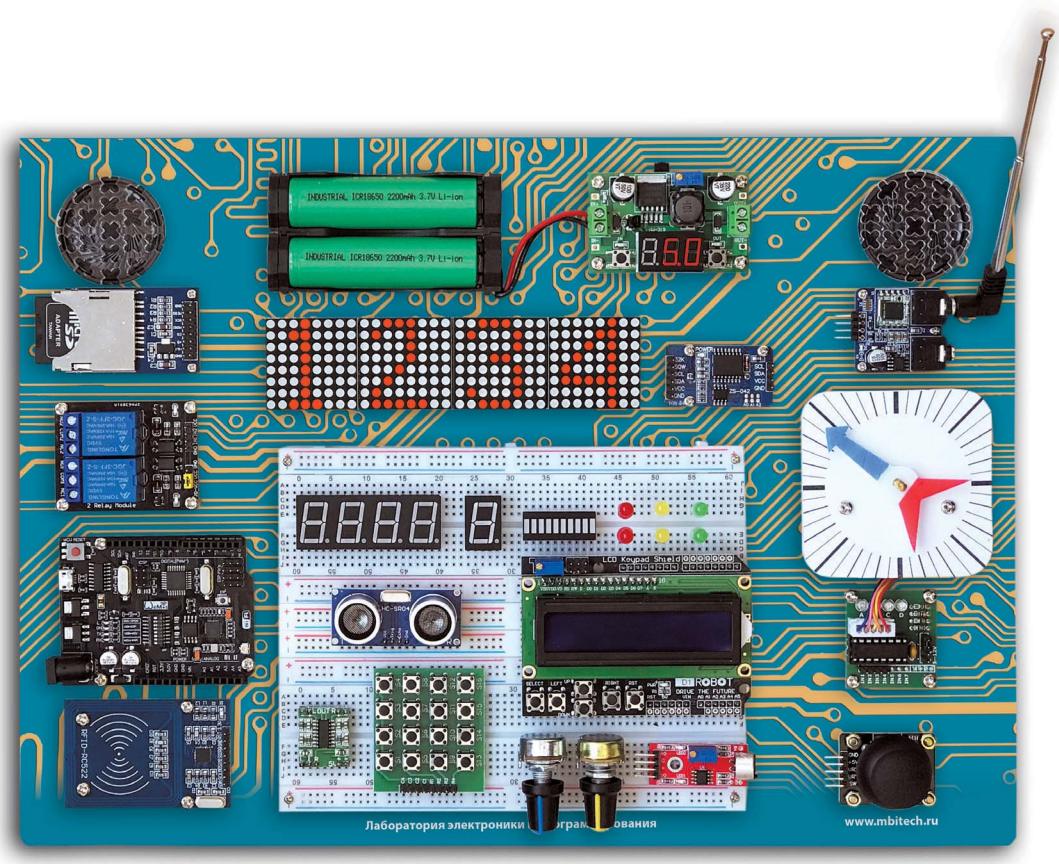
Петин В. А.

77 ПРОЕКТОВ ДЛЯ
ARDUINO



ДМК ПРЕСС
ИЗДАТЕЛЬСТВО

mbitech
ADVANCED TOOLS

УДК 681.4:004.9Arduino
ББК 32.816с515+32.965с515
П29

Петин В.А.
П29 **77 проектов для Arduino. — М. ДМК Пресс. 2020. — 356 с.: ил.**

ISBN 978-5-97060-697-1

В книге представлено 77 экспериментов для Arduino — ценнейшего практического материала для обучения. Каждый эксперимент подразумевает поэтапное изучение электроники и программирования путем создания проектов на Ардуино. Процесс обучения от начала работы до готового устройства занимает не очень много времени. В рамках нашей программы обучения можно собрать полноценный проект. Например, игру «Змейка», домашнюю метеостанцию, WEB-опросник, бегущую строку, электронные часы с будильником, FM радио, электронный компас и многое другое!

Для сборки проектов не требуется паяльник, а порог вхождения в электронику очень легкий, что соответствует нашему девизу: «Arduino — это очень просто!»

Авторы книги использовали современные методики обучения. Книгу можно использовать, как методическое пособие. Она содержит подробные иллюстрации к каждому занятию. Теория полностью совмещена с практикой и излагается постепенно, от простого к сложному, не упуская всех деталей.

УДК 681.4:004.9Arduino
ББК 32.816с515+32.965с515

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-5-97060-697-1

© ООО «ЭМБИТЕХ Групп», 2020
© Оформление, ДМК Пресс, 2020

Содержание

Введение..... 6
Установка Arduino IDE..... 8
Плата Arduino+WiFi..... 17
Проводники и плата прототипирования..... 18
Блоки питания..... 20
Эксперимент 1. Светодиодный маячок на 4 светодиодах..... 21
Эксперимент 2. Бегущий огонек на 8 светодиодах..... 25
Эксперимент 3. Бегущий огонек на 8 светодиодах – совершенствуем программу..... 29
Эксперимент 4. Десятисегментный линейный индикатор. Пульсирующая шкала..... 32
Эксперимент 5. Два светофора на перекрестке..... 36
Эксперимент 6. Подключаем к Arduino кнопку..... 40
Эксперимент 7. Боремся с дребезгом контактов кнопки..... 44
Эксперимент 8. Подключаем несколько кнопок, управляем светодиодами 48
Эксперимент 9. delay() и millis() - управляем скоростью и направлением «бегущего огня» с помощью кнопок..... 53
Эксперимент 10. Подключение 7-сегментного одноразрядного индикатора..... 58
Эксперимент 11. Матрица 4-разрядная из 7-сегментных индикаторов..... 62
Эксперимент 12. Секундомер на 4-разрядной матрице из 7-сегментных индикаторов..... 65
Эксперимент 13. Аналоговые входы Arduino. Подключение потенциометра..... 69
Эксперимент 14. Использование потенциометра в качестве регулятора показаний светодиодной шкалы 74
Эксперимент 15. Клавиатура по однопроводной аналоговой линии..... 77
Эксперимент 16. Широтно-импульсная модуляция. Балансир яркости двух светодиодов 82
Эксперимент 17. Радуга на RGB-светодиоде..... 84
Эксперимент 18. До-ре-ми-фа-соль-ля-си. Воспроизводим звуки на Arduino..... 89
Эксперимент 19. Воспроизводим звуки разных октав. Двумерные массивы..... 93
Эксперимент 20. Музыкальный звонок..... 97
Эксперимент 21. Библиотеки Arduino. Создание собственной библиотеки..... 102
Эксперимент 22. Матричная клавиатура 4x4..... 107
Эксперимент 23. Пианино на матричной клавиатуре..... 112
Эксперимент 24. ЖК-дисплей на контроллере HD44780..... 116
Эксперимент 25. Создаем калькулятор на матричной клавиатуре..... 120
Эксперимент 26. Управляем движущимся символом на экране дисплея..... 125
Эксперимент 27. 4-х разрядная светодиодная матрица..... 130
Эксперимент 28. Вывод спрайтов и символов на 4-х разрядную светодиодную матрицу..... 133

Эксперимент 29. Бегущая строка на 4-х разрядной светодиодной матрице..... 137
Эксперимент 30. Русификация «бегущей строки»
на 4-х разрядной светодиодной матрице..... 140
Эксперимент 31. Загрузка по последовательному порту текста
для "бегущей строки" на 4-х разрядной светодиодной матрице..... 144
Эксперимент 32. Подключаем двухкоординатный джойстик..... 149
Эксперимент 33. Игра «Змейка». Управляем перемещением
"змейки" на светодиодной матрице с помощью джойстика..... 154
Эксперимент 34. Игра «Змейка». Добавляем корм для "змейки"..... 161
Эксперимент 35. Игра «Змейка». Последние штрихи..... 167
Эксперимент 36. Индикатор влажности почвы на датчике FC-28..... 174
Эксперимент 37. Звуковая сигнализация превышения уровня воды..... 177
Эксперимент 38. Индикатор шума на датчике звука..... 180
Эксперимент 39. Измерение влажности и температуры воздуха датчиком DHT11..... 182
Эксперимент 40. Индикатор освещенности на датчике GY30..... 185
Эксперимент 41. Домашняя метеостанция на датчике BMP280 и DHT11..... 191
Эксперимент 42. Часы реального времени DS3231
Установка (корректировка) времени..... 196
Эксперимент 43. Часы на 4-х разрядной светодиодной матрице..... 201
Эксперимент 44. Часы с бегущей строкой на 4-х разрядной светодиодной матрице... 204
Эксперимент 45. Часы на ЖК-дисплее LCD Keypad shield..... 210
Эксперимент 46. Добавляем часам на ЖК-дисплее
LCD Keypad shield функционал будильника..... 213
Эксперимент 47. Память EEPROM. Запись в EEPROM данных для будильников..... 218
Эксперимент 48. Часы с будильниками на EEPROM..... 223
Эксперимент 49. Работа с SD-картой..... 225
Эксперимент 50. Сохранение данных метеостанции на SD-карте..... 230
Эксперимент 51. Подключение исполнительных устройств..... 234
Эксперимент 52. Подключение 4-фазного шагового двигателя..... 237
Эксперимент 53. Управление скоростью и направлением
движения 4-фазного шагового двигателя с LCD Keypad shield..... 241
Эксперимент 54. Беспроводная связь по инфракрасному каналу..... 245
Эксперимент 55. Управление скоростью и направлением движения 248
4-фазного шагового двигателя по ИК каналу.....
Эксперимент 56. Ультразвуковой датчик расстояния HC-SR04..... 252
Эксперимент 57. Радар на шаговом двигателе и датчике HC-SR04..... 255
Эксперимент 58. Компас на шаговом двигателе и модуле GY273 HMC5883..... 258
Эксперимент 59. RFID-идентификация. Считыватель RFID RC522..... 264
Эксперимент 60. Организация контроля доступа по RFID-меткам..... 268
Эксперимент 61. Запись информации на RFID-метку..... 271
Эксперимент 62. Считывание данных с RFID-метки..... 277
Эксперимент 63. Подключение модуля TEA5767..... 280
Эксперимент 64. Радиоприемник на модуле TEA5767..... 283
Эксперимент 65. Загрузка скетчей на модуль ESP8266
платы Arduino+WiFi..... 286

Эксперимент 66. Обмен данными по последовательному порту
между ESP8266 и Arduino Uno платы Arduino+WiFi..... 292
Эксперимент 67. Web-сервер с отображением данных метеостанции..... 297
Эксперимент 68. Web-сервер на ESP8266 для управления светодиодами..... 304
Эксперимент 69. Web-сервер для управления реле через Arduino 310
Эксперимент 70. Web-сервер управления текстом для бегущей строки
на 4-х разрядной светодиодной матрице..... 314
Эксперимент 71. Домашняя метеостанция для сервиса Народный мониторинг..... 319
Эксперимент 72. Отправка данных датчиков домашней метеостанции
на сайт Народного мониторинга 326
Эксперимент 73. Прием на устройстве команд , отправленных
с сайта Народного мониторинга 331
Эксперимент 74. Обработка и исполнение команд, полученных
с сайта Народный мониторинг..... 335
Эксперимент 75. Протокол MQTT. Отправка данных по протоколу MQTT..... 340
Эксперимент 76. Получение данных по протоколу MQTT..... 347
Эксперимент 77. Отправляем с web-сервера в интернет-магазин
Arduino-Kit отзывы и пожелания о книге и наборе..... 352

Эксперимент 1.
Светодиодный маячок на 4 светодиодах

Здесь мы познакомимся с платой Arduino,
узнаем о режимах работы цифровых контактов,
научимся правильно подключать к Arduino светодиоды
и создадим первую программу
светодиодного маячка на 4 светодиодах.

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino + WiFi – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Светодиод красный - 4;
- ❑ Резистор 220 Ом – 4;
- ❑ Провода ММ – 5.

ПЕРЕКЛЮЧАТЕЛЬ						
1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Итак, установите переключатели на плате Arduino+WiFi следующим образом:

В результате получаем обычную Arduino UNO. Данное положение переключателей мы и будем использовать в большинстве экспериментов.

Arduino UNO представляет собой плату, с размещенными на ней компонентами, главным из которых является микроконтроллер Atmel AVR. Он является основной вычислительной системой этой платформы, поскольку именно для него и создается программное обеспечение, с помощью которого микроконтроллер взаимодействует с внешним миром посредством специальных портов ввода/вывода данных. Плата Arduino UNO имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Питание платы Arduino UNO при помощи адаптера AC/DC (рекомендуемое напряжение 7-12В), либо от компьютера посредством кабеля USB. Микроконтроллер ATmega328 располагает 32 кБ флэш-памяти, из которых 0.5 кБ используется для хранения загрузчика, а также 2 кБ ОЗУ (SRAM) и 1 Кб энергонезависимой памяти EEPROM.

Цифровые выводы на платах Arduino позволяют подключать к Arduino датчики, приводы и другие микросхемы. Изучение того, как использовать их, позволит вам использовать Arduino для выполнения практических полезных вещей.

Цифровые сигналы имеют только два отдельных значения: высокий (HIGH, 1) и низкий (LOW, 0) уровни. Вы можете использовать цифровые сигналы в ситуациях, где вход или выход будет принимать одно из этих двух значений. Поскольку цифровые выводы Arduino могут использоваться в качестве и входа, и выхода, сначала необходимо их настроить. Для настройки цифровых выводов в Arduino используется встроенная функция pimMode(), которая имеет следующий синтаксис:

pinMode(pin, mode)

где

- pin – номер вывода Arduino;
- mode – устанавливаемый режим для вывода pin:
  - INPUT – pin в режиме входа;
  - OUTPUT – pin в режиме выхода;
  - INPUT_PULLUP – в этом режиме к выводу подключается внутренний подтягивающий резистор 20 кОм, чтобы привести уровень на выводе к значению HIGH, если к нему ничего не подключено.

В данном эксперименте мы будем использовать выводы Arduino в режимы выходов (OUTPUT), для включения и выключения светодиодов. Светодиод – это полупроводниковый прибор, преобразующий электрический ток непосредственно в световое излучение. Цветовые характеристики светодиодов зависят от химического состава использованного в нем полупроводника. Светодиод излучает в узкой части спектра, его цвет чист, что особенно ценят дизайнеры. Светодиоды поляризованы, имеет значение, в каком направлении подключать их. Положительный вывод светодиода (более длинный) называется анодом, отрицательный – катодом. Как и все диоды, светодиоды позволяют току течь только в одном направлении – от анода к катоду. Поскольку ток протекает от положительного к отрицательному, анод светодиода должен быть подключен к цифровому сигналу, а катод должен быть подключен к земле.

Собираем схему согласно рис. 1.1.

В схеме подключения светодиодов к цифровым выходам мы используем ограничительный резистор номиналом 220 Ом. Рассмотрим, как подобрать ограничительный резистор и как будет влиять номинал резистора на яркость светодиода.

Самым главным уравнением для любого инженера-электрика является закон Ома. Закон Ома определяет отношения между напряжением, током и сопротивлением в цепи. Закон Ома определяется следующим образом:

$$V = I \times R,$$

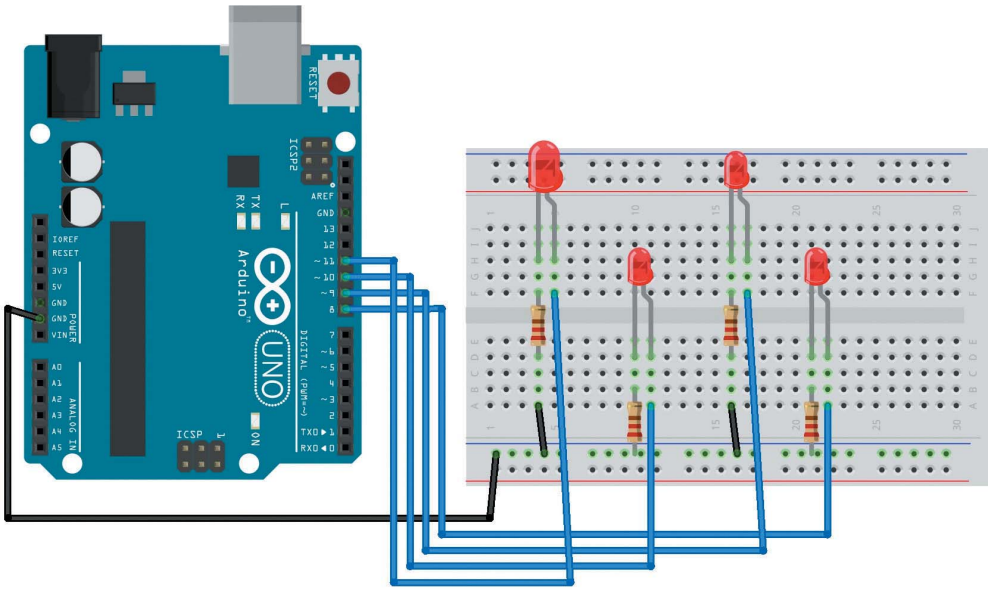


Рис. 1.1. Схема соединений для эксперимента

где V – напряжение в вольтах; I – ток в амперах; R – сопротивление в омах.

В электрической схеме каждый компонент имеет некоторое сопротивление, что снижает напряжение. Светодиоды имеют предопределенное падение напряжения на них и предназначены для работы в определенном значении тока. Чем больше ток через светодиод, тем ярче светодиод светится, до предельного значения. Для наиболее распространенных светодиодов максимальный ток составляет 20 мА. Обычное значение падения напряжения для светодиода – около 2 В. Напряжение питания 5 В должно упасть на светодиоде и резисторе, поскольку доля светодиода 2 В оставшиеся 3 В должны упасть на резисторе. Зная максимальное значение прямого тока через светодиод (20 мА), можете найти номинал резистора.

$$R = V/I = 3/0{,}02 = 150 \text{ Ом}$$

Таким образом, со значением резистора 150 Ом ток 20 мА протекает через резистор и светодиод. По мере увеличения значения сопротивления ток будет уменьшаться. 220 Ом немного более, чем 150 Ом, но все же позволяет светиться светодиоду достаточно ярко, и резистор такого номинала очень распространен.

Приступим к написанию программы (скетча).

Светодиоды должны одновременно мигать с определенной частотой. В процедуре setup() настроим режим работы контактов (пинов), к которым подключены светодиоды, как OUTPUT (выход)

```
void setup() {
   pinMode(8,OUTPUT);
   pinMode(9,OUTPUT);
   pinMode(10,OUTPUT);
   pinMode(11,OUTPUT);
}
```

В процедуре loop() сначала зажигаем все светодиоды (подаем на выводы 8,9,10,11 сигнал HIGH), ждем некоторое время (время "горения"), затем "тушим" светодиоды (подаем на выводы 8,9,10,11 сигнал LOW) и опять ждем некоторое время и т.д. по кругу. Для выполнения операции подождать "некоторое время" мы будем использовать встроенную Arduino функцию delay(). Эта функция просто останавливает выполнение программы на заданное в параметре количество миллисекунд (1000 миллисекунд в 1 секунде). Например:

delay(2000); // останавливает выполнение программы на 2000 мсек (

И весь код программы в листинге 1.1. Скачать данный скетч можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_01_01.

Листинг 1.1

```
void setup() {
   // настроить выводы 8, 9, 10, 11 Arduino как OUTPUT
   pinMode(8,OUTPUT);
   pinMode(9,OUTPUT);
   pinMode(10,OUTPUT);
   pinMode(11,OUTPUT);
}

void loop() {
  // включить светодиоды
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(11, HIGH);
  // пауза 1000 мсек (1 сек)
  delay(1000);
  // выключить светодиоды
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  // пауза 1000 мсек (1 сек)
  delay(1000);
}
```

Загрузим данный скетч на плату Arduino. Вы должны наблюдать включение/выключение светодиодов с частотой 2 секунды.

Эксперимент 2.
Бегущий огонек на 8-ми светодиодах

В данном эксперименте мы создадим "бегущий огонек" на ленте из 8-ми светодиодов, научимся создавать собственные функции для сокращения кода скетча.

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Светодиод – 8;
- ❑ Резистор 220 Ом – 8;
- ❑ Провода ММ – 5.

Переключатели на плате Arduino+WiFi установите следующим образом:

ПЕРЕКЛЮЧАТЕЛЬ						
1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

Собираем схему согласно рис. 2.1

Приступим к написанию программы (скетча).

В процедуре setup() настроим режим работы контактов (пинов), к которым подключены светодиоды, как OUTPUT (выход)

```
void setup() {
   pinMode(5,OUTPUT);
   pinMode(6,OUTPUT);
   pinMode(7,OUTPUT);
   pinMode(8,OUTPUT);
   pinMode(9,OUTPUT);
   pinMode(10,OUTPUT);
   pinMode(11,OUTPUT);
   pinMode(12,OUTPUT);
}
```

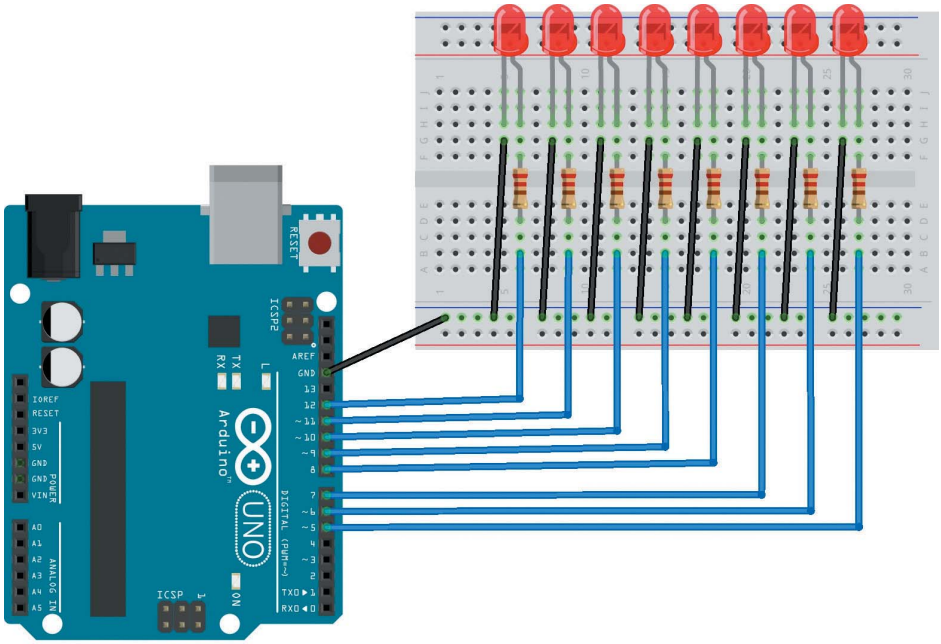


Рис. 2.1. Схема соединений для «бегущего огня» на 8 светодиодах

Алгоритм основного цикла следующий:

1. “гасим” все 8 светодиодов;
2. включаем первый светодиод;
3. включаем в скетче продолжительную паузу, чтобы увидеть горение одного светодиода;
4. повторяем шаги 1-3 для второго, третьего и далее до восьмого светодиода;
5. далее возврат в начало цикла loop().

Сокращенный код цикла loop() представлен в листинге 2.1.

Листинг 1.1

```
void loop() {
  // выключить 8 светодиодов
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  digitalWrite(12, LOW);
  // включить 1 светодиод
```

```
  digitalWrite(5, HIGH);
  // пауза 1000 мсек
  delay(1000);
  // выключить 8 светодиодов
  ….
  // включить 2 светодиод
  digitalWrite(6, HIGH);
  // пауза
  delay(1000);
  // выключить 8 светодиодов
  ….
  // включить 3 светодиод
  digitalWrite(7, HIGH);
  // пауза
  delay(1000);
  …….
  …….
  // выключить 8 светодиодов
  ….
  // включить 8 светодиод
  digitalWrite(12, HIGH);
  // пауза
  delay(1000);
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_02_01.**

Загрузим данный скетч на плату Arduino. Вы должны наблюдать бугущий огонек, перемещающийся по светодиодам с 1 по 8.

При том, что скетч работает верно, у него есть большой недостаток – он очень громоздкий. Рассмотрим, как можно код значительно уменьшить с помощью функций. Функция – это именованная последовательность операций. Причиной создания функции является необходимость выполнять одинаковое действие несколько раз. В цикле loop() восемь раз встречается совершенно одинаковый код для выключения 8 светодиодов. Создадим функцию ledoff() (выключить светодиоды). Функция создается за скобками функций setup(0 и loop(). Создание функции:

```
void ledsOff() {
     // код функции
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  digitalWrite(12, LOW);
}
```

И тогда цикл loop() нашего скетча сократится следующим образом (листинг 2.2)

Листинг 2.2

```
void loop() {
  // выключить 8 светодиодов
  ledsOff();
  // включить 1 светодиод
  digitalWrite(5, HIGH);
  // пауза 1000 мсек
  delay(1000);
  //
  ledsOff();
  digitalWrite(6, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(7, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(8, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(9, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(10, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(11, HIGH);
  delay(1000);
  //
  ledsOff();
  digitalWrite(12, HIGH);
  delay(1000);
}
```

Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_02_02.

Как видно, ввведение функции значительно сократило программу. Здесь мы рассмотрели простейший пример функции, в дальнейшем научимся передавать в функцию параметры и возвращать из функции в программу значения.

Эксперимент 3.
Бегущий огонек на 8 светодиодах – совершенствуем программу

Сегодня мы усовершествуем скетч "бегущего огня" на ленте из 8 светодиодов, научимся создавать константы и переменные и использовать их в программе.

В эксперименте мы будем использовать компоненты из эксперимента 2, но собрав схему соединений рис. 3.1.

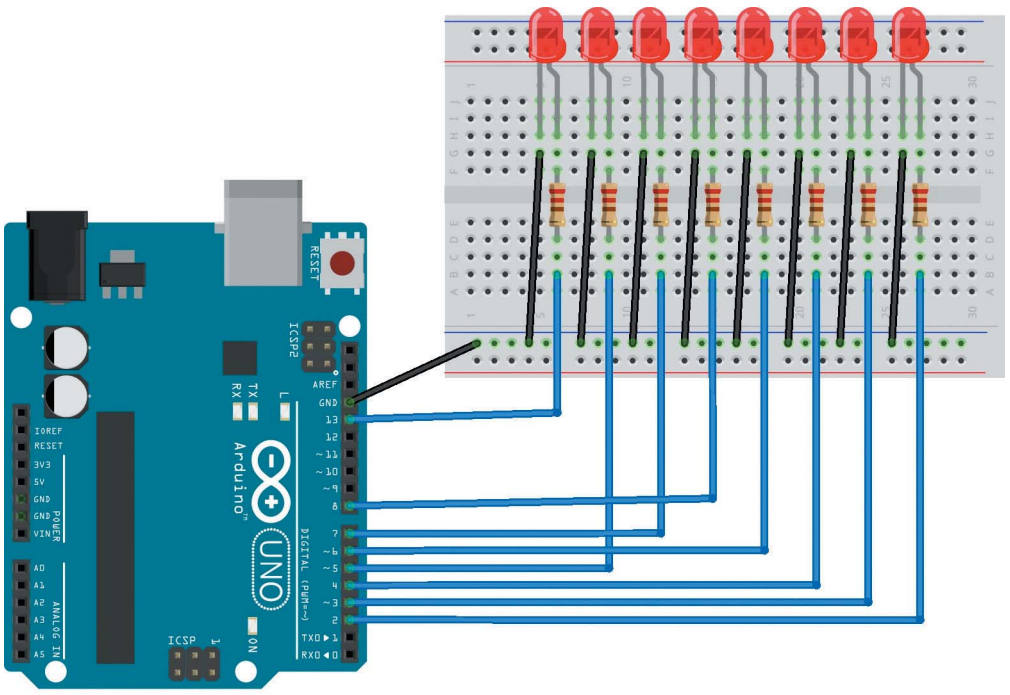


Рис. 3.1. Схема соединений для «бегущего огня» на 8 светодиодах

У нас есть работающий скетч **https://arduino-kit.ru/scetches/exp_02_02.**

Нам необходимо только изменить контакты для подключения светодиодов (например, с 5,6,7,8,9,10,11,12 на 2,3,4,6,8,7,5,13).

Можно сделать изменения в скетче, заменив значения 5,6,7,8,9,10,11,12 на 2,3,4,6,8,7,5,13 соответсвенно. Это 24 правки значений в скетче. В данном случае мы имеем дело с довольно простой программой и проделать изменения не так сложно. Но что если устройство довольно сложное и скетч расписан на сотни строк? В этом случае сделать все изменения правильно, не ошибиться и не пропустить ни одного изменения становится крайне сложно: мы люди и нам свойственно ошибаться из-за невнимательности.

Но можно поступить иначе. Можно каждому пину назначить понятное имя, которое потом используется для обращения:

```
#define LED1      2
#define LED2      3
#define LED3      4
#define LED4      6
#define LED5      8
#define LED6      7
#define LED7      5
#define LED8      13
```

И далее в скетче, например:

```
void setup() {
   pinMode(LED1,OUTPUT);
   pinMode(LED2,OUTPUT);
   pinMode(LED3,OUTPUT);
   pinMode(LED4,OUTPUT);
   pinMode(LED5,OUTPUT);
   pinMode(LED6,OUTPUT);
   pinMode(LED7,OUTPUT);
   pinMode(LED8,OUTPUT);
}
```

Конструкция #define называется макроопределением. Она говорит компилятору о том, что всякий раз, когда он видит указанное имя, стоит использовать на этом месте указанное значение.

Аналогично делаем изменения во всем скетче.

Скачать измененный скетч целиком можно на сайте Arduino-kit по ссылке **https://arduino-kit.ru/scetches/exp_03_01.**

Загрузим скетч на плату Arduino. Вы должны наблюдать бугущий огонек, перемещающийся по светодиодам с 1 по 8, подключенным по схеме на рисунке 3.1.

Все, теперь при изменении контактов подключения светодиодов (например, обратный переход к схеме соединений 2.1) достаточно изменить нужные значения в начале программы и не думать об изменениях в самом скетче.

Макроопределения хороши для именования значений, которые не могут измениться по ходу выполнения скетча. Но что делать, если какие-то параметры должны изменяться при выполнении программы? Для этого существуют переменные.

Переменная – это место хранения данных. Каждая переменная имеет имя, значение и тип. Например, объявление

int t = 1000;

создает переменную с именем t, значением 1000 и типом int (целое число в интервале -215 до 215). Затем, в программе имеется возможность обратиться к данной переменной через имя с целью работы с ее значением. Например:

t=t+100;

В качестве примера использования переменной напишем скетч «бегущего огня», в котором скорость движения (параметр в функции delay()) будет уменьшаться каждый цикл на 100 мсек до бесконечности. Изменение содержимого скетча показано в листинге 3.1.

Листинг 3.1.

```
int t=500;

void loop() {
  // выключить 8 светодиодов
  ledsOff();
  // включить 1 светодиод
  digitalWrite(LED1, HIGH);
  // пауза
  delay(t);
  //
  ledsOff();
  digitalWrite(LED2, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED3, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED4, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED5, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED6, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED7, HIGH);
  delay(t);
  //
  ledsOff();
  digitalWrite(LED8, HIGH);
  delay(t);
  t=t+100;
}
```

Эксперимент 4.
Десятисегментный линейный индикатор. Пульсирующая шкала

В данном эксперименте мы создадим "пульсирующую шкалу" на десятисегментном линейном индикаторе. При программировании познакомимся с массивами и научимся эффективно использовать массивы с помощью операторов for и if … else .

В эксперименте мы будем использовать следующие компоненты:

- ❑ Плата Arduino UNO – 1;
- ❑ Кабель USB;
- ❑ Плата прототипирования – 1;
- ❑ Светодиод – 8;
- ❑ Резистор 220 Ом – 8;
- ❑ Провода ММ – 5.

1	2	3	4	5	6	7
OFF	*OFF*	*ON*	*ON*	*OFF*	*OFF*	*OFF*

10-разрядная линейная светодиодная шкала представляет собой сборку из 10 независимых светодиодов с катодами со стороны надписи на корпусе. Создадим на этой сборке проект "пульсирующей шкалы". Количество одновременно горящих светодиодов шкалы будет меняться от 0 до 8, затем опять с 0. Для подключения шкалы к Arduino будем использовать 10 цифровых выводов D3–D12. Схема соединений показана на рис. 4.1. Каждый из светодиодов шкалы выводом анода соединен с цифровым выводом Arduino, а катодом на землю через последовательно соединенный ограничивающий резистор 220 Ом.

Приступим к написанию программы (скетча). Определим константы для пинов подключения светодиодов в сборке и в процедуре setup() настроим режим работы контактов (пинов), к которым подключены светодиоды, как OUTPUT (выход) (см. листинг 4.1)

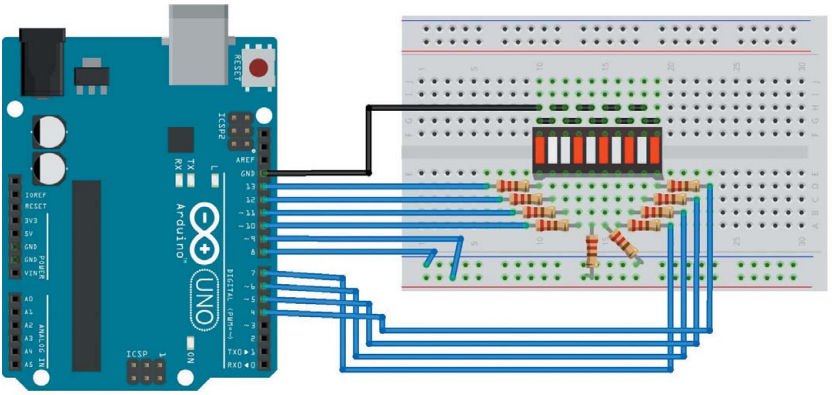


Рис. 4.1. Схема соединений для 10-разрядной линейной светодиодной шкалы

Листинг 4.1

```
#define LED1       13
#define LED2       12
#define LED3       11
#define LED4       10
#define LED5       9
#define LED6       8
#define LED7       7
#define LED7       6
#define LED8       5
#define LED8       4
void setup() {
   pinMode(LED1,OUTPUT);
   pinMode(LED2,OUTPUT);
   pinMode(LED3,OUTPUT);
   pinMode(LED4,OUTPUT);
   pinMode(LED5,OUTPUT);
   pinMode(LED6,OUTPUT);
   pinMode(LED7,OUTPUT);
   pinMode(LED8,OUTPUT);
   pinMode(LED9,OUTPUT);
   pinMode(LED10,OUTPUT);
}
```

И теперь самое время познакомиться с массивами. Массив - это набор переменных, доступ к которым осуществляется через их индекс. Рассмотрим варианты создания (объявления) массива.

Объявление массива без его инициализации:

```
int leds[10];         // массив из 10 элементов типа int
```

Можно объявить массив без непосредственного указания размера. Компилятор считает количество элементов и создает массив соответствующего размера:

```
int leds [] = {13,12,11,10,9,8,7,6,5,4};
```

Можно одновременно инициализировать и указать размер вашего массива:

```
int pin_leds [10] = {13,12,11,10,9,8,7,6,5,4};
```

Теперь рассмотрим доступ к элементам массива. Индексация в массивах начинается с нуля. То есть, первый элемент массива будет иметь порядковый номер 0. Таким образом:

```
pin_leds[0] = 13;
pin_leds[1] = 12;
```

Тогда код из листинга 4.1. можно записать в виде, представленном в листинге 4.2.

Листинг 4.2

```
int leds[] = {13,12,11,10,9,8,7,6,5,4};

void setup() {
   pinMode(leds[0],OUTPUT);
   pinMode(leds[1],OUTPUT);
   pinMode(leds[2],OUTPUT);
   pinMode(leds[3],OUTPUT);
   pinMode(leds[4],OUTPUT);
   pinMode(leds[5],OUTPUT);
   pinMode(leds[6],OUTPUT);
   pinMode(leds[7],OUTPUT);
   pinMode(leds[8],OUTPUT);
   pinMode(leds[9],OUTPUT);
}
```

Пока большого выигрыша в уменьшении написания кода мы не видим. Очень удобный метод работы с массивами - циклы. В этих случаях счетчик цикла используется для индексации каждого элемента массива. Конструкция for используется для повторения блока операторов, заключенных в фигурные скобки. Счетчик приращений обычно используется для приращения и завершения цикла. Пример 10-кратного выполнения операторов в цикле for:

```
int i=0;
for (; i < 10;) {
  // список операторов
  …..
  i=i+1;
}
```

или

```
for (int i=0; i < 10; i=i+1) {
  // список операторов
  …..
}
```

Оператор for подходит для любых повторяющихся действий и часто используется для перебора элементов массива. Например, код из листинга 4.2 будет выглядеть так:

```
int leds[] = {13,12,11,10,9,8,7,6,5,4};
void setup() {
   for (int i=0; i < 10; i=i+1) {
      pinMode(leds[i],OUTPUT);
   }
}
```

Теперь рассмотрим оператор if … else , который используется в сочетании с операторами сравнения, проверяет, достигнута ли истинность условия в if,

и выполняет блок операторов, заключенных в фигурные скобки, если условие не выполнено, выполняется блок операторов, заключенных в фигурные скобки else:

```
if(x<6) {
  // список операторов
   ……
}
else {
  // список операторов
   ……
}
```

Для включения n светодиодов из массива:

```
   for (int i=0;i<10; i=i+1) {
      if(i<3)
         {digitalWrite(leds[i],HIGH);}
      else
         {digitalWrite(leds[i],LOW);}
   }
```

И полный код программы показан в листинге 4.3. Скачать данный скетч целиком можно на сайте Arduino-kit по ссылке https://arduino-kit.ru/scetches/exp_04_03.

Листинг 4.3

```
// массив контактов подключения светодиодов шкалы
int leds[] = {13,12,11,10,9,8,7,6,5,4};
// количество горящих светодиодов
int n=0;

void setup() {
   // настроить выводы Arduino как OUTPUT
   for (int i=0;i < 10; i=i+1) {
      pinMode(leds[i],OUTPUT);
   }
}

void loop() {
  // выключить n светодиодов
   for (int i=0;i<10; i=i+1) {
      if(i<n)
         {digitalWrite(leds[i],HIGH);}
      else
         {digitalWrite(leds[i],LOW);}
   }
  // изменить n - приращение на 1
  // и остаток от деления на 11 (после 10 -> 0)
  n=(n+1)%11;
}
```

Загружаем скетч на плату Arduino и наблюдаем пульсирующую шкалу светодиодов.