

# **Основы кибернетики и робототехники**

Лекция 2, Лабораторная 2

Язык программирования отладочных плат Arduino основан на C/C++, но имеет ряд особенностей, связанных с управлением устройствами реального мира и электронными схемами.

Пример программы для Arduino:

```
int ledPin = 13;
void setup(){
  pinMode(ledPin, OUTPUT);
}
void loop(){
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Этот код управляет светодиодом, включая и выключая его (мигающий светодиод).

Переменную можно представить в виде ящика, который подписан (имеет имя) и в нем что-то лежит (значение), при этом содержимое может меняться. Имя для переменных выбирается программистом, но должно быть уникальным в пределах одной функции, если переменная создана внутри функции, и для всей программы, если переменная создана вне функции (глобальная). В языке, который используется для программирования Arduino, при создании переменной необходимо определять тип данных, который будет храниться в переменной.

Доступные типы данных:

Тип	Размер (байт)	Диапазон значений
bool	1	true или false
byte	1	от 0 до 255
char	1	-128 до 127
int	2	от -32 768 до 32 767
unsigned int	2	от 0 до 65535
long	4	от -2,147,483,648 до 2,147,483,647
float	4	от -3.4028235E+38 до 3.4028235E+38

Операторы позволяют производить действия над переменными:

- **= (оператор присваивания).**
- **+** (сумма). Например, `int i = 10+10`. После выполнения этой строки кода в переменной `i` будет храниться значение 20.
- **- (разность).** Например, `int i = 10-5`. После выполнения этой строки кода в переменной `i` будет храниться значение 5.
- **\*** (умножение). Например, `int i = 10*3`. После выполнения этой строки кода в переменной `i` будет храниться значение 30.
- **/ (деление).** Например, `float i = 10/4`. После выполнения этой строки кода в переменной `i` будет храниться значение 2.5.
- **% (остаток от деления нацело).** Например, `int i = 5/2`. После выполнения этой строки кода в переменной `i` будет храниться значение 1.

Комментарий в одну строку создается при помощи двух прямых слешей - //.

```
// Пример однострочного комментария
```

Комментарий на несколько строк задается при помощи прямого слеша и символа "\*", и продолжается до комбинации "\*".

```
/* все, что находится  
здесь  
является комментарием */
```

Некоторые встроенные функции, для удобства разбиты на группы.

Функции работы с временем:

**delay(x)** - остановка программы на x миллисекунд.

**delayMicroseconds(x)** - остановка программы на x микросекунд.

**millis()** - возвращает количество миллисекунд, прошедших от запуска программы. Работа с контактами (пинами) платы.

**pinMode(x,mode)** - устанавливает режим работы пина x в режим вывода (mode='OUTPUT') или ввод (mode='INPUT').

**digitalWrite(x,value)** - подает логическую единицу (можно записывать как HIGH, true, 1) или логический 0 (можно записывать как LOW, false, 0) значение на цифровой выход x.

**digitalRead(x)** - считывает значение (логическую 1 или 0) с входа x.

**analogRead(x)** - считывает значение с аналогового входа x, преобразуя уровень напряжения в цифровое значение. В большинстве плат Arduino аналоговый вход принимает на вход от 0 до 5 вольт. Этот диапазон будет разбит на 1024 участка от 0 до 1023, с шагом около (0.0049 Вольт).

Таким образом, можно считывать, например, уровень освещенности.

**analogWrite(x,value)** - формирует заданное аналоговое напряжение (value) на выводе номер x в

Математические функции:

**min(x,y)** - функция возвращает минимальное значение из двух значений x и y.

**max(x,y)** - функция возвращает максимальное значение из двух значений x и y.

**random(x,y)** - функция возвращает случайное значение в диапазоне от x до y(не включая y).

**abs(x)** - возвращает модуль числа x.

**map(x, fromLow, fromHigh, toLow, toHigh)** - возвращает значение x, в новом диапазоне.

Например, `map(5, 0, 10, 0, 100)` вернет 50.

**pow(x, exponent)** - возводит число x в указанную степень.

**sqrt(x)** - вычисление квадратного корня из числа x.

**sin(x)** - вычисление синуса x.

**cos(x)** - вычисление косинуса x.

Набор функций Serial позволяет плате Arduino обмениваться данными с компьютером и выводить сообщения в монитор последовательного интерфейса (Serial monitor).

**Serial.begin(speed)** - открывает соединение с заданной скоростью пере- 52 53 Разработка умных устройств на базе Arduino дачи данных в бит/с. Часто используют скорость 9600 бит/с.

**Serial.end()** - закрывает соединение.

**Serial.print(s)** - передает значение переменной s.

**Serial.read()** - считывает байт данных из буфера последовательного соединения.



Если необходимо добавить условия в сценарии, то потребуются условные конструкции.

Пример сценариев:

- стало темно, нужно включить освещение.
- была закрыта дверь, через 5 минут нужно понизить обороты вращения вентилятора.

Для проверки условий в языке программирования Arduino есть конструкция if...else.

```
if (<условие1>) {  
  // блок кода, который выполняется, если <условие1> верное  
}  
else if (<условие2>) {  
  // этот блок выполняется, если <условие2> верное  
}  
else {  
  // этот блок выполнится только в том случае, если  
  // условие 1 и условие 2 не верное  
}
```

> - больше;

>= - больше или  
равно;

< - меньше;

<= - меньше или  
равно;

== - равно

!= - не равно

Можно объединять несколько частей условия при помощи логических операторов:  
&& - И

|| - ИЛИ

! - НЕ

(отрицание)

Для повторяющихся действий или действий, которые должны происходить до выполнения какого-то условия, нужно использовать циклы.

Цикл for является самым гибким, имеет следующую структуру:

```
for (<действия выполняемые один раз перед началом
выполнения цикла>; <условие при котором выполняется
цикл, как только условие дает результат false цикл
прекращается>; <действия, которые выполняются каждый
раз при итерации (повторении) тела цикла>){
    //команды внутри цикла.
}
```

При использовании оператора цикла “**while**” тело цикла в нем выполняется до тех пор, пока верно условие, заданное в заголовке цикла.

Цикл **do... while** позволяет сначала выполнить тело цикла, а затем проверить условие.

При необходимости можно досрочно завершить текущую итерацию цикла при помощи оператора `continue` или завершить работу цикла при помощи оператора **break**.

**Функция** — это обособленный фрагмент программного кода, к которому можно обратиться из другого места программы. Обычно у функции есть имя, также в функцию могут передаваться некоторые параметры - аргументы, а в основную программу или другую функцию будет возвращаться значение, определенное внутри функции.

Пример функции, вычисляющей среднее трех целых чисел:

```
float average(int x1, int x2, int x3){  
    int sum =x1+x2+x3;  
    return sum/3;  
}
```

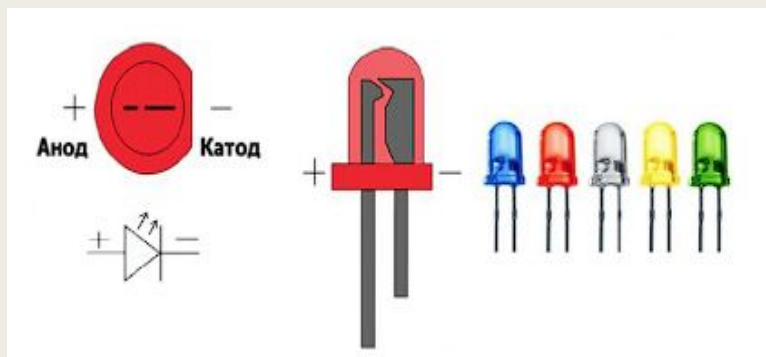
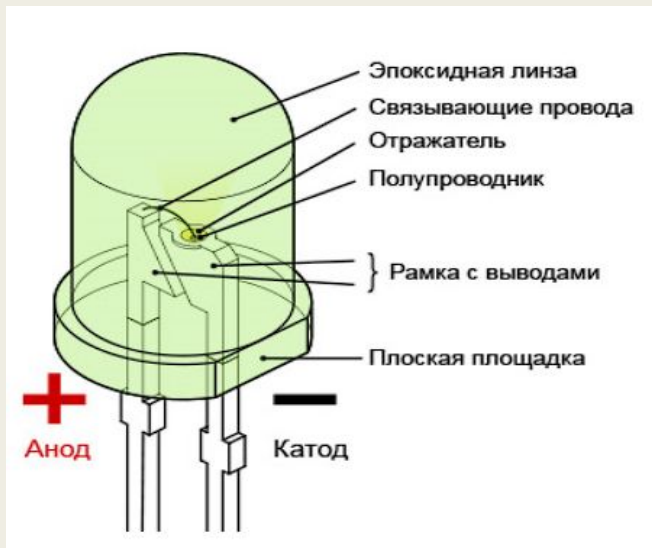
Функция имеет имя “average”, возвращает значение при помощи `return`.

**Светодиод** — это полупроводниковый прибор, способный излучать свет при пропускании через него электрического тока в прямом направлении (от анода к катоду).

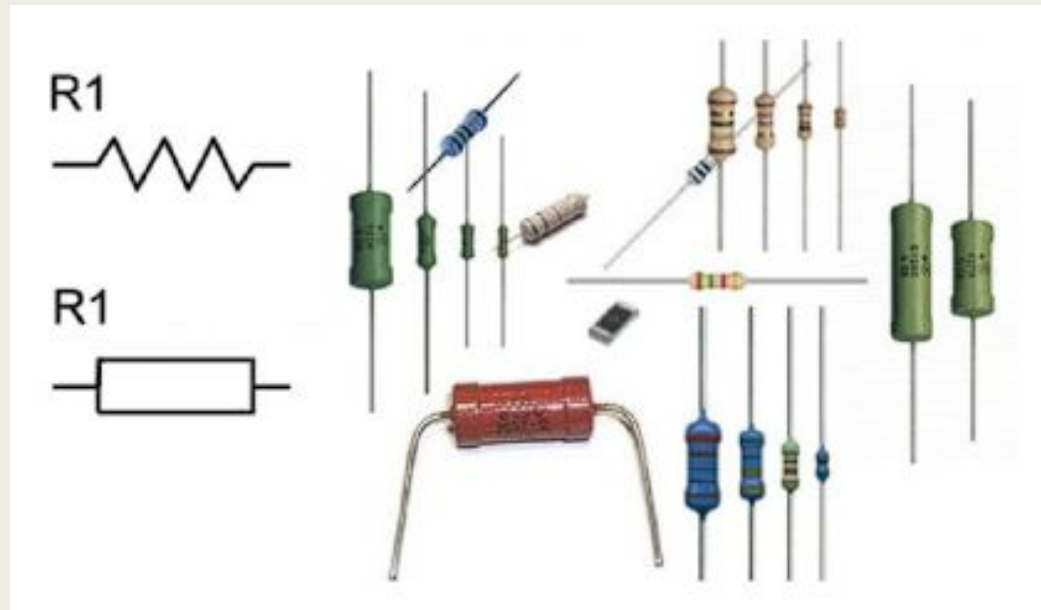
Светодиод имеет 4 ноги. 3 ноги — аноды, соответствующие отдельным цветам и одна — общий катод. Подавая сигнал на один из анодов, можно добиться свечения одним из цветов.

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

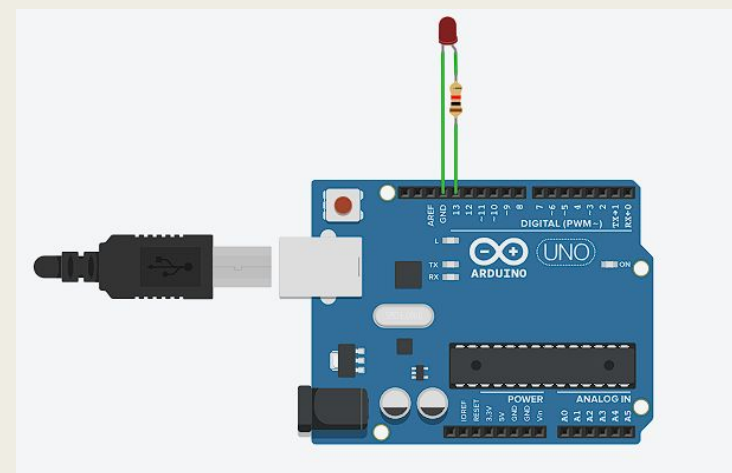
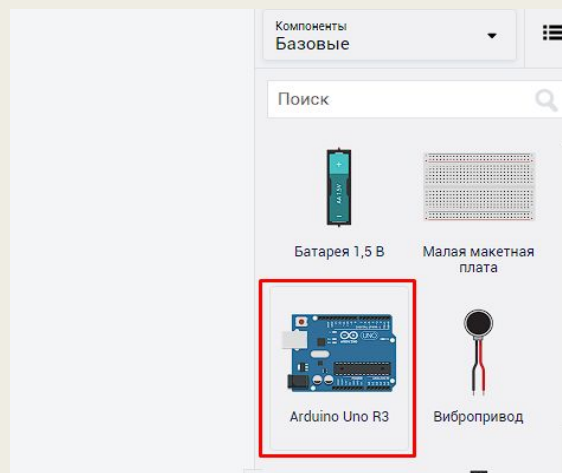
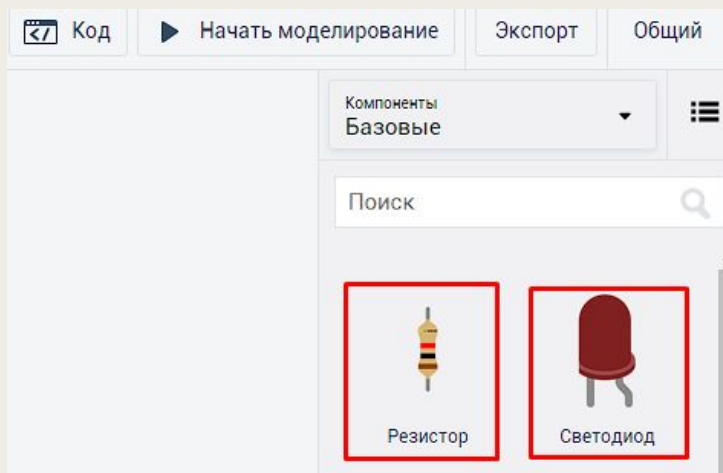
- 1) Анод светодиода имеет более длинный проводник.
- 2) Со стороны катода, корпус светодиода немного срезан.



**Резистор** - это компонент электрической схемы, который ослабляет силу тока. Если не использовать резистор, то светодиод быстро выйдет из строя (сгорит) так-как сила тока в нем будет почти не ограничена, он быстро нагреется и расплавится внутри.



В библиотеке компонентов необходимо найти нужные элементы - светодиод, резистор и плату Arduino Uno R3.



```
Код ▶ Начать моделирование Экспорт Общий
1 void setup()
2 {
3   pinMode(13, OUTPUT);
4 }
5 |
6 void loop()
7 {
8   digitalWrite(13, HIGH);
9   delay(1000); // Wait for 1000 millisecond(s)
10  digitalWrite(13, LOW);
11  delay(1000); // Wait for 1000 millisecond(s)
12 }
```

Текст программы.  
Светодиод мигает с периодичностью в 1 секунду.

В функции `setup()` необходимо инициализировать порт, подключенный к светодиоду, как выход используя функцию `pinmode("номер порта», OUTPUT).`  
`pinMode(13, OUTPUT);`

В функции `loop()` для включения светодиода необходима команда `digitalWrite(pin, value)`. Она устанавливает цифровой вывод на нужный порт. Параметр `pin` означает номер вывода для записи, а `value` - значение записи. Второй параметр может принимать только 2 значения: HIGH (5v или 3.3v) или LOW (0v).

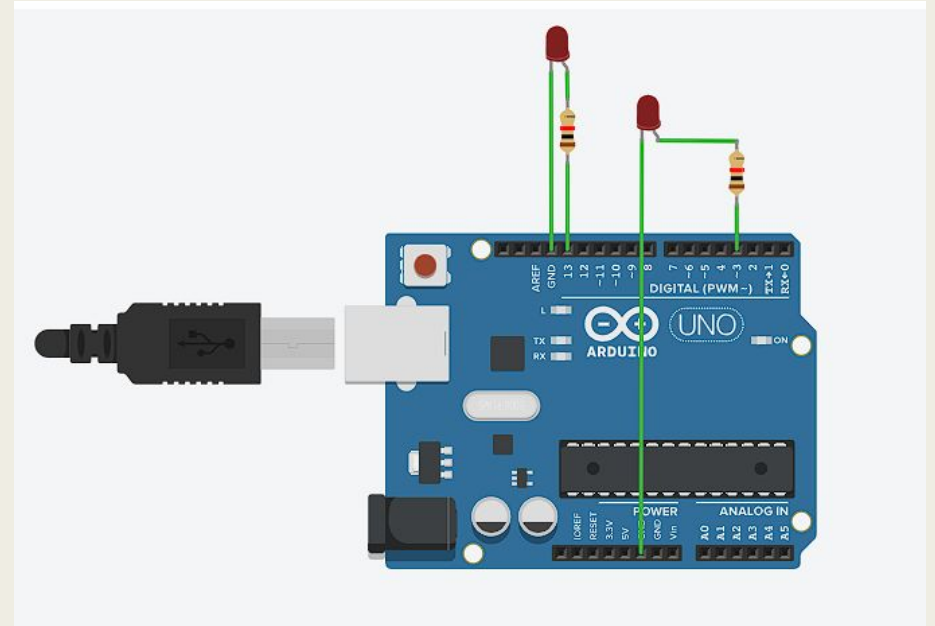
Светодиод включается посредством следующей команды:  
`digitalWrite(13, HIGH);`

Для осуществления задержки на нужное время в миллисекундах используется функция `delay(value)`. В параметр передается значение времени задержки в миллисекундах. Пауза в 1000 миллисекунду выполняется как:  
`delay(1000);`

Для выключения светодиода и последующей задержки прописываются следующие команды:  
`digitalWrite(13, LOW);`  
`delay(1000);`

Что будет происходить в результате этой программы?

```
Код ▶ Начать моделирование Экспорт Общий
Текст
1 void setup()
2 {
3   pinMode(13, OUTPUT);
4   pinMode(03, OUTPUT);
5 }
6
7 void loop()
8 {
9   digitalWrite(13, HIGH);
10  delay(1000); // Wait for 1000 millisecond(s)
11  digitalWrite(13, LOW);
12  delay(1000); // Wait for 1000 millisecond(s)
13  digitalWrite(03, HIGH);
14  delay(1000); // Wait for 1000 millisecond(s)
15  digitalWrite(03, LOW);
16  delay(1000); // Wait for 1000 millisecond(s)
17 }
```





## **Задания к лабораторной работе № 2.**

- 1) Создать программу заставляющую мигать светодиод в следующей последовательности: светодиод 1, светодиод 2, светодиод 3, все сначала.
- 2) Создать программу заставляющую мигать светодиод в следующей последовательности: светодиод 1, светодиод 2, светодиод 3, светодиод 4, светодиод 5, все сначала.
- 3) Создать программу заставляющую мигать светодиод в следующей последовательности: светодиод 1 и 2, светодиод 2 и 3, светодиод 3 и 4, светодиод 4 и 5, все сначала.
- 4) Создать программу заставляющую мигать светодиод в следующей последовательности: светодиод 1 и 2 и 3, светодиод 3 и 4, светодиод 5, все сначала.
- 5) Создать программу заставляющую мигать светодиод в следующей последовательности: светодиод 1, светодиод 5, светодиод 2, светодиод 4, светодиод 3, все сначала.
- 6) Создать программу заставляющую мигать светодиод в следующей последовательности: светодиод 1, светодиод 5, светодиод 2, светодиод 5, светодиод 3, светодиод 5, светодиод 4, светодиод 5, все сначала.
- 7) Сделайте так чтобы светодиод 1 светился полсекунды, а пауза между вспышками равна одной секунде.
- 8) Сделайте так чтобы светодиод 1 светился одну секунду, а пауза между вспышками равна двум