ПРАКТИКУМ
К СПЕЦКУРСУ «ПРОГРАММИРОВАНИЕ»
ДЛЯ ПРОФИЛЬНЫХ ИТ-КЛАССОВ
(10-11 класс)

Составители:
Волкова Г.Ю.
Готовец Д.М.
Мещеринова О.Л.
Познахирко В.В.
Комаров А.А.
Ляндау Ю.В.
Митасов Н.С.
Михайлин Н.А.
Солдатова О.Б.
Шимбирёв А.А.

2021 год

Сборник практических и лабораторных работ «Программирование» (10-11 класс) разработан в соответствии с рабочей программой курса и учебным пособием по специальному курсу «Программирование» (10-11 класс). Сборник расширяет методическое обеспечение учителя для работы в профильных ИТ-классах.

Сборник включает два раздела.

Раздел 1 для 10 класса: включает 25 лабораторных работ по темам спецкурса структурного с элементами объектно-ориентированного программирования на языке С++.

Раздел 2 для 11 класса: включает 25 лабораторных работ по программированию микроконтроллеров Arduino на языке программирования С++. Каждая работа представлена с подробным описанием: цель, задание, список оборудования, программное обеспечение, краткие теоретические сведения, порядок выполнения работы, содержание отчета. Каждая лабораторная работа сопровождается инструкциями для учителя по организации и проверке работ.

Сборник практических и лабораторных работ по специальному курсу «Программирование» (10-11 класс) может применяться в рамках реализации специального курса «Программирование» для учащихся проекта «ИТ-класс в Московской школе» и обеспечивает: развитие методического обеспечения профильной подготовки школьников в IT-классах; повышение эффективности методического обеспечения учебного процесса по специальному курсу «Программирование» для 10 -11 классов; повышение результативности работы учителя в рамках лабораторных практикумов.

СОДЕРЖАНИЕ

РАЗДЕЛ 1. СБОРНИК ЛАБОРАТОРНЫХ РАБОТ ПО СПЕЦКУРСУ «ПРОГРАММИРОВАНИЕ» ДЛЯ 10 ИТ-КЛАССА

РАЗДЕЛ 1.1. ИНСТРУКЦИЯ ПО ОРГАНИЗАЦИИ И ПРОВЕДЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО СПЕЦКУРСУ «ПРОГРАММИРОВАНИЕ» ДЛЯ 11 ИТ-КЛАССА

Лабораторная работа №1. Ввод и вывод данных, операторы присваивания

C++ – это язык программирования, который позволяет разрабатывать высокопроизводительное программное обеспечение. Высокая производительность языка позволяет строить фундамент, на котором были основаны другие языки программирования и средства обработки данных.

Язык C++ используется для профессионального программирования, он не прост в обучении, но после освоения языка можно писать собственные программные решения и изучать другие языки программирования, созданные на его основе.

Для начала работы с языком необходимо познакомиться с несколькими базовыми терминами: **Исходный код** и **Объектный код**.

Исходный код – это текстовая форма программы, то есть это версия, которую может прочитать человек, а объектный код – это форма программы, которую может выполнить компьютер.

Итак, разобрав, что такое *Исходный код* и *Объектный код*, теперь можно приступить к написанию первой программы:

Исторически сложилось, что самая первая программа в независимости от языка программирования представляет собой сообщение «Hello World!», поэтому также начнем с неё.

Каждая программа на C++ состоит из нескольких частей:

1. Библиотеки необходимые для работы;
2. Пространство имен, позволяющее работать с данными;
3. Тело программы;

Для создания первой программы необходимо установить на компьютер программное обеспечение, которое сможет преобразовывать исходный код в объектный.

Для написания кода на языке C++ необходимо выбрать бесплатную интегрированную среду разработки под названием Dev C++ IDE. Скачать её можно с официального сайта: http://dev-cpp.com/. Ниже описан процесс скачивания и установки среды.

Прокрутив главную страницу ниже

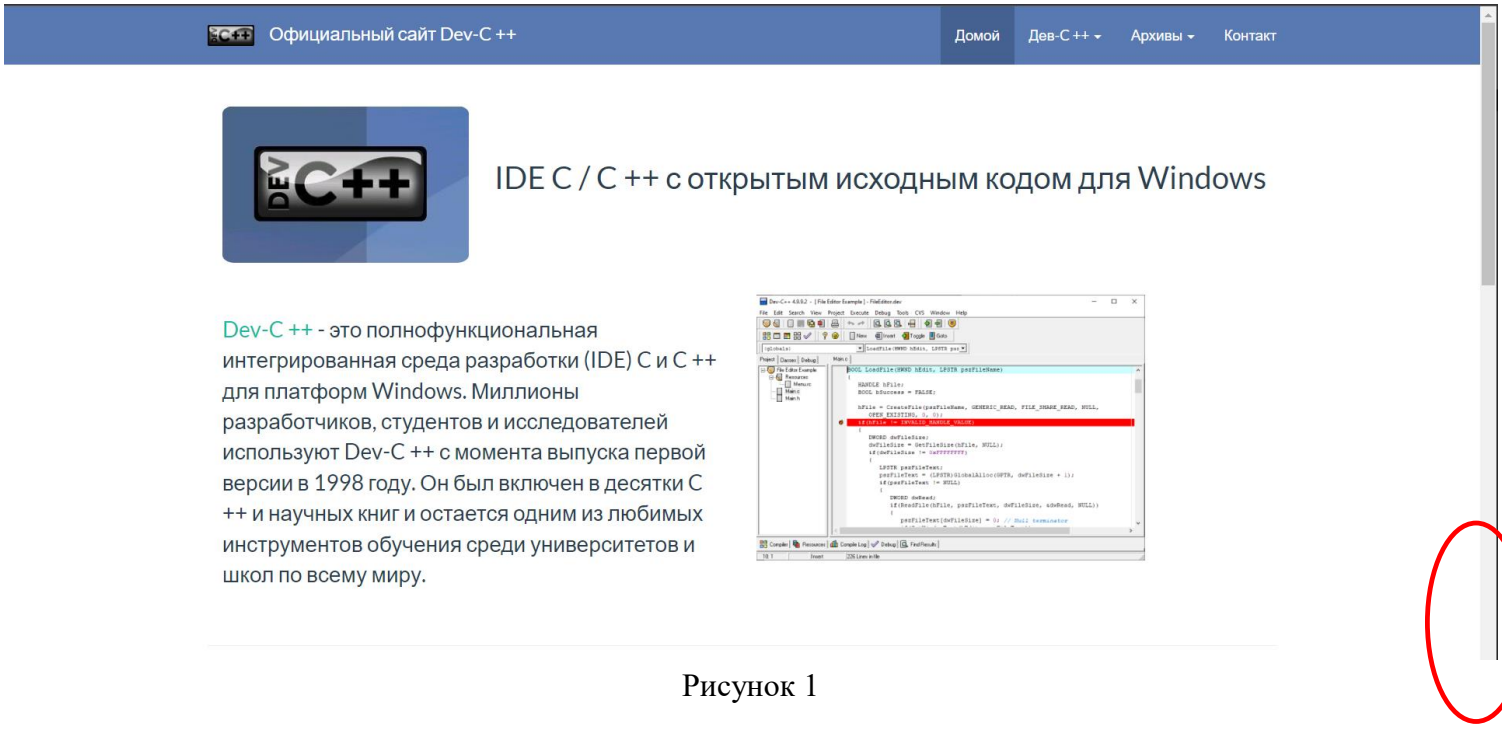


Рисунок 1

На странице http://dev-cpp.com/ в разделе ***Get Dev-C++*** нажмите ***Embarcadero Dev-C++ 6.3*** :

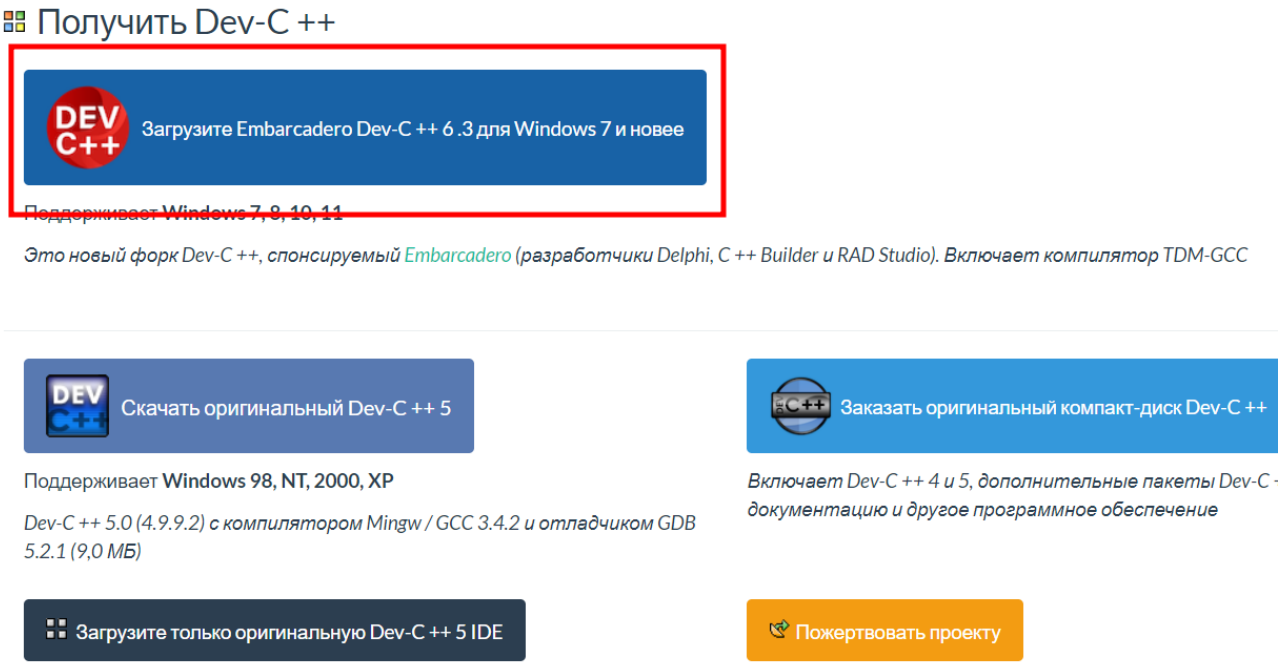


Рисунок 2

Откроется страница, на которой необходимо нажать на кнопку ***Download***

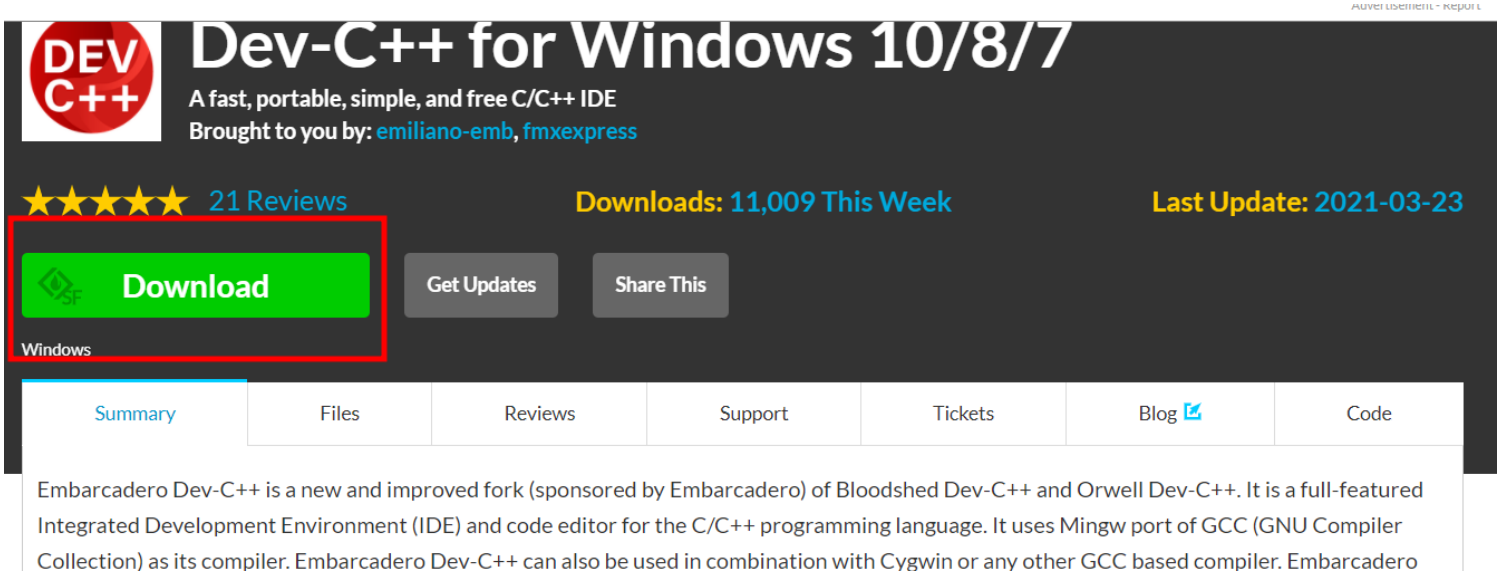


Рисунок 3

После окончания скачивания файла с примерным именем ***_Dev-Cpp_6.3_TDM-GCC 9.2_Setup.exe*** запустите установку программы.

Откроется приветственное окно установщика. Нажмите кнопку “***I Agree***”

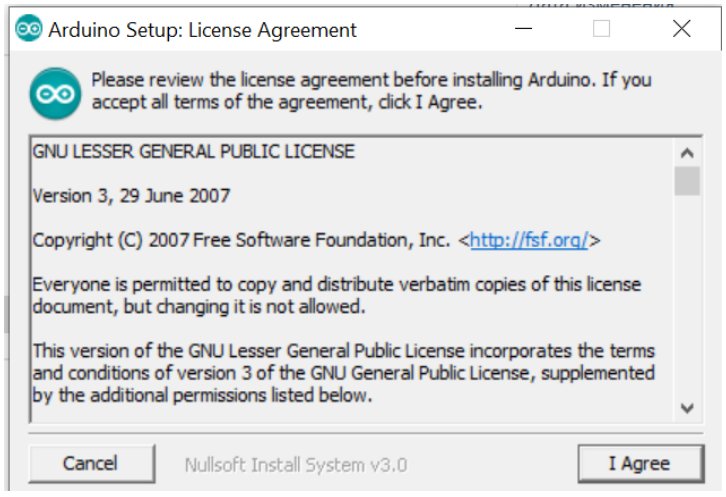


Рисунок 4

В следующем окне будут отмечены компоненты среды, которые планируются для установки. Нажмите кнопку ***Next***:

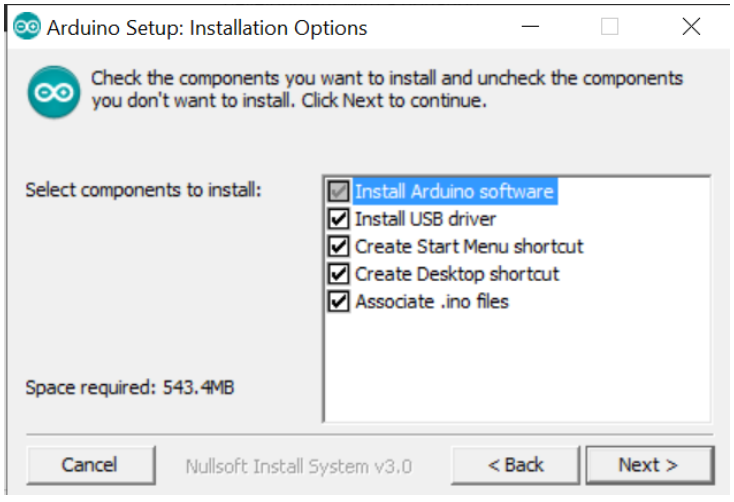


Рисунок 5

В следующем окне будет показан путь к папке, где планируется установить Arduino IDE. Согласитесь с предложенным выбором или укажите другую папку, после этого нажмите кнопку ***Install.***

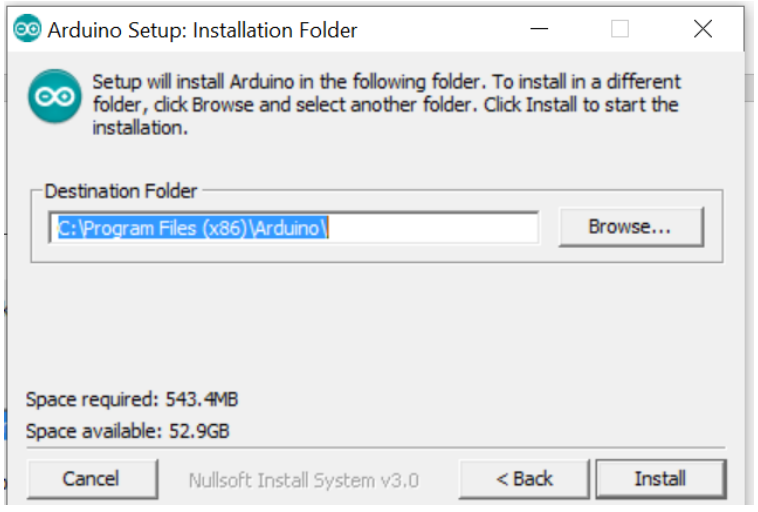


Рисунок 6

После окончания процесса установки вы увидите следующее окно (Рисунок 7). Нажмите кнопку ***Close*** для закрытия окна и завершения процесса установки.

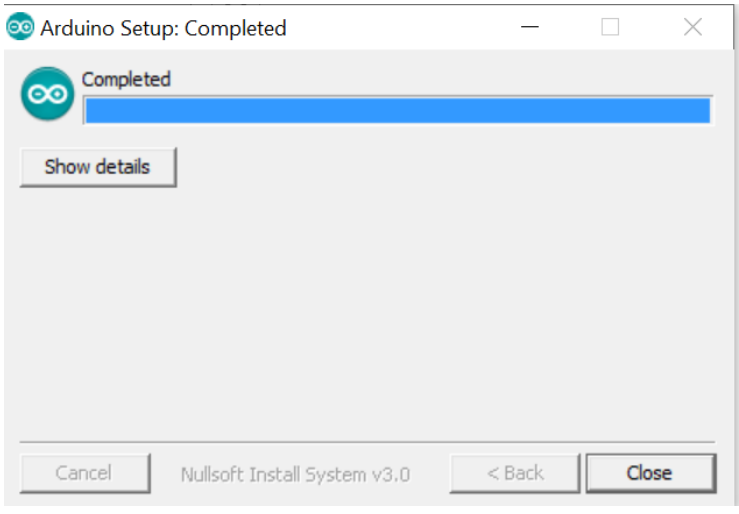


Рисунок 7

Теперь среда Arduino IDE установлена и перед началом работы в ней давайте познакомимся с тем, как выглядит её окно после запуска.

Запустите Arduino IDE, воспользовавшись ярлыком на рабочем столе, который был создан при установке:  .

Примерный вид окна показан ниже:

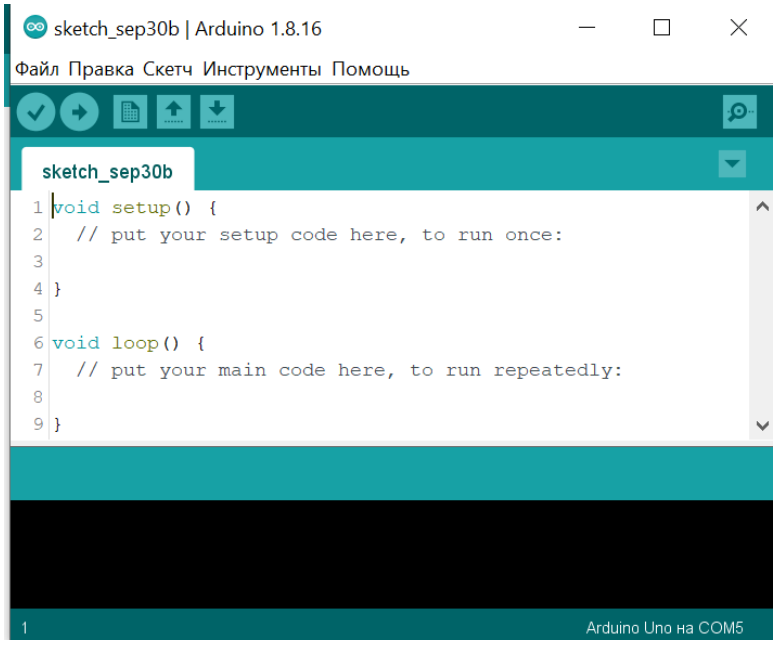


Рисунок 8

На рисунке ниже цифрами обозначены основные области окна среды, рассмотрим их.

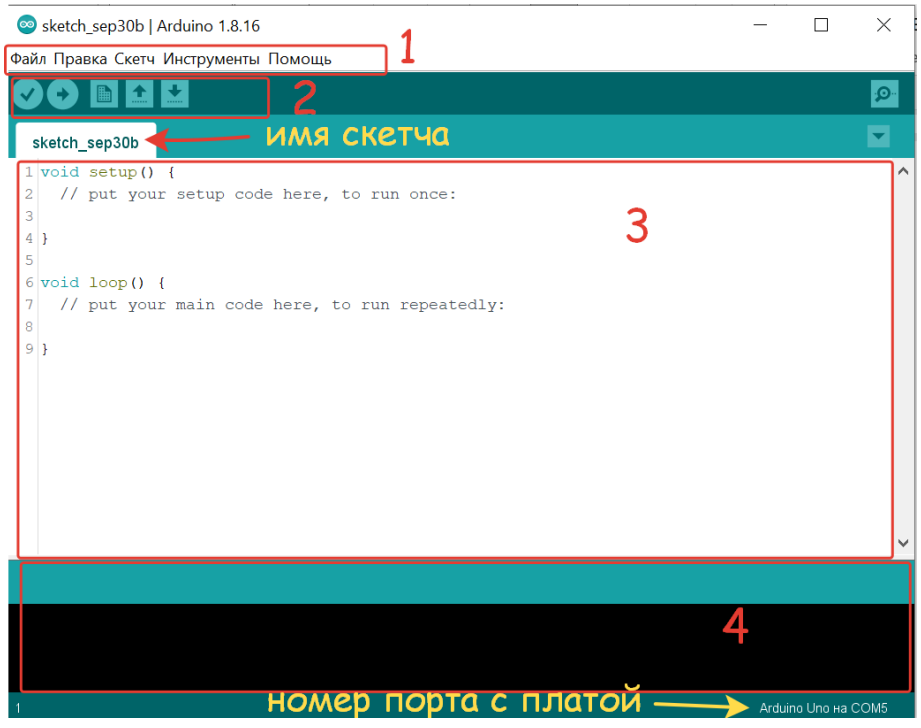


Рисунок 9

Область 1: строка раскрывающихся меню:

- меню ***Файл*** используется для создания и сохранения скетчей (скетчами в среде Arduino IDE принято называть файлы с программным кодом);
- меню ***Правка*** содержит команды настройки оформления кода;
- меню ***Скетч*** используется для подключения библиотек;
- меню ***Инструменты*** используется для настройки подключения платы к персональному компьютеру;
- меню ***Помощь*** содержит справочную информацию.

Область 2: в ней расположены основные инструменты, которые используются в работе:

- кнопка ***Проверить***: выполняет проверку на ошибки и компиляцию программного кода, находящегося в рабочем поле (***Область 3***); этой кнопкой вы будете пользоваться после того, как напишите программный код, чтобы проверить правильность его написания. Если в коде будут обнаружены ошибки, сообщение об этом будет выведено в нижней части окна, а строка, в которой обнаружена ошибка, будет подсвечена.

Рисунок 10

- кнопка ***Загрузить***: эта кнопка также выполняет компиляцию и проверку на ошибки программного кода из рабочего поля (***Область 3***), но при этом она еще и загружает готовый код в плату Arduino. Обратите внимание, что если вы нажмете эту кнопку, а плата Arduino не будет подключена к компьютеру USB кабелем, то в нижней части окна будет выведено сообщение «***Произошла ошибка при загрузке скетча***».

Рисунок 11

Если загрузка скетча прошла успешно, то сообщение будет таким:

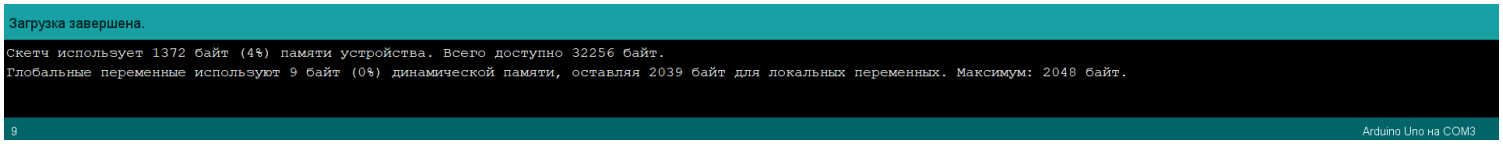


Рисунок 12

Иногда при загрузке скетча возникает проблема, связанная с неправильным определением номера COM-порта.

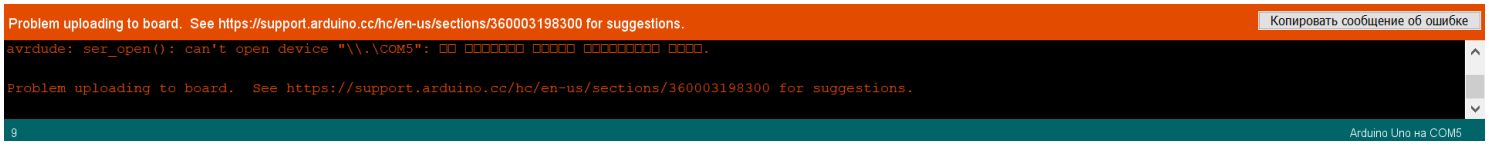


Рисунок 13

В этом случае нужно открыть меню ***Инструменты*** и выбрать предложенный средой номер, например, так, как показано ниже (у вас номера COM-портов могут быть другие).

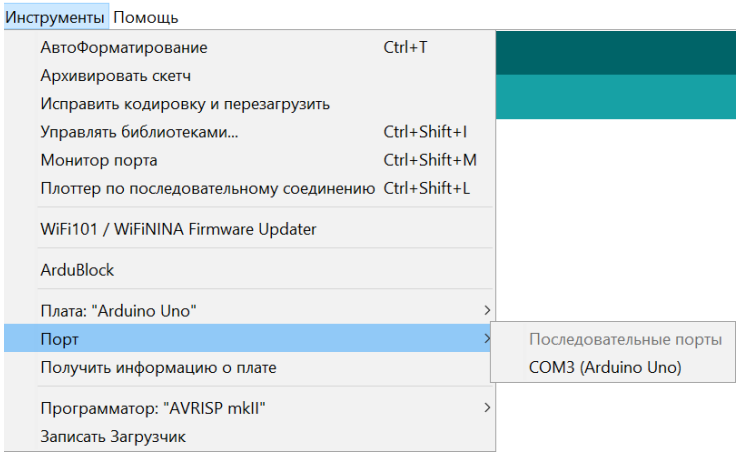


Рисунок 14

- кнопка ***Создать Новый***: этой кнопкой можно воспользоваться для создания нового файла;

- кнопки ***Открыть*** и ***Сохранить*** имеют обычное назначение: открывают существующий файл и сохраняют изменения, внесенные в открытый и редактируемый файл соответственно.

Область 3: это рабочее поле текстового редактора для набора программного кода. В области 3 показан автоматически создаваемый шаблон с обязательными структурами программного кода, к которым относятся:

- функция ***void setup()***;

- функция ***void loop()***;

Это две обязательные функции, которые должны присутствовать в каждом скетче. Все программы, которые вы будете создавать, обязательно будут иметь в своей структуре эти две функции, иначе будут ошибки при компиляции.

Пока про эти функции нужно знать следующее: функция ***setup()*** будет выполняться однократно, а функция ***loop()*** после выполнения кода, записанного в ней, от начала до конца, снова начнет выполнение кода, находящегося в её теле. Таким образом, функция ***loop()*** является по сути бесконечным циклом, и пока на плату Arduino поступает питание, функция ***loop()*** будет выполняться.

Функция ***setup()*** используется для того, чтобы указать в ней настройки оборудования, которое будет использоваться. Это можно сделать один раз, поэтому для функции ***setup()*** нет необходимости в повторении.

Область 4: Эта область в нижней части окна предназначена для вывода информационных сообщений.

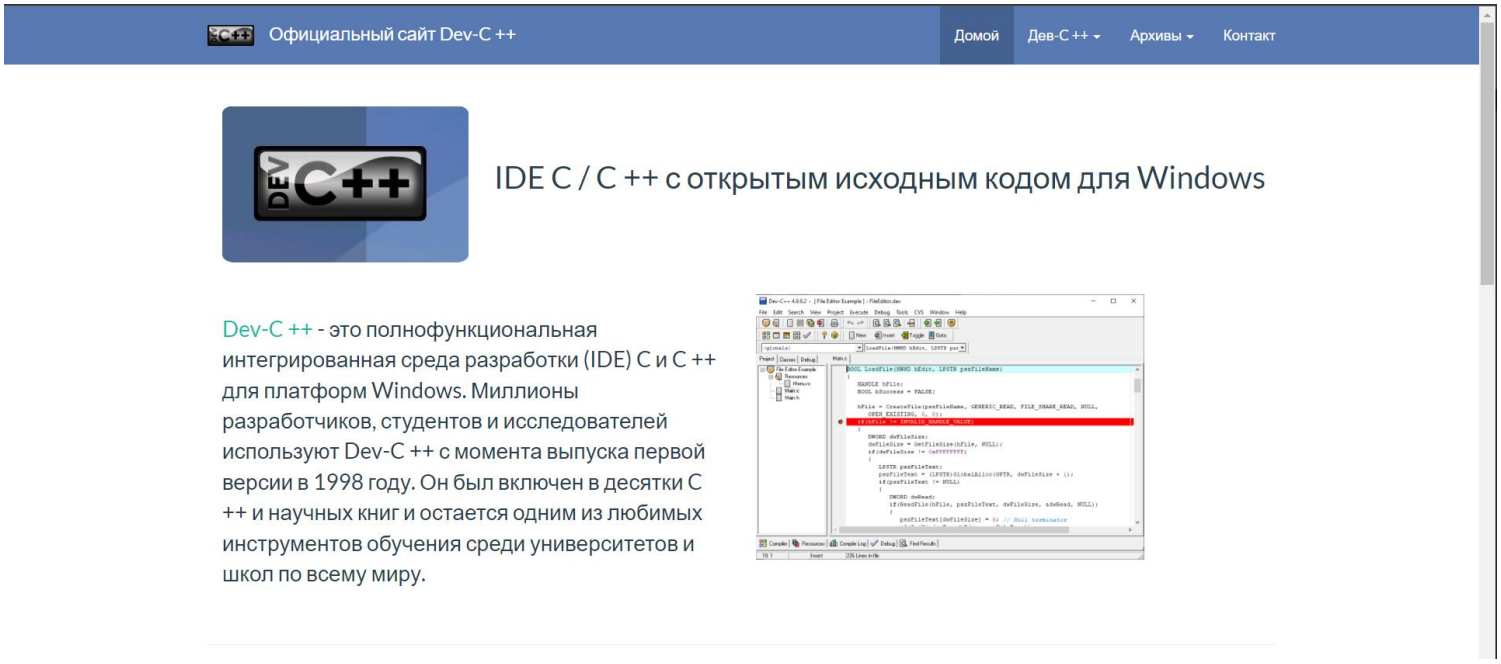


Рисунок 15

Для работы требуется скопировать код из листинга 1 и вставить его в поле размещения кода программы Embracadero Dev C++.

Листинг 1

```
/* Программа №1 - Первая C++-программа.
Введите этот программный код, затем выполните компиляцию.
*/

#include <iostream>
using namespace std;
// main() - начало выполнения программы.
int main()
{
 cout << "Hello world!";
 return 0;
```

```
}
```

При нажатии кнопки F11 на клавиатуре или кнопки «Скомпилировать и выполнить» на панели инструментов программы Dev C++ будет произведена компиляция программы – выполнение программного кода посредством вычислительных средств компьютера.

После успешной компиляции и выполнения первого примера программы, результат можно будет увидеть в всплывающем окне под названием «Консоль» представленном на рисунке 16.

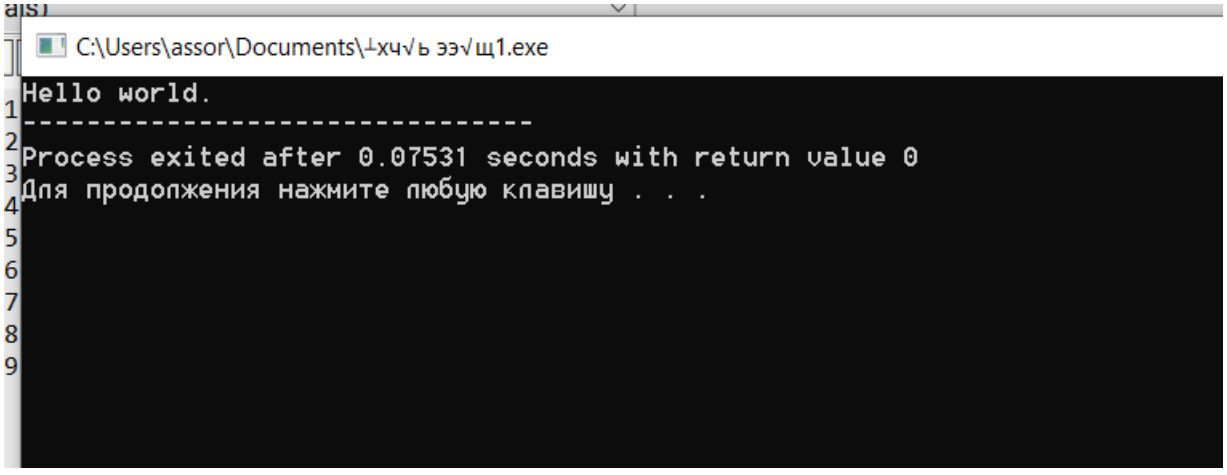
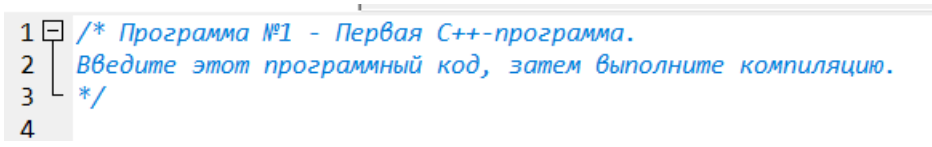


Рисунок 16

Теперь рассмотрим сам код программы подробнее:

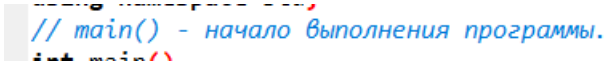
1. Символами **/**/** - обозначен многострочный комментарий, то есть это код, который не обрабатывается программой, а виден только программисту. Комментарии используются для пометок в ходе программирования или указания, для чего предназначен конкретный кусок кода.



```
/* Программа №1 - Первая C++-программа.
Введите этот программный код, затем выполните компиляцию.
*/
```

Рисунок 17

2. Символами // - обозначен однострочный комментарий, его особенность состоит в небольшом размере для написания текста - всего 255 символов.



```
// main() - начало выполнения программы.
```

Рисунок 18

3. Подключенная библиотека - ***#include <iostream>***. язык C++ содержит заголовки (**header**), представляющие информацию, необходимую для выполнения программы. Первая программа включает в себя заголовок ***<iostream>*** - библиотека для обработки данных, введенных с клавиатуры и выведенных на экран пользователя, то есть, он представляет внешний исходный файл, который помещается в начало программы с помощью **#include**.

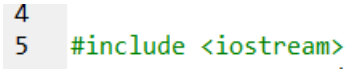


Рисунок 19

4. Пространства имен - **using namespace std;** Эта строка сообщает компилятору, что используется пространство имен **std** в виде дополнения стандарта языка C++. Пространство имен позволяет хранить одно множество имен отдельно от другого. Другими словами, имена, объявленные в одном пространстве имен, не будут конфликтовать с такими же именами, объявленными в другом. Пространства имен позволяют упростить организацию больших программ. Ключевое слово **using** информирует компилятор об использовании заявленного пространства имен (в данном случае **std**).

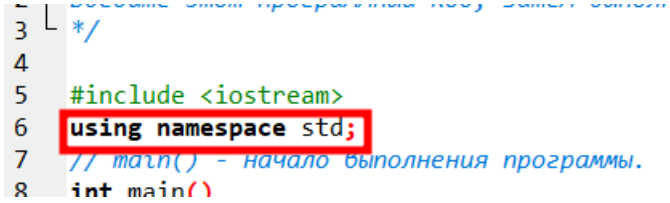


Рисунок 20

Внимание: библиотеки и пространства имен объявляются в верхней части программы!

5. Функция ***int main()*** – главная функция программы, внутри которой вызывается все остальные функции. Функции представляют собой обособленный кусок кода, вызываемый в нужный момент времени, таких кусков может быть неограниченное количество, и они могут быть взаимосвязаны между собой. Каждая функция имеет своё имя и только одна из них должна называться ***main()***, тело каждой функции обозначено фигурными скобками – {}.

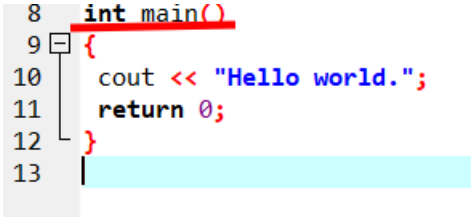


Рисунок 21

Ключевое слово ***int***, стоящие перед именем ***main()***, означает тип данных для значений, возвращаемых функцией *main()*.

6. Команда ***cout*** (**console output**)– сообщает программе, что необходимо вывести на экран сообщение «*Hello World!*», после оператор вывода «<<» обеспечивает вывод выражения, стоящего с правой стороны, на устройство, указанное с левой.

```
cout << "Hello world.";
```

Рисунок 22

7. Слово ***return*** является ключевым и возвращает значения из функции. При завершении программы без ошибок программа будет возвращать значение равное 0.

```
return 0;
```

Рисунок 23

Переменные

C++ использует возможность хранения данных в программе с помощью переменных. Каждая переменная состоит из двух частей: тип данных, имя переменной. В зависимости от этого регулируется, какая информацию может храниться в переменных разного типа.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5  int x ;
6 }
```

Рисунок 24

Рассмотрим задачу:

Необходимо создать локальную переменную с именем X и присвоить ей значение равное 20, для этого воспользуется знаком «=» для произведения операции присвоения значения. Пример программы представлен в листинге 2.

Листинг 2

```
#include <iostream>
using namespace std;
int main()
{
 int x = 20;
 cout << x; //20
 return 0;
}
```

На 5 строке кода произошла инициализация переменной X, где тип данных int (integer – тип данных, который хранит в себе целочисленные данные), далее присвоили значение равное 20.

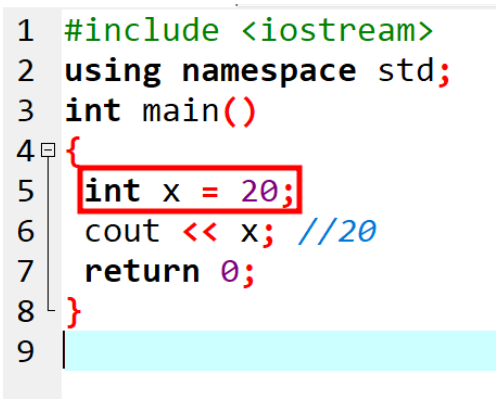


Рисунок 25

После компиляции и запуска скомпилированной программы на консоль будет выведено число 20.

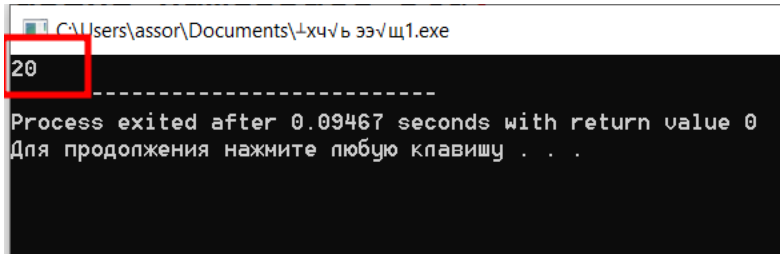


Рисунок 26

Переменная X является – ***локальной***, т.е. нельзя взаимодействовать с ней за пределами функции. Такой вид переменных указывается в функциях и не распространяется на другие функции.

Для видимости переменной из любой части программы нам придётся воспользоваться вторым видом переменных, а именно ***глобальной переменной***. Такие переменные могут вызываться в любой функции и иметь то значение, которое задали ей первоначально.

Листинг 3 содержит пример использования глобальной переменной, вынесений за пределы функции.

Рассмотрим задачу:

Необходимо создать переменную с именем X и присвоить ей значение равное 22, переменная должна быть расположена за пределами функции int main(). Пример программы представлен в листинге 3.

Листинг 3

```
#include <iostream>
using namespace std;
int x = 22; // глобальная переменная
int main()
{
 int y = 28; // локальная переменная
 cout << "X = " << x << endl; // 22
 cout << "Y = " << y; // 28
 return 0;
}
```

В данном примере инициализировали одну глобальную переменную – X, и одну локальную переменную - Y.

```cpp
#include <iostream>
using namespace std;
int x = 22; // глобальная переменная
int main()
{
 int y = 28; // локальная переменная
 cout << "X = " << x << endl; // 22
 cout << "Y = " << y; // 28
 return 0;
}
```

Рисунок 27

Для организации более удобного чтения получившегося результата на 7 строке добавили ключевое слово ***endl***, что позволяет перенести значение, выводимое на 8 строке на новую строчку в консоли.

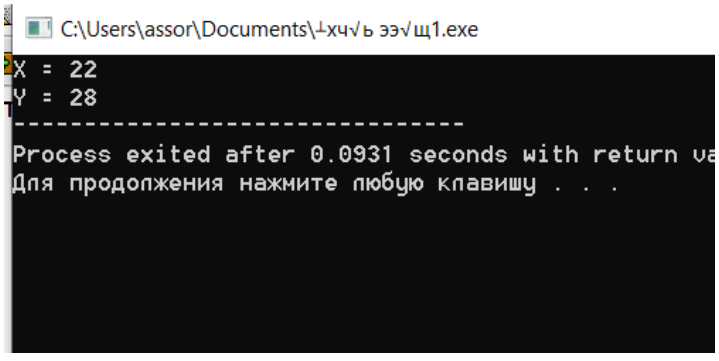


Рисунок 28

Ключевой особенностью переменных является то, что можно изменять их значения с помощью повторного присваивания значения.

Рассмотрим задачу:

Необходимо создать переменную с именем X и присвоить ей значение равное 0, после вывести значение на экран. Пользователь должен с клавиатуры ввести новое значение для переменной X равное 5, и снова вывести его на экран. Пример программы представлен в листинге 4.

Листинг 4

```
#include <iostream>
using namespace std;
int main()
{
	int x = 0;
	cout << "X = ?";
	cin >> x;
	cout << endl;
	cout << "X = " <<x;

	return 0;
}
```

На 7 строке выполняется ввод данных пользователем с клавиатуры, для этого необходимо использовать ключевое слова ***cin*** с оператором ввода «>>» и имя переменной в которую необходимо добавить значение.

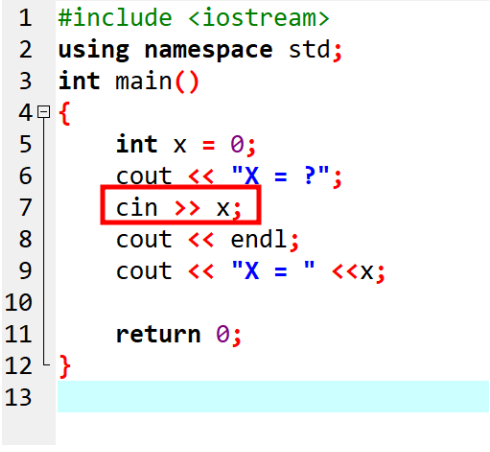


Рисунок 29

Результатом выполнения программы в консоли будет значение равное 5.

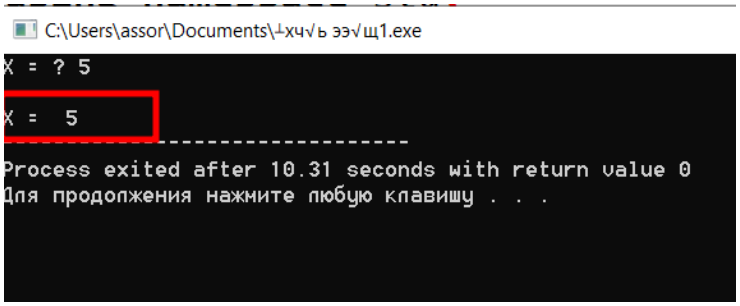


Рисунок 30

Типы данных

Каждая переменная, используемая в программе существует для выполнения конкретной цели, будь то подсчет значения или запись строки, ради этого были определены различные типы данных и количество байт занимаемое ими в памяти. Язык программирования C++ содержит типы, представленные в таблице 1.

Таблица 1

Название	Описание
bool	Логический тип. Может принимать одну из двух значений true (истина) и false (ложь). Размер занимаемой памяти для этого типа точно не определен.
char	Представляет один символ в кодировке ASCII. Занимает в памяти 1 байт (8 бит). Может хранить любое значение из диапазона от -128 до 127, либо от 0 до 255.
string	Представляет набор символов.
short	Представляет целое число в диапазоне от –32768 до 32767. Занимает в памяти 2 байта (16 бит).
int	Представляет целое число. В зависимости от архитектуры процессора может занимать 2 байта (16 бит) или 4 байта (32 бита). Диапазон предельных значений соответственно также может варьироваться от –32768 до 32767 (при 2 байтах) или от −2 147 483 648 до 2 147 483 647 (при 4 байтах).
long	Представляет целое число в диапазоне от −2 147 483 648 до 2 147 483 647. Занимает в памяти 4 байта (32 бита).
double	Представляет вещественное число двойной точности с плавающей точкой в диапазоне +/- 1.7E-308 до 1.7E+308. В памяти занимает 8 байт (64 бита)

float	Представляет вещественное число ординарной точности с плавающей точкой в диапазоне +/- 3.4E-38 до 3.4E+38. В памяти занимает 4 байта (32 бита)
void	тип без значения

Рассмотрим задачу:

Необходимо создать переменные с применением следующих типов данных: int, char, string, double, bool, после вывести значение на экран. Пользователь должен увидеть название каждой переменной, значения переменной. Каждая переменная должна выводиться с новой строки. Пример программы представлен в листинге 6.

Листинг 6

```
#include <iostream>
using namespace std;
int main()
{
      char h = 'H';
      string y = "Hello wordl!";
      int a1 = 5;
      double a2 = 2.4;
      bool c = true;
      cout << "char h = " << h << endl; // H
      cout << "string y = " << y << endl; // Hello wordl!
      cout << "int a1 = " << a1 << endl; // 5
      cout << "double a2 = " << a2 << endl; // 2.4
      cout << "bool c = " << c; // 1
      return 0;
}
```

Результат выполнения такой программы может продемонстрировать нам возможность языка оперировать различными типами данных.

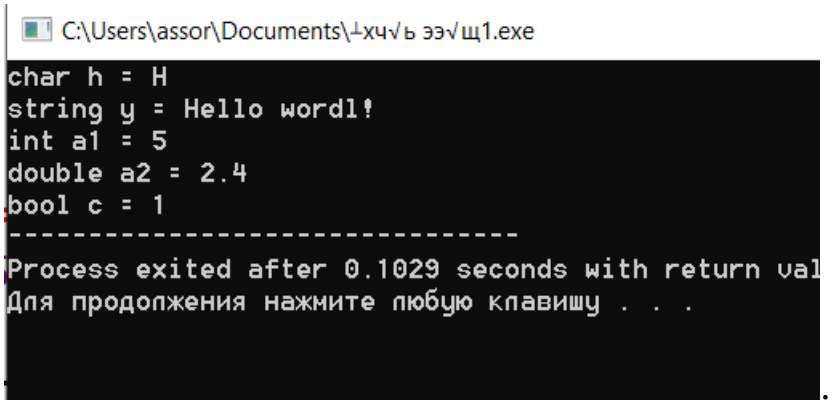


Рисунок 31

Арифметические операции

По аналогу с обычной математикой, в программировании используются арифметические операции для подсчета результата. Вычисления происходят как над целыми, так и над дробными числами. Виды операций представлены в таблице 2.

Таблица 2

Название операции	Обозначение в программе	Производимая операция
Операция сложения	+	возвращает сумму двух или более чисел.
Операцию вычитания	-	возвращает разность двух или более чисел.
Операцию умножения	*	возвращает произведение двух или более чисел.
Операцию деление	/	возвращает частное двух или более чисел.

Рассмотрим задачу:

Необходимо создать переменные X1 = 50, X2=2, Y =0, для выполнения над ними арифметический действий: сложения, умножения, вычитания и деления. Все вычисления должны происходить в переменной Y и быть составлены по формуле: Y = X1 **знак** X2. Результат каждого действия вывести на экран. Пример программы представлен в листинге 7.

Листинг 7

```
#include <iostream>
using namespace std;
int main()
{
	int x1 = 50;
	int x2 = 2;
	int y = 0;
	y = x1 + x2;
	cout << x1 << " + " << x2 << " = " << y << endl; // 52
	y = x1 - x2;
	cout << x1 << " - " << x2 << " = " << y << endl; // 48
	y = x1 * x2;
	cout << x1 << " * " << x2 << " = " << y << endl; // 100
	y = x1 / x2;
	cout << x1 << " / " << x2 << " = " << y; // 25
	return 0;
}
```

Каждая из написанных строк выполняет арифметические операции над введенными данными и выводит результат на экран.

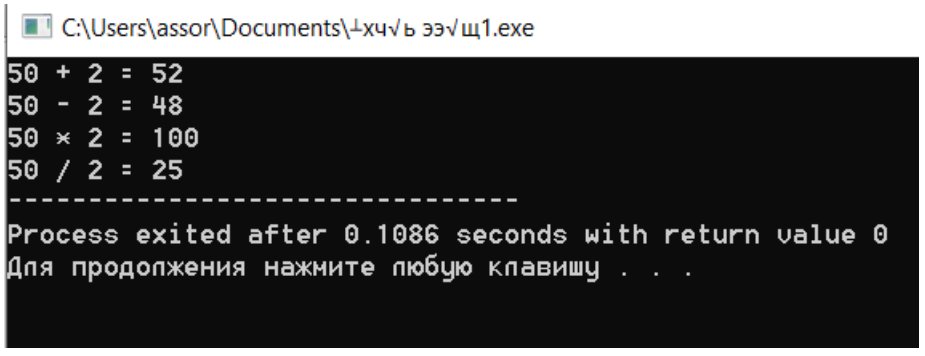


Рисунок 32

Пример решения задачи

Программа принимает 2 целых числа, далее программа должна показать произведение этих чисел, сумму и разницу в консоли. Также, необходимо посчитать среднее арифметическое этих введенных чисел.

Откройте среду Dev C++ и наберите программный код представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>

//Точка входа в программу, с неё программа начинает работать.
int main() {

    //Обьявление переменных
    int num1, num2;

    //Ввод первого числа
    std::cout << "num1: ";
    std::cin >> num1;

    //Ввод второго числа
    std::cout << "num2: ";
    std::cin >> num2;

    //вывод суммы двух чисел
    std::cout << "Сумма введенных чисел: " << num1 + num2 << std::endl;

    //Вывод разности двух чисел
    std::cout << "Разность введенных чисел: " << num1 - num2 << std::endl;

    //Вывод произведения двух чисел
    std::cout << "Произведение введенных чисел: " << num1 * num2 << std::endl;

    //Вывод среднего арифметического двух чисел
    std::cout << "Среднее арифметическое введенных чисел: " << (num1 + num2) / 2 << std::endl;

    //Обязательное указание конца программы
    return 0;
}
```

Лабораторная работа№2. Логические выражения

Логические выражения – это конструкция языка программирования, где итоговым результатом является «истина» (true) или «ложь» (false).

В C++ существует три логических оператора:

1. Логическое И (конъюнкция);
2. Логическое ИЛИ (Дизъюнкция);
3. Логическое НЕ (Отрицание).

Для начала нужно разобрать, что представляют собой логические операторы, указанные в таблице 1.

Таблица 1

Название	Обозначение в программе	Описание
Конъюнкция (логическое умножение)	&&	Выражение, где истинным считается только те случаи, где два простых выражения являются истинными, в других случаях данное сложение является ложным.
Дизъюнкция (логическое сложение)	||	Выражение, где истинным считается только те случаи, где хотя бы одно из простых логических выражений истинно и ложно тогда и только тогда, когда оба простых логических выражений ложны.
Отрицание	!	Выражение, если исходное логическое выражение истинно, то результат отрицания будет ложным, и наоборот.

Листинги 1-3 представляют из себя программы, содержащие: конъюнкцию, дизъюнкцию и отрицание.

Листинг 1

```
#include <iostream>
using namespace std;
int main()
```

```
{
	bool a = true;
	bool b = false;
	bool c = a && b;    // false
	bool d = a && true; // true

	return 0;
}
```

Листинг 2

```
#include <iostream>
using namespace std;
int main()
{
	bool a = true;
	bool b = false;
	bool c = a || b;        // true
	bool d = b || false;    // false

	return 0;
}
```

Листинг 3

```
#include <iostream>
using namespace std;
int main()
{
	bool a = true;
	bool b = !a;    // false
	bool c =!14<8;   // true

	return 0;
}
```

Операции отношений

Существует несколько типов ***операций отношений,*** представленных в таблице 2.

Таблица 2

Название операции	Обозначение в программе	Описание
Операция «равно»	==	Возвращает true, если оба операнда равны, false если они не равны.
Операция «меньше чем»	<	Возвращает true, если первый операнд меньше, чем второй, false если нет.
Операция «больше чем»	>	возвращает true, если первый операнд больше второго, false если нет:
Операция «меньше или равно»	<=	Возвращает true, если первый операнд меньше или равен второму, false если первый операнд больше второго.
Операция «больше или равно»	>=	Возвращает true, если первый операнд больше или равен второму, false если первый операнд меньше второго.
Операция «не равно»	!=	Возвращает true, если первый операнд не равен второму, и false, если оба операнда равны.

Листинги с 4-9 представляют собой примеры программ с приведёнными выше операциями.

Листинг 4

```
#include <iostream>
using namespace std;
int main()
{
      int a = 20;
      int b = 8;
      bool c = a == b;    // false
      bool d = a == 20;   // true
```

```
	return 0;
}
```

Листинг 5

```
#include <iostream>
using namespace std;
int main()
{
	int a = 20;
	int b = 8;
	bool c = a < b;	// false

	return 0;
}
```

Листинг 6

```
#include <iostream>
using namespace std;
int main()
{
	int a = 20;
	int b = 8;
	bool c = a > b;	// true

	return 0;
}
```

Листинг 7

```
#include <iostream>
using namespace std;
int main()
{
	int a = 10;
	int b = 10;
	bool c = a <= b;	// true
```

```
	return 0;
}
```

Листинг 8

```
#include <iostream>
using namespace std;
int main()
{
	int a = 2;
	int b = 8;
	bool c = a >= b;	// false

	return 0;
}
```

Листинг 9

```
#include <iostream>
using namespace std;
int main()
{
	int a = 2;
	int b = 8;
	bool c = a != b;	// true
	bool d = 10 != 10;	// false

	return 0;
}
```

Пример решения задачи

Программа принимает целочисленное число, она должна определить чётность числа и вывести результат на экран.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
```

```
#include <iostream>

int main() {
    //Обьявление переменных
    int num1;

    //Ввод первого числа
    std::cout << "num1: ";
    std::cin >> num1;

    //Проверяем остаток от деления на 2
    //Если остаток от деления равен 0, то число чётное
    if(num1 % 2 == 0){
      std::cout << "Введённое число - чётное" << std::endl;
      //Иначе нечётное
    }
    else
    {
            std::cout << "Введённое число - нечётное" << std::endl;
    }
    return 0;
}
```

Лабораторная работа №3. Оператор условия

Оператор условия - конструкция языка программирования, направленная на выбор одного из возможных путей в зависимости от условия. В C++ определяется ключевым словом **if**.

Конструкция оператора условия представлена в листинге 1.

Листинг 1

```
#include <iostream>
using namespace std;
int main()
{
     if (условие)
     {
```

```
        //Инструкция;
    }

    return 0;
}
```

В качестве ***условия*** используется логическое или условное выражение, которое возвращает **true** или **false**. Условный оператор выполняет команды, когда логическое выражение в условии удовлетворяет истинности.

Для наглядности как это работает рассмотрим небольшую блок схему:

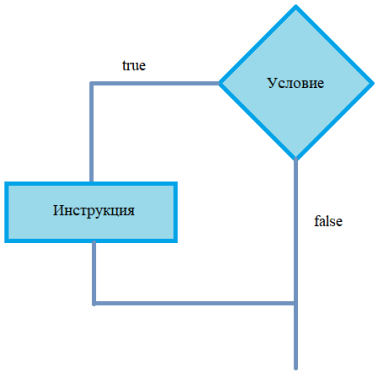


Рисунок 1

Рассмотрим задачу:

Необходимо сравнить два числа A =15 и B = 20 в случае, если A меньше B, требуется вывести на экран сообщение «A > B».

Для этого воспользуемся оператором **if** и сделаем проверку обозначенных переменных. Пример кода представлен в листинге 2.

Листинг 2

```
#include <iostream>
using namespace std;
int main()
{
    int a = 15;
    int b = 20;
```

```
    if (a < b)
    {
      cout << " A < B";
    }

    return 0;
}
```

В экране консоли при введенных данных получим ожидаемое сообщение.

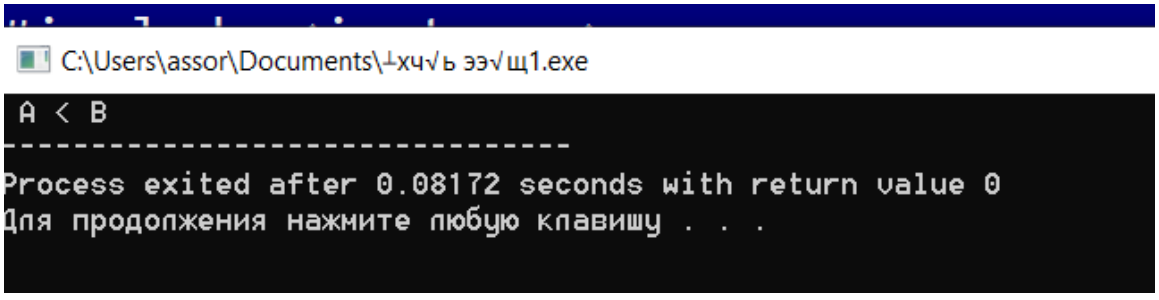


Рисунок 2

В зависимости от сложности задачи требуется более подробная проверка, сообщающая пользователю или программисту что делать в случае, если первое условие не подошло. Такая конструкция называется ***if..else***, обозначающая возможность вызова инструкции при получении отрицательного результата, то есть если условие было выполнено, выполняется 1 инструкция, если нет, то 2 инструкция.

Рассмотрим, как это выглядит в блок схеме:

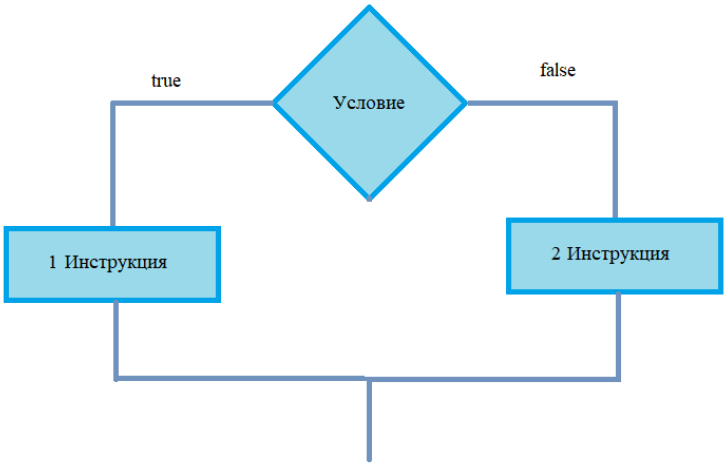


Рисунок 3

Рассмотрим задачу:

Необходимо сравнить два числа A =15 и B = 20 в случае, если A меньше B, требуется вывести на экран сообщение «A > B», иначе «B < A».

Для этого воспользуемся оператором **if..else** и сделаем проверку обозначенных переменных. Пример кода представлен в листинге 3.

Листинг 3

```
#include <iostream>
using namespace std;
int main()
{
      int a = 15;
      int b = 20;
      if (a < b)
      {
        cout << "a < b";
      }
      else
      {
        cout << "b < a";
      }

      return 0;
}
```

При введенных данных, не смотря на наличие ветки условия с отрицательным результатом, получим сообщение, что «A < B».

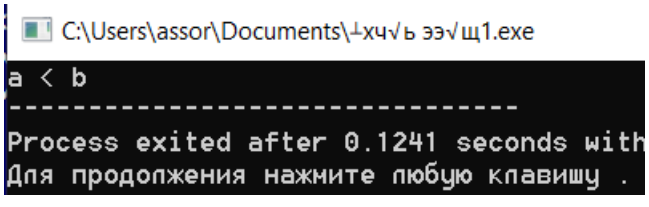


Рисунок 4

Пример решения задачи

Программа принимает целочисленное число, программа должна прибавить 1 к положительному числу, иначе не изменять число.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>

int main() {
    //Объявление переменных
    int num1;

    //Ввод числа с клавиатуры
    std::cout << "num1: ";
    std::cin >> num1;

    //Если число больше, то увеличиваем число на 1
    if(num1 > 0){
      //увеличение на 1
      num1++;
    }
    return 0;
}
```

Лабораторная работа №4. Тернарный оператор

Для увеличения скорости программирования, в противовес конструкции ***if..else***, был создан ***тернарный оператор***, имеющий сокращенную форму вызова, но использующий проверку такого же типа: если условие ***true***, то выполняется ***выражение1***, если ***false*** выполняется **выражение2.**

Синтаксис вызова оператора выглядит следующим образом:

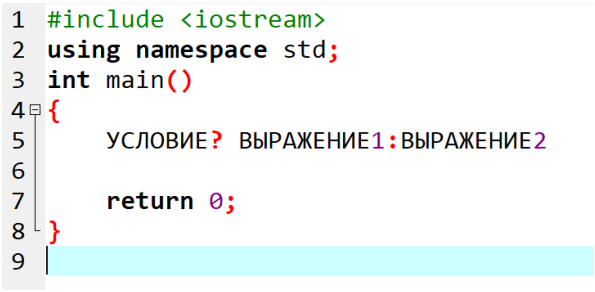


Рисунок 1

Рассмотрим задачу:

Необходимо сравнить два числа A =40 и B = 23. В случае, если A больше или равно B, требуется вывести на экран значение равное 4, иначе значение равное 2.

Для этого воспользуемся оператором тернарным и сделаем проверку обозначенных переменных. Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
using namespace std;
int main()
{
      int a = 40;
      int b = 23;
      int c = (a >= b)?4:2;
      cout << c; // 4

      return 0;
}
```

На 7 строке кода можно заметить, что для удобства разделения условия и проверки, условие было взято в круглые скобочки, а само значение присвоено переменной C. Такая форма написания позволяет сократить ошибки при использовании тернарного оператора.

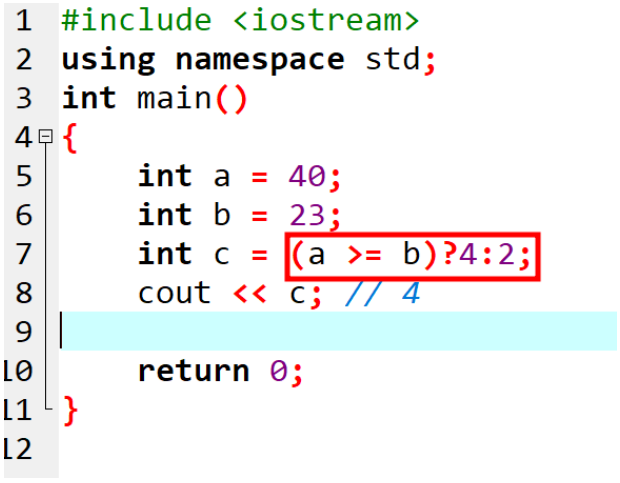


Рисунок 2

Пример решения задачи

Программа принимает целочисленное число, программа должна вывести больше ли нуля число или нет.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>

int main() {
    //Объявление переменных
    int num1;

    //Ввод числа
    std::cout << "num1: ";
    std::cin >> num1;

    //Тернарный оператор, проверяет если число больше 0 то выводит, что оно больше, иначе выводит, что оно меньше или равно 0
    std::cout << num1 > 0 ? "Больше 0" : "Меньше или равно 0";
    return 0;
}
```

Лабораторная работа №5. Оператор выбора

На языке программирования C++ существует оператор выбора из нескольких значений под названием ***switch***. В зависимости от условия, будем получать значение, которое будет выполнять ту или иную инструкцию.

Для наглядности как работает оператор выбора, рассмотрим блок схему:

Рисунок 1

Каждая из инструкций содержит ключевое слово ***case***, после которого указывается необходимое значение для работы и ставиться двоеточие, после него раскрывается тело инструкции, которое необходимо выполнить и ключевое слово ***break***, для завершения такой операции. В случае, если ни одно из получившихся значений не удовлетворит условию, попадаем в раздел программы с названием **default**, и выводим пользователю либо какое-то сообщение, либо вызов предписанной инструкции.

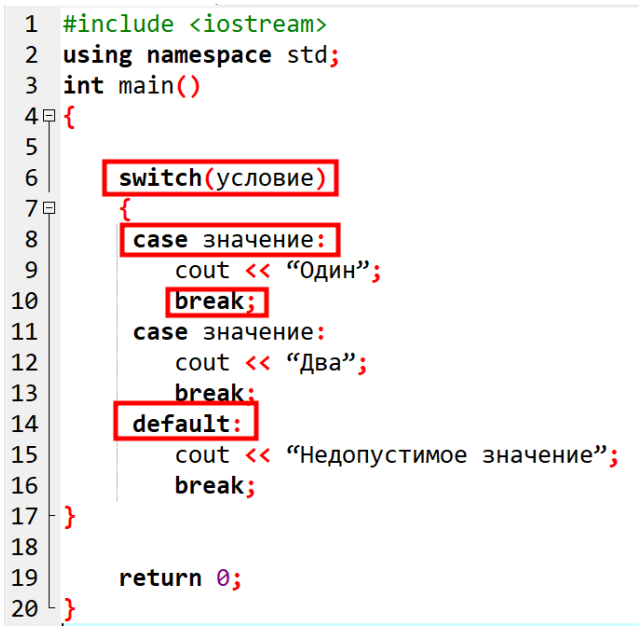


Рисунок 2

Рассмотрим задачу:

Необходимо вывести на экран значения: 1,2 или No, в зависимости от переменной A. Если переменная A = 1, вывести 1; если A = 2, вывести 2; если равна любому другому значению вывести сообщение: No.

Для этого воспользуемся оператором **switch** и сделаем проверку обозначения переменных. Пример кода представлен в листинге 1

Листинг 1

```
#include <iostream>
using namespace std;
int main()
{

      int a = 2;
      switch(a)
      {
       case 1:
          cout << "1";
```

```
            break;
        case 2:
            cout << "2";
            break;
        default:
            cout << "No";
            break;
       }

       return 0;
}
```

Для проверки данных было установлено значение 2, как результат инструкции получается на экране результат из case 2, равное 2.

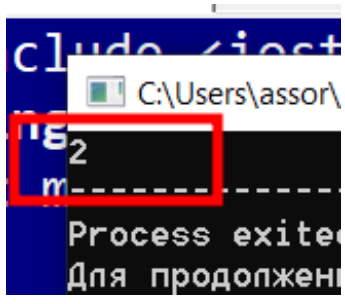


Рисунок 3

Пример решения задачи

Программа принимает целочисленное число от 1 до 7, программа должна вывести день недели в соответствии с днем недели.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>

int main() {

    //Объявление переменных
    int num1;

    //Ввод числа
```

```
    std::cout << "num1: ";
    std::cin >> num1;

    //Оператор выбора, в зависимости от введенного числа будет выполнять тот или иной case.
    switch (k)
    {
      case 1:
           std::cout << "Понедельник\n";
           break;
      case 2:
           std::cout << "Вторник\n";
           break;
      case 3:
           std::cout << "Среда\n";
           break;
      case 4:
           std::cout << "Четверг\n";
           break;
      case 5:
           std::cout << "Пятница\n";
           break;
      case 6:
           std::cout << "Суббота\n";
           break;
      case 7:
           std::cout << "Воскресенье\n";
           break;

      //Если не одно из чисел не подходит, будет выполняться default
      default: std::cout << "Нет такого дня недели.\n";
    }
    return 0;
}
```

Лабораторная работа №6. Циклы с предусловием

Цикл – многократное выполнение одного и того же кода программы, до тех пор, пока условие не будет выполнено.

Существует 3 вида циклов:

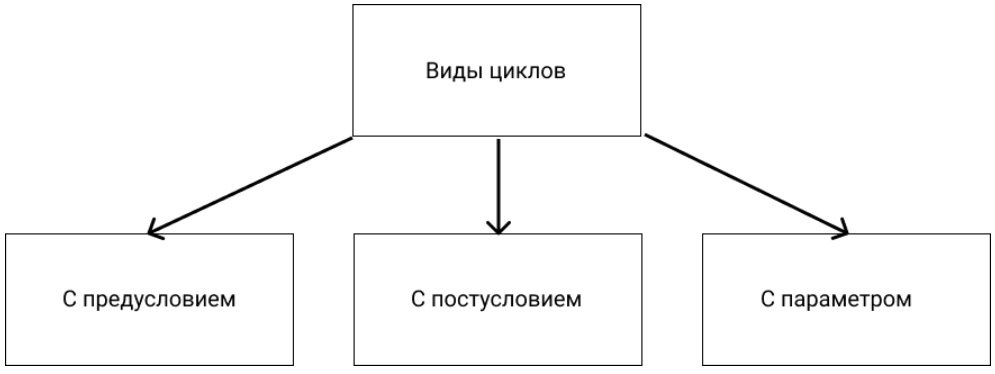


Рисунок 1

Цикл с предусловием – цикл, который будет выполнять свои инструкции, пока условие является истинным, указанное в начале цикла.

Структура цикла с предусловием выглядит следующим образом: после ключевого слова ***while*** в скобках указывается условие, которое возвращает ***true*** или ***false***. Если условие возвращает ***true***, тогда будет выполнять инструкцию данного цикла.

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5
6      while(условие)
7      {
8       Инструкция;
9      }
10     return 0;
11 }
12
```

Рисунок 2

Рассмотрим задачу:

Необходимо выводить на экран значения произведения А на А, до того момента, пока А меньше или равна 5, каждый шаг цикла к А прибавляется +1.

Для решения этой задачи воспользуемся циклом while и присвоим изначальное значение А = 1. Пример кода представлен в листинге 1.

Листинг 1

```
int a = 1;
    while(a <= 5)
    {
        cout << a << " * " << a << " = " << a * a << endl;
        a++;
    }

    return 0;
```

Данный цикл будет выполняться до тех пор, пока А меньше или не будет равна 5. Пока условие ***while*** равняется истине, у нас будет выполняться следующая инструкция: в консоль будет выводиться число А умноженное на А, и после подсчета А будет увеличиться на 1 благодаря ***инкременту***.

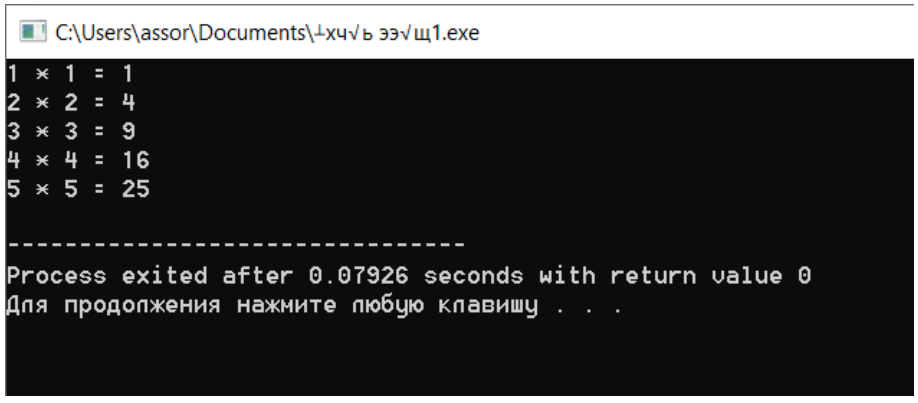


Рисунок 3

Инкремент – операция, увеличивающая значение переменной на единицу. Обозначение А++.

Пример решения задачи

Программа принимает целочисленное число A, которая больше 1. Программа должна вывести значение по следующей формуле: 1 + 2 + . . . + A.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>

int main() {
    //Объявление переменных
    int A;
    int k = 0;
    int summ = 0;

    //Ввод числа
    std::cout << "A: ";
    std::cin >> A;

    //Цикл с предусловием, в зависимости от поставленного условия будет выполнять до тех пор, пока условие является true.
    while (k<=a)
    {
       //увеличение на 1
       ++k;

       //прибавляем к переменной summ значение переменной k
       summ+=k;
    }
    std::cout << "Summ: " << summ;
    return 0;
}
```

Лабораторная работа №7. Циклы с постусловием

Цикл с постусловием – цикл, который выполняет свои инструкции, а потом проверяет условие, пока условие истинно, то цикл будет повторяться.

Структура цикла с предусловием выглядит следующим образом, в начале указывается ключевое слово ***do***, говорящие о начале цикла с постусловием, после выполняется инструкция цикла. В конце указывается ключевое слово ***while*** и условие для цикла. Инструкция будет выполняться до тех пор, пока условие не будет равняться ***false***.

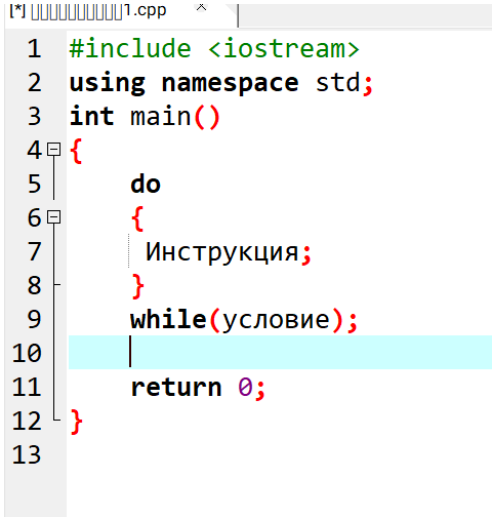


Рисунок 1

Рассмотрим задачу:

Дано число A равное 10. Выводить на экран значение переменной A, до тех пор, пока оно не будет меньше нуля. Для решения задачи необходимо использовать декремент и цикл **do..while**. Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
using namespace std;
int main()
{
      int a = 10;
      do
      {
```

```
            cout << "a = "<< a-- << endl;
        }
        while(a > 0);
        return 0;
}
```

Цикл будет выполняться до тех пор, пока A не станет равной 0. Пока условие равняется истине, у нас будет выполняться следующая инструкция, в консоль будет выводиться число *A*, которое будет уменьшаться после вывода благодаря ***декременту*** каждый шаг цикла.

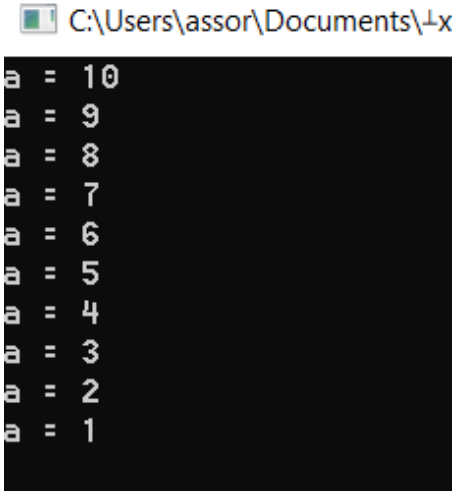


Рисунок 2

Декремент – это обратная операция инкремента, которая уменьшает значение переменной на единицу. Обозначение A--.

Пример решения задачи

Программа принимает целочисленное число A, которая больше 1. Программа должна вывести значение по следующей формуле: 1 + 2 + . . . + A.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>

int main() {
    //Объявление переменных
    int A;
    int k = 0;
    int summ = 0;

    //Ввод числа
    std::cout << "A: ";
    std::cin >> A;

    //Цикл с постусловием, выполнит заданные действия, после этого проверит условие, действие
будет повторяться до тех пор, пока условие выполняется.
    do
    {
       //увеличение на 1
       ++k;

       //прибавляем к переменной summ значение переменной k
       summ+=k;
    } while (k<=a);

    std::cout << "Summ: " << summ;
    return 0;
}
```

Лабораторная работа №8. Циклы с параметром

Цикл с параметром – цикл со счетчиком, он имеет следующую структуру:

1. *Выражение1* выполняется один раз при начале выполнения цикла и представляет установку начальных условий, как правило, это инициализация счетчиков - специальных переменных, которые используются для контроля за циклом;
2. *Выражение2* представляет условие, при соблюдении которого выполняется цикл;
3. *Выражение3* задает изменение параметров цикла, то есть происходит увеличение счетчиков цикла с использованием инкремента или уменьшение счетчика с использованием декремента.

```
#include <iostream>
using namespace std;
int main()
{
    for(выражение1;выражение2;выражение3)
    {
     Инструкция;
    }

    return 0;
}
```

Рисунок 1

Рассмотрим задачу:

Дано число A равное 0. Выводить на экран значение переменной A, до тех пор, пока оно не будет больше 10. Для решения задачи необходимо использовать цикл for. Пример кода представлен в листинге 1.

Листинг 1

```
for(int a=0; a < 10; a++)
{
  cout << "a = " << a << endl;
}
return 0;
```

В первом выражении инициализировали переменную A и задали начальное значение 0, данная переменная может использоваться только внутри цикла **for**. Во втором выражении указали условия выполнения цикла. В третьем выражении указано как будет изменяться переменная A.

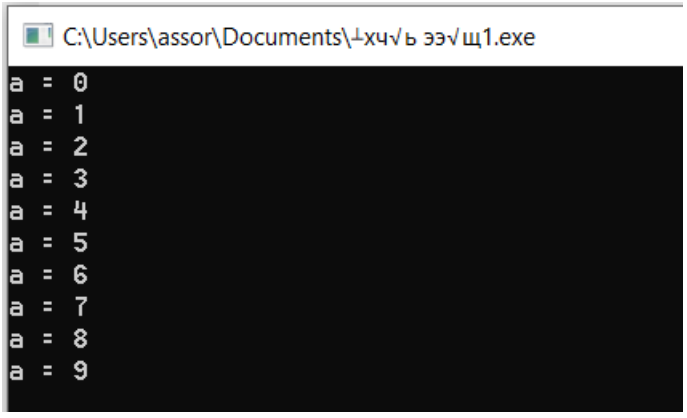


Рисунок 2

Будем использовать инкремент для увеличения переменной на единицу, но возможны и другие способы увеличения или уменьшения значения, например, *a=a+2* или *a+=2*, где значение переменной *a* будет увеличивать на 2.

Пример решения задачи

Программа принимает целочисленное число A, которая больше 1. Программа должна вывести значение по следующей формуле: 1 + 2 + . . . + A.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

Листинг 1

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>

int main() {

    //Объявление переменных
    int A;
    int summ = 0;

    //Ввод числа
    std::cout << "A: ";
    std::cin >> A;

    //Цикл с параметром, выполняет заданные действия, до тех пор, пока переменная i, выполняет условие
```

```
    for(int i = 1; i <= A; i ++)
    {
       //прибавляем к переменной summ значение переменной i
       summ+=i;
    }

    std::cout << "Summ: " << summ;
    return 0;
}
```

Лабораторная работа №9. Вложенные циклы

Вложенные циклы – циклы, имеющие в теле одного цикла другой. Чаще всего используется при работе с математическими матрацами для ввода, вывода и перебора данных.

Рассмотрим задачу:

Дано число А равное 1. С помощью вложенных циклов необходимо вывести на экран значения от 1 до 3 в виде матрицы размером 3 на 3. Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
using namespace std;
int main()
{
      int a = 1;
      for(int i=0; i<3; i++)
      {
            for(int j=0; j<3; j++)
            {
              cout << a << " ";
              a++;
            }
       cout << endl;
      }

      return 0;
}
```

Данный программный код выполняет вывод значения А, где первый цикл, перебирающий значения «i» - будет отвечать за вывод значения в строчку, а второй цикл, перебирающий значения «j» - отвечает вывод значения в столбец.

Отображение благодаря сочетанию команд **end** и **endl** будет в виде матрицы размером 3 на 3:

1 2 3

4 5 6

7 8 9

Рисунок 1

Операторы break и continue

При работе с циклами бывает необходимо использовать принудительное завершение вычисления или пропуск действия для определённого условия.

Оператор ***break*** – немедленно завершает выполнения инструкции, то есть, если программа увидит оператор ***break*** - он прекратит выполнение последующего кода.

Рассмотрим задачу:

Дано число i равное 0. К переменной i с помощью цикла for требуется прибавлять значение +1, пока i не станет равной 10. Прервать выполнение программы, если i будет равна 4. Пример кода представлен в листинге 2.

Листинг 2

```
#include <iostream>
using namespace std;
int main()
{
      for(int i=0; i<10; i++)
      {
        cout << "i = " << i << endl;
        if (i == 4)
                 break;
      }

      return 0;
}
```

Последний результат вывода программы равен 4, в связи с нахождением в цикле ***if*** оператора прерывания ***break***.

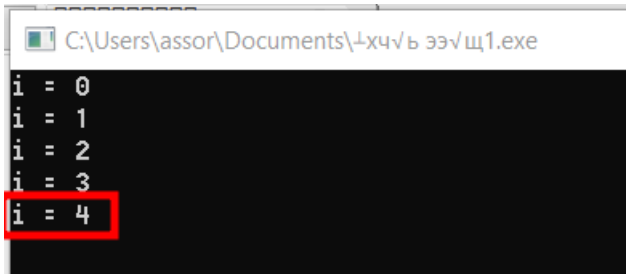


Рисунок 2

Оператор ***continue*** – пропустит текущей действие цикла.

Рассмотрим задачу:

Дано число i равное 0. К переменной i с помощью цикла ***for*** требуется прибавлять значение +1, пока i не станет равной 10. Пропустить действие программы, если i будет равна 4. Пример кода представлен в листинге 3.

Листинге 3

```
#include <iostream>
using namespace std;
int main()
{
      for(int i=0; i<10; i++)
      {
       if (i == 4)
         continue;
      cout << "i = " << i << endl;
      }

      return 0;
}
```

В данном примере будет выводиться значение i и ставим условие, если значение i будет равна 4, тогда цикл приврётся и перейдет на следующею итерацию.

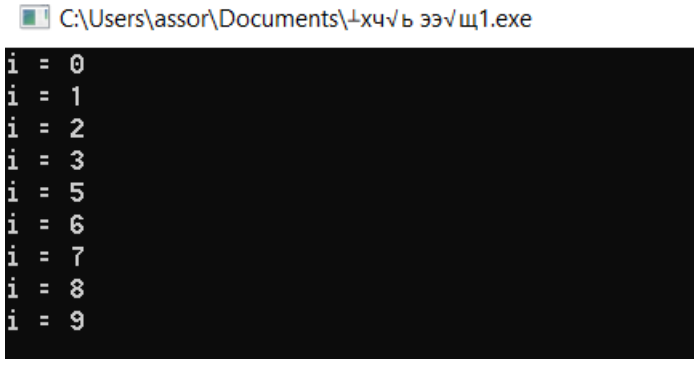


Рисунок 3

Пример решения задачи

Программа принимает целочисленное число x, n и j. Программа должна вывести число x виде куба.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>

int main() {

    //Объявление переменных
    int x;
    int n,j;
    //Ввод числа
    std::cout << "x: ";
    std::cin >> x;
    std::cout << "n: ";
    std::cin >> n;
    std::cout << "j: ";
    std::cin >> j;

    //Выводим значение переменной x виде куба
    for(int i = 0; i < 3; i ++)
    {
       for(int j = 0; i < 3; i ++)
       {
```

```
            std::cout << x;
        }
        std::cout << endl;
    }
    return 0;
}
```

Лабораторная работа №10. Подпрограммы (функции или методы)

Подпрограммы или методы – это именованный блок кода, который выполняет программа.

Оформление метода выглядит следующим образом:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      тип_данных имя_метода(параметры)
6      {
7          инструкции
8      }
9
10     return 0;
11 }
12
```

Рисунок 1

Подробно рассмотрим возможности методов:

1. В типе данных указывается тип возвращающего метода, то есть тот тип значения, который необходим для использования в главном методе или в других методах, это может быть тип данных int, double, void и т.д.;
2. Метод, который возвращает нам значение называется функцией, а метод, который нам ничего не возвращает называется процедурой;
3. Имя функции/процедуры указывается для вызова данной функции;
4. В параметрах указывается значение, которое будет принимать наш метод. Параметр может быть пустым, это означает что метод не имеет параметров для вызова;
5. В инструкции прописывается те действия, которые будет выполнять метод.

Рассмотрим задачу:

Дано число X равное 0. Ввести новое значение X с клавиатуры для расчета факториала этого числа Пример кода представлен в листинге 1.

Листинге 1

```
#include <iostream>
using namespace std;
int x = 0;
int fact()
{
	cout << "Введите число для расчёта факториала: ";
	cin >> x;
	int factorial = 1;
	for (int i = 1; i <= x; i++)
	{
		factorial = factorial * i;
	}
	return factorial;
}

void infoFact(int x, int ResultFactorial)
{
	cout << "Факториал " << x << " = " << ResultFactorial;
}

int main()
{
	setlocale(LC_ALL, "Russian");
	int a = fact();

	infoFact(x, a);
	return 0;
}
```

Объявлены два метода и одна глобальная переменная *x*, первый метод ***intFact*** имеет тип возвращающего метода, который вернет значение типа данных int, второй метод *void* ***intFact*** имеет тип не возвращающего метода.

Рассмотрим первый метод ***intFact***. В данную функцию не посылается никаких параметров. При вызове данной функции пользователя просят вести число, для которого необходимо рассчитать факториал, и с помощью цикла ***for*** рассчитывается факториал числа указанным пользователем, возвращаем результат и записываем в переменную ***A***.

Второй метод имеет два параметра для вызова. Первый параметр отвечает за значение который вел пользователь, а второй за результат вычисления факториала.

В главном методе в первую очередь используется ***setlocale(LC_ALL, "Russian")*** для вывод русских символов в консоль. После этого инициализируем переменную *A* в которую запушиться результат метода ***fact*** и вызываем метод для вывода пользователю результата подсчета.

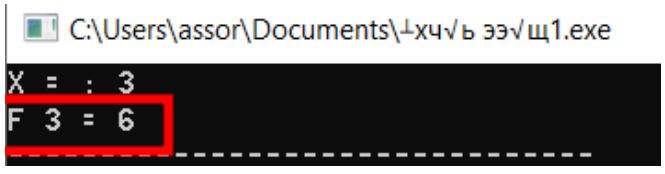


Рисунок 2

Пример решения задачи

Программа принимает трехзначное целочисленное число, программа должна вывести сумму и разность его чисел.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
using namespace std;
int main() {

    //Объявление переменных
    int A;
    int b,c,d;
    int sum, rasn;

    //Вводим значение A
    cout << "A: ";
```

```
    cin >> A;

    //Вызов функции fuct и передача параметром переменную A
    fuct(A);
    return 0;
}

//Инициализация функции fuct
void fuct(int A) {

    //Производим разделение трехзначного числа
    b = A % 10;
    c = A % 100 / 10;
    d = A / 100;

    //Суммируем и вычитаем получившиеся числа
    sum = b + c + d;
    rasn = b – c – d;

    //Выводим ответ
    cout << "sum = " << sum << endl;
    cout << "rasn = " << rasn;
}
```

Лабораторная работа №11. Перезагрузка методов

Перезагрузка методов представляет из себя использование методов с одинаковыми названиями, но имеющие различные параметры.

Рассмотрим задачу:

Необходимо создать два метода с одинаковым именем ***sum***. Каждый из методов имеет собственное назначение:

1. Получает один параметр ***int x***. Возвращает результат действия: x +5.
2. Получает два параметра ***int x*** и ***int y.*** Возвращает результат действия: x +y.

Пример кода представлен в листинге 1.

Листинге 1

```
#include <iostream>
using namespace std;
int sum(int x)
{
     return x + 5;
}

int sum(int x, int y)
{
     return x + y;
}

int main()
{
     cout << "1 = " << sum(4) << endl;
     cout << "2 = " << sum(4, 10);
     return 0;
}
```

В данном примере инициализированы 2 метода с одинаковыми названиями, но имеющие разные параметры для ввода. Тем самым, в зависимости от количества отправленных параметров будет использоваться тот или иной метод.

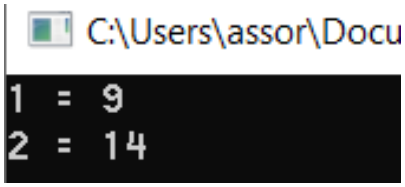


Рисунок 1

Пример решения задачи

Программа принимает два целочисленных числа, программа должна суммировать два числа, если одно значение равна 0, тогда должно выполняться и использованием статического числа, если нет, то два значения суммируются.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
using namespace std;

//Инициализация функции sum
int sum(int num1) {
    return num1 + 20;
}

//Инициализация функции sum
int sum(int num1, int num2) {
    return num1 + num2;
}

int main() {
    //Обьявление переменных
    int num1, num2;

    //Вводим значения
    cout << "num1: ";
    cin >> num1;
    cout << "num2: ";
    cin >> num2;

    //Проверяем значения переменных
    if (num1 != 0 && num2 != 0)

    //Выводим получившейся результат
    cout << num1 << " + " << num2 << " = " << sum(num1,num2);
    else if (num1 == 0)
```

```
    //Выводим получившейся результат
    cout << num1 <<" + 20" << " = " << sum(num1);
    else if (num2 == 0)

    //Выводим получившейся результат
    cout << num2 <<" + 20" << " = " << sum(num2);
    return 0;
}
```

Лабораторная работа №12. Одномерные массивы. Форматированный вывод элементов массива

Массив – это набор данных, имеющий один тип данных. В программе он инициализируется следующим образом:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      тип_данных название_массива [длина_массива]
6  }
```

Рисунок 1

В программе это можно реализовать несколькими способами:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      // с указанием размера массива
6      int numbers[4];
7      // с указанием размера массива и его содержимого
8      int numbers[4] = {1,2,3,4};
9      // с указанием его содержимого
10     int numbers[] = {1, 2, 3, 4, 5, 6};
```

Рисунок 2

Рассмотрим задачу:

Необходимо создать одномерный массив, где его длинна будет равна 4 значениям. Данные вводятся с клавиатуры и выводятся на экран. Пример кода представлен в листинге 1.

Листинге 1

```
#include <iostream>
using namespace std;
int main()
{
      int arr[4];
      cout << "Input array"<<endl;
      for (int i = 0; i < 4; i++)
      {
        cout << "arr[" << i << "] = ";
        cin >> arr[i];
      }
      cout << "Output array" << endl;
      for (int i = 0; i < 4; i++)
        cout << "arr[" << i << "] = " << arr[i] << endl;

      return 0;
}
```

В начале программы инициализировали массив, который принимает 4 значения. У каждого значения есть свой индекс, по которому можно обращаться к данному значению. Индекс начинается с 0, наш массив имеет следующие индексы: 0, 1, 2 и 3, по которым можно вызвать то или иное значение из массива. После это просим пользователя вести в каждую ячейку массива свое значение и показываем ему под каким индексом будет храниться значение, которое он ввел. Во втором цикле выводим содержимое массива.

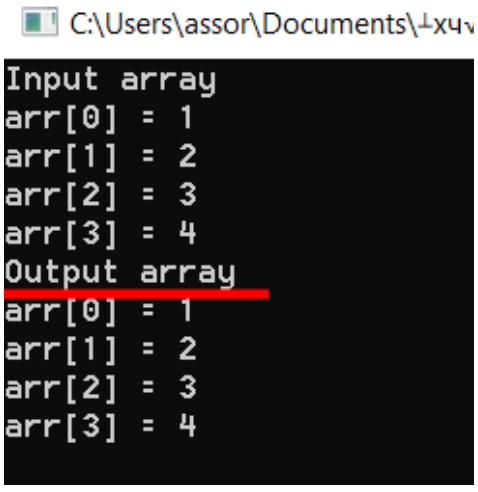


Рисунок 3

Пример решения задачи

Программа принимает целочисленные числа в одномерный массив, программа должна инициализировать одномерный массив длинной 5, и вывести одномерный массив.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
using namespace std;

int main() {

    //инициализируем одномерный массив
    int arr[5];
    cout << "Input array"<<endl;

    //Вводим значения в одномерный массив
    for (int i = 0; i < 5; i++)
    {
        cout << "arr[" << i << "] = ";
        cin >> arr[i];
    }
    cout << "Output array" << endl;

    //Выводим значения в одномерный массив
```

```
    for (int i = 0; i < 5; i++)
        cout << "arr[" << i << "] = " << arr[i] << endl;

    return 0;
}
```

Лабораторная работа №13. Одномерные массивы. Анализ элементов массив

С помощью анализа элементов массива возможно выполнять следующие действия:

1. Подсчет данных в массиве;
2. Нахождение минимально и максимального значения.

Рассмотрим задачу:

Необходимо создать одномерный массив, где его длинна будет равна 4 значениям. Данные вводятся с клавиатуры, произвести суммирование всех элементов массива. Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
using namespace std;

int main()
{
      int arr[4];
      cout << "Input array"<<endl;
      for (int i=0; i<4; i++)
      {
        cout << "arr[" << i << "] = ";
        cin >> arr[i];
      }
      int sum = 0;
      // производим суммирование всех значений в массиве
      for (int i = 0; i < 4; i++)
        sum += arr[i];
           cout << "Sum array = " << sum << endl;
      return 0;
}
```

Для суммирования значений в массиве инициализируем переменную *sum* и с помощью цикла считаем сумму значений в массиве.

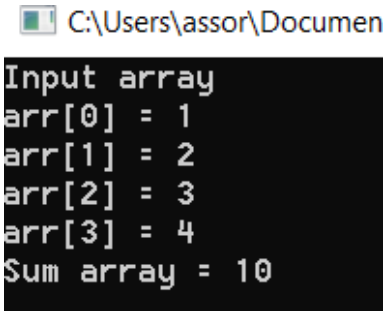


Рисунок 1

Рассмотрим задачу:

Необходимо создать одномерный массив, где его длинна будет равна 4 значениям. Данные вводятся с клавиатуры, найти минимальное и максимальное значения массива, и их индексы. Пример кода представлен в листинге 2.

Листинг 2

```
#include <iostream>
using namespace std;

int main()
{
          int arr[4];
     cout << "Input array"<<endl;
     for (int i = 0; i < 4; i++)
     {
       cout << "arr[" << i << "] = ";
       cin >> arr[i];
     }
     int min = arr[0]; int index_min = 0;
     int max = arr[0]; int index_max = 0;
     // нахождение максимального и минимального элемента массива
     for (int i = 0; i < 4; i++)
     {
        if (max < arr[i])
        {
           max = arr[i];
           index_max = i;
```

```
        }
        if (min > arr[i])
        {
            min = arr[i];
            index_min = i;
        }
    }
    cout << "Max = " << max << "  ;i = " << index_max<< endl;

    return 0;
}
```

Для нахождения максимально и минимального значения числа инициализируем 4 переменные, две из которых будут отвечать за сами значения минимума и максимума, а две другие за их расположения.

Задаем переменным ***min*** и ***max*** первый элемент в массиве для сравнения с другими значениями. Для переменных ***index_min*** и ***index_max*** устанавливает первоначальное значение 0. Благодаря циклу, просматриваем каждый элемент массива и с помощью оператора условий записываем только те значения в наши переменные, которые выполняют условие.

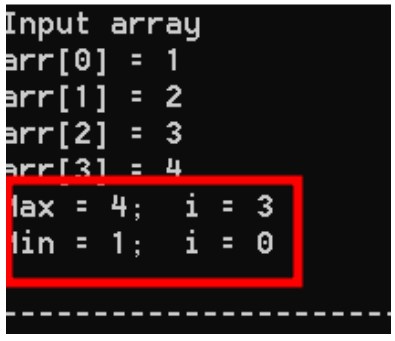


Рисунок 2

Пример решения задачи

Программа принимает целочисленные числа в одномерный массив, программа должна инициализировать одномерный массив длинной 4, и произвести суммирования значений.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
using namespace std;

int main() {

	//Инициализируем одномерный массив
	int arr[4];
	cout << "Input array"<<endl;

	//Заполняем значениями одномерный массив
	for (int i=0; i<4; i++)
	{
	  cout << "arr[" << i << "] = ";
	  cin >> arr[i];
	}

	//Инициализируем переменную
	int sum = 0;

	// производим суммирование всех значений в массиве
	for (int i = 0; i < 4; i++)
	  sum += arr[i];

	//Выводим результат
	cout << "Sum array = " << sum << endl;
	return 0;
}
```

Лабораторная работа №14. Одномерные массивы. Методы сортировки массива

С помощью методов сортировки можно отсортировать массивы. Существует следующие методы сортировки:

1. Сортировка выбором
2. Пузырьковая сортировка
3. Сортировка вставками

Рассмотрим задачу:

Необходимо создать одномерный массив, где длинна будет равна 10 значениям. Массив имеет предопределенные значения: 5, 52, 86, 2, 99, 15, 3, 62, 88, 35. Отсортировать одномерные массивы разными методами. Пример кода представлен в листинге 1-3.

Листинг 1

```
#include <iostream>
#include <algorithm> // для std::swap. В C++11 используйте заголовок <utility>

int main()
{
	const int length = 5;
	int array[length] = { 30, 50, 20, 10, 40 };

	// Перебираем каждый элемент массива (кроме последнего, он уже будет отсортирован к тому времени, когда до него доберемся)
	for (int startIndex = 0; startIndex < length - 1; ++startIndex)
	{
		// В переменной smallestIndex хранится индекс наименьшего значения, которое нашли в этой итерации.
		// Начинаем с того, что наименьший элемент в этой итерации - это первый элемент (индекс 0)
		int smallestIndex = startIndex;

		// Затем ищем элемент поменьше в остальной части массива
		for (int currentIndex = startIndex + 1; currentIndex < length; ++currentIndex)
		{
			// Если нашли элемент, который меньше нашего наименьшего элемента,
```

```
                if (array[currentIndex] < array[smallestIndex])
                      // то запоминаем его
                      smallestIndex = currentIndex;
           }

           // smallestIndex теперь наименьший элемент.
// Меняем местами наше начальное наименьшее число с тем, которое обнаружили
           std::swap(array[startIndex], array[smallestIndex]);
     }

     // Теперь, когда весь массив отсортирован - выводим его на экран
     for (int index = 0; index < length; ++index)
           std::cout << array[index] << ' ';

     return 0;
}
```

Данный метод работает следующим образом: каждый шаг цикла смещается в сторону конца массива. Поэтому, на первой итерации цикла найденное минимальное значение меняется местами со значением в нулевой ячейке массива. На второй итерации «начало» уже будет указывать на следующую (первую) ячейку и так далее.

Листинг 2

```
void main()
{
const int SIZE = 10;
int a[SIZE] = { 5, 52, 86, 2, 99, 15, 3, 62, 88, 35 };

 cout << "Исходный массив:\n";
 for (int i = 0; i < SIZE; i++)
 {
 arr[i] =  SIZE - i; // заполняем значениями по убыванию
 cout << arr[i] << "\n__\n";
 }
 cout << "\n\n";
```

```
 bubbleSort(arr, SIZE);  // передаем в функцию для сортировки

 cout << "Массив после сортировки:\n";
 for (int i = 0; i < SIZE; i++)
 {
 cout << arr[i] << "\n__\n";
 }
 cout << "\n\n";
}

void bubbleSort(int arrForSort[], const int SIZE)
{
 int buff = 0; // для временного хранения значения, при перезаписи

 for (int i = 0; i < SIZE - 1; i++) //
 {
 // вложенный цикл проходит от четвертого элемента
 // до первого, находит с помощью if самое "легкое" значение,
 // сравнивая соседние пары элементов,
 // и поднимает его в нулевую ячейку массива
 for (int j = SIZE - 1; j > i; j--)
 {
 if (arrForSort[j] < arrForSort[j - 1])
 {
 buff = arrForSort[j - 1];
 arrForSort[j - 1] = arrForSort[j];
 arrForSort[j] = buff;
 }
 }
 // далее значение i увеличивается на 1
 // и внутренний цикл будет перебирать элементы
 // от четвертого до второго. Таким образом поднимет самое
 // "легкое" значение из оставшихся в первую ячейку и т.д.
 }
}
```

Алгоритм сортировки «пузырьком» состоит в повторении проходов по массиву с помощью вложенных циклов. При каждом проходе по массиву сравниваются между собой

пары «соседних» элементов. Если числа какой-то из сравниваемых пар расположены в неправильном порядке – происходит обмен (перезапись) значений ячеек массива.

Листинг 3

```
#include <iostream>
using namespace std;

int main()
{
	const int N = 10;
	int a[N] = { 5, 52, 86, 2, 99, 15, 3, 62, 88, 35 };

	int buff = 0;	// для хранения перемещаемого значения
	int i, j;	// для циклов

	// Начало сортировки
	for (i = 1; i < N; i++)
	{
		buff = a[i]; // запомним обрабатываемый элемент
		// и начнем перемещение элементов слева от него
		// пока запомненный не окажется меньше чем перемещаемый
		for (j = i - 1; j >= 0 && a[j] > buff; j--)
				a[j + 1] = a[j];

		a[j + 1] = buff; // и поставим запомненный на его новое место
	}
	// Конец сортировки

	for (int i = 0; i < N; i++) // вывод отсортированного массива
		cout << a[i] << '\t';
	cout << endl;
}
```

Алгоритм «Сортировка вставками» можно описать следующими позициями:

1. Запомнить во временную переменную (buff в примере) значение текущего элемента массива;

2. Пока элементы слева от запомненного значения больше, чем запомненное – перемещаем их на позицию вправо. Получается, что предыдущий элемент займет ячейку запомненного. А тот, что стоит перед предыдущим – переместится в свою очередь на место предыдущего. И так элементы будут двигаться друг за дружкой.

3. Движение элементов заканчивается, если очередной элемент, который нужно сдвинуть, оказывается по значению меньше, чем тот, что запомнили во временную переменную в начале цикла.

4. Цикл берет следующий элемент, и опять сдвигает все элементы, которые расположены перед ним и большие по значению.

Пример решения задачи

Программа принимает целочисленные числа в одномерный массив, программа должна отсортировать одномерный массив.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 14. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
using namespace std;

void main()
{
   const int SIZE = 10;
   int a[SIZE] = { 5, 52, 86, 2, 99, 15, 3, 62, 88, 35 };

   cout << "Исходный массив:\n";
   for (int i = 0; i < SIZE; i++)
   {
      // заполняем значениями по убыванию
      arr[i] =  SIZE - i;
      cout << arr[i] << "\n__\n";
   }
   cout << "\n\n";

   // передаем в функцию для сортировки
   bubbleSort(arr, SIZE);
```

```
   cout << "Массив после сортировки:\n";
   for (int i = 0; i < SIZE; i++)
   {
      cout << arr[i] << "\n__\n";
   }
   cout << "\n\n";
}

void bubbleSort(int arrForSort[], const int SIZE)
{
   // для временного хранения значения, при перезаписи
   int buff = 0;
   for (int i = 0; i < SIZE - 1; i++)
   {
      // вложенный цикл проходит от четвертого элемента
      // до первого, находит с помощью if самое "легкое" значение,
      // сравнивая соседние пары элементов,
      // и поднимает его в нулевую ячейку массива
      for (int j = SIZE - 1; j > i; j--)
      {
           if (arrForSort[j] < arrForSort[j - 1]) {
                buff = arrForSort[j - 1];
                arrForSort[j - 1] = arrForSort[j];
                arrForSort[j] = buff;
           }
      }
      // далее значение i увеличивается на 1
      // и внутренний цикл будет перебирать элементы
      // от четвертого до второго. Таким образом поднимет самое
      // "легкое" значение из оставшихся в первую ячейку и т.д.
   }
}
```

Лабораторная работа №15. Двухмерный массивы. Подсчёта данных в матрице

Кроме одномерных массивов в C++ есть многомерные. Данные в двухмерном массиве (матрица) можно представить виде таблицы, в которой присутствуют строки и столбцы. Структура двухмерного массива выглядит так:

```
#include <iostream>
#include <algorithm> // для std::swap. В C++11 используйте заголовок <utility>

int main()
{
    Тип_данных название_массива[количество_строк][количество_столбцов]

    return 0;
}
```

Рисунок 1

Для инициализации двумерного массива в программе необходимо прописать следующие варианты кода:

```
#include <iostream>
using namespace std;

int main()
{
    //Указания размера матрицы
    int numbers[3][2];
    //Указания размера и содержимого в матрице
    int numbers[3][2] = { {1, 2}, {4, 5}, {7, 8} };
}
```

Рисунок 2

Рассмотрим задачу:

Необходимо создать двумерный массив, где размерность будет 3 на 3. Данные в массив вводятся с клавиатуры. Вывести на экран матрицу с индексом и без индекса. Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
using namespace std;

int main()
{
```

```
     int mtx[3][3];
    cout << "Input array:" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << "mtx[" << i << "][" << j << "] = ";
            cin >> mtx[i][j];
        }
    }
    cout << "Output array with index" << endl;
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            cout << "mtx[" << i << "][" << j << "] = " << mtx[i][j] << endl;
    cout << "Output array without index" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
            cout << mtx[i][j] ;
        cout << endl;
    }
    return 0;
}
```

В начале программного кода инициализировали матрицу, имеющую три строки и три столбца, в итоге у нас получается, что матрица хранит в себе девять значений. Так же, как и в массиве у значений в матрице есть свои индексы, первый индекс обозначает в какой строчке находиться значение, а второй в каком столбце. Для наглядности реализованы два вида вывода матрицы. В первой можем просмотреть какую позицию занимает наше значение, а во второй, как оно храниться в памяти компьютера.

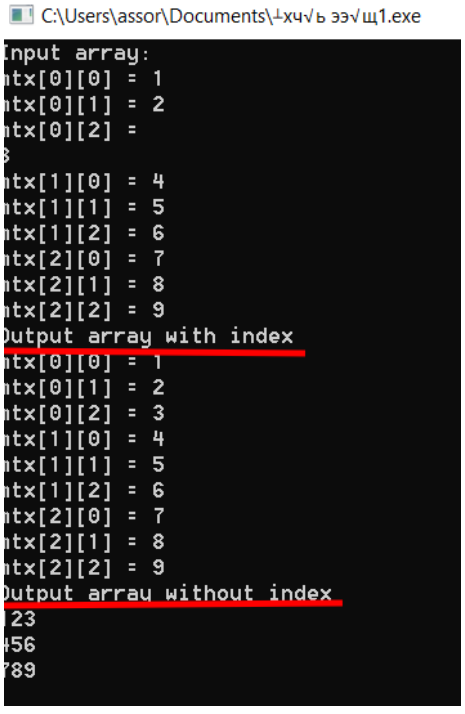


Рисунок 3

Рассмотрим задачу:

Необходимо создать двумерный массив, где размерность будет 3 на 3. Посчитать сумму строк и столбцов, результат вывести на экран. Пример кода представлен в листинге 2.

Листинг 2

```
#include <iostream>
using namespace std;

int main()
{
    int mtx[3][3];
    cout << "Введите значения в матрицу:" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << "mtx[" << i << "][" << j << "] = ";
```

```
            cin >> mtx[i][j];
        }
    }
    int Allsum = 0;
    int sumColum = 0;
    int sumRow = 0;
    //
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            Allsum += mtx[i][j];
    cout << "Общая сумма = " << Allsum << endl;
    //
    for (int j = 0; j < 3; j++)
    {
        sumColum = 0;
        for (int i = 0; i < 3; i++)
            sumColum += mtx[i][j];
        cout << "Сумма столбца " << j <<" = " << sumColum << endl;
    }
    //
    for (int i = 0; i < 3; i++)
    {
        sumRow = 0;
        for (int j = 0; j < 3; j++)
            sumRow += mtx[i][j];
        cout << "Сумма строки " << i << " = " << sumRow << endl;
    }
    return 0;
}
```

Результат работы программы представлен на рисунке 4.

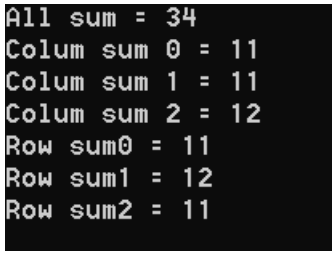

All sum = 34
Colum sum 0 = 11
Colum sum 1 = 11
Colum sum 2 = 12
Row sum0 = 11
Row sum1 = 12
Row sum2 = 11

Рисунок 4

Пример решения задачи

Программа принимает целочисленные числа в двухмерный массив, программа должна рассчитать общую сумму, сумму столбца и сумму строки двухмерного массива.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 15. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
using namespace std;

int main()
{
    //Вывод русского текста
    setlocale(LC_ALL, "Russian");
    //Инициализируем двухмерный массив
    int mtx[3][3];
    cout << "Введите значения в матрицу:" << endl;
    //Вводим значения в двухмерный массив
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << "mtx[" << i << "][" << j << "] = ";
            cin >> mtx[i][j];
        }
    }
    //Инициализируем переменные для записи сумм
    int Allsum = 0;
    int sumColum = 0;
    int sumRow = 0;
    //Производим расчет общий суммы двухмерного массива
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            Allsum += mtx[i][j];
    cout << "Общая сумма = " << Allsum << endl;
    // Производим расчет суммы столбцов двухмерного массива
    for (int j = 0; j < 3; j++)
```

```
    {
        sumColum = 0;
        for (int i = 0; i < 3; i++)
            sumColum += mtx[i][j];
        cout << "Сумма столбца " << j <<" = " << sumColum << endl;
    }
    // Производим расчет суммы строк двухмерного массива
    for (int i = 0; i < 3; i++)
    {
        sumRow = 0;
        for (int j = 0; j < 3; j++)
            sumRow += mtx[i][j];
        cout << "Сумма строки " << i << " = " << sumRow << endl;
    }
    return 0;
}
```

Лабораторная работа №16. Двухмерный массивы. Обход диагоналей матрицы

В двухмерной матрице есть два вида диагоналей: ***главная диагональ*** и ***побочная диагональ.***

Главная диагональ – диагональ, проведённая из левого верхнего угла матрицы в правый нижний угол.

Рассмотрим задачу:

Необходимо создать двумерный массив, где размерность будет 3 на 3. Требуется изменить элементы главной диагонали на число 0. Пример кода представлен в листинге 1.

Листинг 1

```
const int n = 3;
int main()
{
    setlocale(LC_ALL, "Russian");
```

```
    int mtx[n][n];
    cout << "Введите значения в матрицу:" << endl;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cout << "mtx[" << i << "][" << j << "] = ";
            cin >> mtx[i][j];
        }
    }
    //Вывод значение 0 по главной диагонали
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
                mtx[i][j] = 0;
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            cout << mtx[i][j];
        cout << endl;
    }
    return 0;
}
```

Для нахождения главной диагонали используем простое условие, где индекс строки должен равняться индексу столбца.

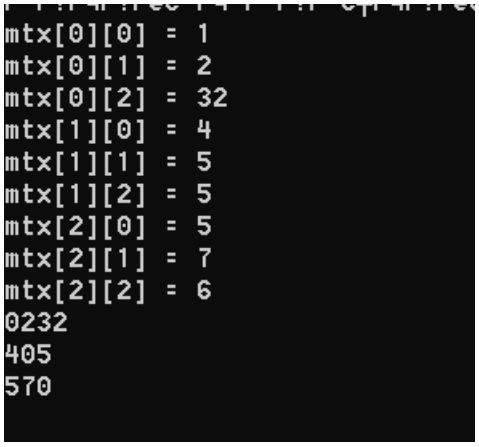


Рисунок 1

Побочная диагональ – диагональ, проведённая из левого нижнего угла матрицы в правый верхний угол.

Рассмотрим задачу:

Необходимо создать двумерный массив, где размерность будет 3 на 3. Требуется изменить элементы побочной диагонали на число 0. Пример кода представлен в листинге 2.

Листинг 2

```
const int n = 3;
int main()
{
    setlocale(LC_ALL, "Russian");

    int mtx[n][n];
    cout << "Введите значения в матрицу:" << endl;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cout << "mtx[" << i << "][" << j << "] = ";
            cin >> mtx[i][j];
        }
```

```
    }
    //Вывод значение 0 по побочной диагонали
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (j == n - i - 1)
                mtx[i][j] = 0;
        }
    }
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
            cout << mtx[i][j];
        cout << endl;
    }
    return 0;
}
```

Для нахождения побочной диагонали используем условие, где индекс столбца должен равняться размеру матрицы, вычтенному из текущего расположения индекса строки и единицы.

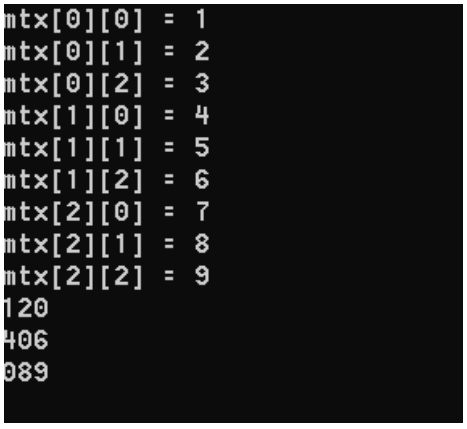


Рисунок 2

Пример решения задачи

Программа принимает целочисленные числа в двухмерный массив, программа должна по главной диагонали вывести нули.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 16. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
using namespace std;

int main()
{
    //Вывод русского текста
    setlocale(LC_ALL, "Russian");

    //Инициализируем двухмерный массив
    int mtx[3][3];
    cout << "Введите значения в матрицу:" << endl;

    //Вводим значения в двухмерный массив
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << "mtx[" << i << "][" << j << "] = ";
            cin >> mtx[i][j];
        }
    }

    //Вывод значение 0 по главной диагонали
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
                mtx[i][j] = 0;
        }
    }
```

```
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            cout << mtx[i][j];
        cout << endl;
    }
    return 0;
}
```

Лабораторная работа №17. Двухмерный массивы. Преобразование матрицы.

С помощью преобразования матрицы можно перезаписать двухмерный массив в одномерный.

Рассмотрим задачу:

Необходимо создать произвольный двухмерный массив, где значения будут записаны с клавиатуры. Требуется преобразовать двумерную произвольную матрицу в одномерный массив и вывести значения на экран. Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "rus");
    int n = 0; int m = 0;
    cout << "Введите количество строк" << endl;
    cin >> n;
    cout << "Введите количество столбцов" << endl;
    cin >> m;
```

```
    int** mtx = new int* [n];  // int - тип элементов матрицы, **matrix - указатель на указатель
    // new - ключевое слово, которое выделяет память в куче под x элементов типа int

    for (int i = 0; i < n; i++)
        mtx[i] = new int[m];

    int  *arr = new int [n*m];
    // Заполняем матрицу
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cout << "mtx[" << i << "][" << j << "] = ";
            cin >> mtx[i][j];
        }
    }
    //Выводим матрицу
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
            cout << mtx[i][j];
        cout << endl;
    }

    //Преобразовываем матрицу в массив
    int c = 0;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
            arr[c++] = mtx[i][j];
    }
    //Выводим массив
    for (int i = 0; i < n*m; i++)
    {
        cout << arr[i] << " ";
    }
```

```
    return 0;
}
```

Результат работы программы представлен на рисунке ниже.

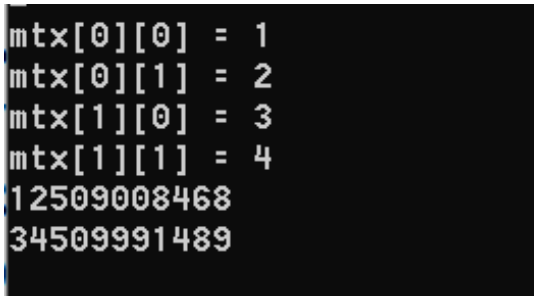


Рисунок 1

Пример решения задачи

Программа принимает целочисленные числа в двухмерный массив, программа должна преобразовать двухмерную массив в одномерный массив.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 17. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
using namespace std;

int main()
{
    //Вывод русского текста
    setlocale(LC_ALL, "rus");
    int n = 0; int m = 0;
    cout << "Введите количество строк" << endl;
    cin >> n;
    cout << "Введите количество столбцов" << endl;
    cin >> m;

    // int - тип элементов матрицы, **matrix - указатель на указатель
    // new - ключевое слово, которое выделяет память в куче под x элементов типа int
    int** mtx = new int* [n];

    for (int i = 0; i < n; i++)
        mtx[i] = new int[m];
```

```
    int *arr = new int [n*m];

    // Заполняем матрицу
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cout << "mtx[" << i << "][" << j << "] = ";
            cin >> mtx[i][j];
        }
    }

    //Выводим матрицу
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
            cout << mtx[i][j];
        cout << endl;
    }

    //Преобразовываем матрицу в массив
    int c = 0;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
            arr[c++] = mtx[i][j];
    }

    //Выводим массив
    for (int i = 0; i < n*m; i++)
    {
        cout << arr[i] << " ";
    }
    return 0;
}
```

Лабораторная работа №18. Двухмерный массивы. Методы сортировки массива.

С помощью методов сортировки можно отсортировать различные матрицы. Также, как и для массивов, можно использовать методы сортировки:

1. Сортировка выбором
2. Пузырьковая сортировка
3. Сортировка вставками

Метод реализации сортировок двухмерного массива представлены в листингах 1-3.

Листинг 1

```
#include <iostream>
#include <cstdlib>
#include <iomanip>

using namespace std;

void Sort(int* arr, size_t size)
{
    for (size_t i = 0; i < size - 1; i++)
    {
        for (size_t j = i + 1; j < size; j++)
        {
            if (arr[j] < arr[i])
            {
                swap(arr[i], arr[j]);
            }
        }
    }
}

void init(int** arr, size_t rows, size_t columns)
{
    for (size_t i = 0; i < rows; i++)
    {
```

```
        arr[i] = new int[columns];
    }
    for (size_t i = 0; i < rows; i++)
    {
        for (size_t j = 0; j < columns; j++)
        {
            arr[i][j] = rand() % 20 - 10;
        }
    }
}

void show(int** arr, size_t rows, size_t columns)
{
    cout << "Matrix:" << endl;
    for (size_t i = 0; i < rows; i++)
    {
        for (size_t j = 0; j < columns; j++)
        {
            cout << setw(3) << arr[i][j] << ' ';
        }
        cout << endl;
    }
}

void delmem(int** arr, size_t rows, size_t columns)
{
    for (size_t i = 0; i < rows; i++)
    {
        delete[] arr[i];
    }
    delete[] arr;
}

int main()
{
    setlocale(0, "");
    size_t n, m; cin >> n >> m;
```

```
    int** arr = new int*[n];
    init(arr, n, m);
    show(arr, n, m);
    int* temp = new int[n*m];
    size_t k = 0;
    for (size_t i = 0; i < n; i++)
    {
        for (size_t j = 0; j < m; j++)
        {
            temp[k] = arr[i][j];
            k++;
        }
    }
    Sort(temp, n*m);
    k = 0;
    for (size_t i = 0; i < n; i++)
    {
        for (size_t j = 0; j < m; j++)
        {
            arr[i][j] = temp[k];
            k++;
        }
    }
    delete[] temp;
    show(arr, n, m);
    delmem(arr, n, m);
    system("pause");
    return 0;
}
```

Листинг 2

```
#include <iostream>

using namespace std;
```

```
int main()
{
    setlocale(LC_ALL,"rus");

int rows,cols,k,temp;

cout<<"Задайте размерность массива:\n";
cout<<"Количество строк: ";
cin>>rows;
cout<<"Количество столбцов: ";
cin>>cols;

cout<<"Номер строки для сортировки: ";
cin>>k;
k--;

int **mas = new int*[rows];

cout<<"Исходный массив: "<<endl;
for (int i=0; i<rows; ++i)
mas[i] = new int[cols];

for (int i=0; i<rows; ++i)
{
    for (int j=0; j<cols; ++j)
    {
        mas[i][j]=rand() %100;
        cout<<mas[i][j]<<"\t";
    }
cout<<endl;
}

for (int i=0; i<cols; ++i)
{
    for (int j=0; j<cols-1; ++j)
    {
        if (mas[k][j]>mas[k][j+1])
```

```
        {
            temp=mas[k][j];
            mas[k][j]=mas[k][j+1];
            mas[k][j+1]=temp;
        }
    }
}

cout<<"Новый массив: "<<endl;

for (int i=0; i<rows; ++i)
{
    for (int j=0; j<cols; ++j)
        cout<<mas[i][j]<<"\t";
cout<<endl;
}



for (int i=0; i<rows; ++i)
    delete [] mas[i];

delete [] mas;

system("pause");
return 0;
}
```

Листинг 3

```
#include<iostream>
#include<vector>
#include<iomanip>
#include<algorithm>
using namespace std;
int main()
```

```
{
  srand(unsigned(time(0)));
  unsigned n = 2*(rand() % 3 + 2);
  vector<vector<int>>reptiloid;
  for (unsigned j = 0; j < n; j++)
  {
    vector<int> ter;
    for (unsigned c = 0; c < n; c++)
    {
      ter.push_back(rand() % 100);
    }
    reptiloid.push_back(ter);
  }

  for (const auto& v : reptiloid)
  {
    for (const auto& b : v)
    {
      cout << setw(7) << b;
    }
    cout << endl;
  }

  cout << endl << endl << endl;
  vector<int> barge;
  for (unsigned u1 = 0; u1 < (n/2); u1++)
  {
    for (unsigned u2 = 0; u2 < n/2; u2++)
    {
      if ((u1 + u2) >= (n / 2) - 1)
      {
        barge.push_back(reptiloid[u1][u2]);
      }
    }
    cout << endl;
  }
```

```
  sort(barge.begin(), barge.end());
    auto c = barge.begin();

  for (unsigned u1 = 0; u1 < n / 2; u1++)
  {
    for (unsigned u2 = 0; u2 < n / 2; u2++)
    {
      if ((u1 + u2) >= (n / 2) - 1)
      {
        reptiloid[u1][u2] = (*c);
        c++;
      }
    }
    cout << endl;
  }

  cout << "Changed:" << endl << endl;
  for (const auto& v : reptiloid)
  {
    for (const auto& b : v)
    {
      cout << setw(7) << b;
    }
    cout << endl;
  }

  return 0;
}
```

Пример решения задачи

Программа принимает целочисленные числа в двухмерный массив, программа должна отсортировать двухмерный массив.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 18. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
using namespace std;

int main()
{
    //Вывод русского текста
    setlocale(LC_ALL,"rus");

    int rows,cols,k,temp;

    cout<<"Задайте размерность массива:\n";
    cout<<"Количество строк: ";
    cin>>rows;
    cout<<"Количество столбцов: ";
    cin>>cols;

    cout<<"Номер строки для сортировки: ";
    cin>>k;
    k--;

    int **mas = new int*[rows];

    cout<<"Исходный массив: "<<endl;
    for (int i=0; i<rows; ++i)
    mas[i] = new int[cols];

    for (int i=0; i<rows; ++i)
    {
        for (int j=0; j<cols; ++j)
        {
            mas[i][j]=rand() %100;
            cout<<mas[i][j]<<"\t";
        }
```

```
        cout<<endl;
    }

    for (int i=0; i<cols; ++i)
    {
        for (int j=0; j<cols-1; ++j)
        {
            if (mas[k][j]>mas[k][j+1])
            {
                temp=mas[k][j];
                mas[k][j]=mas[k][j+1];
                mas[k][j+1]=temp;
            }
        }
    }

    cout<<"Новый массив: "<<endl;

    for (int i=0; i<rows; ++i)
    {
        for (int j=0; j<cols; ++j)
            cout<<mas[i][j]<<"\t";
        cout<<endl;
    }


    for (int i=0; i<rows; ++i)
        delete [] mas[i];

    delete [] mas;

    system("pause");
    return 0;
}
```

Лабораторная работа №19. Двухмерный массивы. Методы поиска элементов массива.

С помощью методов поиска возможно находить нужный элемент в матрице. Существует несколько способов поиска:

1. Линейный поиск.
2. Двоичный (бинарный) поиск. Он более эффективен в случае, если массив заранее отсортирован.

Рассмотрим задачу:

Дана матрица размером 50 на 50, необходимо заполнить ее случайными значениями и произвести линейный поиск числа, введенного с клавиатуры. Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
#include <iomanip>
#include <ctime>
using namespace std;

//прототипы функций
int linSearch(int arr[], int requiredKey, int size); // линейный поиск
void showArr(int arr[], int size); // показ массива

int main()
{
	setlocale(LC_ALL, "rus");
	const int arrSize = 50;
	int arr[arrSize];
	int requiredKey = 0; // искомое значение (ключ)
	int nElement = 0; // номер элемента массива
	srand(time(NULL));

	//запись случ. чисел в массив от 1 до 50
	for (int i = 0; i < arrSize; i++)
	{
		arr[i] = 1 + rand() % 50;
	}
```

```
	showArr(arr, arrSize);

	cout << "Какое число необходимо искать? ";
	cin >> requiredKey; // ввод искомого числа

	//поиск искомого числа и запись номера элемента
	nElement = linSearch(arr, requiredKey, arrSize);

	if (nElement != -1)
	{
		//если в массиве найдено искомое число - выводим индекс элемента на экран
		cout << "Значение " << requiredKey << " находится в ячейке с индексом: "
<< nElement << endl;
	}
	else
	{
		//если в массиве не найдено искомое число
		cout << "В массиве нет такого значения" << endl;
	}
	return 0;
}

//вывод массива на экран
void showArr(int arr[], int arrSize)
{
	for (int i = 0; i < arrSize; i++)
	{
		cout << setw(4) << arr[i];
		if ((i + 1) % 10 == 0)
		{
			cout << endl;
		}

	}
	cout << endl << endl;
}
```

```
//линейный поиск
int linSearch(int arr[], int requiredKey, int arrSize)
{
      for (int i = 0; i < arrSize; i++)
      {
            if (arr[i] == requiredKey)
                  return i;
      }
      return -1;
}
```

Выше представлен самый простой вид поиска (но не самый эффективный). За счет того, что поиск происходит перебором всех элементов массива и сравнения его значений с заданным ключом, скорость выполнения алгоритма достаточно низкая.

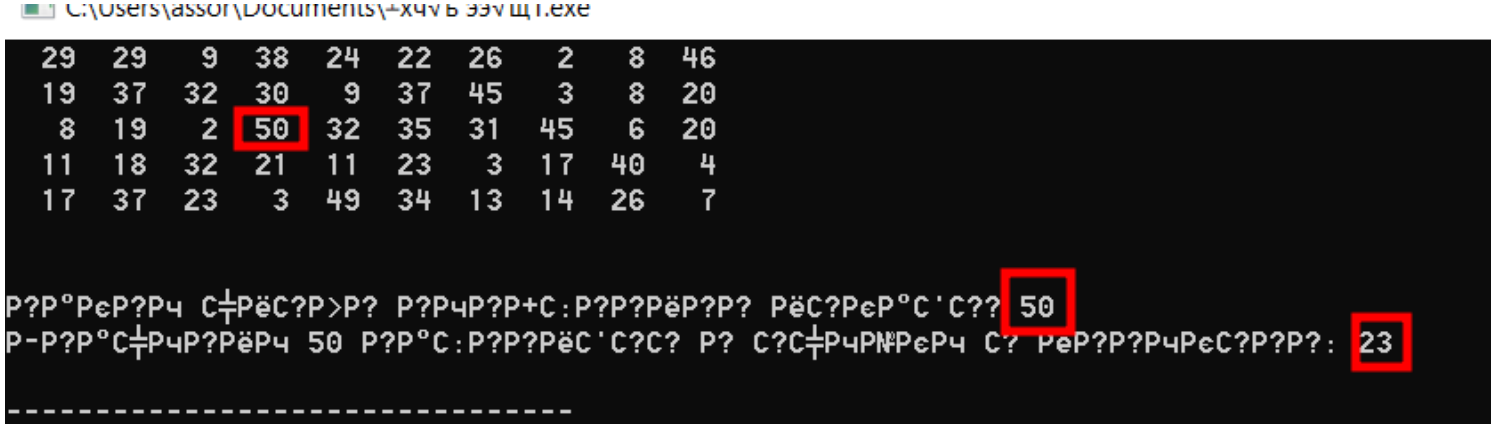


Рисунок 1

Рассмотрим задачу:

Дана матрица размером 50 на 50, необходимо заполнить ее случайными значениями и произвести бинарный поиск числа, введенного с клавиатуры. Пример кода представлен в листинге 2.

Листинг 2

```
#include <iostream>
using namespace std;

// функция с алгоритмом двоичного поиска
int Search_Binary (int arr[], int left, int right, int key)
{
 int midd = 0;
 while (1)
 {
 midd = (left + right) / 2;

 if (key < arr[midd])       // если искомое меньше значения в ячейке
 right = midd - 1;      // смещаем правую границу поиска
 else if (key > arr[midd])  // если искомое больше значения в ячейке
 left = midd + 1;    // смещаем левую границу поиска
 else                       // иначе (значения равны)
 return midd;           // функция возвращает индекс ячейки

 if (left > right)          // если границы сомкнулись
 return -1;
 }
}

int main()
{
 setlocale (LC_ALL, "rus");

 const int SIZE = 12;
 int array[SIZE] = {};
 int key = 0;
 int index = 0; // индекс ячейки с искомым значением

 for (int i = 0; i < SIZE; i++) // заполняем и показываем массив
 {
 array[i] = i + 1;
 cout << array[i] << " | ";
 }
```

```
  cout << "\n\nВведите любое число: ";
  cin >> key;

  index = Search_Binary (array, 0, SIZE, key);

  if (index >= 0)
  cout << "Указанное число находится в ячейке с индексом: " << index << "\n\n";
  else
  cout << "В массиве нет такого числа!\n\n";

  return 0;
}
```

После сортировки массива, проверяется значение среднего элемента массива. Если значение совпадает с ключом – алгоритм прекратит работу, и программа выведет сообщение, что значение найдено. Если ключ меньше значения среднего элемента, алгоритм не будет проводить поиск в той половине массива, которая содержит значения больше ключа и наоборот.

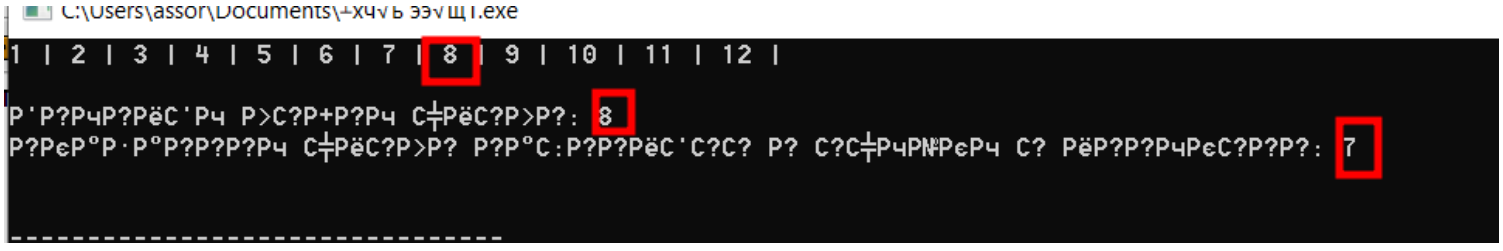


Рисунок 2

Пример решения задачи

Программа принимает целочисленные числа в двухмерный массив, программа должна найти число, введенное с клавиатуры.

Откройте среду Dev C++ и наберите программный код, представленный в листинге 19. Пояснения к коду даны в листинге.

```
#include <iostream>
#include <iomanip>
#include <ctime>
using namespace std;

//прототипы функций
int linSearch(int arr[], int requiredKey, int size); // линейный поиск
void showArr(int arr[], int size);                   // вывод массива

int main()
{
      setlocale(LC_ALL, "rus");
      const int arrSize = 50;
      int arr[arrSize];

      // искомое значение (ключ)
      int requiredKey = 0;

      // номер элемента массива
      int nElement = 0;
      srand(time(NULL));

      //запись случ. чисел в массив от 1 до 50
      for (int i = 0; i < arrSize; i++)
      {
            arr[i] = 1 + rand() % 50;
      }

      showArr(arr, arrSize);

      cout << "Какое число необходимо искать? ";
      // ввод искомого числа
      cin >> requiredKey;
```

```
	//поиск искомого числа и запись номера элемента
	nElement = linSearch(arr, requiredKey, arrSize);

	if (nElement != -1)
	{
		//если в массиве найдено искомое число - выводим индекс элемента на экран
		cout << "Значение " << requiredKey << " находится в ячейке с индексом: " <<
nElement << endl;
	}
	else
	{
		//если в массиве не найдено искомое число
		cout << "В массиве нет такого значения" << endl;
	}
	return 0;
}

//вывод массива на экран
void showArr(int arr[], int arrSize)
{
	for (int i = 0; i < arrSize; i++)
	{
		cout << setw(4) << arr[i];
		if ((i + 1) % 10 == 0)
		{
			cout << endl;
		}

	}
	cout << endl << endl;
}
//линейный поиск
int linSearch(int arr[], int requiredKey, int arrSize)
{
	for (int i = 0; i < arrSize; i++)
	{
		if (arr[i] == requiredKey)
			return i;
	}
	return -1;
```

```
}
```

Лабораторная работа №20. Символьные массивы

Символьный массив имеет тип данных *char* и представляет собой аналог числового, только в виде значений с индексом хранятся символы. Каждая ячейка памяти хранит один символ.

Инициализации символьного массива не отличается от числового:

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      char str[18] = { 'с', 'и', 'м', 'в', 'о', 'л', 'ь', ''и', 'в', 'ы',}
7      return 0;
8  }
9
```

Рисунок 1

Рассмотрим задачу:

Дан символьный массив, длинна которого равна 4 пунктам, необходимо заполнить его символами: a, b, n, c и вывести на экран. Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
using namespace std;

int main()
{
    char str[4] = { 'a', 'n', 'c', 'b'};
    for (int i = 0; i < 4; i++)
    {
       cout << str[i];
    }
    return 0;
}
```

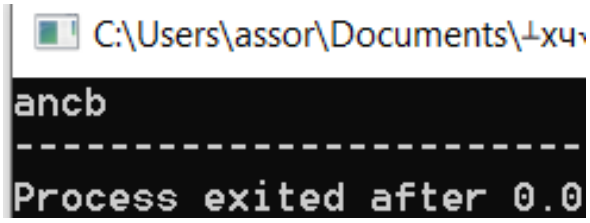


Рисунок 2

Рассмотрим задачу:

Дан символьный массив, длинна которого равна 4 пунктам, необходимо заполнить его символами: a, b, n, c сокращенным способом и вывести на экран. Пример кода представлен в листинге 2.

Листинг 2

```
#include <iostream>
using namespace std;

int main()
{
    char str[] =”ancb”
    for (int i = 0; i < 4; i++)
    {
       cout << str[i];
    }
    return 0;
}
```

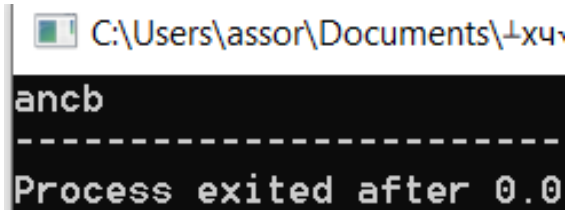


Рисунок 3

Пример решения задачи

Программа содержит в себе код символа, необходимо вывести на экран значение символа из таблицы ASCII.

Откройте среду Dev C++ и набери программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//Подключение библиотеки ввода/вывода
#include <iostream>

using namespace std;

int main()
{
	//объявление переменных
	int ascii;
	//ввод/вывод данных
	cout << "ASCII-code: ";
	cin >> ascii;
	//нахождение переменной в таблице ASCII
	cout << "Symbol: " << (char)ascii << endl;
	return 0;
}
```

Лабораторная работа №21. Строки. Базовые строковые функции. Обработка текстовой информации

Строки в C++ используют для хранения тип данных ***string***. Рассмотрим несколько способов инициализации переменных типа данных ***string***:

1. В первом примере (строка 6) видим простую инициализацию, где изначально переменная не имеет значение.

2. Во втором примере (строка 7) инициализируем переменную, но задаем первоначальное значение.

3. В третьем примере (строка 8) еще один способ инициализации переменной с указанием первоначального значения.

4. В четвертом примере (строка 9) инициализируется переменная, где первым параметром указывается количество повторяемых символов, а вторым параметром указывается символ, который будет повторяться.

5. В пятом примере (строка 10) к новой переменной присваивается значение ранее созданной переменной.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      string s1;                  // пустая строка
7      string s2 = "hello";        // hello
8      string s3("welcome");       // welcome
9      string s4(5, 'h');      // hhhhh
10     string s5 = s2;             // hello
11
12     return 0;
13 }
```

Рисунок 1

Рассмотрим задачу:

Дана строка «Hello World!», необходимо вывести её на экран. Пример кода представлен в листинге 1.

Листинг 1

```
int main()
{
    string s = "Hello World!";
    cout << s;
    return 0;
}
```

Результат выполнения программы представлен на рисунке 2.

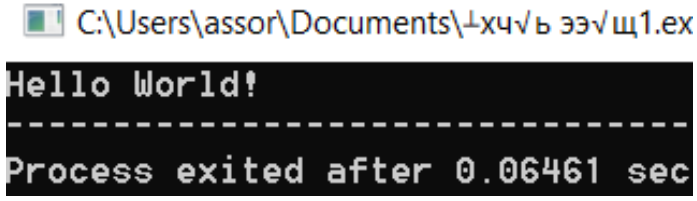


Рисунок 2

Строки имеют возможность использовать встроенные методы обработки значений на языке C++. Например, такие как:

size () - помогает узнать размер строки, то есть из скольких символов она состоит. Пример такой задачи представлен в листинге 2.

Листинг 2

```
int main()
{
 string s1 = "hello world";
 cout << s1.size(); // 11
 return 0;
}
```

empty() - указывает на пустоту переменной, если переменная не имеет значение то возвращает true. Пример такой задачи представлен в листинге 3.

Листинг 3

```
int main()
{
 setlocale(LC_ALL, "rus");
 string s1;
 cin >> s1;
 if(!s1.empty())
  cout << "Переменная имеет какое либо значение" << endl;
 return 0;
}
```

Пример решения задачи

Программа содержит в себе несколько строк, слова в которой необходимо развернуть при условии наличии пробелов между словами.

Откройте среду Dev C++ и набери программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//Подключение библиотек
#include <string>
#include <iostream>
//Создание метода разворота строк

std::string reverse_words(std::string str)
{
    //ввод переменных
    std::string rev_str = str;

    //поиск длинны строки
    int str_len = str.length();

    int k = 0;
    int j = 0;
    //перебор значений для разворота слов

    for (int i = 0; i < str_len; ++i)
    {

        if ((str[i] == 0x20) and (str[i - 1] != 0x20))
        {
            for (j = 0; j < (i - k); j++)
```

```
            {
                 rev_str[j + k] = str[i - j - 1];
            }
            k = i + 1;
        }
        else if ((str[i] == 0x20) and (str[i - 1] == 0x20))
        {
            ++k;
        }

        // Поиск последнего элемента в строке
        if ((i == str_len - 1) and (str[i] != 0x20))
        {
            // Если нет пробел не будет разварачить слова
            for (j = 0; j <= (i - k); j++)
            {
                rev_str[j + k] = str[i - j];
            }
        }
    }
    for (int i = 0; i < str_len; ++i)
    {
        std::cout << rev_str[i];
    }
    return rev_str;
}

int main()
{
    //тестовые строки
    std::string testString1 = "Top website www.test.ru";
    std::string testString2 = "Loki   mobile   www.ya.ru ";

    //вызов методов
    reverse_words(testString1);
    reverse_words(testString2);

    return 0;
}
```

Лабораторная работа №22. Текстовые файлы

Большинство программ работает с текстовыми файлами, из этого возникла необходимость: создавать, удалять, записывать, читать и открывать файлы. Файл – это набор байт, в которых содержится различная информация, которую можно записывать, читать и изменять. Текстовые файлы имеют расширение *.txt*. В языке программирования существует объект класса ***ofstream***, который отвечает за выполнение выше приведенных функций.

Существует несколько режимов работы с файлами:

1. ios_base::in – открытие файла для чтения;
2. ios_base::out – открытие файла для записи;
3. ios_base::ate – при открытие переместить указатель в конец файла
4. ios_base::app – открыть файл для записи в конец файла
5. ios_base::trunc – удалить содержимое файла, если он существует
6. ios_base::binary – открыть файл в двоичном режиме

Данные режимы указываются непосредственно при создании объекта:

```
ifstream fout ("app.txt" , ios_base::app);
```

Рисунок 1

Для использования нескольких режимов необходимо разделять их с помощью символа

```
ifstream fout ("app.txt" , ios_base::app| ios_base::ate);
```

Рисунок 2

Рассмотрим задачу:

Необходимо создать файл с именем «app.txt» и записать в него строку: «Работа с файлами на C++». Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "rus");
    ofstream fout("app.txt"); // создаём объект класса ofstream для
записи и связываем его с файлом app.txt
    fout << "Работа с файлами в C++"; // запись строки в файл
    fout.close(); // закрываем файл
    return 0;
}
```

Для начала работы с ***ofstream*** необходимо добавить библиотеку ***#include <fstream>*** в которой содержатся все необходимые методы для применения ***ofstream***.

Первым действием создаем объект типа ***ofstream*** и связываем его с файлом, где название указывается в скобках. Если данного файла нет в проекте, то он автоматически создатся в корневой папке проекта, если он есть, то будет просто открываться.

Во втором действии записываем значение в файл. И после этого закрываем файл.

Рассмотрим задачу:

Необходимо прочесть значения из файла с именем «app.txt» и вывести информацию из него на экран. Пример кода представлен в листинге 2.

Листинг 2

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "rus");
```

```
    char c[50]; // буфер промежуточного хранения считываемого из файла текста
    ifstream fout ("app.txt"); // открыли файл для чтения
    if (!fout.is_open()) // если файл не открыт
      cout << "Файл не может быть открыт!\n";
    else
    {
      fout.getline(c, 50); // считали строку из файла
      fout.close(); // закрываем файл
      cout << c << endl; // напечатали эту строку
    }
return 0;
}
```

В данном программном коде появились новые методы, такие как:

1. ***is_open()*** - данный метод используется для проверки, что файл был открыт, если это так, то возвращает *true*.
2. ***getline()*** - данный метод отвечает за чтение данных из файла. Для получения информации необходимо задать два параметра: первый - в какую переменную будет записываться данные, второй - количество выводящих символов из файла.

Пример решения задачи

Программа принимает вещественные значение из файла, необходимо вывести на экран их самих и их количество.

Откройте среду Dev C++ и набери программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <stdlib.h>
using namespace std;
int main()
{
   int n=0;
   float a;
   fstream F;
```

```
    //открываем файл в режиме чтения
    F.open("D:\\sites\\accounts.txt");

    //если открытие файла прошло корректно, то
    if (F)
    {
        //цикл для чтения значений из файла; выполнение цикла прервется,
        //когда достигнем конца файла, в этом случае F.eof() вернет истину.
        while (!F.eof())
        {
            //чтение очередного значения из потока F в переменную a
            F>>a;
            //вывод значения переменной a на экран
            cout<<a<<"\t";
            //увеличение количества считанных чисел
            n++;
        }
        //закрытие потока
        F.close();

        //вывод на экран количества считанных чисел
        cout<<"n="<<n<<endl;
    }
    //если открытие файла прошло некорректно, то вывод
    //сообщения об отсутствии такого файла
    else cout<<" Файл не существует"<<endl;
    system("pause");
    return 0;
}
```

Лабораторная работа №23. Бинарные файлы

Бинарный файл – это файл, в котором содержится информация в виде двоичного кода.

Для использования бинарных файлов во время создания объекта **ifstream** необходимо указавать режим ios_base::binary.

Рассмотрим задачу:

Необходимо записать и прочесть значения из бинарного файла с именем «1.txt» и вывести информацию из него на экран. Пример кода представлен в листинге 1.

Листинг 1

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "rus"); // корректное отображение Кириллицы
    int y = 0; //Y будем записывать в файл
    int x = 0; //X будем считывать из файла

    cout << "Y = "; cin >> y; //Вводим число, которое нужно сохранить в файл
    ofstream out("D://1.txt",ios_base::binary|ios_base::out); //Открываем файл в
двоичном режиме для записи
    out.write((char*)&y, sizeof y); //Записываем в файл число y
    out.close(); //Закрываем файл

    cout << "x = " << x << endl; //Показываем X до его изменений

    ifstream in("D://1.txt",ios_base::binary|ios_base::in); //Открываем файл в
двоичном режиме только для чтения
    in.read((char*)&x, sizeof x); //Читаем оттуда информацию и запоминаем её в X
    in.close(); //Закрываем файл

    cout << "x = " << x << endl; //Показываем X после изменения

    return 0;
}
```

В данном примере используем две переменные X и Y, где Y отвечает за значение, которое будет записываться в файл, а X за значение, которое будет считываться из файла.

В начале запрашиваем у пользователя значение, которое будет записано в бинарный файл, после этого создаем объект ***ifstream***, в котором указываем путь создания файла и его

режимы для работы с бинарными файлами для записи данных, далее записываем значение в файл и закрываем его.

После этого используем значение X для записи. Снова создаем объект ***ifstream***, в котором указываем путь создания файла и его режимы для работы с бинарными файлами и для чтения данных, после этого считываем все значения из файла и записываем их в переменную X и выводим её.

Пример решения задачи

Программа принимает вещественные значение из файла, необходимо создать два новых файла для четных и нечетных значений.

Откройте среду Dev C++ и набери программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
     //Y будем записывать в файл
    int y[10] = {1,2,3,4,5,6,7,8,9};
    ofstream df ;
    for(int i = 0; i < 10; i++)
    {
        //Если значение нечетное записываем в первый файл, иначе во второй
        if (y[i] / 2 == 1)
                  //Открываем файл в двоичном режиме для записи
                  df = ofstream ("D://1.txt",ios_base::binary|ios_base::out);
        else
                  //Открываем файл в двоичном режиме для записи
                 df = ofstream ("D://2.txt",ios_base::binary|ios_base::out);

        //Записываем в файл число y
        df.write((char*)&y[i], sizeof y);

        //Закрываем файл
        df.close();
    }
```

```
    return 0;
}
```

Лабораторная работа №24. Классы и объекты

Класс представляет из себя составной тип, который может использовать другие типы, он может описывать несколько типов объектов. Класс является планом объекта, а объект представляет конкретное реализацию класса.

Класс может в себе содержать как переменные, так и методы в которых описаны свои инструкции выполнения программного кода.

Рассмотри инициализацию класса, представленного на рисунке 1.

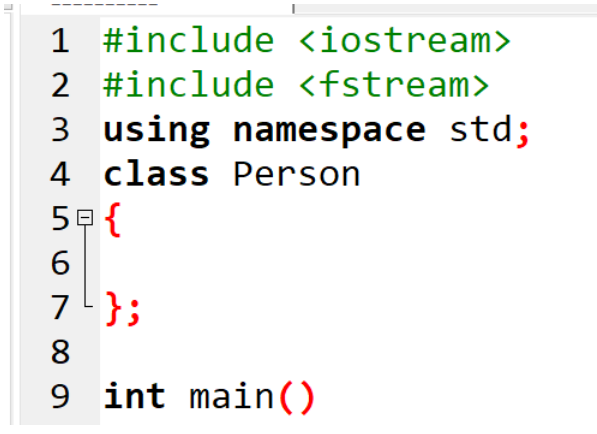


Рисунок 1

В данном случае класс называется Person, он будет описывать характеристики человека, в зависимости от тех параметров, которые будут заданы.

Рассмотрим задачу:

Необходимо создать класс Person, содержащий в себе параметры: имя, возраст. Вывести значения класса на экран. Пример кода представлен в листинге 1.

Листинг 1

```
class Person
{
```

```
public:
    string name;
    int age;
    void info() {
        cout << "Имя: " << name << "; Возраст: " << age << endl;
    }
};
int main()
{
    setlocale(LC_ALL, "rus");
    Person person;
    person.name = "Алексей";
    person.age = 22;
    person.info();

    return 0;
}
```

Класс имеет две переменные ***name*** и ***age***, в которых записывается имя и возраст человека. С помощью метода ***info*** можем вывести информацию о человеке, которую занесли в программу.

Существует два вида доступа к объектам: ***private*** и ***public***.

Private – это спецификатор доступа, который скрывает свои элементы от других классов, ***public*** наоборот предоставляет доступ к своим элементам.

Рассмотрим задачу:

Необходимо модифицировать класс Person, добавив в него спецификатор доступа private. Вывести значения класса на экран. Пример кода представлен в листинге 2.

Листинг 2

```
class Person
{
public:
    Person(string n, int a)
```

```
    {
       name = n; age = a;
    }
    void info() {
        cout << "Имя: " << name << "; Возраст: " << age << endl;
    }
private:
    string name;
    int age;

};
int main()
{
    setlocale(LC_ALL, "rus");
    Person person("Алексей", 22);
    person.info();

    return 0;
}
```

В данном примере указали что переменные ***name*** и ***age*** будут видны только для класса Person, а для записи в них данных используем метод Person, который вызывается при инициализации класса Person, в него посылаем два параметра, первый - имя пользователя, второй - его возраст.

Пример решения задачи

Программа принимает 2 значения: имя студента и возраст, далее программа должна записывать эти данные в поля класса Studentчерез метод Student и выводить их в консоли с помощью метода info.

Откройте среду Dev C++ и набери программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>
class Student
{
```

```
public:
    //Метод для записи данных
    Student(std::string n, int a)
    {
       name = n;
       age = a;
    }

    //Метод для вывода данных
    void info() {
        std::cout << "Name: " << name << "; Age: " << age << std::endl;
    }

    //Объявление полей класса
    private:
    std::string name;
    int age;
};

int main()
{
     Передача данных в класс
     Student student("Alex", 22);
     student.info();
     return 0;
}
```

Лабораторная работа №25. Структуры

Структуры, как и классы представляют собой производный тип данных. Для обозначения структуры используется ключевое слово **struct**.

Рассмотрим инициализацию структуры, представленную на рисунке 1.

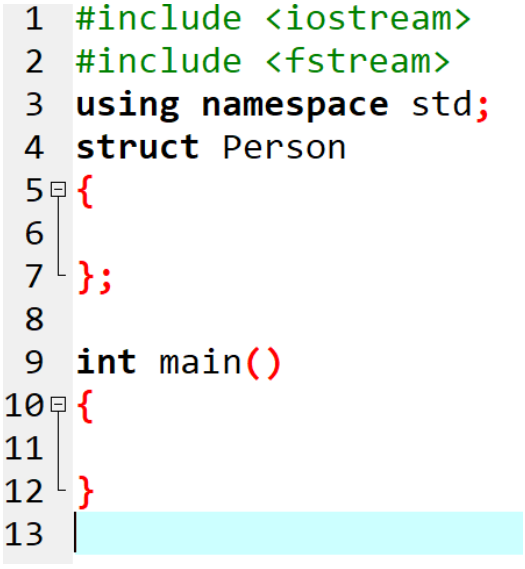


Рисунок 1

Рассмотрим задачу:

Необходимо модифицировать класс Person так, чтобы он превратился в структуру. Вывести значения класса на экран. Пример кода представлен в листинге 1.

Листинг 1

```
struct Person
{
public:
    Person(string n, int a)
    {
       name = n; age = a;
    }
    void info() {
        cout << "Имя: " << name << "; Возраст: " << age << endl;
```

```
    }
private:
    string name;
    int age;

};
int main()
{
    setlocale(LC_ALL, "rus");
    Person person("Алексей", 22);
    person.info();

    return 0;
}
```

Пример решения задачи

Программа принимает 3 целых числа, далее программа должна записывать эти числа в поля структуры Route и выводить их в консоли.

Откройте среду Dev C++ и набери программный код, представленный в листинге 1. Пояснения к коду даны в листинге.

```
//iostream - стандартная библиотека ввода/вывода
#include <iostream>

struct Route
{
      //Обьявление полей структуры
      int start_route;
      int end_route;
      int number_route;
};

int main()
{
      //Создание экземпляра структуры
      Route rt = Route();

      //Ввод данных в поля структуры
```

```
	std::cin >> rt.start_route;
	std::cin >> rt.end_route;
	std::cin >> rt.number_route;

	//Вывод данных на экран
	std::cout << rt.start_route <<endl;
	std::cout>> rt.end_route <<endl;
	std::cout>> rt.number_route <<endl;
	return 0;
}
```

РАЗДЕЛ 1.2. СБОРНИК ЗАДАНИЙ ЛАБОРАТОРНЫХ РАБОТ ПО СПЕЦКУРСУ «ПРОГРАММИРОВАНИЕ» ДЛЯ 10 ИТ-КЛАССА

Лабораторная работа №1. Ввод и вывод данных, оператор присваивания

1. Пользователь вводит число являющиеся стороной квадрата. Необходимо найти его площадь S. S вычисляется по формуле *сторона квадрата * сторона квадрата*.
2. Пользователь вводит число являющиеся диаметром окружности. Необходимо найти длину L. π берём, как 3.14. L = *π * диаметр окружности*.
3. Пользователь вводит число являющиеся длиной ребра куба. Необходимо найти объем куба V. V = длина ребра куба^3
4. Пользователь вводит три числа a и b, c. Необходимо посчитать среднее арифметическое этих чисел.
5. Пользователь вводит два числа a и b. Необходимо найти среднее геометрическое. Среднее геометрическое считается, как корень из произведения этих чисел.
6. Пользователь вводит три числа a, b и c. Необходимо изменить содержимое значений следующим образом: a – в b, b – в c, c – в a. Вывести получившиеся значения a, b и c.
7. Пользователь вводит три числа a, b и c. Необходимо вычислить значение *дискриминанта* по следующей формуле: *D = (b*b) - 4a*c*.
8. Пользователь вводит два числа s, t. Необходимо вычислить значение *скорости движения тела* по следующей формуле: *v=s/t*.
9. Пользователь вводит два числа F, m. Необходимо рассчитать *Второй закон Ньютона* по следующей формуле: a=F/m.
10. Пользователь вводит число X. Необходимо найти значение уравнения по следующей формулы: *y=7(x-5) -4(x-5) +3*.
11. Пользователь вводит число x и z. Необходимо найти значение уравнения по следующей формулы: *y=23(zx-4) -7(x-2) +15*.

Лабораторная работа №2. Логические выражения

Типовые варианты заданий:

1. Пользователь вводит число, необходимо проверить является ли число положительным.
2. Пользователь вводит два числа. Необходимо проверить, что введённые числа удовлетворяют A > 2 и B ≤ 3.

3. Пользователь вводит два числа. Необходимо проверить, что введённые числа удовлетворяют $A \geq 0$ или $B < -2$.
4. Пользователь вводит три числа. Необходимо проверить, что введённые числа удовлетворяют $A < B < C$.
5. Пользователь вводит два числа. Необходимо проверить, что оба числа чётные.
6. Пользователь вводит три числа. Необходимо проверить, что введённые числа удовлетворяют $A > B > C$.
7. Пользователь вводит два числа. Необходимо проверить, что оба числа нечётные.
8. Пользователь вводит три числа. Необходимо определить самое большое число и самое малое число.
9. Пользователь вводит два числа a и b. Необходимо поменять значение переменных таким образом, чтобы значение a оказалась больше значение b и наоборот.
10. Пользователь вводит три числа. Необходимо определить самое малое число.
11. Пользователь вводит три числа. Необходимо определить самое большое число.

Лабораторная работа №3. Оператор условия

1. Пользователь вводит число. Если оно больше 0, то необходимо прибавить к нему 1; иначе вычесть из него 2. Вывести полученное число на экран.
2. Пользователь вводит число. Если оно больше 0, то прибавить к нему 1; если меньше 0, то вычесть из него 2; если равно нулю, то заменить его на 10. Вывести полученное число на экран.
3. Пользователь вводит три целых числа. Необходимо найти и вывести количество положительных чисел из данного набора.
4. Пользователь вводит три целых числа. Необходимо найти и вывести количество положительных и количество отрицательных чисел в данном наборе.
5. Пользователь вводит два числа. Вывести то число, которое является большим.
6. Пользователь вводит два числа. Вывести значения в порядке возрастания.
7. Пользователь вводит два числа. Вывести значения в порядке убывания.
8. Пользователь вводит три числа. Вывести только те значение, которые больше 10.
9. Пользователь вводит четыре числа. Вывести минимальное значение.
10. Пользователь вводит одно отрицательное число a и одно положительное число b. Вывести значение a, если оно меньше 10, и вывести значение b, если оно больше 10.

11. Пользователь вводит одно число. Вывести данное число, если оно равно 0, в противные случаи выдать соответствующее сообщение.

Лабораторная работа №4. Тернарный оператор

1. Пользователь вводит число. Если оно больше 0, то необходимо прибавить к нему 1; иначе вычесть из него 2. Вывести полученное число на экран.
2. Пользователь вводит число. Если оно больше 0, то прибавить к нему 1; если меньше 0, то вычесть из него 2; если равно нулю, то заменить его на 10. Вывести полученное число на экран.
3. Пользователь вводит три целых числа. Необходимо найти и вывести количество положительных чисел из данного набора.
4. Пользователь вводит три целых числа. Необходимо найти и вывести количество положительных и количество отрицательных чисел в данном наборе.
5. Пользователь вводит два числа. Вывести то число, которое является большим.
6. Пользователь вводит два числа. Вывести значения в порядке возрастания.
7. Пользователь вводит два числа. Вывести значения в порядке убывания.
8. Пользователь вводит три числа. Вывести только те значение, которые больше 10.
9. Пользователь вводит четыре числа. Вывести минимальное значение.
10. Пользователь вводит одно отрицательное число a и одно положительное число b. Вывести значение a, если оно меньше 10, и вывести значение b, если оно больше 10.
11. Пользователь вводит одно число. Вывести данное число, если оно равно 0, в противные случаи выдать соответствующее сообщение.

Лабораторная работа №5. Оператор выбора

1. Пользователь вводит два числа и один символ (+, -, *, /). Вывести результат операции над числами в зависимости от выбранного символа.
2. Пользователь вводит одно число и выбирает одно действие (1-милиметр, 2-сантиметр, 3-дециметр, 4-метр, 5-километр). Вывести результат значения в зависимости от выбранного действия.
3. Пользователь вводит одно число (месяц). Вывести количество дней в зависимости от выбранного месяца.

4. Пользователь вводит одно число от 1 до 12. Вывести строку с правильным окончанием «часа», например, «1 час», «2 часа», «6 часов».
5. Пользователь вводит возраст человека в диапазоне от 10 до 38 лет. Вывести строку веденного возраста человека, например, «десять лет», «тридцать два года».
Пользователь вводит число от 1 до 12. Вывести название времени года.

Лабораторная работа №6. Циклы с предусловием

1. Найти среднее арифметическое целых чисел.

1	от -17 до -231
2	от -45 до -114
3	от 71 до -51
4	от 423 до 64
5	от 14 до 52
6	от -5 до -29
7	от -83 до -69
8	от 154 до -55
9	от 91 до -5
10	от -47 до 125

2. Написать программу для расчёта формулы в соответствии со своим вариантом.

1	$\frac{1}{2}+\frac{1}{3}+\ldots+(-1)^{n+1}\frac{1}{n}+\ldots$
2	$\left(1+\frac{1}{1^2}\right)\cdot\left(1+\frac{1}{2^2}\right)\ldots\left(1+\frac{1}{n^2}\right)$
3	$\frac{1}{4}+\frac{1}{9}+\ldots+\frac{1}{n^2}+\ldots$
4	$\frac{1}{\sqrt{2}}-\frac{1}{\sqrt{3}}+\frac{1}{\sqrt{4}}+\ldots+\frac{(-1)^{n+1}}{\sqrt{n+1}}+\ldots$
5	$S=1-\frac{n}{2\cdot 3}+\left(\frac{n}{3\cdot 4}\right)^2-\left(\frac{n}{4\cdot 5}\right)^3+\ldots+(-1)^k\left(\frac{n}{(k+1)(k+2)}\right)^k+\ldots$
6	$1 + 1/2 + 1/3 + \ldots + 1/N$
7	$1.1 \cdot 1.2 \cdot 1.3 \cdot \ldots$
8	$1.1 – 1.2 + 1.3 – \ldots$
9	$1 + A + A^2 + A^3 + \ldots + A^N$.

Лабораторная работа №7. Циклы с предусловием

1. Найти среднее арифметическое целых чисел.

1	от -13 до -231
2	от -45 до -174
3	от 74 до -12
4	от 313 до 64
5	от 84 до 0
6	от -5 до -29
7	от -53 до -19
8	от 154 до -26
9	от 51 до -5
10	от -412 до 125

2. Написать программу для расчёта формулы в соответствии со своим вариантом.

1	$\frac{1}{2}+\frac{1}{3}+\ldots+(-1)^{n+1}\frac{1}{n}+\ldots$
2	$\left(1+\frac{1}{1^2}\right)\cdot\left(1+\frac{1}{2^2}\right)\ldots\left(1+\frac{1}{n^2}\right)$
3	$\frac{1}{4}+\frac{1}{9}+\ldots+\frac{1}{n^2}+\ldots$
4	$\frac{1}{\sqrt{2}}-\frac{1}{\sqrt{3}}+\frac{1}{\sqrt{4}}+\ldots+\frac{(-1)^{n+1}}{\sqrt{n+1}}+\ldots$
5	$S=1-\frac{n}{2\cdot 3}+\left(\frac{n}{3\cdot 4}\right)^2-\left(\frac{n}{4\cdot 5}\right)^3+\ldots+(-1)^k\left(\frac{n}{(k+1)(k+2)}\right)^k+\ldots$
6	$1 + 1/2 + 1/3 + \ldots + 1/N$
7	$1.1 \cdot 1.2 \cdot 1.3 \cdot \ldots$
8	$1.1 – 1.2 + 1.3 – \ldots$
9	$1 + A + A^2 + A^3 + \ldots + A^N$.

Лабораторная работа №8. Циклы с параметром

1. Найти среднее арифметическое целых чисел.

1	от -13 до -231
2	от -45 до -174
3	от 74 до -12
4	от 613 до 245
5	от 14 до 100
6	от -5 до -29
7	от -53 до -69
8	от 154 до -25
9	от 231 до -5
10	от -47 до 125

2. Написать программу для расчёта формулы в соответствии со своим вариантом.

1	$\frac{1}{2}+\frac{1}{3}+\ldots+(-1)^{n+1}\frac{1}{n}+\ldots$
2	$\left(1+\frac{1}{1^2}\right)\cdot\left(1+\frac{1}{2^2}\right)\ldots\left(1+\frac{1}{n^2}\right)$
3	$\frac{1}{4}+\frac{1}{9}+\ldots+\frac{1}{n^2}+\ldots$
4	$\frac{1}{\sqrt{2}}-\frac{1}{\sqrt{3}}+\frac{1}{\sqrt{4}}+\ldots+\frac{(-1)^{n+1}}{\sqrt{n+1}}+\ldots$
5	$S=1-\frac{n}{2\cdot 3}+\left(\frac{n}{3\cdot 4}\right)^2-\left(\frac{n}{4\cdot 5}\right)^3+\ldots+(-1)^k\left(\frac{n}{(k+1)(k+2)}\right)^k+\ldots$
6	$1 + 1/2 + 1/3 + \ldots + 1/N$
7	$1.1 \cdot 1.2 \cdot 1.3 \cdot \ldots$
8	$1.1 – 1.2 + 1.3 – \ldots$
9	$1 + A + A^2 + A^3 + \ldots + A^N$.

Лабораторная работа №9. Вложенные циклы

1. Пользователем введена цепочка чисел размером N. Необходимо рассчитать сумму всех нечётных целых числе в диапазоне, который введёт пользователь с клавиатуры (в переменную N).

2. Пользователем введена цепочка чисел размером N. Необходимо рассчитать произведение всех чётных целых числе в диапазоне, который введёт пользователь с клавиатуры (в переменную N).

3. Пользователем введена цепочка чисел размером N. Необходимо рассчитать сумму каждого третьего числа в диапазоне, который введёт пользователь с клавиатуры (в переменную N).
4. Пользователь вводит два числа A и B. Необходимо вывести результат функции: 2A+4A+...+AB.
5. Пользователь вводит два числа A и B. Необходимо вывести результат функции: 1/A+2/A+...+1/B.
6. Пользователем введена цепочка чисел размером N. Необходимо рассчитать сколько раз встречается цифра A в введенной последовательности чисел, который введёт пользователь с клавиатуры (в переменную N).
7. Пользователь вводит два числа A и B. Необходимо вывести результат функции: 1A+3A+...+AB.
8. Пользователь вводит два числа A и B. Необходимо вывести результат функции: 1/A+3/A+5/A ...+1/B.

Лабораторная работа №10. Подпрограммы (функции или методы)

1. Пользователь вводит трехзначное число. Необходимо переставить сотое число и единицу исходного числа, например, 321 перейдет в 123.
2. Пользователь вводит трехзначное число. Необходимо переставить десятичное число и единицу исходного числа, например, 321 перейдет в 312.
3. Пользователь вводит число A. Необходимо посчитать факториал данного числа.
4. Пользователь вводит число N. Необходимо вывести результат функции, $1! + 2! + 3! + \ldots + N!$
5. Пользователь вводит число N. Необходимо вывести результат функции, $1 - X^2/(2!) + X^4/(4!) - \ldots + (-1)^N \cdot X^{2 \cdot N}/((2 \cdot N)!)$
6. Пользователь вводит значение в двоичной системе счисления. Необходимо перевести и вывести результат в десятеричной С/С.
7. Пользователь вводит значение в десятеричной системе счисления. Необходимо перевести и вывести результат в восьмеричной С/С.
8. Пользователь вводит значение в восьмеричной системе счисления. Необходимо перевести и вывести результат в двоичной С/С.

9. Пользователь вводит значение в десятеричной системе счисления. Необходимо перевести и вывести результат в двоичной С/С.
10. Пользователь вводит значение в двоичной системе счисления. Необходимо перевести и вывести результат в шестнадцатеричной С/С.

Лабораторная работа №11. Перегрузка методов

Написать функции для расчёта формул в соответствии со своим вариантом. Первая функция должна возвращать пустое значение и выводить на экран результаты расчетов по формуле. Вторая функция, возвращать значение и выводить результат в переменную.

1) $\dfrac{2*c - d + \sqrt{23}}{\frac{a}{4} - 1}$ $\qquad$ $\dfrac{c + 4*d - \sqrt{123}}{1 - \frac{a}{2}}$

2) $\dfrac{-2*c + d*82}{tg(\frac{a}{4} - 1)}$ $\qquad$ $\dfrac{lg(2*c) + d - 52}{\frac{a}{4} + 1}$

3) $\dfrac{arctg(c/4) - d*62}{a*a - 1}$ $\qquad$ $\dfrac{-2*c - ln(d) + 53}{\frac{a}{4} - 1}$

4) $\dfrac{2*c - lg(d/4)}{a*a - 1}$ $\qquad$ $\dfrac{tg(c) - d*23}{2*a - 1}$

5) $\dfrac{2*c - d/23}{ln(1 - \frac{a}{4})}$ $\qquad$ $\dfrac{4*c + d - 1}{c - tg\frac{a}{2}}$

6) $\dfrac{2*c - d*\sqrt{42}}{c + a - 1}$ $\qquad$ $\dfrac{\sqrt{\frac{25}{c}} - d + 2}{d + a*a - 1}$

7) $arctg(c - d/2)$ $\qquad$ $4*lg(c) - d/2 + 23$

$$\frac{}{2*a - 1} \qquad \frac{}{a*a - 1}$$

8) $$\frac{c*tg(b + 23)}{a/2 - 4*d - 1} \qquad \frac{c/d + \ln(3*a/2)}{c - a + 1}$$

9) $$\frac{2*c + \lg(d)*51}{d - a - 1} \qquad \frac{2*c + \ln(d/4) + 23}{a*a - 1}$$

10) $$\frac{42*c - d/2 + 1}{a*a - \ln(b-5)} \qquad \frac{arctg(2*c)/d + 2}{d - a*a - 1}$$

Лабораторная работа №12. Одномерные массивы. Форматированный вывод элементов массива

1. Дан одномерный массив размера 10. Необходимо вывести только нечетные значения одномерного массива.
2. Дан одномерный массив размера 10. Необходимо вывести только четные значения одномерного массива.
3. Дан одномерный массив размера 10. Необходимо вывести только значения, которые кратны двум.
4. Дан одномерный массив размера 5. Необходимо вывести только значения, которые кратны трем.
5. Дан одномерный массив размера 15. Необходимо вывести одномерный массив в обратном порядке.
6. Дан одномерный массив размера 8. Необходимо вывести значения одномерного массива имеющий нечетный индекс.
7. Дан одномерный массив размера 8. Необходимо вывести значения одномерного массива имеющий четный индекс.
8. Дан одномерный массив размера 8. Необходимо вывести произведение значений одномерного массива.

Лабораторная работа №13. Одномерные массивы. Анализ элементов массив

1. Пользователь вводит значения в одномерный массив размера 8. Необходимо вывести средние арифметическое значений одномерного массива.
2. Пользователь вводит значения в одномерный массив размера 8. Пользователь вводит целочисленные значения k и l. Необходимо посчитать сумму значений одномерного массива от индекса k до индекса l.
3. Пользователь вводит значения в одномерный массив размера 10. Пользователь вводит целочисленные значения k и l. Необходимо посчитать произведение значений одномерного массива от индекса k до индекса l.
4. Пользователь вводит значения в одномерный массив размера 5. Пользователь вводит целочисленные значения k и l. Необходимо посчитать разницу значений одномерного массива от индекса k до индекса l.
5. Пользователь вводит значения в одномерный массив размера 5. Пользователь вводит целочисленные значения k и l. Необходимо посчитать среднее арифметическое значение одномерного массива от индекса k до индекса l.
6. Пользователь вводит значения в одномерный массив размера 8. Необходимо вывести максимальное элемент.
7. Пользователь вводит значения в одномерный массив размера 12. Необходимо вывести максимальное элемент.
8. Пользователь вводит значения в одномерный массив размера 9. Необходимо вывести максимальное элемент с четным индексом.
9. Пользователь вводит значения в одномерный массив размера 13. Необходимо вывести максимальное элемент с нечетным индексом.

Тема №14. Одномерные массивы. Методы сортировки массива

1. Пользователь вводит значения в одномерный массив размера 13. Необходимо отсортировать одномерный массив по возрастанию.
2. Пользователь вводит значения в одномерный массив размера 11. Необходимо отсортировать одномерный массив по убыванию.
3. Пользователь вводит значения в одномерный массив размера 13. Необходимо в отдельный одномерный массив вывести только четные значения и отсортировать по возрастанию.
4. Пользователь вводит значения в одномерный массив размера 9. Необходимо в отдельный одномерный массив вывести только нечетные значения и отсортировать по возрастанию.

5. Пользователь вводит значения в одномерный массив размера 6. Необходимо в отдельный одномерный массив вывести только четные значения и отсортировать по убыванию.
6. Пользователь вводит значения в одномерный массив размера 10. Необходимо в отдельный одномерный массив вывести только нечетные значения и отсортировать по убыванию.
7. Реализовать сортировку выбором.
8. Реализовать пузырьковую сортировку.
9. Реализовать сортировку вставками.

Тема №15. Двухмерный массивы. Подсчёта данных в матрице

1. Пользователь вводит значения в двухмерного массив размера 3х3. Необходимо в отдельный одномерный массив вывести только нечетные значения и вывести сумму этих значений.
2. Пользователь вводит значения в двухмерного массив размера 3х3. Необходимо в отдельный одномерный массив вывести только четные значения и вывести сумму этих значений.
3. Пользователь вводит значения в двухмерного массив размера 4х4. Необходимо вывести среднее арифметическое каждого столбца.
4. Пользователь вводит значения в двухмерного массив размера 4х4. Необходимо вывести среднее арифметическое каждой строки.
5. Пользователь вводит значения в двухмерного массив размера 4х4. Необходимо вывести количество четных чисел.
6. Пользователь вводит значения в двухмерного массив размера 4х4. Необходимо вывести количество нечетных чисел.
7. Пользователь вводит значения в двухмерного массив размера 3х3. Необходимо вывести номер столбца, в котором не повторяются значения.
8. Пользователь вводит значения в двухмерного массив размера 3х3. Необходимо вывести номер строки, в котором повторяются значения.
9. Пользователь вводит значения в двухмерного массив размера3х3. Необходимо вывести номер столбца, в котором не повторяются значения.
10. Пользователь вводит значения в двухмерного массив размера 3х3. Необходимо вывести номер строки, в котором повторяются значения.

Тема №16. Двухмерный массивы. Обход диагоналей матрицы

1. Пользователь вводит значения в двухмерного массив размера 3x3. Необходимо вывести сумму элементов главной диагонали.
2. Пользователь вводит значения в двухмерного массив размера 3x3. Необходимо вывести сумму элементов побочной диагонали.
3. Пользователь вводит значения в двухмерного массив размера 3x3. Необходимо вывести среднее арифметическое значение главной диагонали.
4. Пользователь вводит значения в двухмерного массив размера 3x3. Необходимо вывести среднее арифметическое значение побочной диагонали.
5. Пользователь вводит значения в двухмерного массив размера 3x3. Необходимо вывести максимальное значение главной диагонали.
6. Пользователь вводит значения в двухмерного массив размера 3x3. Необходимо вывести минимальное значение побочной диагонали.

Пользователь вводит значения в двухмерного массив размера 3x3. Необходимо обнулить элементы выше главной диагонали. Операторы условия не использовать.

Тема №17. Двухмерный массивы. Преобразование матрицы.

1. Пользователь вводит значения в двухмерного массив размера 2x2. Необходимо поменять местами левую и правую половины матрицы.
2. Пользователь вводит значения в двухмерного массив размера 4x4. Необходимо преобразовать двухмерный массив в одномерный массив и вывести сумму одномерного массива.
3. Пользователь вводит значения в двухмерного массив размера NxM. Необходимо преобразовать двухмерный массив в одномерный массив и вывести среднее арифметическое одномерного массива.
4. Пользователь вводит значения в двухмерного массив размера NxM и целочисленные значения A и B. Необходимо поменять местами в двухмерном массиве строки A и B.
5. Пользователь вводит значения в двухмерного массив размера NxM. Необходимо аннулировать столбец с минимальным значением.
6. Пользователь вводит значения в двухмерного массив размера NxM. Необходимо аннулировать столбец с максимальным значением.
7. Пользователь вводит значения в двухмерного массив размера NxM. Необходимо вывести столбец, имеющий минимальное значение в одномерный массив.

Пользователь вводит значения в двухмерного массив размера NxM. Необходимо вывести столбец, имеющий максимальное значение в одномерный массив.

Тема №18. Двухмерный массивы. Методы сортировки массива.

Пользователь вводит значения в одномерный массив размера N. Необходимо сформировать из него двухмерный отсортированный змейкой:

1. Начиная с левого верхнего угла вертикально.
2. Начиная с левого верхнего угла горизонтально.
3. Начиная с правого верхнего угла вертикально.
4. Начиная с правого верхнего угла горизонтально.
5. Начиная с левого нижнего угла вертикально.
6. Начиная с левого нижнего угла горизонтально.
7. Начиная с правого нижнего угла вертикально.
8. Начиная с правого нижнего угла горизонтально.

Тема №19. Двухмерный массивы. Методы поиска элементов массива.

1. Пользователь ввел значения в двухмерного массив размера NxM. Необходимо найти элементы, которые больше 20 и вывести номер элемента.
2. Пользователь ввел значения в двухмерного массив размера NxM. Необходимо найти элементы, которые меньше 70 и вывести номер элемента.
3. Пользователь ввел значения в двухмерного массив размера NxM. Необходимо найти элементы, которые больше 15 и вывести в отдельный одномерный массив.
4. Пользователь ввел значения в двухмерного массив размера NxM. Необходимо найти элементы, которые больше 20, вывести в отдельный одномерный массив и отсортировать по возрастанию.
5. Пользователь ввел значения в двухмерного массив размера NxM. Необходимо найти элементы, которые меньше 38, вывести в отдельный одномерный массив и отсортировать по убыванию.
6. Пользователь ввел значения в двухмерного массив размера NxM. Необходимо найти элементы, которые меньше 38, вывести в отдельный одномерный массив и отсортировать по возрастанию.

7. Пользователь ввел значения в двухмерного массив размера NxM. Необходимо найти элементы, которые меньше 85, вывести в отдельный одномерный массив и найти минимальный элемент.
8. Пользователь ввел значения в двухмерного массив размера NxM. Необходимо найти элементы, которые больше 1, вывести в отдельный одномерный массив и найти максимальный элемент.

Лабораторная работа №20. Символьный тип данных

1. Пользователь ввел символ с клавиатуры. Необходимо вывести на экран номер введённого символа из таблицы ASCII.
2. Пользователь ввел число в диапазоне от 32 до 126. Необходимо вывести на экран символ с таким кодом.
3. Пользователь ввел символ с клавиатуры. Вывести на экран символ с предыдущим и последующим кодом.
4. Пользователь ввел число в диапазоне от 1 до 26. Вывести на экран заглавные буквы латинского алфавита до первого символа до значения введённого числа включительно.
5. Пользователь ввел число в диапазоне от 1 до 26. Вывести на экран строчные буквы латинского алфавита до последнего символа до значения введённого числа включительно.
6. Дан символ C, изображающий цифру или букву (латинскую или русскую). Если C изображает цифру, то вывести строку «digit», если латинскую букву — вывести строку «lat», если русскую — вывести строку «rus».
7. Пользователь ввел строку с клавиатуры. Найти код первой и последней буквы в строке.
8. Пользователь ввел целое число больше 0 и символ с клавиатуры. Вывести строку размером с целое число, состоящая из введенных символов.

Пользователь ввел целое число больше 0 и два символа с клавиатуры. Вывести строку размером с целое число, состоящая из чередующихся введенных символов.

Лабораторная работа №21. Текстовые файлы

1. Пользователь ввел строку с клавиатуры. Развернуть строку в обратном порядке и вывести на экран.
2. Пользователь ввел две строки с клавиатуры. Вывести данные о количестве подстрок содержащее символы первой строки, между которыми вставлено по одному пробелу.

3. Пользователь ввел строку с клавиатуры и целое число больше 0. Вывести на экран новую строку, в которой между каждым символом присутствует символ звездочка в количестве значения числа введённым пользователем.
4. Пользователь ввел строку из цифр и букв. Найти количество введенных цифр. Размер строки произвольный.
5. Пользователь ввел строку из цифр и букв. Найти в ней количество прописных латинских букв.
6. Пользователь ввел строку из цифр и букв разной раскладки. Найти в ней количество прописных латинских букв, русских букв. Вывести с подписями на экран.
7. Пользователь ввел строку, все строчные буквы превратить в прописные и наоборот.
8. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные.
9. Пользователь ввел целое положительное число, вывести первые буквы для названия каждой цифры в нем.

Лабораторная работа №22. Текстовые файлы

1. Пользователь ввел название файла и два целых положительных числа. Сформировать файл с введенным именем и записать в него такое количество строк как первое число, каждая из которых состоит из символов «*» в количестве второго введенного числа.
2. Пользователь ввел название файла и целое положительное число с клавиатуры. Сформировать текстовый файл с указанным именем и записать в него количество строк, указанных пользователем. Формирование текстового файла происходит в формате: первая строка должна содержать строчную (то есть маленькую) латинскую букву «a», вторая — буквы «ab», третья — буквы «abc» и т. д.; последняя строка должна содержать N начальных строчных латинских букв в алфавитном порядке.
3. Пользователь ввел название файла и целое положительное число с клавиатуры. Создать текстовый файл с указанным именем и записать в него указанное количество строк длины, строка с номером K (K = 1, ... ,N) должна содержать K начальных прописных (то есть заглавных) латинских букв, дополненных справа символами «*» (звездочка). Например, для N = 4 файл должен содержать строки «A***», «AB**», «ABC*», «ABCD».
4. Дан текстовый файл. Вывести количество содержащихся в нем символов и строк (маркеры концов строк EOLN и конца файла EOF при подсчете количества символов не учитывать).

5. Создать текстовый файл и ввести с клавиатуры строку S, введенную строку необходимо записать в конец файла.
6. Создать два текстовых файла. Количество строк первого файла определяет пользователь, введя значение с клавиатуры, количество строк второго файла равна длинна первого умноженного на три. В каждом из файлов данные должны генерироваться в случайном порядке. Значение из второго файла записать в конец первого. Все три файла вывести на экран.
7. Создать текстовый файл и ввести с клавиатуры строку S, введенную строку необходимо записать в начало файла.

Лабораторная работа №23. Бинарные файлы

1. Пользователь создал файл с целыми числами от 0 до 9. Создать два новых файла для четных и нечетных чисел. Вывести значения на экран.
2. Пользователь создал файл с целыми числами от 0 до 9 Создать два новых файла для четных и нечетных чисел. Если четные или нечетные числа в исходном файле отсутствуют, то соответствующий результирующий файл оставить пустым.
3. Пользователь создал файл с целыми числами от 0 до 9. Создать два новых файла, для положительных и отрицательных чисел. Если положительные или отрицательные числа в исходном файле отсутствуют, то соответствующий результирующий файл оставить пустым.
4. Пользователь создал файл со случайными вещественными числами в диапазоне от 0 до 27. Вывести среднее арифметическое чисел из файла на экран.
5. Пользователь создал файл со случайными вещественными числами в диапазоне от 0 до 31. Вывести сумму четных элементов.
6. Пользователь создал файл с целыми числами от 0 до 9. Найти количество содержащихся в нем серий чисел.
7. Пользователь создал файл с целыми числами от 0 до 9. Сформировать новый файл, содержащий длины всех серий исходного файла.

Лабораторная работа №24. Классы и объекты

1. Создать класс Animals (животные), который будет содержать поля с открытым доступом: type_animal, kindOfAnimal, price и метод класса getData(). В главной функции объявить пару объектов класса и внести данные в поля. Затем отобразить их, вызвав метод getData().

2. Создать класс с двумя переменными. Добавить функцию вывода на экран и функцию изменения этих переменных. Добавить функцию, которая находит сумму значений этих переменных, и функцию, которая находит наибольшее значение из этих двух переменных.
3. Создать класс с двумя переменными. Добавить функцию вывода на экран и функцию изменения этих переменных. Добавить функцию, которая находит произведение значений этих переменных, и функцию, которая находит наибольшее значение из этих двух переменных.
4. Создать класс с двумя переменными. Добавить функцию вывода на экран и функцию изменения этих переменных. Добавить функцию, которая находит разность значений этих переменных, и функцию, которая находит наименьшее значение из этих двух переменных.
5. Создать класс Product (Продукт), который будет содержать поля с открытым доступом: type_product, name_product, price и метод класса getData(). В главной функции объявить пару объектов класса и внести данные в поля. Затем отобразить их, вызвав метод getData().
6. Создать класс Student (Студент), который будет содержать поля с открытым доступом: FIO, group, AGV(средний балл) и метод класса getData(). В главной функции объявить пару объектов класса и внести данные в поля. Затем отобразить их, вызвав метод getData().
7. Создать класс Student (Студент), который будет содержать поля с открытым доступом: FIO, group, array_evaluation(массив оценок) и метод класса getData(). В главной функции объявить пару объектов класса и внести данные в поля. Затем отобразить их, вызвав метод getData() и посчитать средний балл студента.
8. Создать класс с массивом числовых значений. Добавить функцию вывода на экран и функцию изменения этого массива данных. Добавить функцию, которая находит средние значение этого массива, и функцию, которая находит наименьшее значение из этого массива.
9. Создать класс с массивом числовых значений. Добавить функцию вывода на экран и функцию изменения этого массива данных. Добавить функцию, которая находит средние значение этого массива, и функцию, которая находит наибольшее значение из этого массива.
10. Создать класс с массивом числовых значений. Добавить функцию вывода на экран и функцию изменения этого массива данных. Добавить функцию, которая находит средние значение этого массива, и функцию, которая находит все положительные значение из этого массива.

Лабораторная работа №25. Структуры

1. Создать структуру Product (Продукт), который будет содержать поля с открытым доступом: type_product, name_product, price и метод класса getData(). В главной функции объявить пару объектов структуры и внести данные в поля. Затем отобразить их, вызвав метод getData().
2. Создать структуру Student (Студент), который будет содержать поля с открытым доступом: FIO, group, AGV (средний балл) и метод класса getData(). В главной функции объявить пару объектов структуры и внести данные в поля. Затем отобразить их, вызвав метод getData().
3. Создать структуру Student (Студент), который будет содержать поля с открытым доступом: FIO, group, array_evaluation(массив оценок) и метод класса getData(). В главной функции объявить пару объектов класса и внести данные в поля. Затем отобразить их, вызвав метод getData() и посчитать средний балл студента.
4. Создать класс Animals (животные), который будет содержать поля с открытым доступом: type_animal, kindOfAnimal, price и метод класса getData(). В главной функции объявить пару объектов структуры и внести данные в поля. Затем отобразить их, вызвав метод getData().
5. Создать структуру Student (Студент), который будет содержать поля с открытым доступом: FIO, group, array_evaluation(массив оценок). В главной функции объявить пару объектов структуры и внести данные в поля. Затем вывести данные только тех студентов чей средний балл больше 4.0, если нет таких студентов, то выдать соответствующее сообщение.
6. Создать структуру Student (Студент), который будет содержать поля с открытым доступом: FIO, group, array_evaluation(массив оценок). В главной функции объявить пару объектов структуры и внести данные в поля. Затем вывести данные только тех студентов чьи оценки равны 4 и 5, если нет таких студентов, то выдать соответствующее сообщение.
7. Создать структуру Student (Студент), который будет содержать поля с открытым доступом: FIO, group, array_evaluation (массив оценок). В главной функции объявить пару объектов структуры и внести данные в поля. Затем вывести данные только тех студентов, которые имеют хотя бы одну оценку 2, если нет таких студентов, то выдать соответствующее сообщение.
8. Создать структуру Route (Маршрут), который будет содержать поля с открытым доступом: start_route, end_route, number_route. В главной функции объявить пару объектов структуры и внести данные в поля. Затем данные всех маршрутов.

9. Создать структуру Student (Студент), который будет содержать поля с открытым доступом: FIO, group, array_evaluation(массив оценок). В главной функции объявить пару объектов структуры и внести данные в поля. Затем вывести данные только тех студентов, которые не имеют оценки, если нет таких студентов, то выдать соответствующее сообщение.
10. Создать структуру Info (Информация), который будет содержать поля с открытым доступом: FIO, phone_number, DOB (дата рождения). В главной функции объявить пару объектов структуры и внести данные в поля. Затем вывети информацию о человеке, чьи фамилия введена с клавиатуры.

РАЗДЕЛ 2.1. ИНСТРУКЦИЯ ПО ОРГАНИЗАЦИИ И ПРОВЕДЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО СПЕЦКУРСУ «ПРОГРАММИРОВАНИЕ» ДЛЯ 11 ИТ-КЛАССА

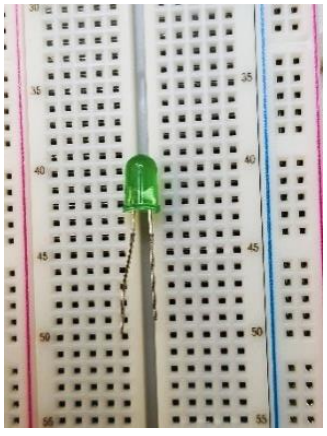
Лабораторная работа № 1. Знакомство с платой Arduino

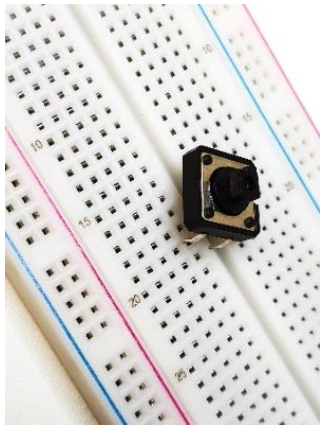
Для закрепления практических навыков программирования удобно использовать платформу Arduino, которая создавалась именно для целей обучения. Среди семейства Arduino наиболее популярной платой является плата Arduino UNO.

Не углубляясь в технические подробности, скажем, что на плате имеются 14 цифровых и 6 аналоговых контактов, которые могут работать как в режиме входа (принимают сигнал извне), так и в режиме выхода (выводят сигнал на управляемое устройство) (Рисунок 1).

Рисунок 1

В качестве устройств, которыми будет управлять плата Arduino, в этой лабораторной работе будем использовать светодиоды и кнопки (Рисунок 2).





а б

Рисунок 2

а) светодиод; б) кнопка

Для сборки схем и обеспечения их работоспособности, будем также использовать «вспомогательные» компоненты:

- макетные платы (Breadboard на 800 точек) (Рисунок 3);
- резисторы (Рисунок 4)
- соединительные провода.

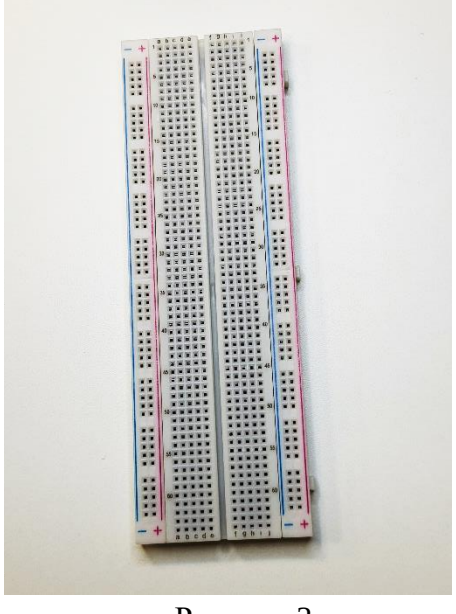


Рисунок 3

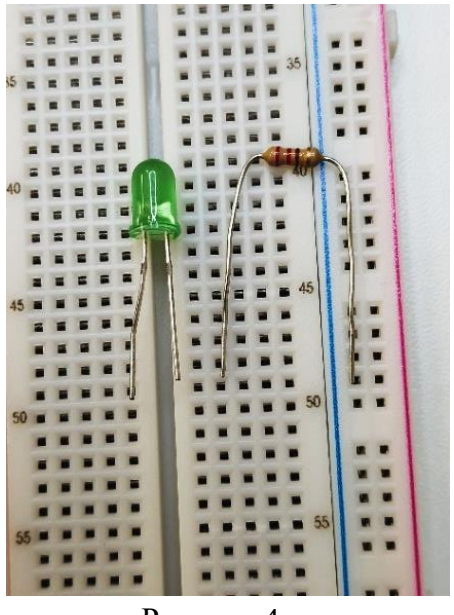


Рисунок 4

Для написания программного кода будем использовать среду разработки Arduino IDE. Эта среда является свободно распространяемым программным обеспечением. Скачать её можно с официального сайта https://www.arduino.cc//. Ниже описан процесс скачивания и установки среды.

- Перейдите в раздел SOFTWARE (Рисунок 5):

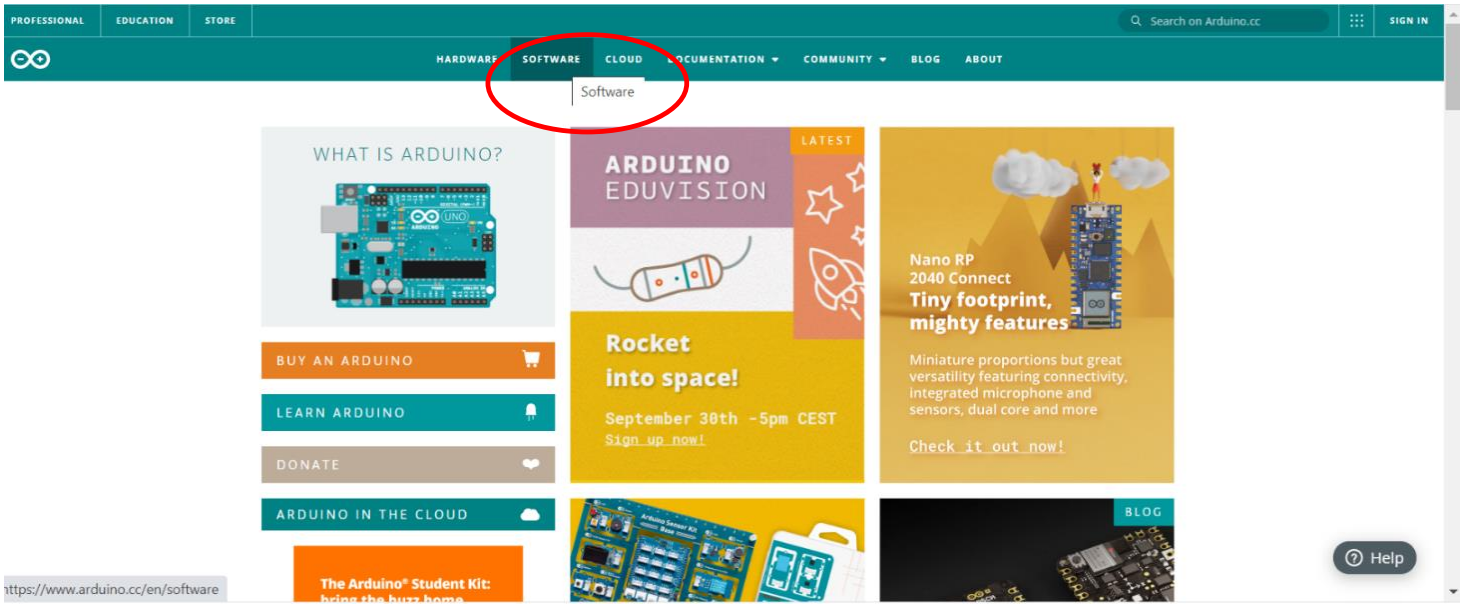


Рисунок 5

- На странице SOFTWARE в разделе ***Downloads*** нажмите ***Windows Win 7 and newer*** (Рисунок 6):

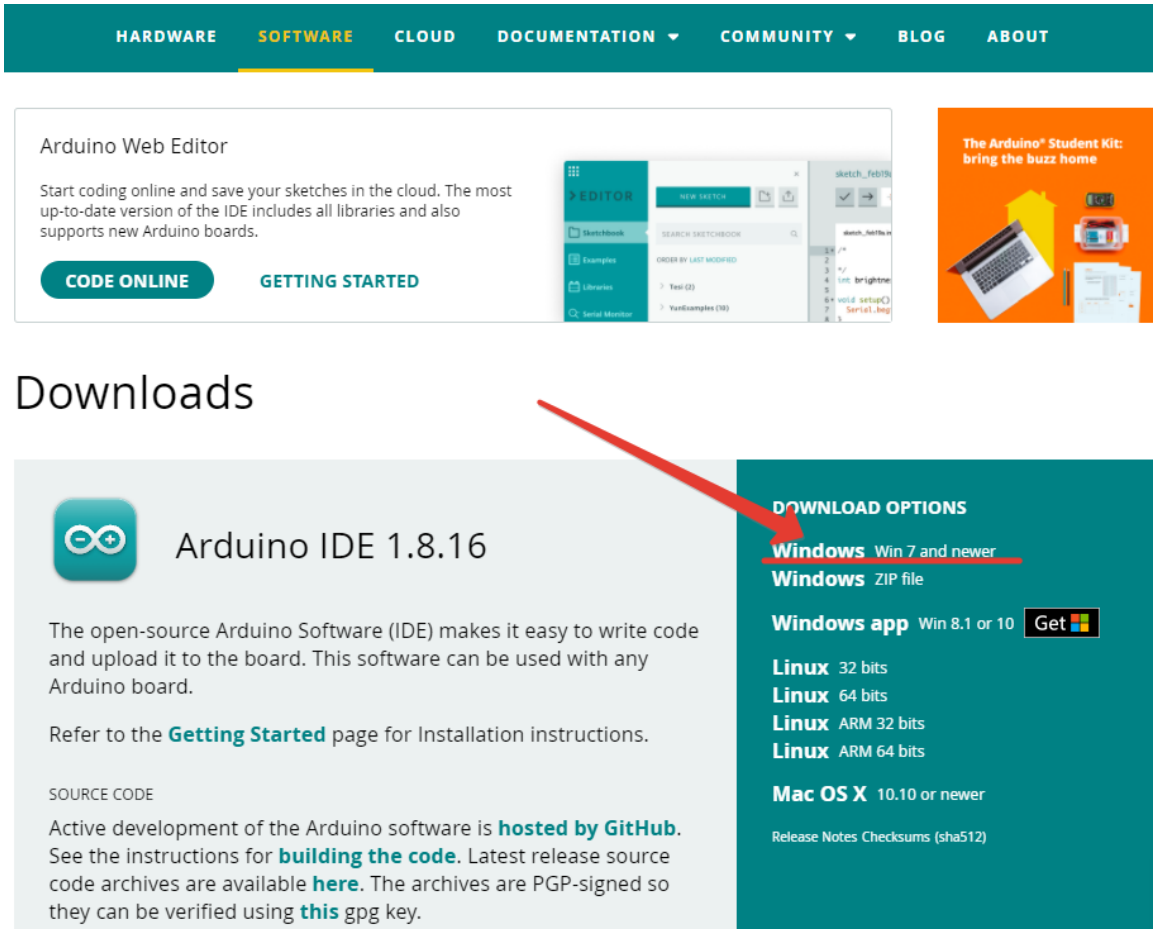


Рисунок 6

- Откроется страница, на которой необходимо нажать на кнопку ***Just download*** (Рисунок 7):

Рисунок 7

- После окончания скачивания файла с примерным именем ***arduino-1.8.16-windows.ex***e запустите установку программы.

- Откроется приветственное окно установщика. Нажмите кнопку "***I Agree***" (Рисунок 8):

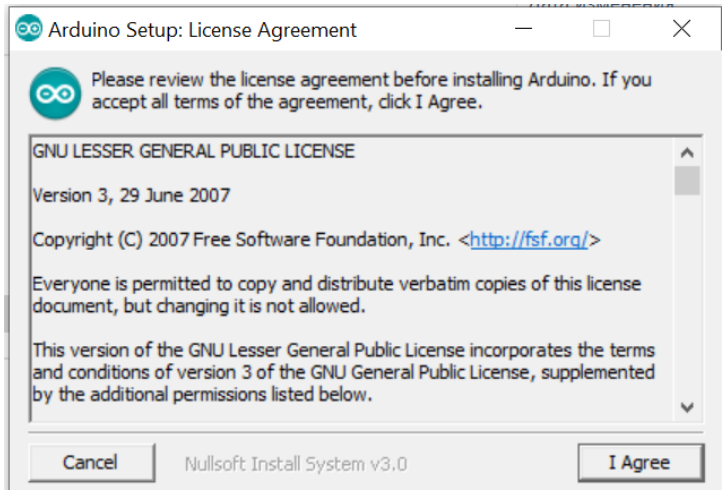


Рисунок 8

- В следующем окне будут отмечены компоненты среды, которые планируются для установки. Нажмите кнопку ***Next*** (Рисунок 9):

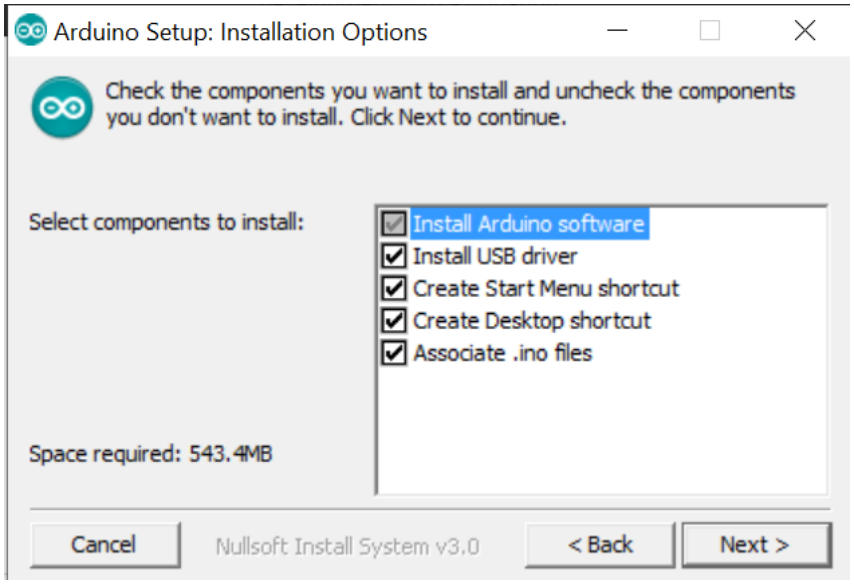


Рисунок 9

- В следующем окне будет показан путь к папке, где планируется установить Arduino IDE. Согласитесь с предложенным выбором или укажите другую папку, после этого нажмите кнопку ***Install*** (Рисунок 10):

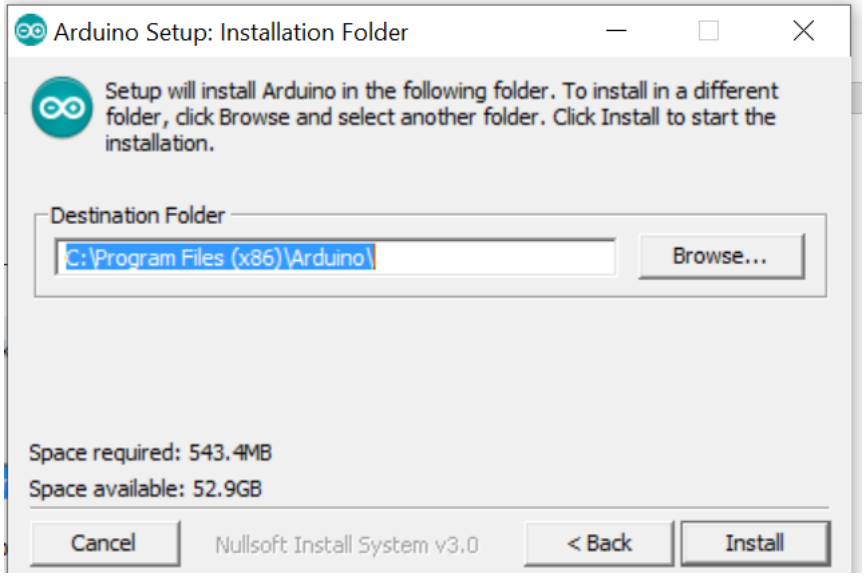


Рисунок 10

- После окончания процесса установки вы увидите следующее окно (Рисунок 11). Нажмите кнопку ***Close*** для закрытия окна и завершения процесса установки.

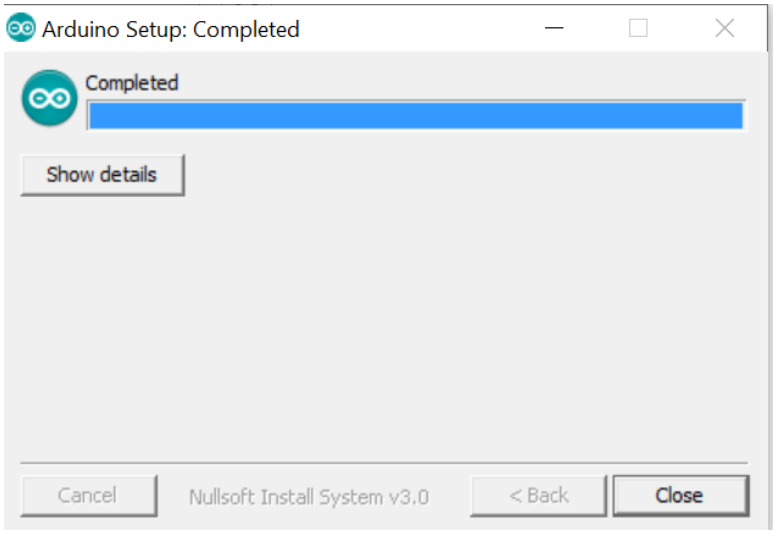


Рисунок 11

Теперь среда Arduino IDE установлена и перед началом работы в ней давайте познакомимся с тем, как выглядит её окно после запуска.

Запустите Arduino IDE, воспользовавшись ярлыком на рабочем столе, который был создан при установке:  .

Примерный вид окна показан ниже (Рисунок 12):

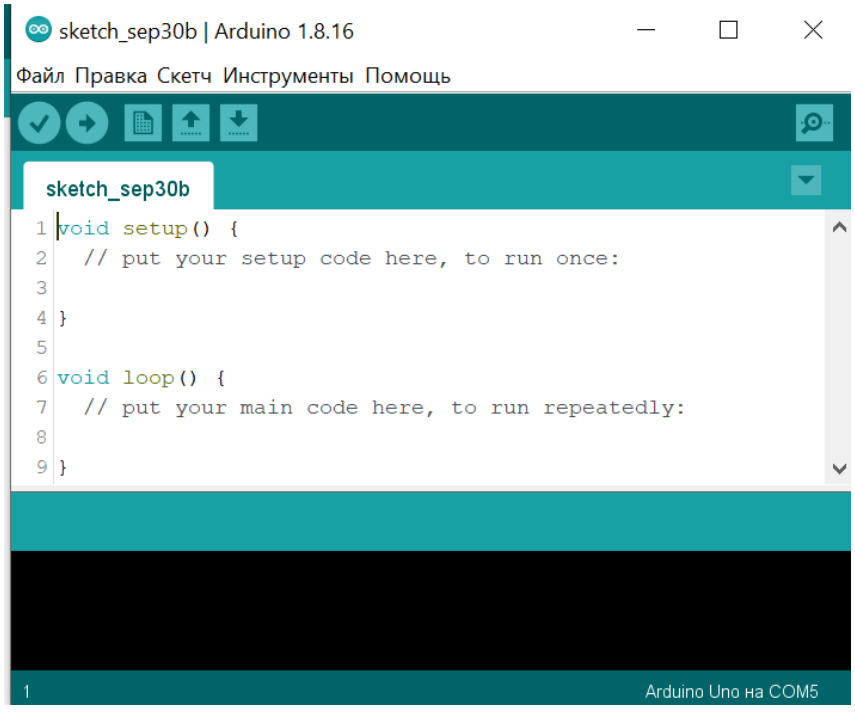


Рисунок 12

На рисунке ниже (Рисунок 13) цифрами обозначены основные области окна среды, рассмотрим их.

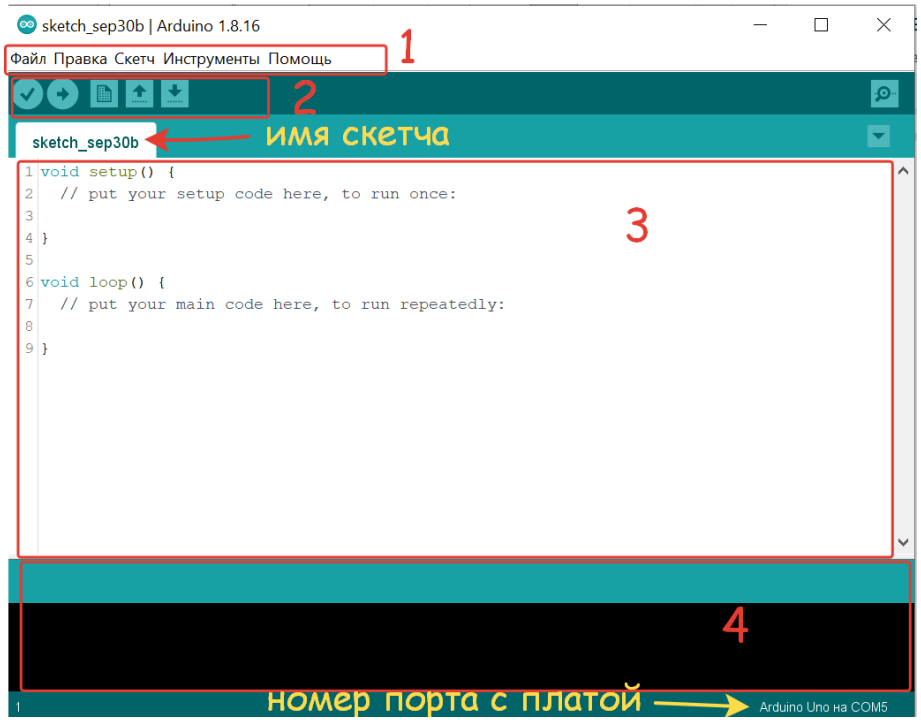


Рисунок 13

Область 1: строка раскрывающихся меню:

- меню ***Файл*** используется для создания и сохранения скетчей (скетчами в среде Arduino IDE принято называть файлы с программным кодом);
- меню ***Правка*** содержит команды настройки оформления кода;
- меню ***Скетч*** используется для подключения библиотек;
- меню ***Инструменты*** используется для настройки подключения платы к персональному компьютеру;
- меню ***Помощь*** содержит справочную информацию.

Область 2: в ней расположены основные инструменты, которые используются в работе:

- кнопка ***Проверить***: выполняет проверку на ошибки и компиляцию программного кода, находящегося в рабочем поле (***Область 3***); этой кнопкой вы будете пользоваться после

того, как напишите программный код, чтобы проверить правильность его написания. Если в коде будут обнаружены ошибки, сообщение об этом будет выведено в нижней части окна, а строка, в которой обнаружена ошибка, будет подсвечена. Например, вот так (Рисунок 14):

Рисунок 14

- кнопка ***Загрузить***: эта кнопка также выполняет компиляцию и проверку на ошибки программного кода из рабочего поля (***Область 3***), но при этом она еще и загружает готовый код в плату Arduino. Обратите внимание, что если вы нажмете эту кнопку, а плата Arduino не будет подключена к компьютеру USB кабелем, то в нижней части окна будет выведено сообщение «***Произошла ошибка при загрузке скетча***» (Рисунок 15).

Рисунок 15

Если загрузка скетча прошла успешно, то сообщение будет таким (Рисунок 16):

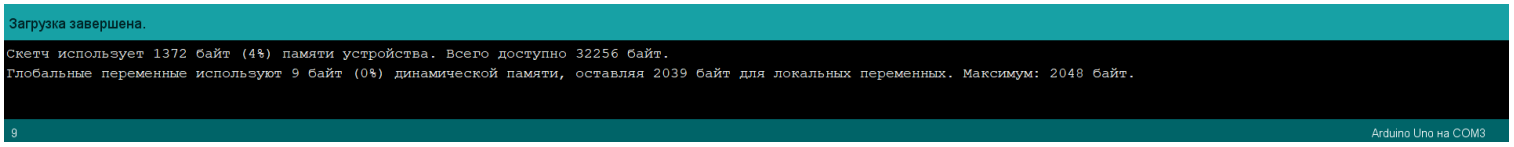


Рисунок 16

Иногда при загрузке скетча возникает проблема, связанная с неправильным определением номера COM-порта (Рисунок 17).

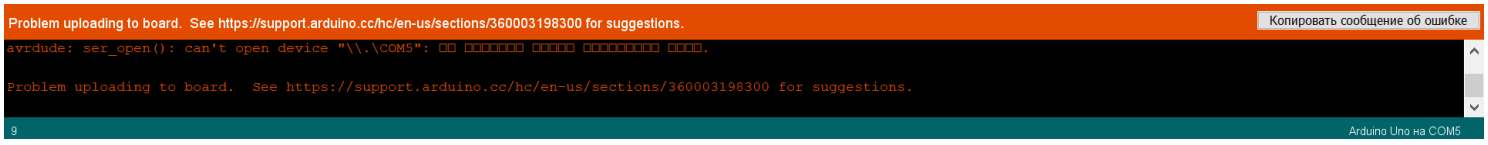


Рисунок 17

В этом случае нужно открыть меню ***Инструменты*** и выбрать предложенный средой номер, например, так, как показано ниже (у вас номера COM-портов могут быть другие) (Рисунок 18).

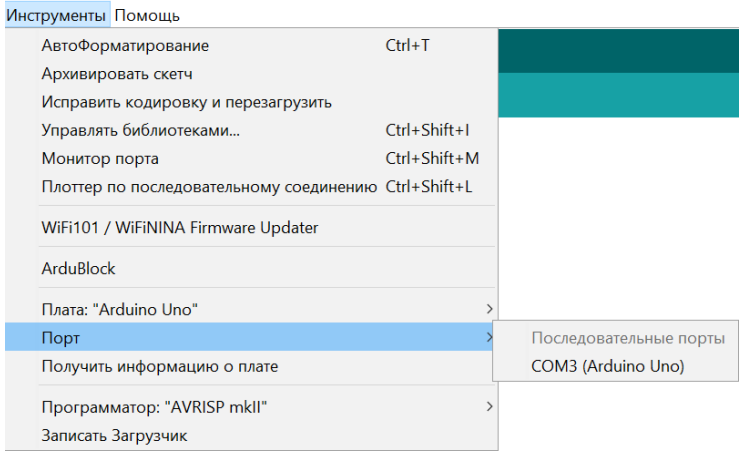


Рисунок 18

- кнопка ***Создать Новый***: этой кнопкой можно воспользоваться для создания нового файла;

- кнопки ***Открыть*** и ***Сохранить*** имеют обычное назначение: открывают существующий файл и сохраняют изменения, внесенные в открытый и редактируемый файл соответственно.

Область 3: это рабочее поле текстового редактора для набора программного кода. В области 3 показан автоматически создаваемый шаблон с обязательными структурами программного кода, к которым относятся:

- функция ***void setup()***;
- функция ***void loop()***;

Это две обязательные функции, которые должны присутствовать в каждом скетче. Все программы, которые вы будете создавать, обязательно будут иметь в своей структуре эти две функции, иначе будут ошибки при компиляции.

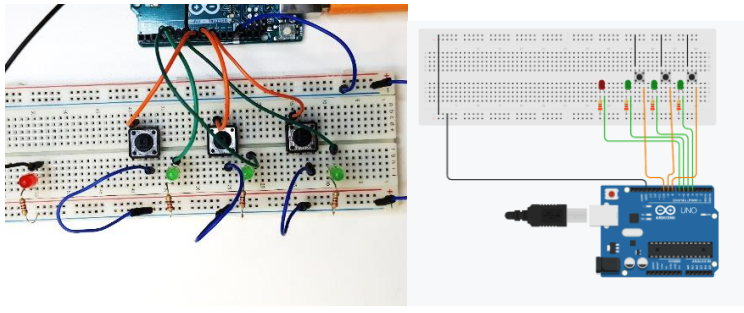
Пока про эти функции нужно знать следующее: функция ***setup()*** будет выполняться однократно, а функция ***loop()*** после выполнения кода, записанного в ней, от начала до конца, снова начнет выполнение кода, находящегося в её теле. Таким образом, функция ***loop()*** является по сути бесконечным циклом, и пока на плату Arduino поступает питание, функция ***loop()*** будет выполняться.

Функция ***setup()*** используется для того, чтобы указать в ней настройки оборудования, которое будет использоваться. Это можно сделать один раз, поэтому для функции ***setup()*** нет необходимости в повторении.

Область 4: Эта область в нижней части окна предназначена для вывода информационных сообщений (например, как на рисунках 14 - 17).

Теперь, когда вы познакомились со средой, в которой будут создаваться программы для платы Arduino, давайте соберем первую схему и напишем код для неё с подробными комментариями.

Задача 1. Собрать схему, которая имитирует работу щита с кнопками и световой индикацией в кабине лифта (Рисунок 19).



а б

Рисунок 19 Общий вид схемы

а) физическое исполнение; б) рисунок схемы

Пояснения. На макетной плате расположены четыре светодиода: три (зеленые) имитируют отображение световой индикации при нажатии на кнопку соответствующего «этажа», а четвертый (красный) мигает во время «движения лифта». Когда не нажата ни одна кнопка, ничего не происходит. Как только в «лифт» зашел пассажир и нажал на кнопку, загорается соответствующий ей светодиод и горит, пока «лифт едет». Красный светодиод в этом время мигает, имитируя индикацию движение «лифта». После «достижения определенного этажа» все светодиоды перестают гореть, и только после нового нажатия на кнопку загорается соответствующий зеленый светодиод и вместе с ним красный.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Светодиоды (зеленого цвета – 3 шт., красного цвета – 1 шт.);
- Кнопка тактовая (3 шт.);
- Резисторы 220 Ом (4 шт.);
- Соединительные провода («папа-папа», 12 шт.);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

Сначала соберем схему, потом в среде Arduino IDE напишем программный код. И то, и другое сделаем с подробными комментариями (общий вид схемы представлен на рисунке 19).

1. **Подключение светодиодов**. В первой работе, в которой будет использована макетная плата и светодиоды, подробно разберем, как подключить светодиоды. В дальнейшем вы можете возвращаться к этому описанию.

Светодиод – это полупроводниковый элемент, который при прохождении через него электрического тока, излучает видимый свет. Нет цели изучения внутреннего устройства светодиода, но для правильного подключения светодиода в схему, нужно знать следующее: у светодиода есть положительный вывод (анод) и отрицательный вывод (катод). Различить анод и катод у светодиода можно по разной длине выводов: длинная «ножка» - это «+» (анод), а короткая «ножка» - это «-» (катод) (Рисунок 20).

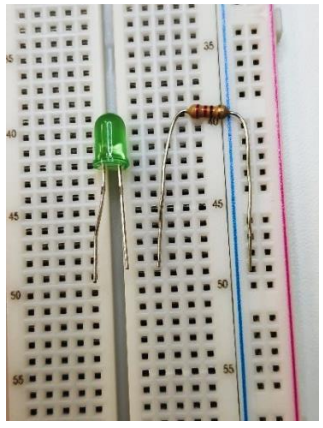


Рисунок 20

Светодиод чувствителен к величине силы тока, протекающего через него; если сила тока будет слишком велика, светодиод «сгорит». Поэтому при подключении светодиода используется токоограничивающий резистор. Этим объясняется наличие в схеме резисторов. В нашем случае будем использовать резисторы сопротивлением 220 Ом.

Схему подключения светодиода с токоограничивающим резистором смотрите на Рисунке 19 (б).

2. **Подключение кнопки** (Рисунок 21). Кнопка имеет четыре контакта, замыкание контактов при нажатии кнопки происходит между левыми и правыми контактами. Схему подключения кнопки см. на Рисунке 19 (б).

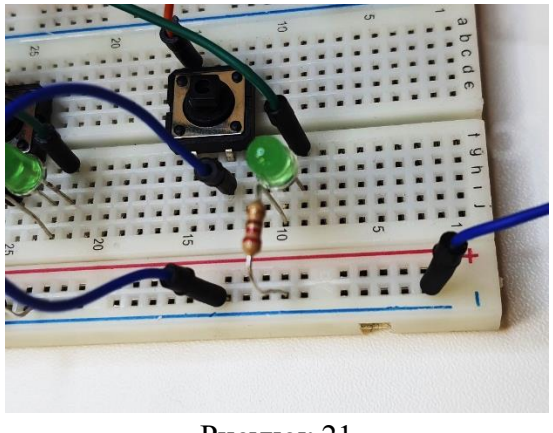


Рисунок 21

3. Выполните сборку схемы целиком, как показано на Рисунке 19.

4. Откройте среду Arduino IDE и наберите программный код, представленный в листинге 1. После листинга даны пояснения к коду.

Листинг 1

```
#define ET1 4  //задаем текстовое имя для контакта 4
#define ET2 5  //задаем текстовое имя для контакта 5
#define ET3 6  //задаем текстовое имя для контакта 6
#define DV 7   //задаем текстовое имя для контакта 7
#define KN1 8  //задаем текстовое имя для контакта 8
#define KN2 9  //задаем текстовое имя для контакта 9
#define KN3 10 //задаем текстовое имя для контакта 10

void setup() {
   //режим подключения кнопки на контакте 8
   pinMode(KN1, INPUT_PULLUP);

   //режим подключения кнопки на контакте 9
   pinMode(KN2, INPUT_PULLUP);

   //режим подключения кнопки на контакте 10
   pinMode(KN3, INPUT_PULLUP);

   //включение режима вывода для контакта 4
   pinMode(ET1, OUTPUT);

   //включение режима вывода для контакта 5
   pinMode(ET2, OUTPUT);

   //включение режима вывода для контакта 6
   pinMode(ET3, OUTPUT);

   //включение режима вывода для контакта 7
   pinMode(DV, OUTPUT);
}

void loop(){
     //чтение состояния кнопки этажа 1
     //если кнопка нажата
     if (!(digitalRead(KN1)))
     {
          //светодиод 1-го этажа горит
```

```
          digitalWrite(ET1, HIGH);
          delay(100);

          digitalWrite(DV, HIGH);
          delay(100);

          digitalWrite(DV, LOW);
          delay(100);

          digitalWrite(DV, HIGH);
          delay(100);

          digitalWrite(DV, LOW);
          delay(100);

          //мигание светодиодом движения лифта
          //светодиод 1-го этажа гаснет
          digitalWrite(ET1, LOW);
          delay(100);
     }
     //чтение состояния кнопки этажа 2
     //если кнопка нажата
     if (!(digitalRead(KN2)))
     {
          //светодиод 2-го этажа горит
          digitalWrite(ET2, HIGH);
          delay(100);

          digitalWrite(DV, HIGH);
          delay(100);

          digitalWrite(DV, LOW);
          delay(100);

          digitalWrite(DV, HIGH);
          delay(100);

          digitalWrite(DV, LOW);
```

```
          delay(100);

          //мигание светодиодом движения лифта
          //светодиод 2-го этажа гаснет
          digitalWrite(ET2, LOW);
          delay(100);
     }
     //чтение состояния кнопки этажа 3
     //если кнопка нажата
     if (!(digitalRead(KN3)))
     {
          //светодиод 3-го этажа горит
          digitalWrite(ET3, HIGH);
          delay(100);

          digitalWrite(DV, HIGH);
          delay(100);

          digitalWrite(DV, LOW);
          delay(100);

          digitalWrite(DV, HIGH);
          delay(100);

          digitalWrite(DV, LOW);
          delay(100);

          //мигание светодиодом движения лифта
          //светодиод 3-го этажа гаснет
          digitalWrite(ET3, LOW);
          delay(100);
     }
}
```

Пояснения к коду

Рассмотрим код листинга 1 (некоторые строки удалены для уменьшения объёма кода и повышения наглядности (так как функциональные команды повторяются).

Листинг 2

```
#define ET1 4
…
#define KN1 8

void setup() {
  pinMode(KN1, INPUT_PULLUP);
  …
  pinMode(ET1, OUTPUT);
}

void loop()
{
     if (!(digitalRead(KN1)))
     {
          digitalWrite(ET1, HIGH);
          delay(100);
          digitalWrite(ET1, LOW);
          delay(100);
     …
     }
}
```

Рассмотрим строку 1 в листинге 2: `#define ET1 4`. Команда `#define` для цифры 4 (к цифровом контакту 4 подключен светодиод) создает текстовое имя (для удобства чтения кода). То же самое сделано для цифры 8 (к цифровому контакту 8 подключена кнопка).

В строке 5 создается обязательная функция ***void setup()*** для первичной настройки оборудования: в строке 6 команда `pinMode(KN1, INPUT_PULLUP);` настраивает цифровой контакт 8 (текстовое имя `KN1`) для подключения кнопки с помощью параметра `INPUT_PULLUP`. В результате, когда кнопка не нажата проверка её состояния будет возвращать `TRUE`, а при нажатой кнопке проверка вернет `FALSE`.

В строке 8 pinMode(ET1, OUTPUT); при помощи команды pinMode настраиваем цифровой контакт 4 (текстовое имя ET1) в режим вывода уровней напряжения, которые обозначаются HIGH (+5 В) и LOW (0 В) при помощи параметра OUTPUT.

На этом начальная настройка оборудования завершена. Переходим к рассмотрению основного кода программы.

В строке 12 создается функция void loop() для выполнения основного кода программы. Функция работает как цикл, бесконечно повторяя код своего тела.

В строке 14 в операторе условия при помощи команды digitalRead(KN1)(читает состояние цифрового контакта, подключенного к кнопке; возвращает TRUE, если кнопка НЕ нажата, и FALSE при нажатии кнопки) выполняется проверка логического значения с кнопки KN1.

Блок, представленный строками 16 – 19, выполняет операцию мигания светодиодом: команда digitalWrite выполняет вывод на цифровой контакт ET1 значений HIGH или LOW (при HIGH светодиод горит, а при LOW – гаснет). Команда delay формирует временную задержку в миллисекундах, чтобы у нас было время визуально различить режимы «светодиод горит» (сигнал HIGH) – «светодиод не горит» (сигнал LOW).

5. **Загрузка кода в плату**. Теперь, когда у вас открыта среда ***Arduino IDE*** с набранным кодом из листинга 1, подключите плату Arduino к компьютеру с помощью кабеля USB. Используйте кнопку ***Загрузить*** (см. описание области 2 на Рисунке 13). После загрузки кода в плату проверьте работу схемы нажатием кнопок.

Примечание. *Из-за особенностей работы платы Arduino, при загрузке кода в плату цифровые контакты 0 и 1 должны оставаться не занятыми. После загрузки они доступны для работы.*

Задания для самостоятельного выполнения

1. Измените программный код так, чтобы:

а) время мигания светодиода движения составило 2 секунды;

б) время мигания светодиода движения по длительности соответствовало этажу назначения (1-й этаж – 1 секунда, 2-й этаж – 2 секунды, 3-й этаж – 3 секунды);

2. Модифицируйте схему, добавив светодиод и кнопку для 4-го этажа.

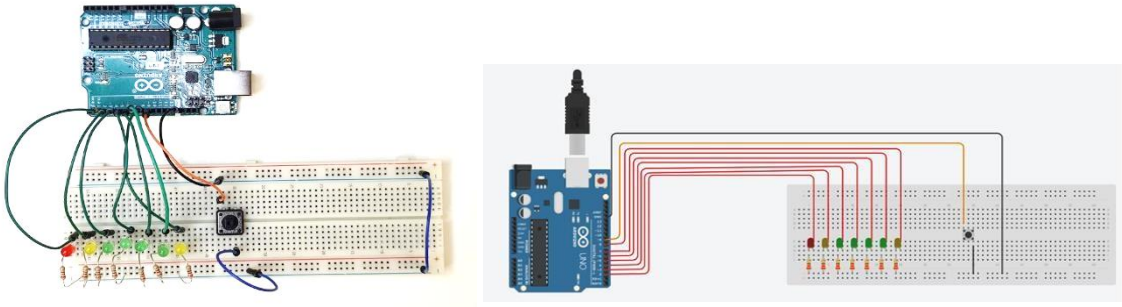
3. Измените программный код так, чтобы по прибытии на этаж зеленый светодиод соответствующего этажа несколько раз приветственно мигнул.

Лабораторная работа № 2. Панель индикации роутера

В лабораторной работе № 1 вы познакомились с платой Arduino UNO и сборкой схем на макетной плате.

В этой лабораторной работе будут использованы те же элементы, что и раньше: светодиоды, кнопки, резисторы, соединительные провода. Но теперь уже не будет необходимости в таком подробном рассмотрении всех деталей подключения светодиодов или работы в среде Arduino IDE, так как вам это уже известно, а если вы что-то забыли, необходимо вернуться к описанию лабораторной работы № 2 и прочитать его еще раз.

Задача 1. Создать модель, имитирующую панель индикации Wi-Fi роутера. На макетной плате расположены семь светодиодов. Красный светодиод эмулирует индикатор питания. Следующий за ним желтый светодиод эмулирует индикатор мощности Wi-Fi сигнала. Четыре зеленых светодиода отвечают за индикацию подключения проводных клиентов (персональных компьютеров). Крайний правый желтый светодиод загорается на непродолжительное время при нажатии кнопки, эмулируя работу функции WPS (Wi-Fi Protected Setup – стандарт и протокол быстрого и безопасного сопряжения (соединения) двух Wi-Fi устройств между собой) (Рисунок 1).



а б

Рисунок 1 Общий вид схемы
а) физическое исполнение; б) рисунок схемы

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;

- Макетная плата Breadboard (800 точек);
- Светодиоды (зеленого цвета – 4 шт., желтого цвета – 2 шт., красного цвета – 1 шт.);
- Кнопка тактовая (1 шт.);
- Резисторы 220 Ом (7 шт.);
- Соединительные провода («папа-папа», 11 шт.);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему как показано на Рисунке 2. На рисунках 3 – 4 отдельные области подключения показаны крупнее.

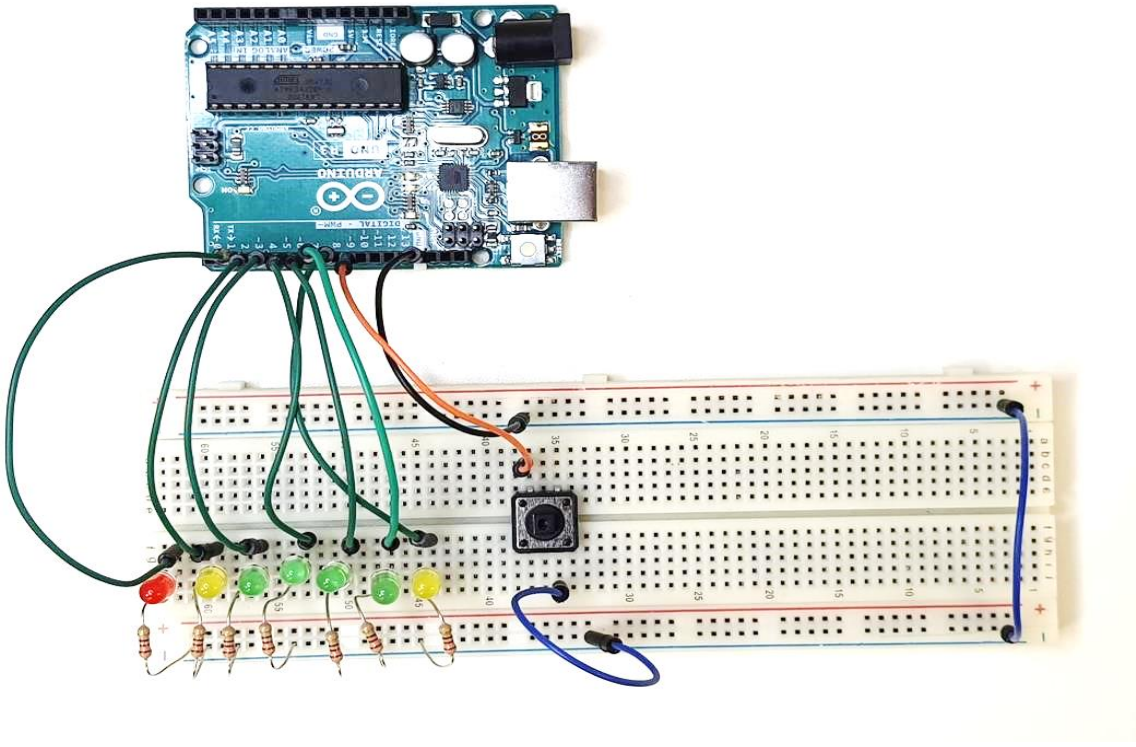


Рисунок 2 Общий вид устройства

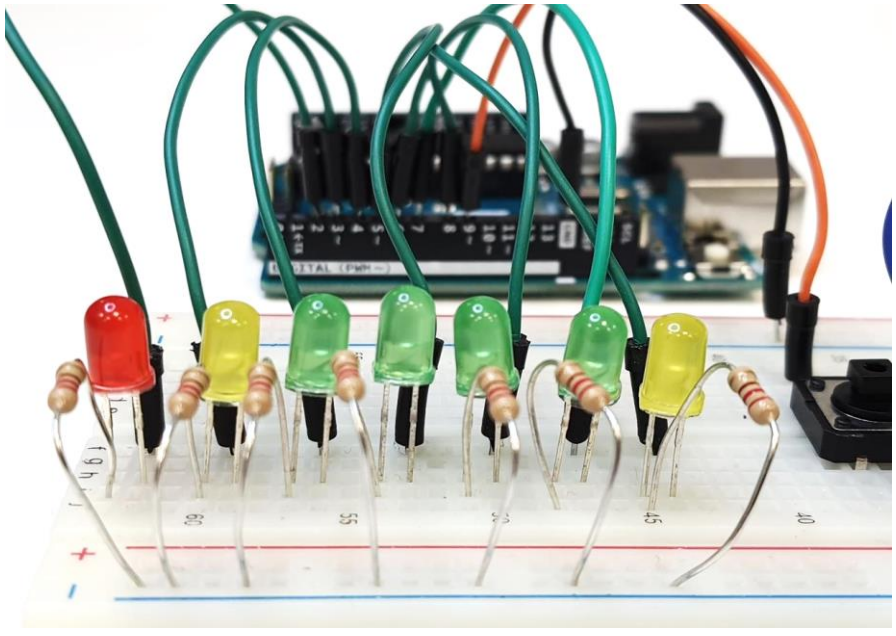


Рисунок 3 Подключение светодиодов и резисторов

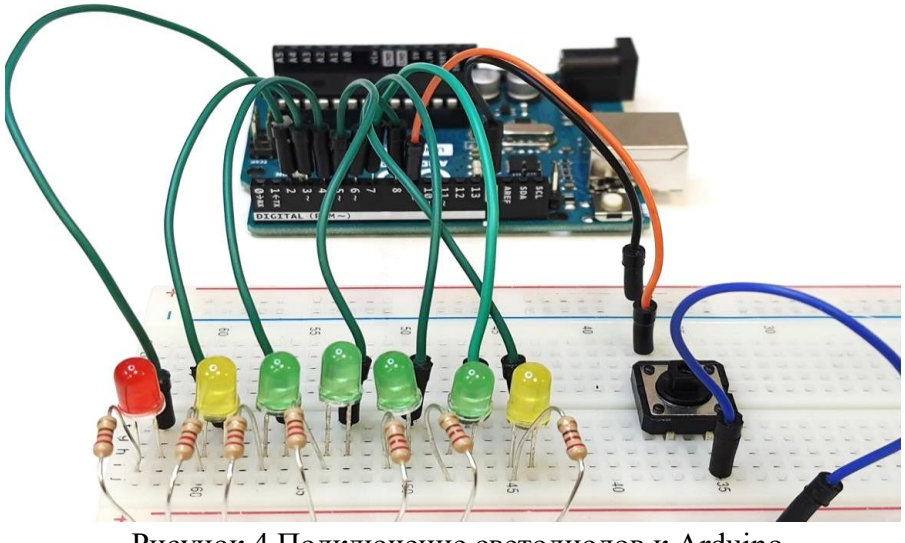


Рисунок 4 Подключение светодиодов к Arduino

2. Откройте среду Arduino IDE и наберите программный код, представленный в листинге 1. После листинга даны краткие пояснения к коду, а в самом коде есть комментарии.

Листинг 1

```
#define VD1 2  // создание имени VD1 для цифрового вывода pin 2
#define VD2 3  // создание имени VD2 для цифрового вывода pin 3
#define VD3 4  // создание имени VD3 для цифрового вывода pin 4
#define VD4 5  // создание имени VD4 для цифрового вывода pin 5
#define VD5 6  // создание имени VD5 для цифрового вывода pin 6
#define VD6 7  // создание имени VD6 для цифрового вывода pin 7
#define WPS 8  // создание имени WPS для цифрового вывода pin 8
#define KN 9   // создание имени KN для цифрового вывода pin 9

/*создание имен для цифровых выводов при помощи директивы #define для
облегчения читаемости кода, так как вместо цифр будут использоваться
смысловые имена*/

/*объявление обязательной функции setup(); функция используется один раз
при запуске микроконтроллера для настройки режимов работы оборудования*/

void setup()
{

  /*использование цикла для настройки цифровых выводов VD1 – VD6 и WPS в
  режим выходов*/
  for (int i=2; i<9; i++)
  {

    /*команда настройки режима работы цифрового вывода*/
    pinMode(i, OUTPUT);
  }

  /*настройка режима работы цифрового вывода в режим входа с подключением
  внутреннего нагрузочного резистора для обработки нажатий кнопки*/
  pinMode(KN, INPUT_PULLUP);
}

/*объявление обязательной функции loop(); в функции пишется основной
исполняемый код программы; тело функции повторяется бесконечно*/
void loop()
{
   /*команда вывода цифрового сигнала на цифровой вывод платы*/
   digitalWrite(VD1, HIGH);

   /*создание статической локальной переменной без потери ее значений в
   итерациях; используется для формирования уровня ШИМ у индикатора уровня
   мощности сигнала Wi-Fi*/
```

```
static int time1 = 0;

//вывод уровня ШИМ-сигнала
analogWrite(VD2, time1);

/*создание статической локальной переменной q – для выбора поочередной
одного из зеленых светодиодов, а c – для выбора режима светодиода
(«горит – не горит»)*/

static int c=0, q=4;
//поочередный выбор одного из зеленых светодиодов
switch (q)
{
  case 4:
     c = random(2);
     digitalWrite(VD3, c);
     delay(50);
   break;
   case 5:
     c = random(2);
     digitalWrite(VD4, c);
     delay(50);
   break;
   case 6:
     c = random(2);
     digitalWrite(VD5, c);
     delay(50);
   break;
   case 7:
     c = random(2);
     digitalWrite(VD6, c);
     delay(50);
   break;
   }

/*использование функции random() для случайного выбора одного из
режимов работы зеленых светодиодов*/

//подсчет времени свечения сигнала WPS
static int time2=0;
if (!(digitalRead(KN)))
{
    digitalWrite(WPS, HIGH);
```

```
   }

   /*определение нажата ли кнопка; если кнопка нажата, светодиод сигнала
   WPS загорается */

   q++;
   if (q==8)
   {
       q=4;
   }

   //реализация поочередного перебора зеленых светодиодов
   time1+=1;
   if (time1>254)
   {
       time1=0;
   }

   //реализация изменения уровня сигнала ШИМ
   time2+=1;
   if(time2==400)
   {
       time2=0;
       digitalWrite(WPS, LOW);
   }
   //реализация продолжительности включения светодиода индикации WPS
}
```

Пояснения к коду

Команда `#define` вам знакома; как вы помните, она создает «текстовые имена», в данном случае для цифровых выводов со второго по девятый (строки 1 – 9).

В лабораторной работе № 2 рассмотрены структуры скетча для Arduino (в нем обязательно должны быть функции `void setup ()` и `void loop()`.

Рассмотрим функцию `void setup()` в данной лабораторной работе:

- строки 16 – 26: в этих строках в цикле FOR с помощью команды `pinMode` производится настройка выводов со второго по девятый в режим выхода;
- строка 27: команда `pinMode(KN, INPUT_PULLUP);` настраивает цифровой контакт 9 (текстовое имя KN) для подключения кнопки с помощью параметра INPUT_PULLUP. В результате, когда кнопка не нажата проверка её состояния будет возвращать TRUE, а при нажатой кнопке проверка вернет FALSE.

Рассмотрим функцию `void loop()` в данной лабораторной работе: как вы помните, функция работает как цикл, бесконечно повторяя код своего тела.

В строке 40 создается переменная `time1`, значение которой будет последовательно изменяться (увеличиваться от 0 до 255 и по достижении максимально возможного значения (254) будет сбрасываться в 0, строки 75 – 79), что будет означать изменение уровня сигнала ШИМ (широтно-импульсной модуляции – формирование уровней напряжения в вольтах при помощи наборов цифровых сигналов), а уровень сигнала ШИМ (255 – 5В, 127 □ 2,5В, 0 – 0В) влияет на яркость свечения светодиода: чем больше значение переменной `time1`, тем ярче горит светодиод, и наоборот.

Значение переменной `time1` в качестве параметра передается функции `analogWrite(VD2, time1)`. Функция `analogWrite()` выдает аналоговую величину, в данном случае величину ШИМ.

В строках 49 – 55 наибольший интерес представляет функция `random(2)`. Эта функция генерирует случайное число в диапазоне от 0 до 2 (2 не входит в диапазон генерируемых чисел). Таким образом, в качестве случайных чисел будут генерироваться 0 или 1, что можно использовать как HIGH (1) и LOW (0) для функций `digitalWrite()`.

Дополнительно нужно отметить, что в операторе switch(q) переменная q используется для обращения к цифровым выходам не по текстовым именам, а при помощи номеров цифровых выходов. Это необходимо для организации поочередного обращения к цифровым контактам.

В строках 68 – 72 последовательно увеличивается значение переменной q, а при достижении ею значения 8, её значение возвращается к 4. Тем самым происходит обращение к нужным контактам через оператор `switch()`.

Задания для самостоятельного выполнения

1. В операторе `switch()` в метках case замените цифры выходных контактов на их текстовые имена (строки 4 – 7). Проверьте работу программы.
2. Измените код в строках 75 – 79 так, чтобы длительность нарастания сигнала ШИМ была увеличена.
3. Подключите дополнительный светодиод на контакт 10 и настройте вывод уровня сигнала ШИМ при помощи функции `random()`. Не забудьте добавить небольшую задержку `(delay())` после вывода уровня, чтобы визуально различить изменение яркости светодиода.

Лабораторная работа № 3. Знакомство с потенциометром

В этой лабораторной работе продолжим знакомство с резисторами, а именно с потенциометром.

Потенциометр – это переменный резистор, уровень сопротивления которого определяется поворотом ручки (Рисунок 1).

Рисунок 1

Задача 1. Изучить изменение сопротивления потенциометра в зависимости от положения поворота управляющей ручки. От изменения сопротивления потенциометра зависит яркость свечения светодиода.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Светодиод (любого цвета – 1 шт.);
- Резистор 220 Ом;
- Потенциометр (1 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 2 – 5.

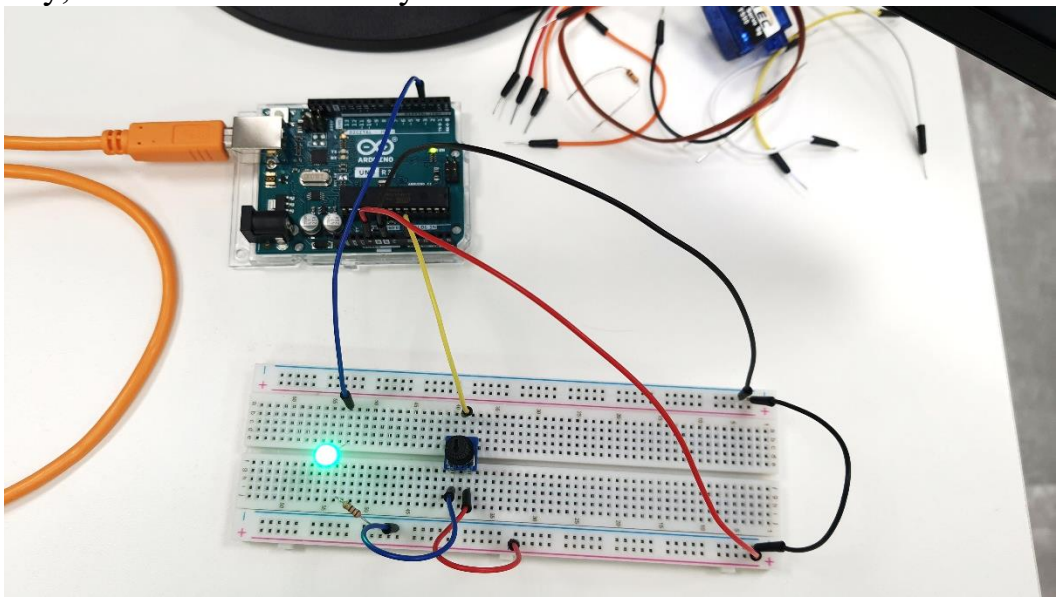


Рисунок 2

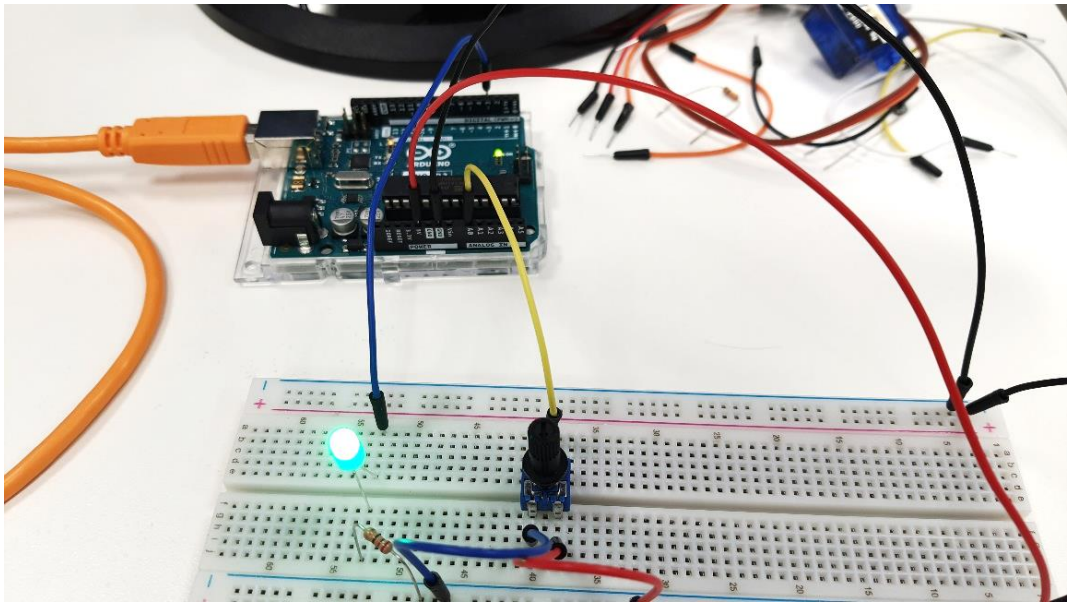


Рисунок 3

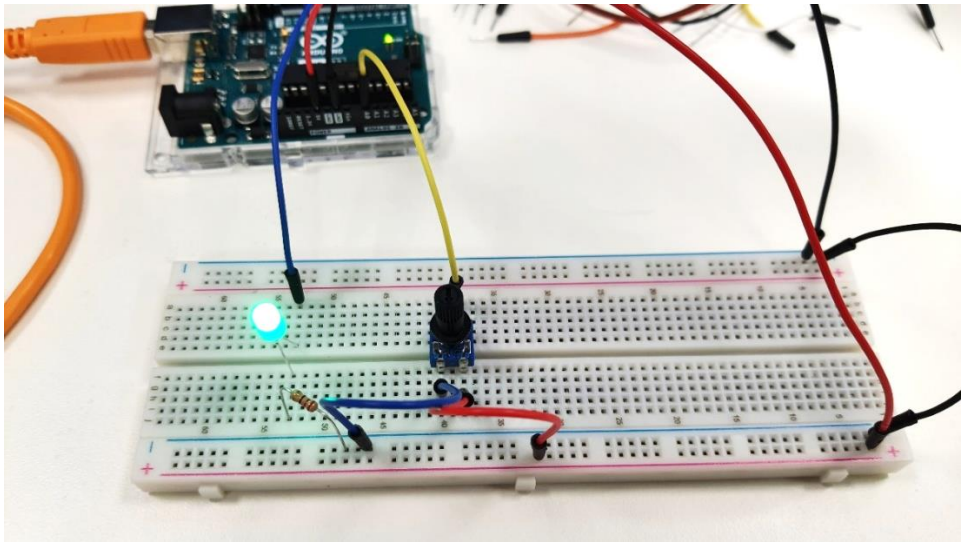


Рисунок 4

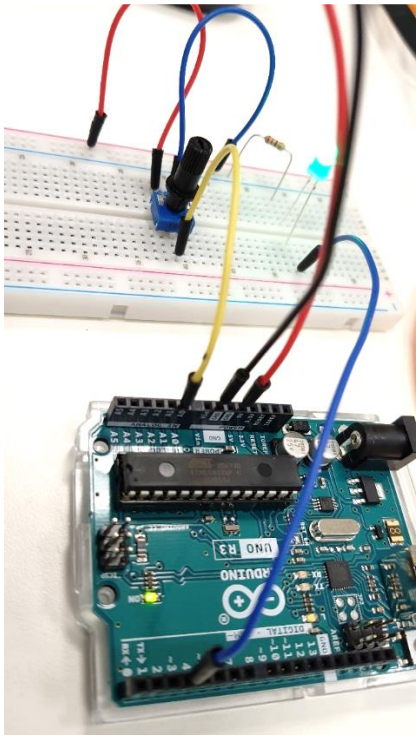


Рисунок 5

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.

4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Измените диапазоны преобразований функции `map`. Опишите полученные результаты.
2. Добавьте между контактом A0 и выходом потенциометра резистор 10 кОм. Опишите, что вы наблюдаете.

Листинг 1

```
#define VD 3    //текстовое имя для цифрового вывода
#define POT A0  //текстовое имя для аналогового входа

void setup() {
    //настройка цифрового выхода в режим вывода
    pinMode(VD, OUTPUT);
}

void loop() {

    //определяем уровень поворота ручки потенциометра
    int p = analogRead(POT);

    //преобразуем прочитанные значения в форму, доступную для вывода
    int pl = map(p, 0, 1023, 0, 255);

    //вывод уровня освещенности на светодиод
    analogWrite(VD, pl);
}
```

Лабораторная работа № 4. Знакомство с термистором (терморезистором)

Термистор – это резистор, сопротивление которого зависит от температуры (Рисунок 1).

Рисунок 1

Задача 1. Изучить изменение сопротивления термистора в зависимости от температуры окружающей среды. От изменения сопротивления термистора зависит яркость свечения светодиода.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Светодиод (любого цвета – 1 шт.);
- Резистор 220 Ом – 1 шт.;
- Резистор 10 кОм – 1 шт.;
- Термистор (1 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 2 – 4.

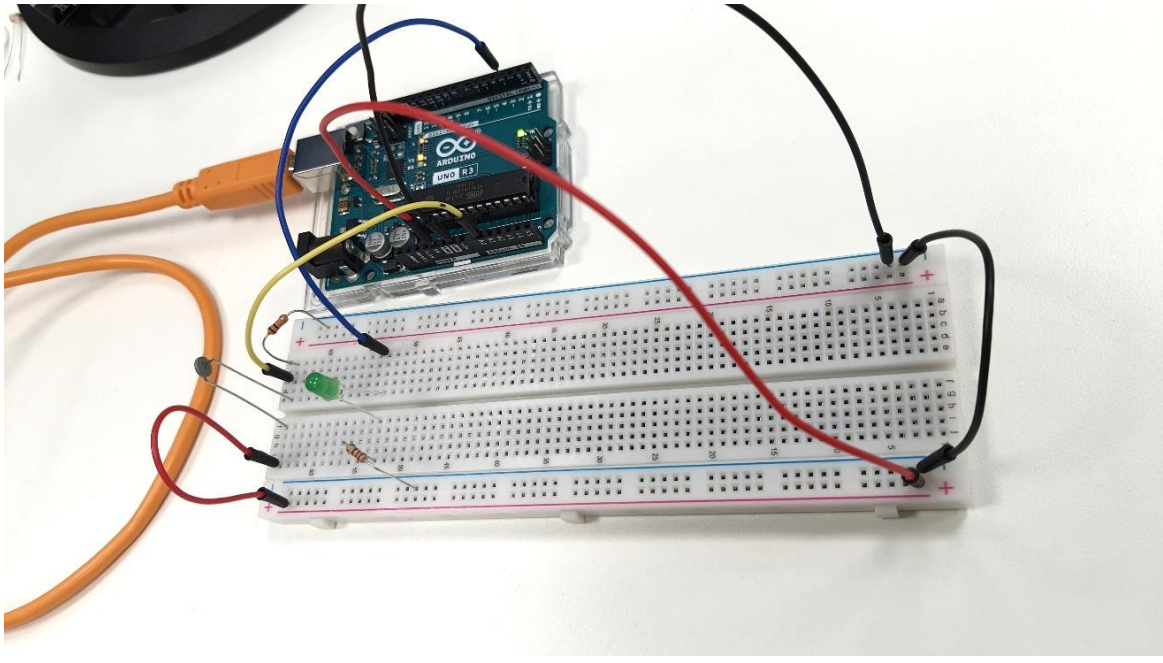


Рисунок 2

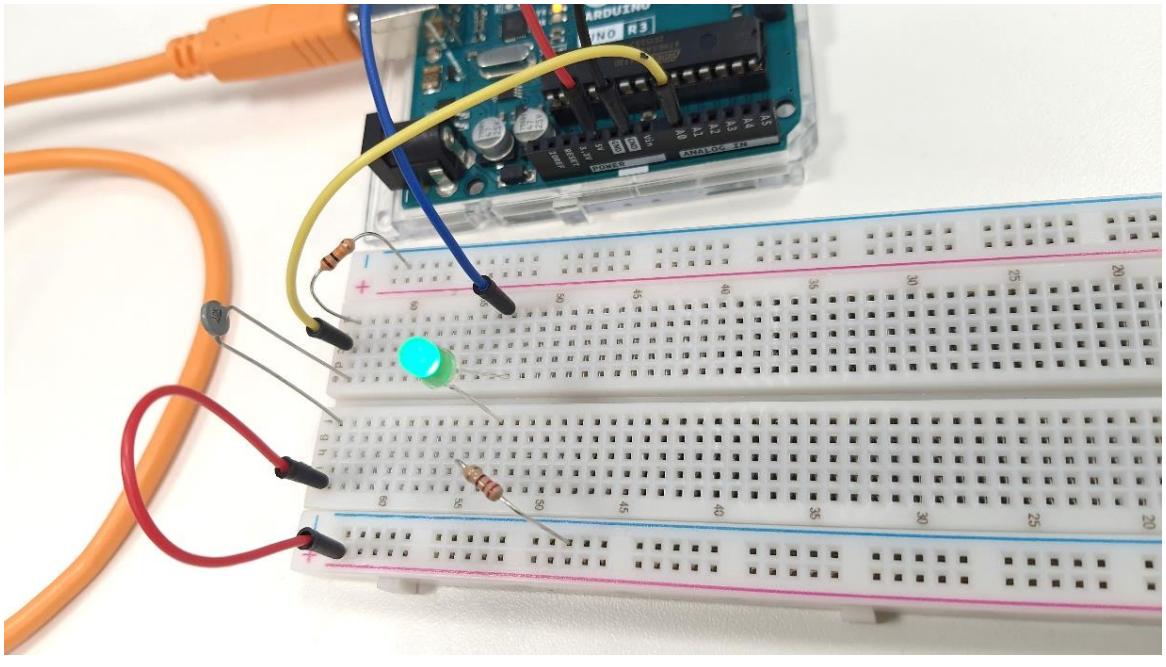


Рисунок 3

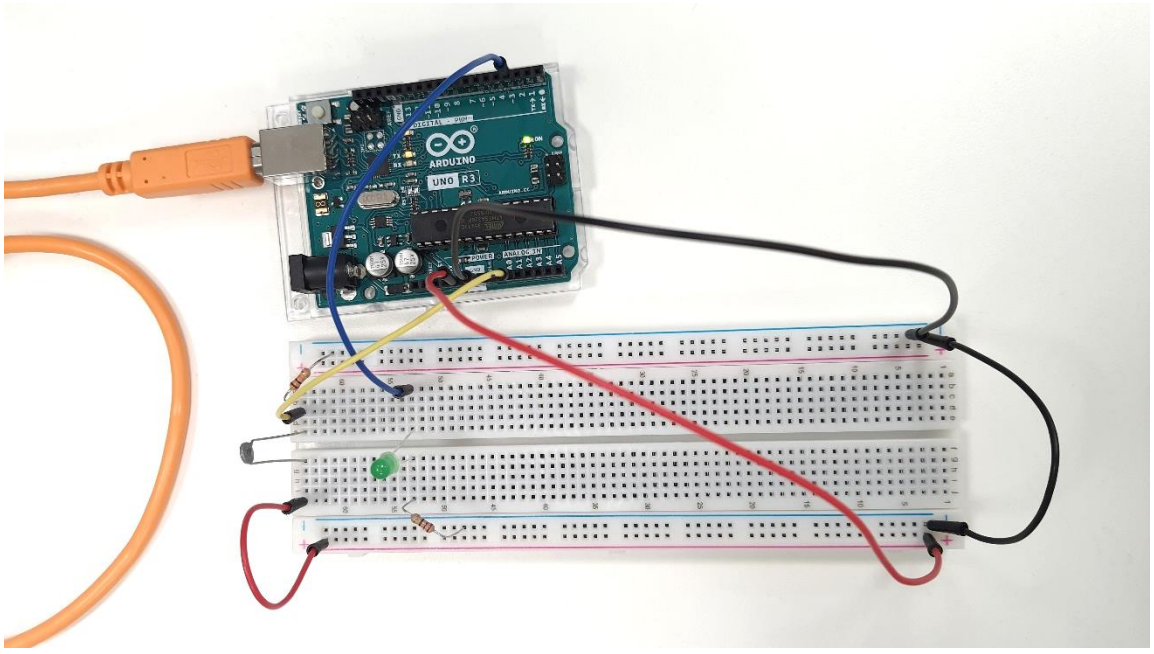


Рисунок 4

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте второй термистор и второй светодиод. Попробуйте подать разные значения температуры на каждый из них.

2. Попробуйте подобрать значение первого диапазона функции `map` (из которого происходит преобразование) для наиболее резкого изменения яркости светодиода в рамках вашей окружающей среды.

Листинг 1

```
#define VD 3     //создание текстового имени для цифрового вывода
#define TEMP A0  //создание текстового имени для аналогового входа

void setup() {
     //настройка цифрового вывода в режим выхода
     pinMode(VD, OUTPUT);
}

void loop() {
     //определение уровня температуры окружающей среды
     int p = analogRead(SVR);

     //преобразование полученных значений температуры в формат, доступный для
вывода
     int pl = map(p, 450, 600, 0, 255);

     //вывод измеренных значений
     analogWrite(VD, pl);
}
```

Лабораторная работа № 5. Знакомство с RGB-светодиодом

RGB-светодиод в одном корпусе содержит светодиоды красного (red), зеленого (green) и синего (blue) цветов (Рисунок 1).

Рисунок 1

У RGB-светодиода четыре вывода: три вывода соответствуют светодиодам указанных цветов, а четвертый вывод – это катод (самый длинный вывод).

Задача 1. Изучить работу RGB-светодиода.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- RGB-светодиод (1 шт.);
- Резистор 220 Ом – 3 шт.;
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 2 – 4.

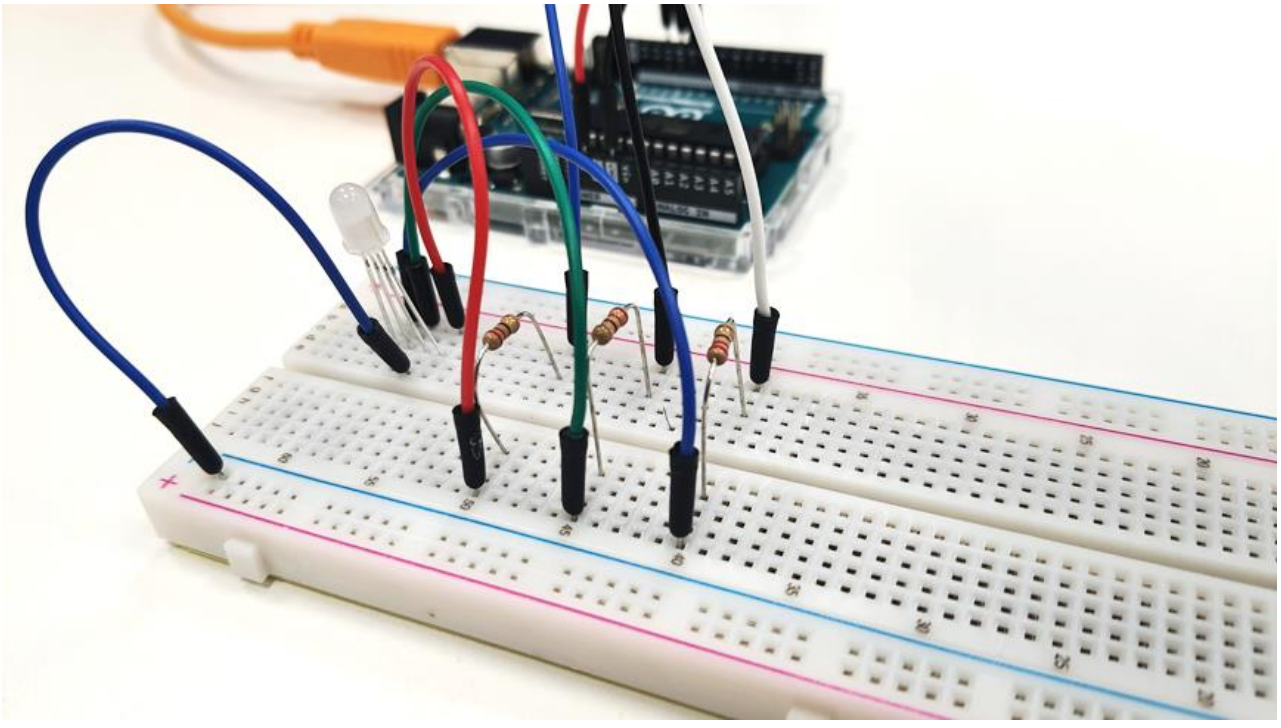


Рисунок 2

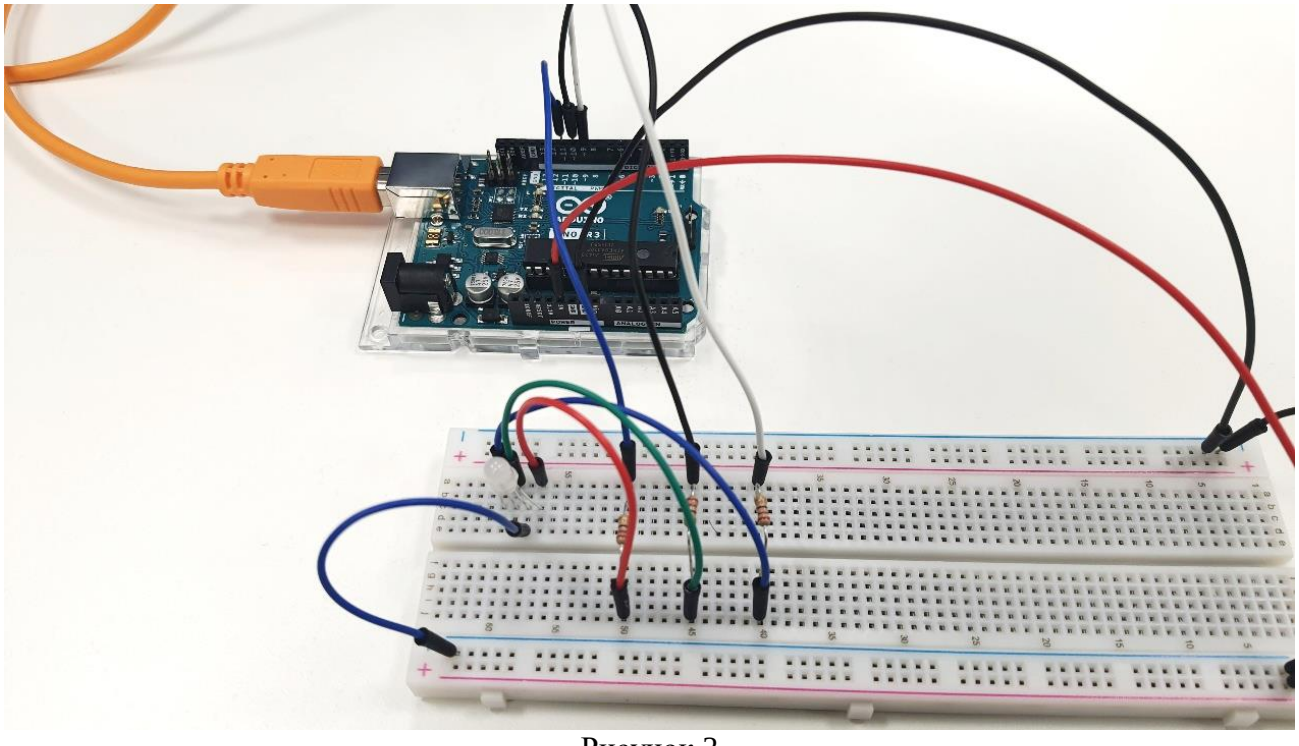


Рисунок 3

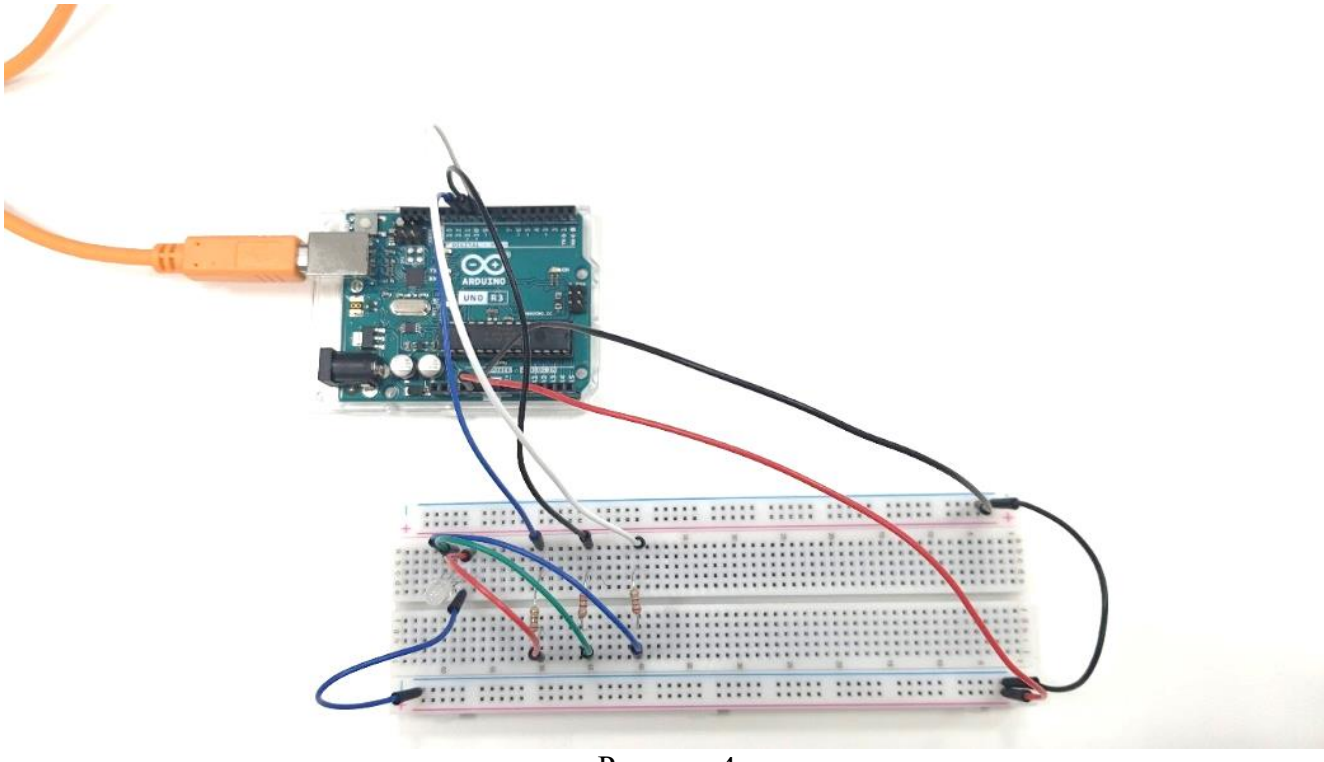


Рисунок 4

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Сформируйте на RGB-светодиоде сначала красный, потом зеленый, потом синий цвет.
2. Для любого из трех цветов создайте эффект плавного изменения цвета от светлого оттенка до темного.

Листинг 1

```
//создание текстового имени для цифрового контакта для красной составляющей
светодиода
#define Red_VD 9

//создание текстового имени для цифрового контакта для зеленой составляющей
светодиода
#define Green_VD 10

//создание текстового имени для цифрового контакта для синей составляющей
светодиода
#define Blue_VD 11

void setup() {
    //настройка контактов 9, 10, 11 в режим выхода в цикле FOR
    for(int i = 9; i < 12; i++)
    {
       pinMode(i, OUTPUT);
    }
}

void loop() {

    //формирование ШИМ-сигнала для красного цвета
    analogWrite(Red_VD, random(0,255));
    delay(200);

    //формирование ШИМ-сигнала для зеленого цвета
    analogWrite(Green_VD, random(0,255));
    delay(200);

    //формирование ШИМ-сигнала для синего цвета
    analogWrite(Blue_VD, random(0,255));
    delay(200);
}
```

Лабораторная работа № 6. Знакомство с пьезодинамиком

Пьезодинамик при поступлении на него ШИМ-сигнала генерирует звуковую волну (Рисунок 1).

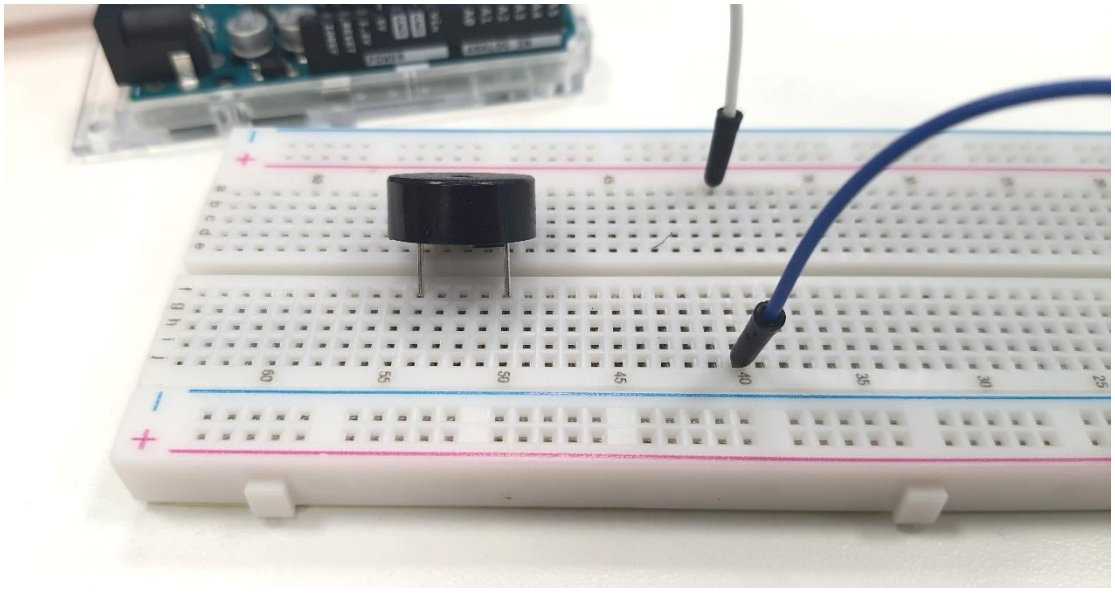


Рисунок 1

Для управления работой пьезодинамика используется функция tone(). Она принимает три параметра:

- номер вывода, к которому подключен пьезодинамик;
- частоту сигнала в Герцах;
- длительность сигнала в миллисекундах.

Задача 1. Изучить работу пьезодинамика.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Пьезодинамик (1 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунке 2 – 3.

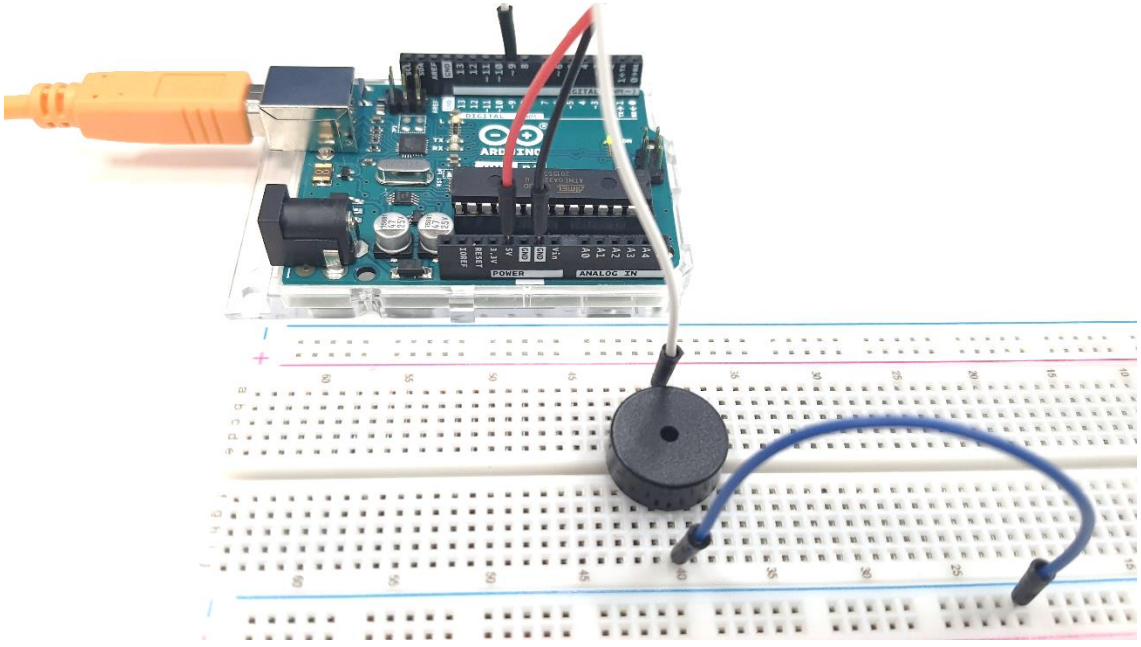


Рисунок 2

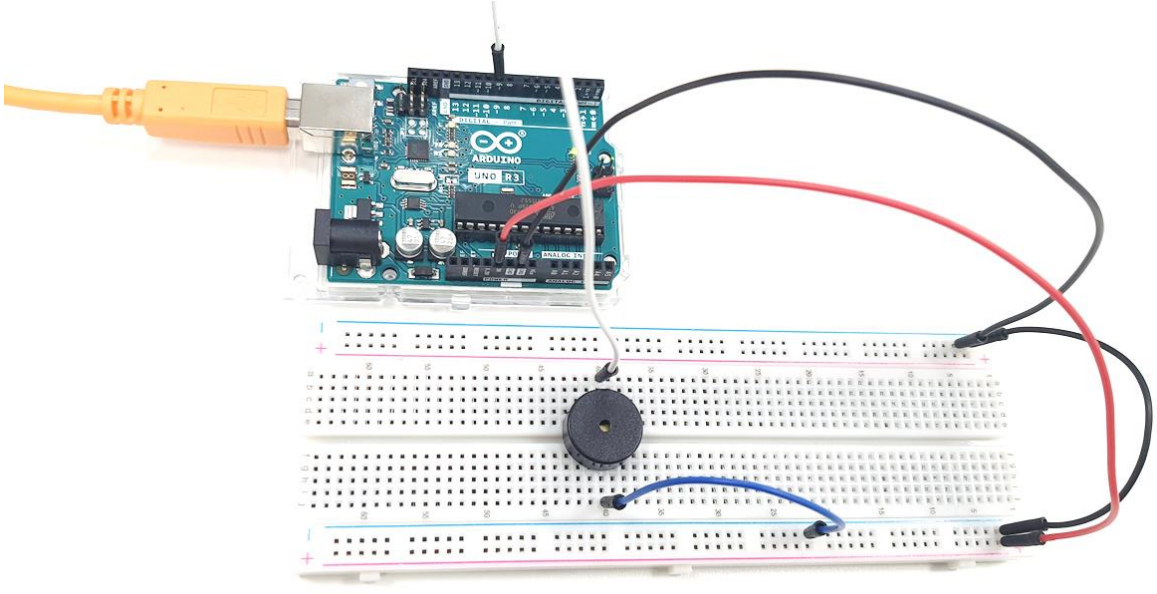


Рисунок 3

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Сформируйте звук, который будет плавно повышаться до определенного предела, некоторое время звучать, а потом также плавно понижаться до исходного значения.
2. Сымитируйте сигнал автомобильной сирены (часто чередующиеся повышения и понижения звука).

Листинг 1

```
//создание текстового имени для цифрового вывода, к которому подключен пьезодинамик
#define sound 9

void setup() {
    //настройка цифрового вывода в режим выхода
    pinMode(sound, OUTPUT);
}

void loop() {
    //вызов функции tone() и передача ей параметров
    tone(sound,3500,1000);
    delay(1000);
    tone(sound,4500,1000);
    delay(1000);
    tone(sound,5500,1000);
    delay(1000);
}
```

Лабораторная работа № 7. Знакомство с биполярным транзистором N-P-N типа

Биполярный транзистор **N-P-N** типа – это полупроводниковое устройство, выполняющее роль электронной кнопки для маломощной нагрузки. **База** (обозначена буквой Б на Рисунке 1) транзистора выполняет условную роль ключа, если подключить базу к 5В, то транзистор замкнет цепь как нажатая кнопка, если к 0В, то разомкнет цепь как не нажатая кнопка.

Главная задача транзистора – позволить цифровым выходам платы Arduino управлять нагрузкой, обладающей значительным токопотреблением, и иногда даже большим, чем на плате уровнем напряжения (Рисунок 1).

Коллектор (К на Рисунке 1) подключается к «-» светодиода, а «+» светодиода через резистор 220 Ом подключается к +5В платы. Эмиттер транзистора (Э на Рисунке 1) подключается к контакту GND платы. Контакт базы подключается к цифровому выводу № 9.

Рисунок 1

Задача 1. Изучить работу биполярного транзистора n-p-n типа.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);

- Биполярный транзистор n-p-n типа (1 шт.);
- Светодиод (1 шт.)
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 2 – 4.

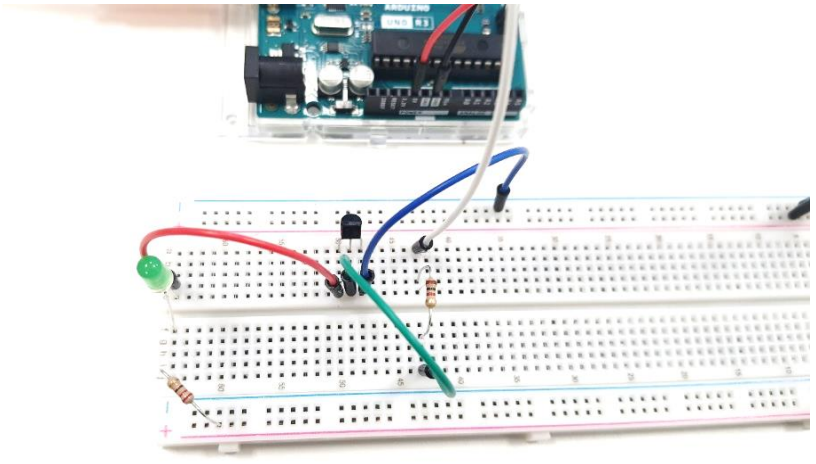


Рисунок 2

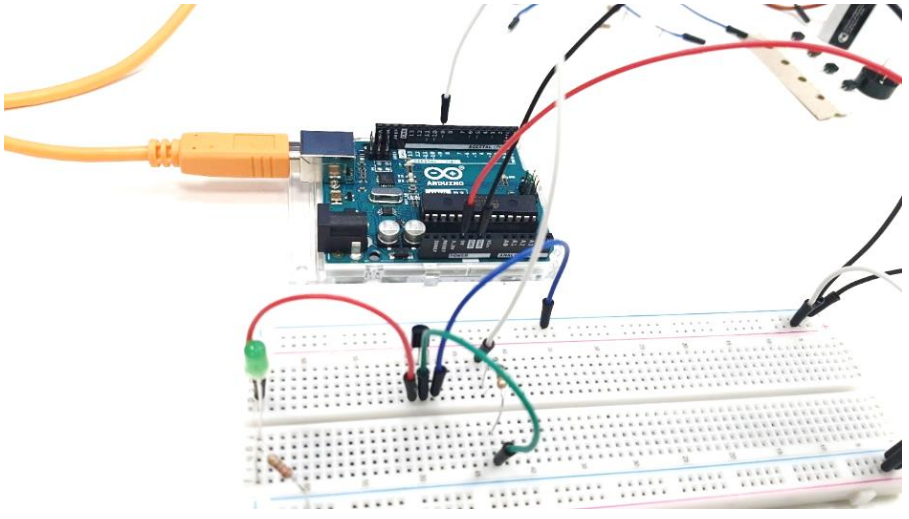


Рисунок 3

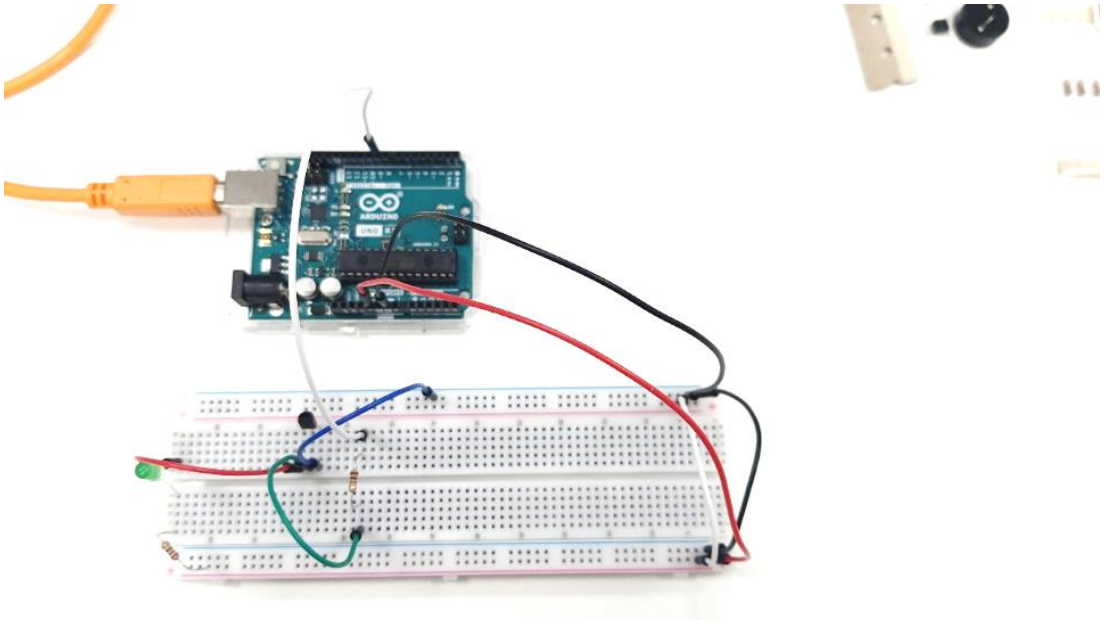


Рисунок 4

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте в схему потенциометр для регулирования уровня сигнала ШИМ на базе транзистора.

2. Подключите «+» светодиода к контакту платы Arduino 3,3 В. Опишите результат.

Листинг 1

```
#define Base_Tr 9

void setup() {
       pinMode(Base_Tr, OUTPUT);

}

void loop() {
       digitalWrite(Base_Tr, LOW);
       delay(2000);
       digitalWrite(Base_Tr, HIGH);
```

```
        delay(2000);

        for(int i=0; i < 255; i++)
        {
          analogWrite(Base_Tr, i);
          delay(50);
        }
}
```

Лабораторная работа № 8. Знакомство полевым MOSFET-транзистором

Полевой MOSFET-транзистор – это полупроводниковое устройство, выполняющее роль электронной кнопки, но для нагрузки с высоким токопотреблением (Рисунок 1 (G (Gate) – затвор; D (Drain) – сток; S (Source) – исток).

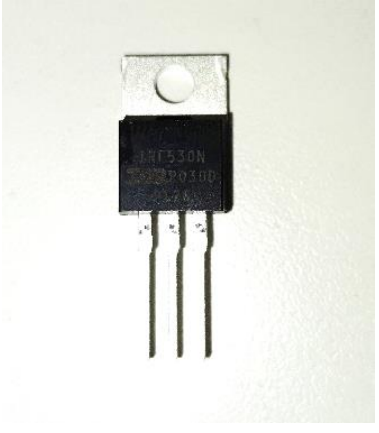


Рисунок 1

Задача 1. Изучить работу MOSFET-транзистора.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- MOSFET-транзистор (1 шт.);
- Светодиод (1 шт.)
- Резистор 220 Ом;
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 2 – 6

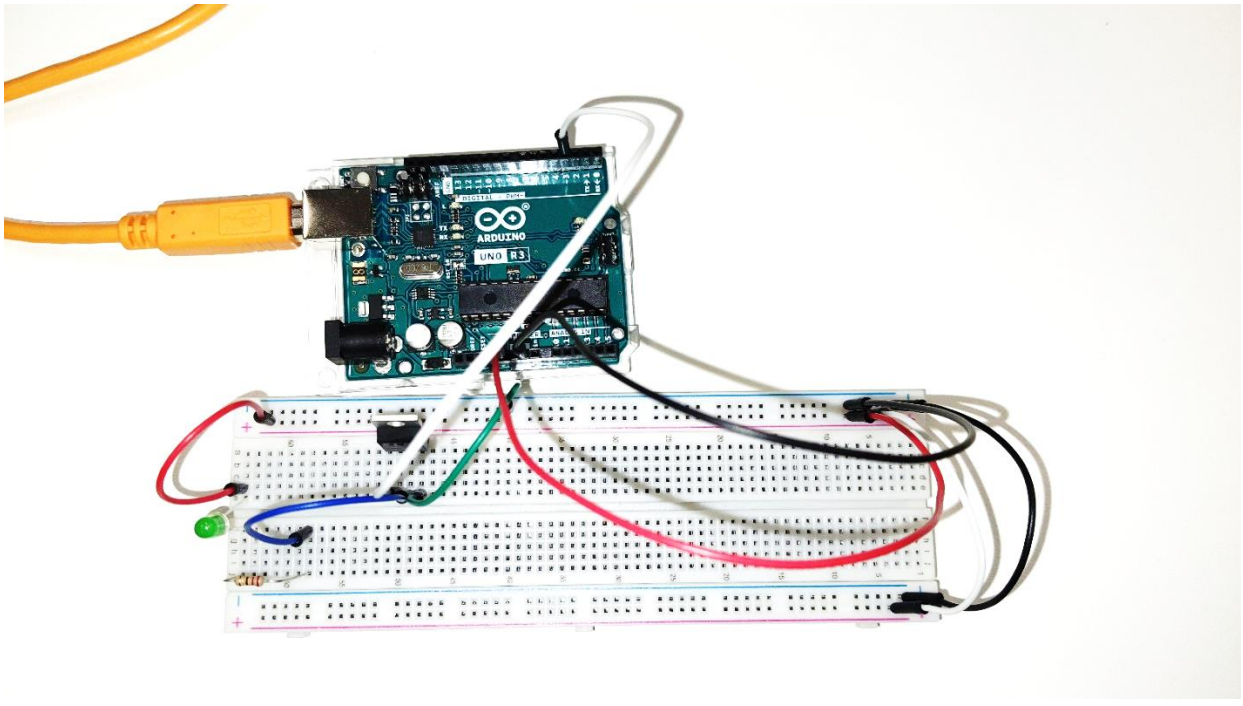


Рисунок 2

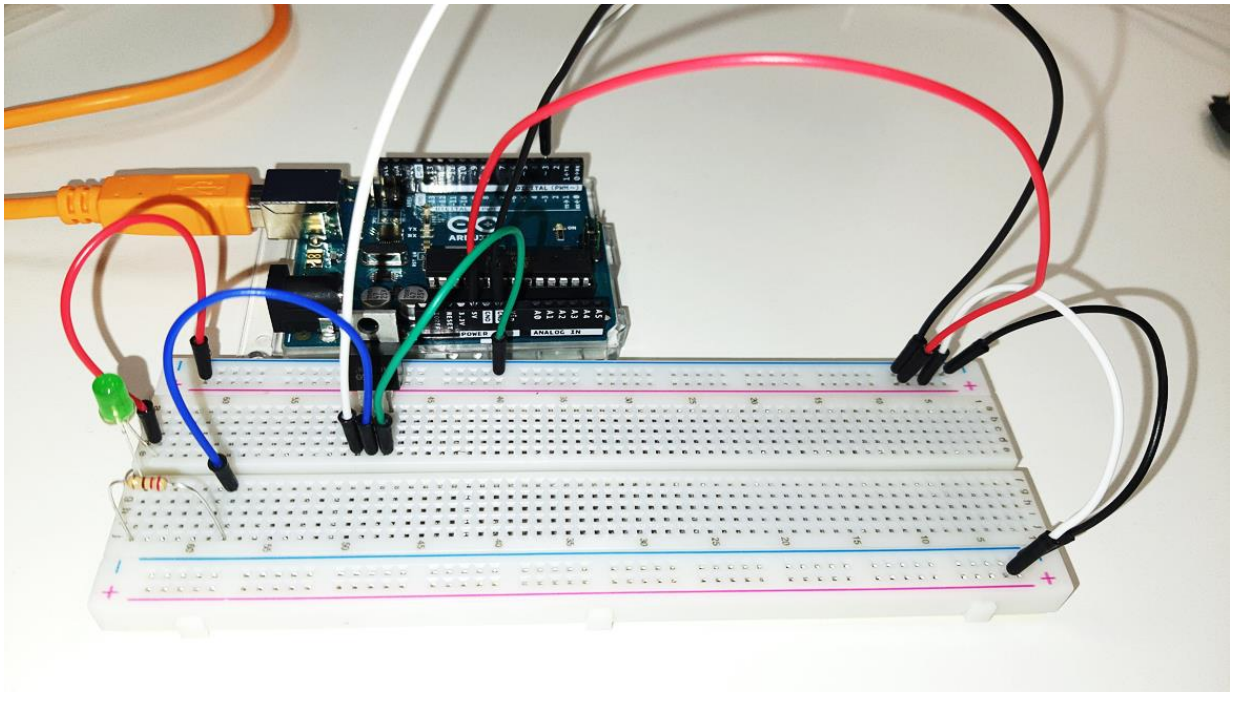


Рисунок 3

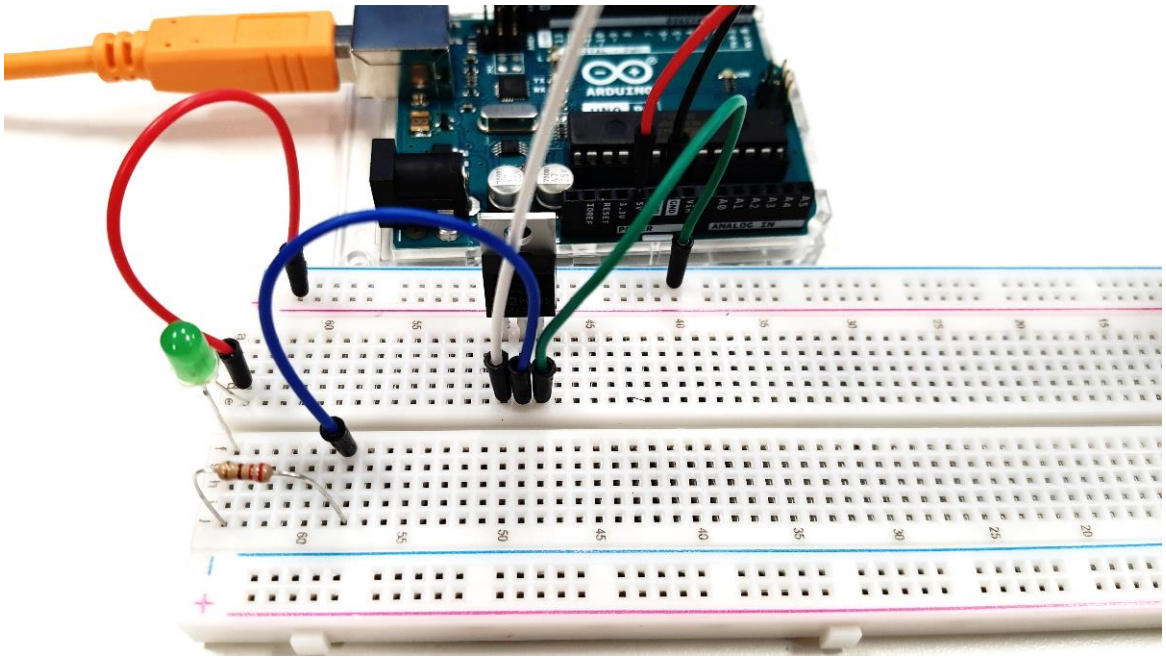


Рисунок 4

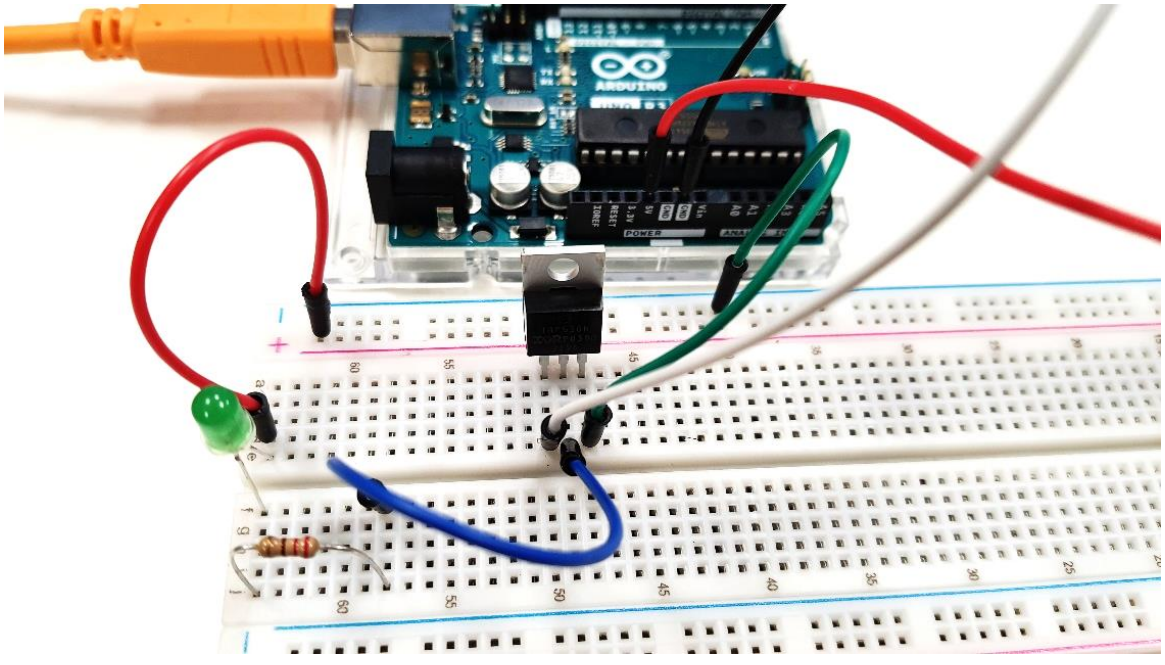


Рисунок 5

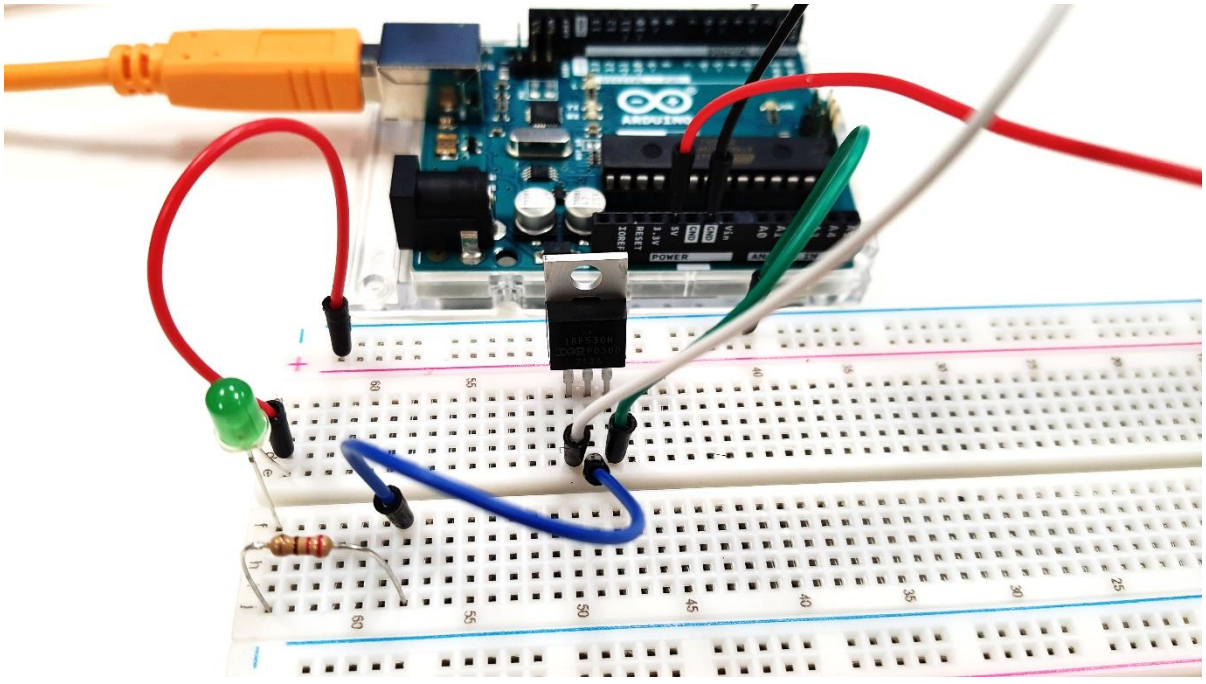


Рисунок 6

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте в схему потенциометр для регулирования уровня сигнала ШИМ.
2. Подключите «+» светодиода к контакту платы Arduino 3,3 В. Опишите результат.

Листинг 1

```
#define GATE 3

void setup() {
	pinMode(GATE, OUTPUT);
}

void loop() {
	//подача управляющего сигнала на затвор транзистора
	analogWrite(GATE,50);
	delay(1000);
	analogWrite(GATE,100);
	delay(1000);
	analogWrite(GATE,150);
	delay(1000);
	analogWrite(GATE,200);
	delay(1000);
	analogWrite(GATE,250);
	delay(1000);
}
```

Лабораторная работа № 9. Знакомство с одноразрядным семисегментным индикатором

Для отображения цифр от 0 до 9 часто используется *семисегментный светодиодный индикатор* (Рисунок 1).

Рисунок 1 Одноразрядный семисегментный индикатор

Управляя работой сегментов, можно составлять изображения цифр и некоторых букв.

Для отображения арабских цифр от 0 до 9 достаточно всего семи сегментов. Почти всегда к семи сегментам добавляют восьмой сегмент – десятичную точку (decimal point – DP) (Рисунок 2).

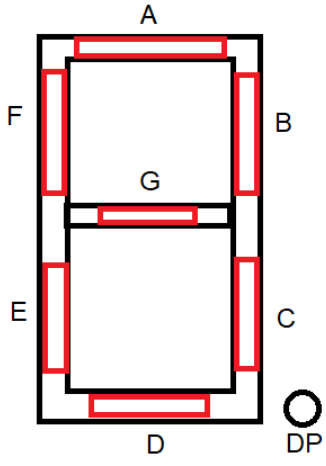


Рисунок 2

Сегменты обозначаются буквами от A до G; восьмой сегмент – десятичная точка (decimal point, DP).

В одноразрядном индикаторе девять выводов: восемь идут к аноду или к катоду **каждого** из сегментов, а один соответственно к катодам или анодам **всех** сегментов – этот вывод называется общим. Соответственно, схемы называются «с общим катодом» и «с общим анодом».

По сути, семисегментный индикатор представляет собой группу светодиодов, расположенных геометрически по форме цифры. Единственным отличием является наличие общего контакта катода для всех светодиодов. В нашей работе используется семисегментный индикатор с общим катодом.

Задача 1. Изучить работу семисегментого индикатора.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Семисегментный индикатор (1 шт.);
- Резистор 220 Ом (8 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 3 – 5.

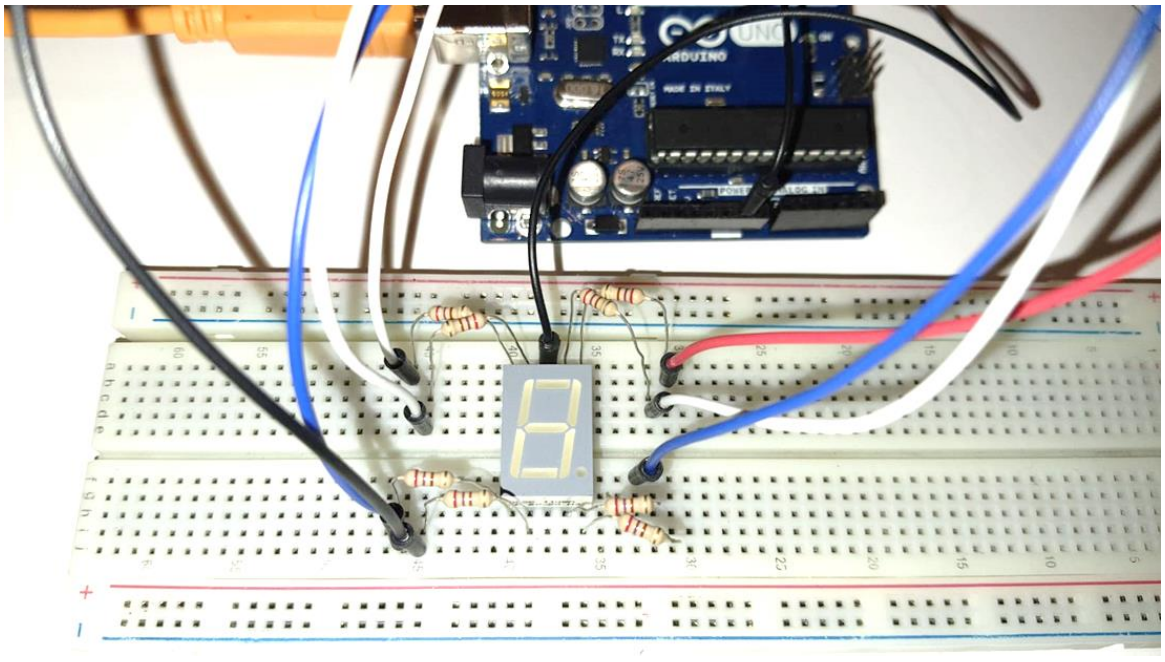


Рисунок 3

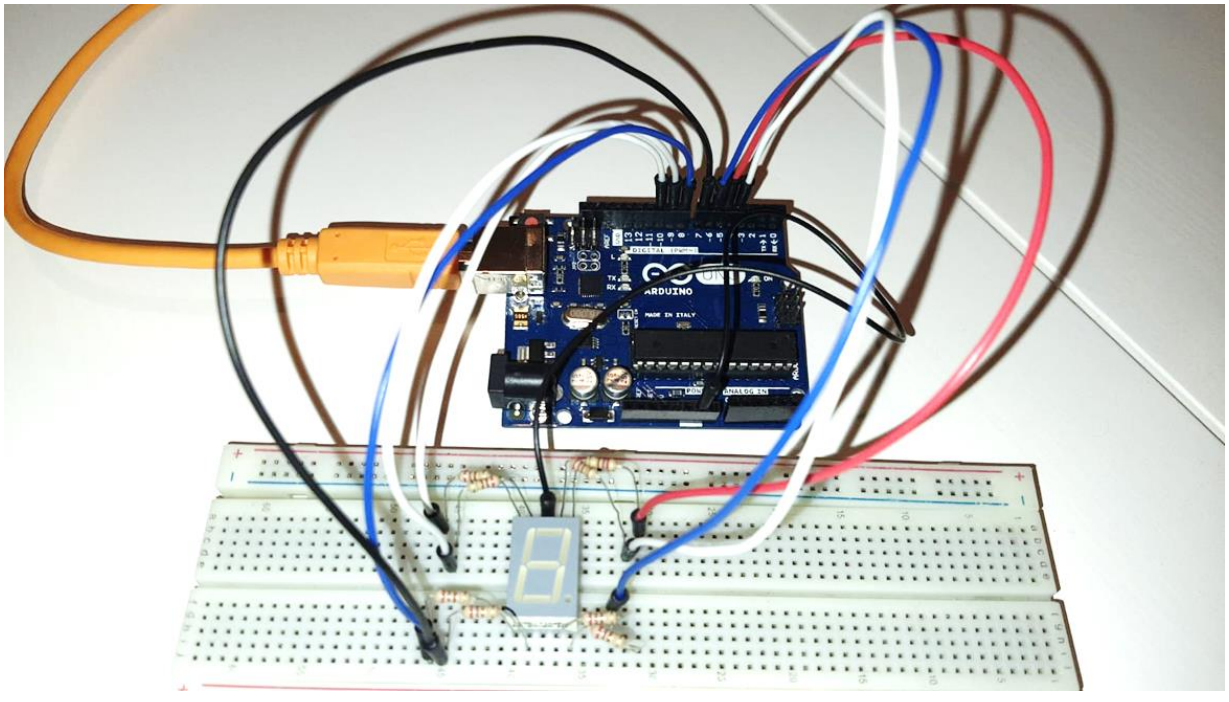


Рисунок 4

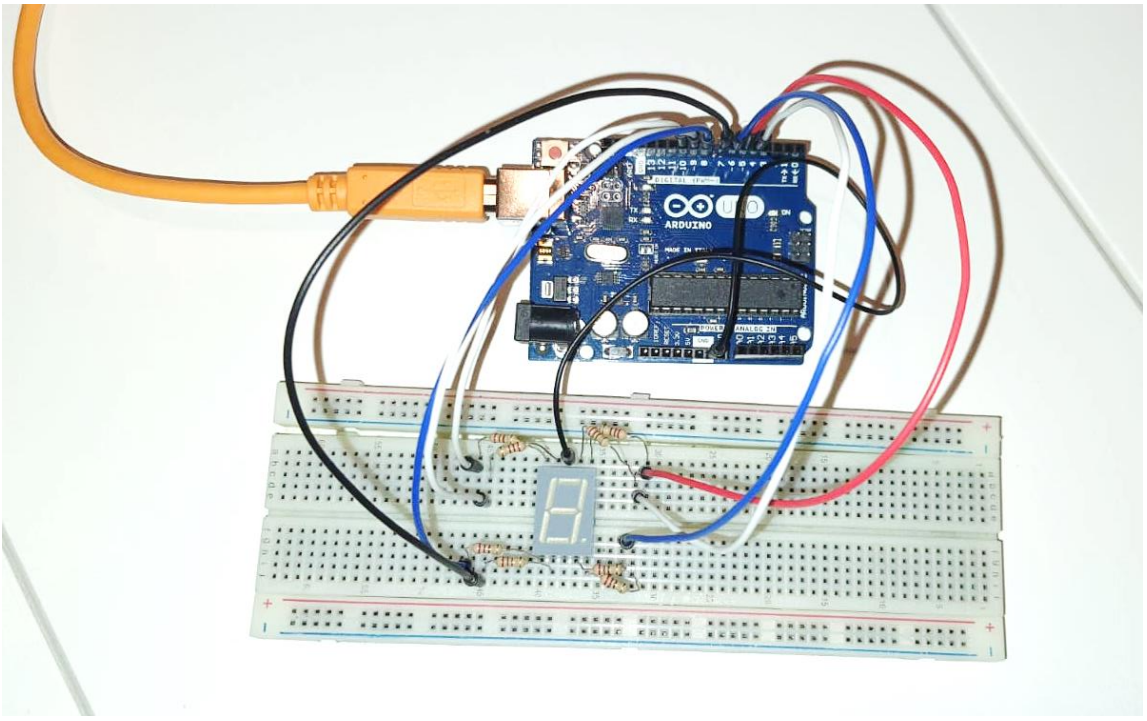


Рисунок 5

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1. После загрузки кода в плату на семисегментный будут последовательно выводиться цифры от 0 до 9.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Измените код таким образом, чтобы вывести буквы a, b, c.
2. Измените код таким образом, чтобы на индикаторе чередовались цифры и буквы a, b, c в случайном порядке.

Листинг 1

```
#define a 4   //сегмент a
#define b 5   //сегмент b
#define c 6   //сегмент c
#define d 7   //сегмент d
#define e 8   //сегмент e
#define f 9   //сегмент f
#define g 10  //сегмент g
#define DP 11 //сегмент десятичной точки

void setup() {
       //установка цифровых выходов в режим вывода
       pinMode(a, OUTPUT);
       pinMode(b, OUTPUT);
       pinMode(c, OUTPUT);
       pinMode(d, OUTPUT);
       pinMode(e, OUTPUT);
       pinMode(f, OUTPUT);
       pinMode(g, OUTPUT);
       pinMode(DP, OUTPUT);
}
void loop() {
       //цифра 0
       digitalWrite(a, HIGH);
       digitalWrite(b, HIGH);
       digitalWrite(c, HIGH);
       digitalWrite(d, HIGH);
       digitalWrite(e, HIGH);
       digitalWrite(f, HIGH);
       delay(1000);
       digitalWrite(a, LOW);
       digitalWrite(b, LOW);
       digitalWrite(c, LOW);
       digitalWrite(d, LOW);
       digitalWrite(e, LOW);
       digitalWrite(f, LOW);
       delay(1000);
```

```
      // цифра 1
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      delay(1000);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      delay(1000);

      // цифра 2
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(g, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(d, HIGH);
      delay(1000);

      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(g, LOW);
      digitalWrite(e, LOW);
      digitalWrite(d, LOW);
      delay(1000);

      // цифра 3
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(g, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, HIGH);
      delay(1000);

      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(g, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      delay(1000);
```

```
      // цифра 4
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      delay(1000);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      delay(1000);

      // цифра 5
      digitalWrite(a, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, HIGH);
      delay(1000);
      digitalWrite(a, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      delay(1000);

      // цифра 6
      digitalWrite(a, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, HIGH);
      delay(1000);
      digitalWrite(a, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
      digitalWrite(e, LOW);
```

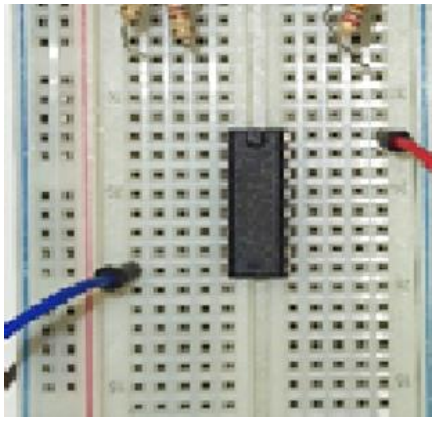
```
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      delay(1000);

      // цифра 7
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      delay(1000);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      delay(1000);

      // цифра 8
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
      delay(1000);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
      delay(1000);

      // цифра 9
      digitalWrite(f, HIGH);
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(g, HIGH);
      digitalWrite(c, HIGH);
```

```
        digitalWrite(d, HIGH);
        delay(1000);
        digitalWrite(f, LOW);
        digitalWrite(a, LOW);
        digitalWrite(b, LOW);
        digitalWrite(g, LOW);
        digitalWrite(c, LOW);
        digitalWrite(d, LOW);
        delay(1000);
}
```

Лабораторная работа № 10. Знакомство с подключением семисегментного индикатора через имс cd4026

На Рисунке 1 даны пояснения по расположению контактов на корпусе ИМС.

1 – clock (подключен ко второму выводу)
2 – disable clock
3 – enable display
4 – enable out
5 - ÷10 output
6 – output f
7 – output g
8 – 0V



16 - +3 to +15V
15 – reset (подключен к 3-му выводу)
14 – not 2 output
13 – output c
12 – output b
11 – output e
10 – output a
9 – output d

Рисунок 1

Задача 1. Вывод чисел от 0 до 9 на семисегментный индикатор.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Семисегментный индикатор (1 шт.);
- Драйвер семисегментного индикатора ИМС CD4026
- Резистор 220 Ом (7 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 2 – 5.

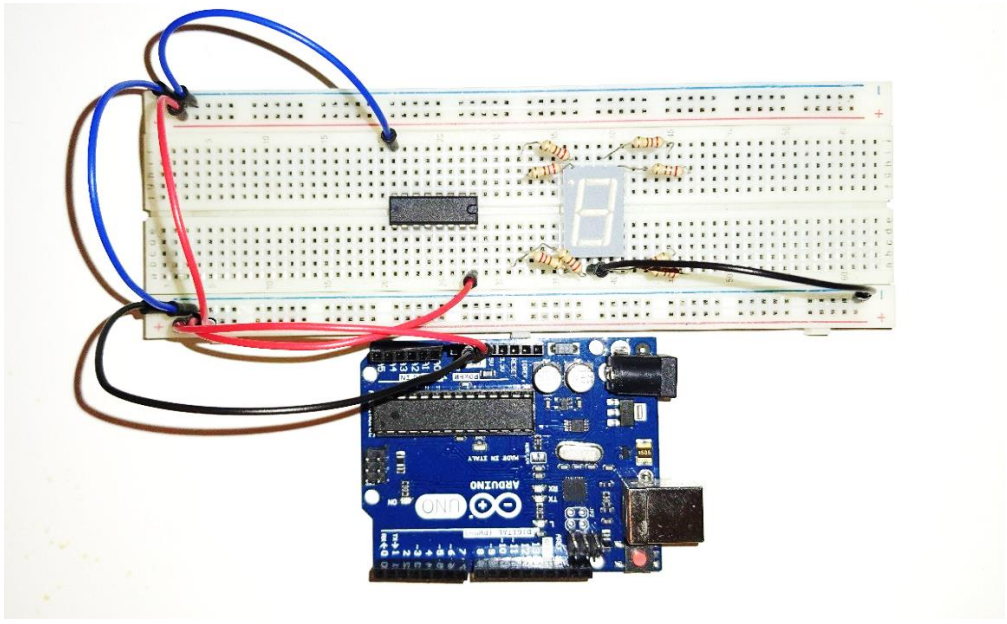


Рисунок 2

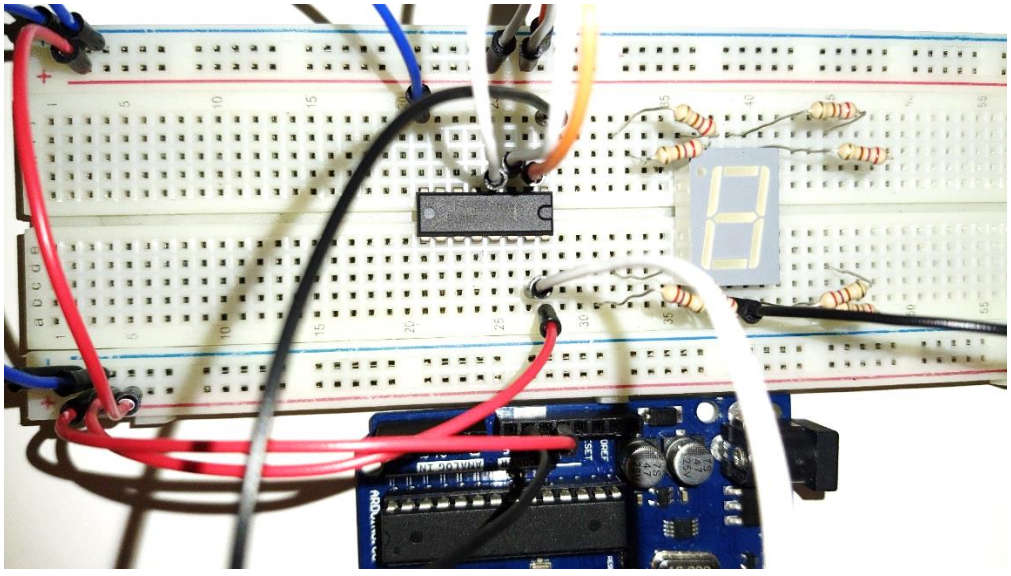


Рисунок 3

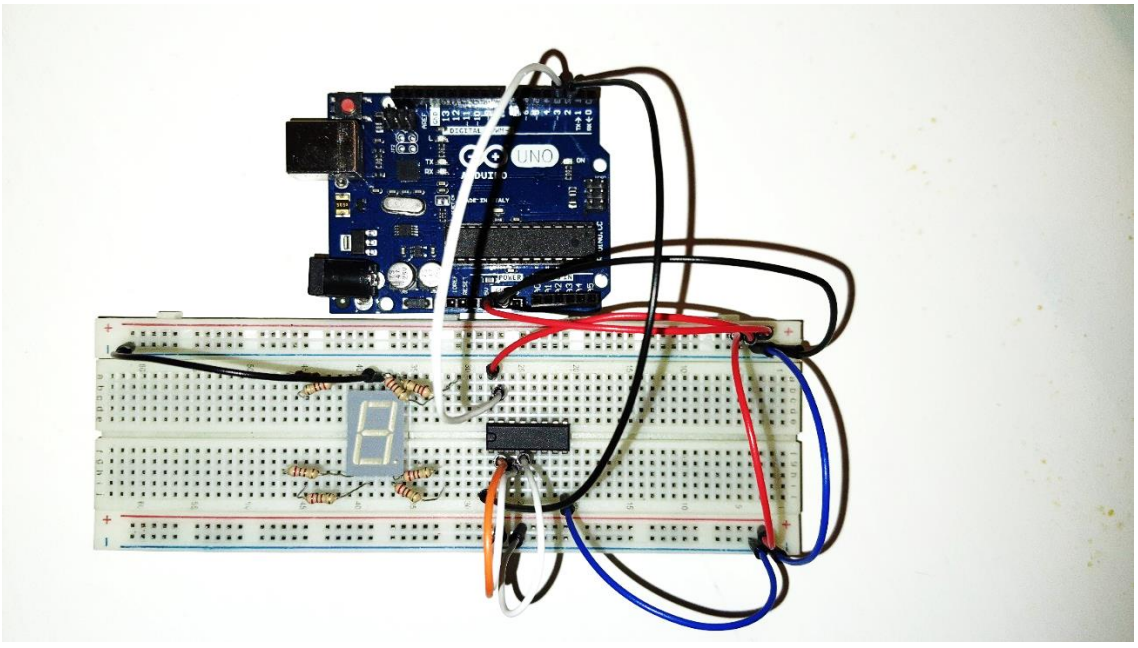


Рисунок 4

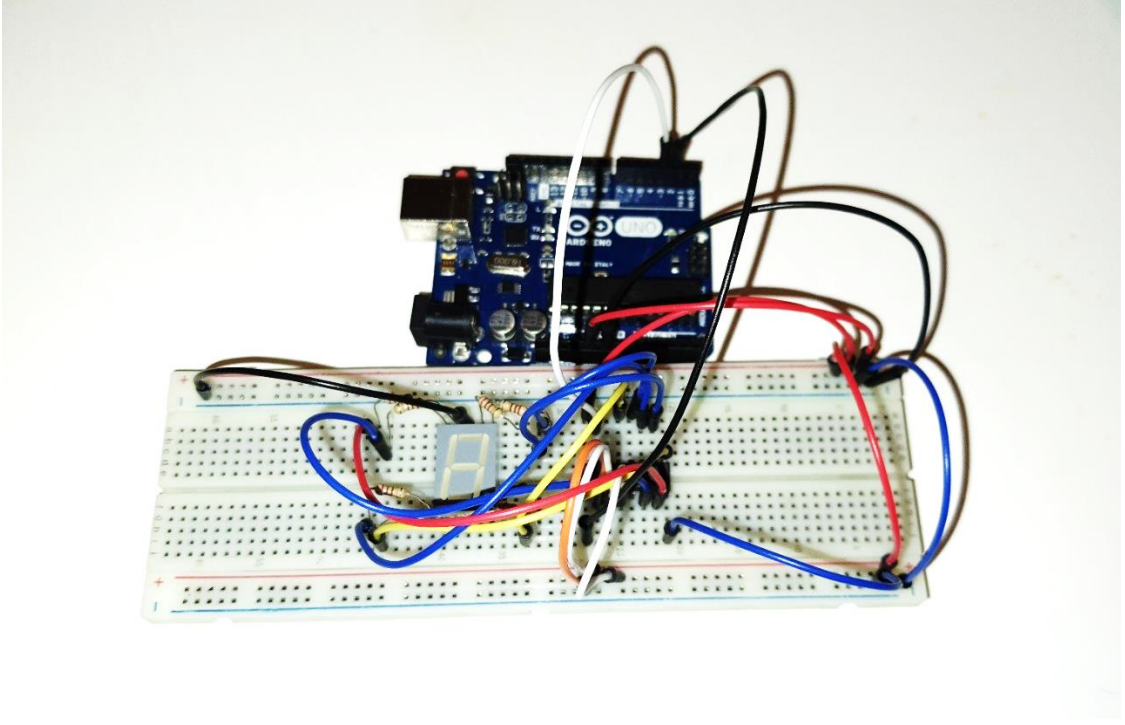


Рисунок 5

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1. После загрузки кода в плату на семисегментный индикатор будут последовательно выводиться цифры от 0 до 9.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Измените код таким образом, чтобы после вывода цифры 5 производилось обнуление подсчитанного числа и вывод цифр начинался заново (с 0).
2. Добавьте в схему светодиод и измените код таким образом, чтобы после вывода на индикатор очередной цифры была пауза в выводе цифр, а светодиод мигал столько раз, какая цифра была выведена.

Листинг 1

```
#define CL_pin 2  //Выход для счета
#define Res_pin 3 //Выход для сброса ИМС

void setup() {
       //Режим работы выход
       pinMode(CL_pin, OUTPUT);

       //Режим работы выход
       pinMode(Res_pin, OUTPUT);

       //Подача сигнала для сборса ИМС в 0
       digitalWrite(Res_pin, HIGH);

       //Время для сброса ИМС (необязательно)
       delay(10);

       //Подача пассивного сигнала для завершения сброса и перехода к работе ИМС
       digitalWrite(Res_pin, LOW);
}

void loop() {
```

```
        //Подача первой части счетного импульса
        digitalWrite(CL_pin, HIGH);
        //Задержка на реакцию ИМС в мкс (необязательно)
        delayMicroseconds(10);
        //Подача второй части счетного импульса (завершение увеличения на +1)
        digitalWrite(CL_pin, LOW);
        //Задержка на реакцию ИМС в мкс (необязательно)
        delayMicroseconds(10);
        delay(1000); //Задержка 1с для счета времени

}
```

Лабораторная работа № 11. Подсчет нажатия кнопки с выводом на семисегментный индикатор

Пояснения по расположению контактов на корпусе ИМС CD4026 см. в лабораторной работе № 10.

Задача 1. Вывести на семисегментный индикатор количества нажатий на кнопку.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Семисегментный индикатор (1 шт.);
- Драйвер семисегментного индикатора ИМС CD4026
- Резистор 220 Ом (7 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 1 – 3.

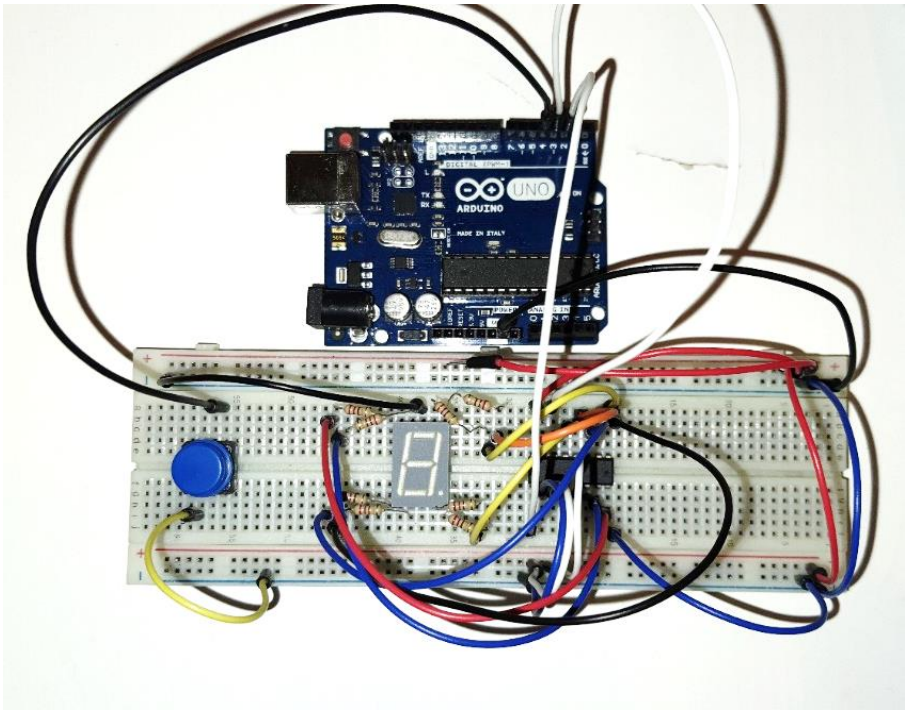


Рисунок 1

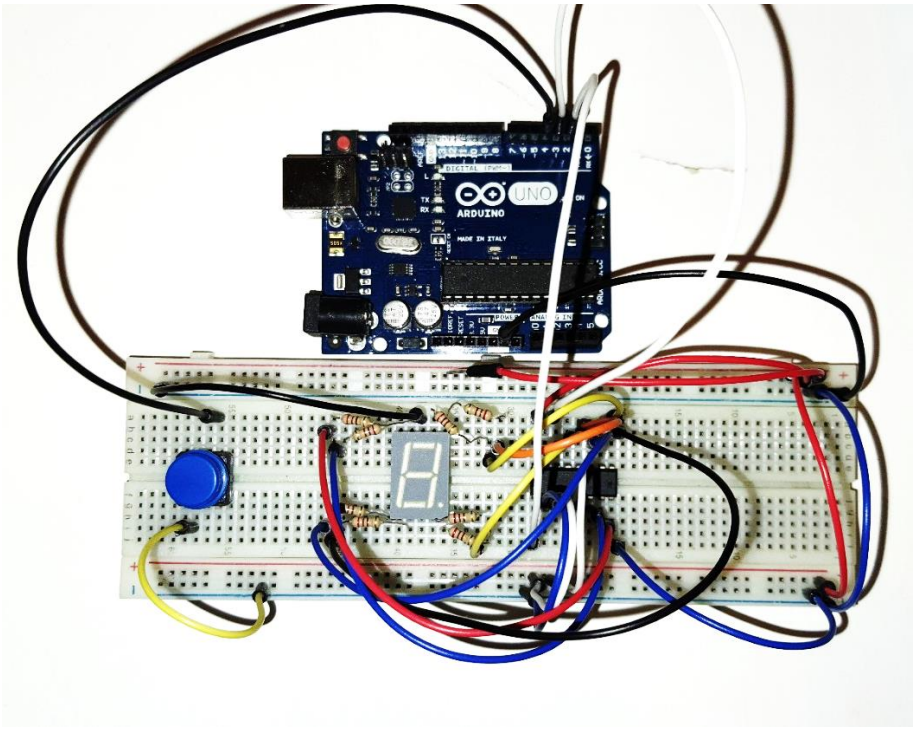


Рисунок 2

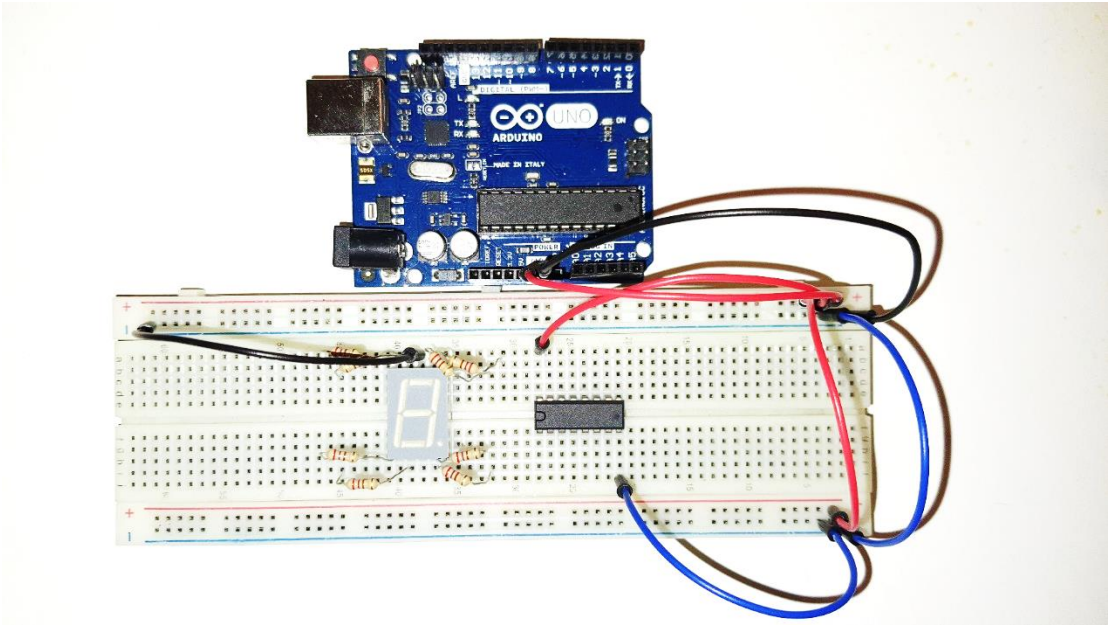


Рисунок 3

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте в схему светодиод и измените код таким образом, чтобы после вывода каждой цифры светодиод мигал столько раз, какая цифра была выведена.

2. Добавьте в схему второй светодиод и измените код таким образом, чтобы после вывода на индикатор очередной цифры была пауза в выводе цифр и первый светодиод мигал столько раз, какая цифра была выведена, а второй светодиод мигал бы столько раз, какова разность между 9 и количеством произведенных нажатий.

Листинг 1

```
#define C_pin 2 //Выход для счета
#define R_pin 3 //Выход для сброса ИМС
#define B_pin 4 //

void setup() {
	//Режим работы выход
	pinMode(C_pin, OUTPUT);

	//Режим работы выход
	pinMode(R_pin, OUTPUT);

	//Режим работы вход для кнопки
	pinMode(B_pin, INPUT_PULLUP);

	//Подача сигнала для сброса ИМС в 0
	digitalWrite(R_pin, HIGH);

	//Время для сброса ИМС (необязательно)
	delay(10);

	//Подача пассивного сигнала для завершения сброса и перехода к работе ИМС
	digitalWrite(R_pin, LOW);
}

void loop() {
	//Чтение нажатия кнопки
	if (!digitalRead(4))
	{
	  //Подача первой части счетного импульса
	  digitalWrite(C_pin, HIGH);
	  //Задержка на реакцию ИМС в мкс (необязательно)
	  delayMicroseconds(10);
	  //Подача второй части счетного импульса (завершение увеличения на +1)
	  digitalWrite(C_pin, LOW);
	  delayMicroseconds(10); //Задержка на реакцию ИМС в мкс (необязательно)
	}
}
```

Лабораторная работа № 12. Построение на двух семисегментных индикаторах

Две ИМС CD4026 объединены между собой через соединение: на 1-й ИМС 5-й контакт (÷ 10 output), а на 2-й ИМС – контакт 1-й (clock) (см. Рисунок 2).

Задача 1. Вывести на два семисегментных индикатора числа от 0 до 99.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Семисегментный индикатор (2 шт.);
- Драйвер семисегментного индикатора ИМС CD4026 (2 шт.)
- Резистор 220 Ом (14 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 1 – 3.

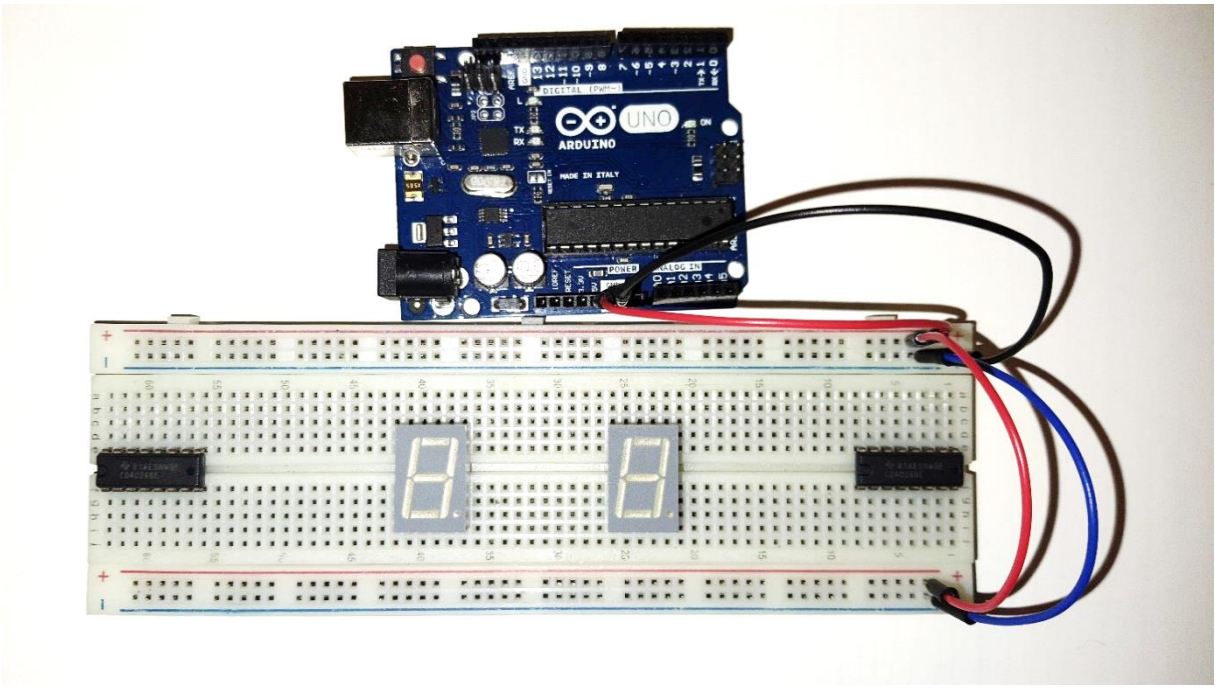


Рисунок 1

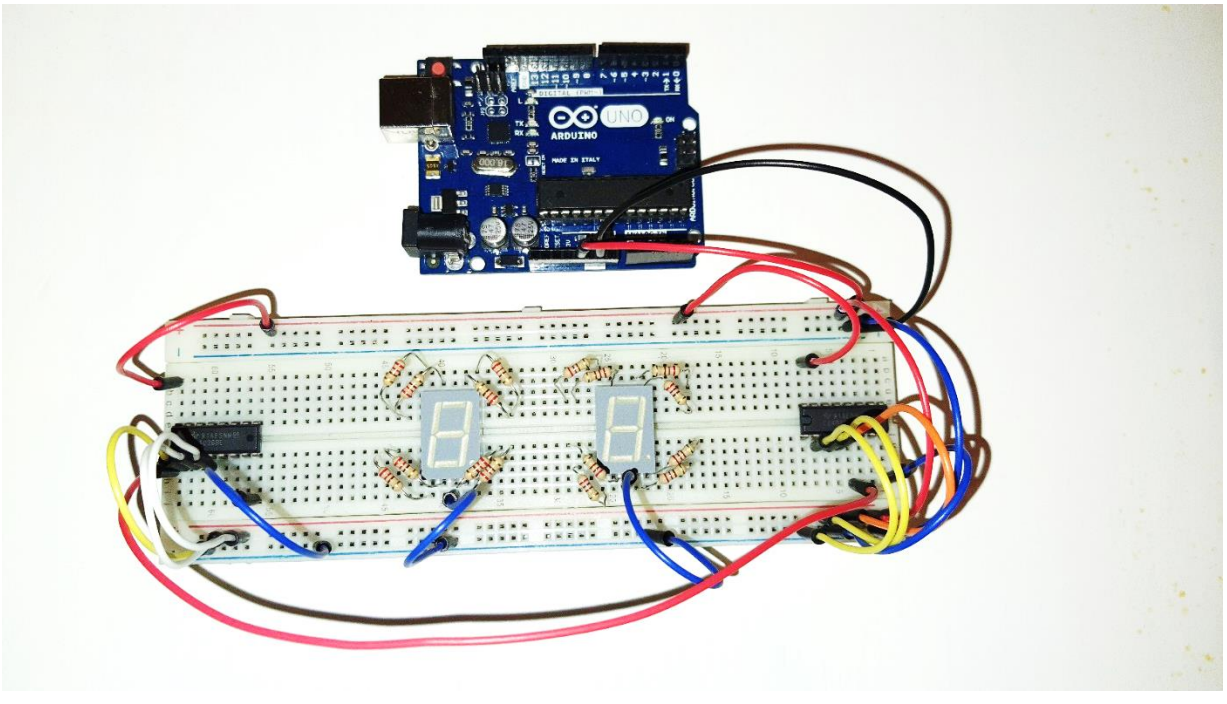


Рисунок 2

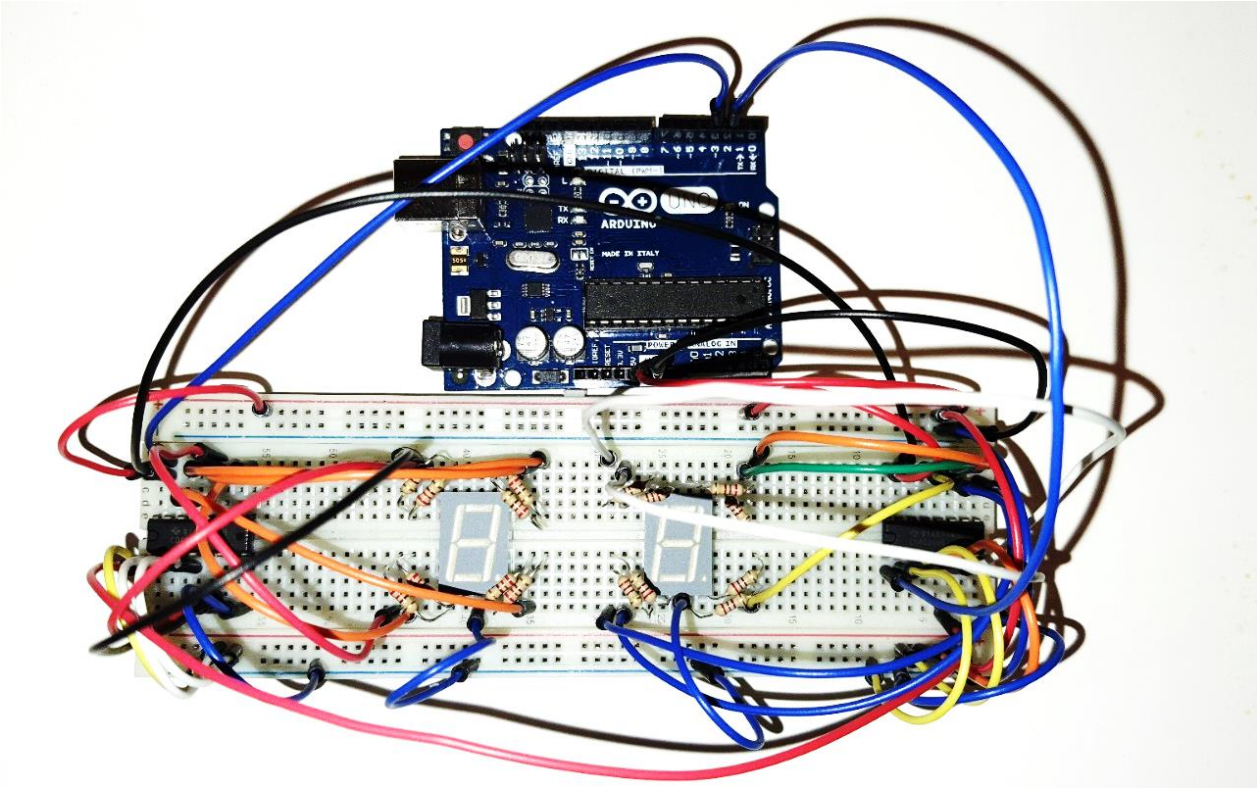


Рисунок 3

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте в схему клавишу принудительного программного сброса.

2. Добавьте в схему три светодиода: красный, желтый, зеленый. В начале работы программы светодиоды не горят. Когда выводимые цифры достигнут 30, должен загореться желтый светодиод. После вывода числа 60, должен загореться красный светодиод. И только после вывода числа 99, должен загораться зеленый светодиод.

Листинг 1

```
#define CL_pin 2  //Выход для счета
#define Res_pin 3 //Выход для сброса ИМС

void setup() {

	//Режим работы выход
	pinMode(CL_pin, OUTPUT);

	//Режим работы выход
	pinMode(Res_pin, OUTPUT);

	//Подача сигнала для сброса ИМС в 0
	digitalWrite(Res_pin, HIGH);

	//Время для сброса ИМС (необязательно)
	delay(10);

	//Подача пассивного сигнала для завершения сброса и перехода к работе ИМС
	digitalWrite(Res_pin, LOW);
}

void loop() {

	//Подача первой части счетного импульса
	digitalWrite(CL_pin, HIGH);

	//Задержка на реакцию ИМС в мкс (необязательно)
	delayMicroseconds(10);

	//Подача второй части счетного импульса (завершение увеличения на +1)
	digitalWrite(CL_pin, LOW);

	//Задержка на реакцию ИМС в мкс (необязательно)
	delayMicroseconds(10);

	//Задержка 100мс для счета времени
	delay(100);
```

```
}
```

Лабораторная работа № 13. Вывод случайных чисел на семисегментный индикатор

Задача 1. Выводить на семисегментный индикатор случайное число по нажатию кнопки.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Семисегментный индикатор (1 шт.);
- Резистор 220 Ом (7 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 1 – 3.

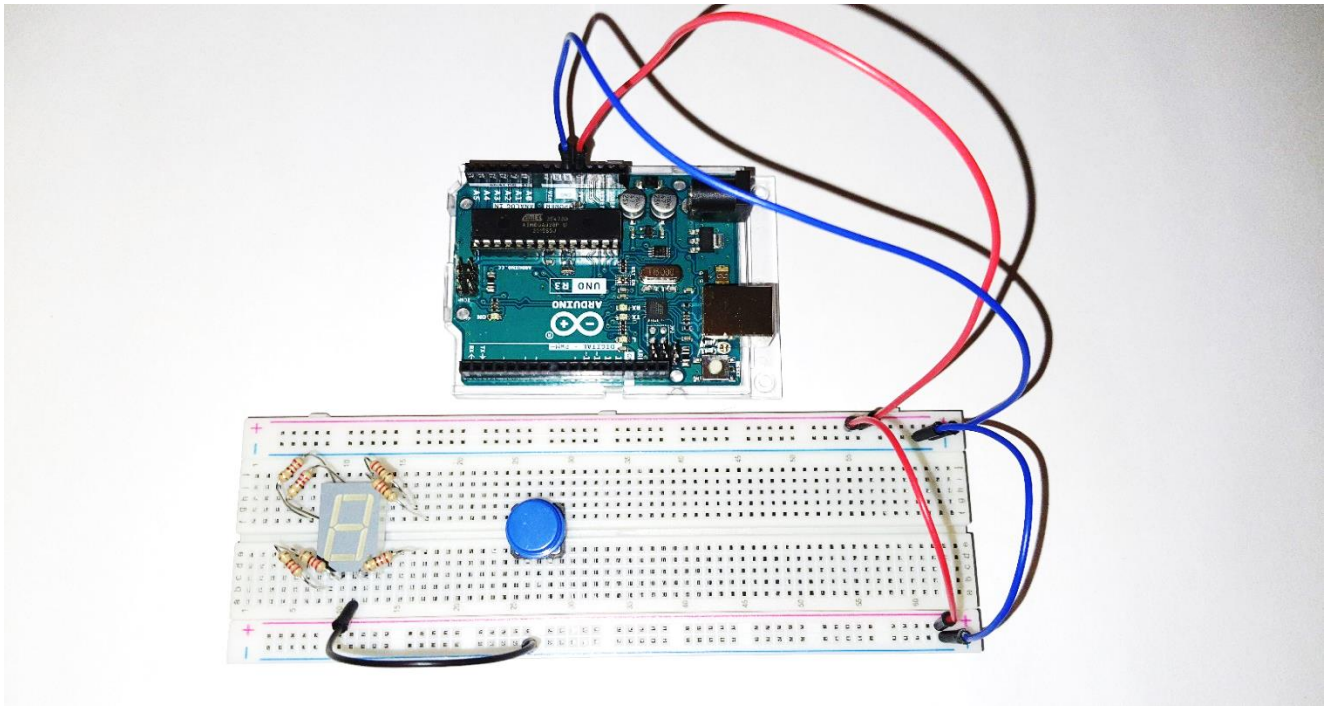


Рисунок 1

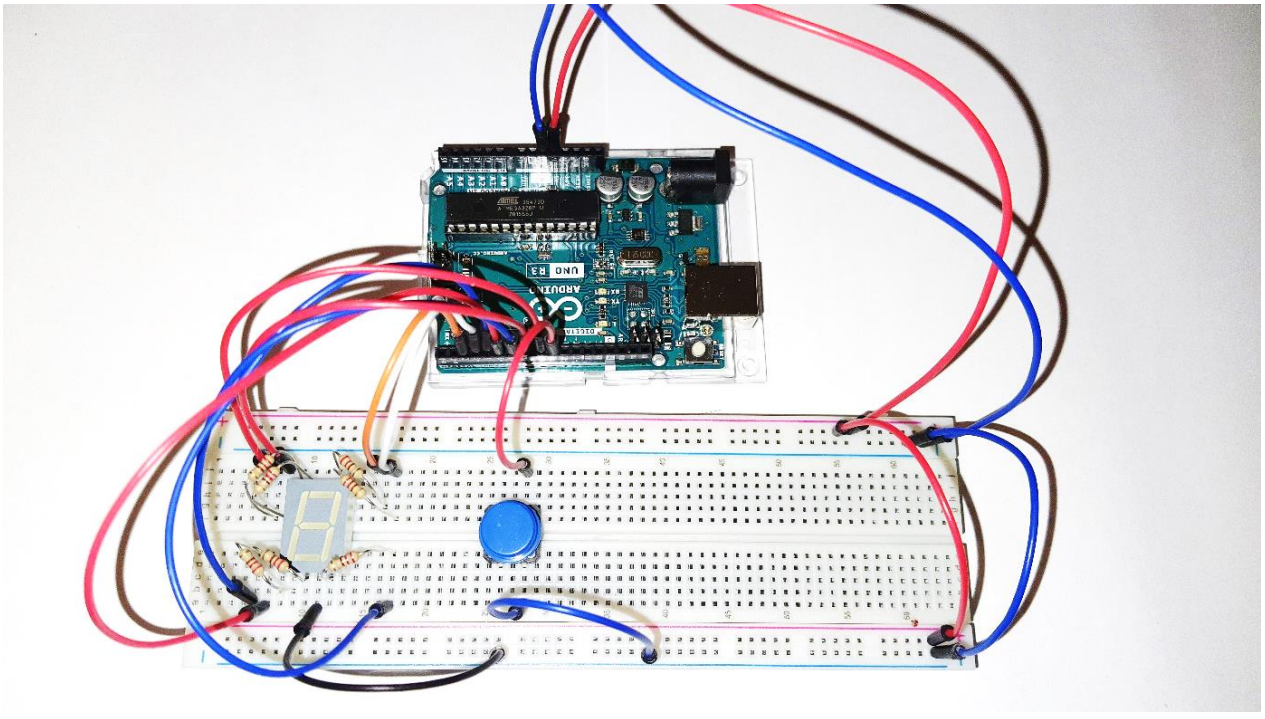


Рисунок 2

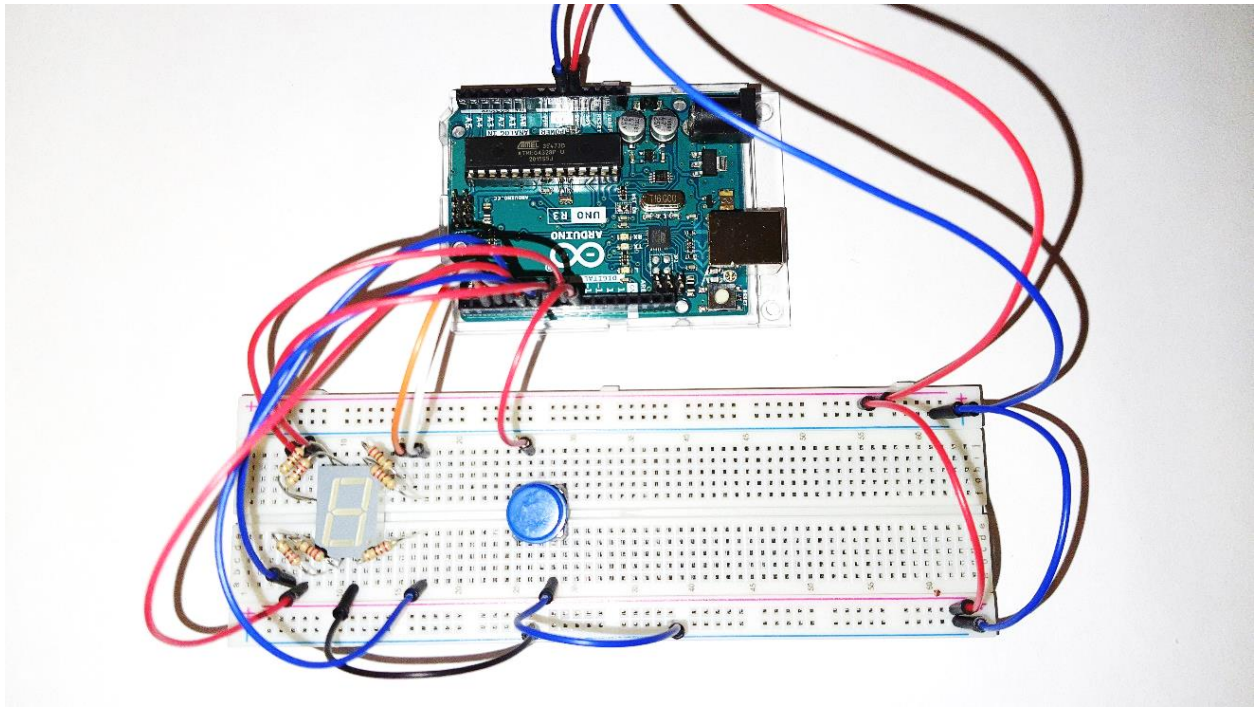


Рисунок 3

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте кнопку для сброса в значение 0.

2. Добавьте в схему светодиод, яркость которого будет соответствовать выводимому случайному числу.

Листинг 1

```
#define sA 2 //Контакт сегмента
#define sB 3 //Контакт сегмента
#define sC 4 //Контакт сегмента
#define sD 5 //Контакт сегмента
#define sE 6 //Контакт сегмента
#define sF 7 //Контакт сегмента
#define sG 8 //Контакт сегмента
#define BT 9 //Контакт кнопки

void setup() {

       //Настройка контактов на вывод
       for(int i=2; i<=8; i++)
       {
         pinMode(i, OUTPUT);
       }
       //Контакт кнопки установлен в режим вход с нагрузочным резистором
       pinMode(9, INPUT_PULLUP);
}

void loop() {

       //Чтение состояния кнопки
       if(!digitalRead(9))
       {

          //Генерация случайного числа для индикатора
          int Num=random(0, 10);
```

```
        //Вывод на индикатор цифры через управление каждым сегментом
        if(Num==0)
        {
           digitalWrite(sA, HIGH);
           digitalWrite(sB, HIGH);
           digitalWrite(sC, HIGH);
           digitalWrite(sD, HIGH);
           digitalWrite(sE, HIGH);
           digitalWrite(sF, HIGH);
           digitalWrite(sG, LOW);
        }

        if(Num==1)
        {
           digitalWrite(sA, LOW);
           digitalWrite(sB, HIGH);
           digitalWrite(sC, HIGH);
           digitalWrite(sD, LOW);
           digitalWrite(sE, LOW);
           digitalWrite(sF, LOW);
           digitalWrite(sG, LOW);
        }

        if(Num==2)
        {
           digitalWrite(sA, HIGH);
           digitalWrite(sB, HIGH);
           digitalWrite(sC, LOW);
           digitalWrite(sD, HIGH);
           digitalWrite(sE, HIGH);
           digitalWrite(sF, LOW);
           digitalWrite(sG, HIGH);
         }

         if(Num==3)
         {
           digitalWrite(sA, HIGH);
           digitalWrite(sB, HIGH);
           digitalWrite(sC, HIGH);
           digitalWrite(sD, HIGH);
           digitalWrite(sE, LOW);
           digitalWrite(sF, LOW);
           digitalWrite(sG, HIGH);
         }
```

```
  if(Num==4)
  {
    digitalWrite(sA, LOW);
    digitalWrite(sB, HIGH);
    digitalWrite(sC, HIGH);
    digitalWrite(sD, LOW);
    digitalWrite(sE, LOW);
    digitalWrite(sF, HIGH);
    digitalWrite(sG, HIGH);
  }

  if(Num==5)
  {
    digitalWrite(sA, HIGH);
    digitalWrite(sB, LOW);
    digitalWrite(sC, HIGH);
    digitalWrite(sD, HIGH);
    digitalWrite(sE, LOW);
    digitalWrite(sF, HIGH);
    digitalWrite(sG, HIGH);
  }

  if(Num==6)
  {
    digitalWrite(sA, HIGH);
    digitalWrite(sB, LOW);
    digitalWrite(sC, HIGH);
    digitalWrite(sD, HIGH);
    digitalWrite(sE, HIGH);
    digitalWrite(sF, HIGH);
    digitalWrite(sG, HIGH);
  }

  if(Num==7)
  {
    digitalWrite(sA, HIGH);
    digitalWrite(sB, HIGH);
    digitalWrite(sC, HIGH);
    digitalWrite(sD, LOW);
    digitalWrite(sE, LOW);
    digitalWrite(sF, LOW);
    digitalWrite(sG, LOW);
  }
```

```
            if(Num==8)
            {
              digitalWrite(sA, HIGH);
              digitalWrite(sB, HIGH);
              digitalWrite(sC, HIGH);
              digitalWrite(sD, HIGH);
              digitalWrite(sE, HIGH);
              digitalWrite(sF, HIGH);
              digitalWrite(sG, HIGH);
            }

            if(Num==9)
            {
              digitalWrite(sA, HIGH);
              digitalWrite(sB, HIGH);
              digitalWrite(sC, HIGH);
              digitalWrite(sD, HIGH);
              digitalWrite(sE, LOW);
              digitalWrite(sF, HIGH);
              digitalWrite(sG, HIGH);
            }

        }

}
```

Лабораторная работа № 14. Формирование нот на пьезодинамике с выводом индикации на семисегментный индикатор

Задача 1. Формирование нот на пьезодинамике с выводом индикации на семисегментный индикатор.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Семисегментный индикатор (1 шт.);
- Пьезопищалка (1 шт.)
- Резистор 220 Ом (7 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 1 – 3.

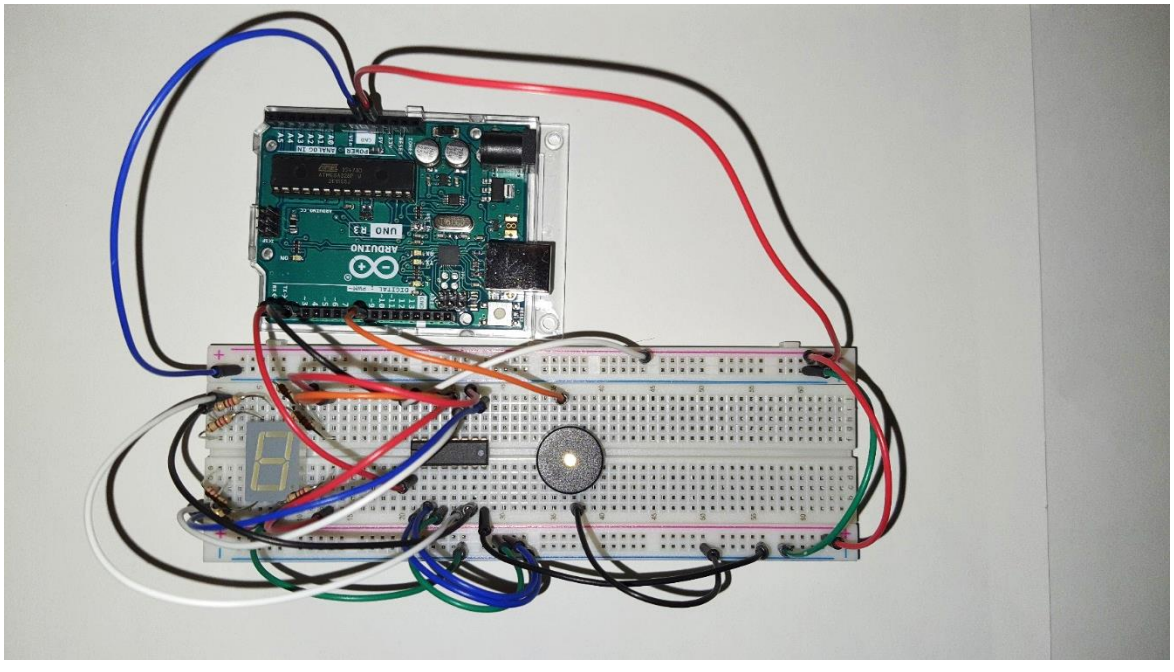


Рисунок 1

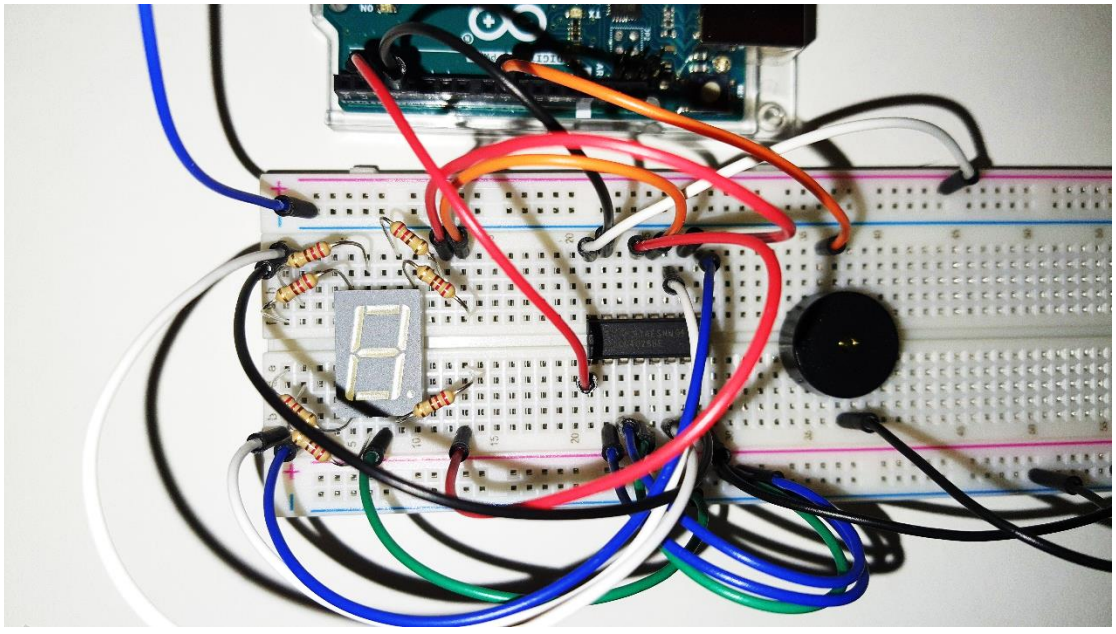


Рисунок 2

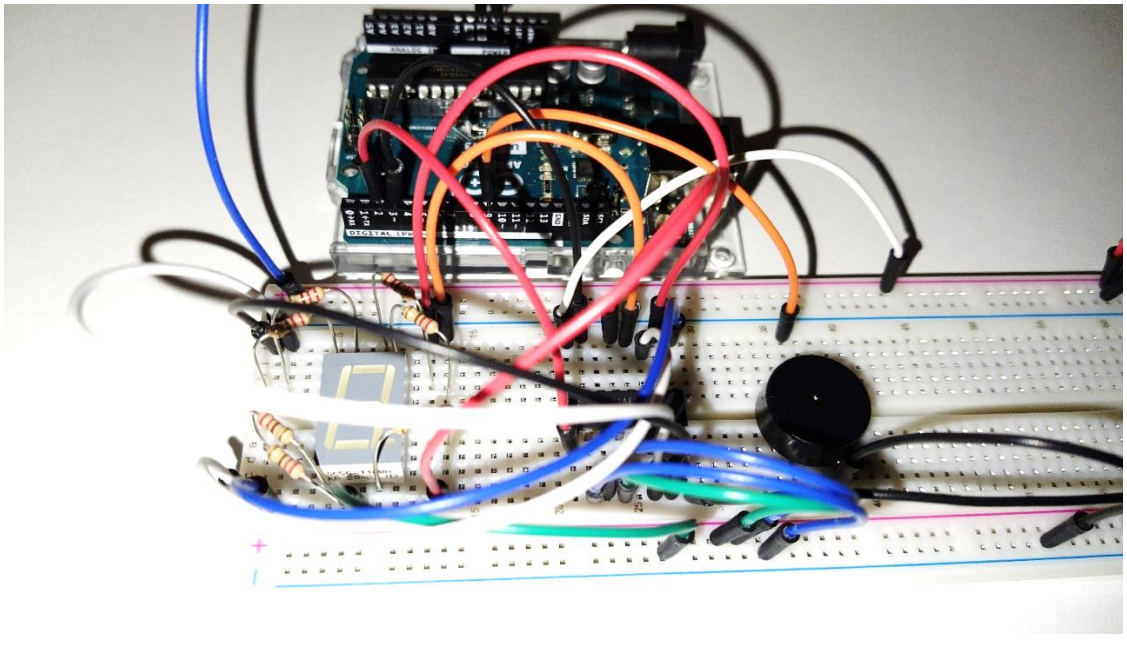


Рисунок 3

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.

4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте кнопку для вызова определенной мелодии.
2. Добавьте в схему семисегментный индикатор, на который выводятся случайные числа, а мелодии соответствуют выводимому случайному числу.

Листинг 1

```
/*
* Работа №14
* Формирование нот на пьезодинамике с выводом индикации на семисегментоном индикаторе
*/

#define Ck 2
#define Rs 3
#define Sound 9

void setup() {
     pinMode(Ck, OUTPUT);
     pinMode(Rs, OUTPUT);
     pinMode(Sound, OUTPUT);

     //Стартовый сброс индикатора
     digitalWrite(Rs, HIGH);
     delayMicroseconds(10);
     digitalWrite(Rs, LOW);
}

void loop() {

       //Переменная для частоты ноты
       //Цикл перебора нот
       int F=0;
       for (int i=0; i<=6; i++)
       {
          //Формирование импульса для ИМС +1 на индикаторе
          digitalWrite(Ck, HIGH);
          delayMicroseconds(10);
          digitalWrite(Ck, LOW);
```

```
          //Увеличение частоты ноты в каждой итерации цикла
          F=2500+i*100;

          //Вывод звука на контакте Sound с частотой F, на 1000мс
          tone(Sound, F, 1000);

          // Задержка для цикла for на 10мс больше длительности ноты
          delay(1010);
        }

        //Сброс индикатора после последней ноты
        digitalWrite(Rs, HIGH);
        delayMicroseconds(10);
        digitalWrite(Rs, LOW);

        //Задержка что бы увидеть 0 на индикаторе
        delay(1000);
      }
}
```

Лабораторная работа № 15. Таймер с индикацией на семисегментном индикаторе

Задача 1. Выполнить сборку и написание кода для таймера с индикацией на семисегментном индикаторе.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Семисегментный индикатор (1 шт.);
- ИМС CD4026 (2 шт.)
- Резистор 220 Ом (14 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 1 – 3.

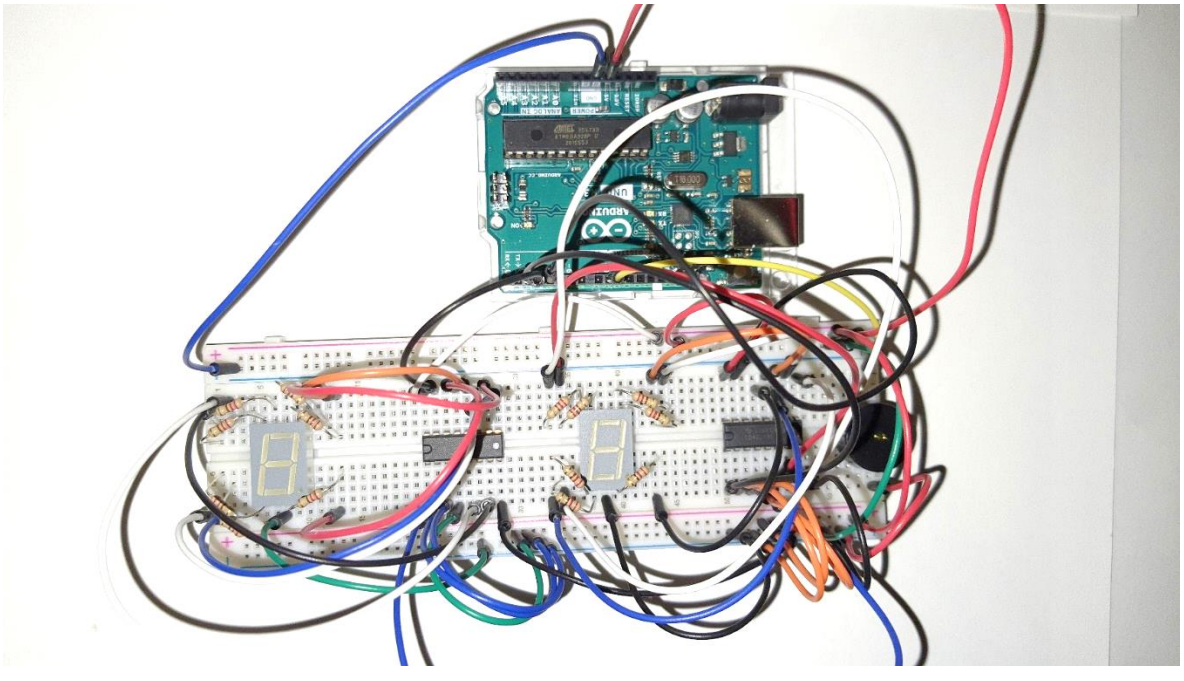


Рисунок 1

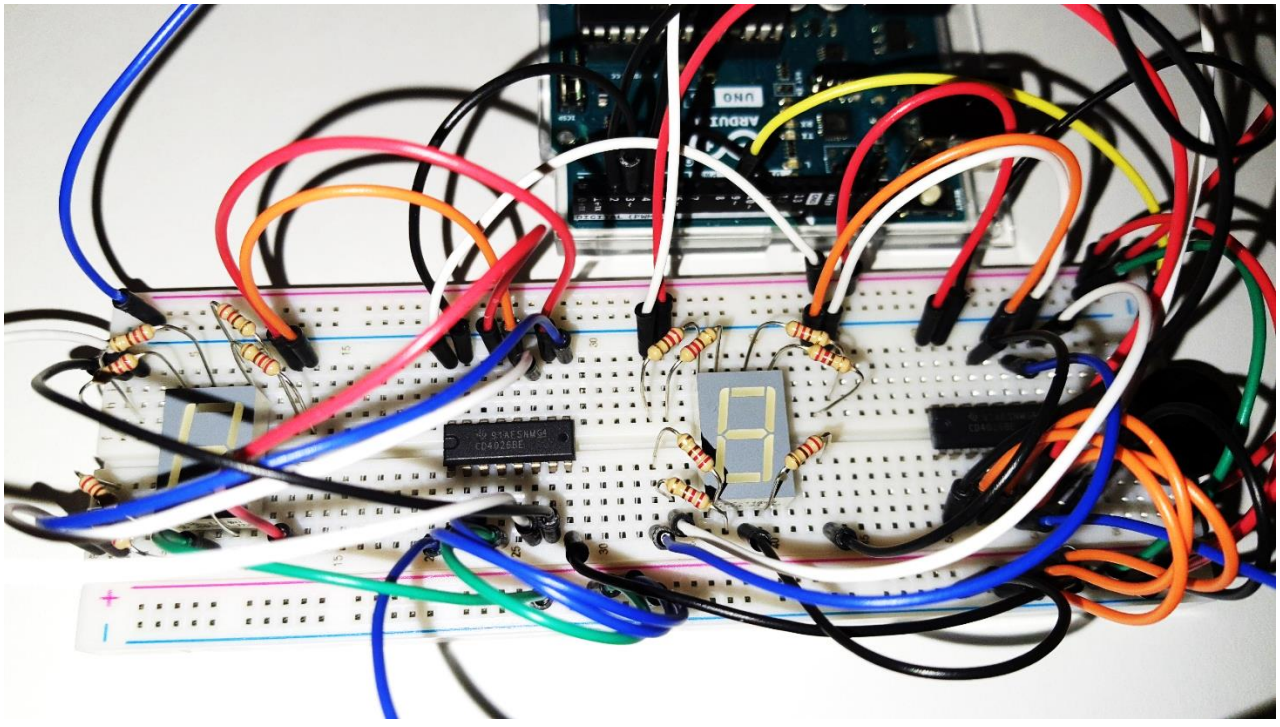


Рисунок 2

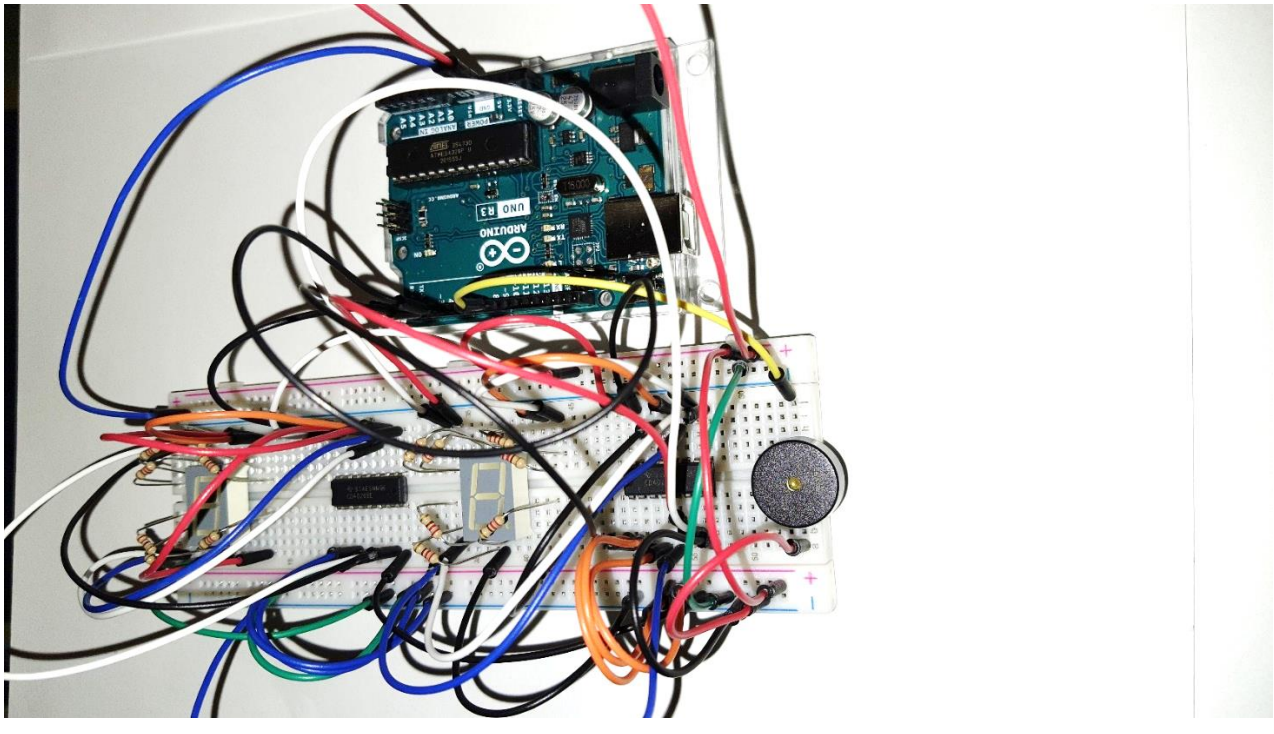


Рисунок 3

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте в схему RGB-светодиод, цвет которого будет меняться от зеленого до красного.
2. Добавьте в схему пьезопищалку, звук которой будет меняться от низкой до высокой частоты.

Листинг 1

```
/*
* Работа №15
* Таймер на семисегментном индикаторе
*/

#define Ck 2    //Контакт счета
#define Rs1 4   //Контакт сброса индикаторя единиц
#define Rs2 3   //Контакт сброса индикатора десятков
#define Sound 9 //Контакт звукового излучателя

void setup() {

       pinMode(Ck, OUTPUT);
       pinMode(Rs1, OUTPUT);
       pinMode(Rs2, OUTPUT);

       //Сброс индикатора 1
       digitalWrite(Rs1, HIGH);
       delayMicroseconds(10);
       digitalWrite(Rs1, LOW);

       //Сброс индикатора 2
       digitalWrite(Rs2, HIGH);
       delayMicroseconds(10);
       digitalWrite(Rs2, LOW);

}

void loop() {
```

```
        //Цикл считающий предел таймера в десятках секунд
        for (int j=0; j<=2; j++)
        {
          //Цикл считающий секунды
          for (int i=0; i<=9; i++)
          {
            //Импульс для изменения индикатора единиц на +1
            digitalWrite(Ck, HIGH);
            delayMicroseconds(10);
            digitalWrite(Ck, LOW);

            //Задержка формирующая 1 секунду
            delay(1000);
          }
        }
        //Звуковая индикация окончания таймера
        tone(Sound, 3500, 3900);
        //Задержка работы индикации пока звучит звук
        delay(4000);
        //Сброс индикатора десятков
        digitalWrite(Rs2, HIGH);
        delayMicroseconds(10);
        digitalWrite(Rs2, LOW);
        //Задержка для индикации нулевого состояния таймера
        delay(1000);
}
```

Лабораторная работа № 16. Знакомство с ЖКИ

Задача 1. Выполнить сборку и написание кода для знакомства с ЖКИ.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- ЖКИ экран (1 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунках 1 – 5.

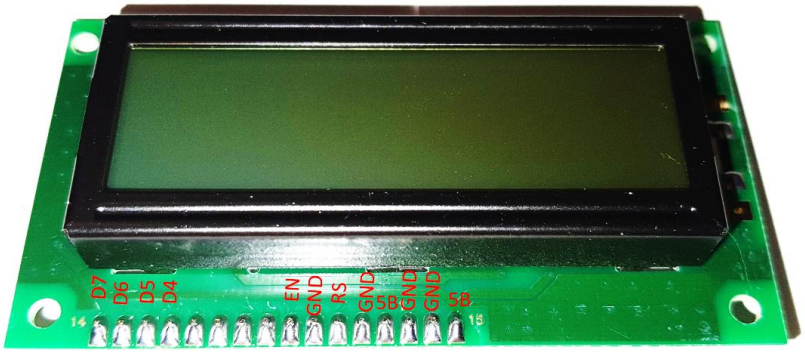


Рисунок 1

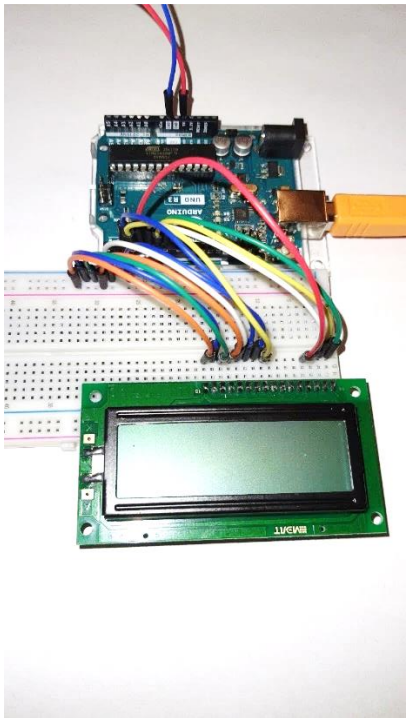


Рисунок 2

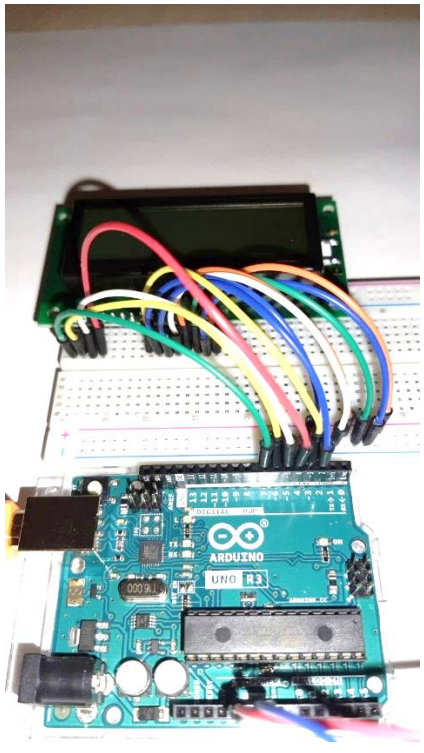


Рисунок 3

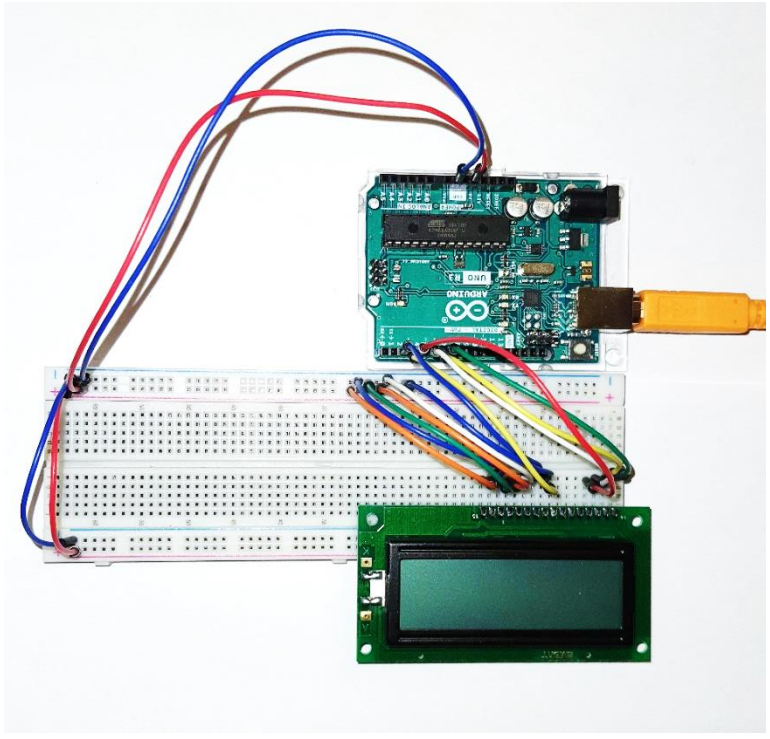


Рисунок 4

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Выведите свои имя и фамилию на экран.

2. Выведите на экран в цикле набор из пяти любых слов (разной длины). Изучите процесс обновления ЖКИ.

Листинг 1

```
/*
* Работа №15
* Знакомство с ЖКИ
*/

//Библиотека для работы с ЖКИ
#include <LiquidCrystal.h>

//Назначение контакотов для работы с ЖКИ
#define RS 2
#define EN 3
#define D4 4
#define D5 5
#define D6 6
#define D7 7

//Создание объекта для ЖКИ и передача ему номеров контактов платы
LiquidCrystal LCDnew(RS, EN, D4, D5, D6, D7);

void setup(){
	//Инициализация работы с ЖКИ, с передачей размерности экрана (16 символов
	в 2 строках)
	LCDnew.begin(16,2);

	//Вывод начиная со стартовой позиции ЖКИ (0,0) сообщения в скобках
	LCDnew.print("Hello student");
}
```

Лабораторная работа № 17. Вывод нажатия кнопки на жки

Задача 1. Написать программу, которая выводит на ЖКИ сообщение о нажатия кнопки.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- ЖКИ экран (1 шт.);
- Кнопка (1 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунке 1.

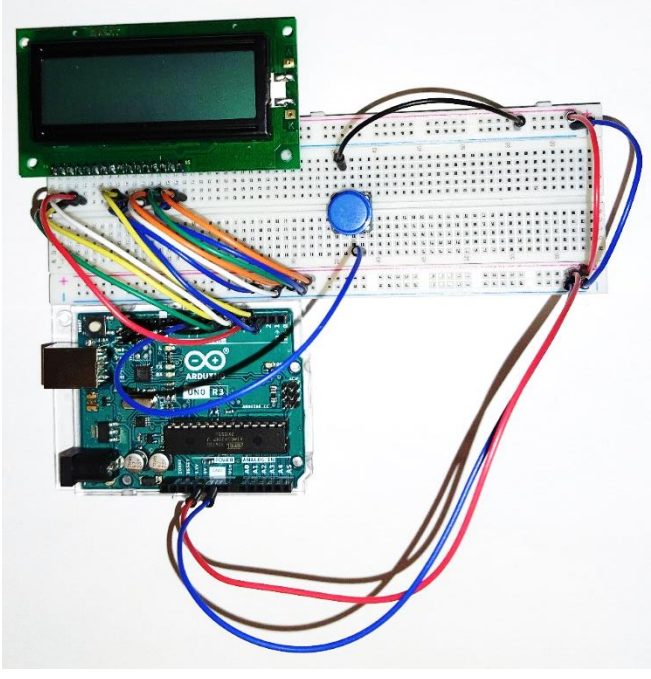


Рисунок 1

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте в схему вторую кнопку и напишите программу, в результате выполнения которой сообщение о нажатии одной кнопки выводится в первую строку, а сообщении о нажатии другой – во вторую строку.
2. Добавьте в схему вторую кнопку, по нажатии которой экран будет очищаться.

Листинг 1

```
/*
* Работа №17
* Вывод нажатия кнопки на ЖКИ
*/

//Библиотека для работы с ЖКИ
#include <LiquidCrystal.h>

//Назначение контактов для работы с ЖКИ
#define RS 2
#define EN 3
#define D4 4
#define D5 5
#define D6 6
#define D7 7

//Контакт кнопки
#define BT 9

//Создание объекта для ЖКИ и передача ему номеров контактов платы
LiquidCrystal LCDnew(RS, EN, D4, D5, D6, D7);

//Переменная, хранящая количество нажатий кнопки
int Num=0;

void setup() {
```

```
        //Инициализация работы с ЖКИ, с передачей размерности экрана (16 символов
        в 2 строках)
        LCDnew.begin(16,2);
        LCDnew.print("Nagatia BUTTON");
        pinMode(9, INPUT_PULLUP); //Настройка контакта кнопки
}

void loop() {
        //Чтение значения кнопки
        if(!digitalRead(9))
        {
          //Увеличение значения на +1, на одно нажатие
          Num++;
          //Перемещение курсора ЖКИ для вывода количества нажатий во второй строке
          LCDnew.setCursor(0,1);
          //Вывод содержимого переменной на ЖКИ
          LCDnew.print(Num);
          //Ожидание пока нажатие на кнопку прекратится
          while(!digitalRead(9))
          {

          }
        }
}
```

Лабораторная работа № 18. Вывод уровня освещенности в помещении на жки

Задача 1. Написать программу, которая выводит на ЖКИ сообщение об уровне освещенности в помещении.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- ЖКИ экран (1 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунке 1.

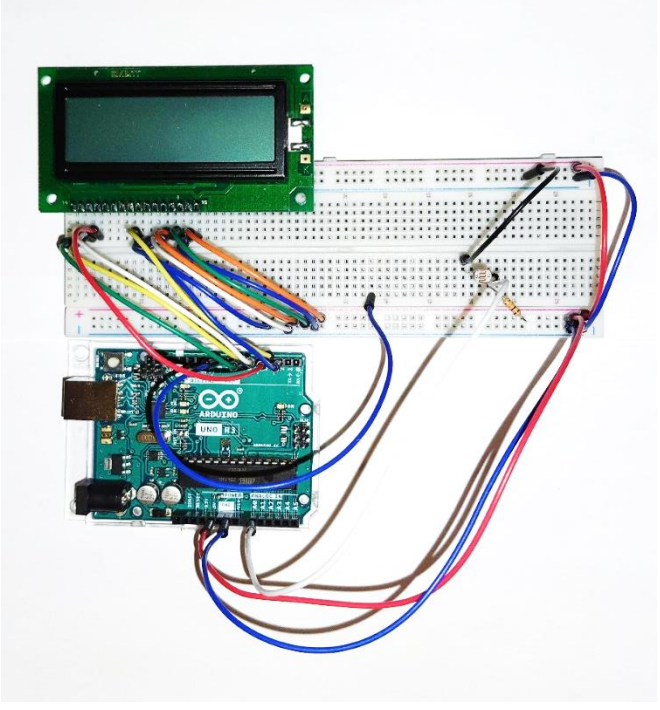


Рисунок 1

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.

4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Измените программный код так, чтобы во вторую строку выводились сведения о том, является ли данный уровень освещенности достаточным.

2. Добавьте в схему пять светодиодов. Все светодиоды горят, когда уровень освещенности максимальный; если уровень освещенности уменьшается, часть светодиодов гаснет.

Листинг 1

```
/*
* Работа №18
* Вывод уровня освещенности в помещении на ЖКИ
* P.S. Во второй строке буква 'E' от слова "SREDNE"
* не перезаписывается и не обновляется -> Д.З.
* воспользоваться библиотекой LiquidCrustal что бы
* очищать эту позицию на ЖКИ
*/

//Библиотека для работы с ЖКИ
#include <LiquidCrystal.h>

//Назначение контактов для работы с ЖКИ
#define RS 2
#define EN 3
#define D4 4
#define D5 5
#define D6 6
#define D7 7

//Контакт для фоторезистора
#define OSV A0

//Создание объекта для ЖКИ и передача ему номеров контактов платы
LiquidCrystal LCDnew(RS, EN, D4, D5, D6, D7);

void setup() {
     //Инициализация работы с ЖКИ, с передачей размерности экрана (16 символов в
     2 строках)
     LCDnew.begin(16,2);
     //Вывод начиная со стартовой позиции ЖКИ (0,0) сообщения в скобках
```

```
     LCDnew.print("Yroven osvechennosti");
}

void loop() {
     //Чтение значения освещенности
     int ReadOSV = analogRead(A0);

     //Выбор выводимого сообщения в зависимости от уровня освещенности
     if((ReadOSV>500) && (ReadOSV<1024))
     {

         //Выбор начала со второй строки для вывода на ЖКИ
         LCDnew.setCursor(0,1);

         //Вывод сообщения о яркости
         LCDnew.print("TEMNO");
     }

     if((ReadOSV>300) && (ReadOSV<400))
     {
         LCDnew.setCursor(0,1);
         LCDnew.print("SREDNE");
     }

     if((ReadOSV>0) && (ReadOSV<250))
     {
         LCDnew.setCursor(0,1);
         LCDnew.print("YARKO");
     }
}
```

Лабораторная работа № 19. Вывод температуры на ЖКИ

Задача 1. Написать программу, которая будет выводить на ЖКИ сведения о температуре.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- ЖКИ экран (1 шт.);
- Терморезистор (1 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунке 1.

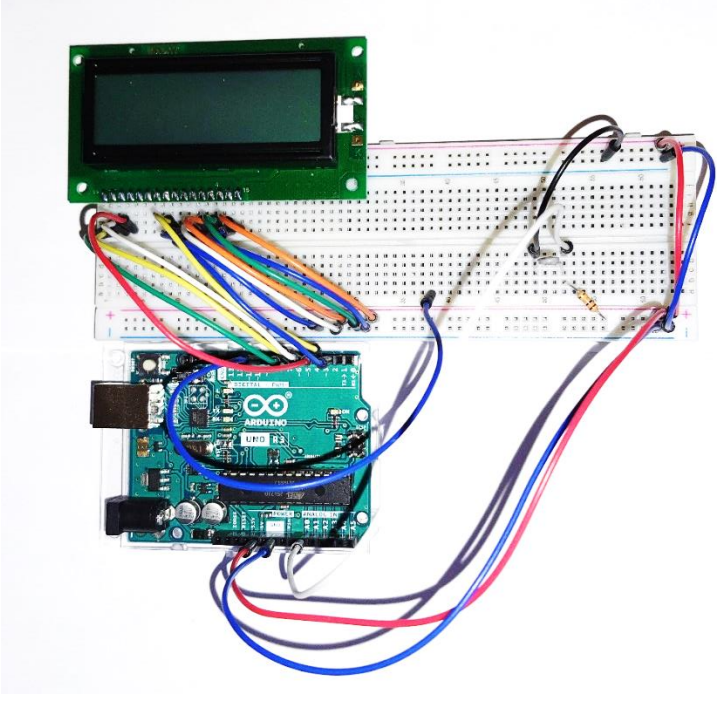


Рисунок 1

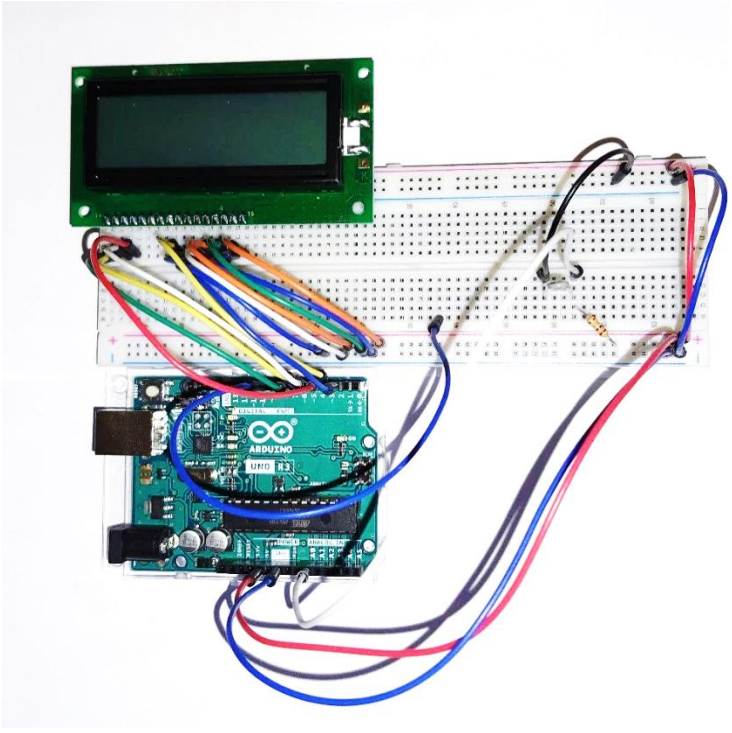


Рисунок 2

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте в схему второй терморезистор и измените код так, чтобы показания первого терморезистора выводились слева в первой строке, а второго – справа во второй строке.

2. Добавьте в схему два светодиода (по одному рядом с каждым терморезистором). Загораться должен светодиод у того терморезистора, на котором температура выше. Если терморезисторы показывают одинаковую температуру, должны загораться оба светодиода.

Листинг 1

```
/*
* Работа №19
* Вывод температуры на ЖКИ
*/

//Библиотека для работы с ЖКИ
#include <LiquidCrystal.h>

//Назначение контактов для работы с ЖКИ
#define RS 2
#define EN 3
#define D4 4
#define D5 5
#define D6 6
#define D7 7

//Контакт для терморезистора
#define TEMP A0

//Создание объекта для ЖКИ и передача ему номеров контактов платы
LiquidCrystal LCDnew(RS, EN, D4, D5, D6, D7);

//Переменная для прочитанного значения температуры
int readTemp=0;

void setup() {
     //Инициализация работы с ЖКИ, с передачей размерности экрана (16 символов в
     2 строках)
     LCDnew.begin(16,2);
}

void loop() {
     //Очистка экрана ЖКИ и перемещение указателя в позицию 0, 0; левый верхний
     угол
     LCDnew.clear();
     //Вывод в левом верхнем угле сообщения
     LCDnew.print("Temperatura");
     //Чтение значения температуры
     readTemp=analogRead(TEMP);
     //Перевод курсора на вторую строку
     LCDnew.setCursor(0,1);
     //Вывод прочитанного значения с терморезистора
```

```
      LCDnew.print(readTemp);
      //Задержка что бы было возможно прочесть информацию на ЖКИ до очистки
      delay(200);
}
```

Лабораторная работа № 20. 60-секундный таймер с выводом информации на жки

Задача 1. Написать программу, которая будет выводить на ЖКИ.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- ЖКИ экран (1 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунке 1.

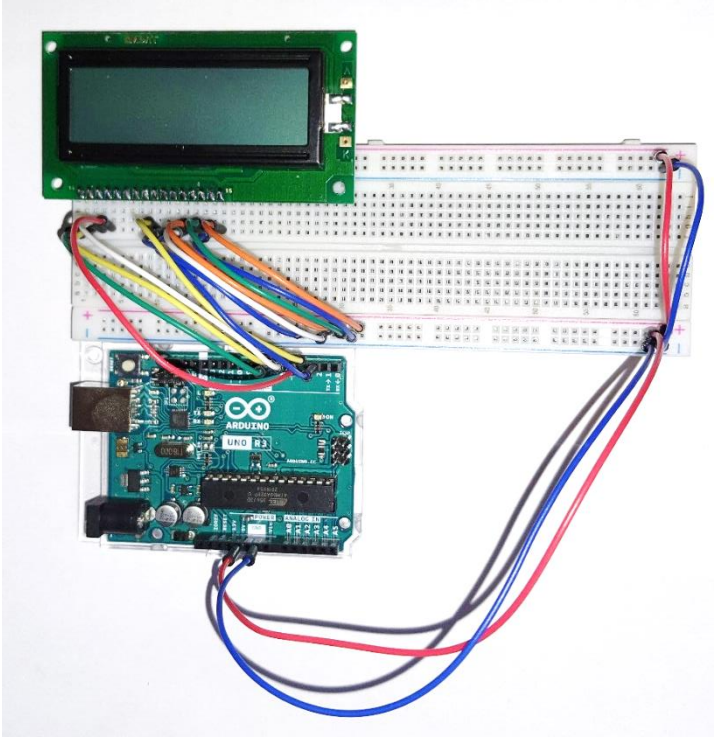


Рисунок 1

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.

4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Измените код так, чтобы одновременно с выводом в одной строке секунд с шагом в одну секунду, во второй строке выводились бы секунды с шагом 3.
2. Измените код так, чтобы в одной строке секунды выводились от 0 до 60, а в другой строке одновременно выводились бы секунды в обратном порядке.

Листинг 1

```
/*
* Работа №20
* 60-секундный таймер с выводом информации на ЖКИ
*/

//Библиотека для работы с ЖКИ
#include <LiquidCrystal.h>

//Назначение контактов для работы с ЖКИ
#define RS 2
#define EN 3
#define D4 4
#define D5 5
#define D6 6
#define D7 7

//Создание объекта для ЖКИ и передача ему номеров контактов платы
LiquidCrystal LCDnew(RS, EN, D4, D5, D6, D7);

//Переменная для подсчета секунд
int Sec = 0;

void setup() {
      //Инициализация работы с ЖКИ, с передачей размерности экрана (16 символов
      в 2 строках)
      LCDnew.begin(16,2);
      //Вывод начиная со стартовой позиции ЖКИ (0,0) сообщения в скобках
      LCDnew.print("Taimer");
}

void loop() {
```

```
        //Задержка для формирования 1 секунды
        delay(1000);

        //Увеличение переменной на +1 по прошествии 1с
        Sec++;

        //Предел счета таймера
        if (Sec==60)
        {
          // Старт счета с начала
          Sec = 0;
          //Очистка экрана ЖКИ
          LCDnew.clear();
          //Вывод надписи после очистки экрана
          LCDnew.print("Taimer");
        }

        //Перенос курсора на вторую строку
        LCDnew.setCursor(0,1);

        //Вывод значения секунд
        LCDnew.print(Sec);
}
```

Лабораторная работа № 21. Часы с будильником

Задача 1. Написать программу, при выполнении которой на ЖКИ выводится отсчёт времени и по достижении установленного срока звучит сигнал.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- ЖКИ экран (1 шт.);
- Пьезопищалка (1 шт.);
- Соединительные провода («папа-папа»);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему, как показано на Рисунке 1.

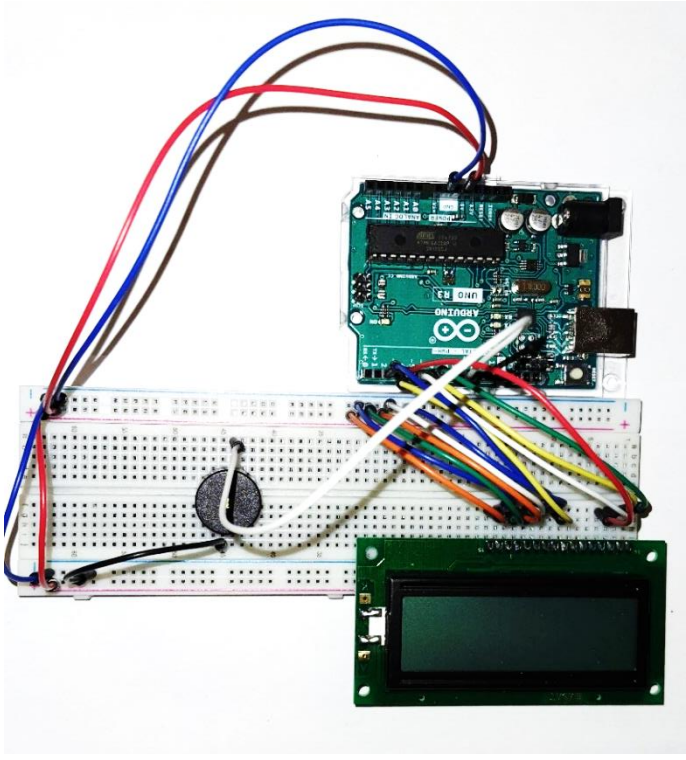


Рисунок 1

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Задания для самостоятельного выполнения

1. Добавьте в схему светодиод, который будет загораться одновременно со звуковым сигналом.

2. Измените код так, чтобы за 10 и потом за 5 минут перед срабатыванием в нижней строке выводилось бы предупреждение.

Листинг 1

```
/*
* Работа №21
* Часы с будильником
*/

//Библиотека для работы с ЖКИ
```

```
#include <LiquidCrystal.h>

//Назначение контактов для работы с ЖКИ
#define RS 2
#define EN 3
#define D4 4
#define D5 5
#define D6 6
#define D7 7

//Контакт звукового излучателя
#define SOUND 9

//Создание объекта для ЖКИ и передача ему номеров контактов платы
LiquidCrystal LCDnew(RS, EN, D4, D5, D6, D7);

//Переменные для подсчета секунд, минут, часов
int Sec = 0, Min = 0, Ch = 0;

//В них можно записать текущее значение времени перед загрузкой в плату
void setup() {

     //Инициализация работы с ЖКИ, с передачей размерности экрана (16 символов в
     2 строках)
     LCDnew.begin(16,2);

     //Вывод начиная со стартовой позиции ЖКИ (0,0) сообщения в скобках
     LCDnew.print("Chasi");
}

void loop() {

       //Задержка в 1 секуду
       delay(1000);

       //Увеличение значения переменной на +1 каждую секунду
       Sec++;

       //Если посчитано 60с увеличиваем минуты на +1
       if(Sec==60)
       {

         //Установка секунд в 0 для счета новой минуты
         Sec = 0;
```

```
    //Увеличение значения минут на +1
    Min++;

    //Если подсчитано 60мин увеличиваем часы на +1
    if(Min==60)
    {
      //Установка минут в 0 для счета нового часа
      Min = 0;
      //Увеличение значения часа на +1
      Ch++;

      //Очищаем экран и заново заполняем
      //Некоторые знакоместа на ЖКИ хранят теперь устаревшую не обновленную
      информацию
      LCDnew.clear();
      LCDnew.print("Chasi");

      //Если подсчитано 24 часа сбрасываем значение часов в 0
      if(Ch==24)
      {
        //Установка значения часов в 0
        Ch = 0;

        //Очищаем экран и заново заполняем
        //Некоторые знакоместа на ЖКИ хранят теперь устаревшую не
        обновленную информацию
        LCDnew.clear();
        LCDnew.print("Chasi");
      }
    }
  }

  //Будильник по часам и минутам
  //В скобках устанавливается значение времени для будильника

  if((Ch == 10) && (Min == 34))
  {
    //Звуковой сигнал для будильника
    tone (SOUND, 3500, 5000);
  }

  //перенос курсора в начало второй строки
  LCDnew.setCursor(0,1);
```

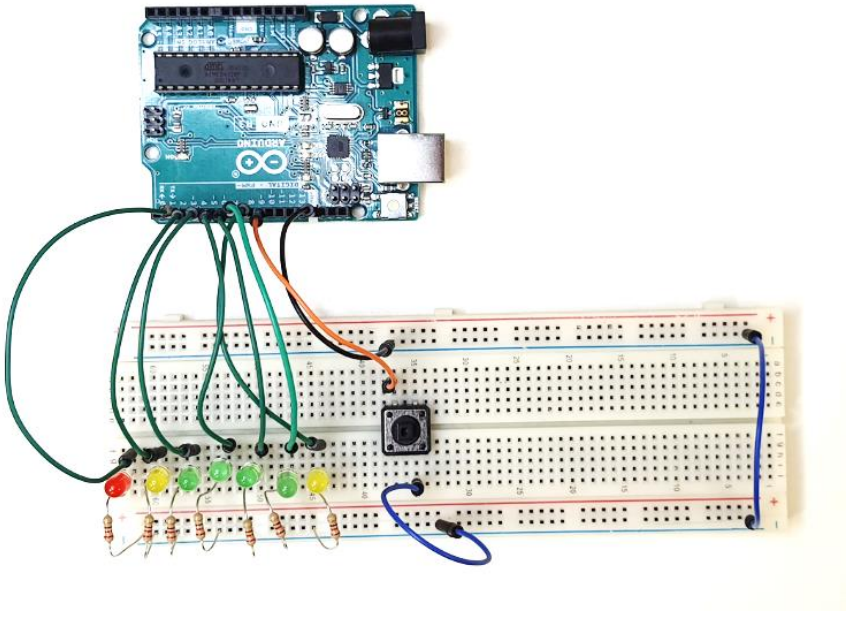
```
        //Вывод обозначения для часов
        LCDnew.print("H");

        //Вывод значения часов
        LCDnew.print(Ch);

        //Вывод обозначения для минут
        LCDnew.print("M");

        //Вывод значения минут
        LCDnew.print(Min);

        // Вывод обозначения для секунд
        LCDnew.print("S");

        //Вывод значения секунд
        LCDnew.print(Sec);
}
```

Лабораторная работа № 22. Панель индикации роутера

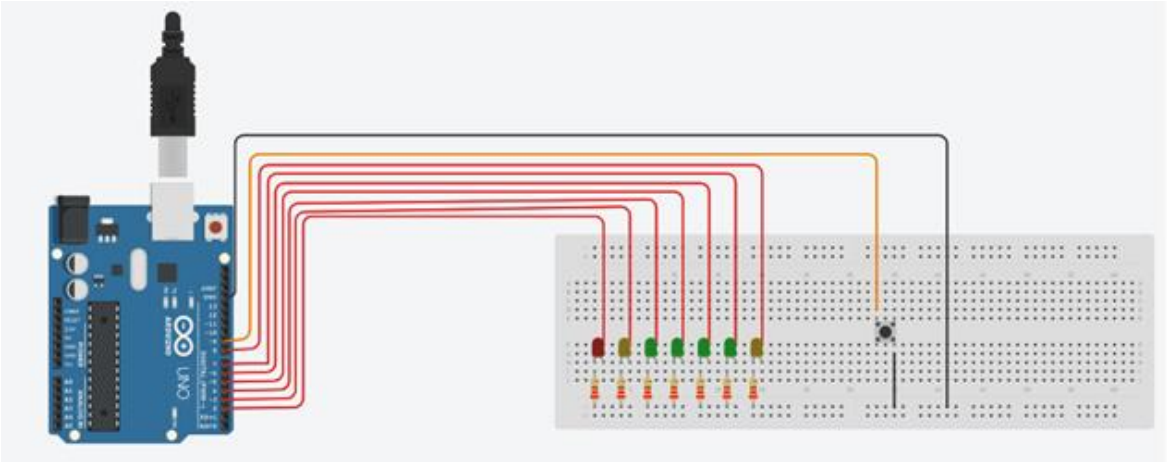
В лабораторной работе № 2 вы познакомились с платой Arduino UNO и сборкой схем на макетной плате.

В этой лабораторной работе будут использоваться те же элементы, что и раньше: светодиоды, кнопки, резисторы, соединительные провода. Но теперь уже не будет необходимости в таком подробном рассмотрении всех деталей подключения светодиодов или работы в среде Arduino IDE, так как вам это уже известно, а если вы что-то забыли, необходимо вернуться к описанию лабораторной работы № 2 и прочитать его еще раз.

Задача 1. Создать модель, имитирующую панель индикации Wi-Fi роутера. На макетной плате расположены семь светодиодов. Красный светодиод эмулирует индикатор питания. Следующий за ним желтый светодиод эмулирует индикатор мощности Wi-Fi сигнала. Четыре зеленых светодиода отвечают за индикацию подключения проводных клиентов (персональных компьютеров). Крайний правый желтый светодиод загорается на непродолжительное время при нажатии кнопки, эмулируя работу функции WPS (Wi-Fi Protected Setup – стандарт и протокол быстрого и безопасного сопряжения (соединения) двух Wi-Fi устройств между собой) (Рисунок 1).



а



б

Рисунок 1 Общий вид схемы
а) физическое исполнение; б) рисунок схемы

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Светодиоды (зеленого цвета – 4 шт., желтого цвета – 2 шт., красного цвета – 1 шт.);
- Кнопка тактовая (1 шт.);
- Резисторы 220 Ом (7 шт.);
- Соединительные провода («папа-папа», 11 шт.);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Соберите схему как показано на Рисунке 2. На рисунках 3 – 4 отдельные области подключения показаны крупнее.

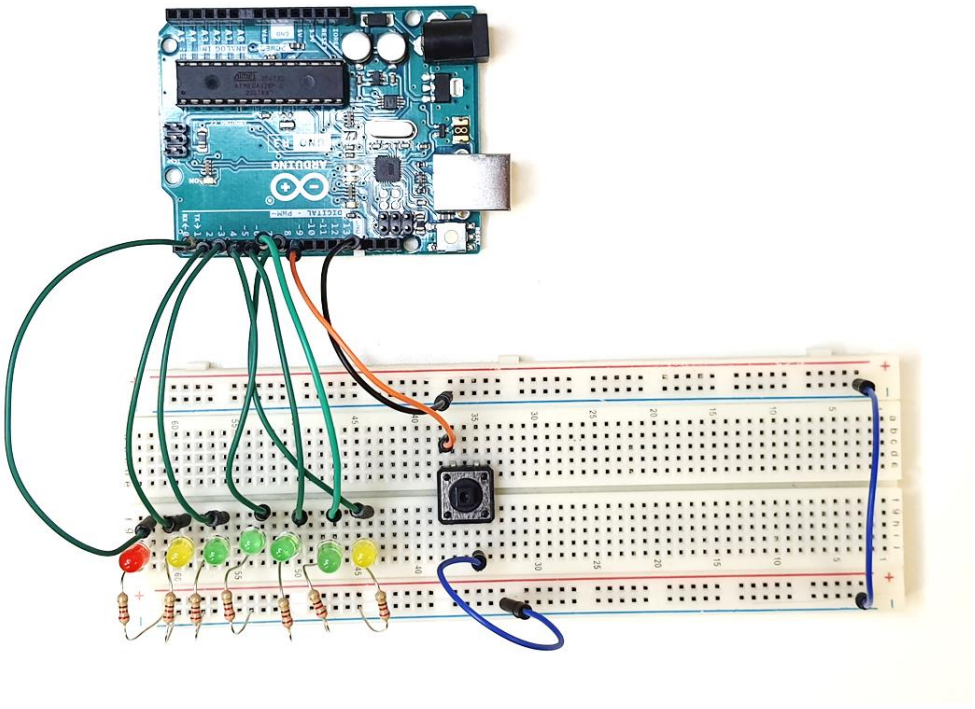


Рисунок 2 Общий вид устройства

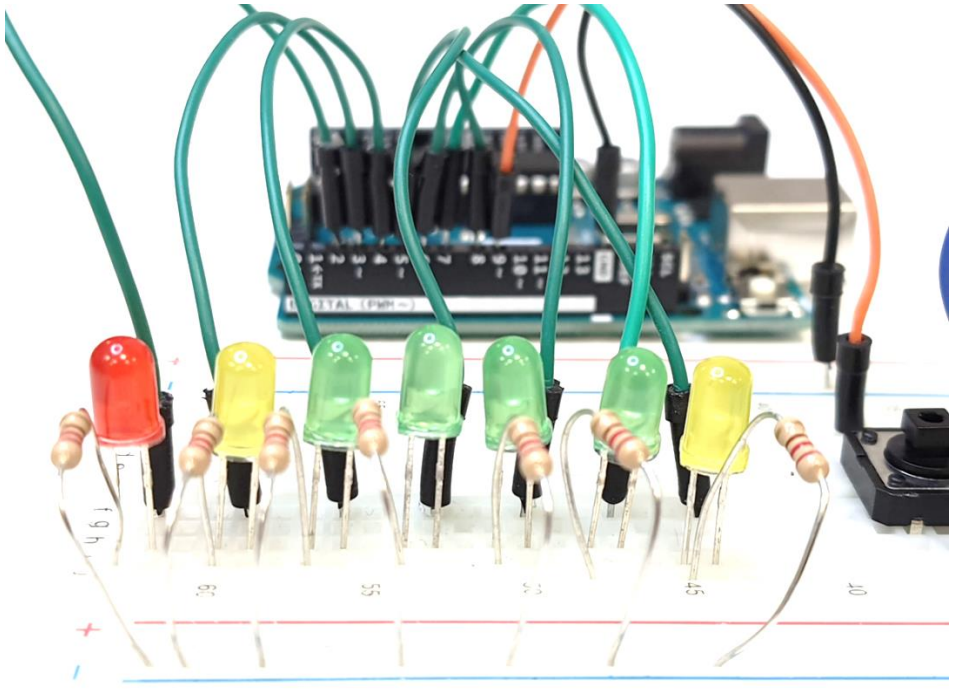


Рисунок 3 Подключение светодиодов и резисторов

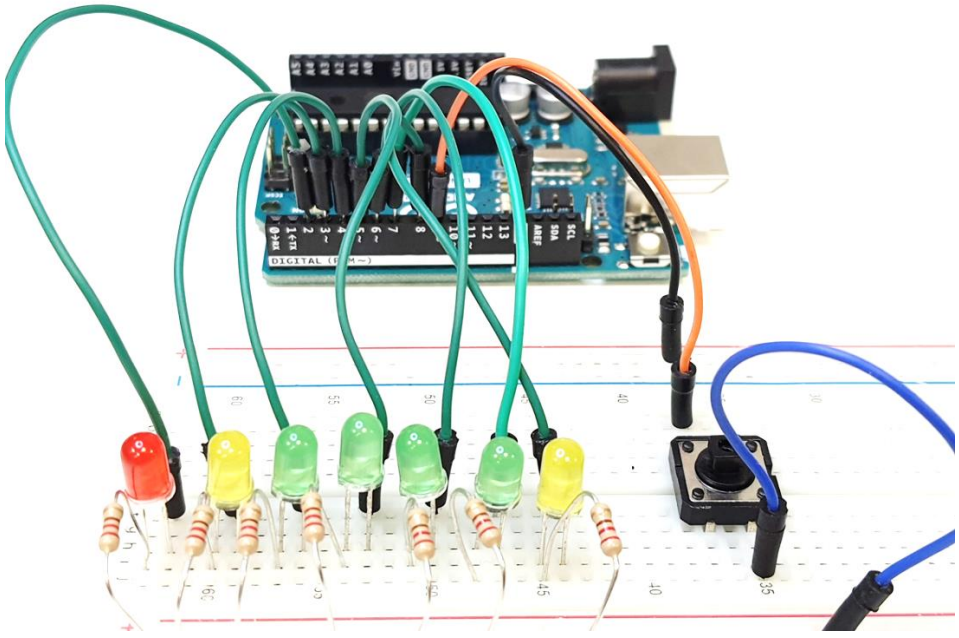


Рисунок 4 Подключение светодиодов к Arduino

2. Откройте среду Arduino IDE и наберите программный код, представленный в листинге 1. После листинга даны краткие пояснения к коду, а в самом коде есть комментарии.

Листинг 1

```
#define VD1 2 /*создание имени VD1 для цифрового вывода pin 2 платы Arduino UNO*/
#define VD2 3 //создание имени VD2 для цифрового вывода pin 3
#define VD3 4 // создание имени VD3 для цифрового вывода pin 4
#define VD4 5 // создание имени VD4 для цифрового вывода pin 5
#define VD5 6 // создание имени VD5 для цифрового вывода pin 6
#define VD6 7 // создание имени VD6 для цифрового вывода pin 7
#define WPS 8 // создание имени WPS для цифрового вывода pin 8
#define KN 9  // создание имени KN для цифрового вывода pin 9

/*создание имен для цифровых выводов при помощи директивы #define для облегчения читаемости кода, так как вместо цифр будут использоваться смысловые имена*/

/*объявление обязательной функции setup(); функция используется один раз при запуске микроконтроллера для настройки режимов работы оборудования*/
```

```
void setup()
{
     /*использование цикла для настройки цифровых выводов VD1 – VD6 и WPS в
     режим выходов*/
     for (int i=2; i<9; i++)
     {
         /*команда настройки режима работы цифрового вывода*/
         pinMode(i, OUTPUT);
     }
     pinMode(KN, INPUT_PULLUP);
 }

/*объявление обязательной функции loop(); в функции пишется основной
исполняемый код программы; тело функции повторяется бесконечно*/
void loop(){

     /*команда вывода цифрового сигнала на цифровой вывод платы*/
     digitalWrite(VD1, HIGH);

     /*создание статической локальной переменной без потери ее значений в
     итерациях; используется для формирования уровня ШИМ у индикатора уровня
     мощности сигнала Wi-Fi*/
     static int time1 = 0;

     //вывод уровня ШИМ-сигнала
     analogWrite(VD2, time1);

     /*создание статической локальной переменной q – для выбора поочередной
     одного из зеленых светодиодов, а c – для выбора режима светодиода («горит
     – не горит»)*/
     static int c=0, q=4;

     //поочередный выбор одного из зеленых светодиодов
     switch (q)
     {
       case 4:
          c = random(2);
```

```
          digitalWrite(VD3, c);
          delay(50);
          break;
       case 5:
          c = random(2);
          digitalWrite(VD4, c);
          delay(50);
          break;
       case 6:
          c = random(2);
          digitalWrite(VD5, c);
          delay(50);
          break;
       case 7:
          c = random(2);
          digitalWrite(VD6, c);
          delay(50);
          break;
       }

    /*использование функции random() для случайного выбора одного из режимов
    работы зеленых светодиодов*/

    //подсчет времени свечения сигнала WPS
    static int time2=0;
    if (!(digitalRead(KN)))
    {
        digitalWrite(WPS, HIGH);
    }

    /*определение нажата ли кнопка; если кнопка нажата, светодиод сигнала
    WPS загорается */
    q++;

    if (q==8)
    {
        q=4;
    }
```

```
    //реализация поочередного перебора зеленых светодиодов

    time1+=1;
    if (time1>254)
    {
       time1=0;
    }

    //реализация изменения уровня сигнала ШИМ
    time2+=1;
    if(time2==400)
    {
        time2=0;
        digitalWrite(WPS, LOW);
    }
    //реализация продолжительности включения светодиода индикации WPS
        }
```

Пояснения к коду

Команда `#define` вам знакома; как вы помните, она создает «текстовые имена», в данном случае для цифровых выводов со второго по девятый (строки 1 – 9).

В лабораторной работе № 26 была рассмотрена структуру скетча для Arduino (в нем обязательно должны быть функции `void setup ()` и `void loop()`.

Рассмотрим функцию `void setup()` в данной лабораторной работе:

– строки 16 – 26: в этих строках в цикле FOR с помощью команды `pinMode` производится настройка выводов со второго по девятый в режим выхода;

- строка 27: команда `pinMode(KN, INPUT_PULLUP);` настраивает цифровой контакт 9 (текстовое имя KN) для подключения кнопки с помощью параметра INPUT_PULLUP. В результате, когда кнопка не нажата проверка её состояния будет возвращать TRUE, а при нажатой кнопке проверка вернет FALSE.

Рассмотрим функцию `void loop()` в данной лабораторной работе: как вы помните, функция работает как цикл, бесконечно повторяя код своего тела.

В строке 40 создается переменная `time1`, значение которой будет последовательно изменяться (увеличиваться от 0 до 255 и по достижении максимально возможного значения (254) будет сбрасываться в 0, строки 75 – 79), что будет означать изменение уровня сигнала ШИМ (широтно-импульсной модуляции – формирование уровней напряжения в вольтах при помощи наборов цифровых сигналов), а уровень сигнала ШИМ (255 – 5В, 127 ≈ 2,5В, 0 – 0В) влияет на яркость свечения светодиода: чем больше значение переменной `time1`, тем ярче горит светодиод, и наоборот.

Значение переменной `time1` в качестве параметра передается функции `analogWrite(VD2, time1)`. Функция `analogWrite()` выдает аналоговую величину, в данном случае величину ШИМ.

В строках 49 – 55 наибольший интерес представляет функция `random(2)`. Эта функция генерирует случайное число в диапазоне от 0 до 2 (2 не входит в диапазон генерируемых чисел). Таким образом, в качестве случайных чисел будут генерироваться 0 или 1, что можно использовать как HIGH (1) и LOW (0) для функций `digitalWrite()`.

Дополнительно нужно отметить, что в операторе switch(q) переменная q используется для обращения к цифровым выходам не по текстовым именам, а при помощи номеров цифровых выходов. Это необходимо для организации поочередного обращения к цифровым контактам.

В строках 68 – 72 последовательно увеличивается значение переменной q, а при достижении ею значения 8, её значение возвращается к 4. Тем самым происходит обращение к нужным контактам через оператор `switch()`.

Задания для самостоятельного выполнения

1. В операторе `switch()` в метках case замените цифры выходных контактов на их текстовые имена (строки 4 – 7). Проверьте работу программы.
2. Измените код в строках 75 – 79 так, чтобы длительность нарастания сигнала ШИМ была увеличена.
3. Подключите дополнительный светодиод на контакт 10 и настройте вывод уровня сигнала ШИМ при помощи функции `random()`. Не забудьте добавить небольшую задержку `(delay())` после вывода уровня, чтобы визуально различить изменение яркости светодиода.

Лабораторная работа № 23. Железнодорожный переезд

В лабораторной работе № 4 продолжается знакомство со схемами, в состав которых входят светодиоды и другие знакомые вам элементы.

Задача 1. Создать модель железнодорожного переезда. На переезде предусмотрены светофоры, по два с каждой стороны железной дороги: один для автомобиля, другой для пешехода. Пешеход 1 и Машина 1 находятся слева, а Пешеход 2 и Машина 2 находятся справа. При этом, если пешеход переходит переезд, то машинам двигать запрещено. Когда Машина 1 пересекает переезд, то Машине 2 переезд запрещен, и наоборот. Если поезд приближается к переезду, то для машин и пешеходов переезд закрывается автоматически, о чем сообщает красный мигающий сигнал на всех светофорах (светодиодах). Если пешеход желает перейти переезд, он должен нажать кнопку.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Макетная плата Breadboard (800 точек);
- Светодиоды (зеленого цвета – 4 шт., желтого цвета – 2 шт., красного цвета – 4 шт.);
- Кнопка тактовая (2 шт.);
- Резисторы 220 Ом (10 шт.);
- Arduino Troyka Shield;
- Серводвигатели (2 шт.);
- Соединительные провода («папа-папа», 16 шт.);
- Кабель USB для подключения платы Arduino Uno к компьютеру.

Ход выполнения работы:

1. Выполните сборку схемы, общий вид которой представлен на рисунках 1 – 2. Детали подключения, смотрите на рисунках 3 – 10.

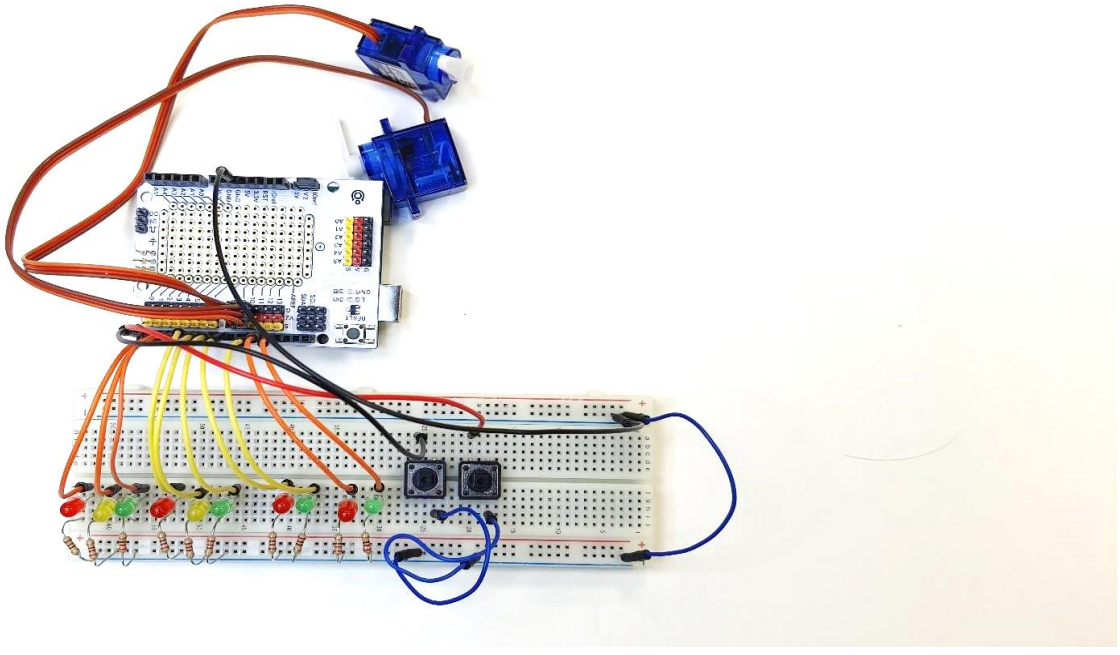


Рисунок 1

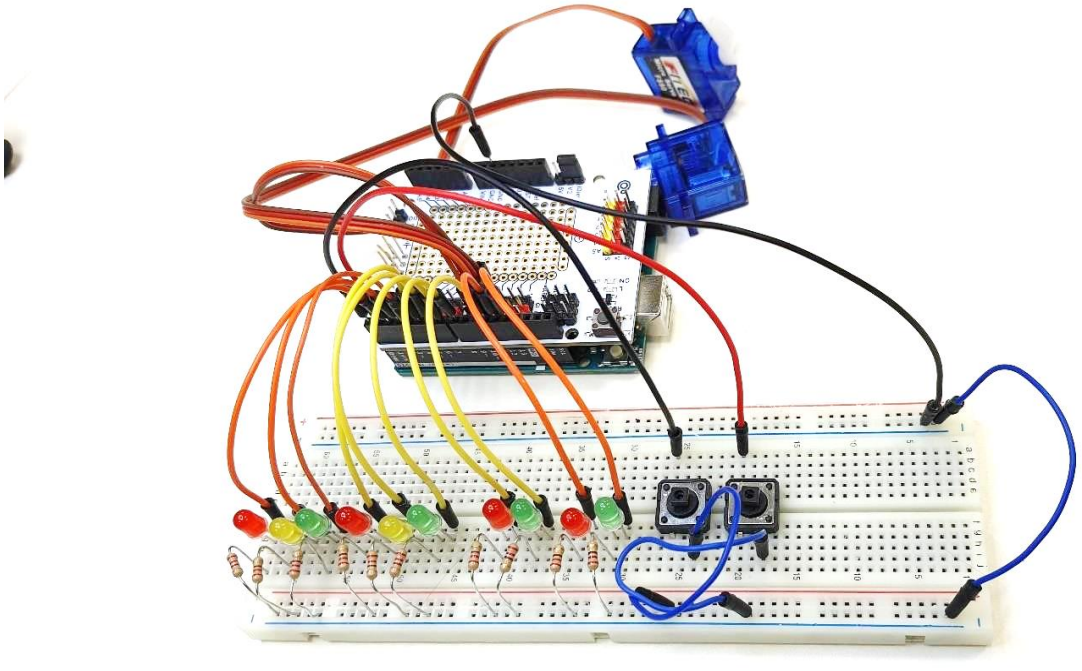


Рисунок 2

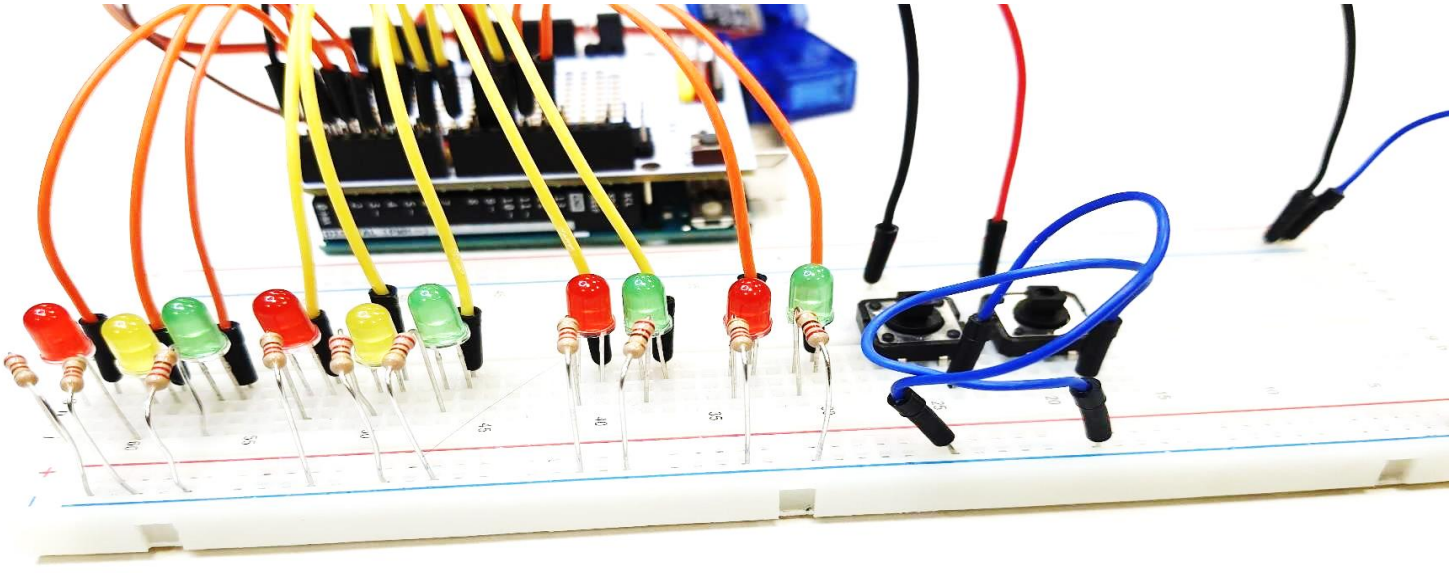


Рисунок 3

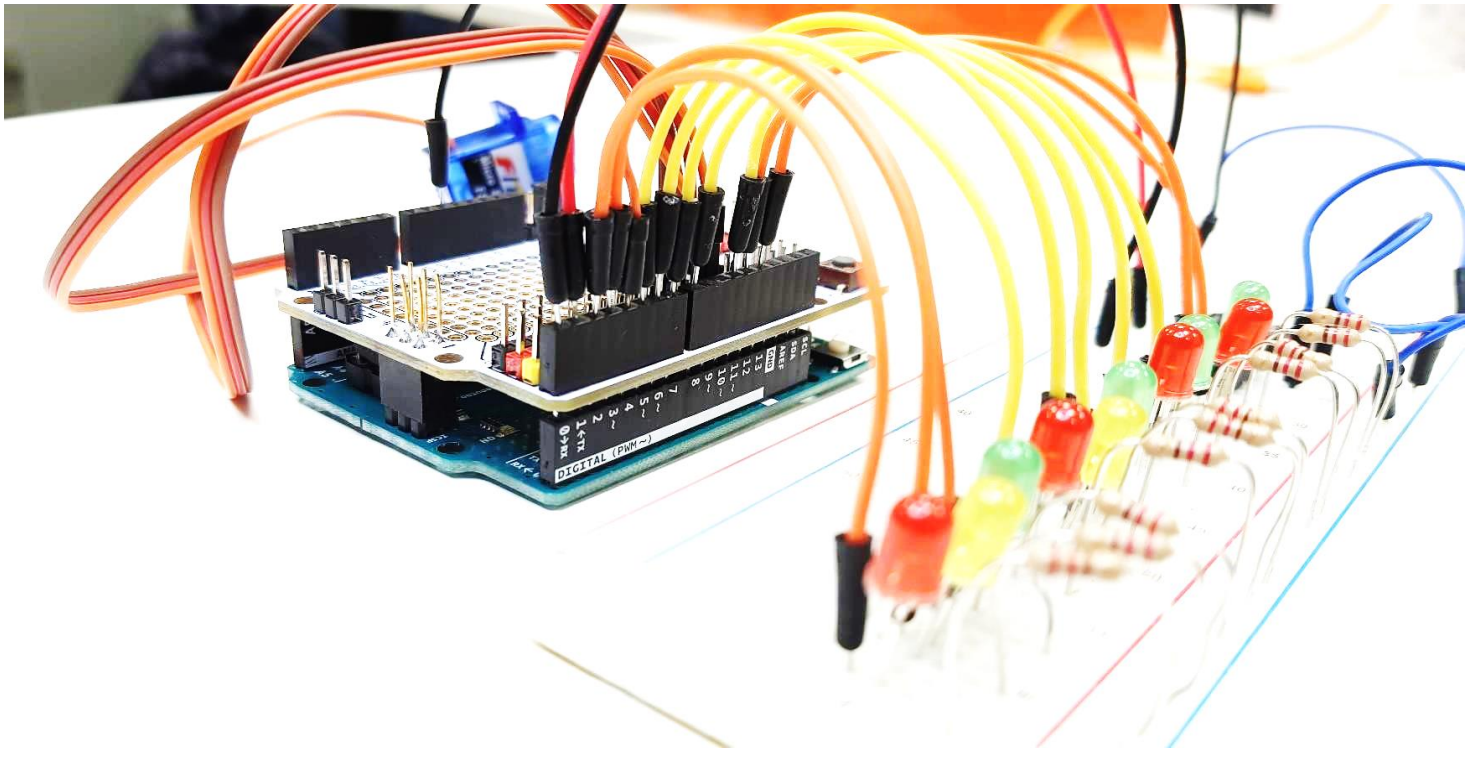


Рисунок 4

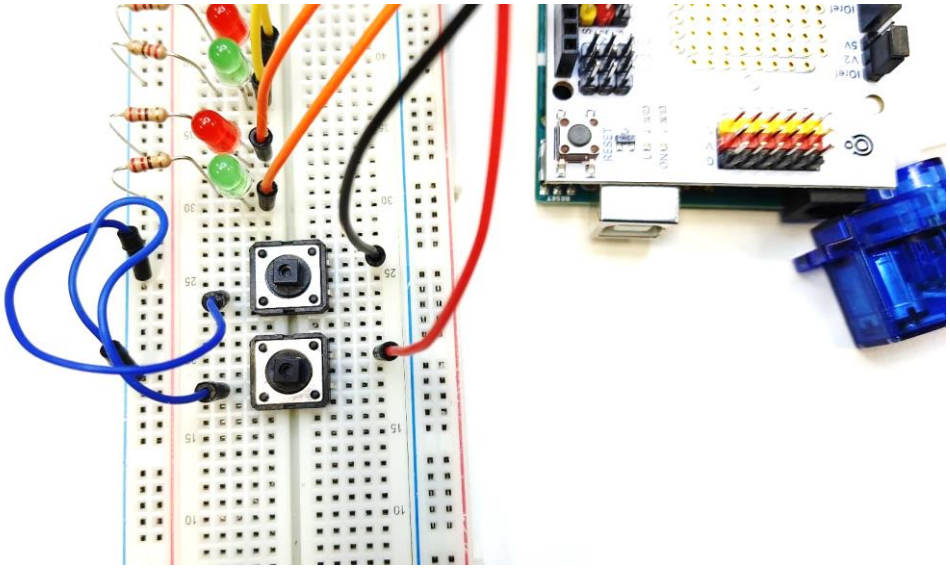


Рисунок 5

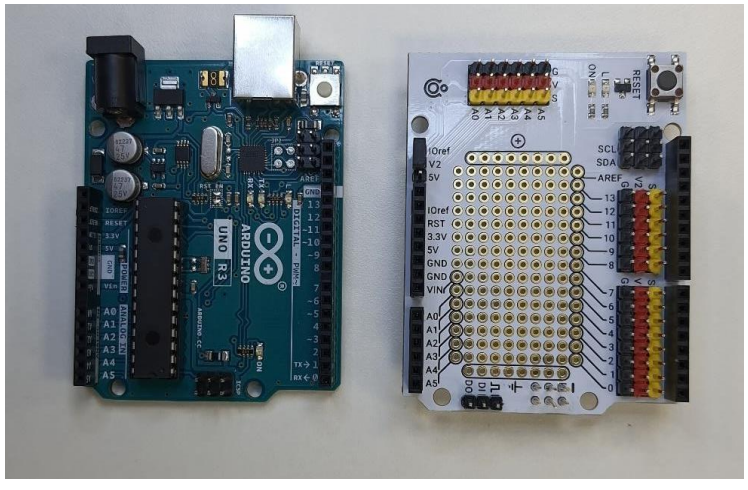


Рисунок 6

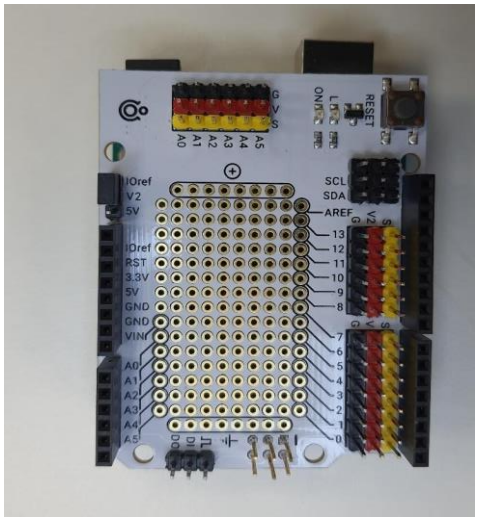


Рисунок 7

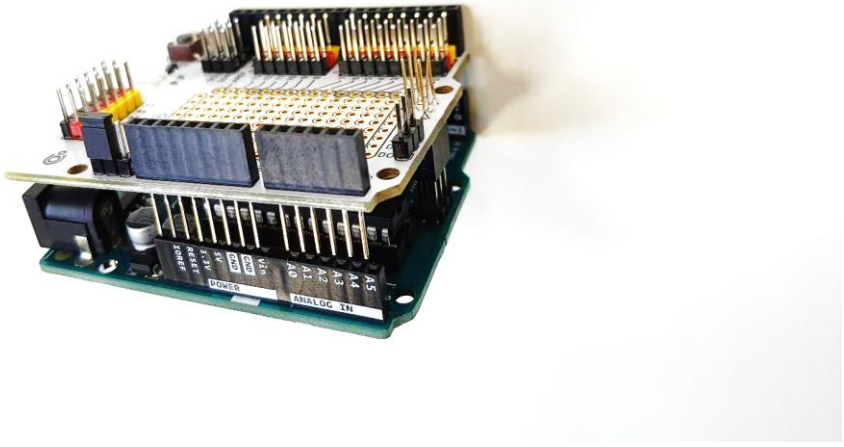


Рисунок 8

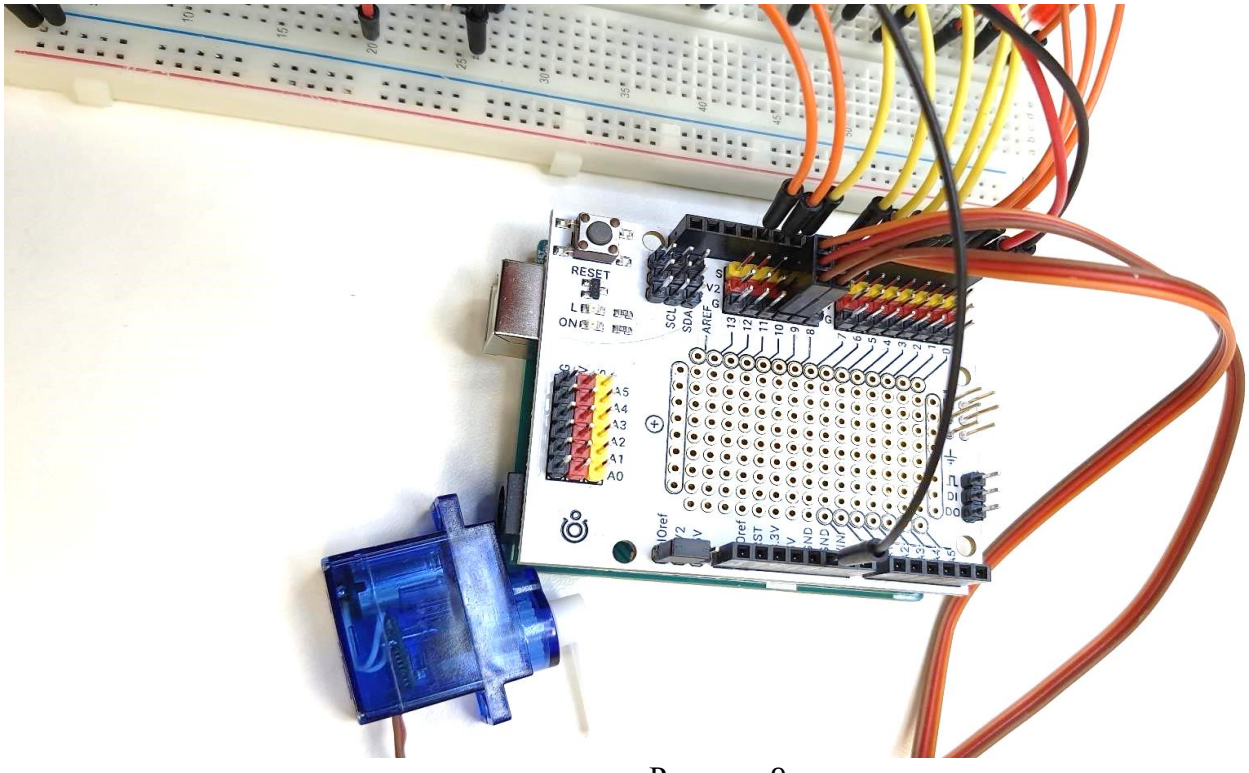


Рисунок 9

Рисунок 10

2. Откройте среду Arduino IDE и наберите код листинга 1.

Листинг 1

```
#define SF_M1K 2  //Светофор 1 для машин, светодиод красный
#define SF_M1G 3  //Светофор 1 для машин, светодиод желтый
#define SF_M1Z 4  //Светофор 1 для машин, светодиод зеленый
#define SF_M2K 5  //Светофор 2 для машин, светодиод красный
#define SF_M2G 6  //Светофор 2 для машин, светодиод красный
#define SF_M2Z 7  //Светофор 2 для машин, светодиод красный
#define SF_P1K 8  //Светофор 1 для пешеходов, светодиод красный
#define SF_P1Z 11 //Светофор 1 для пешеходов, светодиод зеленый
#define SF_P2K 12 //Светофор 2 для пешеходов, светодиод красный
#define SF_P2Z 13 //Светофор 2 для пешеходов, светодиод зеленый
#define KN_SFP1 0 //Светофор 1 для пешеходов кнопка перехода
#define KN_SFP2 1 //Светофор 2 для пешеходов кнопка перехода

#include <Servo.h>

Servo BOM1;
Servo BOM2;

boolean B_KN1_SFP1 = false;
boolean B_KN2_SFP2 = false;
boolean temp1 = 1, temp2 = 1;
boolean tempKN1 = 0, tempKN2 = 0;

void setup() {
     BOM1.attach(9);
     BOM2.attach(10);

     for (int i=2; i<14; i++)
     {
       pinMode(i, OUTPUT);
     }
     pinMode(KN_SFP1, INPUT_PULLUP);
     pinMode(KN_SFP2, INPUT_PULLUP);
}

void loop() {
     boolean poezd = random(2);
     if (poezd)
     {
       digitalWrite(SF_M1K, HIGH);
       digitalWrite(SF_M1G, LOW);
       digitalWrite(SF_M1Z, LOW);
```

```
      digitalWrite(SF_M2K, HIGH);
      digitalWrite(SF_M2G, LOW);
      digitalWrite(SF_M2Z, LOW);

      digitalWrite(SF_P1K, HIGH);
      digitalWrite(SF_P1Z, LOW);

      digitalWrite(SF_P2K, HIGH);
      digitalWrite(SF_P2Z, LOW);

      for(int j=0; j<11; j++)
      {
          if ((!digitalRead(KN_SFP1))&&(temp1))
          {
              tempKN1=1;
              temp1=0;
          }
          if ((!digitalRead(KN_SFP2))&&(temp2))
          {
              tempKN2=1;
              temp2=0;
          }
          delay(100);
      }

      BOM1.write(90);
      delay(4);
      BOM2.write(90);

      digitalWrite(SF_M1K, HIGH);
      digitalWrite(SF_M2K, HIGH);
      digitalWrite(SF_P1K, HIGH);
      digitalWrite(SF_P2K, HIGH);

      for(int j=0; j<11; j++)
      {
         if ((!digitalRead(KN_SFP1))&&(temp1))
         {
              tempKN1=1;
              temp1=0;
         }
         if ((!digitalRead(KN_SFP2))&&(temp2))
         {
```

```
            tempKN2=1;
            temp2=0;
         }
         delay(100);
      }
      digitalWrite(SF_M1K, LOW);
      digitalWrite(SF_M2K, LOW);
      digitalWrite(SF_P1K, LOW);
      digitalWrite(SF_P2K, LOW);

      for(int j=0; j<11; j++)
      {
          if ((!digitalRead(KN_SFP1))&&(temp1))
          {
             tempKN1=1;
             temp1=0;
          }
          if ((!digitalRead(KN_SFP2))&&(temp2))
          {
             tempKN2=1;
             temp2=0;
          }
          delay(100);
       }
       digitalWrite(SF_M1K, HIGH);
       digitalWrite(SF_M2K, HIGH);
       digitalWrite(SF_P1K, HIGH);
       digitalWrite(SF_P2K, HIGH);

       for(int j=0; j<11; j++)
       {
          if ((!digitalRead(KN_SFP1))&&(temp1))
          {
             tempKN1=1;
             temp1=0;
          }
          if ((!digitalRead(KN_SFP2))&&(temp2))
          {
             tempKN2=1;
             temp2=0;
          }
          delay(100);
       }
       digitalWrite(SF_M1K, LOW);
```

```
        digitalWrite(SF_M2K, LOW);
        digitalWrite(SF_P1K, LOW);
        digitalWrite(SF_P2K, LOW);

        for(int j=0; j<11; j++)
        {
            if ((!digitalRead(KN_SFP1))&&(temp1))
            {
              tempKN1=1;
              temp1=0;
            }
            if ((!digitalRead(KN_SFP2))&&(temp2))
            {
              tempKN2=1;
              temp2=0;
            }
            delay(100);
        }
        BOM1.write(0);
        delay(4);
        BOM2.write(0);
    }
    else
    {
      digitalWrite(SF_M1K, HIGH);
      digitalWrite(SF_M1G, LOW);
      digitalWrite(SF_M1Z, LOW);

      digitalWrite(SF_M2K, LOW);
      digitalWrite(SF_M2G, LOW);
      digitalWrite(SF_M2Z, HIGH);

      digitalWrite(SF_P1K, HIGH);
      digitalWrite(SF_P1Z, LOW);

      digitalWrite(SF_P2K, HIGH);
      digitalWrite(SF_P2Z, LOW);

      for(int j=0; j<11; j++)
      {
          if ((!digitalRead(KN_SFP1))&&(temp1))
          {
              tempKN1=1;
              temp1=0;
```

```
            }
            if ((!digitalRead(KN_SFP2))&&(temp2))
            {
                tempKN2=1;
                temp2=0;
            }
            delay(100);
        }
        digitalWrite(SF_M1K, LOW);
        digitalWrite(SF_M1G, HIGH);
        digitalWrite(SF_M1Z, LOW);

        digitalWrite(SF_M2K, LOW);
        digitalWrite(SF_M2G, HIGH);
        digitalWrite(SF_M2Z, LOW);

        digitalWrite(SF_P1K, HIGH);
        digitalWrite(SF_P1Z, LOW);

        digitalWrite(SF_P2K, HIGH);
        digitalWrite(SF_P2Z, LOW);

        for(int j=0; j<11; j++)
        {
           if ((!digitalRead(KN_SFP1))&&(temp1))
           {
                tempKN1=1;
                temp1=0;
           }

           if ((!digitalRead(KN_SFP2))&&(temp2))
           {
                tempKN2=1;
                temp2=0;
           }
           delay(100);
        }

        digitalWrite(SF_M1K, LOW);
        digitalWrite(SF_M1G, LOW);
        digitalWrite(SF_M1Z, HIGH);

        digitalWrite(SF_M2K, HIGH);
        digitalWrite(SF_M2G, LOW);
```

```
      digitalWrite(SF_M2Z, LOW);

      digitalWrite(SF_P1K, HIGH);
      digitalWrite(SF_P1Z, LOW);

      digitalWrite(SF_P2K, HIGH);
      digitalWrite(SF_P2Z, LOW);

      for(int j=0; j<11; j++)
      {
         if ((!digitalRead(KN_SFP1))&&(temp1))
         {
            tempKN1=1;
            temp1=0;
         }
         if ((!digitalRead(KN_SFP2))&&(temp2))
         {
            tempKN2=1;
            temp2=0;
         }
         delay(100);
      }
    }
    if(tempKN1)
    {
        digitalWrite(SF_P1K, LOW);
        digitalWrite(SF_P1Z, HIGH);

        digitalWrite(SF_P2K, LOW);
        digitalWrite(SF_P2Z, HIGH);

        digitalWrite(SF_M1K, HIGH);
        digitalWrite(SF_M1G, LOW);
        digitalWrite(SF_M1Z, LOW);

        digitalWrite(SF_M2K, HIGH);
        digitalWrite(SF_M2G, LOW);
        digitalWrite(SF_M2Z, LOW);

        delay(4000);
        temp1=1;
        tempKN1=0;
    }
    if(tempKN2)
```

```
      {
         digitalWrite(SF_P1K, LOW);
         digitalWrite(SF_P1Z, HIGH);

         digitalWrite(SF_P2K, LOW);
         digitalWrite(SF_P2Z, HIGH);

         digitalWrite(SF_M1K, HIGH);
         digitalWrite(SF_M1G, LOW);
         digitalWrite(SF_M1Z, LOW);

         digitalWrite(SF_M2K, HIGH);
         digitalWrite(SF_M2G, LOW);
         digitalWrite(SF_M2Z, LOW);

         delay(4000);
         temp2=1;
         tempKN2=0;
     }
}
```

Пояснения к коду

Строки 1 – 12: как и в предыдущих работах создаем «текстовые имена» для цифровых выводов (см. комментарии в листинге 1).

Строка 14: подключение библиотеки Servo.h. В этой библиотеке содержатся функции для управления сервоприводами. Как правило, эта библиотека устанавливается со средой Arduino IDE. Подключить её можно с помощью меню Скетч – Подключить библиотеку (Рисунок 11).

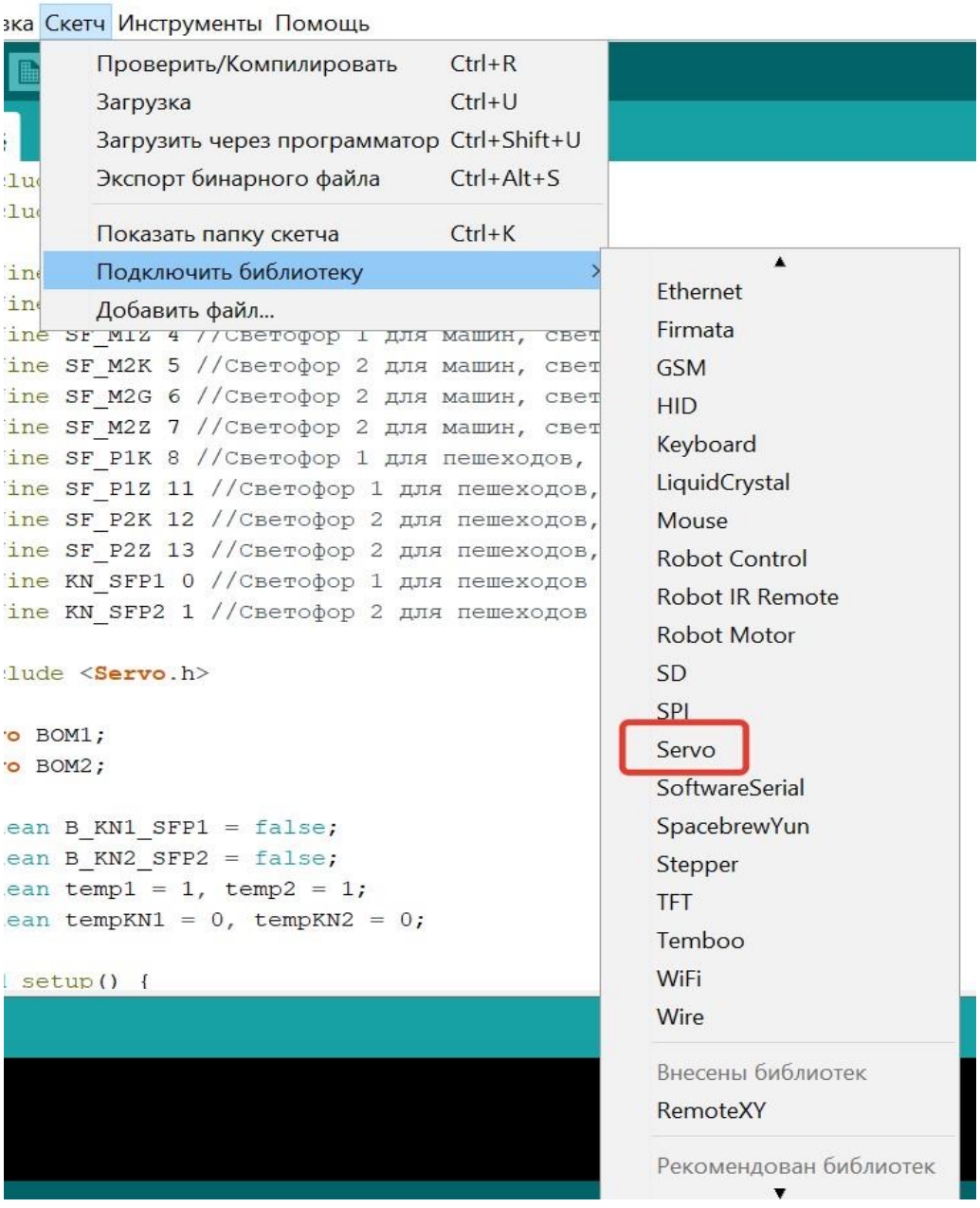


Рисунок 11

Строки 16 – 17: объявляем переменные `BOM1` и `BOM2` типа `Servo`. `BOM1` и `BOM2` – это имена для серводвигателей.

Строки 19 – 22: объявляем переменные типа `boolean` для определения состояния кнопок: нажата или нет (по условию задачи кнопки нажимаются пешеходом, если он желает перейти дорогу).

Строки 24 – 35: в них в функции `void setup()` выполняется предварительная настройка аппаратуры;

- строки 25 – 26: это указание управляющих выводов, к которым подключены сервоприводы;
- строки 28 – 31: в цикле происходит настройка работы цифровых выводов со 2-го по 14-й в режим выходов;
- строки 32 – 33: это уже знакомая вам настройка цифровых контактов для подключения кнопки с помощью параметра `INPUT_PULLUP` (см. лабораторную работу № 3).

Переходим к функции `void loop()`.

Строка 38: переменной `poezd` присваиваем значение случайного числа, которое генерирует функция `random()`: если сгенерирована 1, то поезд «едет через переезд», а если 0, то переезд свободен от поезда.

Строка 39 – проверка значения переменной `poezd`;

Строки 40 – 155: формирование действий, когда поезд проезжает переезд:

Строки 41 – 53: индикация сигналов о закрытии всего движения на переезде;

Строки 55 – 68: повторение цикла `FOR` 10 раз, что при использовании в 67 строке `delay (100)` в сумме дает задержку в мигании светодиодов на переезде в 1 секунду; данный блок устроен таким образом, потому что в процессе мигания светодиодов можно потерять нажатие кнопки, пока исполняется задержка в 1 секунду;

Строка 57: двойное условие для проверки нажатия кнопки: `((!digitalRead(KN_SFP1))&&(temp1))`. Первая скобка в условии проверяет нажатие кнопки пешеходом, а вторая проверяет состояние булевой переменной; если `temp1` равна 1, то кнопка еще не нажималась, а значит, текущее нажатие нужно обработать; если `temp1` равна 0, то кнопка уже была нажата, и действия для этого нажатия еще не производились и текущее нажатие должно быть проигнорировано.

Строки 59 – 60: переменной `tempKN1` присваивается 1, чтобы сообщить операторам обработки нажатия кнопки, что кнопка была нажата; переменной temp0 присваивается значение 0, чтобы заблокировать обработку последующих нажатий.

Строки 62 – 66: повторяют строки 57 – 61, но уже для второй кнопки.

Строки 69 и 71: установка выходного вала серводвигателя в положение 90 градусов; Строка 70: время на поворот вала серводвигателя;

Строки 73 – 76: включение красных сигналов светофоров (красных светодиодов);

Строки 78 – 91: задержка в 1 секунду с опросом нажатия кнопка пешеходом каждые 100 миллисекунд;

Строки 93 – 96: выключение красных сигналов светофоров (красных светодиодов);

Строки 98 – 111: задержка в 1 секунду с опросом нажатия кнопка пешеходом каждые 100 миллисекунд;

Строки 113 – 151: повторение мигания красными светодиодами для иллюстрации мигающего красного сигнала при проезде поезда;

Строки 152 и 154: поворот выходного вала серводвигателя в 0 градусов.

Строки 156 – 245: иллюстрация работы переезда в отсутствие поезда;

Строки 246 – 265: закрытие переезда на 4 секунды, реализация обработки нажатия кнопки 1 пешеходом 1;

Строки 266 – 287: закрытие переезда на 4 секунды, реализация обработки нажатия кнопки 2 пешеходом 2;

Задания для самостоятельного выполнения

1. Измените код листинга 1 таким образом, чтобы при нажатии кнопки пешеходом, зеленый сигнал для него формировался только при отсутствии поезда на переезде. В текущей версии проверки, есть ли поезд, при разрешении движения пешеходам, не происходит.
2. Измените код светофоров для машин так, чтобы желтый светодиод мигал четыре раза, а не горел постоянно.
3. Измените код так, чтобы зеленый сигнал для пешеходов загорался только в случае, когда были одна за другой нажаты обе кнопки.

Лабораторная работа № 24. Знакомство с простейшим роботом на arduino

Задача 1. Создать простейшего робота на Arduino.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Кабель USB для подключения платы Arduino Uno к компьютеру;
- Комплектующие для сборки робота в образовательном наборе «Амперка».

Ход выполнения работы:

1. Соберите робота последовательно, как показано на Рисунках 1 – 20.

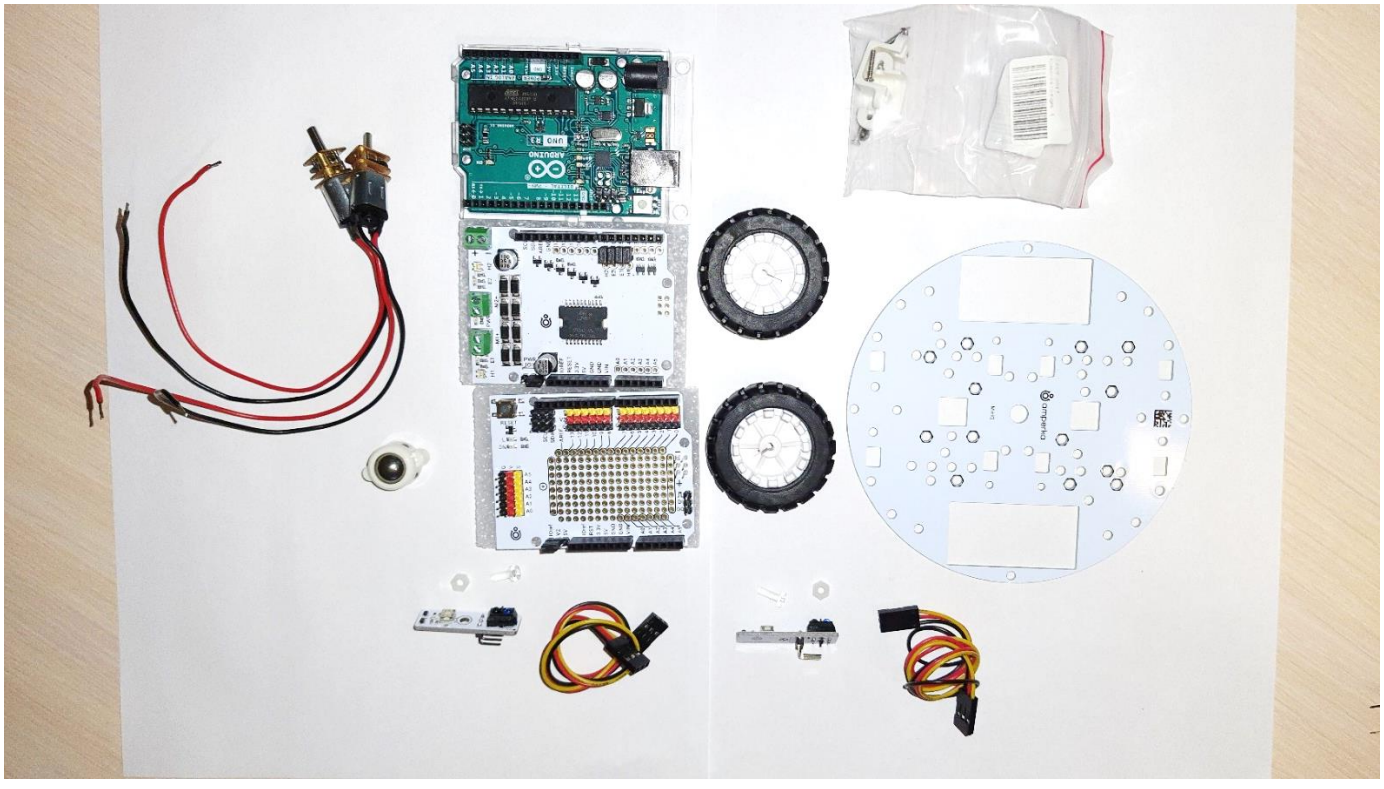


Рисунок 1

Рисунок 2

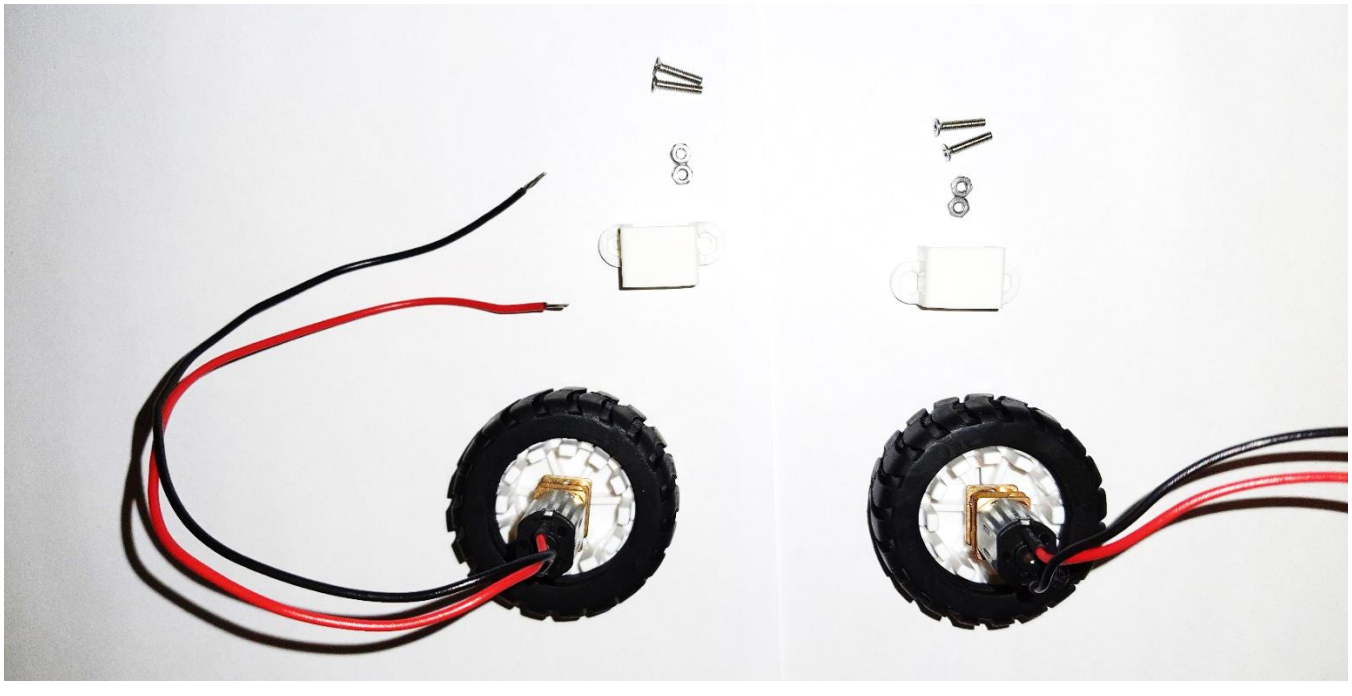


Рисунок 3

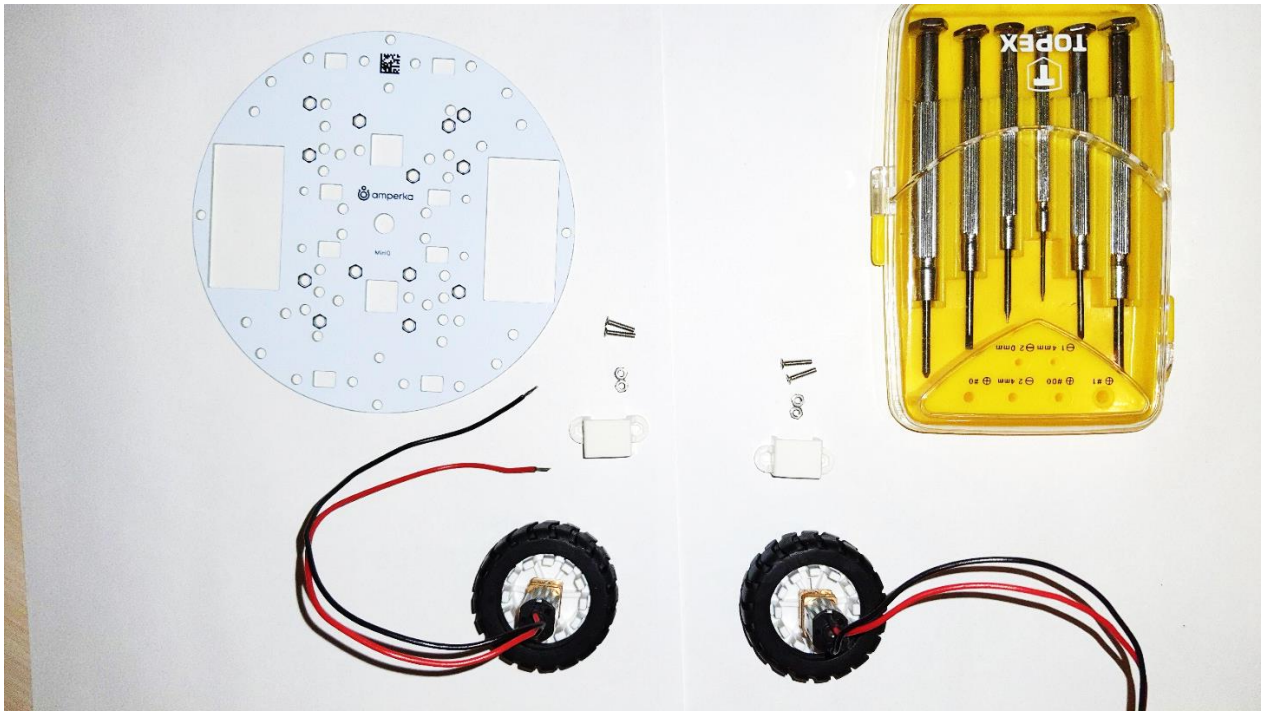


Рисунок 4

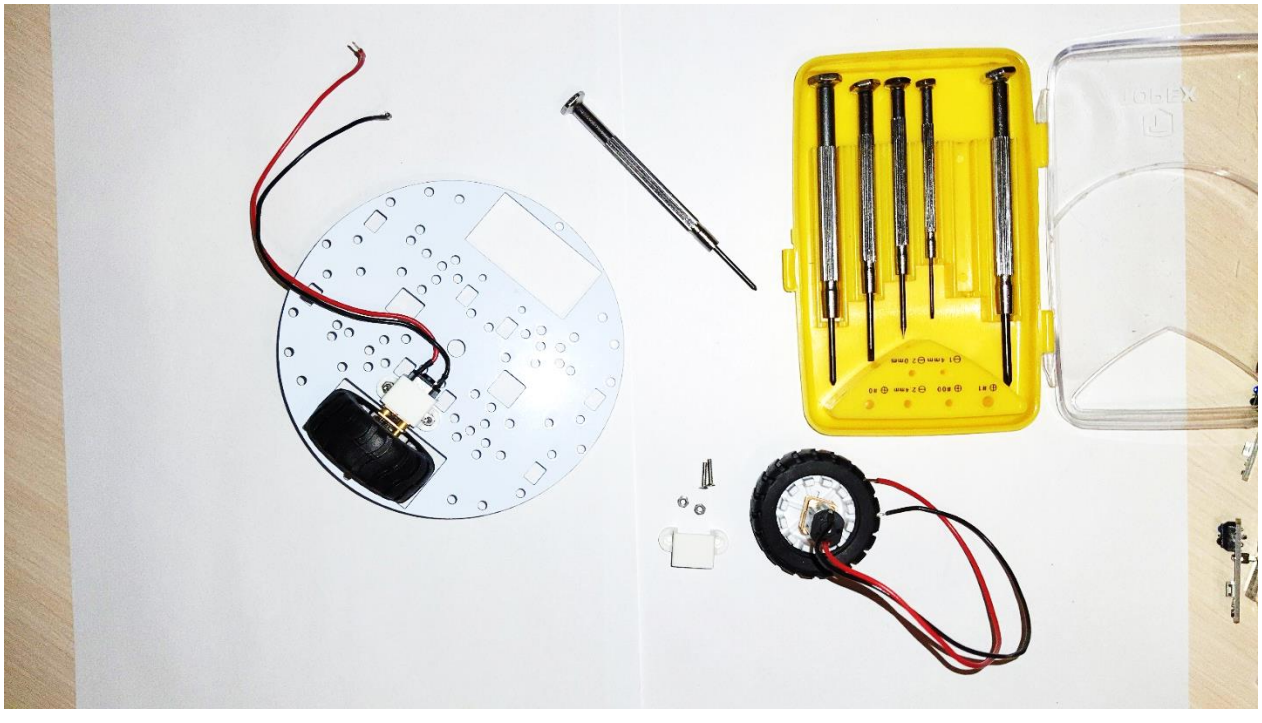


Рисунок 5

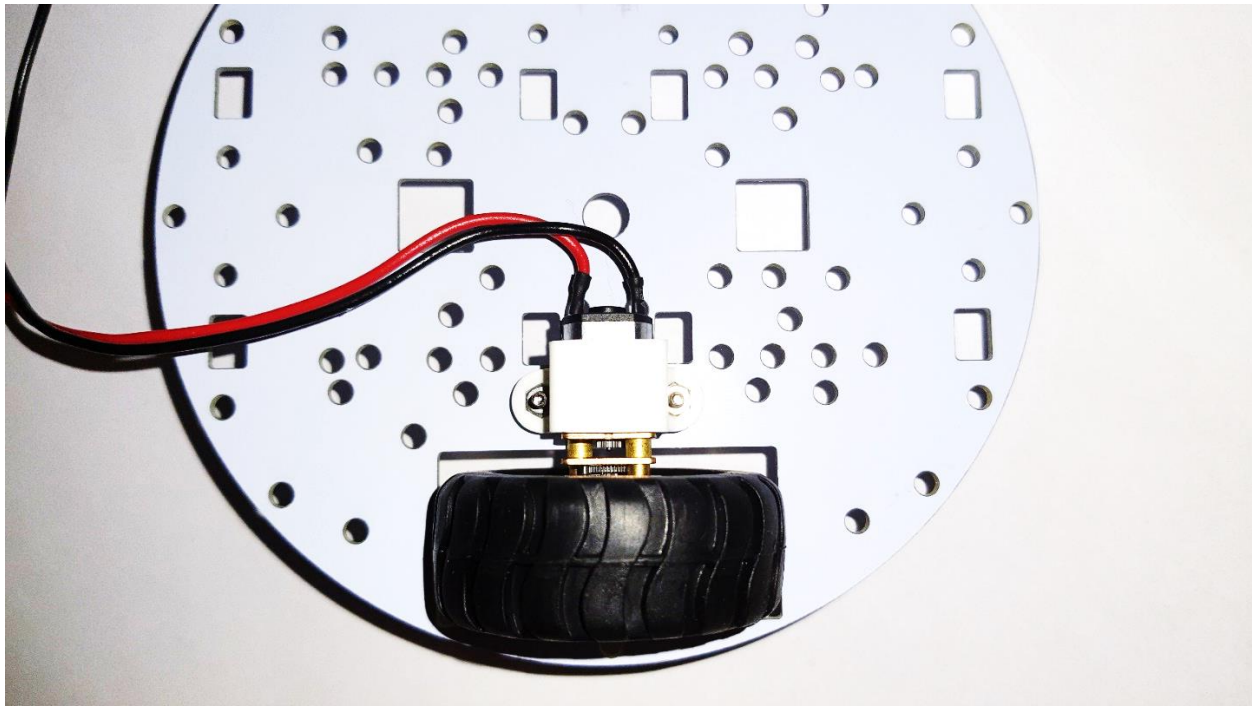


Рисунок 6

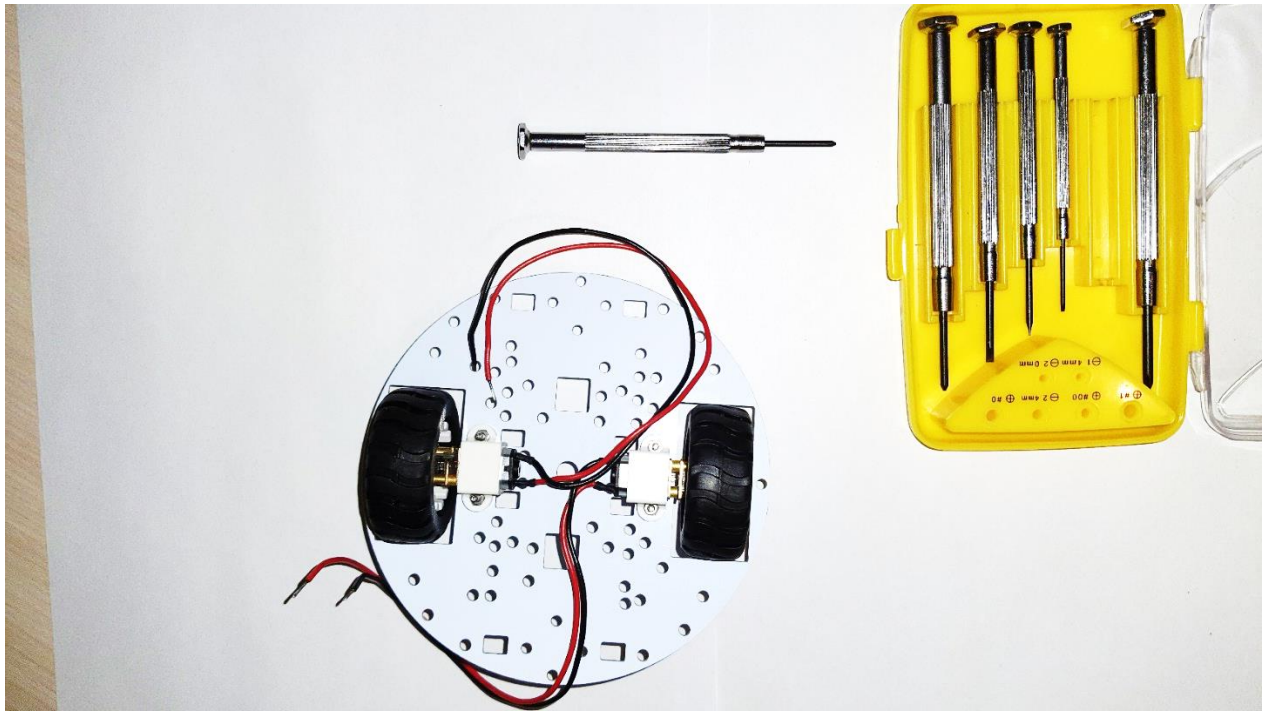


Рисунок 7

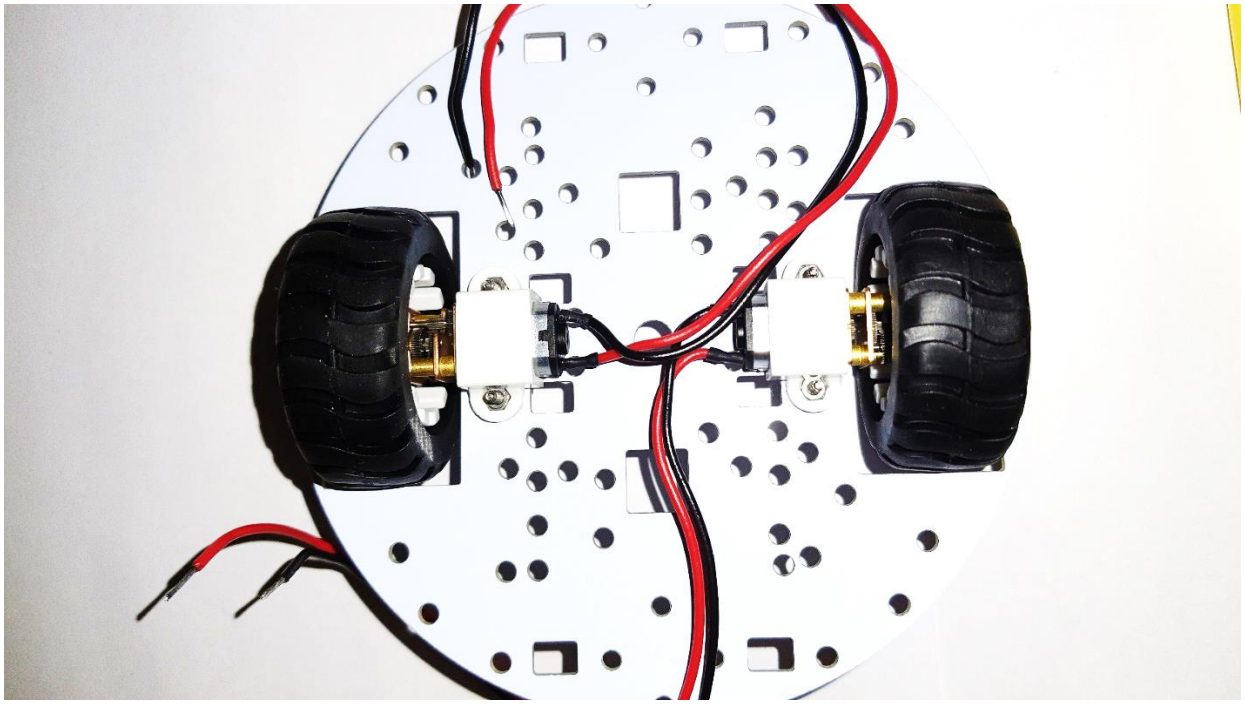


Рисунок 8

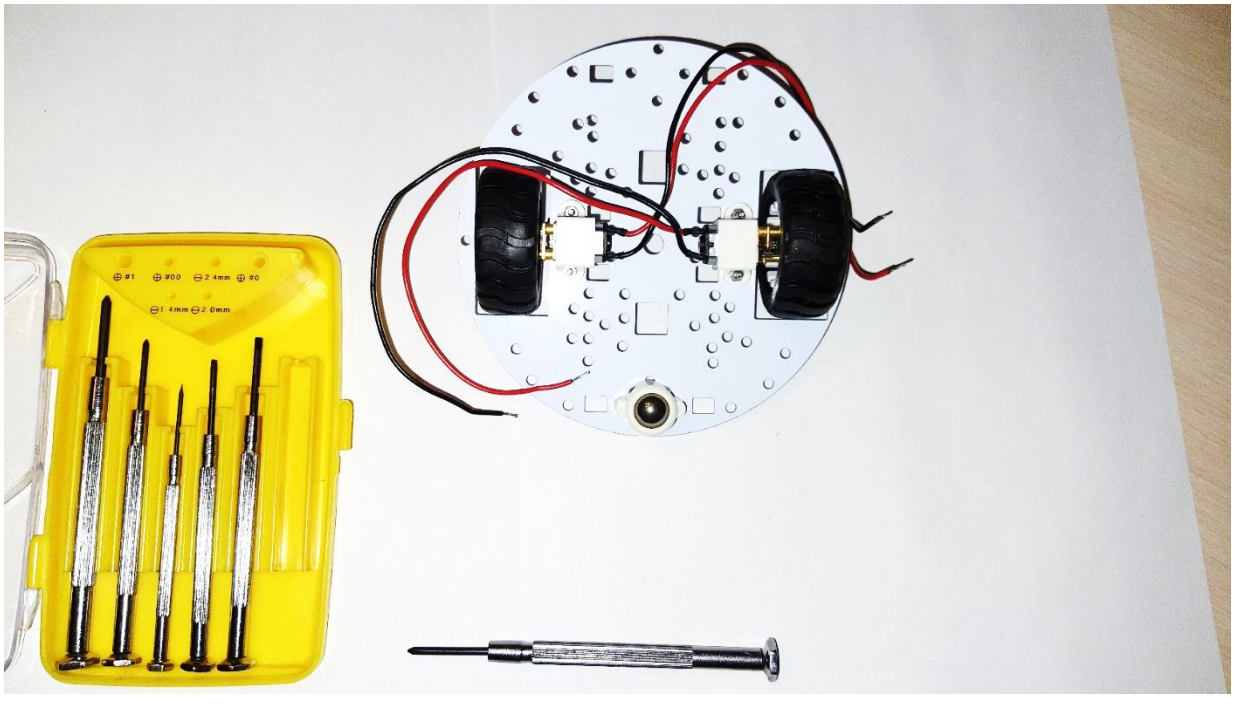


Рисунок 9

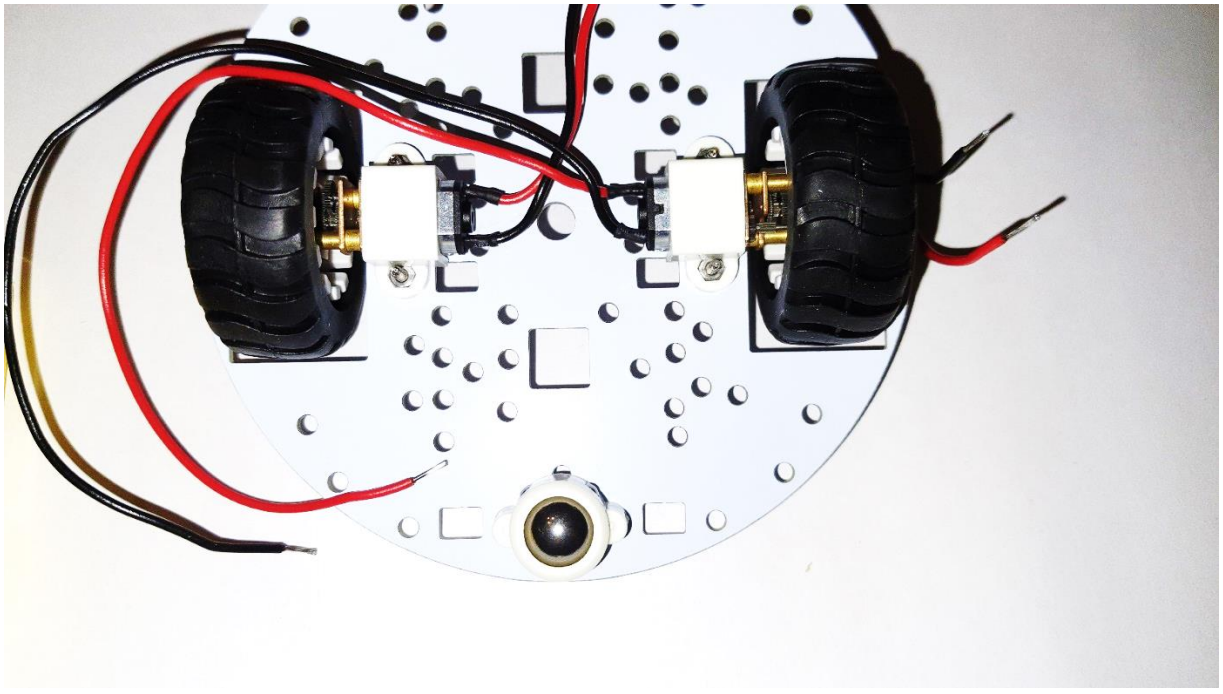


Рисунок 10

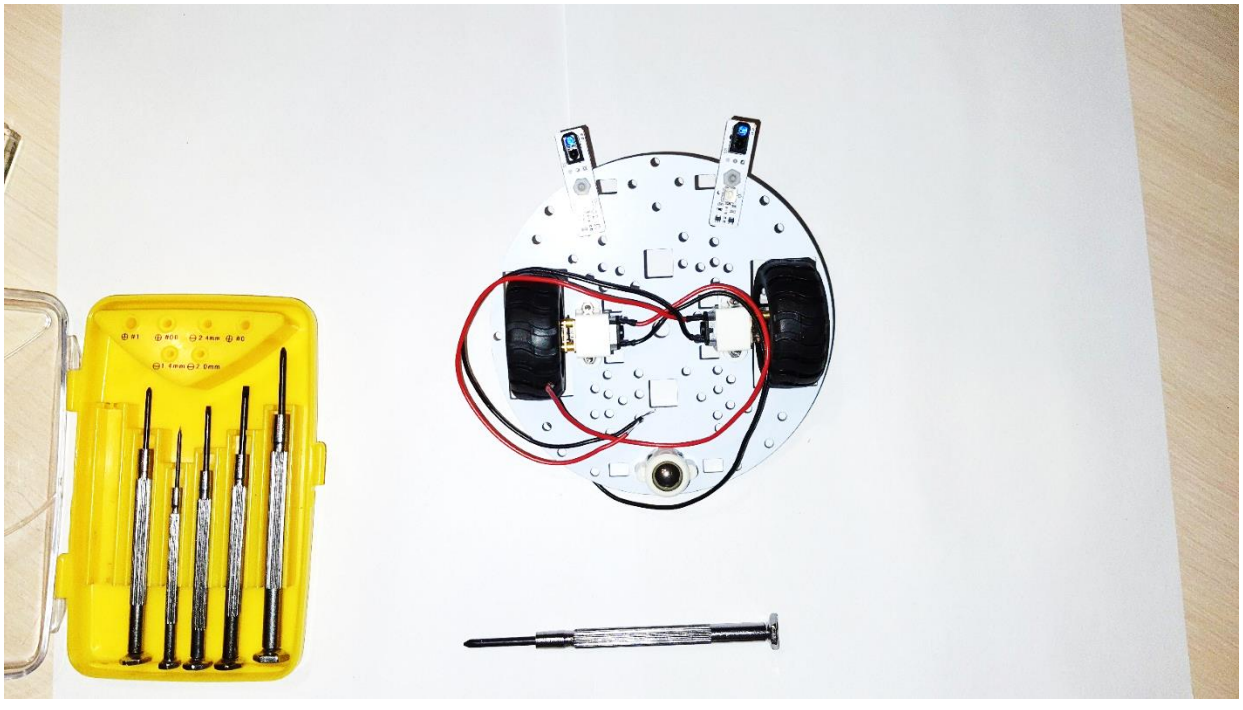


Рисунок 11

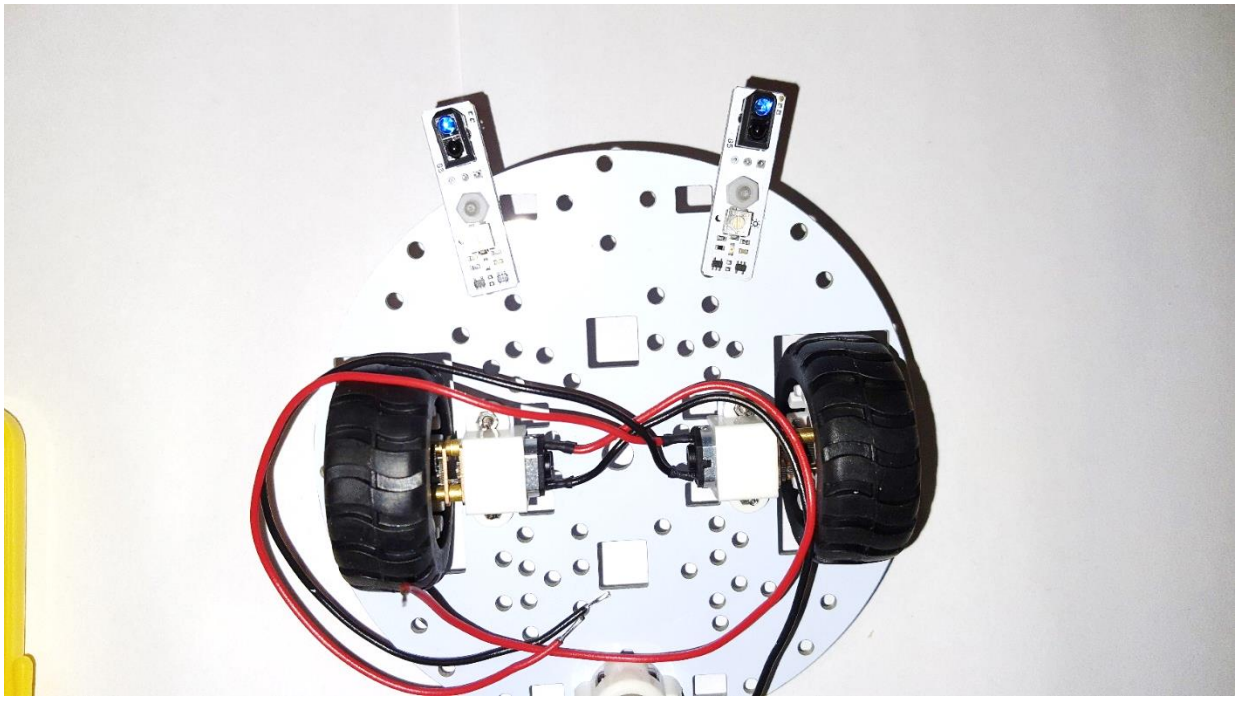


Рисунок 12

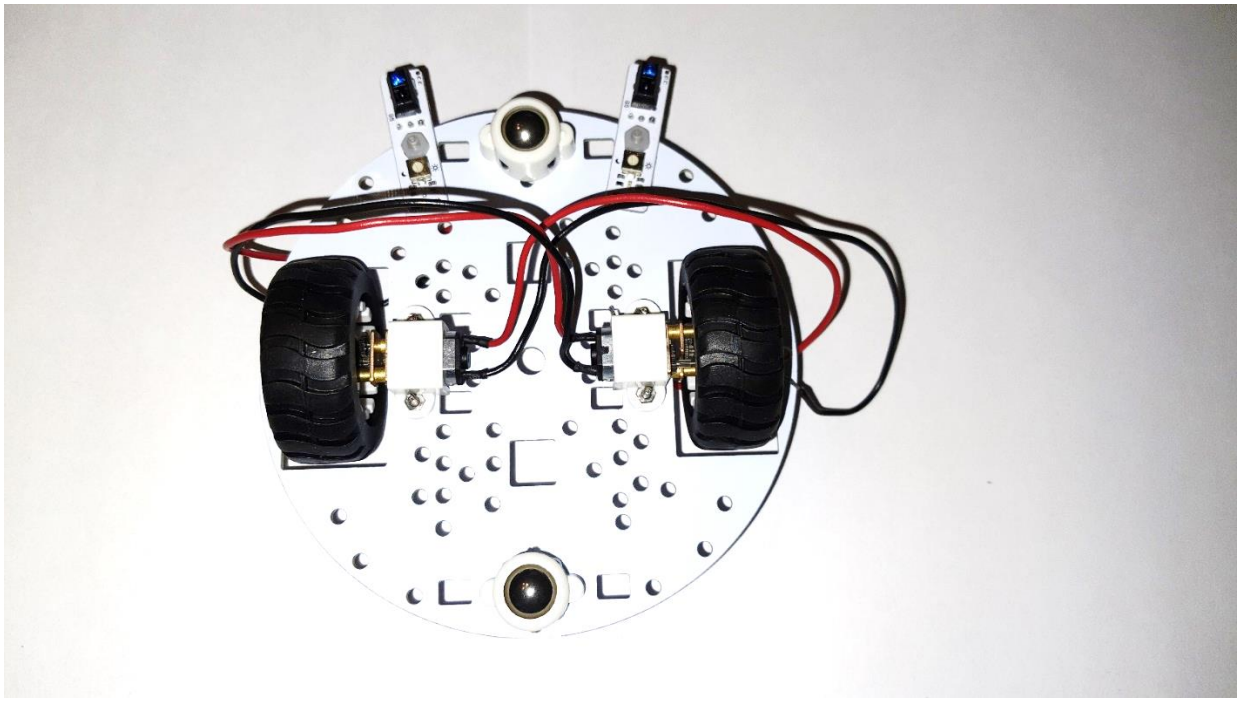


Рисунок 13

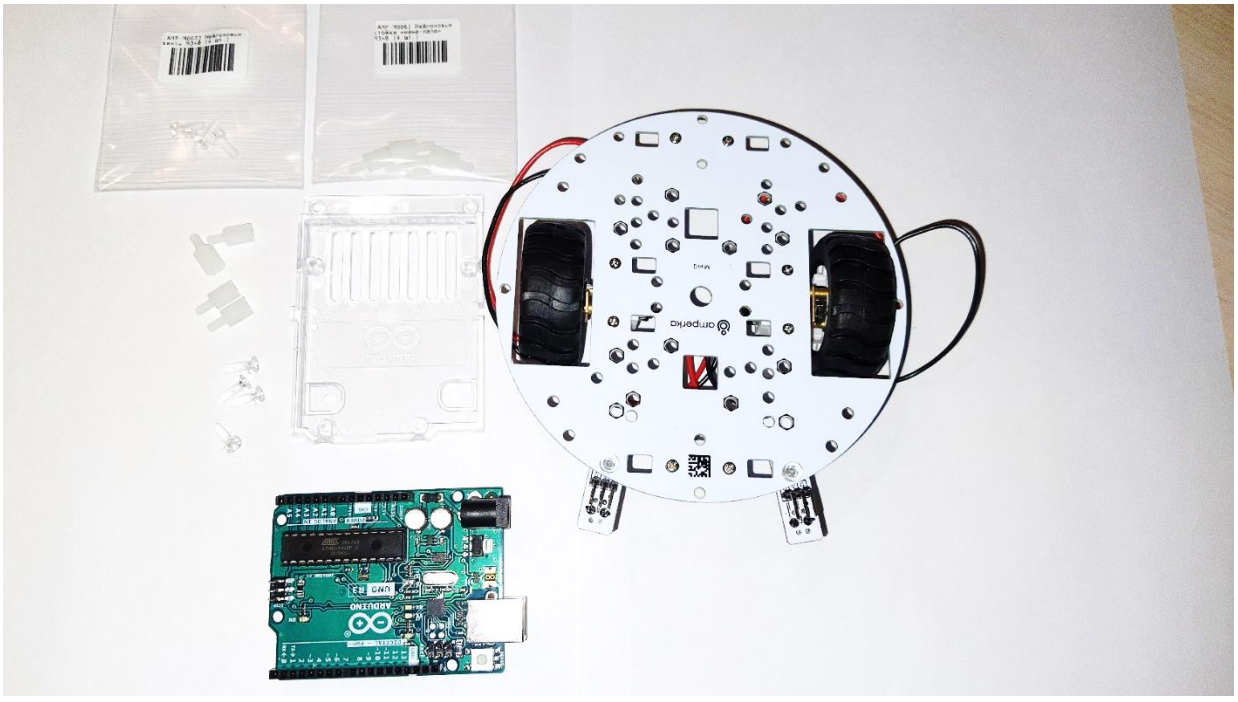


Рисунок 14

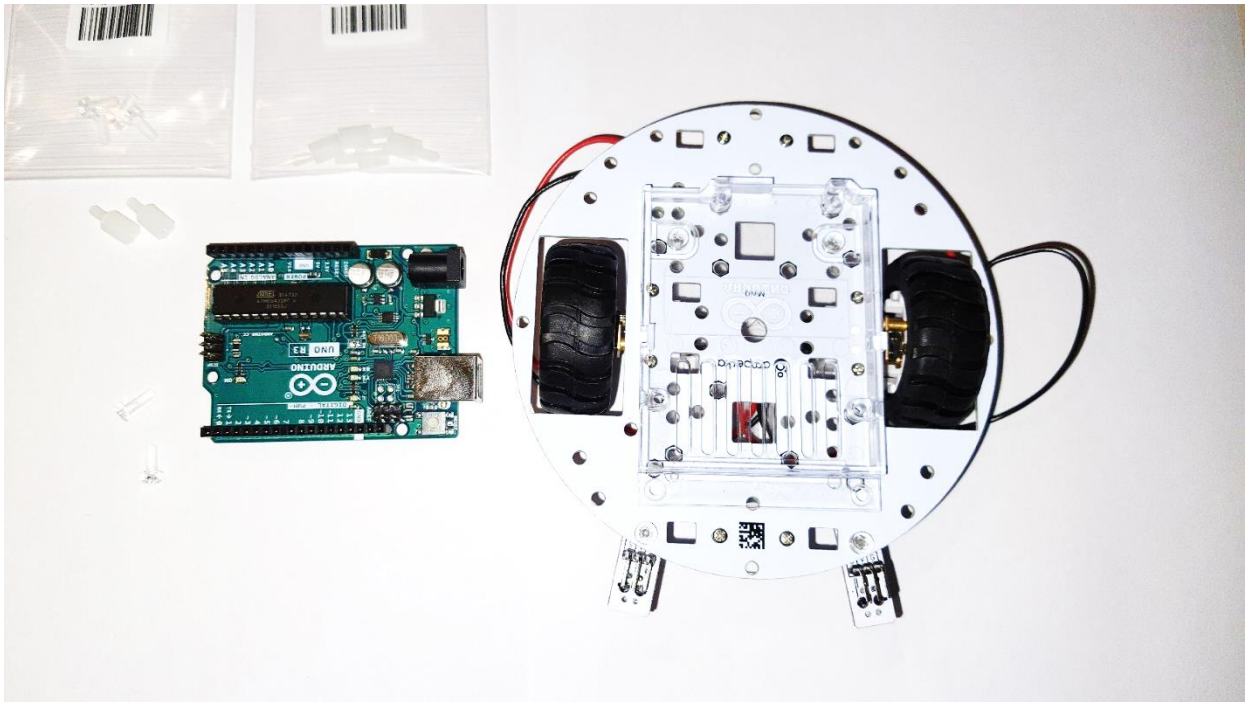


Рисунок 15

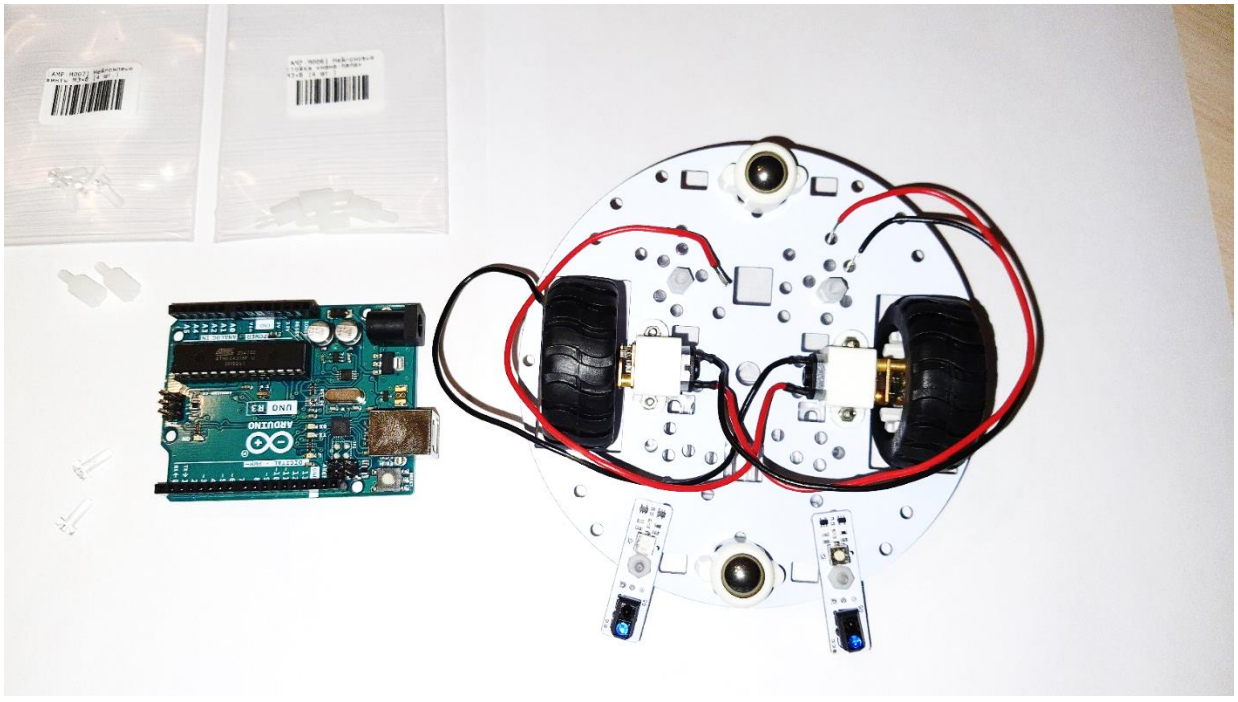


Рисунок 16

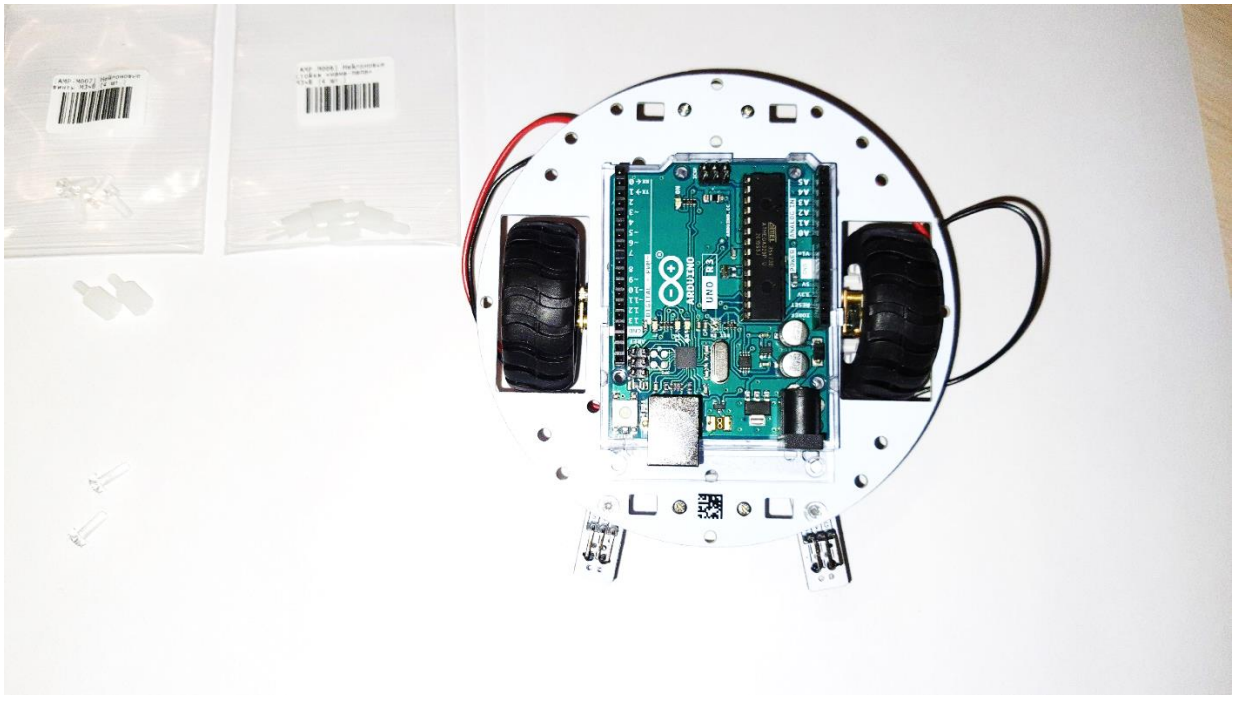


Рисунок 17

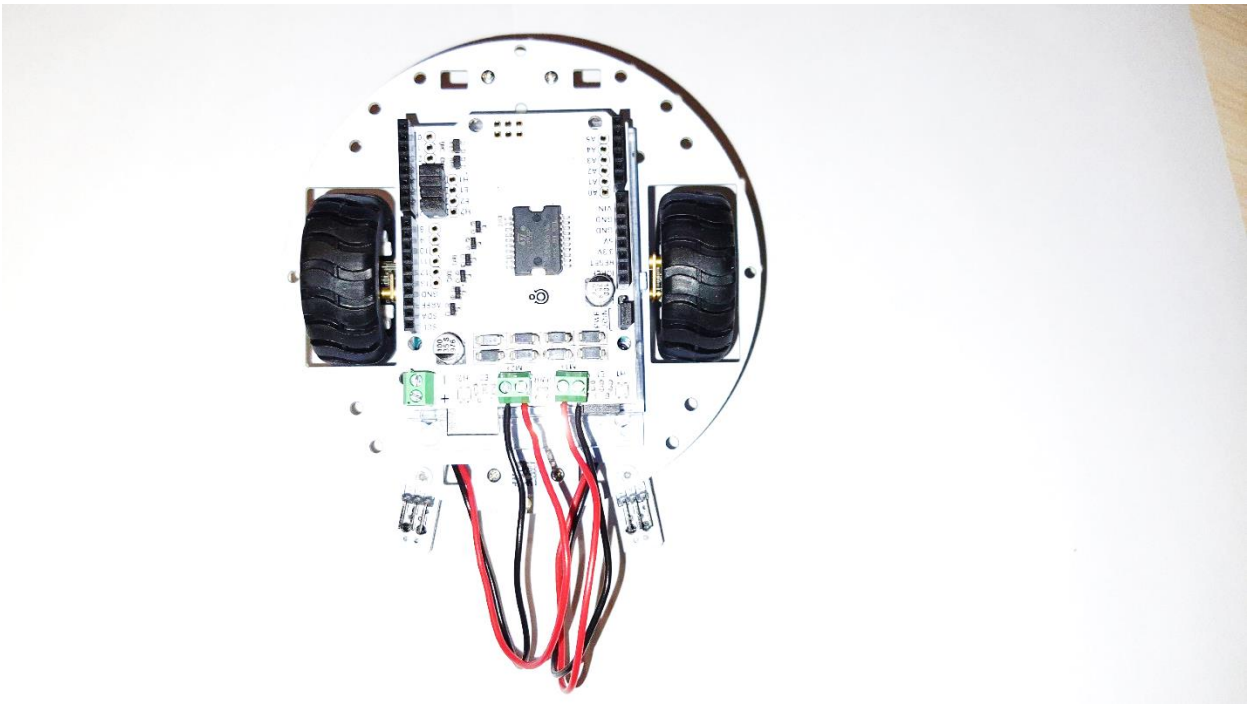


Рисунок 18

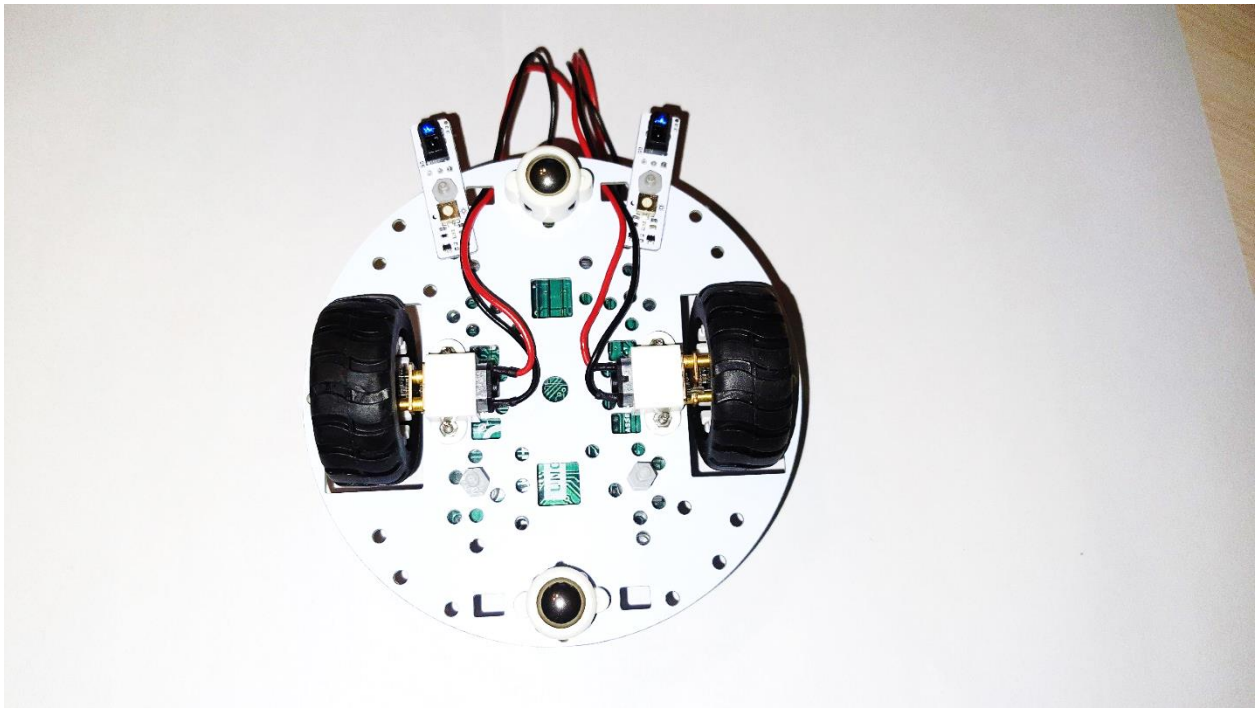


Рисунок 19

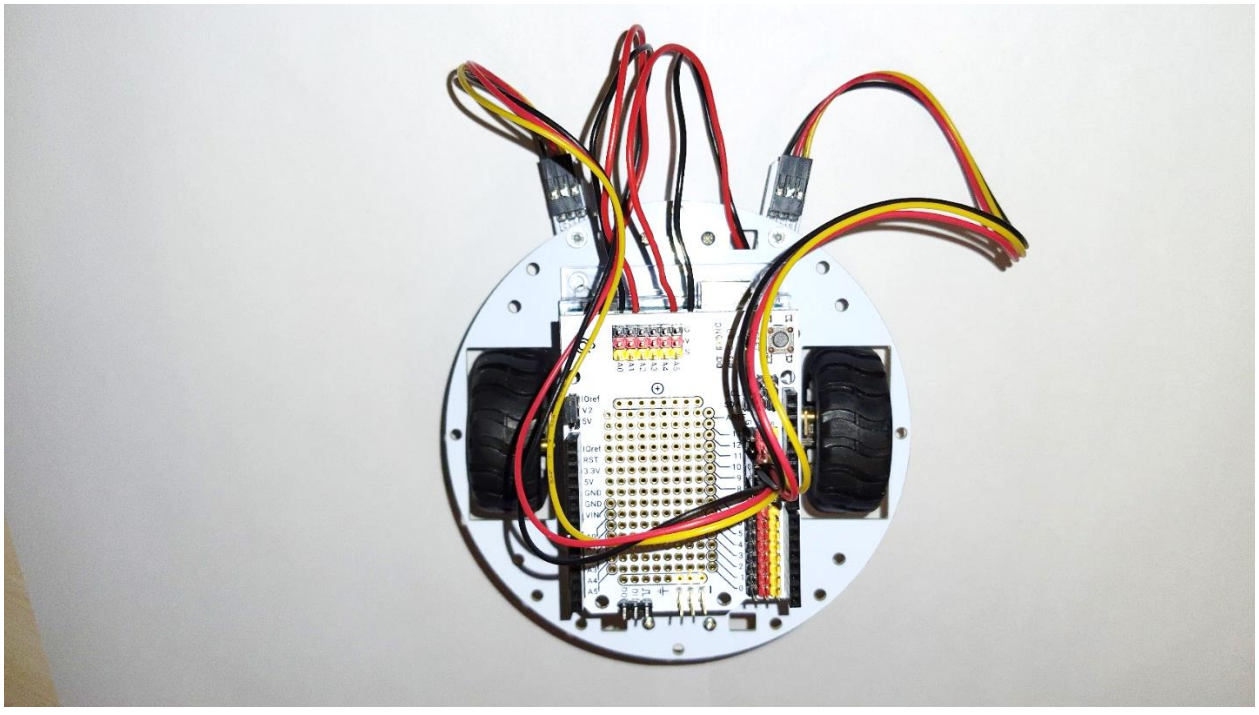


Рисунок 20

2. Откройте среду Arduino IDE и напишите код, представленный в листинге 1.
3. Загрузите код в плату.
4. Проверьте работу схемы на оборудовании.

Листинг 1

```
/*
 * Работа №24
 * Знакомство с простейшим роботом
 */

#define MotLNap 4 //Контакт направления левого двигателя
#define MotLSp 5  //Контакт скорости левого двигателя
#define MotRNap 7 //Контакт направления правого двигателя
#define MotRSp 6  //Контакт скорости правого двигателя

void setup() {
       //Настройка контактов для двигателей
       pinMode(MotLNap, OUTPUT);
       pinMode(MotLSp, OUTPUT);
```

```
        pinMode(MotRNap, OUTPUT);
        pinMode(MotRSp, OUTPUT);
}

void loop() {
        //Выбор направления движения вперед для левого двигателя
        digitalWrite(MotLNap, HIGH);

        //Выбор направления движения вперед для правого двигателя
        digitalWrite(MotRNap, HIGH);

        //Установление скорости движения для левого и правого двигателя
        // В зависимости от двигателей и покрытия минимальная скорость = 80 +/-
20
        analogWrite(MotLSp, 100);
        analogWrite(MotRSp, 100);

        //При помощи задержки формируем длительность работы двигателей
        delay(6000);

        //Длительность работы двигателей = пройденному расстоянию; при заданной
скорости двигателей
        //Задание разной скорости на двигателях для поворота

        analogWrite(MotLSp, 250);
        analogWrite(MotRSp, 90);

        delay(10000);
}
```

Лабораторная работа № 25. Движение по линии

Задача 1. Создать простейшего робота на Arduino.

Оборудование:

- Плата Arduino Uno;
- Персональный компьютер;
- Кабель USB для подключения платы Arduino Uno к компьютеру;
- Робот, собранный в лабораторной работе № 24/

Ход выполнения работы:

1. Установите робота, собранного в лабораторной работе № 24, на линию, напишите код, представленный в листинге 1, проверьте движение робота (Рисунок 1 – 3).

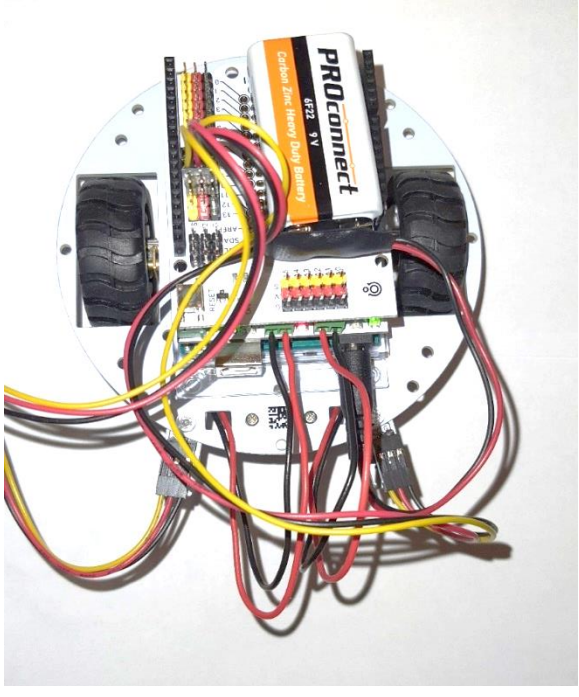


Рисунок 1

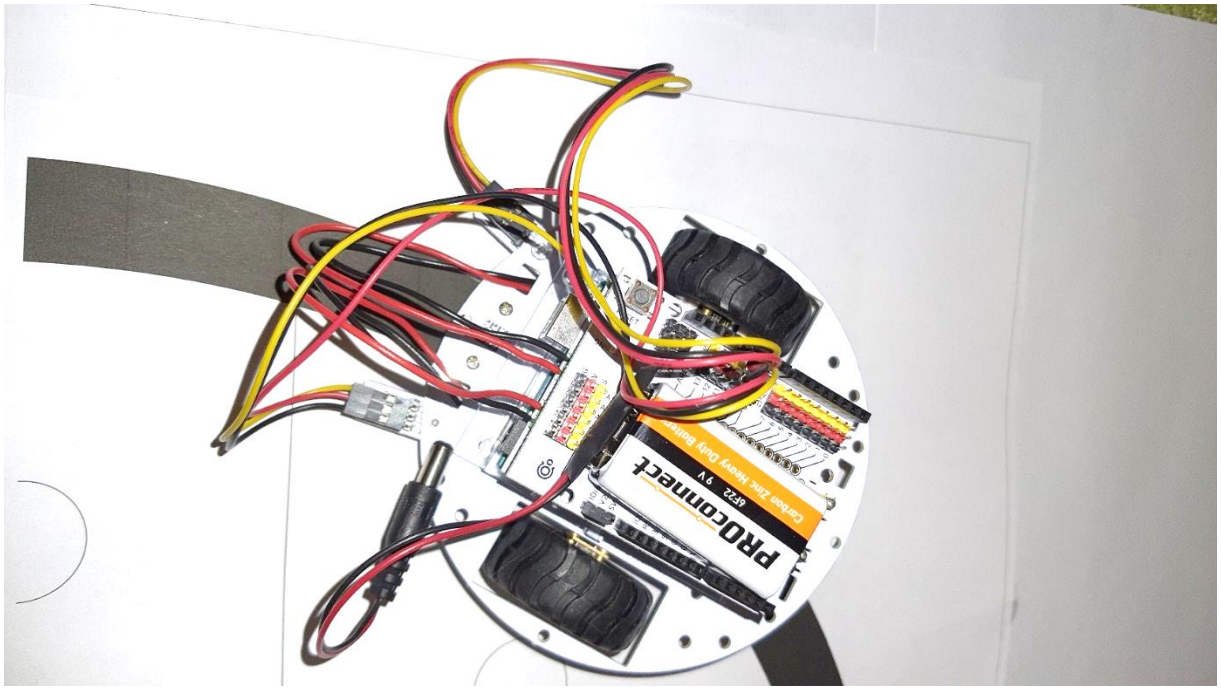


Рисунок 2

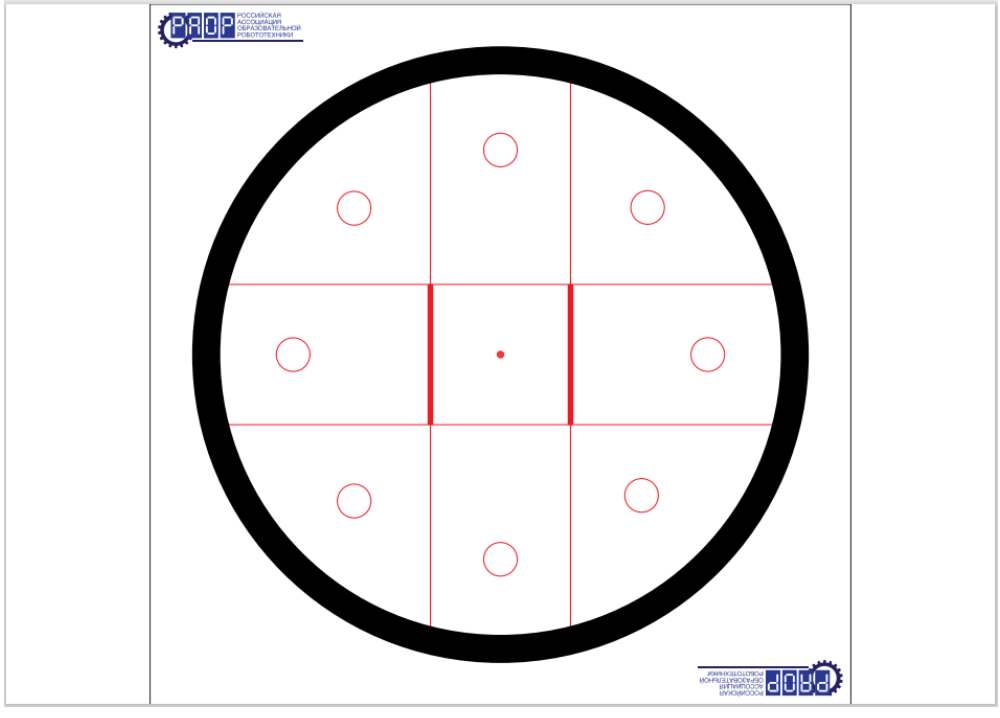


Рисунок 3

Листинг 1

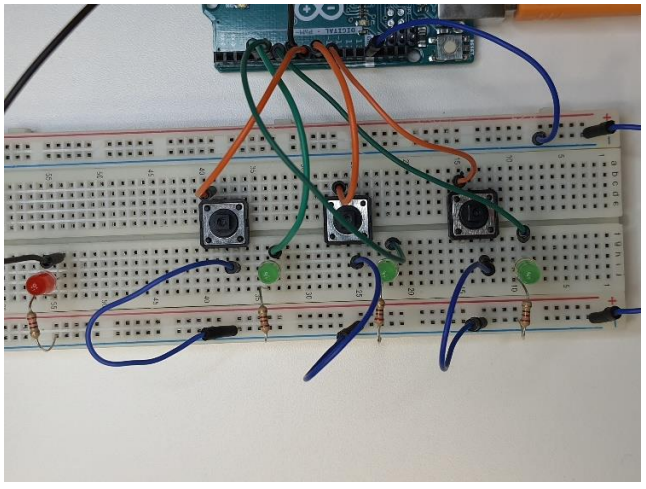
```
/*
 * Работа №25
 * Движение по линии
 */

#define MotLNap 4 //Контакт направления левого двигателя
#define MotLSp 5  //Контакт скорости левого двигателя
#define MotRNap 7 //Контакт направления правого двигателя
#define MotRSp 6  //Контакт скорости правого двигателя

#define Lline 10  //Контакт левого датчика линии
#define Rline 11  //Контакт правого датчика линии

void setup() {

  //Настройка контактов для двигателей
  pinMode(MotLNap, OUTPUT);
  pinMode(MotLSp, OUTPUT);
  pinMode(MotRNap, OUTPUT);
  pinMode(MotRSp, OUTPUT);

  //Настройка контактов для датчиков линии
  pinMode(Lline, INPUT);
  pinMode(Rline, INPUT);
}

void loop() {

  //Выбор направления движения вперед для левого двигателя (Вперед)
  digitalWrite(MotLNap, HIGH);
  //Выбор направления движения вперед для правого двигателя (Вперед)
  digitalWrite(MotRNap, HIGH);

  if ((digitalRead(Lline)) && (digitalRead(Rline)))
  {
    //Белый цвет на левом и правом датчике (линия по центру - движение вперед)
    analogWrite(MotLSp, 80);
    analogWrite(MotRSp, 80);
  }

  if ((!digitalRead(Lline)) && (!digitalRead(Rline)))
```

```
  {
    //Черный цвет на левом и правом датчике (Авария - съезд с трассы)
    analogWrite(MotLSp, 0);
    analogWrite(MotRSp, 0);
  }


  if (digitalRead(Lline))
  {
    //Под левым датчиком черный - под правым белый --> поворот влево
    analogWrite(MotLSp, 90);
    analogWrite(MotRSp, 50);
  }

  if(digitalRead(Rline))
  {
      //Под правым датчиком черный - под левым белый --> поворот вправо
      analogWrite(MotLSp, 50);
      analogWrite(MotRSp, 90);
  }
}
```

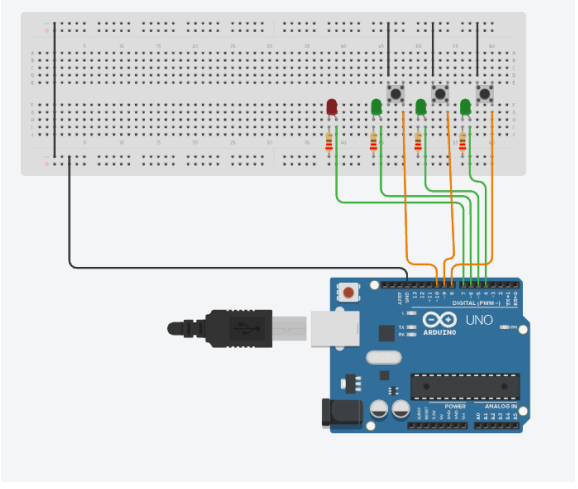
РАЗДЕЛ 2.2. СБОРНИК ЗАДАНИЙ ЛАБОРАТОРНЫХ РАБОТ ПО СПЕЦКУРСУ «ПРОГРАММИРОВАНИЕ» ДЛЯ 11 ИТ-КЛАССА

Лабораторная работа № 1. Знакомство с платой Arduino

Собрать схему, которая имитирует работу щита с кнопками и световой индикацией в кабине лифта.



а



б

Общий вид схемы
а) физическое исполнение; б) рисунок схемы

Пояснения. На макетной плате расположены четыре светодиода: три (зеленые) имитируют отображение световой индикации при нажатии на кнопку соответствующего «этажа», а четвертый (красный) мигает во время «движения лифта». Когда не нажата ни одна кнопка, ничего не происходит. Как только в «лифт» зашел пассажир и нажал на кнопку, загорается соответствующий ей светодиод и горит, пока «лифт едет». Красный светодиод в этом время мигает, имитируя индикацию движение «лифта». После «достижения определенного этажа» все светодиоды перестают гореть, и только после нового нажатия на кнопку загорается соответствующий зеленый светодиод и вместе с ним красный.

Дополнительное задание для самостоятельного выполнения

1. Измените программный код так, чтобы:

а) время мигания светодиода движения составило 2 секунды;

б) время мигания светодиода движения по длительности соответствовало этажу назначения (1-й этаж – 1 секунда, 2-й этаж – 2 секунды, 3-й этаж – 3 секунды);

2. Модифицируйте схему, добавив светодиод и кнопку для 4-го этажа.
3. Измените программный код так, чтобы по прибытии на этаж зеленый светодиод соответствующего этажа несколько раз приветственно мигнул.

Лабораторная работа № 2. Панель индикации роутера

Создать модель, имитирующую панель индикации Wi-Fi роутера. На макетной плате расположены семь светодиодов. Красный светодиод эмулирует индикатор питания. Следующий за ним желтый светодиод эмулирует индикатор мощности Wi-Fi сигнала. Четыре зеленых светодиода отвечают за индикацию подключения проводных клиентов (персональных компьютеров). Крайний правый желтый светодиод загорается на непродолжительное время при нажатии кнопки, эмулируя работу функции WPS (Wi-Fi Protected Setup – стандарт и протокол быстрого и безопасного сопряжения (соединения) двух Wi-Fi устройств между собой).

Дополнительное задание для самостоятельного выполнения

1. В операторе switch() в метках case замените цифры выходных контактов на их текстовые имена (строки 4 – 7). Проверьте работу программы.
2. Измените код в строках 75 – 79 так, чтобы длительность нарастания сигнала ШИМ была увеличена.
3. Подключите дополнительный светодиод на контакт 10 и настройте вывод уровня сигнала ШИМ при помощи функции random(). Не забудьте добавить небольшую задержку (delay()) после вывода уровня, чтобы визуально различить изменение яркости светодиода.

Лабораторная работа № 3. Знакомство с потенциометром

Изучить изменение сопротивления потенциометра в зависимости от положения поворота управляющей ручки. От изменения сопротивления потенциометра зависит яркость свечения светодиода.

Дополнительное задание для самостоятельного выполнения

1. Измените диапазоны преобразований функции map. Опишите полученные результаты.
2. Добавьте между контактом A0 и выходом потенциометра резистор 10 кОм. Опишите, что вы наблюдаете.

Лабораторная работа № 4. Знакомство с термистором (терморезистором)

Изучить изменение сопротивления термистора в зависимости от температуры окружающей среды. От изменения сопротивления термистора зависит яркость свечения светодиода.

Дополнительное задание для самостоятельного выполнения

1. Добавьте второй термистор и второй светодиод. Попробуйте подать разные значения температуры на каждый из них.
2. Попробуйте подобрать значение первого диапазона функции map (из которого происходит преобразование) для наиболее резкого изменения яркости светодиода в рамках вашей окружающей среды.

Лабораторная работа № 5. Знакомство с RGB-светодиодом

Изучить работу RGB-светодиода.

Дополнительное задание для самостоятельного выполнения

1. Сформируйте на RGB-светодиоде сначала красный, потом зеленый, потом синий цвет.
2. Для любого из трех цветов создайте эффект плавного изменения цвета от светлого оттенка до темного.

Лабораторная работа № 6. Знакомство с пьезодинамиком

Изучить работу пьезодинамика.

Дополнительное задание для самостоятельного выполнения

1. Сформируйте звук, который будет плавно повышаться до определенного предела, некоторое время звучать, а потом также плавно понижаться до исходного значения.
2. Сымитируйте сигнал автомобильной сирены (часто чередующиеся повышения и понижения звука).

Лабораторная работа № 7. Знакомство с биполярным транзистором N-P-N типа

Изучить работу биполярного транзистора n-p-n типа.

Дополнительное задание для самостоятельного выполнения

1. Добавьте в схему потенциометр для регулирования уровня сигнала ШИМ на базе транзистора.
2. Подключите «+» светодиода к контакту платы Arduino 3,3 В. Опишите результат.

Лабораторная работа № 8. Знакомство полевым mosfet-транзистором

Изучить работу MOSFET-транзистора.

Дополнительное задание для самостоятельного выполнения

1. Добавьте в схему потенциометр для регулирования уровня сигнала ШИМ.
2. Подключите «+» светодиода к контакту платы Arduino 3,3 В. Опишите результат.

Лабораторная работа № 9. Знакомство с одноразрядным семисегментным индикатором

Изучить работу семисегментого индикатора.

Дополнительное задание для самостоятельного выполнения

1. Измените код таким образом, чтобы вывести буквы a, b, c.
2. Измените код таким образом, чтобы на индикаторе чередовались цифры и буквы a, b, c в случайном порядке.

Лабораторная работа № 10. Знакомство с подключением семисегментного индикатора через ИМС CD4026

Вывод чисел от 0 до 9 на семисегментный индикатор.

Дополнительное задание для самостоятельного выполнения

1. Измените код таким образом, чтобы после вывода цифры 5 производилось обнуление подсчитанного числа и вывод цифр начинался заново (с 0).
2. Добавьте в схему светодиод и измените код таким образом, чтобы после вывода на индикатор очередной цифры была пауза в выводе цифр, а светодиод мигал столько раз, какая цифра была выведена.

Лабораторная работа № 11. Подсчет нажатия кнопки с выводом на семисегментный индикатор

Вывести на семисегментный индикатор количества нажатий на кнопку.

Дополнительное задание для самостоятельного выполнения

1. Добавьте в схему светодиод и измените код таким образом, чтобы после вывода каждой цифры светодиод мигал столько раз, какая цифра была выведена.
2. Добавьте в схему второй светодиод и измените код таким образом, чтобы после вывода на индикатор очередной цифры была пауза в выводе цифр и первый светодиод мигал столько раз, какая цифра была выведена, а второй светодиод мигал бы столько раз, какова разность между 9 и количеством произведенных нажатий.

Лабораторная работа № 12. Построение на двух семисегментных индикаторах

Вывести на два семисегментных индикатора числа от 0 до 99.

Дополнительное задание для самостоятельного выполнения

1. Добавьте в схему клавишу принудительного программного сброса.
2. Добавьте в схему три светодиода: красный, желтый, зеленый. В начале работы программы светодиоды не горят. Когда выводимые цифры достигнут 30, должен загореться желтый светодиод. После вывода числа 60, должен загореться красный светодиод. И только после вывода числа 99, должен загораться зеленый светодиод.

Лабораторная работа № 13. Вывод случайных чисел на семисегментный индикатор

Выводить на семисегментный индикатор случайное число по нажатию кнопки.

Дополнительное задание для самостоятельного выполнения

1. Добавьте кнопку для сброса в значение 0.
2. Добавьте в схему светодиод, яркость которого будет соответствовать выводимому случайному числу.

Лабораторная работа № 14. Формирование нот на пьезодинамике с выводом индикации на семисегментный индикатор

Формирование нот на пьезодинамике с выводом индикации на семисегментный индикатор.

Дополнительное задание для самостоятельного выполнения

1. Добавьте кнопку для вызова определенной мелодии.
2. Добавьте в схему семисегментный индикатор, на который выводятся случайные числа, а мелодии соответствуют выводимому случайному числу.

Лабораторная работа № 15. Таймер с индикацией на семисегментном индикаторе

Выполнить сборку и написание кода для таймера с индикацией на семисегментном индикаторе.

Дополнительное задание для самостоятельного выполнения

1. Добавьте в схему RGB-светодиод, цвет которого будет меняться от зеленого до красного.
2. Добавьте в схему пьезопищалку, звук которой будет меняться от низкой до высокой частоты.

Лабораторная работа № 16. Знакомство с ЖКИ

Выполнить сборку и написание кода для знакомства с ЖКИ.

Дополнительное задание для самостоятельного выполнения

1. Выведите свои имя и фамилию на экран.
2. Выведите на экран в цикле набор из пяти любых слов (разной длины). Изучите процесс обновления ЖКИ.

Лабораторная работа № 17. Вывод нажатия кнопки на ЖКИ

Написать программу, которая выводит на ЖКИ сообщение о нажатия кнопки.

Дополнительное задание для самостоятельного выполнения

1. Добавьте в схему вторую кнопку и напишите программу, в результате выполнения которой сообщение о нажатии одной кнопки выводится в первую строку, а сообщение о нажатии другой – во вторую строку.
2. Добавьте в схему вторую кнопку, по нажатии которой экран будет очищаться.

Лабораторная работа № 18. Вывод уровня освещенности в помещении на ЖКИ

Написать программу, которая выводит на ЖКИ сообщение об уровне освещенности в помещении.

Дополнительное задание для самостоятельного выполнения

1. Измените программный код так, чтобы во вторую строку выводились сведения о том, является ли данный уровень освещенности достаточным.
2. Добавьте в схему пять светодиодов. Все светодиоды горят, когда уровень освещенности максимальный; если уровень освещенности уменьшается, часть светодиодов гаснет.

Лабораторная работа № 19. Вывод температуры на ЖКИ

Написать программу, которая будет выводить на ЖКИ сведения о температуре.

Дополнительное задание для самостоятельного выполнения

1. Добавьте в схему второй терморезистор и измените код так, чтобы показания первого терморезистора выводились слева в первой строке, а второго – справа во второй строке.
2. Добавьте в схему два светодиода (по одному рядом с каждым терморезистором). Загораться должен светодиод у того терморезистора, на котором температура выше. Если

терморезисторы показывают одинаковую температуру, должны загораться оба светодиода.

Лабораторная работа № 20. 60-секундный таймер с выводом информации на ЖКИ

Написать программу, которая будет выводить на ЖКИ.

Дополнительное задание для самостоятельного выполнения

1. Измените код так, чтобы одновременно с выводом в одной строке секунд с шагом в одну секунду, во второй строке выводились бы секунды с шагом 3.
2. Измените код так, чтобы в одной строке секунды выводились от 0 до 60, а в другой строке одновременно выводились бы секунды в обратном порядке.

Лабораторная работа № 21. Часы с будильником

Написать программу, при выполнении которой на ЖКИ выводится отсчёт времени и по достижении установленного срока звучит сигнал.

Дополнительное задание для самостоятельного выполнения

1. Добавьте в схему светодиод, который будет загораться одновременно со звуковым сигналом.
2. Измените код так, чтобы за 10 и потом за 5 минут перед срабатыванием в нижней строке выводилось бы предупреждение.

Лабораторная работа № 22. Панель индикации роутера

Создать модель, имитирующую панель индикации Wi-Fi роутера. На макетной плате расположены семь светодиодов. Красный светодиод эмулирует индикатор питания. Следующий за ним желтый светодиод эмулирует индикатор мощности Wi-Fi сигнала. Четыре зеленых светодиода отвечают за индикацию подключения проводных клиентов (персональных компьютеров). Крайний правый желтый светодиод загорается на непродолжительное время при нажатии кнопки, эмулируя работу функции WPS (Wi-Fi Protected Setup – стандарт и протокол быстрого и безопасного сопряжения (соединения) двух Wi-Fi устройств между собой).

Дополнительное задание для самостоятельного выполнения

1. В операторе switch() в метках case замените цифры выходных контактов на их текстовые имена (строки 4 – 7). Проверьте работу программы.
2. Измените код в строках 75 – 79 так, чтобы длительность нарастания сигнала ШИМ была увеличена.
3. Подключите дополнительный светодиод на контакт 10 и настройте вывод уровня сигнала ШИМ при помощи функции random(). Не забудьте добавить небольшую задержку (delay()) после вывода уровня, чтобы визуально различить изменение яркости светодиода.

Лабораторная работа № 23. Железнодорожный переезд

Создать модель железнодорожного переезда. На переезде предусмотрены светофоры, по два с каждой стороны железной дороги: один для автомобиля, другой для пешехода. Пешеход 1 и Машина 1 находятся слева, а Пешеход 2 и Машина 2 находятся справа. При этом, если пешеход переходит переезд, то машинам двигать запрещено. Когда Машина 1 пересекает переезд, то Машине 2 переезд запрещен, и наоборот. Если поезд приближается к переезду, то для машин и пешеходов переезд закрывается автоматически, о чем сообщает красный мигающий сигнал на всех светофорах (светодиодах). Если пешеход желает перейти переезд, он должен нажать кнопку.

Дополнительное задание для самостоятельного выполнения

1. Измените код листинга 1 таким образом, чтобы при нажатии кнопки пешеходом, зеленый сигнал для него формировался только при отсутствии поезда на переезде. В текущей версии проверки, есть ли поезд, при разрешении движения пешеходам, не происходит.
2. Измените код светофоров для машин так, чтобы желтый светодиод мигал четыре раза, а не горел постоянно.
3. Измените код так, чтобы зеленый сигнал для пешеходов загорался только в случае, когда были одна за другой нажаты обе кнопки.

Лабораторная работа № 24. Знакомство с простейшим роботом на Arduino

Создать простейшего робота на Arduino для изучения механики движения робототехнической платформы в базовых направлениях (вперед – назад, влево – вправо).

Дополнительное задание для самостоятельного выполнения

1. Создать программу, реализующую алгоритм движения по кругу.
2. Создать программу, реализующую движение робота серией поворотов по 90°.

Лабораторная работа № 25. Движение по линии

Изучить алгоритм движения робота по линии.

Дополнительное задание для самостоятельного выполнения

1. Создать программу, реализующую алгоритм плавного движения по круговой линии.
2. Создать программу, реализующую алгоритм движения робота по трассе с изменяющимся направлением линии.

Список используемых источников:

1. Онлайн справочник программиста на C и C++. URL: http://www.c-cpp.ru/books/proishozhdenie-yazyka-s-0 (Дата обращения: 13.12.2021)
2. Герберт Шилдт. C++ базовый курс. изд. Диалектика-Вильямс, - 2018 г. URL: https://www.bsuir.by/m/12_100229_1_98220.pdf (Дата обращения: 13.12.2021)
3. Знать о чем угодно – strephonsays. URL: https://ru.strephonsays.com/source-code-and-vs-object-code-12793 (Дата обращения: 13.12.2021)
4. Обзор элементов языка C++ — Основы программирования. URL: http://pydev.ru/c/obzor-elementov-yazyka-c.html (Дата обращения: 13.12.2021)
5. Ravesli Программирование для начинающих. URL: https://ravesli.com/urok-49-globalnye-peremennye/ (Дата обращения: 13.12.2021)
6. METANIT.COM - Сайт о программировании. URL: https://metanit.com/cpp/tutorial/2.3.php (Дата обращения: 13.12.2021)
7. Логические выражения в C_C++. Как ошибаются профессионалы _ Хабр. URL: https://habr.com/ru/company/pvs-studio/blog/281316/ (Дата обращения: 13.12.2021)
8. Студенческие реферативные статьи и материалы. URL: https://studref.com/417698/informatika/proverka_usloviy#:~:text=В%20качестве%20условия%20можно%20использовать,значению%20False%2C%20ненулевое%20—%20True (Дата обращения: 13.12.2021)
9. Как я учился программировать на C++. URL: https://www.sites.google.com/site/mql5cpp/home/razdel-12 (Дата обращения: 13.12.2021)
10. Студопедия — Ваша школопедия. URL: https://studopedia.ru/5_67305_operator-vibora.html (Дата обращения: 13.12.2021)
11. Сайт для учащихся. URL: https://topuch.ru/operator-vibora-operator-vibora/index.html (Дата обращения: 13.12.2021)
12. Студенческие реферативные статьи и материалы. URL: https://studref.com/324589/informatika/operator_vybora (Дата обращения: 13.12.2021)
13. Средства разработчика, техническая документация и примеры кода _ Microsoft Docs. URL: https://docs.microsoft.com/ru-ru/cpp/cpp/break-statement-cpp?view=msvc-160 (Дата обращения: 13.12.2021)