

# Бегущий огонёк

## Эксперимент №7

В этом эксперименте мы заставляем огонёк бежать по светодиодной шкале.

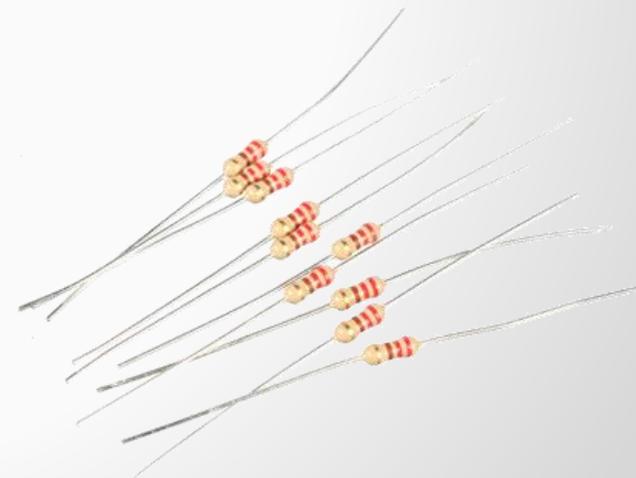
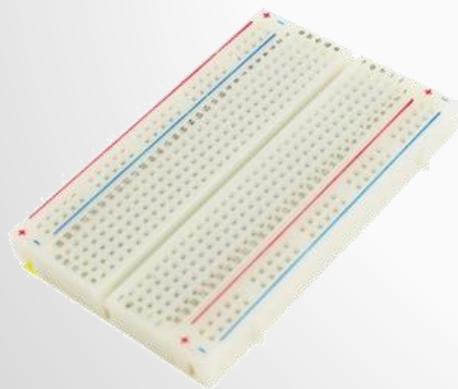
Педагог дополнительного образования

МБУДО «РЦВР» Юрышев Д.Э.

# Детали для эксперимента



[www.arduino.cc/](http://www.arduino.cc/)

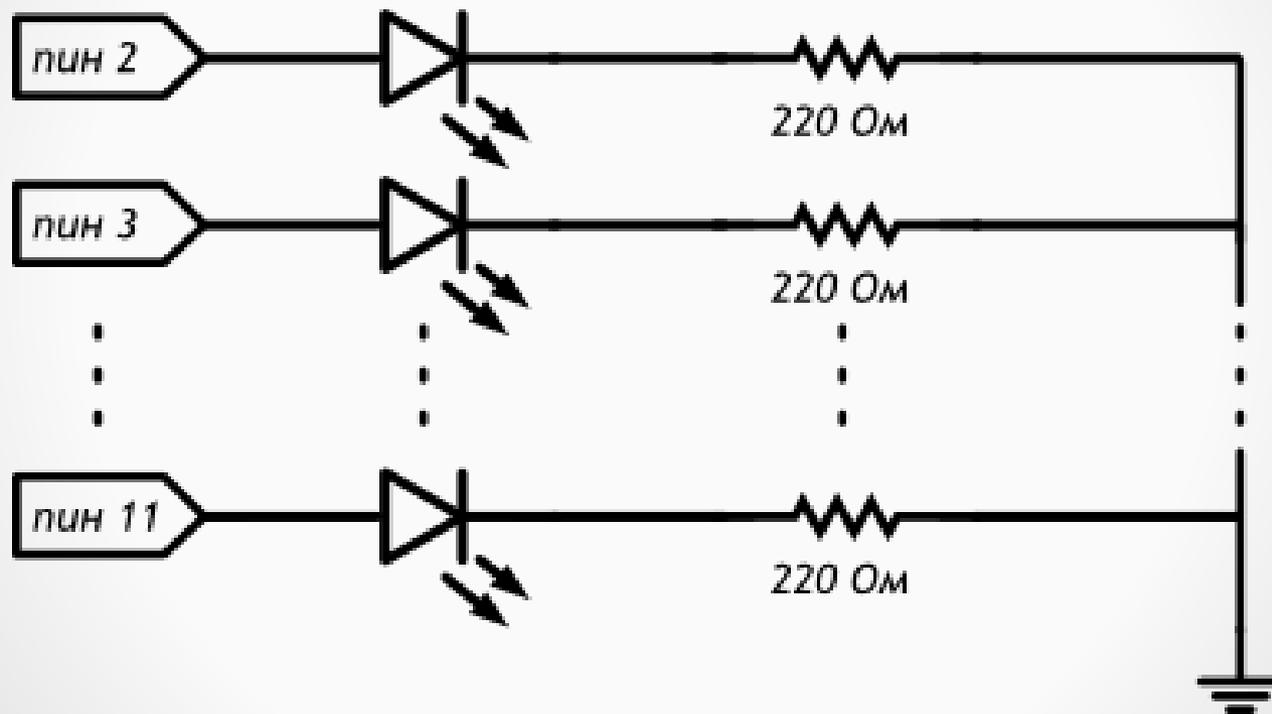


20.10.2019

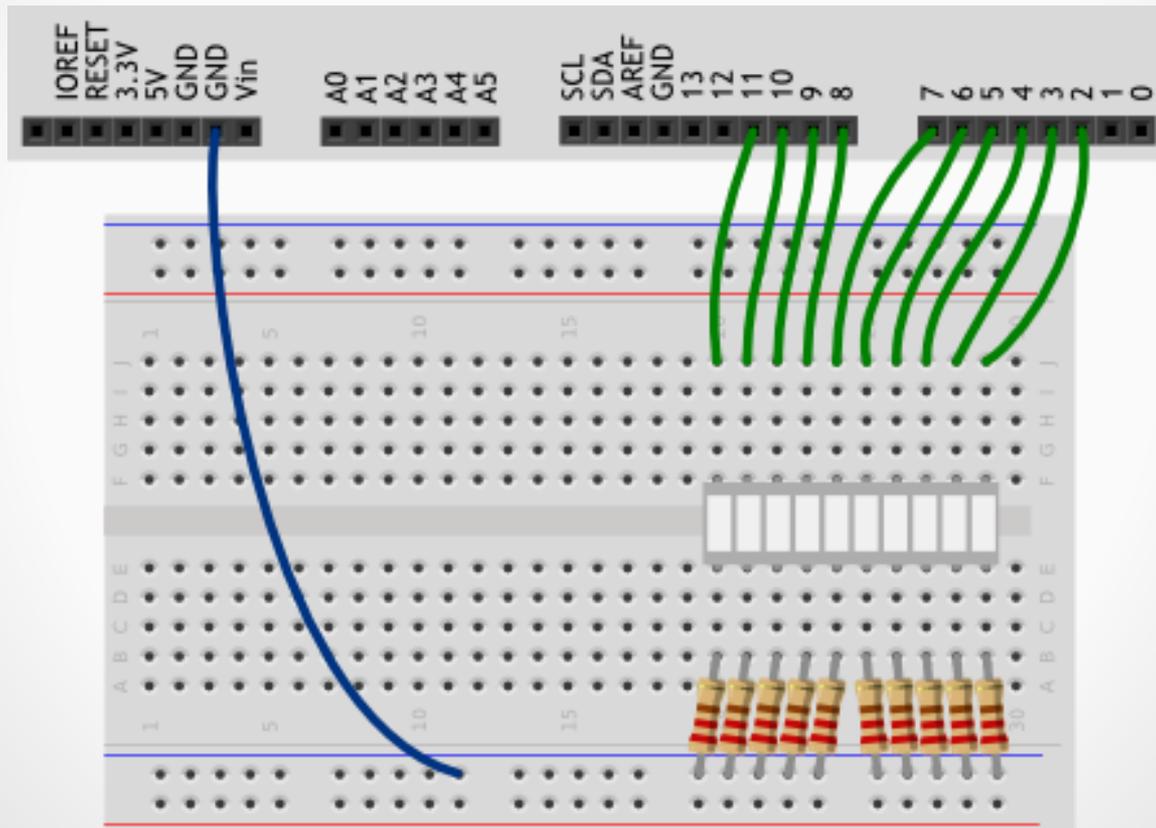
Эксперимент №7 «Бегущий огонёк»

2

# Принципиальная схема



# Схема на макетке



# Обратите внимание

- Обратите внимание, что в данном эксперименте резисторы установлены между катодами и землей в отличие от эксперимента пульсар.
- Мы подключаем светодиоды к цифровым портам, начиная с порта 2. Мы можем использовать порты 0 и 1, но они являются каналами передачи данных последовательного порта и для каждой перепрошивки платы придется отключать устройства, подключенные к ним.

# Скетч

## p070\_running\_bar.ino

```
// светодиодная шкала подключена к группе пинов расположенных подряд. Даём понятные имена первому и последнему пинам
#define FIRST_LED_PIN 2
#define LAST_LED_PIN 11

void setup()
{
  // в шкале 10 светодиодов. Мы бы могли написать pinMode 10 раз: для каждого из пинов, но это бы раздуло код и сделало его
  // изменение более проблематичным. Поэтому лучше воспользоваться циклом. Мы выполняем pinMode для (англ. for) каждого пина
  // (переменная pin) от первого (= FIRST_LED_PIN) до последнего включительно (<= LAST_LED_PIN), всякий раз продвигаясь к следующему
  // (++pin увеличивает значение pin на единицу) Так все пины от 2-го по 11-й друг за другом станут выходами
  for (int pin = FIRST_LED_PIN; pin <= LAST_LED_PIN; ++pin)
    pinMode(pin, OUTPUT);
}

void loop()
{
  // получаем время в миллисекундах, прошедшее с момента включения микроконтроллера
  unsigned int ms = millis();
  // нехитрой арифметикой вычисляем, какой светодиод должен гореть именно сейчас. Смена будет происходить каждые 120
  // миллисекунд. Y % X — это остаток от деления Y на X; плюс, минус, скобки — как в алгебре.
  int pin = FIRST_LED_PIN + (ms / 120) % 10;
  // включаем нужный светодиод на 10 миллисекунд, затем — выключаем. На следующем проходе цикла он снова включится, если
  // гореть его черёд, и мы вообще не заметим отключения
  digitalWrite(pin, HIGH);
  delay(10);
  digitalWrite(pin, LOW);
}
```

# Пояснения к коду

С помощью выражения `for` мы организуем цикл со счетчиком. В данном случае для настройки портов на выход. Чтобы сделать такой цикл, нужно:

Инициализировать переменную-счетчик, присвоив ей первоначальное значение. В нашем случае: `int pin = FIRST_LED_PIN`

Указать условие, до достижения которого будет повторяться цикл. В нашем случае: `pin <= LAST_LED_PIN`

Определить правило, по которому будет изменяться счетчик. В нашем случае `++pin` (см. ниже об операторе `++`).

Например, можно сделать цикл `for (int i = 10; i > 0; i = i - 1)`. В этом случае:

Переменной `i` присваивается значение `10`

Это значение удовлетворяет условию `i > 0`

Поэтому блок кода, помещенный в цикл, выполняется первый раз

Значение `i` уменьшается на единицу, согласно заданному правилу, и принимает значение `9`

Блок кода выполняется второй раз.

Всё повторяется снова и снова вплоть до значения `i` равного `0`

Когда `i` станет равна `0`, условие `i > 0` не выполнится, и выполнение цикла закончится

Контроллер перейдет к коду, следующему за циклом `for`

# Пояснения к коду

Помещайте код, который нужно зациклить, между парой фигурных скобок `{}`, если в нем больше одной инструкции.

Переменная-счетчик, объявляемая в операторе `for`, может использоваться внутри цикла. Например, в данном эксперименте `pin` последовательно принимает значения от 2 до 11 и, будучи переданной в `pinMode`, позволяет настроить 10 портов одной строкой, помещенной в цикл.

Переменные-счетчики видны только внутри цикла. Т.е. если обратиться к `pin` до или после цикла, компилятор выдаст ошибку о необъявленной переменной.

Конструкция `i = i - 1` в пояснении выше не является уравнением! Мы используем оператор присваивания `=` для того, чтобы в переменную `i` поместить значение, равное текущему значению `i`, уменьшенному на 1.

Выражение `++pin` — это т.н. оператор инкремента, примененный к переменной `pin`. Эта инструкция даст тот же результат, что `pin = pin + 1`

Аналогично инкременту работает оператор декремента `--`, уменьшающий значение на единицу. Подробнее об этом в статье про арифметические операции.

Тип данных `unsigned int` используют для хранения целых чисел без знака, т.е. только неотрицательных. За счет лишнего бита, который теперь не используется для хранения знака, мы можем хранить в переменной такого типа значения до 65 535.

# Пояснения к коду

Функция `millis` возвращает количество миллисекунд, прошедших с момента включения или перезагрузки микроконтроллера. Здесь мы используем ее для отсчета времени между переключениями светодиодов.

С помощью выражения  $(ms / 120) \% 10$  мы определяем, который из 10 светодиодов должен гореть сейчас. Перефразируя, мы определяем какой отрезок длиной в 120 мс идет сейчас и каков его номер внутри текущего десятка. Мы добавляем порядковый номер отрезка к номеру того порта, который в текущем наборе выступает первым.

То, что мы гасим светодиод с помощью `digitalWrite(pin, LOW)` всего через 10 мс после включения не заметно глазу, т.к. очень скоро будет вновь вычислено, какой из светодиодов включать, и он будет включен — только что погашенный или следующий.

# Вопросы для проверки себя

1. Почему в данном эксперименте мы подключаем светодиодную шкалу, не используя транзистор?
2. Если бы мы включали светодиоды только на портах 5, 6, 7, 8, 9, что нужно было бы изменить в программе?
3. С помощью какой другой инструкции можно выполнить действие, эквивалентное `++pin`?
4. В чем разница между переменными типов `int` и `unsigned int`?
5. Что возвращает функция `millis()`?
6. Как в данном эксперименте мы вычисляем номер порта, на котором нужно включить светодиод?

# Задания для самостоятельного решения

1. Измените код так, чтобы светодиоды переключались раз в секунду.
2. Не выключая порты, сделайте так, чтобы огонёк бежал только по средним четырем делениям шкалы.
3. Переделайте программу так, чтобы вместо `int pin = FIRST_LED_PIN + (ms / 120) % 10` перемещением огонька управлял цикл `for`
4. Не меняя местами провода, измените программу так, чтобы огонёк бегал в обратном направлении.



**Единственное условие, от  
которого зависит успех,  
есть терпение.**

**Толстой Лев Николаевич**