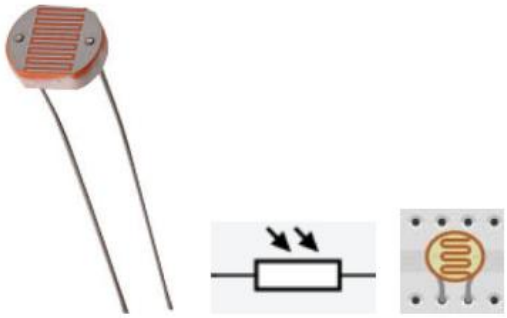


Рисунок 20-Внешний вид фоторезистора

Фоторезистор – это простой компонент с двумя входами, который изменяет свое сопротивление в зависимости от уровня освещенности. В отличие от датчика температуры TMP36 фоторезистор не выдает напряжение, зависящее от измеряемого им параметра. Он может только изменять свое сопротивление.Чтобы считать показания фоторезистора, надо сначала превратить изменение сопротивления в изменение напряжения. Для этого используют схему, которая называется делителем напряжения. Соединим один вывод фоторезистора с напряжением питания, а ко второму подключим обычный резистор, соединенный с землей. Напряжения на резисторах будут пропорциональны их сопротивлениям. Например, если сопротивление фоторезистора 20 kOm, а обычного резистора – 10 kOm, то напряжение на фоторезисторе будет в два раза больше, чем на обычном. Если, как в нашем случае, напряжение питания 5 вольт, то эти значения составят 3.33 и 1.66 вольта. При изменении сопротивления фоторезистора напряжения будут меняться. Соединив среднюю точку между резисторами с аналоговым входом платы Arduino можно измерять напряжение на обычном резисторе и делать выводы об освещенности. Фоторезисторы, которые обычно идут в наборах с платами Arduino, как правило изменяют сопротивление от 200 kOm (полная темнота) до 1 kOm (яркость 10 люкс). Экспериментируя с датчиком, можно заметить, что показания на аналоговом входе изменяются не линейно, это особенность работы датчика.

1.6.9 Маркировка фоторезистора

Современная маркировка моделей, выпускаемых в России, довольно простая. Первые две буквы — ФотоРезистор, цифры после чёрточки обозначают номер разработки. ФР - 765 — фоторезистор, разработка 765. Обычно маркируется прямо на корпусе детали.У датчика VT в схеме маркировке указаны диапазон сопротивлений. Например:

- VT83N1 — 12-100кОм (12K – освещенный, 100K – в темноте);
- VT93N2 — 48-500кОм (48K –освещенный, 100K – в темноте).

Иногда для уточнения информации о моделях продавец предоставляет специальный документ от производителя. Кроме параметров работы там же указывается точность детали. У всех моделей диапазон чувствительности расположен в видимой части спектра. Собирая датчик света нужно понимать, что точность срабатывания — понятие условное. Даже у моделей одного производителя, одной партии, одной закупки отличаться она может на 50% и более.

1.6.10 Достоинства и недостатки датчика

Основным недостатком фоторезисторов является чувствительность к спектру. В зависимости от типа падающего света сопротивление может меняется на несколько порядков. К минусам также относится низкая скорость реакции на изменение

освещённости. Если свет мигает — датчик не успевает отреагировать. Если же частота изменения довольно велика — резистор вообще перестанет «видеть», что освещённость меняется.

К плюсам можно отнести простоту и доступность. Прямое изменение сопротивления в зависимости от попадающего на неё света позволяет упростить электрическую схему подключения. Сам фоторезистор очень дешев, входит в состав многочисленных наборов и конструкторов ардуино, поэтому доступен практически любому начинающему ардуинщику.

1.6.11 Подключение фоторезистора к Ардуино

В проектах arduino фоторезистор используется как датчик освещения. Получая от него информацию, плата может включать или выключать реле, запускать двигатели, отсылать сообщения. Естественно, при этом мы должны правильно подключить датчик.Схема подключения датчика освещенности к ардуино довольна проста. Если мы используем фоторезистор, то в схеме подключения датчик реализован как делитель напряжения. Одно плечо меняется от уровня освещённости, второе – подаёт напряжение на аналоговый вход. В микросхеме контроллера это напряжение преобразуется в цифровые данные через АЦП. Т.к. сопротивление датчика при попадании на него света уменьшается, то и значение падающего на нем напряжения будет уменьшаться.

В зависимости от того, в каком плече делителя мы поставили фоторезистор, на аналоговый вход будет подаваться или повышенное или уменьшенное напряжение. В том случае, если одна нога фоторезистора подключена к земле, то максимальное значение напряжения будет соответствовать темноте (сопротивление фоторезистора максимальное, почти все напряжение падает на нем), а минимальное – хорошему освещению (сопротивление близко к нулю, напряжение минимальное). Если мы подключим плечо фоторезистора к питанию, то поведение будет противоположным.

Сам монтаж платы не должен вызывать трудностей. Так как фоторезистор не имеет полярности, подключить можно любой стороной, к плате его можно припаять, подсоединить проводами с помощью монтажной платы или использовать обычные клипсы (крокодилы) для соединения. Источником питания в схеме является сам ардуино. Фоторезистор подсоединяется одной ногой к земле, другая подключается к АЦП платы (в нашем примере – А0). К этой же ноге подключаем резистор 10 кОм. Естественно, подключать фоторезистор можно не только на аналоговый пин А0, но и на любой другой, как представлено на рисунке 21.

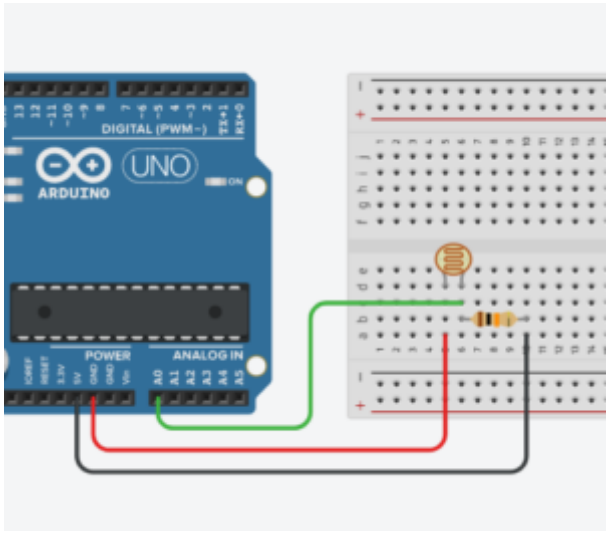


Рисунок 21- Схема подключения фоторезистора к Ардуино

Назначение дополнительного резистора на 10К. У него в нашей схеме две функции: ограничивать ток в цепи и формировать нужное напряжение в схеме с делителем. Ограничение тока нужно в ситуации, когда полностью освещенный фоторезистор резко уменьшает свое сопротивление. А формирование напряжения – для предсказуемых значений на аналоговом порту. На самом деле для нормальной работы с нашими фоторезисторами хватит и сопротивления 1К.

Меняя значение резистора мы можем “сдвигать” уровень чувствительности в “темную” и “светлую” сторону. Так, 10 К даст быстрое переключение наступления света. В случае 1К датчик света будет более точно определять высокий уровень освещенности.
Если на плате представлен цифровой выход, то отправляем его на цифровыепины. Если аналоговый – то на аналоговые. В первом случае мы получим сигнал срабатывания – превышения уровня освещенности (порог срабатывания может быть настроен с помощью резистора подстройки). С аналоговых же пинов мы сможем получать величину напряжения, пропорциональную реальному уровню освещенности.

Фоторезистор и резистор 10 кОм питаются от источника питания 5 В Arduino и образуют делитель потенциала, который защищает Arduino от коротких замыканий и гарантирует, что по крайней мере какое-то сопротивление всегда присутствует на линии.

Провод от этой схемы соединен с аналоговым входом 0 на Arduino. Резисторы понижают напряжение, проходящее через них, и поэтому для считывания изменений в освещении этой цепи вы можете использовать аналого-цифровые преобразователи (АЦП) Arduino для измерения уровня напряжения на входе. АЦП преобразуют аналоговое значение в целое число в диапазоне от 0 до 1023.

Когда фоторезистор подвергается воздействию света, его сопротивление уменьшается, и поэтому показания напряжения будут выше. Когда свет блокируется, сопротивление фоторезистора увеличивается, и поэтому показания напряжения будут ниже.Фоторезистор представляет собой простой пассивный компонент с двумя клеммами и не имеет полярности - не имеет значения, в каком направлении вы поместите его в цепь.

1.6.12 Код проекта с фоторезистором

Точные значения, выводимые на последовательном мониторе в скетче выше, будут различаться в зависимости от нескольких факторов:

- блок питания от Arduino. В частности, при питании от USB-кабеля обычно 5 В блока питания Arduino немного меньше этого идеала;
- минимальное и максимальное значения сопротивления используемого фоторезистора;
- точность резистора 10К;
- конструкция макета и используемых проводов - они имеют небольшие уровни сопротивления, которые могут повлиять на АЦП;
- и количество окружающего света в комнате.

Гораздо важнее обнаруживать изменения уровня освещенности, чем иметь дело с реальными цифрами.Считывается уровень освещенности в процедуре настройки для использования в качестве базового измерения, а затем определяется, когда фоторезистор закрыт. Когда это происходит, при вызове **digitalWrite()** загорается встроенный светодиод Arduino на цифровом выводе 13.Скетч для работы с фоторезистором приведен ниже:

```
// pin assignments

 int LED = 13;

 int LDR = 0;

  // variables

 int base;

 int threshold = 100;

  // declare inputs and outputs

 // and take a baseline reading

 void setup() {

   pinMode(LED, OUTPUT);

   pinMode(LDR, INPUT);

   base = analogRead(LDR);

 }
```

```
 // read from the analog input connected to the LDR
// and print the value to the serial port.
// the delay is only to avoid sending so much data
// as to make it unreadable.
void loop() {
  int v = analogRead(LDR);
  if ((base - v) > threshold) {
    digitalWrite(LED, HIGH);
  } else {
    digitalWrite(LED, LOW);
  }
}
```

1.6.13 Установка порогов

Приведенный выше эскиз устанавливает порог - значение, которое определяет, сколько изменений ожидается, прежде чем что-то произойдет - в коде. В зависимости от вашей среды и приложения может потребоваться отрегулировать этот порог. Чтобы избежать необходимости подключать Arduino обратно к компьютеру и перепрограммировать его, вы можете использовать потенциометр для регулировки величины сопротивления в цепи.Вы можете подключить потенциометр разными способами, пример которого показан на рисунке 22.

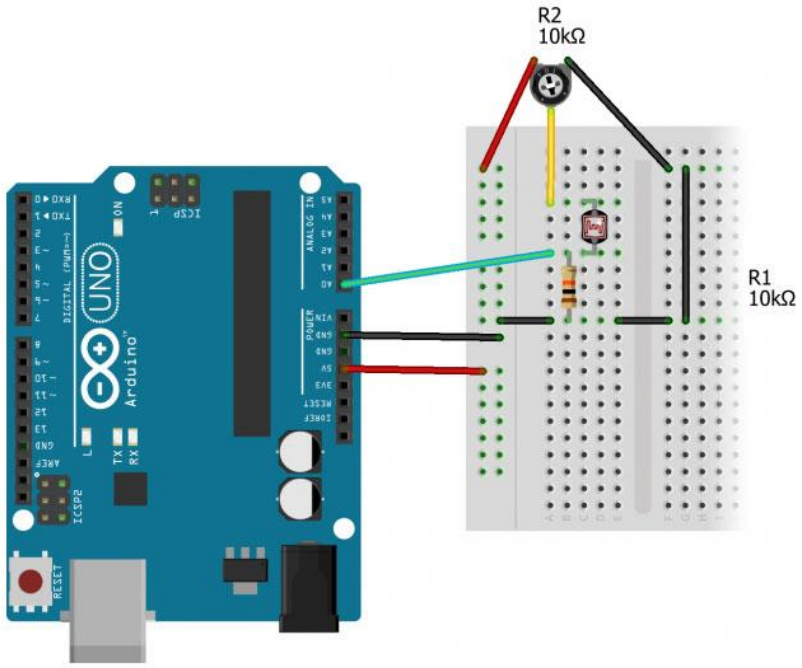


Рисунок 22-Схема подключения фоторезистора для установки порогов

Потенциометры - это другой тип переменного резистора - они обычно присоединяются к регулятору, а их сопротивление устанавливается поворотом ручки влево и вправо. На этой схеме триммер используется для изменения напряжения, подаваемого на фоторезистор. Это влияет на его способность обнаруживать свет и изменяет баланс потенциального делителя, так что количество изменений, зарегистрированных эскизом (**base - v**, в приведенном выше коде), может быть увеличено или уменьшено.

Для более цифрового подхода вы можете подключить потенциометр так же, как фоторезистор, и прочитать его, используя второй аналоговый вход. Затем вы можете использовать это измерение в эскизе, чтобы определить новое значение для переменного порога.Два примера схем в уроке демонстрируют основные шаги, связанные с обнаружением изменений в уровнях освещенности с помощью фоторезистора и Arduino. Более интересные проекты, такие как системы домашней автоматизации и сигнализации, могут быть построены с использованием дополнительных компонентов, таких как реле, двигатели и устройства беспроводной связи.

1.6.14 Пример работы датчика освещенности. Напишите скетч для датчика освещенности, включающего или выключающего светодиод, в зависимости от освещения и подключенный по следующей схеме, приведенной на рисунке 23.

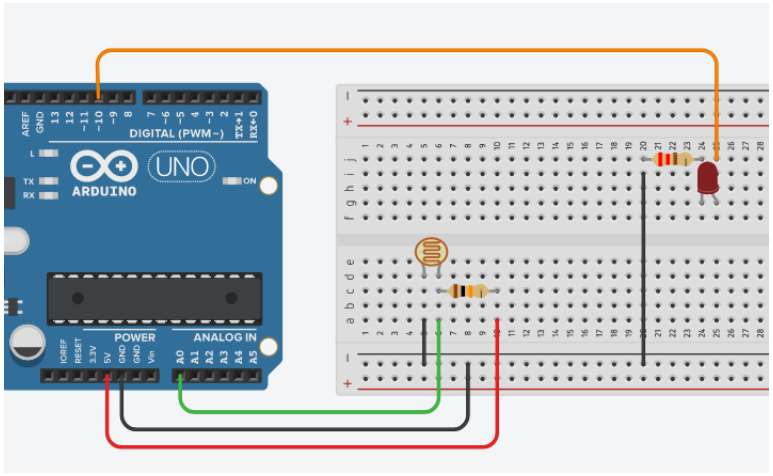


Рисунок 23-Подключение датчика освещенности и светодиода

Алгоритм работы следующий:

- определяем уровень сигнала с аналогового пина;
- сравниваем уровень с пороговым значением. Максимально значение будет соответствовать темноте, минимальное – максимальной освещенности. Пороговое значение выберем равное 300;
- если уровень меньше порогового – темно, нужно включать светодиод;
- иначе – выключаем светодиод.

Приведем код по алгоритму работы:

```
#define PIN_LED 13
#define PIN_PHOTO_SENSOR A0

void setup() {
Serial.begin(9600);
pinMode(PIN_LED, OUTPUT);
}

void loop() {
int val = analogRead(PIN_PHOTO_SENSOR);
Serial.println(val);
if (val < 300) {
digitalWrite(PIN_LED, LOW);
} else {
digitalWrite(PIN_LED, HIGH);
}
}
```

Прикрывая фоторезистор (руками или светонепроницаемым предметом), можем наблюдать включение и выключение светодиода. Изменяя в коде пороговый параметр, можем заставлять включать/выключать светодиод при разном уровне освещения.При монтаже постарайтесь расположить фоторезистор и светодиод максимально далеко друг от друга, чтобы на датчик освещенности попадало меньше света от яркого светодиода.

1.6.15 Самостоятельная работа. Реализуйте автоматический выключатель света. Подключите к плате Arduino фоторезистор и несколько светодиодов. Запрограммируйте микроконтроллер таким образом, чтобы он включал светодиоды при снижении уровня освещения и включал их, если опять становится светлее.

1.7 Индикация

1.7.1 Семисегментный индикатор

Мы уже познакомились со светодиодом, который является одним из наиболее часто используемых индикаторов. Обычным светодиодом легко информировать пользователя о каких-то бинарных событиях, то есть событиях, о которых важно знать, наступили они или нет: включении или выключении какого-либо устройства, о превышении пороговых значений и тому подобном.

Но что, если требуется сообщать пользователю более сложную информацию, например, числовую? Другими словами, что, если прибор должен сообщать измеряемые значения явно, в виде чисел? Для этих целей в электронике часто используется сегментный (или семисегментный) светодиодный индикатор. Этот прибор представляет собой набор обычных светодиодов, расположенных в виде восьмерки таким образом, что, зажигая некоторые из них, можно получить контуры различных цифр. Конструктивно, светодиодные индикаторы подразделяются на приборы с общим катодом и с общим анодом. Общий катод, к примеру, означает, что все светодиоды внутри индикатора соединены вместе катодами, а их аноды разведены по отдельным контактам. Именно такой тип индикатора используется в нашем учебном пособии.

Внешний вид и обозначение на схеме сегментного индикатора представлены на рисунке 24. Все сегменты промаркированы буквами латинского алфавита, начиная с a и заканчивая g. Как правило, счет ведут с самого верхнего сегмента по часовой стрелке. Точка маркируется отдельно, словом dot (точка). Кроме контактов, соединенных с сегментами, на схеме видны два контакта com и один dot. Слово dot используется для обозначения точки, а com – это общий катод (или анод). Он часто бывает выведен в двух местах на индикаторе.

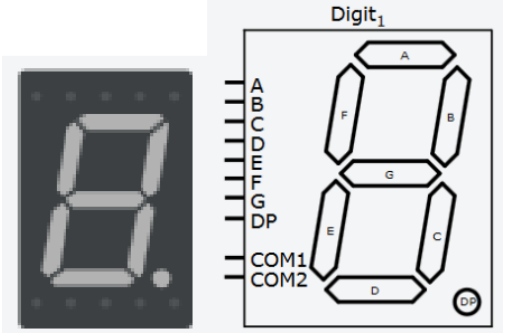


Рисунок 24-Внешний вид сегментного индикатора

Подключение семисегментного индикатора осуществляется так же, как и обычного светодиода. Так как в индикаторе присутствуют 7 диодов, потребуется использовать 7 выводов микроконтроллера и 7 резисторов, как представлено на рисунке 25.

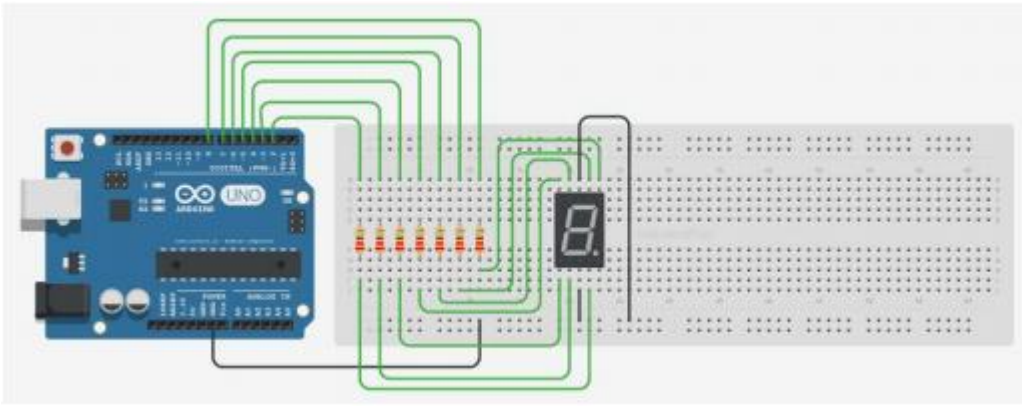


Рисунок 25-Пример подключения семисегментного индикатора

Скетч, приведенный ниже, отображает цифру 5, включая сегменты a, f, g, c, d.

```
int seg_a = 7;

int seg_b = 8;

int seg_c = 2;

int seg_d = 3;

int seg_e = 4;

int seg_f = 6;

int seg_g = 5;

 void setup() {

pinMode( seg_a, OUTPUT );

pinMode( seg_b, OUTPUT );

pinMode( seg_c, OUTPUT );

pinMode( seg_d, OUTPUT );

pinMode( seg_e, OUTPUT );

pinMode( seg_f, OUTPUT );

pinMode( seg_g, OUTPUT );

digitalWrite( seg_a, HIGH );

digitalWrite( seg_f, HIGH );

digitalWrite( seg_g, HIGH );

digitalWrite( seg_c, HIGH );

digitalWrite( seg_d, HIGH );

}

void loop() {}
```

Самостоятельная работа. 1.Реализуйте программу, осуществляющую анимацию сегментов. После включения платы на индикаторе должны по очереди загораться и потухать сегменты в порядке: верхний, правый верхний, правый нижний, нижний, левый нижний, левый верхний и т.д.

2.Реализуйте светофор с обратным отсчетом. Добавьте в созданный ранее в Задании 6.2 светофор семисегментный индикатор, на котором будет выводиться количество секунд, оставшихся до конца очередной фазы его работы. Увеличьте длительность фазы до 9 секунд.

1.7.2 Жидкокристаллический дисплей

Самый информативный и вместе с тем самый сложный вид индикаторов – это дисплеи. Познакомимся с улучшенной версией LCD дисплея, которую называют **LCD Keypad Shield**. Данный shield является очень удобным тем, что к нему подключены 5 активных кнопок, а также 1 кнопка для перезагрузки контроллера, с помощью которых вы сможете управлять некой системой. Его очень удобно использовать для создания проектов из-за того, что он занимает мало места. Подключение кнопок произведено с помощью разных резисторов, подключенных к одному аналоговому выводу. Это позволило сэкономить 5 дискретных пинов, которые мы сможем использовать в других полезных целях, как представлено на рисунке 26.

Рисунок 26-Внешний вид LCD Keypad Shield

Рассматриваемый шилд представляет собой плату с встроенными модулями индикации и управления. Индикация осуществляется с помощью LCD-дисплея TC1602, управление – через встроенные кнопки. Есть возможность регулировки яркости дисплея прямо на плате с помощью подстроечного резистора. Плата снабжена разъемами, в которые могут быть подключены другие устройства, например, датчики. Для работы с экраном используются пины 4-10, для определения нажатия кнопок – только один

аналоговый пин A0. Свободными являются цифровые пины 0-3, 11-13 и аналоговые пины A1-A5.

Основные области применения шилда: создание управляющих модулей, реализующих настройки устройства с помощью интерфейса меню. Экран шилда можно использовать для вывода информации, получаемой с датчиков, с возможностью выполнения пользователем каких-либо действий путем нажатия на встроенные кнопки. Естественно, можно найти и другие способы использования платы: например, реализовать игру типа тетрис.Перейдем к техническим характеристикам для подробного изучения шилда

1.7.3Технические характеристики LCD Keypad Shield

LCD Keypad Shield имеет следующие характеристики :

- работа дисплея: в 4 битном режиме;
- 5 активных кнопок и 1 кнопка перезагрузки контроллера;
- максимально разрешение экрана 16x2;
- для питания шилда необходимо 5 Вольт;
- частота обновления экрана до 5 Гц.

Перед загрузкой скетча скачайте и установите библиотеку LiquidCrystal. При подключении shielda к Arduino пины "4", "5", "6", "7", "8", "9" будут задействованы для управления LCD дисплея. На аналоговый пин "0" считываются сигналы с кнопок, которые различаются за счет резисторов разного сопротивления. Для управления яркостью и подсветкой шилда используется цифровой пин "10".Ниже на фотографии показана схема соединения самого шилда, как представлено на рисунке 27.

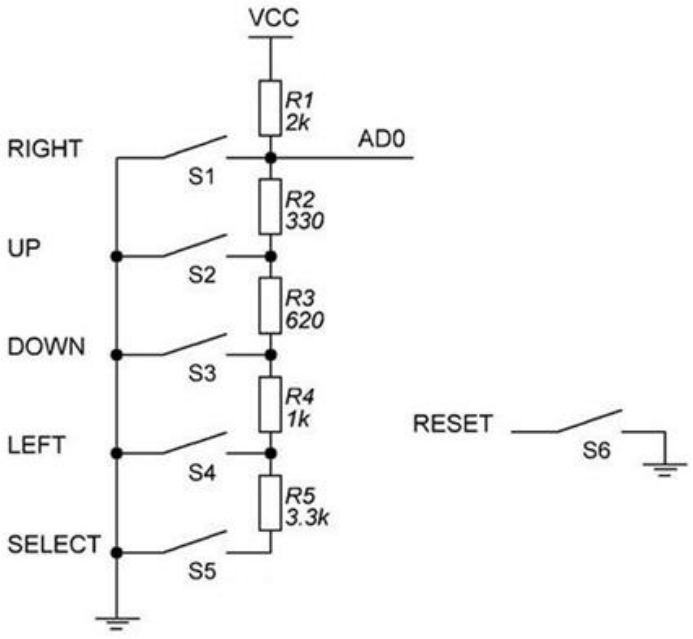


Рисунок 27- Сигналы с кнопок, которые различаются за счет резисторов разного сопротивления

1.7.4 Схема подключения

Подключение LCD Keypad Shield производится несколькими способами. Первое – нужно попасть ножками в соответствующие разъемы платы Ардуино и аккуратно совместить их. Ничего дополнительно подсоединять или припаивать не надо. Нужно помнить и учитывать тот факт, что часть пинов зарезервированы для управления дисплеем и кнопками и не может быть использована для других нужд! Для удобства подключения дополнительного оборудования на плате выведены дополнительные разъемы 5В и GND к каждой контактной площадке аналоговых пинов. Это, безусловно, упрощает работу с датчиками. Также можно подключать цифровые устройства через свободные пины 0-3 и 11-13 с помощью проводов из комплекта. Подключив шилд, мы можем работать с экраном и кнопками на нем так же, как с отдельными устройствами, учитывая только номера пинов, к которым припаяны соответствующие контакты, как показано на рисунке 28.

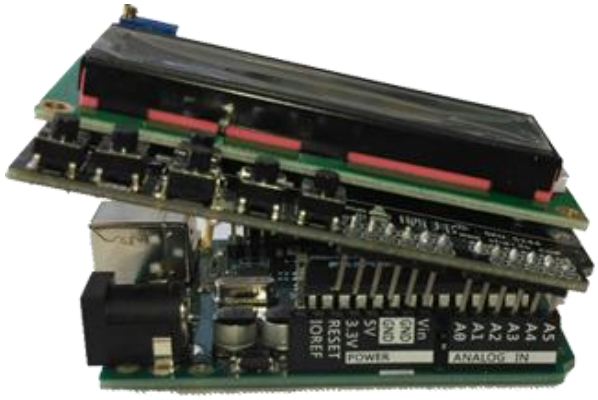


Рисунок 28-Пример установки LCD Keypad Shield в плату Arduino

1.7.5 Работа в среде Arduino IDE

Для работы с данным модулем необходимо установить библиотеку LiquidCrystal**.** Скачиваем, распаковываем и закидываем в папку libraries в папке Arduino. В случае, если на момент добавления библиотеки, Arduino IDE была открытой, перезагружаем среду. Проверим функционал библиотеки следующим кодом:

```
//Работа с курсором
lcd.setCursor(0, 0);            // Устанавливаем курсор (номер ячейки, строка)
lcd.home();                     // Установка курсора в ноль (0, 0)
lcd.cursor();                   // Включить видимость курсора (подчеркивание)
lcd.noCursor();                 // Убрать видимость курсора (подчеркивание)
lcd.blink();                    // Включить мигание курсора (курсор 5x8)
lcd.noBlink();                  // Выключить мигание курсора (курсор 5x8)

//Вывод информации
```

```
lcd.print("zelectro.cc");        // Вывод информации
lcd.clear();                    // Очистка дисплея, (удаление всех данных) установка курсора в ноль
lcd.rightToLeft();              // Запись производится справа на лево
lcd.leftToRight();              // Запись производится слева на право
lcd.scrollDisplayRight();       // Смещение всего изображенного на дисплее на один символ вправо
lcd.scrollDisplayLeft();        // Смещение всего изображенного на дисплее на один символ влево
//Информация полезная для шпионов:)
lcd.noDisplay();                // Информация на дисплее становится невидимой, данные не стираются
// если, в момент когда данная функция активна, ничего не выводить на дисплей, то
lcd.display();                  // При вызове функции display() на дисплее восстанавливается вся информация которая была
//Подсветка
lcd.backlight();                // Включение подсветки
lcd.noBacklight();              // Выключение подсветки
```

1.7.6 Пример программного кода для вывода текста на дисплей

Приведем пример программного кода для вывода текста на дисплей:

```
// Тестировалось на Arduino IDE 1.0.5
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x20,16,2);   // Задаем адрес и размерность дисплея.
int BL = 11; // Вывод управления подсветкой подключен к цифровому выводу 11 с поддержкой ШИМ
void setup()
{
 lcd.init();                    // Инициализация lcd
 // analogWrite (BL, число от 0 до 255); 0 - min яркость, 255 - max яркость
 analogWrite(BL, 255); // Включаем подсветку на максимальное значение
 // В случае, если микросхема работает от вывода P3 (в режиме ВКЛ/ВЫКЛ)
 //  lcd.backlight();             // Включаем подсветку
 //  lcd.noBacklight();           // Выключаем подсветку
 lcd.setCursor(0, 0);             // Устанавливаем курсор в начало 1 строки
 lcd.print("Hello, world!");      // Выводим текст
 lcd.setCursor(0, 1);             // Устанавливаем курсор в начало 2 строки
 lcd.print("zelectro.cc");        // Выводим текст
```

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

1.7.7 Скетч № 2 для освоения работы LCD Keypad Shield

Представим скетч, который при загрузке в Arduino сделает из LCD Keypad Shield настоящий таймер. Причем вы сможете с помощью кнопок устанавливать время до срабатывания.

```
#include  <LiquidCrystal.h> //подключаем библиотеку
LiquidCrystal lcd(8, 9, 4, 5, 6, 7 );
int x; //шина кнопок
unsigned long currentTime; //текущее время
unsigned long loopTime; //время окончания
unsigned long LimitTime = 0; //добавочное время таймера
boolean running = false; //Флаг Запуска отсчета
 void setup()
{
  lcd.begin(16, 2); //Инициализируем дисплей: 2 строки по 16 символов
  lcd.print("Timer"); //заставка
  delay(500); //ждем половину секунды
  lcd.begin(16, 2); //очистить экран
  lcd.print("www.Helpduino.ru"); //заставка
  delay(500); //ждем половину секунды
  lcd.begin(16, 2); //очистить экран
  currentTime = millis();
  // считываем время, прошедшее с момента запуска программы
}
void loop()
{
  if (running == true) {    //Выбран РЕЖИМ ОТСЧЕТА
  currentTime = millis();

 loopTime = currentTime + LimitTime;
 //Указываем время окончания (текущее время + добавочное время таймера)
 while(currentTime < loopTime){
 // сравниваем текущее время с вр окончания
```

```
//Пока текущее время меньше времени окончания
//Проверяем кнопку "Прервать"
x = analogRead (0); //считываем шину кнопок
delay(200); //защита от дребезга
if (x < 100) { // Если нажата кнопка "Right "
//"прервать" кнопка "Right "
lcd.begin(16, 2); //очистить экран
lcd.print("Interrupted"); //выводим прервано
//устанавливаем начальные значения
running = false; //останавливаем таймер
LimitTime = 0; //устанавливаем начальные значения
delay(3000); //пауза 3 секунды
lcd.begin(16, 2); //очистить экран
goto bailout; //переход на опрос клавиатуры
 }
lcd.begin(16, 2); //очистить экран
lcd.setCursor(0, 0); //курсор в 0
lcd.print("Time to off: "); //время до окончания
lcd.setCursor(13, 0); //Устанавливаем курсор
lcd.print(LimitTime/1000);
//Указываем добавочное время, переводя миллисекунды в секунды
delay(800); //ждем 0.8 секунды
LimitTime = LimitTime - (millis() - currentTime);
//уменьшаем таймер для вывода на экран
currentTime = millis(); //получаем новое время
 }
//окончание работы таймера
lcd.print("             "); //стираем экран
lcd.setCursor(0, 0); // установка курсора на нулевую строку
lcd.print("Timer: OFF"); //Выводим надпись "Timer: OFF"
running = false; //останавливаем таймер
delay (3000); //ждем 3 секунды
lcd.begin(16, 2); //очистить экран
 }
else { //иначе, выбираем режим отсчета
bailout:
keypad ();
}
```

```
}
 void keypad () { //функция опроса клавиш: вверх, вниз и выбор
 x = analogRead (0); //считываем шину кнопок
 delay(200); //ожидаем 0.2 секунды
 lcd.setCursor(0,1); //устанавливаем курсор
 lcd.print ("                "); //стираем экран
 lcd.setCursor(0,0);  //устанавливаем курсор
 lcd.print ("Time:");  //выводим слово "Time:"

 if (x < 200) { // если нажата клавиша вверх

  if ((LimitTime + 60000) <= 300005)
  //если мы указываем время меньше или равно 5 минутам минут
  {
  LimitTime += 60000; // прибавляем 1 минуту
  lcd.setCursor(7,0); //устанавливаем курсор
  lcd.print (LimitTime/60000); //выводим значение таймера в секундах
  }
  else {
  //иначе, при попытке прибавить больше 5 минут
  lcd.setCursor(0,1); //устанавливаем курсор
  lcd.print ("maximum 5 min"); //выводим надпись "maximum 5 min"
  delay (1000); //ждем 1 секунду
  lcd.setCursor(0,1); //устанавливаем курсор
  lcd.print ("                "); //стираем экран
  }
 }
 else if (x < 400){  //иначе, если нажимаем кнопку вниз
  if (LimitTime  > 60000) // если мы указываем время больше 1 минуты
  {
  LimitTime -= 60000; //отнимаем 1 минуту
  lcd.setCursor(7,0); //устанавливаем курсор
  lcd.print (LimitTime/60000); //выводим значение таймера в секундах
  }
  else { // иначе, при выборе значения меньше 1 минуты
  lcd.setCursor(0,1); //устанавливаем курсор
  lcd.print ("minimum 1 min"); //выводим надпись "minimum 1 min"
  delay (1000); //ждем 1 секунду
```

```
  lcd.setCursor(0,1); //устанавливаем курсор
  lcd.print ("               "); //стираем экран
  }

 }
  else if (x < 600){ //иначе, если нажимаем на кнопку left
  if (LimitTime !=0 ) //если добавочное время равно 0
  {running = true;                //Устанавливаем запуск отсчета
  lcd.begin(16, 2);
  lcd.setCursor(0,0); //устанавливаем курсор
  lcd.print ("Start"); //выводим надпись "Start"
 delay(1000); } //ждем 1 секунду
 else { //иначе, если не выбрали время
 lcd.setCursor(0,1); //устанавливаем курсор
 lcd.print ("Warning"); //выводим надпись "Warning"
 delay(700);//ждем 0.7 секунды
 }
}
} // Конец
```

1.7.8 Комментарии к программному коду

Команда "HelpDuino" подробно описывает суть каждой написанной строки в скетче. С помощью данного программного кода можно сделать таймер из вашего arduino и LCD Keypad Shield. Для управления таймером понадобятся четыре активные кнопки: "right", "left", "down" и "up" . Кнопки "down" и "up" нужны для выбора времени, через которое сработает таймер. Кнопкb "left" и "right" нужнs для запуска или остановки таймера соответсвенно.

1.7.9 Самостоятельная работа. 1.Реализуйте секундомер с выводом времени на дисплей. Для этого добавьте к схеме подключения дисплея две кнопки. Напишите программу, которая по одной кнопке запускает и останавливает секундомер, а по второй – сбрасывает его показания в 0. Когда секундомер запущен, на дисплее должно отображаться прошедшее с момента его запуска время в формате минуты:секунды. 2.Реализуйте вывод бегущей строки “Hello, world!”. После подачи питания в первой строке дисплея должен появиться перемещающийся справа налево текст.

1.8 Реле

1.8.1 Принцип действия и виды реле

Реле – это электромеханическое устройство для замыкания и размыкания электрической цепи. В классическом варианте реле содержит электромагнит, который управляет размыканием или замыканием контактов. Если в нормальном положении контакты реле разомкнуты, а при подаче управляющего напряжения замыкаются, такое реле называется замыкающим. Если в нормальном состоянии контакты реле сомкнуты, а при подаче управляющего напряжения размыкаются, такой тип реле называется размыкающим, как представлено на рисунке 29.

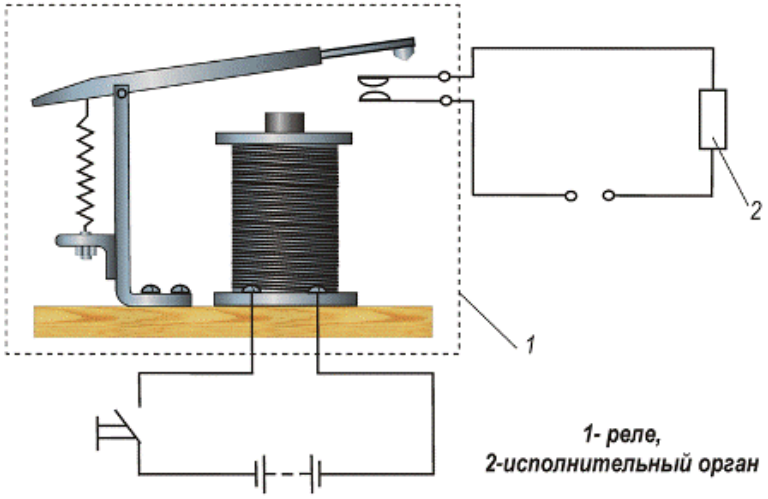


Рисунок 29-Принцип действия замыкающего реле

Кроме того, существует множество других видов реле: переключающие, одноканальные, многоканальные, реле постоянного или переменного тока, и другие.

1.8.2 Модуль реле SRD-05VDC для Arduino

В большинстве реле для Ардуино используется N-канальное управление, его мы и рассмотрим. Для примера возьмем одноканальный модуль, представленный на рисунке 30.

Рисунок 30-Внешний вид модуля реле SRD-05VDC

Рассмотрим устройство реле на широко распространенном в области Arduino реле фирмы SONGLE SRD-05VDC.Данное реле управляется напряжением 5V и способно коммутировать до 10A 30V DC и 10A 250V AC. Подача управляющего напряжения на вход IN1 или IN2 (слаботочный управляющий разъём) заставит реле скоммутировать средний контакт контактной группы K1 или K2 с правым (силовой разъём). Ток, достаточный для переключения реле – около 20 мА, цифровые выводы Arduino могут выдавать до 40 мА.

Реле имеет две раздельных цепи: цепь управления, представленная контактами A1, A2 и управляемая цепь, контакты 1, 2, 3. Цепи никак не связаны между собой.Между контактами A1 и A2 установлен металлический сердечник, при протекании тока по которому к нему притягивается подвижный якорь(2). Контакты же 1 и 3 неподвижны. Стоит отметить что якорь подпружинен и пока мы не пропустим ток через сердечник, якорь будет удерживается прижатым к контакту 3. При подаче тока, как уже говорилось, сердечник превращается в электромагнит и притягивается к контакту 1. При обесточивании пружина снова возвращает якорь к контакту 3, как представлено на рисунке 31.

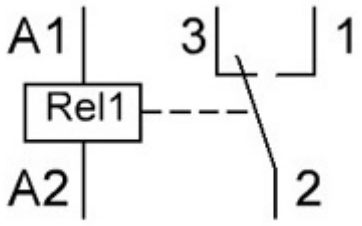


Рисунок 31-Функциональная схема реле

Модуль реле имеет 3 вывода **стандарта 2.54 мм:**

- **VCC:** "+" питания(контакт + 5 вольт на плате Ардуино);
- **GND:** "-" питания(любой из GND пинов на плате Ардуино);
- **IN:** вывод входного сигнала(цифровых входов/выходов Ардуино).

Рассмотрим скетч. В данном примере реле будет включаться и выключаться с интервалом в 2 секунды. Пример программного кода реле:

```
// Реле модуль подключен к цифровому выводу 4 int Relay = 4;

void setup () {

pinMode (Relay, OUTPUT );

}

void loop () {

digitalWrite (Relay, LOW ); // реле включено delay (2000);

digitalWrite (Relay, HIGH ); // реле выключено delay (2000);
```

}

Для подключения лампы накаливания следует поставить реле в разрыв одного из проводов:1, 2 или 3, как показано на рисунке 32.

Рисунок 32-Схема подключений реле

Далее приведем примерную схему данного модуля, показанную на рисунке 33. Необходимыми для управления реле являются следующие детали: резистор(R1) , p-n-p транзистор(VT1) , диод(VD1) и, непосредственно само реле(Rel1) . Оставшиеся два светодиода установлены для индикации. LED1 (красный) - индикация подачи питания на модуль, загорание LED2 (зеленый) свидетельствует о замыкании реле.

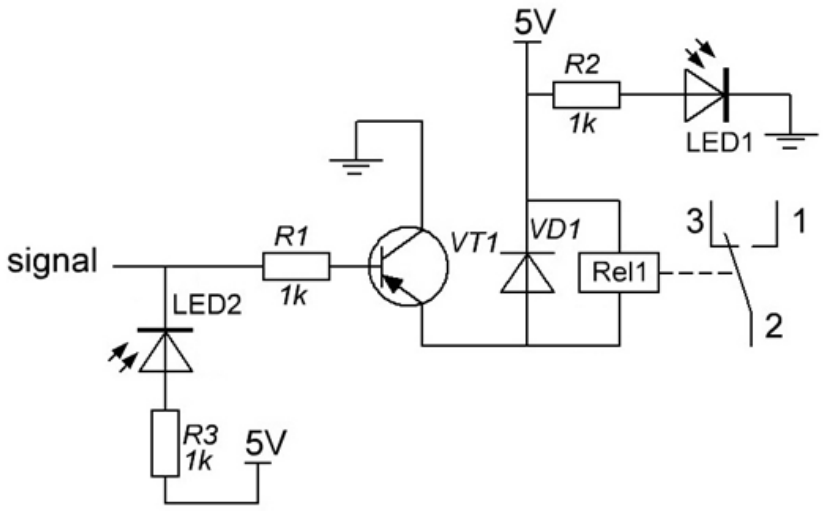


Рисунок 33- Схема модуля реле SRD-05VDC

Рассмотрим как работает схема. При включении контроллера выводы находятся в высокоомном состоянии, транзистор не открыт. Так как у нас транзистор p-n-p типа, то для его открытия нужно подать на базу минус. Для этого используем функцию digitalWrite (pin, LOW); .Теперь транзистор открыт и через управляющую цепь течет ток и реле срабатывает. Для отключения реле следует закрыть транзистор, подав на базу плюс, вызвав функцию digitalWrite (pin, HIGH).

1.8.3 Схема подключения модуля реле SRD-05VDC

Рассмотрим подключение светодиодов к модулю реле. В данном случае нам не нужна гальваническая развязка Arduino и модуля реле, поэтому будем питать модуль напрямую от платы Arduino, а джампер оставим на своём месте. Соберём схему, как показано на рисунке. Используемые резисторы – 220 Ом, светодиоды любые, как представлено на рисунке 34.

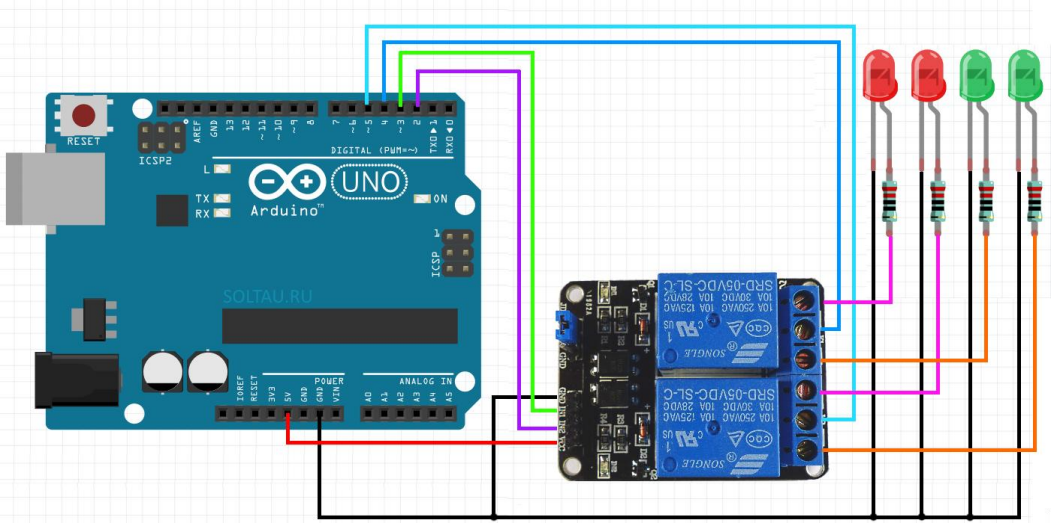


Рисунок 34-Схема подключения модуля реле SRD-05VDC к Arduino

Если светодиоды не должны никогда отключаться, можно подключить центральную точку реле не на выводы *D4* и *D5* Arduino, а напрямую на питание +5 В.

Будем поочерёдно зажигать пару светодиодов одного цвета, и каждую секунду переключаться на пару другого цвета. Напишем скетч управления реле с помощью Arduino:

```
const int relay1 = 2; // пин управления 1-ым реле

const int relay2 = 3; // пин управления 2-ым реле


const int led1 = 4; // коммутируемый вывод - питание 1-го светодиода

const int led2 = 5; // коммутируемый вывод - питание 2-го светодиода


void setup() {

 pinMode(relay1, OUTPUT);

 pinMode(relay2, OUTPUT);

 pinMode(led1, OUTPUT);

 pinMode(led2, OUTPUT);

 // установим оба реле в исходное положение:

 digitalWrite(relay1, HIGH);

 digitalWrite(relay2, HIGH);

 // подадим питание на светодиоды:

 digitalWrite(led1, HIGH);

 digitalWrite(led2, HIGH);

}
```

```
void loop() {
// переключим оба реле:
  digitalWrite(relay1, LOW);
  digitalWrite(relay2, LOW);
  delay(1000);
// переключим оба реле обратно:
  digitalWrite(relay1, HIGH);
  digitalWrite(relay2, HIGH);
  delay(1000);
}
```

Если вы собирали не по приведённой схеме, а вместо D4 и D5 подключали центральную точку реле напрямую к питанию +5V, то от констант *led1* и *led2* и от всего связанного с ними кода в скетче можно избавиться.Теперь загрузим скетч в память Arduino.Реле громко пощёлкивают раз в секунду, а светодиоды весело моргают.

Для подключения лампы накаливания 220 вольт переменного тока следует поставить реле в разрыв одного из проводов.Должно получиться так, как показано на рисунке 35.

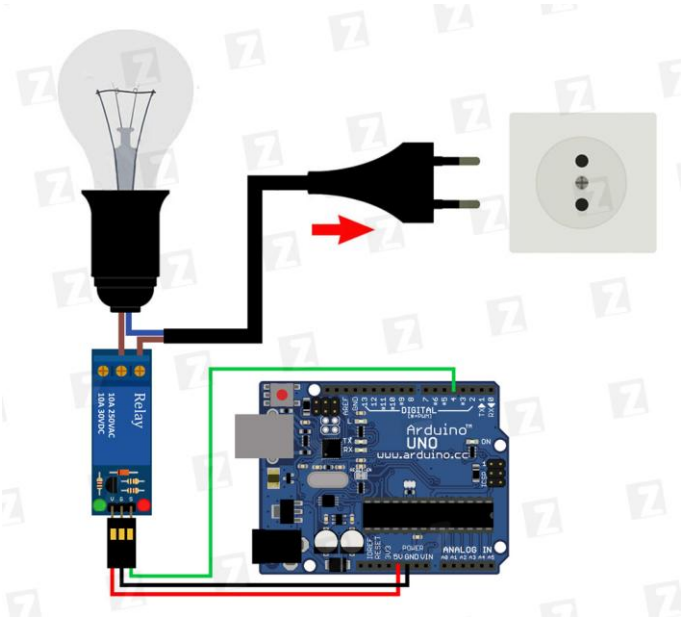


Рисунок 35-Подключение лампы накаливания к Arduino

Приведем скетч для примерного проекта-«При повышении температуры включается реле»:

```
#include <dht11.h>

dht11 DHT;

#define DHT11_PIN 2

#include <LiquidCrystal.h>
```

```
LiquidCrystal  lcd (8,9,4,5,6,7);
int SVET = 11;
void setup(){
  Serial.begin(9600);
    lcd.begin(16, 2);
  //optionally, now set up our application-specific display settings, overriding whatever the lcd did in lcd.init()
  //lcd.commandWrite(0x0F);//cursor on, display on, blink on.  (nasty!)
  lcd.clear();
  Serial.println("DHT TEST PROGRAM ");
  lcd.write("DHT TEST PROGRAM");
  delay(5000);
  Serial.print("LIBRARY VERSION: ");
  lcd.setCursor(0,0);
  lcd.write("LIBRARY VERSION:");
  lcd.setCursor(0,2);
  lcd.write(DHT11LIB_VERSION);
  Serial.println(DHT11LIB_VERSION);
  Serial.println();
  delay(5000);
  Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
  lcd.clear();
  lcd.write("ElitZoo Control");
  lcd.setCursor(0,2);
  lcd.write("Status H(%) T(C)");
  delay(5000);
  lcd.setCursor(0,2);
  lcd.write("                ");
  pinMode(SVET, OUTPUT);
  pinMode(3, OUTPUT);
}
void loop(){
```

```
int chk;
int x;
x = analogRead(A0);
Serial.print("DHT11, \t");
chk = DHT.read(DHT11_PIN);    // READ DATA
switch (chk){
  case DHTLIB_OK:
                lcd.setCursor(0,2);  //line=2, x=0
        lcd.write("OK");
                Serial.print("OK,\t");
                break;
  case DHTLIB_ERROR_CHECKSUM:
                lcd.setCursor(0,2);  //line=2, x=0
        lcd.write("ERR");
                Serial.print("Checksum error,\t");
                break;
  case DHTLIB_ERROR_TIMEOUT:
                lcd.setCursor(0,2);  //line=2, x=0
        lcd.write("TOUT");
                Serial.print("Time out error,\t");
                break;
  default:
                lcd.setCursor(0,2);  //line=2, x=0
        lcd.write("UERR");
                Serial.print("Unknown error,\t");
                break;
}
// DISPLAT DATA
 Serial.print("H=");
 Serial.print(DHT.humidity,1);
 Serial.print(",\t");
```

```
  Serial.print("T=");
  Serial.println(DHT.temperature,1);
  Serial.print(",\t");
  Serial.print("Вл=");
  Serial.print(x);
  lcd.setCursor(5,2);  //line=2, x=0
  lcd.write("H=");
  lcd.print(DHT.humidity,1);
  lcd.write("% T=");
  lcd.print(DHT.temperature,1);
  lcd.write("C");
  digitalWrite(3,HIGH);
  digitalWrite(SVET,HIGH);
  delay(5000);
  digitalWrite(3,LOW);
  digitalWrite(SVET,LOW);
  lcd.setCursor(0,2);
  lcd.write("            ");
}
```

Лабораторная работа № 1 Установка среды разработки Arduino IDE. Настройка, начало работы.

Цель работы: изучить методику установки среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы разработки Arduino IDE.

Содержание работы: ознакомиться с методикой установки среды разработки Arduino IDE на ПК, выполнив следующие шаги:

- установить среду разработки Arduino IDE;
- настроить оборудование перед началом работы-подключить плату Arduino UNO к ПК через USB-кабель, определить необходимый COM-порт;
- выполнить тестовую проверку работы разработки Arduino IDE;
- загрузить код программы, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в контроллер;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков (температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1 Краткие теоретические сведения

1.1 Платформа Arduino

Arduino (Ардуино) — аппаратная вычислительная платформа, основными компонентами которой является плата ввода-вывода и среда разработки. Алгоритм - набор последовательных действий/команд, соблюдающих логичный порядок, который позволяет решать какие-либо задачи. Для графического представления алгоритмов используются, так называемые блок-схемы. Блок-схемы — это графическое представление алгоритма, состоящее из набора геометрических фигур. Каждая фигура изображает определенную операцию или действие, представленное на рисунке 36.

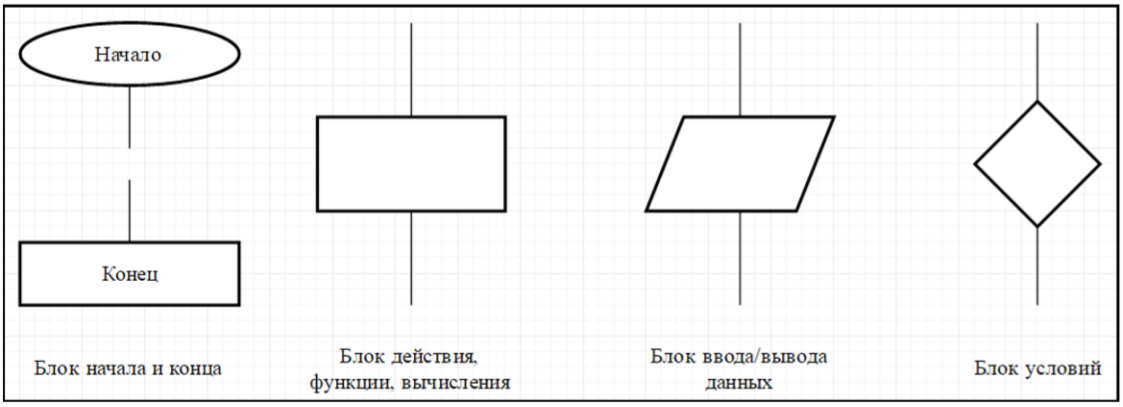


Рисунок 36- Графическое представление алгоритма

Алгоритмы бывают нескольких типов: линейные, циклические, разветвляющиеся. Среда разработки Arduino IDE — в основу языка программирования в данной среде положен C++ — один из самых популярных языков программирования. Структура простой программы, написанной в среде Arduino IDE, в сравнении с Turbo Pascal, как представлено в таблице 2.

Таблица 2- Сравнение простых программ

Arduino IDE	Turbo Pascal

int a; //обозначение переменных
int b;
void setup () {//настройки программы
Serial.begin(9600); //скорость связи данных
}
void loop () {// основная программа
b=5;
a=b*10;
Serial.println(a); //вывод на ПК ответ
delay(5000); // задержка 5 секунд
}
	PROGRAM sum; {заголовок программы}
	USES CRT; {подключение библиотек}
	VAR a, b: integer; {объявление переменных}
	BEGIN {начало блоков операторов}
	CLRSCR; {очистка экрана}
	b:=5; {оператор 1}
	a:=b*10; {оператор 2}
	WRITELN(a); {Вывод a}
	END. {конец блока операторов}

Переменные отличаются по типу данных, представленных в таблице 3.

Таблица 3- Типы данных в сравнении с Turbo Pascal

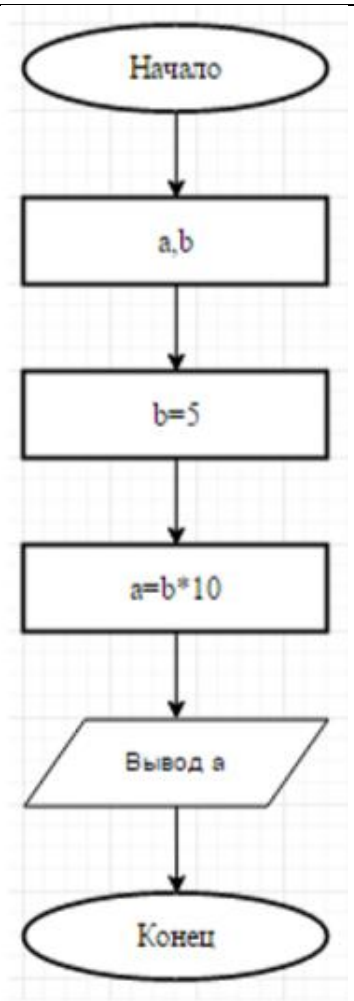
Arduino IDE	Turbo Pascal	Принимаемые значения
boolean	BOOLEAN	True/False
char	CHAR	-128/+127

byte	BYTE	0-255
int	INTEGER	-32768/+32767
long	LONGINT	-2147483648/+214748367
float	REAL/SINGLE/DOUBLE/EX TENDED	-3,4028235·1038/+3,4028235·1038

Программы с линейным алгоритмом представлены в таблице 4. Сравниваемые части программ выделены жирным курсивом.

Таблица 4- Примеры линейной программы в сравнении Arduino IDE с Turbo Pascal

Arduino IDE	Блок-схема	Turbo Pascal

int a;
int b;
void setup () {
Serial.begin(9600);
}
void loop()
{
b=5;
a=b*10;
Serial.println(a);
delay(5000);
}
		PROGRAM sum;
		USES CRT;
		VAR a, b: integer;
		BEGIN
		CLRSCR;
		b:=5;
		a:=b*10;
		WRITELN(a);
		END.

1.2 Краткий алгоритм работы со средой разработки Arduino IDE

1.2.1 Для начала зайдем на официальный сайт проекта Arduino в Интернете по адресу www.arduino.cc/en/Main/Software. Для того, чтобы скачать инсталлятор выберем в разделе Download the Arduino IDE вариант Windows Installer, как представлено на рисунке 37.

Рисунок 37- Окно скачивания программы Arduino IDE

1.2.2 После загрузки файла arduino-1.6.7-windows.exe и выше запустите его с административными полномочиями, примите условия лицензионного соглашения и принимайте последующие варианты установки. Установщик будет предлагать установить драйвера порта, на что следует отвечать утвердительно.

1.2.3 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно (рисунок 38), в котором содержится заготовка программы. В окне имеется заготовка, которая состоит из функций setup и loop. Функция setup содержит команды, выполняемые одни раз при включении — это так называемые настройки программы. А loop включает в себя основную программу. Она выполняется бесконечное число раз, до тех пор, пока мы не отключим питание.

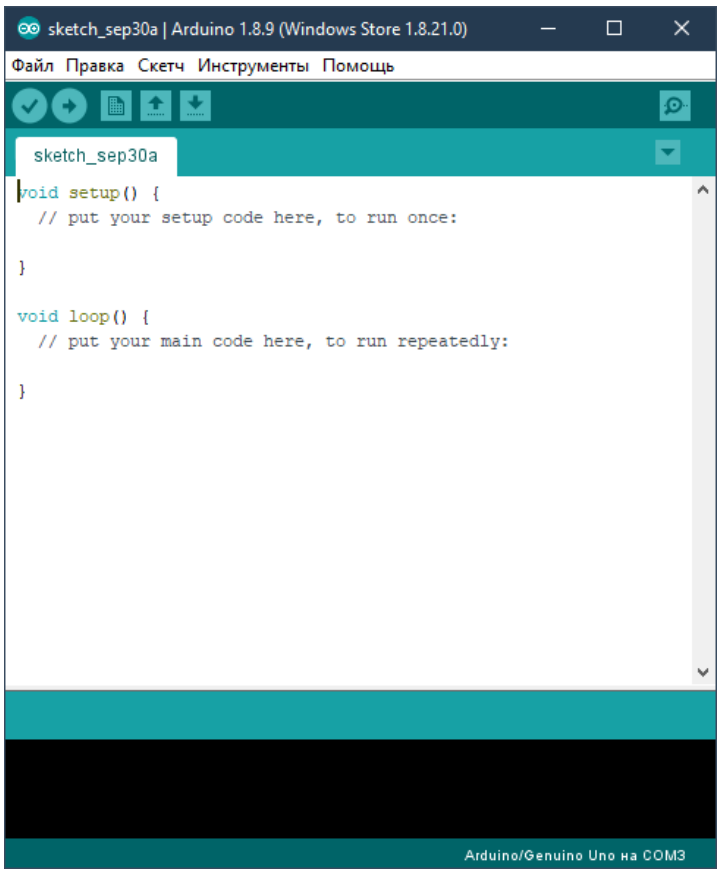


Рисунок 38- Окно программы Arduino IDE

1.2.4 Для подключения контроллера используются USB-кабели. Подключите контроллер Arduino к компьютеру кабелем. Вы увидите, как на плате загорится светодиод «ON», и начнёт мигать светодиод «L». Это означает, что на плату подано питание, и микроконтроллер ArduinoUno начал выполнять прошитую на заводе программу «Blink» (мигание светодиодом).

После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер порт (рисунок 39). Если портов много, рекомендуется запомнить все имеющиеся, а затем физически отсоединить Arduino от кабеля, и снова проанализировать список портов, — тот, который исчез, и есть нужный. Далее установите флажок на нужном порте.

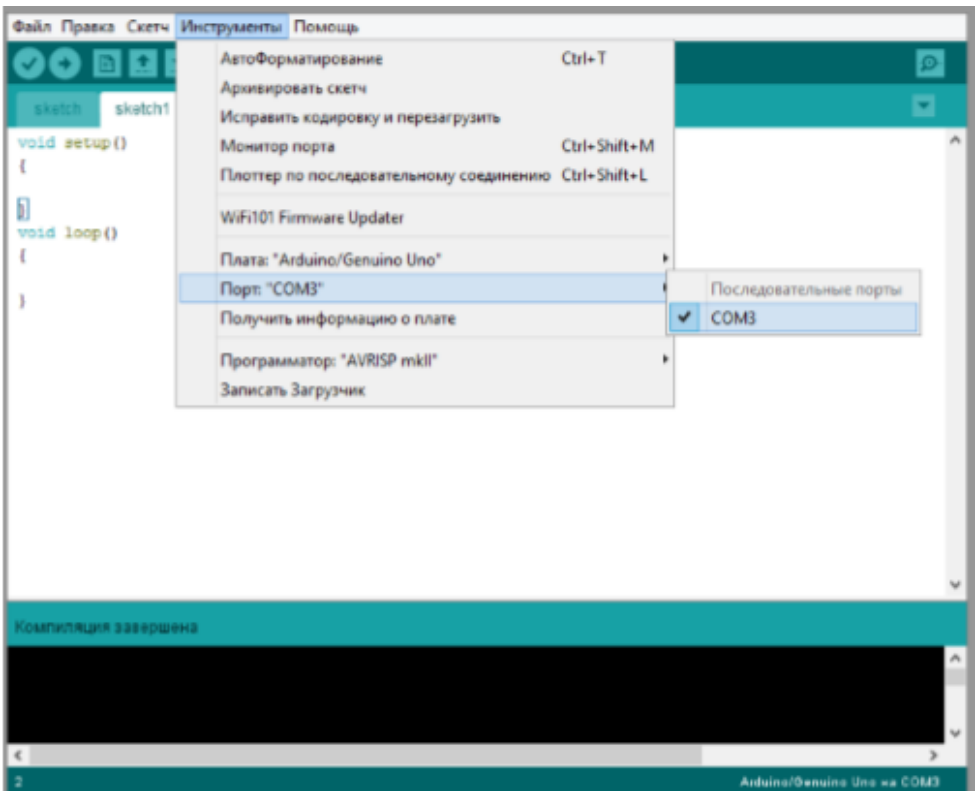


Рисунок 39- Выбор порта

Далее выберите тип вашего контроллера. На рисунке 40 можно видеть, что выбрать Arduino/Genuino Uno.

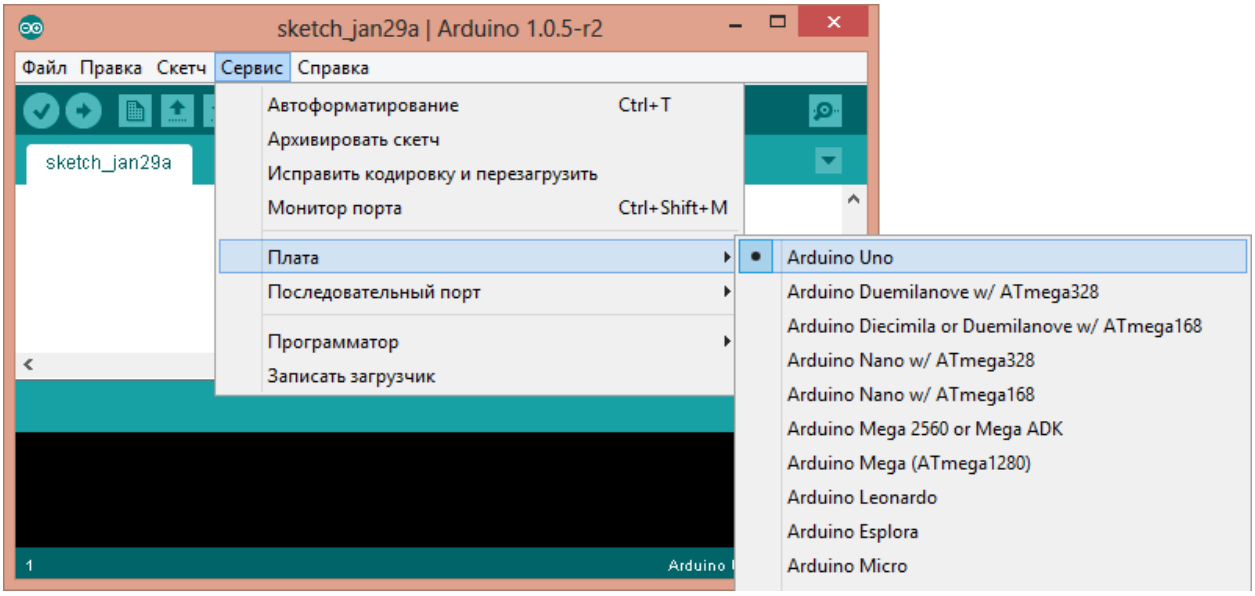


Рисунок 40- Выбор типов контроллера Arduino

Если контроллер выбран автоматически, то убедитесь в правильности выбора.

Arduino IDE содержит очень много готовых примеров, в которых можно быстро подсмотреть решение какой-либо задачи, как представлено на рисунке 41.

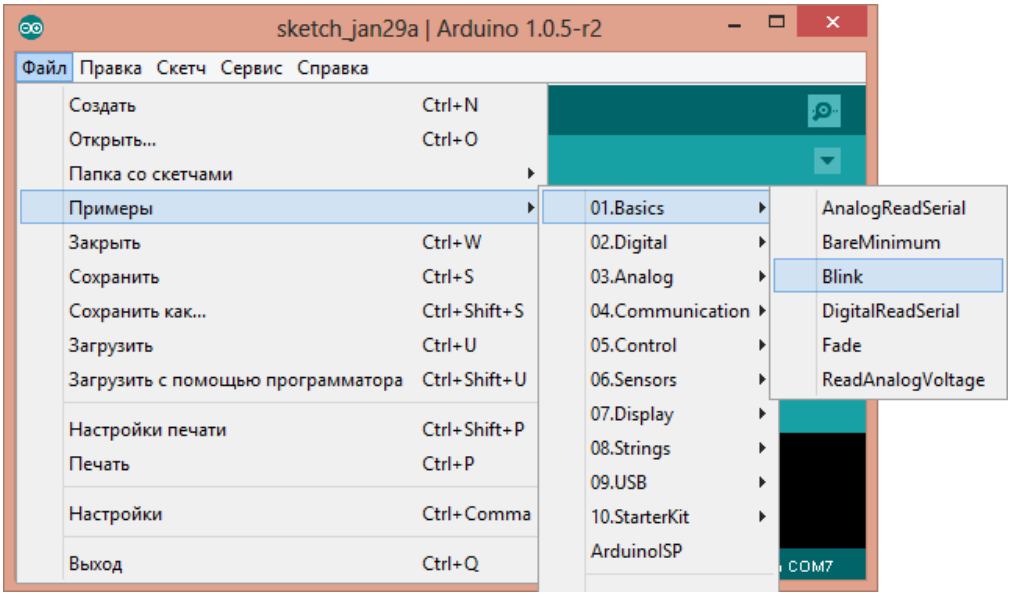


Рисунок 41- Arduino IDE и готовые примеры

1.2.5 Настройка закончена. Чтобы загрузить программу необходимо нажать кнопку

Оболочка проверит программу на наличие ошибок, а затем переведет ее в двоичный код данных и команд выбранного микроконтроллера и запишет в Arduino. 7. В качестве примера, заставим Arduino мигать встроенным светодиодом на 13-м порту (рисунок 42). Диод — это электронный элемент, который пропускает электрический ток, только в одном направлении. Светодиод — это диод, который начинает светиться при протекании по нему тока. На плате контроллера Arduino, как правило, уже установлен один светодиод — на 13-м порту (ножке).

```
void setup() // настройка
{
  pinMode (13, OUTPUT); // перевод 13-го порта в состояние вывода информации
}

void loop() // основная программа
{
  digitalWrite (13, 1);  // включение светодиода на плате
  delay (1000);          // задержка в 1 секунду
  digitalWrite (13,0);   // выключение светодиода на плате
  delay (1000);          // задержка в 1 секунду
}
```

Рисунок 42 –Фрагмент программы мигания светодиодом

Аналогичным образом перепишем программу и загрузим ее в микроконтроллер. На выходе получим мигающий на плате светодиод с задержкой в 1 секунду (рисунок 43).

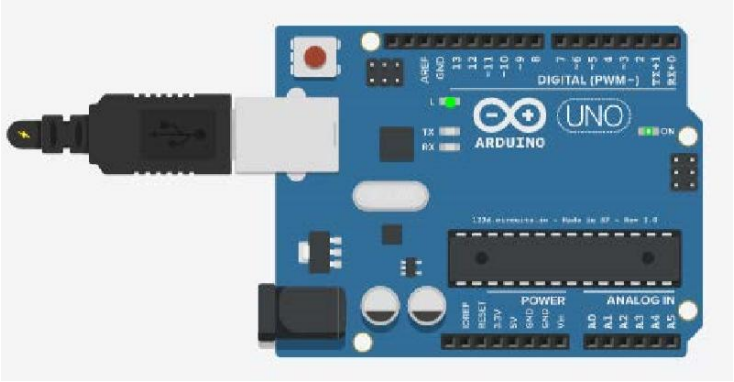


Рисунок 43-Работы светодиода на плате (L — светодиод)

1.2.6 Для подключения монитора порта, необходимо открыть вкладку Инструменты |Монитор порта и скорректировать скорость порта (9600 бод) — в окне Монитор порта скорость задается в правом нижнем углу выбором из всплывающего списка. Команда Serial.println( ) передает с контроллера на ПК значение, указанное в скобках.

1.2.7 Мигание LED-светодиодом, установленного на плате Arduino UNO

Для проверки правильности установки Arduino IDE выполним управление установленным в Arduino UNO светодиодом. Контроллер Arduino UNO уже содержит резистор и LED-светодиод, подключенный к 13 выводу, поэтому никаких других внешних радиоэлементов нам не понадобится.Приведем скетч, который можно загрузить в контроллер Arduino UNO для тестовой проверки оборудования:

```
/* Мигание LED
 * ------------
 *
 * Включает и выключает светодиод (LED) подсоединенный
 * к выходу 13, с интервалом в 2 секунды
 *
 */

int ledPin = 13;                // LED подсоединен к выводу 13

void setup()
{
  pinMode(ledPin, OUTPUT);      // устанавливаем вывод 13 как выход
}

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

Функция **delay(n)** приостанавливает обработку программы на n миллисекунд. Все это происходит в вечном цикле **loop()**.

2.Порядок выполнения работы:

2.1Скачайте инсталлятор arduino-1.6.8 и выше или возьмите у преподавателя.
2.2 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер порт ПК.

2.4 После загрузки файла arduino-1.6.8-windows.exe запустите его с административными полномочиями, примите условия лицензионного соглашения и принимайте последующие варианты установки. Установщик будет предлагать установить драйвера порта, на что следует отвечать утвердительно.

2.5 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.6 Настройка закончена. Чтобы загрузить программу необходимо нажать кнопку 

2.7 Протестируйте среду разработки Arduino IDE с помощью написанной программы, согласно своему варианту, представленному в Приложении А.

2.8 Оболочка проверит программу на наличие ошибок(компиляция), а затем переведет ее в двоичный код данных и команд выбранного микроконтроллера и запишет в Arduino.

3.Вывод о проделанной работе

4. Контрольные вопросы

1. Объясните возможности микроконтроллера ArduinoUNO для практических целей.
2. Поясните, что такое алгоритм для построения кода.
3. Назовите виды алгоритмов.
4. Дайте определение понятию блок-схема.
5. Выберите верные утверждения:
   a. «Класс» – это тип данных, а «объект» – это значение такого типа.
   b. Для использования любой стандартной функции в среде Arduino, надо в начале программы написать директиву #include и указать имя библиотеки.
   c. Библиотеки пишут для упрощения работы с различными устройствами.
   d. Чтобы работать с двумя одинаковыми устройствами, для которых есть библиотека, надо будет два раза подключить эту библиотеку с помощью #include.
   e. Чтобы работать с двумя одинаковыми устройствами, для которых есть библиотека, надо будет один раз подключить библиотеку с помощью #include и создать два объекта соответствующего класса.

Приложение А

(Варианты заданий)

Вариант 1: Выполнить установку среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы ПО Arduino IDE и контроллера Arduino UNO с помощью тестовых программ на ПК№ 1.

Вариант 2: Выполнить установку среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы ПО Arduino IDE и контроллера Arduino UNO с помощью тестовых программ на ПК№ 2.

Вариант 3: Выполнить установку среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы ПО Arduino IDE и контроллера Arduino UNO с помощью тестовых программ на ПК№ 3.

Вариант 4: Выполнить установку среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы ПО Arduino IDE и контроллера Arduino UNO с помощью тестовых программ на ПК№ 4.

Вариант 5: Выполнить установку среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы ПО Arduino IDE и контроллера Arduino UNO с помощью тестовых программ на ПК№ 5.

Вариант 6: Выполнить установку среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы ПО Arduino IDE и контроллера Arduino UNO с помощью тестовых программ на ПК№ 6.

Вариант 7: Выполнить установку среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы ПО Arduino IDE и контроллера Arduino UNO с помощью тестовых программ на ПК№ 7.

Вариант 8: Выполнить установку среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы ПО Arduino IDE и контроллера Arduino UNO с помощью тестовых программ на ПК№ 8.

Вариант 9: Выполнить установку среды разработки Arduino IDE на ПК, осуществить настройку ПО Arduino IDE, выполнить тестовую проверку работы ПО Arduino IDE и контроллера Arduino UNO с помощью тестовых программ на ПК№ 9.

Лабораторная работа № 2 Управление поочередным включением светодиодов вправо и влево в Ардуино UNO

Цель работы: разработать код для организации поочередного включения светодиодов вправо и влево с заданными интервалами времени на базе микроконтроллера Arduino UNO.

Содержание работы:

- собрать схему подключения светодиодов к макетной плате согласно задания;
- разработать код программы для заданного проекта;
- загрузить код программы в Ардуино, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в Ардуино на исполнение;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1 Краткие теоретические сведения

1.1 Подключение светодиода

Светодиод — это устройство, которое представляет собой полупроводниковый прибор, способный излучать свет при пропускании через него электрического тока в прямом направлении (от анода к катоду). Приведем схему подключения светодиодов к макетной плате согласно проекта, приведенную на рисунке 44.

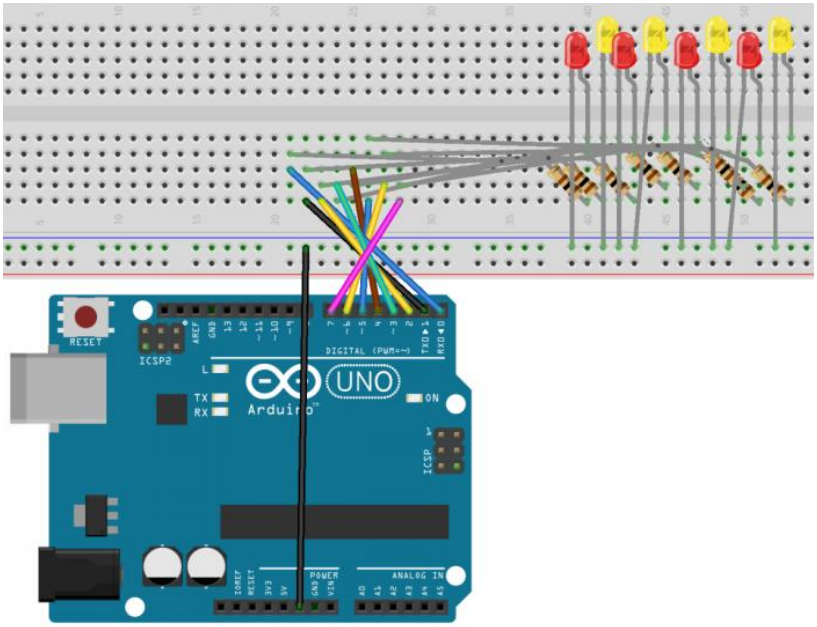


Рисунок 44- Схема подключения светодиодов в гирлянду

1.2 Побитовый сдвиг влево. Побитовая инверсия

Ознакомимся с программными основами работы с пинами (выводами) микроконтроллера как портами: «Побитовый сдвиг влево. Побитовая инверсия».Порт D Ардуино – это 8 пинов с 0-го по 7-й, порт B – 6 пинов с 8-го по 13-й. Светодиод мы зажигаем, выставляя в порту ноль. У нас восемь светодиодов и нам нужно зажигать их друг за другом, по одному.

По сути дела нам нужно записывать в порт последовательность чисел:

0b11111110
0b11111101
0b11111011
0b11110111
и так далее пока нолик не дойдет до старшего разряда.

В принципе мы могли бы сохранить в памяти микроконтроллера эту последовательность чисел, а потом просто загружать эти числа по одному в порт, но мы поступим по-другому – будем генерировать их, а для этого нам понадобится переменная. Переменную можно представить как ячейку памяти, в которой хранится некоторое значение. Это значение мы можем считывать или изменять, как заблагорассудится нашей воле. Перед тем как использовать переменную ее нужно сначала объявить - указать тип переменной и ее имя. Тип переменной определяет, сколько места в памяти будет занимать переменная.

С двумя целочисленными типами переменных мы уже сталкивались. Теперь познакомимся с наиболее родным типом данных для нашего 8-ми битного AVR микроконтроллера – **unsigned char**

unsigned char – это беззнаковое символьное число, его размер 1 байт, а диапазон значений от 0 до 255.

```
unsigned char tmp;          //объявление переменной
unsigned char var1, var2;   //можно объявлять сразу несколько переменных
unsigned char led = 0;      //объявляя переменную ей можно присвоить значение
```

Написание имени переменной подчиняется определенным правилам. Можно использовать заглавные и строчные буквы, цифры и символ подчеркивания _. Первый символ имени переменной должен быть либо буквой, либо символом подчеркивания. В качестве имени переменной нельзя использовать ключевые слова (while, if, return и все остальные зарезервированные слова). Желательно, чтобы имя переменной отражало смысл ее содержимого, это облегчает чтение программы.

У AVR микроконтроллера есть несколько видов памяти – память программ или flash память, EEPROM, оперативная память (ОЗУ) и регистры. Память программ, помимо своего прямого назначения, используется для хранения констант – переменых которые не меняют своего значения в ходе выполнения программы. EEPROM имеет ограниченное число циклов перезаписи, поэтому хранить там переменную можно только если ее значение изменяется довольно редко. А вот оперативная память и регистры наилучшим образом подходят для хранения переменных.

Переменные объявленные внутри функции называются локальными, их область видимости ограничена телом функции, Это означает, во-первых, что другие функции не могут обращаться к этим переменным (они их не видят), а во вторых, что в функциях могут быть объявлены переменные с одинаковыми именами. Локальные переменные хранятся в регистрах общего назначения микроконтроллера. При выходе из функции значение локальной переменной не сохраняется. Если нам нужно сохранять значение локальной переменной, то нужно использовать статические локальные переменные. Они сохраняются в оперативной памяти. Об этом мы поговорим позже.

Переменные объявленные вне функций называются глобальными, к ним можно обратиться из любой функции. Глобальные переменные хранятся в оперативной памяти. Когда происходит обращение к глобальной переменной, ее значение считывается в регистры общего назначения. Для непосредственного «зажигания» светодиодов с выводов Ардуино пины от 0 до 7-го подключаем через резисторы к светодиодам (к их плюсам (анодам) – длинным ножкам, минусы (катоды) – на землю) (можно резисторы поставить между светодиодами и землей, это не играет роли).

Итак, фрагмент программы:

```
//программа бегущего светодиода
#include <ioavr.h>
#include <intrinsics.h>

int main(void)
{
 unsigned char led;
 DDRC = 255;

 while(1){


                                                                    }
 return 0;
}
```

Мы подключили заголовочные файлы, объявили локальную переменную, настроили порт С на выход. Поскольку программа будет постоянно выполнять один и тот же кусок,

сразу вписали бесконечный цикл.Нам нужно заставить нолик сдвигаться влево, при этом все остальные разряды должны быть единичками. И вот как мы это реализуем. В начальном состоянии в переменной led будет единица. Мы будем сдвигать ее влево на один разряд, а в порт B будем записывать проинвертированное значение led.

1.2.1 Побитный сдвиг влево

Сдвигает число на n разрядов влево. Старшие n разрядов при этом исчезают, а младшие n разрядов заполняются нулями. Операция сдвига влево эквивалентна умножению на 2n.

```
 unsigned char tmp = 1;
tmp = tmp << 1;
//теперь в переменной tmp число 2 или 0b00000010

tmp = tmp << 3;
//теперь в переменной tmp число 16  или 0b00010000

tmp = 255;
tmp <= 2; //сокращенный вариант записи
//теперь в переменной tmp число 252 или 0b11111100
```

1.2.2 Побитная инверсия

Поразрядно инвертирует число. Разряды, в которых были нули – заполняются единицами. Разряды, в которых были единицы – заполняются нулями.

```
   unsigned char tmp =  94;     //0b01011110
tmp = ~tmp;
//теперь в переменной tmp число 161 или 0b10100001

tmp = ~tmp;
//теперь в tmp снова число 94 или 0b01011110
```

С учетом новых знаний, можно написать уже это.

```
//программа бегущего светодиода
#include <ioavr.h>
#include <intrinsics.h>

int main(void)
{
 unsigned char led = 1;
 DDRC = 255;

 while(1){
  PORTC = ~led;
  __delay_cycles(400000);
  led = led<<1;
 }
```

```
  return 0;
}
```

Данный вариант программы будет работать, но ... не долго. Как только единичка в переменной led будет сдвинута 8 раз, она затрется и в led окажется нулевое значение. А ноль как не сдвигай, толку никакого. Получается, что нам нужно после каждого сдвига проверять, не обнулилась ли наша переменная и как только это произойдет снова записать в led единицу. Для этого воспользуемся оператором ветвления **if**.

1.3 Подключение схемы к микроконтроллеру Arduino UNO

Проверим правильность выбранных элементов, соберем схему, подключим микроконтроллер Arduino UNO к ПК. Работу следует выполнять, придерживаясь следующего алгоритма:

- ✓ установка среды разработки Arduino IDE. Установка среды производится обучающимися строго по приведённому алгоритму;
- ✓ заходим в папку Студент-Ардуино;
- ✓ открываем архив Arduino-1-8-7;
- ✓ устанавливаем Ардуино на компьютер;
- ✓ устанавливаем(определяем) COM-порт ПК;
- ✓ подключаем USB-кабель;
- ✓ открываем программу Ардуино на рабочем столе;
- ✓ открываем скетч программы;
- ✓ выполняем настройку Arduino IDE на работу с ArduinoUno.

1.4 Пример кода:

```
//программа 1 «Бегущие огни»

int led = 1; // глобальная переменная, хранящая номер «зажигаемого» вывода

void setup() {

DDRD = 255; // все выводы порта D настраиваем как выходы

}

void loop() {
PORTD = led; // «зажигаются» соответствующие выводы порта D

delay(300); // задержка в 0,3 сек.

led = led<<1; // сдвиг влево – если горел 1-й светодиод, то следующим зажжётся второй

if (led > 128) // 128 = B10000000

led = 1; // после 8-го светодиода возвращаемся к первому

}
```

Пример скетча «3 светодиода горят по очереди»:

```
// объявление переменных
```

```
int RED = 3;  // Красный цвет - на 11-м пине
int GREEN = 10;  // Зеленый - на 10-м
int BLUE = 11;  // Синий - на 9
int val;  // Рабочая переменная. Используем для счетчика цикла
int count = 1;

void setup()
{
pinMode( RED, OUTPUT);
pinMode( GREEN, OUTPUT);
pinMode( BLUE, OUTPUT);
analogWrite(RED, 255);    // Красный горит
analogWrite(GREEN, 255 );    // Зеленый не горит
analogWrite( BLUE  , 255); // Синий не горит
}

void LightDown (int pin)  // Постепенно гасим цвет на пине pin
{
for (val = 0; val <=255; val++)
{
analogWrite(pin, val);  delay (20);
}
}
void LightUp (int pin)  // Постепенно усиливаем цвет на пине pin
{

for (val = 255; val >=0; val--)
{
analogWrite(pin, val);  delay (20);
}
```

```
}
void loop()
{
  if (count==1) {
analogWrite(RED, 255);    // Красный горит
analogWrite(GREEN, 0 );    // Зеленый не горит
analogWrite( BLUE  , 0);
count=2;
delay(1000);
  }
  if (count==2) {
analogWrite(RED, 0);    // Красный горит
analogWrite(GREEN, 0 );    // Зеленый не горит
analogWrite( BLUE  , 0);
count=3;
delay(1000);
  }
  if (count==3) {
analogWrite(RED, 0);    // Красный горит
analogWrite(GREEN, 255 );    // Зеленый не горит
analogWrite( BLUE  , 0);
count=4;
delay(1000);
  }
  if (count==4) {
analogWrite(RED, 0);    // Красный горит
analogWrite(GREEN, 0 );    // Зеленый не горит
analogWrite( BLUE  , 0);
count=5;
delay(1000);
  }
```

```
  if (count==5) {
analogWrite(RED, 0);    // Красный горит
analogWrite(GREEN, 0 );    // Зеленый не горит
analogWrite( BLUE  , 255);
count=6;
delay(1000);
  }
  if (count==6) {
analogWrite(RED, 0);    // Красный горит
analogWrite(GREEN, 0 );    // Зеленый не горит
analogWrite( BLUE  , 0);
delay(1000);
  }
}
```

2. Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта

2.5 Собрать схему проекта на монтажной плате для **организации поочередного включения светодиодов вправо и влево с заданными интервалами времени на базе Arduino UNO.** Подключить к питанию согласно схемы и полярности напряжения.

2.6 Разработать код для организации поочередного включения светодиодов вправо и влево с заданными интервалами времени на базе микроконтроллера Arduino UNO.

2.7 Загрузите разработанный код в среду разработки Arduino IDE.

2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino

UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.11 Продемонстрируйте преподавателю результат выполненной работы.

3.Вывод о проделанной работе.

4. Контрольные вопросы

1. Поясните, какое напряжение соответствует логическому 0 и 1 в Ардуино?

2. Выберите правильное назначение функции loop():
а) loop() выполняется один раз для инициализации (начальных установок) задействованных контактов (пинов);
б) loop() - цикл, который выполняется столько раз, сколько указано в качестве значения параметра функции;
в) loop() - бесконечный цикл, который можно остановить только отключением питания на платформе.

3.Объясните, каким образом программно изменяется временная задержка?

4. Назовите функции, какие обязательно должны присутствовать в скетче для Arduino?

a. setup();
b. main();
c. startup();
d. loop();
e. init().

5 Расскажите, для чего служат так называемые «боды».

Лабораторная работа № 3 Программирование работы светофора на светодиодах в Ардуино UNO

Цель работы: написать код программы для работы реального светофора на светодиодах на базе микроконтроллера Arduino UNO, организовав включение/выключение 3-х светодиодов попеременно.

Содержание работы:

- собрать электрическую схему проекта;
- разработать код программы для заданного проекта;
- загрузить код программы, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в контроллер;
- выполнить проверку работоспособности оборудования, согласно проекта программы.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1 Краткие теоретические сведения

1.1 Как работает светофор?

Организация работы простого светофора из 3-х светодиодов. Реальный светофор, представленный на рисунке 45, должен работать согласно следующей схеме и правилам:

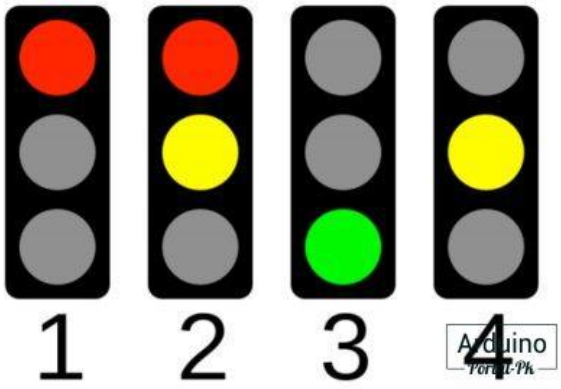


Рисунок 45- Внешний вид реального светофора

1. Светит только красный цвет нашего будущего светофора.
2. Не выключая красный сигнал светофора включаем желтый .
3. Выключаем красный и желтый включаем зеленый.
4. Выключаем зеленый сигнал светофора включаем желтый.

Примечание: Учитывая, что светодиодов у нас 3, а количество gnd портов на плате только 2, используйте следующую схему подключения, скетч напишите самостоятельно:

1.2 Схема подключения

Давайте начнем проект со сборки электрической цепи. Схема достаточно проста – соединяем три светодиода, как представлено на рисунке 46. Плюс к цифровому пину, минус – к земле. Обратите внимание, что мы объединили три контакта в один с помощью общей шины макетной платы. Красный свет светофора мы соединим с пином 11, желтый – с 10, зеленый – с 9.

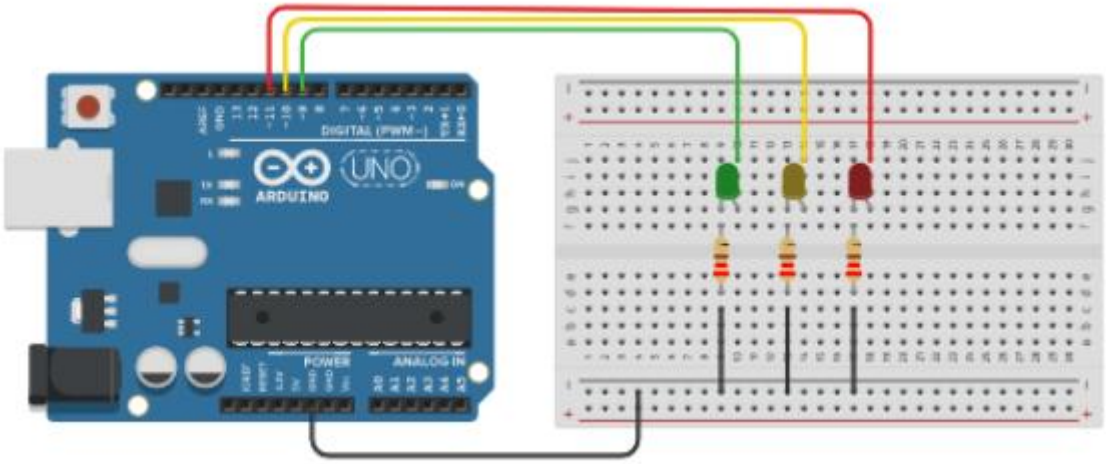


Рисунок 46-Схема светофора со светодиодами на Ардуино

1.3 Алгоритм работы

Приведем алгоритм 3-х секционного светофора для водителей, принятый за стандарт в России:

- начинается все с зеленого света, включаем его;
- спустя определенное количество времени зеленый начинает мигать. Водители и пешеходы завершают движение (или, как это часто бывает, ускоряются);
- зеленый выключается и включается желтый;
- спустя какое-то время выключается и желтый – загорается красный;
- эпоха красного цвета заканчивается не миганием, как у зеленого, а параллельным включением красного и желтого;

- спустя какое-то время красный и желтый выключаются, включается зеленый и все начинается сначала.

Согласно ГОСТу Р52282-2004 с правилами применения дорожных знаков и светофоров, длительность сигнала “красный с желтым” не должна превышать 2 секунд, длительность желтого строго равна 3 секундам. Мигание зеленого цвета происходит с частотой 1миг/секунду в течение 3-х секунд.

Если вы разобрались с алгоритмом, то написать скетч для Ардуино будет совсем не сложно. Надо лишь заменить каждое слово “включить” на **digitalWrite** с атрибутом **HIGH**, “выключить” на **digitalWrite** с атрибутом **LOW**, а задержку сформировать с помощью **delay**. Вот, например, фрагмент программы, определяющий переход с красного на зеленый цвет.

```
1. // Отключаем желтый и красный
2. digitalWrite(11, LOW); // Красный
3. digitalWrite(10, LOW); // Желтый
4. // Включаем зеленый
5. digitalWrite(9, HIGH);
6. // Ставим задержку 3 секунды
7. delay(3000);
```

Перед вами скетч мигания 1 светодиода:

```
int led = 8;
void setup() {
pinMode(led, OUTPUT); // установка 2-го контакта в режим вывода
}
void loop() {
digitalWrite(led, HIGH); // вывод №8 в активное состояние
delay(1000); // пауза 1-секунда
digitalWrite(led, LOW); // вывод №8 в неактивное состояние
delay(1000); // пауза 1-секунда
}
```

1.4 Пример скетчей для проекта светофора

Для того, чтобы не привязываться в программе к конкретным номерам пинов можно и нужно создать константы, содержащие нужный номер пина. В коде мы будем использовать эти константы, а не номера. И если нам нужно будет поменять схему подключения, то менять номера в скетче нам придется только в одном месте. Не нужно будет делать глобальную замену по документу.Вот так бы выглядел приведенный выше пример с использованием констант:

```
1. const int LED_RED = 11; // Порт 11, красный светодиод
2. const int LED_YELLOW = 10; // Порт 10, желтый светодиод
3. const int LED_GREEN = 9; // Порт 9, зеленый светодиод
4.
5. const int TIMEOUT_GREEN = 3000;
```

```
6.
7.  // Отключаем желтый и красный светодиоды.
8.  digitalWrite(LED_YELLOW, LOW);
9.  digitalWrite(LED_RED, LOW);
10.
11. // Включаем зеленый светодиод на GrnTime
12. digitalWrite(LED_GREEN, HIGH);
13. delay(TIMEOUT_GREEN);
```

Вот так можно заставить **мигать зеленый свет**. Точь в точь как обычная мигалка:

```
1.  // Мигаем зеленым светодиодом
2.  // Первый раз
3.  digitalWrite(LED_GREEN, LOW);
4.  delay(TIMEOUT_FLASH_GREEN);
5.  digitalWrite(LED_GREEN, HIGH);
6.  delay(TIMEOUT_FLASH_GREEN);
7.
8.  // Второй раз
9.  digitalWrite(LED_GREEN, LOW);
10. delay(TIMEOUT_FLASH_GREEN);
11. digitalWrite(LED_GREEN, HIGH);
12. delay(TIMEOUT_FLASH_GREEN);
13.
14. // Третий раз
15. digitalWrite(LED_GREEN, LOW);
16. delay(TIMEOUT_FLASH_GREEN);
17. digitalWrite(LED_GREEN, HIGH);
18. delay(TIMEOUT_FLASH_GREEN);
```

Второй и более правильный вариант мигания – использовать цикл FOR. Более подробно о нем написано в отдельной статье про циклы.

```
1.  for (int i=0; i<3; i++){
2.  digitalWrite(LED_GREEN, LOW);
3.  delay(TIMEOUT_FLASH_GREEN);
4.  digitalWrite(LED_GREEN, HIGH);
5.  delay(TIMEOUT_FLASH_GREEN);
6.  }
```

Вот и все особенности. Соберем код вместе и напишем итоговую программу:

```
1.  const int LED_RED = 13; // Порт 13, красный светодиод
2.  const int LED_YELLOW = 12; // Порт 12, желтый светодиод
3.  const int LED_GREEN = 11; // Порт 11, зеленый светодиод
4.  const int TIMEOUT_RED = 3000; // Время горения красного сетодиода
5.  const int TIMEOUT_YEL = 1690; // Время горения желтого светодиода
6.  const int TIMEOUT_GREEN = 2000; // Время горения зеленого светодиода
7.  const int TIMEOUT_FLASH_GREEN = 500; // Время мигания зеленого светодиода
8.  void setup()
9.  {
10. // Все порты светодиодов будут у нас установлены в режим "внешняя нагрузка", OUTPUT
11. pinMode(LED_RED, OUTPUT);
12. pinMode(LED_YELLOW, OUTPUT);
13. pinMode(LED_GREEN, OUTPUT);
14. // Устанавливаем начальное значение светодиодов
```

```
15. digitalWrite(LED_RED, LOW);
16. digitalWrite(LED_YELLOW, LOW);
17. digitalWrite(LED_GREEN, LOW);
18. }
19. void loop()
20. {
21. // Включаем зеленый цвет светофора
22. digitalWrite(LED_GREEN, HIGH); // Включаем светодиод
23. delay(TIMEOUT_GREEN); // Ждем
24.
25. // Мигаем зеленым светодиодом 3 раза
26. for (int i=0; i<3; i++)
27. {
28. digitalWrite(LED_GREEN, LOW);
29. delay(TIMEOUT_FLASH_GREEN);
30. digitalWrite(LED_GREEN, HIGH);
31. delay(TIMEOUT_FLASH_GREEN);
32. }
33. // Теперь отключаем зеленый и включаем желтый светодиод
34. digitalWrite(LED_GREEN, LOW);
35. digitalWrite(LED_YELLOW, HIGH);
36. delay(TIMEOUT_YEL);
37. // Отключаем желтый светодиод.
38. digitalWrite(LED_YELLOW, LOW);
39. // Теперь включаем красный цвет
40. digitalWrite(LED_RED, HIGH);
41. delay(TIMEOUT_RED);
42. // Включаем желтый светодиод,не выключая красный
43. digitalWrite(LED_YELLOW, HIGH);
44. delay(TIMEOUT_YEL);
45. // Отключаем желтый и красный светодиоды.
46. digitalWrite(LED_YELLOW, LOW);
47. digitalWrite(LED_RED, LOW);
48. }
```

Загрузите скетч в Ардуино и убедитесь, что все работает правильно.

2.Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта

2.5 Собрать схему проекта на монтажной плате, подключить к питанию согласно схемы и полярности напряжения. Схема состоит 3-х светодиодов в последовательности-красный, желтый, зеленый.

2.6 Разработать код программы согласно варианта, указанного в задании и Приложении В.

2.7 Загрузите разработанный код в среду разработки Arduino IDE.

2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.11 Продемонстрируйте преподавателю результат выполненной работы.

3.Вывод о проделанной работе.

4. Контрольные вопросы

1.Объясните, каким образом программно изменить частоту сигнала, подаваемого на динамик.

2. Опишите, каким образом с помощью светодиода определяется нажатое состояние кнопки.

3. Определите правильный объем памяти микроконтроллера Arduino UNO из предложенных вариантов: 32 Мб, 32 Кб, 256 Мб или 1 Гб?

4. Определите количество размещенных на платформе Arduino UNO цифровых контактов (ввод/вывод) из предложенных вариантов:5, 10, 14 или 20?

5.Определите количество размещенных на платформе Arduino UNO контактов аналогового ввода из предложенных вариантов :6, 10, 15 или 20?

Приложение В

(Варианты заданий)

Вариант 1: 1. Организуйте мигание двух светодиодов (желтый и красный) по правилу:

- желтый светодиод мигает 4 секунды;
- задержка 1 секунда;
- красный светодиод мигает 6 секунд;
- задержка 1 секунда.

2. Написать скетч мигания 3-х светодиодов по типу работы реального светофора в последовательности-красный, желтый, зеленый.

Вариант 2: 1. Организуйте мигание двух светодиодов (синий и красный) по правилу:

- ✓ синий светодиод горит постоянно;
- ✓ красный светодиод мигает 10 секунд;
- ✓ задержка 4 секунды.

2. Написать скетч мигания 3-х светодиодов по типу работы реального светофора в последовательности-красный, желтый, зеленый.

Лабораторная работа № 4 Программирование работы светофора с секцией для пешеходов на светодиодах в Ардуино UNO

Цель работы: написать код программы для работы реального светофора с секцией для пешеходов на светодиодах в Arduino UNO.

Содержание работы:

- собрать схему подключения светодиодов к макетной плате согласно задания;
- разработать код программы;
- загрузить код программы в Ардуино, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в Ардуино на исполнение;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

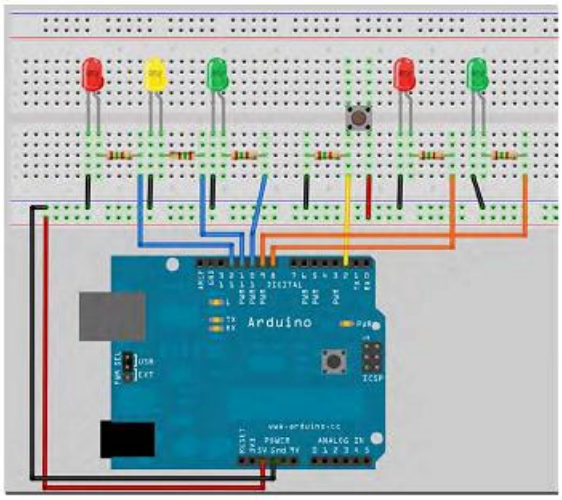
- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1. Краткие теоретические сведения

1.1Светофор с секцией для пешеходов

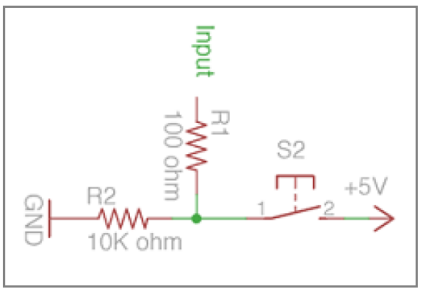
Требуется построить светофор. Светофор имеет две секции — автомобильную и пешеходную. Светофор управляется кнопкой.В начальный момент горит зеленый свет для автомобилей и красный — для пешеходов.При нажатии кнопки на 5 секунд открывается пешеходный переход: через желтый загорается красный свет для автомобилей и загорается зеленый свет для пешеходов. Через 5 секунд пешеходный переход закрывается и открывается движение автомобилей.

Цикл выполняется бесконечно. Интервал между последовательными включениями пешеходного перехода не может быть меньше 5 секунд.Соберите схему в соответствии с рисунком 47:



Особенности подключения кнопки

(рисунок из книги Michael MacRoberts «Beginning Arduino»)



Кнопка подключается к электрической сети через так называемый подтягивающий резистор (Pull-Down resistor). Этот резистор необходим для исключения электрического шума в сети, который может привести к неправильному срабатыванию кнопки.

Рисунок 47-Схема подключения светодиодов для проекта светофора с секцией

Примерный вид скетча «Интерактивный светофор с секцией для пешехода»:

```
int carRed = 12;           // Выход (пин) для автомобильной секции (красный)

int carYellow = 11;     // Выход (пин) для автомобильной секции (жёлтый)

int carGreen = 10;     // Выход (пин) для автомобильной секции (зелёный)

int pedRed = 9;         // Выход (пин) для пешеходной секции (красный)

int pedGreen = 8;     // Выход (пин) для пешеходной секции (зелёный)

int button = 2;                          // Пин кнопки

int crossTime = 5000;                 // Задаем время для пешеходного перехода

int carTime = 5000;                   // Задаем минимальное время проезда автомобилей

unsigned long changeTime=0;    //Время последнего включения пешеходного перехода

// Тип long позволяет работать с переменными в диапазоне

// от -2 147 483 648 до 2 147 483 647

// Тип unsigned long определяет неотрицательные числа, то есть числа // в диапазоне от 0 до 4 294 967 295 (занимают 32 бита или 4 байта)

// Инициализация пинов

void setup() {

    pinMode(carRed, OUTPUT);

   pinMode(carYellow, OUTPUT);

   pinMode(carGreen, OUTPUT);

   pinMode(pedRed, OUTPUT);

   pinMode(pedGreen, OUTPUT);
```

```
  pinMode(button, INPUT); // Кнопка подключена к пину 2
  // Начальные установки на светофоре:
 // Зелёный свет для автомобиля, красный — для пешеходов
 digitalWrite(carGreen, HIGH); // Зеленый свет для автомобиля
 digitalWrite(pedRed, HIGH); // Красный свет для пешеходов
}
void loop() {
       int state = digitalRead(button); // Запоминаем состояние кнопки
   // Проверяем два условия:
  // нажата ли кнопка и
  // прошло ли 5 секунд (carTime)с момента закрытия перехода.
 // Если условия выполнены, то вызывается процедура
 // изменения режима работы светофора для разрешения перехода
 if (state == HIGH && (millis() - changeTime) > carTime) {
 changeLights();
 }
}
// Процедура изменения режима работы светофора для разрешения перехода
void changeLights() {
  digitalWrite(carGreen, LOW); // Выключаем зеленый для автомобилей
 digitalWrite(carYellow, HIGH); // Включаем желтый
 delay(2000); // Ждем 2 секунды
 digitalWrite(carYellow, LOW); // Выключаем желтый
 digitalWrite(carRed, HIGH); // Включаем красный
 delay(1000); // В целях безопасности ждем 1 секунду
 digitalWrite(pedRed, LOW); // Выключаем красный пешеходной секции
 digitalWrite(pedGreen, HIGH); // Включаем зеленый пешеходной секции
 delay(crossTime); // Ждем заданное время
 for (int i=0; i<10; i++) { // Предупреждающее мигание зеленого света
     digitalWrite(pedGreen, HIGH);
       delay(250);
```

```
        digitalWrite(pedGreen, LOW);
        delay(250);
}
  digitalWrite(pedRed, HIGH); // Включаем красный пешеходной секции
  delay(500);
  digitalWrite(carYellow, HIGH); // Включаем желтый автомобильной секции
  digitalWrite(carRed, LOW); // Выключаем красный
  delay(1000);
  digitalWrite(carGreen, HIGH); // Включаем зеленый
  digitalWrite(carYellow, LOW); // Выключаем желтый
  // Запоминаем момент, когда пешеходный переход был закрыт
  changeTime = millis();
  // Возвращаемся в основную процедуру loop()
  }
```

1.2 Что необходимо помнить?

Приведем примеры использования типа данных и сложных логических условий, представленных в таблице 5.

Таблица 5-Использование типа данных и условий

1. Применение подтягивающего резистора, цифровой кнопки	http://wiki.amperka.ru/схемотехника:резисторы
2. Использование типа данных unsign long	Определяет неотрицательные числа в диапазоне от 0 до 4 294 967 295 = 2^32 – 1. Данные занимают 32 бита или 4 байта.
3. Использование сложного логического условия	*if (state == HIGH && (millis()-changeTime)> carTime)* *{ changeLights(); }* В сложных логических условиях могут быть использованы операции: Логическое отрицание ! Логическое сложение || Логическое умножение &&
4. Использование функции millis()	Возвращает количество миллисекунд с момента начала выполнения текущей программы на плате Arduino (unsigned long) Параметры: нет

2. Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта

2.5 Собрать схему проекта светофора с секцией для пешехода из светодиодов на монтажной плате, подключить к питанию согласно схемы и полярности напряжения. Собранная схема должна быть проверена преподавателем.

2.6 Разработайте код программы согласно варианта, указанного в задании и Приложении С.

2.7 Загрузите разработанный код в среду разработки Arduino IDE.

2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.11 Продемонстрируйте преподавателю результат выполненной работы.

3.Контрольные вопросы

1.Определите соответствие контактов и обозначение знаков, допускающие широтно-импульсную модуляцию (ШИМ): #, $, & или ~.
2. Определите количество уровней сигнала, которые позволяет использовать широтно-импульсная модуляция (ШИМ): 64, 128, 256 и 1024.
3. Поясните, какой вид напряжения использует платформа Arduino: 5В, 9В, 3.3В или 12 В.
4. Назовите язык для программирования Arduino: PascalABC, Free Pascal, Wiring, упрощенная версия C++, Pithon или Visual Basic.

4.Вывод о проделанной работе.

Приложение С

(Варианты заданий)

Вариант 1: Построить модель светофора. Светофор имеет две секции-автомобильную и пешеходную. Светофор управляется кнопкой.В начальный момент горит зеленый свет для автомобилей и красный — для пешеходов.При нажатии кнопки на 15 секунд открывается пешеходный переход: через желтый загорается красный свет для автомобилей и загорается зеленый свет для пешеходов. Через 15 секунд пешеходный переход закрывается и открывается движение автомобилей.Цикл выполняется бесконечно. **Добавить- звуковой сигнал для пешеходов.**

Вариант 2: Построить модель светофора. Светофор имеет две секции-автомобильную и пешеходную. Светофор управляется кнопкой.В начальный момент горит зеленый свет для автомобилей и красный — для пешеходов.При нажатии кнопки на 25 секунд открывается пешеходный переход: через желтый загорается красный свет для автомобилей и загорается зеленый свет для пешеходов. Через 25 секунд пешеходный переход закрывается и открывается движение автомобилей.Цикл выполняется бесконечно.**Добавить- временной отсчет для пешеходов (с помощью 7-сегментного числового индикатора или** LCD Keypad Shield**).**

Лабораторная работа № 5 Программирование работы светофора с дополнительной секцией для поворотов направо или налево на светодиодах в Ардуино UNO

Цель работы: написать код программы для программирования работы светофора на светодиодах на базе микроконтроллера Arduino UNO, организовав работу дополнительной секции светофора для поворотов направо или налево.

Содержание работы:

- обрать схему подключения светодиодов к макетной плате согласно задания;
- разработать код программы;
- загрузить код программы в Ардуино, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в Ардуино на исполнение;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1.Краткие теоретические сведения

Приведем реальный светофор с секцией для поворота налево а) и поворота направо б), показанные на рисунке 48.

а) б)

Рисунок 48-Реальный светофор с секцией для поворота налево а) и поворота направо б)

Приведем реальный светофор с секцией одновременно для поворота налево и поворота направо , показанный на рисунке 49.

Рисунок 49- Реальный светофор с секцией одновременно для поворота налево и поворота направо

Используя скетч для светофора с секцией в выполненной лабораторной работе № 4 «Программирование работы светофора с секцией для пешеходов на светодиодах в Ардуино UNO» разработайте алгоритм работы реального светофора с секцией для поворота налево или направо согласно варианта.

Для примера приведем скетч работы реального светофора со стрелкой.

```
const int pinPhoto = A0;

int raw = 0;

// подключение RGB-светодиода

const int pinRED=6; // вывод красной ноги RGB-светодиода

const int pinGREEN=5; // вывод зеленой ноги RGB-светодиода

const int pinBLUE=3; // вывод синей ноги RGB-светодиода

const int pinARR=11; // ввод стрелки

int count = 1;

void setup() {

  Serial.begin(9600);

  pinMode(pinPhoto, INPUT);

  // настройка выводов подключения к RGB светодиоду

pinMode(pinRED,OUTPUT);

pinMode(pinGREEN,OUTPUT);

pinMode(pinBLUE,OUTPUT);

pinMode(pinARR, OUTPUT);

}


void loop() {
```

```
raw = analogRead(pinPhoto);
Serial.println(raw);
 if (count==4){
  setRGB(0,1,0,0);
  count=5;
  delay(1800);
}
if (count==3){
  setRGB(0,0,1,0);
  count=4;
  delay(4000);
  setRGB(0,0,0,0);  //off
  delay(200);
  setRGB(0,0,1,0);  //1
  delay(200);
  setRGB(0,0,0,0);  //off
  delay(200);
  setRGB(0,0,1,0);  //2
  delay(200);
  setRGB(0,0,0,0);  //off
  delay(200);
  setRGB(0,0,1,0);  //3
}
if (count==2){
  setRGB(0,1,0,0);
  count=3;
  delay(1800);
}
if (count==1){
  setRGB(1,0,0,0);
  count=2;
```

```
    delay(1000);
    setRGB(1,0,0,1);
    delay(2000);
    setRGB(1,0,0,0);  //off
    delay(300);
    setRGB(1,0,0,1);  //1
    delay(300);
    setRGB(1,0,0,0);  //off
    delay(300);
    setRGB(1,0,0,1);  //2
    delay(300);
    setRGB(1,0,0,0);  //off
    delay(300);
    setRGB(1,0,0,1);  //3
    delay(300);
    setRGB(1,0,0,0);  //off
    delay(2000);
  }
  if (count==5){
    count=1;
  }
  delay(200);
}

void setRGB(int R, int G, int B, int A) {
digitalWrite(pinRED,R);
digitalWrite(pinGREEN,G);
digitalWrite(pinBLUE,B);
digitalWrite(pinARR,A);
}
```

2.Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.

2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта

2.5 Собрать схему проекта реального светофора с дополнительной секцией для поворотов направо или налево на монтажной плате, подключить к питанию согласно схемы и полярности напряжения.

2.6 Разработайте код программы согласно варианта, указанного в задании и Приложении D.

2.7 Загрузите разработанный код в среду разработки Arduino IDE.

2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.11 Продемонстрируйте преподавателю результат выполненной работы.

3.Вывод о проделанной работе.

4. Контрольные вопросы

1.Назовите функции, которые должны присутствовать в скетче для Arduino: setup(), main(), loop(), function().

2.Поясните, верно ли утверждение: программу для Arduino можно писать в любом регистре - строчные и заглавные буквы не различаются.

3. Назовите ошибки какие сделал программист, написав команду для подачи высокого напряжения на встроенный светодиод, подключенный к 13 контакту (пину): DigitalWrite(High): а) неправильное использование регистра при вводе команды digitalWrite;

б) неправильное количество параметров в команде;

в) неправильное использование регистра в параметре HIGH;

г) неправильное использование значение HIGH вместо LOW.

4. Выберите правильное назначение функции loop():

а) loop() выполняется один раз для инициализации (начальных установок) задействованных контактов (пинов);

б) loop() - цикл, который выполняется столько раз, сколько указано в качестве значения параметра функции;

в) loop() - бесконечный цикл, который можно остановить только отключением питания на платформе.

Приложение D

(Варианты заданий)

Вариант 1: Построить модель светофора на светодиодах.Написать код программы для работы светофора на светодиодах на базе микроконтроллера Arduino UNO, организовав работу дополнительной секции светофора для поворотов направо.

Вариант 2:Построить модель светофора на светодиодах.Написать код программы для работы светофора на светодиодах на базе микроконтроллера Arduino UNO, организовав работу дополнительной секции светофора для поворотов налево.

Лабораторная работа № 6 Управление цветом RGB-светодиода для проекта «Все цвета радуги» в Ардуино UNO

Цель работы: написать код программы для управления цветом RGB-светодиода для проекта «Все цвета радуги» в Ардуино UNO.

Содержание работы:

- собрать схему подключения светодиодов к макетной плате согласно задания;
- разработать код программы для заданного проекта;
- загрузить код программы в Ардуино, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в Ардуино на исполнение;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, RGB-светодиод, кнопки.

1 Краткие теоретические сведения

В этом проекте мы используем RGB-светодиод. В отличие от предыдущих моделей – одноцветных светодиодов — RGB-светодиод является полноцветным, то есть может светиться любым цветом, представленный на рисунке 50.

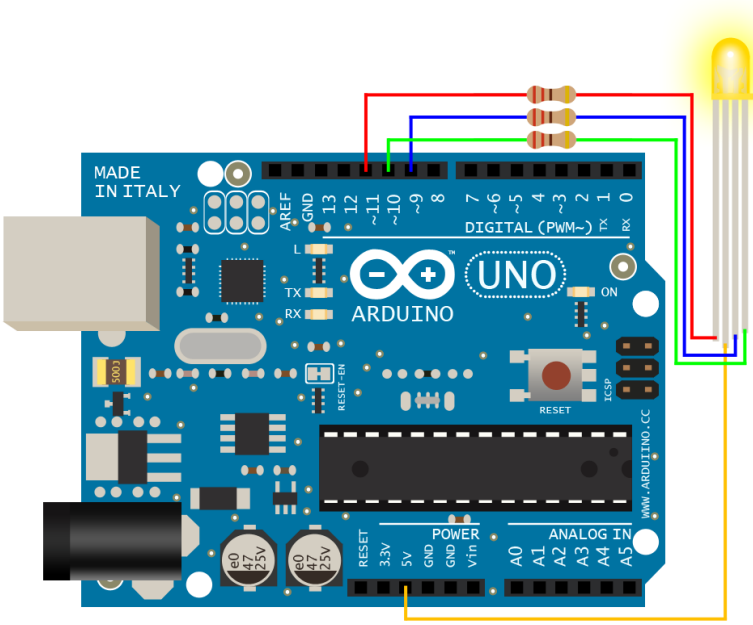


Рисунок 50-Схема подключения RGB-светодиода

Как мы знаем, произвольный цвет может быть получен смешением трех основных — красного, зеленого и синего. Светодиоды этих основных цветов совмещены в 5-миллиметровой колбе RGB-светодиода. У них может быть общий катод и различные аноды или общий анод и различные катоды. В первом случае, варьируя напряжение на анодах, а во втором — на катодах, мы сможем получить произвольный цветовой оттенок. Для плавного изменения напряжения, как мы уже знаем, надо использовать ШИМ-пины, например, 9, 10 и 11.

Выводы RGB-светодиода слева направо: красный, общий анод, зелёный, синий

Указанная в проекте схема сборки предполагает, что RGB-светодиод имеет общий анод. Анод подключается к питанию, а катоды светодиода — к пинам 9, 10 и 11. Особенность такого подключения заключается в том, что, чем ниже напряжение на пине, тем ярче горит соответствующий светодиод. Если напряжение максимальное (255), то светодиод соответствующего цвета не горит.

ШИМ-пины Ардуино позволяют управлять не только двумя значениями выходного напряжения (HIGH и LOW), но и промежуточными — в диапазоне от 0 до 255. Это нам и позволит получать разнообразные оттенки цветов. Программной особенностью проекта является то, что мы научимся использовать параметр во вспомогательной подпрограмме. Это значительно сократит длину нашего кода.Действительно, для того, чтобы напряжение плавно менялось на 9-м, 10 или 11 пине, нужно выполнить одни и те же действия, независимо от номера пина. Например, для того, чтобы красный цвет плавно угасал, надо написать следующие строки кода (в предположении, что красный цвет подключен к пину RED):

for (val = 0; val <=255; val++) // Вспомните, что запись val++ равносильна val = val + 1

```
  {
        analogWrite(RED, val); delay (20);
      }
```

Для зеленого цвета — то же самое, но заменив RED на GREEN. И для синего то же самое. Поэтому этот код надо оформить отдельной подпрограммой, указав, что у нее будет параметр (целого типа int), значение которого мы будем менять при вызове этой подпрограммы:

```
void LightDown (int pin) // Постепенно гасим цвет на пине pin
  {
    for (val = 0; val <=255; val++)
    {
       analogWrite(pin, val); delay (20);
    }
}
```

То же самое надо сделать для фрагмента кода, который усиливает свечение какого-либо цвета. Код программы «Все цвета радуги на RGB-светодиоде»:

Основные цвета: красный, зеленый, синий. Основные «смешанные» цвета:

красный + зеленый = жёлтый зеленый + синий = сине-зелёный

синий + красный = фиолетовый красный + синий + зеленый = белый

Скетч выполняет переходы:

красный => жёлтый => зеленый => сине-зеленый => синий => фиолетовый => красный

```
// объявление переменных
int RED = 9;              // Красный цвет - на 11-м пине
int GREEN = 10;      // Зеленый - на 10-м
int BLUE = 11;        // Синий - на 9
int val;                    // Рабочая переменная. Используем для счетчика цикла
void setup()
{
 pinMode( RED, OUTPUT);
 pinMode( GREEN, OUTPUT);
```

```
  pinMode( BLUE, OUTPUT);
  analogWrite(RED, 0);              // Красный горит
  analogWrite(RED, 0);              // Красный горит
  analogWrite(GREEN, ??? );   // Зеленый не горит
  analogWrite( ??? , 255);        // Синий не горит
}
void LightDown (int pin) // Постепенно гасим цвет на пине pin
 {
    for (val = 0; val <=255; val++)
     {
        analogWrite(pin, val); delay (20);
    }
 }
void LightUp (int pin) // Постепенно усиливаем цвет на пине pin
  {
     … ??? …
  }
void loop()
  {
    LightUp (GREEN); delay(100); // Плавный переход к желтому цвету
    LightDown (RED); delay(100); // Плавный переход к зеленому цвету
    … ??? …    // Плавный переход к сине-зеленому цвету
    … ??? …    // Плавный переход к синему цвету
    … ??? …    // Плавный переход к фиолетовому цвету
    … ??? …    // Плавный переход к красному цвету
  }
```

Самостоятельная работа. С помощью RGB-светодиода построить лампу, плавно переливающуюся всеми цветами радуги. Напишите алгоритм построения лампы.

2.Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.

2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта

2.5 Собрать схему **для управления цветом RGB-светодиода для проекта «Все цвета радуги»** в Ардуино UNO на монтажной плате, подключите к питанию согласно схемы и полярности напряжения.

2.6 Разработайте код программы согласно варианта, указанного в задании и Приложении Е.

2.7 Загрузите разработанный код в среду разработки Arduino IDE.

2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.11родемонстрируйте преподавателю результат выполненной работы.

3.Вывод о проделанной работе.

4. Контрольные вопросы

1. Назовите ошибки, какие сделал программист при инициализации контакта ledPin, настроенного на вывод PinMode(OutPut, LedPin):
а) неправильное использование регистра при вводе команды pinMode;
б) неправильный порядок параметров в команде;
в) Неправильное использование регистра при вводе имени пина;
г) неправильное использование регистра при вводе значения OUTPUT.
2. Выберите правильное назначение функции loop():
а) loop() выполняется один раз для инициализации (начальных установок) задействованных контактов (пинов);
б) loop() - цикл, который выполняется столько раз, сколько указано в качестве значения параметра функции;
в) loop() - бесконечный цикл, который можно остановить только отключением питания на платформе.
3.Поясните, верно ли утверждение: программу для Arduino можно писать в любом регистре - строчные и заглавные буквы не различаются.

4. Определите **к**оличество уровней сигнала, которые позволяет использовать широтно-импульсная модуляция (ШИМ): 64, 128, 256 и 1024.

Приложение Е

(Варианты заданий)

Вариант 1: С помощью RGB-светодиода разработайте вариант включения цветовых переходов синий, красный, зеленый, желтый.Должны загореться по очереди, а затем гореть вместе.

Вариант 2: С помощью RGB-светодиода реализуйте на базе разработанной схемы программу «проблесковые маячки». Светодиоды должны попеременно мигать с интервалом 3 сек.

Вариант 3: С помощью RGB-светодиода построить лампу, плавно переливающуюся всеми цветами радуги.

Лабораторная работа № 7 Управление работой светодиодов с помощью кнопки в Ардуино UNO.

Цель работы: написать код программы для управления работой светодиодов с помощью кнопки на базе микроконтроллера Arduino UNO.

Содержание работы:

- собрать схему подключения светодиодов к макетной плате согласно задания;
- разработать код программы для заданного проекта;
- загрузить код программы в Ардуино, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в Ардуино на исполнение;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1. Краткие теоретические сведения

1.1 Кнопка

Кнопка (или кнопочный переключатель) – самый простой и доступный из всех видов датчиков. Нажав на нее, вы подаете контроллеру сигнал, который затем приводит к каким-то действиям: включаются светодиоды, издаются звуки, запускаются моторы. В своей жизни мы часто встречаемся с разными выключателями и хорошо знакомы с этим устройством. Что такое кнопка? По сути, это достаточно простое устройство, замыкающее и размыкающее электрическую сеть. Выполнять это замыкание/размыкание можно в разных режимах, при этом фиксировать или не фиксировать свое положение. Соответственно, все кнопки можно поделить на две большие группы:

- кнопки переключатели с фиксацией. Они возвращаются в исходное состояние после того, как их отпустили. В зависимости от начального состояния разделяют на нормально-замкнутые и нормально-разомкнутые кнопки;
- кнопки без фиксации (тактовые кнопки). Они фиксируются и остаются в том положении, в котором их оставили.

1.2 Включение светодиода по нажатии кнопки

Когда кнопка отключена, вход D2 будет подтянут к «земле» через резистор номиналом 10 кОм, который будет ограничивать поток тока, и на входном контакте будет установлено значение напряжения LOW. При нажатии на кнопку входной контакт напрямую связан с 5 В. Большая часть тока будет протекать по пути наименьшего сопротивления через замкнутую кнопку, и на входе генерируется уровень HIGH. При нажатии на кнопку включаем светодиод, при отпускании – гасим.

Будем включать светодиод по нажатии кнопки и выключать по отпускании кнопки. Рассмотрим понятие дребезга и программные методы егоустранения.В данном эксперименте мы будем использовать контакт D2 Arduino в качестве входа. Это позволяет подключить к нему кнопку для взаимодействия с проектом в режиме реального времени. При использовании Arduino в качестве входов используют pull-up- и pulldown-резисторы, чтобы вход Arduino не находился в «подвешенном» состоянии (в этом состоянии он будет собирать внешние наводки и принимать произвольные значения), а имел заранее известное состояние (0 или 1). Резисторы pull-up подтягивают вход к питанию +5 В, pull-down-резисторы подтягивают вход к GND. Кроме этого, pull-up- и pull-down-резисторы гарантируют, что кнопка не создаст короткого замыкания между +5 В и землей при нажатии. В нашем эксперименте для подключения кнопки мы будем использовать pulldown-резистор. Схема подключения представленана рисунке 51.

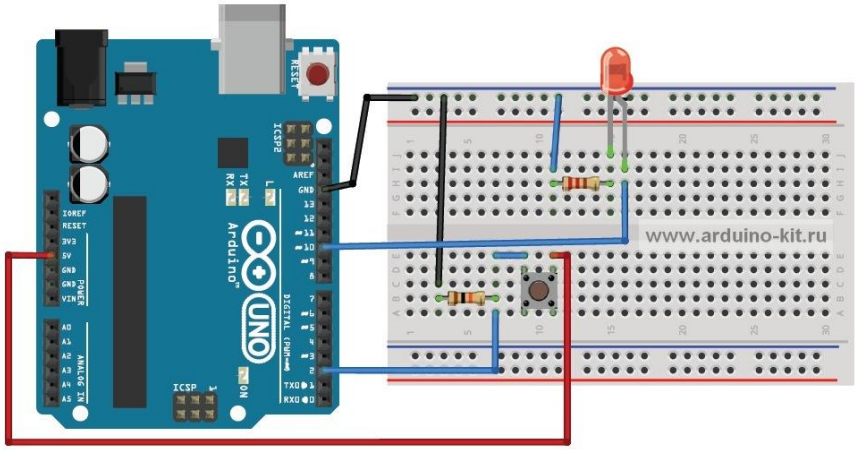


Рисунок 51- Схема подключения кнопки и светодиода

Когда кнопка отключена, вход D2 будет подтянут к «земле» через резистор номиналом 10 кОм, который будет ограничивать поток тока, и на входном контакте будет установлено значение напряжения LOW. При нажатии на кнопку входной контакт напрямую связан с 5 В. Большая часть тока будет протекать по пути наименьшего сопротивления через замкнутую кнопку, и на входе генерируется уровень HIGH. При нажатии на кнопку включаем светодиод, при отпускании – гасим. Код данного скетча приведен ниже:

Приведем код данного проекта:

```
const int LED=10; // вывод для подключения светодиода 10 (D10)

void setup()

{
```

```
// Конфигурируем вывод подключения светодиода как выход (OUTPUT)
pinMode(LED, OUTPUT);
}
void loop()
{
// включаем светодиод, подавая на вывод 1 (HIGH)
digitalWrite(LED,HIGH);
// пауза 1 сек (1000 мс)
delay(1000);
// выключаем светодиод, подавая на вывод 0 (LOW)
digitalWrite(LED,LOW);
// пауза 1 сек (1000 мс)
delay(1000);
}
```

1.3 Порядок подключения светодиода на макетной плате:

1. Длинную ножку светодиода (анод) подключаем к цифровому выводу D10 Arduino, другую (катод) – через резистор 220 Ом к выводу GND (см. рисунок 67).
2. Один вход кнопки подключаем к +5 В, другой – через резистор 10 кОм к GND, выход кнопки подключаем к входу D2 Arduino.
3. Загружаем в плату Arduino скетч.
4. При нажатии на кнопку светодиод должен гореть, при отпускании – затухнуть.

1.3.1 Переключение состояния светодиодов. Усложним задачу – будем переключать состояние светодиода (включен/выключен) при каждом нажатии кнопки. Загрузим на плату Arduino скетч.

```
const int LED=10; // Контакт 10 для подключения светодиода
const int BUTTON=2; // Контакт 2 для подключения кнопки
int tekButton = LOW; // Переменная для сохранения текущего состояния кнопки
int prevButton = LOW; // Переменная для сохранения предыдущего состояния
// к нопки
boolean ledOn = false; // Текущее состояние светодиода (включен/выключен)
void setup()
{
// Сконфигурировать контакт светодиода как выход
pinMode (LED, OUTPUT);
// Сконфигурировать контакт кнопки как вход
pinMode (BUTTON, INPUT);
```

```
}
void loop()
{
tekButton=digitalRead(BUTTON);
if (tekButton == HIGH && prevButton == LOW)
{
// нажатие кнопки – изменить состояние светодиода
ledOn=!ledOn;
digitalWrite(LED, ledOn);
}
prevButton=tekButton;
}
```

При нажатии кнопки светодиод должен изменять свое состояние. Но это будет происходить не всегда. Виной тому – дребезг кнопок.Кнопки представляют из себя механические устройства с системой пружинного контакта. Когда вы нажимаете на кнопку вниз, сигнал не просто меняется от низкого до высокого, он в течение нескольких миллисекунд меняет значение от одного до другого, прежде чем контакты плотно соприкоснутся и установится значение HIGH.Микроконтроллер зафиксирует все эти нажатия, потому что дребезг неотличим от настоящего нажатия на кнопку.Устранить влияние дребезга можно программно. Алгоритм следующий:

1. Сохраняем предыдущее состояние кнопки и текущее состояние кнопки (при инициализации LOW).
2. Считываем текущее состояние кнопки.
3. Если текущее состояние кнопки отличается от предыдущего состояния кнопки, ждем 5 мс, потому что кнопка, возможно, изменила состояние.
4. После 5 мс считываем состояние кнопки и используем его в качестве текущего состояния кнопки.
5. Если предыдущее состояние кнопки было LOW, а текущее состояние кнопки HIGH, переключаем состояние светодиода.
6. Устанавливаем предыдущее состояние кнопки для текущего состояния кнопки.
7. Возврат к шагу 2. Добавляем к нашему скетчу подпрограмму устранения дребезга.

Получаем следующий код:

```
const int LED=10; // Контакт 10 для подключения светодиода
const int BUTTON=2; // Контакт 2 для подключения кнопки
int tekButton = LOW; // Переменная для сохранения текущего состояния кнопки
int prevButton = LOW; // Переменная для сохранения предыдущего состояния
// к нопки
boolean ledOn = false; // Текущее состояние светодиода (включен/выключен)
```

```
void setup()
{
// Сконфигурировать контакт светодиода как выход
pinMode (LED, OUTPUT);
// Сконфигурировать контакт кнопки как вход
pinMode (BUTTON, INPUT);
}
// Функция сглаживания дребезга. Принимает в качестве
// аргумента предыдущее состояние кнопки и выдает фактическое.
boolean debounce(boolean last)
{
boolean current = digitalRead(BUTTON); // Считать состояние кнопки,
if (last != current) // если изменилось...
{
d elay(5); // ж дем 5 м с
current = digitalRead(BUTTON); // считываем состояние кнопки
return current; // возвращаем состояние кнопки
}
}
void loop()
{
tekButton = debounce(prevButton);
if (prevButton == LOW && tekButton == HIGH) // если нажатие...
{
ledOn = !ledOn; // инвертировать значение состояния светодиода
}
prevButton = tekButton;
digitalWrite(LED, ledOn); // изменить статус состояния светодиода
}
```

Загружаем скетч в плату Arduino и проверяем работу. Теперь каждое нажатие кнопки должно приводить к изменению состояния светодиода.

1.4 Управление яркостью светодиода

Теперь светодиод подключен к 11 пину Ардуино, которой умеет делать ШИМ. И нам пришлось добавить токоограничивающий резистор на 220 Ом перед светодиодом, что бы его не сжечь. Это необходимо потому, что светодиоды работают при напряжении 3.3 В, а пин Ардуино отдает 5 В. Разработаем скетч для управления яркостью светодиода:

```
// переменные с пинами подключенных устройств
int switchPin = 8;
int ledPin = 11;
 // переменные для хранения состояния кнопки и светодиода
boolean lastButton = LOW;
boolean currentButton = LOW;
int ledLevel = 0;
 void setup() {
pinMode(switchPin, INPUT);
pinMode(ledPin, OUTPUT);
}
 // функция для подавления дребезга
boolean debounse(boolean last) {
boolean current = digitalRead(switchPin);
if(last != current) {
delay(5);
current = digitalRead(switchPin);
}
return current;
}
 void loop() {
currentButton = debounse(lastButton);
if(lastButton == LOW && currentButton == HIGH) {
ledLevel = ledLevel + 51;
}
lastButton = currentButton;

if(ledLevel > 255) ledLevel = 0;
analogWrite(ledPin, ledLevel);
```

}

В этом примере мы изменили значение переменной ledPin на 11. Так же добавили переменную для хранения уровня ШИМ ledLevel. При нажатии на кнопку будем увеличивать эту переменную. Функция debounce() осталась без изменений. В цикле мы теперь используем функцию analogWrite().

2. Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта

2.5 Собрать схему для управления работой светодиодов с помощью кнопки на монтажной плате, подключить к питанию согласно схемы и полярности напряжения.

2.6 Разработайте код программы согласно варианта, указанного в задании и Приложении F.

2.7 Загрузите разработанный код в среду разработки Arduino IDE.

2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.11 Продемонстрируйте преподавателю результат выполненной работы.

3.Вывод о проделанной работе.

4. Контрольные вопросы

1. Поясните, какова разрядность встроенного в контроллер Arduino АЦП?
2. Объясните, в каком диапазоне изменяется напряжение на выходе потенциометра?
3. Объясните, в каком диапазоне формируется результат преобразования на выходе АЦП?
4. Расскажите, в каком диапазоне выводится значение напряжения на дисплей ПК и почему?

5. Поясните, на какую величину должно измениться напряжение на входе АЦП, чтобы результат преобразования на выходе АЦП изменился на единицу младшего разряда?

Приложение F

(Варианты заданий)

Вариант 1: Выполните последовательное включение 5 светодиодов одной кнопкой. Затем также этой же кнопкой осуществите последовательное выключение этих же светодиодов.

Вариант 2: Выполните последовательное включение 6 светодиодов одной кнопкой. Затем также этой же кнопкой осуществите последовательное выключение этих же светодиодов.

Вариант 3: Выполните последовательное включение 7 светодиодов одной кнопкой. Затем также этой же кнопкой осуществите последовательное выключение этих же светодиодов.

Вариант 4: Выполните последовательное включение 8 светодиодов одной кнопкой. Затем также этой же кнопкой осуществите последовательное выключение этих же светодиодов.

Лабораторная работа № 8 Разработка кода для вывода названия кнопки на экран ЖК- дисплея LCD Keypad Shild Arduino UNO.

Цель работы: разработка кода для вывода названия кнопки на экран ЖК- дисплея LCD Keypad Shild Arduino UNO.

Содержание работы:

- подключить к плате Arduino UNO LCD Keypad Shield с помощью проводов. Подключите плату Arduino UNO к ПК через USB-порт;
- разработать код программы «По нажатию встроенных на шильде кнопок *(SELECT, LEFT, UP, DOWN, RIGHT)* вывести название соответствующей кнопки в Монитор порта и на экран дисплея»;
- выполнить компиляцию кода, если результат положительный, то загрузить код программы в память контроллера Arduino UNO, переслав её из ПК;
- загрузить код программы, прошедший компиляцию, в контроллер;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет**;**
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты**;**
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1 Краткие теоретические сведения

Индикация осуществляется с помощью ЖК-дисплея 1602, управление – через встроенные кнопки. Есть возможность регулировки яркости дисплея прямо на плате с помощью подстроечного резистора. Плата снабжена разъемами, в которые могут быть подключены другие устройства, например, датчики. Для работы с экраном используются **цифровые пины 4 – 10**, для определения нажатия кнопок — только один **аналоговый пин A0**. Свободными являются **цифровые пины 0-3, 11-13** и **аналоговые пины A1-A5**, как представлено на рисунке 52.

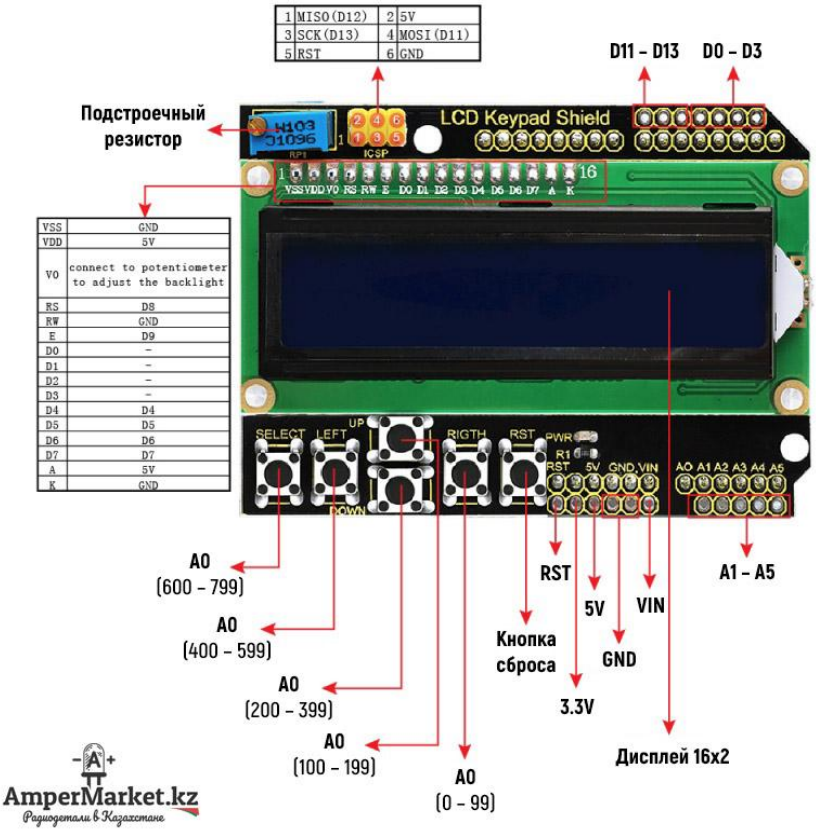


Рисунок 52- Встроенные кнопки для управления индикации

1.1Скетч для освоения работы LCD Keypad Shield

По нажатию встроенных на шилд кнопок *(SELECT, LEFT, UP, DOWN, RIGHT)* выведем название соответствующей кнопки в Монитор порта и на экран дисплея. С помощью подстроечного резистора можно настроить контрастность дисплея. Перед загрузкой скетча скачайте библиотеку LiquidCrystal. Рассмотрим простейший скетч для анализа работы шилда. Каждая строчка в скетче подробна расписана после двойного "//"

```
#include  <LiquidCrystal.h> //подключаем библиотеку
LiquidCrystal lcd(8, 9, 4, 5, 6, 7 );
int button;  //вводим числовые значения для кнопок
const int BUTTON_NONE   = 0; //присваиваем постоянное значение для BUTTON_NONE
const int BUTTON_RIGHT  = 1; //присваиваем постоянное значение для BUTTON_RIGHT
const int BUTTON_UP     = 2; //присваиваем постоянное значение для BUTTON_UP
const int BUTTON_DOWN   = 3; //присваиваем постоянное значение для BUTTON_DOWN
const int BUTTON_LEFT   = 4; //присваиваем постоянное значение для BUTTON_LEFT
const int BUTTON_SELECT = 5; //присваиваем постоянное значение для BUTTON_SELECT
int getPressedButton() //инициализация переменной
{
  int buttonValue = analogRead(0); // считываем значения с аналогового входа
```

```
  if (buttonValue < 100) { //если при нажатии кнопки значение меньше 100
    return BUTTON_RIGHT;   // выводим значение BUTTON_RIGHT
  }
  else if (buttonValue < 200) { //если при нажатии кнопки значение меньше 200
    return BUTTON_UP; // выводим значение BUTTON_UP
  }
  else if (buttonValue < 400){ //если при нажатии кнопки значение меньше 400
    return BUTTON_DOWN; // выводим значение BUTTON_DOWN
  }
  else if (buttonValue < 600){ //если при нажатии кнопки значение меньше 600
    return BUTTON_LEFT; // выводим значение BUTTON_LEFT
  }
  else if (buttonValue < 800){ //если при нажатии кнопки значение меньше 800
    return BUTTON_SELECT; // выводим значение BUTTON_SELECT
  }
  return BUTTON_NONE; //иначе, выводим значение BUTTON_NONE
}
void setup()
{
  lcd.begin(16, 2); //Инициализируем дисплей: 2 строки по 16 символов
  lcd.print("www.helpduino.ru");  //Выводи надпись www.helpduino.ru
}
void loop()
{
  button = getPressedButton();
  //Присваиваем занчение переменной getPressedButton к переменной button
  switch (button) //перебираем значения в цикле
  {
  case BUTTON_RIGHT: // при нажатии кнопки со значением BUTTON_RIGHT
    lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
    lcd.print("                "); //стираем текст дисплея
    lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
    lcd.print("BUTTON: RIGHT");//выводим надпись BUTTON: RIGHT на экран
    break; //переходим к следующему значению цикла
  case BUTTON_LEFT: // при нажатии кнопки со значением BUTTON_LEFT
     lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
    lcd.print("                "); //стираем текст дисплея
    lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
```

```
    lcd.print("BUTTON: LEFT"); //выводим надпись BUTTON: LEFT на экран
    break; //переходим к следующему значению цикла
  case BUTTON_UP: // при нажатии кнопки со значением BUTTON_UP
     lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
    lcd.print("                "); //стираем текст дисплея
    lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
    lcd.print("BUTTON: UP"); //выводим надпись BUTTON: UP на экран
    break; //переходим к следующему значению цикла
  case BUTTON_DOWN: // при нажатии кнопки со значением BUTTON_DOWN
     lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
    lcd.print("                "); //стираем текст дисплея
    lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
    lcd.print("BUTTON: DOWN"); //выводим надпись BUTTON: DOWN на экран
    break; //переходим к следующему значению цикла
  case BUTTON_SELECT: // при нажатии кнопки со значением BUTTON_SELECT
     lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
    lcd.print("                "); //стираем текст дисплея
    lcd.setCursor(0, 0); //устанавливаем курсор на 1 строку 1 столба
    lcd.print("BUTTON: SELECT"); //выводим надпись BUTTON: SELECT на экран
    break; //переходим к следующему значению цикла
  }
}
```

1.1.1 Пояснения к скетчу

Когда вы выгрузите скетч в программу и загрузите ее к вашему Arduino, то у вас откроется стартовая страница. При нажатии кнопки на экране шилда появится название кнопки, на которую вы нажали. Этот скетч помогает разобраться в управлении самого LCD Keypad Shield и понять принцип его работы.

2.Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта. Установите библиотеку LiquidCrystal.

2.5 Собрать схему проекта на монтажной плате, подключить к питанию согласно схемы и полярности напряжения.

2.6 Подключите к плате Arduino UNO LCD Keypad Shield с помощью проводов.Подключите плату Arduino UNO к ПК через USB-порт.
2.7 Разработайте код программы «По нажатию встроенных на шильде кнопок *(SELECT, LEFT, UP, DOWN, RIGHT)* вывести название соответствующей кнопки в Монитор порта и на экран дисплея». С помощью подстроечного резистора можно настроить контрастность дисплея.

2.8 Разработайте код программы согласно варианта, указанного в задании и Приложении G.

2.9 Загрузите разработанный код в среду разработки Arduino IDE.

2.10 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.11 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.12 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.13 Продемонстрируйте преподавателю результат выполненной работы.

4.Контрольные вопросы

1. Опишите устройство и принцип работы LCD Keypad Shield.
2. Скажите, какое назначение у функции setup():
а) setup() выполняется один раз для инициализации (начальных установок) задействованных контактов (пинов;
б) setup() - цикл, который выполняется столько раз, сколько указано в качестве значения параметра функции;
в) setup() - бесконечный цикл, который можно остановить только отключением питания на платформе.
3. Объясните, как правильно написать команду, которая приостанавливает выполнение программы на 5 секунд:
а) delay(5);
б) delay(5000;
в) Delay(5);
г) Delay(5000);
д) delay(500).
4. Определите количество размещенных на платформе Arduino UNO цифровых контактов (ввод/вывод) из предложенных вариантов:5, 10, 14 или 20.

5.Назовите номинал напряжения питания LCD Keypad Shield и поясните, от какого источника он запитывается.

5.Вывод о проделанной работе.

Приложение G

(Варианты заданий)

Вариант 1: По нажатию встроенной на шилд кнопки ***SELECT*** вывести название этой кнопки в Монитор порта и на экран дисплея. С помощью подстроечного резистора можно настроить контрастность дисплея.

Вариант 2: По нажатию встроенной на шилд кнопки ***LEFT*** вывести название этой кнопки в Монитор порта и на экран дисплея. С помощью подстроечного резистора можно настроить контрастность дисплея.

Вариант 3: По нажатию встроенной на шилд кнопки ***UP*** вывести название этой кнопки в Монитор порта и на экран дисплея. С помощью подстроечного резистора можно настроить контрастность дисплея.

Вариант 4: По нажатию встроенной на шилд кнопки ***DOWN*** вывести название этой кнопки в Монитор порта и на экран дисплея. С помощью подстроечного резистора можно настроить контрастность дисплея.

Вариант 5: По нажатию встроенной на шилд кнопки ***RIGHT*** вывести название этой кнопки в Монитор порта и на экран дисплея. С помощью подстроечного резистора можно настроить контрастность дисплея.

Лабораторная работа № 9 Разработка кода для организации часов реального времени на ЖК- дисплее LCD Keypad Shild Arduino UNO.

Цель работы: разработка кода для организации часов реального времени на ЖК- дисплее LCD Keypad Shild Arduino UNO.

Содержание работы:

- подключить к плате Arduino UNO LCD Keypad Shield с помощью проводов.Подключите плату Arduino UNO к ПК через USB-порт;
- разработать код программыдля организации часов реального времени;
- выполнить компиляцию кода, если результат положительный, то загрузить код программы в память контроллера Arduino UNO, переслав её из ПК;

- загрузить код программы, прошедший компиляцию, в контроллер;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, часы реального времени(модуль DS1307), кнопки.

1 Краткие теоретические сведения

1.1 Модуль DS1307 для подсчета времени

Использовании модуля DS1307 зачастую очень оправдано, например, когда данные считываются редко, интервалом более недели, использовать собственные ресурсы контроллера, неоправданно или невозможно. Обеспечивание бесперебойное питание, например платы Arduino, на длительный срок дорого, даже при использовании батареи. Благодаря собственной памяти и автономностью, можно регистрировать события, (при автономном питании) например изменение температуры и так далее, данные сохраняются в памяти их можно считать из памяти модуля. Так что модуль DS1307 часто используют, когда контроллерам Arduino необходимо знать точное время, для запуска какого то события и так далее. DS1307 это небольшой модуль, предназначенный для подсчета времени. Собранный на базе микросхемы DS1307ZN с реализацией питания от литиевой батарейки (LIR2032), что позволяет работать автономно в течение длительного времени. Также на модуле, установлена энергонезависимая память EEPROM объемом 32 Кбайт

(AT24C32). Микросхема AT24C32 и DS1307ZN связаны обшей шиной интерфейсом I2C. Модуль DS1307 для подсчета времени представлен на рисунке 53.

Рисунок 53- Модуль DS1307 для подсчета времени

Технические параметры модуля следующие:

► напряжение питания: 5В;
► рабочая температура: – 40℃ … + 85℃;
► память: 56 байт (энергонезависимая);
► батарейка**:** LIR2032 (автоматическое определение источника питания);
► интерфейса: I2C;
► габариты: 28мм х 25мм х 8 мм.

Обмен данными с другими устройствами осуществляется по интерфейсу I2C с выводов SCL и SDA. Конденсаторы C1 и C2 необходимы для снижения помех по линию питания. Чтобы обеспечить надлежащего уровня сигналов SCL и SDA установлены резисторы R2 и R3 (подтянуты к питанию). Для проверки работоспособности модуля, на вывод 7 микросхему DS1307Z, подается сигнал SQ, прямоугольной формы с частотой 1 Гц. Элементы R4, R5, R6, VD1 необходимы для подзарядку литиевой батарейки. Так же, на плате предусмотрено посадочное место (U1), для установки датчика температуры DS18B20 (при необходимости можно впаять его), считывать показания, можно с вывода DS, который подтянут к пиатнию, через резистор R1 сопротивлением 3.3 кОм. Принципиальную схему и назначение контактов можно посмотреть на рисунках ниже.

1.2 Подключение

Для подключения часы реального времени DS1307, необходимо впаять впаять штыревые разъемы в первую группу контактов. Далее, подключаем провода SCL (DS1307) к выводу 4 (Arduino UNO) и SDA (DS1307) к выводу 5 (Arduino UNO), осталось подключить питания VCC к +5V и GND к GND. Кстати, в различных платах Arduino вывода интерфейса I2C отличаются, назначение каждого можно посмотреть на рисунке 54.

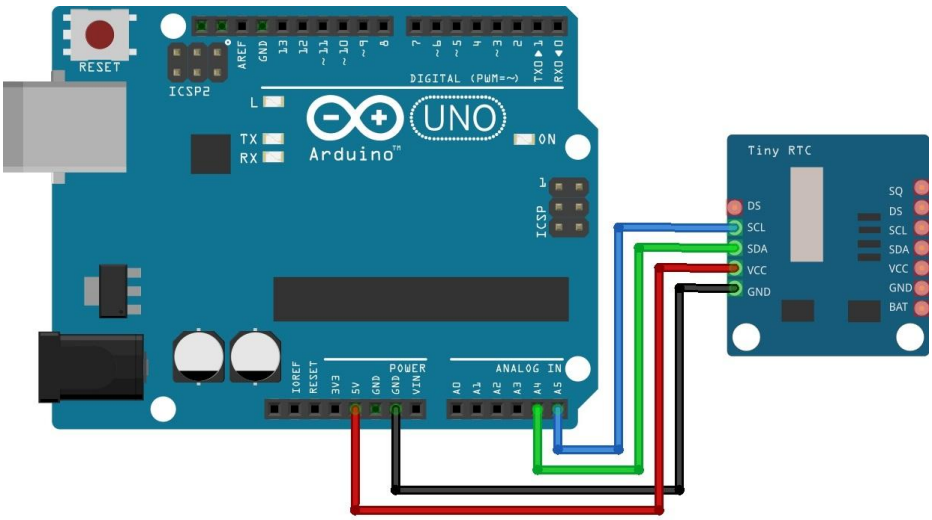


Рисунок 54-Подключение модуля DS1307 для подсчета времени к Ардуино

1.3 Установка времени DS1307

Первым делом, необходимо скачать и установить библиотеку «DS1307RTC» и «TimeLib» в среду разработки IDE Arduino, далее необходимо настроить время, открываем пример из библиотеки DS1307RTC «Файл» —> «Примеры» —> «DS1307RTC» —> «SetTime» или копируем код снизу.Для подключения RTC часов реального времени DS1302, DS1307, DS3231, была разработана универсальная библиотека.

Скачать библиотеку можно по ссылке : Универсальная библиотека для RTC DS1302, DS1307, DS3231 к Arduino.

Подключение DS1307 к Arduino:

GND	GND
VCC	+5V
SDA	A4
SCL	A5

Подключение DS1302 к Arduino:

RTC DS1302	**Arduino UNO**
GND	GND
VCC	+5V
RST	10 (Можно изменить на другие в скетче)
CLK	13 (Можно изменить на другие в скетче)
DAT	12 (Можно изменить на другие в скетче)

Подключение DS3231 к Arduino:

RTC DS3231	Arduino UNO
GND	GND
VCC	+5V
SDA	A4
SCL	A5

1.4 Программа

В зависимости от того какой модуль Вы подключаете, необходимо в программе указать

Для DS1307**:**

```
time.begin(RTC_DS1307);
```

Для DS1302**:**

```
time.begin(RTC_DS1302,10,13,12);
```

Для DS3231**:**

```
time.begin(RTC_DS3231);
```

Пример установки текущего времени в RTC модуль (DS1307):

```
#include <iarduino_RTC.h>
iarduino_RTC time(RTC_DS1307);
void setup() {
  delay(300);
  Serial.begin(9600);
  time.begin();
  time.settime(0,51,21,27,10,15,2);  // 0  сек, 51 мин, 21 час, 27, октября, 2015 года, вторник
}
void loop(){
  if(millis()%1000==0){ // если прошла 1 секунда
    Serial.println(time.gettime("d-m-Y, H:i:s, D")); // выводим время
    delay(1); // приостанавливаем на 1 мс, чтоб не выводить время несколько раз за 1мс
   }
}
```

Пример считывания текущего времени с RTC модуля (DS1307) и вывод в "Последовательный порт" **:**

```
#include <iarduino_RTC.h>
iarduino_RTC time(RTC_DS1307);
void setup() {
  delay(300);
  Serial.begin(9600);
  time.begin();
}
void loop(){
```

```
  if(millis()%1000==0){ // если прошла 1 секунда
    Serial.println(time.gettime("d-m-Y, H:i:s, D")); // выводим время
    delay(1); // приостанавливаем на 1 мс, чтоб не выводить время несколько раз за 1мс
  }
}
```

Преимущества библиотеки:

- библиотека имеет внутренние функции аппаратной обработки протоколов передачи данных I2C и SPI, а следовательно не требует подключения дополнительных библиотек, но и не конфликтует с ними, если таковые всё же подключены;

- библиотека имеет внутренние функции программой обработки протокола передачи данных 3-Wire;

- для инициализации модуля необходимо вызвать функцию begin с названием модуля;

- подключение модулей осуществляется к аппаратным выводам arduino используемой шины (за исключением 3-Wire);

- простота установки и чтения времени функциями settime и gettime;

функция settime может устанавливать дату и время, как полностью, так и частично (например только минуты, или только день, и т.д.);

функция gettime работает как функция date в php, возвращая строку со временем, но если её вызвать без параметра, то функция ничего не вернёт, а время можно прочитать из переменных в виде чисел;

- библиотека расширяемая, то есть для того, чтоб она работала с новым модулем, нужно указать параметры этого модуля в уже существующих массивах файла RTC.h (тип шины, частота шины в кГц, режимы работы, адреса регистров и т.д.), как всё это сделать, описано в файле extension.txt;

Таким образом добавив новый модуль в библиотеку, мы лишь увеличим область занимаемой динамической памяти на ~ 36 байт, при этом не затронув область памяти программ;

- при вызове функции begin, библиотека читает флаги регистров модуля и при необходимости устанавливает или сбрасывает их так, чтоб модуль мог работать от аккумуляторной батареи, а на программируемом выводе меандра (если таковой у модуля есть) установилась частота 1Гц, тогда этот вывод можно использовать в качестве внешнего посекундного прерывания;

- при работе с модулем DS1302 не нужны никакие резисторы на выводе GND (которые нужны для его работы с другими библиотеками этого модуля), это достигнуто тем, что для шины 3-Wire указана конкретная частота 10кГц, не зависимо от частоты CPU arduino;

- в библиотеке реализована еще одна не обязательная функция period, принимающая в качестве единственного аргумента - количество минут (от 1 до 255).

Если в течении указанного времени была вызвана функция gettime несколько раз, то запрос к модулю по шине будет отправлено только в первый раз, а ответом на все остальные запросы будет сумма времени последнего ответа модуля и времени прошедшего с этого ответа.Функцию period достаточно вызвать один раз.Ниже приведен скетч для модуля:

```
}//   ОПИСАНИЯ ПАРАМЕТРОВ ФУНКЦИЙ:
//  Подключение библиотеки:
//  #include <iarduino_RTC.h>
//  iarduino_RTC time(название модуля [, вывод SS/RST [, вывод CLK [, вывод DAT]]]);
//          если модуль работает на шине I2C или SPI, то достаточно указать 1 параметр, например: iarduino_RTC time(RTC_DS3231);
//          если модуль работает на шине SPI, а аппаратный вывод SS занят, то номер назначенного вывода SS для модуля указывается вторым параметром, например: iarduino_RTC time(RTC_DS1305,22);
//      если модуль работает на трехпроводной шине, то указываются номера всех выводов, например: iarduino_RTC time(RTC_DS1302, 1, 2, 3); // RST, CLK, DAT
//  Для работы с модулями, в библиотеке реализованы 5 функции:
//      инициировать модуль  begin();
//       указать время          settime(секунды [, минуты [, часы [, день [, месяц [, год [, день недели]]]]]]);
//      получить время       gettime("строка с параметрами");
//       мигать времем            blinktime(0-не_мигать / 1-мигают_сек / 2-мигают_мин / 3-мигают_час / 4-мигают_дни / 5-мигают_мес / 6-мигает_год / 7-мигают_дни_недели / 8-мигает_полдень)
//      разгрузить шину      period (минуты);
//  Функция begin():
//          функция инициирует модуль: проверяет регистры модуля, запускает генератор модуля и т.д.
//  Функция settime(секунды [, минуты [, часы [, день [, месяц [, год [, день недели]]]]]]):
//      записывает время в модуль
//      год указывается без учёта века, в формате 0-99
//      часы указываются в 24-часовом формате, от 0 до 23
//      день недели указывается в виде числа от 0-воскресенье до 6-суббота
//          если предыдущий параметр надо оставить без изменений, то можно указать отрицательное или заведомо большее значение
//        пример: settime(-1, 10); установит 10 минут, а секунды, часы и дату, оставит без изменений
//       пример: settime(0, 5, 13); установит 13 часов, 5 минут, 0 секунд, а дату оставит без изменений
//      пример: settime(-1, -1, -1, 1, 10, 15); установит дату 01.10.2015 , а время и день недели оставит без изменений
//  Функция gettime("строка с параметрами"):
//       функция получает и выводит строку заменяя описанные ниже символы на текущее время
//      пример: gettime("d-m-Y, H:i:s, D"); ответит строкой "01-10-2015, 14:00:05, Thu"
```

```
//      пример: gettime("s");              ответит строкой "05"
//      указанные символы идентичны символам для функции date() в PHP
// s   секунды                  от      00    до       59  (два знака)
// i   минуты                   от      00    до       59  (два знака)
// h   часы в 12-часовом формате     от      01    до       12  (два знака)
// H   часы в 24-часовом формате     от      00    до       23  (два знака)
// d   день месяца              от      01    до       31  (два знака)
// w   день недели              от       0    до        6  (один знак: 0-воскресенье, 6-суббота)
// D   день недели наименование      от      Mon    до      Sun  (три знака: Mon Tue Wed Thu Fri Sat Sun)
// m   месяц                    от      01    до       12  (два знака)
// M   месяц наименование            от      Jan    до      Dec  (три знака: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec)
// Y   год                      от    2000    до     2099  (четыре знака)
// y   год                      от      00    до       99  (два знака)
// a   полдень                       am   или       pm  (два знака, в нижнем регистре)
// A   полдень                       AM   или       PM  (два знака, в верхнем регистре)
//      строка не должна превышать 50 символов
//  если требуется получить время в виде цифр, то можно вызвать функцию gettime() без параметра, после чего получить время из переменных
//      seconds  секунды     0-59
//      minutes  минуты      0-59
//      hours    часы        1-12
//      Hours    часы        0-23
//      midday   полдень     0-1 (0-am, 1-pm)
//      day      день месяца 1-31
//      weekday  день недели 0-6 (0-воскресенье, 6-суббота)
//      month    месяц       1-12
//      year     год         0-99
//  Функция blinktime(параметр):
//      указывает функции gettime мигать одним из параметров времени (заменять параметр пробелами)
//          функция может быть полезна, для отображения на дисплее, устанавливаемого параметра времени
//      функция получает один параметр в виде числа от 0 до 8
//  0   не мигать
//  1   мигают сек
//  2   мигают мин
//  3   мигают час
//  4   мигают дни
//  5   мигают мес
//  6   мигает год
//  7   мигают дни недели
//  8   мигает полдень
//  Функция period(минуты):
```

```
//      устанавливает минимальный период обращения к модулю в минутах (от 0 до 255)
//          функция может быть полезна, если шина сильно нагружена, на ней имеются несколько устройств
//       period(10); период 10 минут, означает что каждые 10 минут к модулю может быть отправлен только 1 запрос на получение времени
//          ответом на все остальные запросы будет результат последнего полученного от модуля времени + время прошедшее с этого запроса
```

Приведем пример скетча «Дата и время»:

```
#include <Wire.h>
const byte DS3231 = 0x68; // I2C адрес таймера DS3231
void setup() {
  Wire.begin();
  Serial.begin(9600);
}
void loop() {
  Wire.beginTransmission(DS3231); //Начинаем обмен с DS3231
  Wire.write(byte(0x00)); //Записываем адрес регистра, с которого начинаем чтение!!!
  Wire.endTransmission(); //Завершаем передачу
  byte t[7]; //Массив для хранения даты и времени
  int i = 0; //Индекс текущего элемента массива
  Wire.beginTransmission(DS3231); //Начинаем обмен с DS3231
  Wire.requestFrom(DS3231, 7); //Запрашиваем 7 байтов у DS3231
  while(Wire.available()) { //Пока есть данные от DS3231
    t[i] = Wire.read(); //Читаем 1 байт и сохраняем в массив t
    i++; //Инкрементируем индекс элемента массива
  }
  Wire.endTransmission(); //Завершаем обмен
    printDateTime(t); //Выводим дату и время
  delay(1000); //Пауза 1 секунда
}
//Разбирает считанный массив и выводит дату и время
void printDateTime(byte *arr) {
```

```
  if (arr[4]<10)
    Serial.print("0");//Дополнение нулями слева для выравнивания
    Serial.print(arr[4], HEX); //Выводим дату
    Serial.print(".");
  if (arr[5]<10)
    Serial.print("0");
    Serial.print(arr[5], HEX); //Выводим месяц
    Serial.print(".20"); //2000-ые годы подразумеваются
    Serial.print(arr[6], HEX); //Выводим год
    Serial.print(" ");
  if (arr[2]<10)
    Serial.print("0");
    Serial.print(arr[2], HEX); //Выводим час
    Serial.print(":");
  if (arr[1]<10)
    Serial.print("0");
    Serial.print(arr[1], HEX); //Выводим минуты
    Serial.print(":");
  if (arr[0]<10)
    Serial.print("0");
    Serial.println(arr[0], HEX);//Выводим секунды
}
```

2.Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта и установить библиотеку «DS1307RTC» и «TimeLib».

2.5 Собрать схему проекта **для организации часов реального времени на ЖК- дисплее LCD Keypad Shild** на монтажной плате, подключить к питанию согласно схемы и полярности напряжения. Для организации часов реального времени DS1307 необходимо вставить штыревые разъемы в первую группу контактов. Далее, подключаем провода SCL (DS1307) к выводу 4 (Arduino UNO) и SDA (DS1307) к выводу 5 (Arduino UNO), затем подключить выводы питания VCC к +5V и GND к GND.

2.6 Загрузите разработанный код в среду разработки Arduino IDE.

2.7 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.8 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.9 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.10 Продемонстрируйте преподавателю результат выполненной работы.

3.Контрольные вопросы

1. Определите количество размещенных на платформе Arduino UNO цифровых контактов (ввод/вывод) из предложенных вариантов:5, 10, 14 или 20?

2.Объясните, как правильно написать команду, которая приостанавливает выполнение программы на 5 секунд:
а) delay(5);
б) delay(5000;
в) Delay(5);
г) Delay(5000);
д) delay(500).

3.Назовите функции, которые должны присутствовать в скетче для Arduino: setup(), main(), loop(), function().
4. Поясните, верно ли утверждение: программу для Arduino можно писать в любом регистре - строчные и заглавные буквы не различаются.
5. Назовите библиотеку для RTC_DS1307 и способы ее установки на ПК.

4.Вывод о проделанной работе.

Лабораторная работа № 10 Измерение температуры датчиком DHT 11 с выводом на COM-порт и LCD Keypad SHILD Ардуино UNO.

Цель работы: написать код программы для одновременной работы шилда и датчика температуры и влажности DHT 11 Ардуино UNO.

Содержание работы:

- собрать схему подключения светодиодов к макетной плате согласно задания;
- разработать код программы;
- загрузить код программы в Ардуино, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в Ардуино на исполнение;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1.Краткие теоретические сведения

1.1 Как подключить датчик температуры и влажности DHT 11к Ардуино UNO

Самые частые измеряемые параметры в промышленности и быту — это температура и влажность. В быту измеряют в теплицах и в контурах отопления и горячего водоснабжения. Датчик DHT11 Ардуино прекрасно справляется со своими задачами и определяет более-менее точно температуру и влажность.

DHT11 — это маленький сенсор в небольшом пластиковом корпусе. На выходе сенсора находится цифровой сигнал, причем сразу два параметра и температура и влажность. Смысл общения с контроллером Ардуино заключается в следующем:

- ✓ микроконтроллер запрашивает показания и меняет сигнал с 0 на 1;
- ✓ датчик видит запрос, и отвечает ему, меняя битовый сигнал с 0 на 1;
- ✓ когда они договорились между собой, датчик выдаёт ему пакет данных в размере 5 байт(40 бит), при чем в двух первых байтах температура, в третьем и четвертом влажность. Пятый байт — контрольная сумма для исключения ошибок измерения.

Характеристики сенсора температуры и влажности DHT11:

- определение влажности в диапазоне 20-80%;

- определение температуры от 0°C до +50°C;
- частота опроса 1 раз в секунду.

Недостаток сенсора в том, что он не обладает высокой точностью и быстродействием. Большой плюс — это цена. .В составе сенсора находится ёмкостной датчик для измерения влажности и термистор для измерения температуры. Все показания снимает чип АЦП и выдает цифровой сигнал.Датчики зачастую изготавливают в виде готовых шильдов. На выходе он имеет 3 пина:

- питание 5 В;
- сигнал (S);
- земля GND.

Сопротивление в 10 кОм ставить не нужно, так как оно уже впаяно в плату. Схема подключения датчика DHT11к Ардуино UNO представлена на рисунке 55

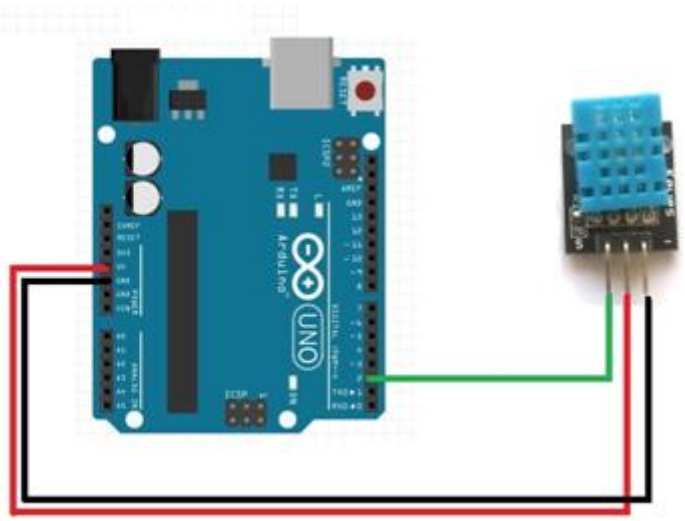


Рисунок 55- Схема подключения DHT11к Ардуино UNO представлена на рисунке 67

Необходимая бибилиотека для работы с датчиком температуры- **dht11**.

1.1 Вывод температуры и влажности на на LCD Keypad SHILD дисплей и COM-порт

Для подключения LCD Keypad SHILD дисплея используются контакты 4-9, а также питание 5V и GND. Соединив шилд с ардуиной с помощью перемычек - получили свободные контакты для подключения модуля датчика температуры и влажности, который мы также запитаем от 5V и GND и подключим к 3 цифровому порту ардуины. Получаем уже автономное устройство, которое с интервалом 5 сек обновлят информацию с датчика температуры и влажности и выводит на дисплей, как показано на рисунке 56.

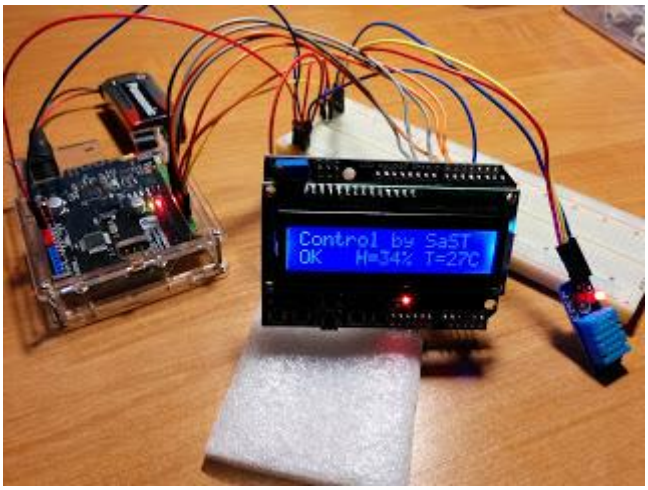


Рисунок 56-Вывод на LCD Keypad SHILD дисплей показаний температуры и влажности

Вместе с выводом на дисплей, информация с датчика передается на COM-порт Ардуино и может быть получена через монитор порта, как показано на рисунке 57.

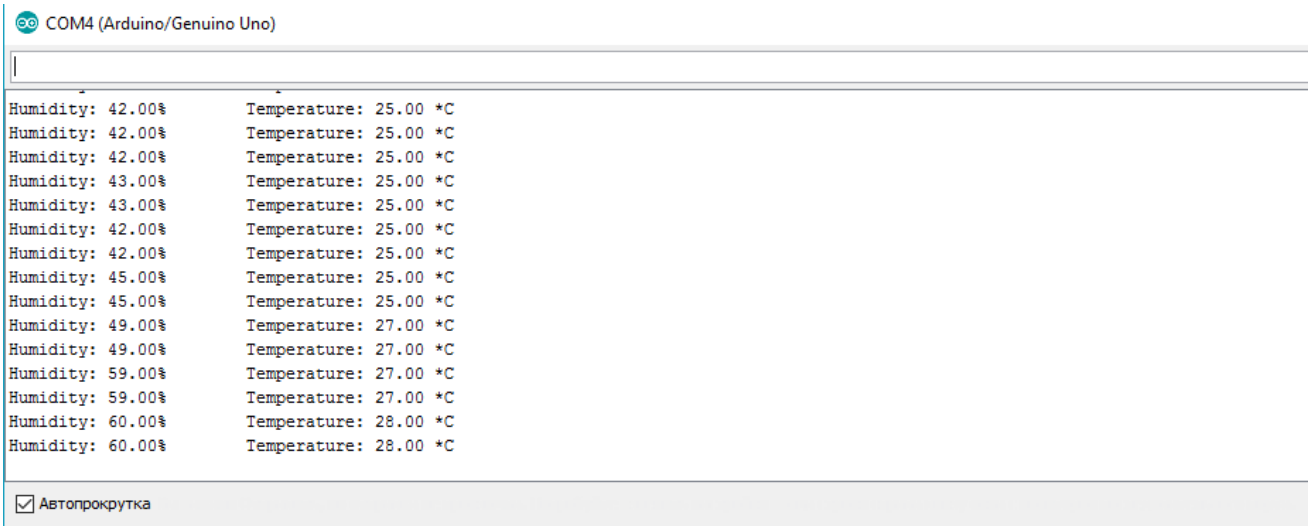


Рисунок 57- Вывод информации с датчика на экран дисплея

1.2 Скетч программы для работы с сенсором.

```
#include "DHT.h"

#define DHTPIN 2 // номер пина, к которому подсоединен датчик

// Раскомментируйте в соответствии с используемым датчиком

// Инициируем датчик
//DHT dht(DHTPIN, DHT22);

DHT dht(DHTPIN, DHT11);

void setup() {
```

```
Serial.begin(9600);

dht.begin();

}

void loop() {

// Задержка 2 секунды между измерениями

delay(1000);

//Считываем влажность

float h = dht.readHumidity();

// Считываем температуру

float t = dht.readTemperature();

// Проверка удачно прошло ли считывание.

if (isnan(h) || isnan(t)) {

Serial.println("Не удается считать показания");

} else {
Serial.print ("Humidity: ");
Serial.print (h);
Serial.print ("%\t");
Serial.print ("Temperature: ");
Serial.print (t);
Serial.println (" *C");
 }
}
```

1.3 Скетч вывода значений температуры датчиком DHT 11 на LCD Keypad SHILD

Скрестив скетч по работе с шилдом и модулем датчика температуры и влажности, получим код который информацию с датчика выводит на экран дисплея, а также его состояние (наличие ошибок):

```
#include <dht11.h>

dht11 DHT;

#define DHT11_PIN 2

#include <LiquidCrystal.h>

LiquidCrystal  lcd (8,9,4,5,6,7);

int ft;

int fh;

int SVET = 11;

void setup(){
```

```
  pinMode( SVET, OUTPUT);
  digitalWrite( SVET, LOW);
  Serial.begin(9600);
    lcd.begin(16, 2);
  //optionally, now set up our application-specific display settings, overriding whatever the lcd did in lcd.init()
  //lcd.commandWrite(0x0F);//cursor on, display on, blink on.  (nasty!)
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.write("LIBRARY VERSION:");
  lcd.setCursor(0,2);
  lcd.write(DHT11LIB_VERSION);
  Serial.println(DHT11LIB_VERSION);
  Serial.println();
  delay(5000);
  Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
  lcd.clear();
  ft=1;
  fh=1;
  lcd.setCursor(0,2);
  delay(1000);
  lcd.setCursor(0,2);
}
void loop(){
  int chk;
  Serial.print("DHT11, \t");
  chk = DHT.read(DHT11_PIN);    // READ DATA
  switch (chk){
    case DHTLIB_OK:
                lcd.setCursor(0,2);  //line=2, x=0
          lcd.write("OK");
                Serial.print("OK,\t");
```

```
            break;
  case DHTLIB_ERROR_CHECKSUM:
            lcd.setCursor(0,2);  //line=2, x=0
        lcd.write("ERR");
            Serial.print("Checksum error,\t");
            break;
  case DHTLIB_ERROR_TIMEOUT:
            lcd.setCursor(0,2);  //line=2, x=0
        lcd.write("TOUT");
            Serial.print("Time out error,\t");
            break;
  default:
            lcd.setCursor(0,2);  //line=2, x=0
        lcd.write("UERR");
            Serial.print("Unknown error,\t");
            break;
 }
// DISPLAT DATA
if ((ft==1)&&(fh==1)){
    ft=DHT.temperature;
    fh=DHT.humidity;
 }
 lcd.setCursor(0,0);
 lcd.write("H=");
 lcd.print(fh);
 lcd.setCursor(5,0);
 lcd.write("% T=");
 lcd.print(ft);
 lcd.write("C");

 lcd.setCursor(0,2);
```

```
  Serial.print("H=");
  Serial.print(DHT.humidity,1);
  Serial.print(",\t");
  Serial.print("T=");
  Serial.println(DHT.temperature,1);
  lcd.setCursor(5,2);  //line=2, x=0
  lcd.write("H=");
  lcd.print(DHT.humidity,1);
  lcd.write("% T=");
  lcd.print(DHT.temperature,1);
  lcd.write("C");
  lcd.setCursor(0,2);
  lcd.write("            ");

  if((DHT.humidity>fh)||(DHT.temperature>ft)){
    digitalWrite(SVET,HIGH);
  } else {
    digitalWrite(SVET,LOW);
  }
}
```

2.Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта- **dht11**

2.5 Собрать схему проекта одновременного вывода значений температуры и влажности на LCD Keypad SHILD и COM–порт ПК на монтажной плате, подключить к питанию согласно схемы и полярности напряжения.
2.6 Разработайте код программы согласно проекта, указанного в п.2.5

2.7 Загрузите разработанный код в среду разработки Arduino IDE.

2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.11 Продемонстрируйте преподавателю результат выполненной работы.

3. Контрольные вопросы

1.Определите количество размещенных на платформе Arduino UNO контактов аналогового ввода из предложенных вариантов :6, 10, 15 или 20?
2. Определите количество размещенных на платформе Arduino UNO цифровых контактов (ввод/вывод) из предложенных вариантов:5, 10, 14 или 20?
3.Назовите ошибки какие сделал программист, написав команду для подачи высокого напряжения на встроенный светодиод, подключенный к 13 контакту (пину): DigitalWrite(High): а) неправильное использование регистра при вводе команды digitalWrite;
б) неправильное количество параметров в команде;
в) неправильное использование регистра в параметре HIGH;
г) неправильное использование значение HIGH вместо LOW.
4. Назовите ошибки, какие сделал программист при инициализации контакта ledPin, настроенного на вывод PinMode(OutPut, LedPin):
а) неправильное использование регистра при вводе команды pinMode;
б) неправильный порядок параметров в команде;
в) Неправильное использование регистра при вводе имени пина;
г) неправильное использование регистра при вводе значения OUTPUT.
5. Назовите библиотеку для DHT 11 и LCD Keypad SHILD и способ установки их на ПК.

4.Вывод о проделанной работе.

Лабораторная работа №11 Программирование режимов включения светодиодов с помощью датчика освещенности

Цель работы: написать код программы для включения в работу светодиодов с помощью датчика освещенности по заданному проекту.

Содержание работы:

- собрать схему подключения светодиода к макетной плате по заданным условиям;
- подключить Ардуино к ПК;
- подключить фоторезистор и светодиоды к макетной плате;
- написать скетч мигания светодиода в зависимости от показаний фоторезистора и устанавливаемого порогового значения;
- загрузить код программы, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в контроллер;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1 Краткие теоретические сведения

1.1 Датчики освещенности

Фоторезисторы, которые обычно идут в наборах с платами Arduino, как правило изменяют сопротивление от 200 kOm (полная темнота) до 1 kOm (яркость 10 люкс). Экспериментируя с датчиком, можно заметить, что показания на аналоговом входе изменяются не линейно, это особенность работы датчика.На рисунке 58 представлена схема подключения фоторезистора к Ардуино на макетной плате.

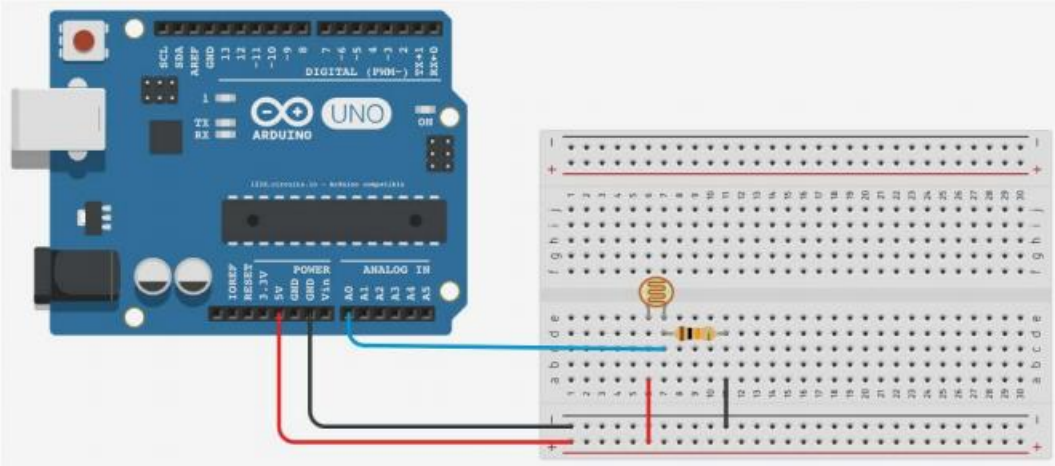


Рисунок 58- Схема подключения фоторезистора к Ардуино на макетной плате.

В зависимости от того, в каком плече делителя мы поставили фоторезистор, на аналоговый вход будет подаваться или повышенное или уменьшенное напряжение. В том случае, если одна нога фоторезистора подключена к земле, то максимальное значение напряжения будет соответствовать темноте (сопротивление фоторезистора максимальное, почти все напряжение падает на нем), а минимальное – хорошему освещению (сопротивление близко к нулю, напряжение минимальное). Если мы подключим плечо фоторезистора к питанию, то поведение будет противоположным, как показано на рисунке 59.

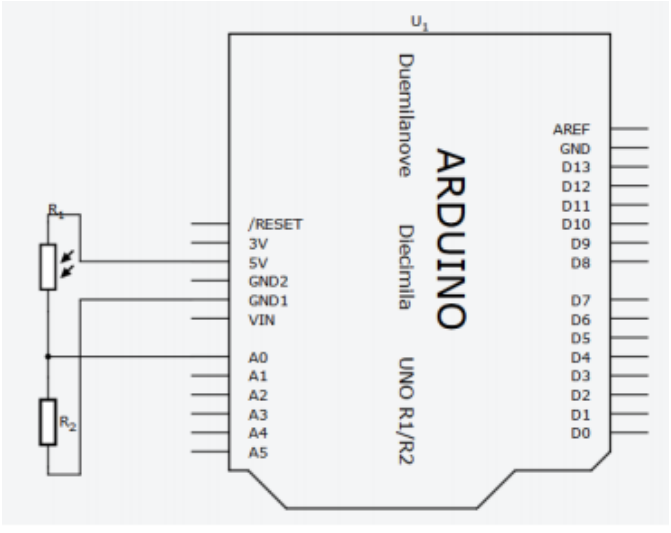


Рисунок 59- Подключение плеча фоторезистора к питанию

Сам монтаж платы не должен вызывать трудностей. Так как фоторезистор не имеет полярности, подключить можно любой стороной, к плате его можно припаять, подсоединить проводами с помощью монтажной платы или использовать обычные клипсы (крокодилы) для соединения. Источником питания в схеме является сам

ардуино. **Фоторезистор** подсоединяется одной ногой к земле, другая подключается к АЦП платы (в нашем примере – А0). К этой же ноге подключаем резистор 10 кОм. Естественно, подключать фоторезистор можно не только на аналоговый пин А0, но и на любой другой.

Несколько слов относительно дополнительного резистора на 10 К. У него в нашей схеме две функции: ограничивать ток в цепи и формировать нужное напряжение в схеме с делителем. Ограничение тока нужно в ситуации, когда полностью освещенный фоторезистор резко уменьшает свое сопротивление. А формирование напряжения – для предсказуемых значений на аналоговом порту. На самом деле для нормальной работы с нашими фоторезисторами хватит и сопротивления 1К.

Меняя значение резистора мы можем “сдвигать” уровень чувствительности в “темную” и “светлую” сторону. Так, 10 К даст быстрое переключение наступления света. В случае 1К датчик света будет более точно определять высокий уровень освещенности.Если вы используете готовый модуль датчика света, то подключение будет еще более простым. Соединяем выход модуля VCC с разъемом 5В на плате, GND – с землей. Оставшиеся выводы соединяем с разъемами ардуино.

Если на плате представлен цифровой выход, то отправляем его на цифровые пины. Если аналоговый – то на аналоговые. В первом случае мы получим сигнал срабатывания – превышения уровня освещенности (порог срабатывания может быть настроен с помощью резистора подстройки). С аналоговых же пинов мы сможем получать величину напряжения, пропорциональную реальному уровню освещенности.

1.1.1 Программа вывода значений фоторезистора на COM-порт

Подключив фоторезистор по приведенной схеме, начинаем писать программу. Первое, что мы сделаем, это выведем необработанный сигнал с аналогового входа в последовательный порт, для того чтобы просто понять, как меняется значение на входе А0. Соответствующая программа имеет вид:

```
const int pinPhoto = A0;
int raw = 0;
void setup() {
  Serial.begin(9600);
  pinMode( pinPhoto, INPUT );
}
void loop() {
  raw = analogRead( pinPhoto );
  Serial.println( raw );
  delay(200);
}
```

Запустив эту программу, мы получим следующие значения с датчика, представленные на рисунке 60.

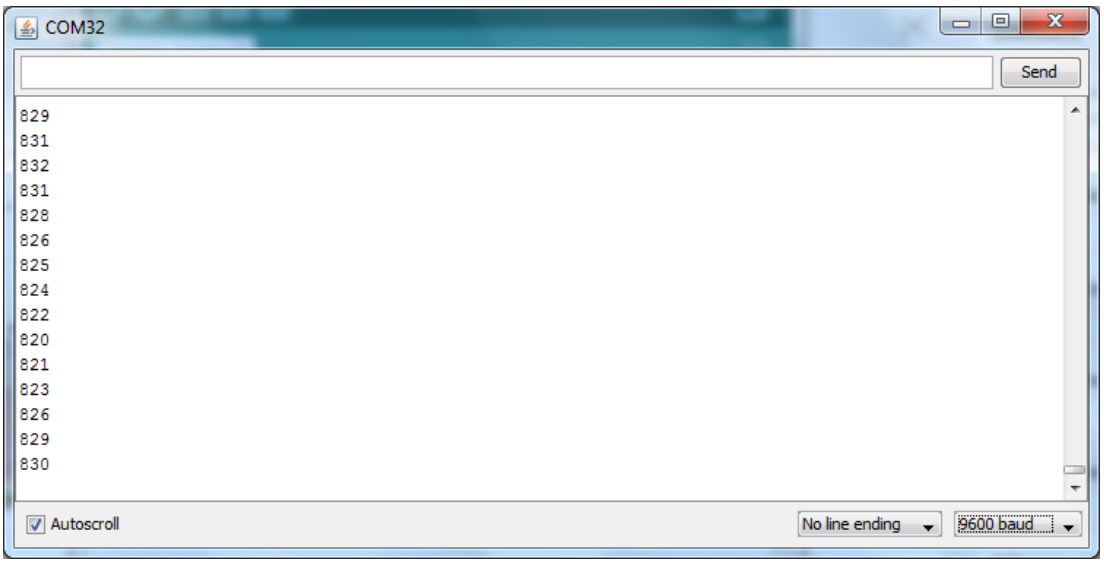


Рисунок 60- Значения с датчика на COM-порте.

А теперь прикроем датчик рукой и получим результат, представленный на рисунке 61.

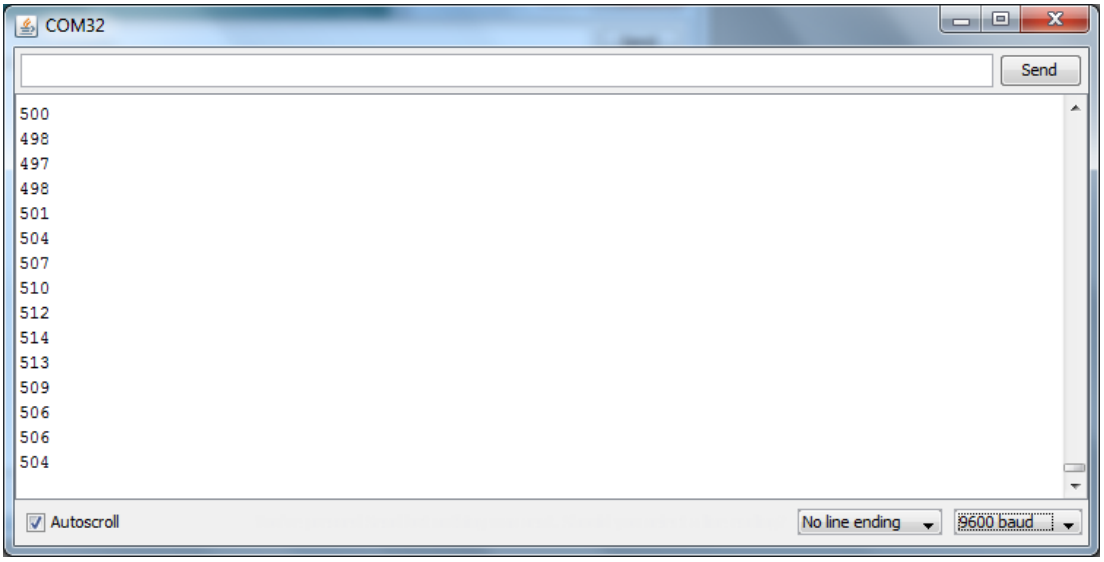


Рисунок 61- Значения с датчика на COM-порте после затемнения датчика.

Видно, что значение сильно меняется. От 830 при прямом попадании света, до 500 в случае затенения (появление преграды на пути света). Зная такое поведение, мы можем численно определить порог срабатывания. Пусть он будет равен, скажем, 600. Не ровно 500, потому что мы хотим обезопасить себя от случайного срабатывания. Вдруг над датчиком пролетит муха — он слегка затенится, и покажет 530.Наконец, добавим в программу некое действие, которое будет совершаться если уровень освещенности станет ниже заданного порога. Самое простое, что мы можем сделать — это зажигать на Ардуино штатный светодиод #13. Получается такая вот программа:

```
const int pinPhoto = A0;
const int led = 13;
int raw = 0;
void setup() {
```

```
 pinMode( pinPhoto, INPUT );
 pinMode( led, OUTPUT );
}
void loop() {
 raw = analogRead( pinPhoto );
 if( raw < 600)
   digitalWrite( led, HIGH );
 else
   digitalWrite( led, LOW );
 delay(200);
}
```

Накрываем датчик рукой (или выключаем свет в комнате) — светодиод зажигается. Убираем руку — гаснет. А теперь представьте, что вы зажигаете не светодиод, а подаете сигнал на реле, которое включает лампу в подъезде вашего дома. Получаеся готовый прибор для экономии электроэнергии.

1.2 Пример скетча датчика освещенности на фоторезисторе

Мы подключили схему с фоторезистором к ардуино, убедились, что все сделали правильно. Теперь осталось запрограммировать контроллер.Написать скетч для датчика освещенности довольно просто. Нам нужно только снять текущее значение напряжения с того аналогового пина, к которому подключен датчик. Делается это с помощью известной нам всем функции analogRead(). Затем мы можем выполнять какие-то действия, в зависимости от уровня освещенности.Давайте напишем скетч для датчика освещенности, включающего или выключающего светодиод, подключенный по следующей схеме, представленной на рисунке 62.

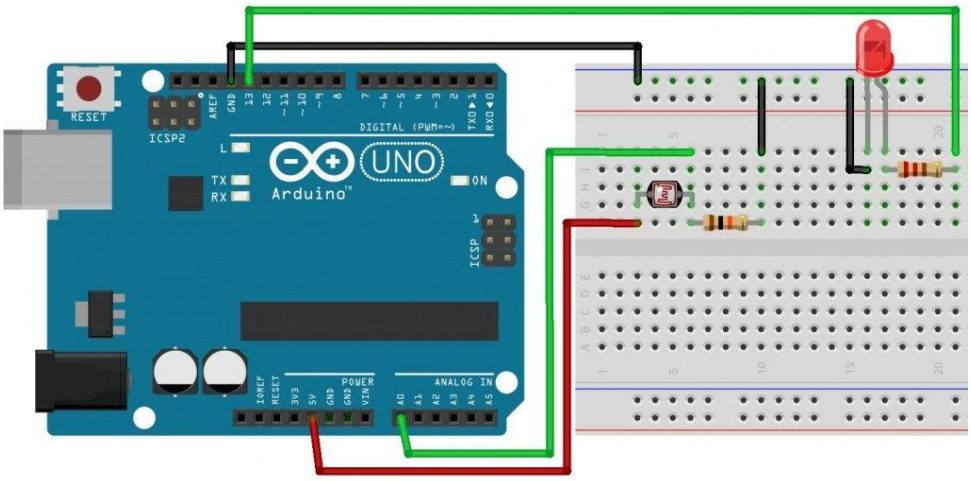


Рисунок 62- Подключение схемы с фоторезистором и светодиодом к Ардуино

Алгоритм работы проекта таков:

- определяем уровень сигнала с аналогового пина;
- сравниваем уровень с пороговым значением. Максимально значение будет соответствовать темноте, минимальное – максимальной освещенности. Пороговое значение выберем равное 300;

- если уровень меньше порогового – темно, нужно включать светодиод;
- иначе – выключаем светодиод.

Приведем код программы для данного проекта.

```
#define PIN_LED 13
#define PIN_PHOTO_SENSOR A0

void setup() {
Serial.begin(9600);
pinMode(PIN_LED, OUTPUT);
}

void loop() {
int val = analogRead(PIN_PHOTO_SENSOR);
Serial.println(val);
if (val < 300) {
digitalWrite(PIN_LED, LOW);
} else {
digitalWrite(PIN_LED, HIGH);
}
}
```

Прикрывая фоторезистор (руками или светонепроницаемым предметом), можем наблюдать включение и выключение светодиода. Изменяя в коде пороговый параметр, можем заставлять включать/выключать лампочку при разном уровне освещения.При монтаже постарайтесь расположить фоторезистор и светодиод максимально далеко друг от друга, чтобы на датчик освещенности попадало меньше света от яркого светодиода.

1.3 Датчик освещенности и плавное изменение яркости подсветки

Можно модифицировать проект так, чтобы в зависимости от уровня освещенности менялась яркость светодиода. В алгоритм мы добавим следующие изменения:

- яркость лампочки будем менять через ШИМ, посылая с помощью analogWrite() на пин со светодиодом значения от 0 до 255;
- для преобразования цифрового значения уровня освещения от датчика освещенности (от 0 до 1023) в диапазон ШИМ яркости светодиода (от 0 до 255) будем использовать функцию map().

Пример скетча для данного проекта:

```
#define PIN_LED 10
#define PIN_PHOTO_SENSOR A0

void setup() {
Serial.begin(9600);
pinMode(PIN_LED, OUTPUT);
}
```

```
8.
9.  void loop() {
10. int val = analogRead(PIN_PHOTO_SENSOR);
11. Serial.println(val);
12.
13. int ledPower = map(val, 0, 1023, 0, 255); // Преобразуем полученное значение в уровень PWM-сигнала. Чем меньше значение освещенности, тем меньше мощности мы должны подавать на светодиод через ШИМ.
14.
15. analogWrite(PIN_LED, ledPower); // Меняем яркость
16.
17. }
```

В случае другого способа подключения, при котором сигнал с аналогового порта пропорционален степени освещенности, надо будет дополнительно «обратить» значение, вычитая его из максимального:

```
1. int val = 1023 – analogRead(PIN_PHOTO_RESISTOR)
```

1.4 Схема датчика освещения на фоторезисторе и реле

Примеры скетча для работы с реле приведены в статье, посвященной программированию реле в ардуино. В данном случае, нам не нужно делать сложных телодвижений: после определения «темноты» мы просто включаем реле, подавай на его пин соответствующее значение.

```
1.  #define PIN_RELAY 10
2.  #define PIN_PHOTO_SENSOR A0
3.
4.  void setup() {
5.  pinMode(PIN_RELAY, OUTPUT);
6.  digitalWrite(PIN_RELAY, HIGH);
7.  }
8.
9.  void loop() {
10. int val = analogRead(PIN_PHOTO_SENSOR);
11. if (val < 300) {
12. // Светло, выключаем реле
13. digitalWrite(PIN_RELAY, HIGH);
14. } else {
15. // Темновато, включаем лампочку
16. digitalWrite(PIN_RELAY, LOW);
17. }
18. }
```

Подключение фоторезистора осуществляется по схеме делителя напряжения с дополнительным сопротивлением. Датчик подключается к аналоговому порту для измерения различных значений уровня освещенности или к цифровому, если нам важен лишь факт наступления темноты. В скетче мы просто считываем данные с аналогового (или цифрового) порта и принимаем решение, как реагировать на изменения.

2.Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта

2.5 Добавьте в приведенную выше в примере программу некое действие, которое будет совершаться если уровень освещенности станет ниже заданного порога. Вместо одного возьмите несколько светодиодов(два или три).Накрываем датчик рукой (или выключаем свет в комнате) — светодиод зажигается. Убираем руку — гаснет. Соберите схему проекта на монтажной плате, подключить к питанию согласно схемы и полярности напряжения.
2.6 Разработайте код программы согласно проекта, указанного в задании и Приложении J.

2.7 Загрузите разработанный код в среду разработки Arduino IDE.

2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.11 Продемонстрируйте преподавателю результат выполненной работы.

Контрольные вопросы.

1.Приведите принцип работы датчика освещенности на основе фоторезистора.

2.Скажите, что указано на маркироровке фоторезистора.

3.Приведите достоинства и недостатки датчика на основе фоторезистора.

4.Приведите схему подключения датчика на основе фоторезистора к Ардуино.

5. Объясните, с какого пина снимается текущее значение напряжения датчика на основе фоторезистора и с помощью какой функции.

Лабораторная работа № 12 Программирование включения светодиодов через реле в Arduino UNO

Цель работы: написать код программы для включения светодиодов через реле srd-05vdc-sl в Arduino UNO.

Содержание работы:

- собрать схему подключения светодиодов к макетной плате согласно задания;
- разработать код программы;
- загрузить код программы в Ардуино, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в Ардуино на исполнение;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

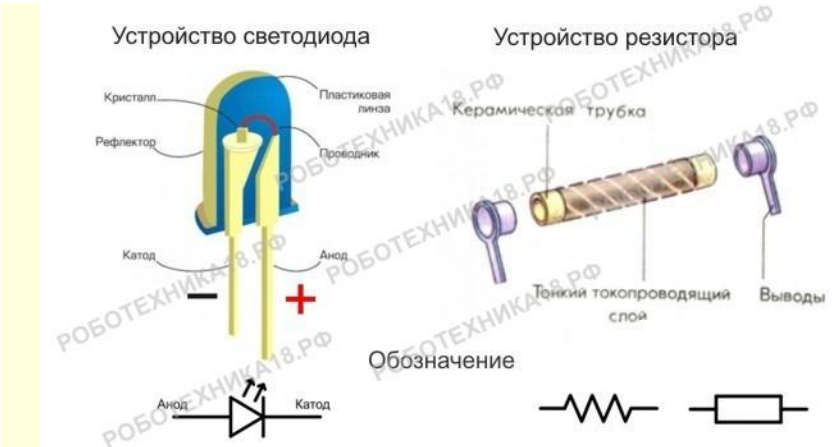
Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле, светодиоды, кнопки.

1.Краткие теоретические сведения

Светодиоды имеют полярность (+ и —) и чувствуют направление движения постоянного тока. Если подключить светодиод неправильно, то постоянный ток не пройдет и прибор не засветится. Кроме того, светодиод может выйти из строя при неправильном подключении. Анод (длинная ножка светодиода) всегда подключается к плюсу.

Для каких целей диод включают с резистором? Дело в том, что в светодиоде стоит кристалл который боится больших токов. Резистор же призван ограничивать силу тока (Ампер) в светодиоде, чтобы он не перегорел. Большой ток губителен для светодиода, меньший ток (благодаря подключению резистора) обеспечивает длительную работу диода.



Приведем готовый скетч подключения светодиода:

```
void setup() // процедура setup

{
pinMode(13, OUTPUT); // объявляем пин 13 как выход
}

void loop() // процедура loop

{

digitalWrite(13, HIGH); // зажигаем светодиод

delay(1000); // ждем 1 секунду

digitalWrite(13, LOW); // выключаем светодиод

delay(1000); // ждем 1 секунду

}
```

Пояснения к коду:

1. Процедура `setup` выполняется при запуске микроконтроллера один раз. Используется для конфигурации портов микроконтроллера и других настроек;
2. После выполнения `setup` запускается процедура `loop`, которая выполняется в бесконечном цикле. Это мы используем, чтобы светодиод мигал постоянно;
3. Процедуры `setup` и `loop` должны присутствовать в любой программе (скетче), даже если вам не нужно ничего выполнять в них — пусть они будут пустые, просто не пишите ничего между фигурными скобками;
4. Запомните, что каждой открывающей фигурной скобке `{` всегда соответствует закрывающая `}`. Они обозначают границы некого логически завершенного фрагмента кода. Следите за вложенностью фигурных скобок;
5. Использованные константы: `INPUT`, `OUTPUT`, `LOW`, `HIGH`, пишутся заглавными буквами, иначе компилятор их не распознает и выдаст ошибку. Когда ключевое слово распознано, оно подсвечивается синим цветом в Arduino IDE.

Пример

2.Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.

2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.

2.4 Необходимо проверить наличие необходимой библиотеки для проекта

2.5 Собрать схему проекта на монтажной плате, подключить к питанию согласно схемы и полярности напряжения.

2.6 Разработайте код программы согласно проекта, указанного в задании и Приложении К.

2.7 Загрузите разработанный код в среду разработки Arduino IDE.

2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.

2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.

2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.

2.11 Продемонстрируйте преподавателю результат выполненной работы.

3.Контрольные вопросы

1.Приведите функциональную схему реле(контактную группу и напряжение питания)
2.Расскажите принцип работы реле в Ардуино.
3. Зарисуйте схему подключения реле к Ардуино для включения мощной нагрузки.
4.Расскажите работу скетча по схеме управления модулем реле в Ардуино
5.Поясните, какое действие выполнит реле при при подаче логической единицы (HIGH) в строке digitalWrite(relayPin, HIGH)

4. Вывод

Приложение I

(Варианты заданий)

Вариант 1: Подключите к Ардуино следующее оборудование, согласно схемам, рекомендациям приведенным в этой методичке. Разработайте алгоритм проекта: «При повышении температуры относительно текущей, измеренной датчиком DHT 11, включается реле 1 типа srd-05vdc-sl. Через нормально открытый контакт реле включает два светодиода. При установившейся текущей температуре светодиоды выключаются.Текущая температура выводится на LCD Keypad SHILD дисплей и COM-порт ПК».

Вариант 2: Подключите к Ардуино следующее оборудование, согласно схемам, рекомендациям приведенным в этой методичке. Разработайте алгоритм проекта: «При повышении температуры относительно текущей, измеренной датчиком DHT 11, включается реле 1 типа srd-05vdc-sl. Через нормально открытый контакт реле включает RGB-светодиод. При установившейся текущей температуре светодиоды выключаются.Текущая температура выводится на LCD Keypad SHILD дисплей и COM-порт ПК».

Лабораторная работа № 13 Управление мощной нагрузкой через реле в Arduino UNO

Цель работы: написать код программы для управления мощной нагрузкой через реле в Arduino UNO.

Содержание работы:

- собрать схему подключения светодиодов к макетной плате согласно задания;
- разработать код программы;
- загрузить код программы в Ардуино, выполнить его компиляцию;
- загрузить код программы, прошедший компиляцию, в Ардуино на исполнение;
- выполнить проверку работоспособности оборудования проекта, согласно кода.

Средства обучения (оборудование и материалы):

- ПК преподавателя с мультимедийным проектором, с доступом в Интернет;
- ученический ПК с соответствующими системными требованиями, с доступом в Интернет для поиска справочной информации;
- программное обеспечение:
  - ОС Windows 8 и выше;
  - загрузочный файл Arduino версии 1.6.7 и выше;
  - набор библиотек для Arduino UNO.
- методические указания, конспекты;
- набор инструментов;
- плата Arduino UNO с комплектом датчиков(температуры, влажности, освещенности), соединительные провода, провод USB, макетная плата, LCD Keypad SHILD дисплей, реле srd-05vdc-sl, светодиоды, кнопки.

1 Краткие теоретические сведения

Подключение модуля реле к Ардуино потребуется, если необходимо управлять с помощью Ардуино мощной нагрузкой или переменным током. Модуль реле SRD-05VDC-SL-C позволяет управлять электрическими цепями с переменным током до 250 Вольт и нагрузкой до 10 Ампер. Рассмотрим схему подключения реле, как управлять модулем для включения лампы накаливания на 12 Вольт. Схема подключения реле srd-05vdc-sl к Ардуино Уно приведена на рисунке 63.

Рисунок 63-Схема подключения реле srd-05vdc-sl к Ардуино Уно

Соберите схему, как показано на рисунке 5. Модуль имеет три контакта для управления от микроконтроллера Ардуино и два контакта для подключения мощной электрической цепи. Схема подключения реле к Ардуино УНО:

- ✓ GND — GND;
- ✓ VCC — 5V;
- ✓ In — любой цифровой порт.

После сборки электрической схемы, загрузите следующий скетч в микроконтроллер. Скетч для управления реле от Ардуино

```
void setup() {

pinMode(3, OUTPUT); // объявляем пин 3 как выход

}


void loop() {

digitalWrite(3, HIGH); // замыкаем реле


delay(3000); // ждем 3 секунды


digitalWrite(3, LOW); // размыкаем реле


delay(1000); // ждем 1 секунду

}
```

Сначала мы устанавливаем переменную relPin, модуль реле подключается к пину 3. Далее устанавливается сигнал с реле как выходной.А в цикле программы у нас включается реле, через секунду выключается и через 3 секунды снова включается.

После загрузки скетча включите блок питания лампочки 12 Вольт в цепь. Реле при этом должно устанавливаться в разрыве одного из проводов, идущего к нагрузке. Для безопасности лучше устанавливать реле в провод заземления. К минусам реле следует отнести щелчки при замыкании/размыкании контакта, поэтому для включения приборов до 40 Вольт удобнее использовать транзисторы.

К минусам данного типа реле можно отнести большое потребление тока, малую живучесть при больших нагрузках и возможное залипание контактов, если была подключена большая нагрузка.

Приведем пример 1 скетча включения реле при повышении температуры и его выключении при понижении:

```
#include <dht11.h>

dht11 DHT;

#define DHT11_PIN 2

#include <LiquidCrystal.h>

LiquidCrystal  lcd (8,9,4,5,6,7);

int SVET = 11;

void setup(){

  Serial.begin(9600);

    lcd.begin(16, 2);

  //optionally, now set up our application-specific display settings, overriding whatever the lcd did in lcd.init()

  //lcd.commandWrite(0x0F);//cursor on, display on, blink on.  (nasty!)

  lcd.clear();

  Serial.println("DHT TEST PROGRAM ");

  lcd.write("DHT TEST PROGRAM");

  delay(5000);

  Serial.print("LIBRARY VERSION: ");

  lcd.setCursor(0,0);

  lcd.write("LIBRARY VERSION:");

  lcd.setCursor(0,2);

  lcd.write(DHT11LIB_VERSION);

  Serial.println(DHT11LIB_VERSION);

  Serial.println();
```

```
  delay(5000);
  Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
  lcd.clear();
  lcd.write("ElitZoo Control");
  lcd.setCursor(0,2);
  lcd.write("Status H(%) T(C)");
  delay(5000);
  lcd.setCursor(0,2);
  lcd.write("                ");
  pinMode(SVET, OUTPUT);
  pinMode(3, OUTPUT);
}
void loop(){
  int chk;
  int x;
  x = analogRead(A0);
  Serial.print("DHT11, \t");
  chk = DHT.read(DHT11_PIN);    // READ DATA
  switch (chk){
    case DHTLIB_OK:
                lcd.setCursor(0,2);  //line=2, x=0
          lcd.write("OK");
                Serial.print("OK,\t");
                break;
    case DHTLIB_ERROR_CHECKSUM:
                lcd.setCursor(0,2);  //line=2, x=0
          lcd.write("ERR");
                Serial.print("Checksum error,\t");
                break;
    case DHTLIB_ERROR_TIMEOUT:
                lcd.setCursor(0,2);  //line=2, x=0
```

```
        lcd.write("TOUT");
            Serial.print("Time out error,\t");
            break;
   default:
            lcd.setCursor(0,2);  //line=2, x=0
        lcd.write("UERR");
            Serial.print("Unknown error,\t");
            break;
 }
// DISPLAT DATA
 Serial.print("H=");
 Serial.print(DHT.humidity,1);
 Serial.print(",\t");
 Serial.print("T=");
 Serial.println(DHT.temperature,1);
 Serial.print(",\t");
 Serial.print("Вл=");
 Serial.print(x);
 lcd.setCursor(5,2);  //line=2, x=0
 lcd.write("H=");
 lcd.print(DHT.humidity,1);
 lcd.write("% T=");
 lcd.print(DHT.temperature,1);
 lcd.write("C");
 digitalWrite(3,HIGH);
 digitalWrite(SVET,HIGH);
 delay(5000);
 digitalWrite(3,LOW);
 digitalWrite(SVET,LOW);
 lcd.setCursor(0,2);
 lcd.write("            ");
```

}

2.Порядок выполнения работы:

2.1 Настройте оборудование перед началом работы-подключите плату Arduino UNO к ПК через USB-кабель.
2.2 После установки запустите на рабочем столе ярлык Arduino.exe. При запуске появится окно, в котором содержится заготовка программы.
2.3 После того как физический контакт ПК с контроллером установлен, нужно установить связи между ним и средой разработки Arduino IDE. Для этого необходимо выбрать номер COM-порта ПК.
2.4 Необходимо проверить наличие необходимой библиотеки для проекта
2.5 Собрать схему проекта на монтажной плате, подключить к питанию согласно схемы и полярности напряжения.
2.6 Разработайте код программы согласно проекта, указанного в задании и Приложении К.
2.7 Загрузите разработанный код в среду разработки Arduino IDE.
2.8 Среда разработки Arduino IDE выполнит компиляцию кода и, тем самым, проверит программу на наличие ошибок.
2.9 Если после компиляции выявились ошибки, то необходимо их исправить с помощью методических указаний, приведенных в пункте 1.2.1 «Ошибки компиляции для Arduino UNO» .Если компиляция прошла успешно, то загружаем скетч в контроллер Ардуино UNO на выполнение.
2.10 После загрузки кода в контроллер Arduino UNO необходимо проверить работу оборудования согласно проекта.
2.11 Продемонстрируйте преподавателю результат выполненной работы.

3.Контрольные вопросы.

1.Перечислите и назовите контакты модуля реле srd-05vdc-sl для подключения к Ардуино.
2. Приведите фрагмент скетча, где реле включается, через секунду выключается и через 3 секунды снова включается.
3.Известно, что вход реле srd-05vdc-sl является инвертированным. Поясните, это означает?
4. Объясните, нужна ли библиотека для работы реле srd-05vdc-sl.
5. Расскажите, каким образом можно определить, что реле сработало?

4. Вывод

\

Приложение К

(Варианты заданий)

Вариант 1: Подключите к Ардуино следующее оборудование, согласно схемам, рекомендациям приведенным в этой методичке. При повышении температуры относительно текущей, измеренной датчиком DHT 11, включается реле 1 типа srd-05vdc-sl. Через нормально открытый контакт реле включает вентилятор +12 Вольт.При понижении освещенности, измеренной датчиком освещенности на фоторезисторе, включается реле 2. Через нормально открытый контакт реле включает два светодиода. При установившейся текущей температуре вентилятор выключается. При повышении освещенности два светодиода отключаются.

Вариант 2: Подключите к Ардуино следующее оборудование, согласно схемам, рекомендациям приведенным в этой методичке. При повышении температуры относительно текущей, измеренной датчиком DHT 11, включается реле 1 типа srd-05vdc-sl. Через нормально открытый контакт реле включает три светодиода и RGB-светодиод.При понижении освещенности, измеренной датчиком освещенности на фоторезисторе, включается реле 2. Через нормально открытый контакт реле включает вентилятор +12. При установившейся текущей температуре все светодиоды отключаются. При повышении освещенности вентилятор отключается.

Используемая литература, Интернет-ресурсы

Основные источники:

1. Ромель А.П., Финкова М.А., Матвеев М.Д. «Windows 10. Все об использовании и настройках», самоучитель, М.:2016, Лань,336 стр.
2. Конова Е.А., Поллак Г.А. Алгоритмы и программы. Язык С++ - Издательство "Лань". 2018.

3.Кувшинов, Д. Р. Основы программирования : учеб. пособие для СПО / Д. Р. Кувшинов. — М. : Издательство Юрайт, 2019

4.Огнева, М. В. Программирование на языке С++: практический курс : учеб. пособие для СПО / М. В. Огнева, Е. В. Кудрина. — М. : Издательство Юрайт, 2019

5.Дело в программировании. Пособие по программированию Arduino Автор: Колмаков С. Год: 2017 Количество страниц:100 с., ил. Язык:русский Формат: pdf Размер:13,86 Мб
6. Конова, Е.А. Алгоритмы и программы. Язык С++ [Электронный ресурс] : учебное пособие / Е.А. Конова, Г.А. Поллак. — Электрон. дан. — Санкт-Петербург : Лань, 2018

7.Новожилов, О. П. Архитектура компьютерных систем в 2 ч. Часть 1 : учеб. пособие для СПО / О. П. Новожилов. — М. : Издательство Юрайт, 2019

8.Новожилов, О. П. Архитектура компьютерных систем в 2 ч. Часть 2 : учеб. пособие для СПО / О. П. Новожилов. — М. : Издательство Юрайт, 2019

Дополнительные источники:

1.Новожилов, О. П. Информатика в 2 ч. Часть 2 : учебник для СПО / О. П. Новожилов. — 3-е изд., перераб. и доп. — М. : Издательство Юрайт, 2019

Интернет-ресурсы:

1.www.4pc.info- новости, статьи, обзоры, драйвера.
2.www.winsov.ru-на сайте можно получить много полезной информации: статьи и документация.
3.www.ezpc.ru-энциклопедия компьютерных знаний.
4.www.chip-dip.ru-Чип и Дип – электронные компоненты и приборы.
5.www.platan.ru-электронные компоненты и измерительная техника.
6.www.mirmk.net-мир микроконтроллеров.
7.https://radiohata.ru/arduino/1327-programmirovanie-arduino.html