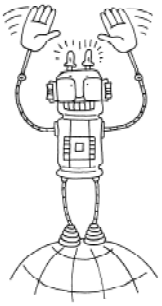
Яркоев Константин Евгеньевич

ПРАКТИЧЕСКАЯ МИКРОЭЛЕКТРОНИКА. ЗАНЯТИЕ 3. СВЕТОФОР

Это третья статья из цикла «Сделай Сам», в которой мы продолжим знакомство с Ардуино, написав программу «светофор».



ПРОЕКТ 1. РЕФАКТОРИНГ HELLO, WORLD!

В прошлый раз мы поприветствовали мир, помигав светодиодом. Код программы см. в листинге 1.

Данный код можно улучшить. Улучшение не всегда означает ускорение работы программы или уменьшение ее размера. Можно, например, улучшить читаемость кода, повысить удобство его восприятия и поддержки, то есть провести так называемый *рефакторинг кода*.

Легко заметить, что, если потребуется поменять номер вывода с 13 на 12 (или любой другой), необходимо будет внести изменения сразу в трех местах. Можно улучшить код, введя переменную **ledPin**. Теперь, чтобы поменять номер вывода, необходимо внести изменение в одном месте (см. листинг 2).

Можно заметить, что в цикле **loop** две строчки кода повторяются два раза, разница лишь в параметре **HIGH** и **LOW**. Если мы захотим изменить период мигания, то нам потребуется внести изменения сразу в двух ме-

Листинг 1

```
/*
  Blink.
  Включает светодиод на секунду, затем выключает на секунду в цикле.
*/


// Инициализация. Метод вызывается только 1 раз, когда стартует скетч, после подачи питания
// или после сброса платы. Используется для инициализации переменных, определения режимов
// работы выводов, запуска используемых библиотек
void setup() {
  pinMode(13, OUTPUT); // назначить 13-й вывод как выход
}

// Бесконечный цикл. После выполнения setup(), данный метод вызывается каждый раз после
// завершения последнего оператора в цикле
void loop() {
  digitalWrite(13, HIGH);   // включить светодиод на 13 выводе
  delay(1000);              // подождать 1 секунду = 1000 миллисекунд
  digitalWrite(13, LOW);    // выключить светодиод на 13 выводе
  delay(1000);              // подождать 1 секунду = 1000 миллисекунд
}
```

Листинг 2

```
byte ledPin =  13; // Глобальная переменная. Используется, чтобы хранить номер вывода

// Инициализация.
void setup() {
  pinMode(ledPin, OUTPUT); // назначить ногу ledPin как выход
}

// Бесконечный цикл.
void loop() {
  digitalWrite(ledPin, HIGH);   // включить светодиод на ноге ledPin
  delay(1000);              // подождать 1 секунду = 1000 миллисекунд
  digitalWrite(ledPin, LOW);    // выключить светодиод на ноге ledPin
  delay(1000);              // подождать 1 секунду = 1000 миллисекунд
}
```

стах. Можно улучшить код, введя переменную **ledState** (листинг 3).

Тип данных **boolean** имеет всего два значения: true (истина) и false (ложь). Данный тип данных поддерживает операции сравнения (==, !=), отрицания (!), логическое И (&&), логическое ИЛИ (||). В языке программирования Arduino ключевое слово **HIGH** эквивалентно true, а **LOW** эквивалентно false. В первой строчке цикла **loop** происходит изменение значения логической переменной на противоположное. Если **ledState** был равен **HIGH**, то станет **LOW**, и наоборот. Таким образом, мы улучшили нашу программу: чтобы изменить период мигания, необходимо внести изменения всего в одном месте.

УСКОРЕНИЕ HELLO, WORLD!

Итак, мы произвели рефакторинг кода. Однако в нашей программе есть один крупный недостаток: мы фактически искусственно занизили частоту цикла **loop** до 1 Гц, то есть почти в 16 000 000 раз! Ведь **delay(1000)** означает не производить никаких операций (кроме прерываний) в течение 1 секунды. Через одну секунду, то есть 16 млн. тактов, микропроцессор закончит выполнение **delay(1000)**, выйдет из этой функции, выйдет из loop и снова зайдет в цикл **loop**, чтобы продолжить выполнение последовательности операторов сначала.

Все остальные операторы выполняются всего за несколько тактов, поэтому лучше из-

Листинг 3

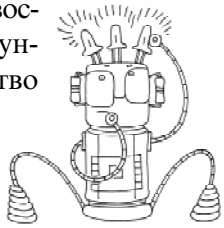
```
byte ledPin =  13;   // Используется, чтобы хранить номер вывода
boolean ledState = HIGH; // Используется, чтобы хранить текущее значение светодиода ВКЛ/ВЫКЛ

// Инициализация.
void setup() {
  pinMode(ledPin, OUTPUT); // назначить ногу ledPin как выход
}

// Бесконечный цикл.
void loop() {
  ledState = !ledState;  // присвоить переменной ledState противоположное значение
  digitalWrite(ledPin, ledState);   // задать на ноге ledPin значение ledState
  delay(1000);  // подождать 1 секунду = 1000 миллисекунд
}
```

Листинг 4

```
byte ledPin =  13;   // Используется, чтобы хранить номер вывода
boolean ledState = HIGH; // Используется, чтобы хранить текущее значение состояния светодиода ВКЛ/ВЫКЛ
long previousTimeStamp = 0; // Будет хранить время последнего изменения состояния светодиода

// Инициализация.
void setup() {
  pinMode(ledPin, OUTPUT); // назначить ногу ledPin как выход
}

// Бесконечный цикл.
void loop() {
  // Объявляем локальную переменную currentTimeStamp, и инициализируем ее значением функции millis()
  // Функция millis() возвращает количество миллисекунд с момента начала выполнения текущей программы на плате Arduino
  long currentTimeStamp = millis();

  // проверяем, а не изменить ли нам напряжение на светодиоде? Меняем, только если текущее время в мс отличается
  // от времени последнего изменения больше чем на 1000 мс
  if( currentTimeStamp - previousTimeStamp > 1000 )
  {
    previousTimeStamp = currentTimeStamp; // запоминаем текущее время как время последнего изменения
    ledState = !ledState; // присвоить переменной ledState противоположное значение (воскл. знак означает отрицание)
    digitalWrite(ledPin, ledState); // подождать 1 секунду = 1000 миллисекунд
  }
}
```

бавиться от «узкого горлышка» – функции **delay**. Ведь кто-то другой возможно захочет использовать нашу программу, добавив в неё, например, опрос датчика. Датчик сработает, но микроконтроллер сможет обработать это событие только через секунду или вовсе не узнает про это событие, если оно длилось меньше секунды.

Чтобы убрать узкое горлышко, воспользуемся функцией **millis()**. Функция **millis()** возвращает количество миллисекунд с начала выполнения текущей программы на Arduino. Введем еще одну глобальную переменную **previousTimeStamp** и будем запоминать в ней время последнего изменения состояния светодиода. Если между текущим временем и **previousTimeStamp** прошла 1 секунда, то надо поменять состояние светодиода и обновить **previousTimeStamp** (листинг 4).



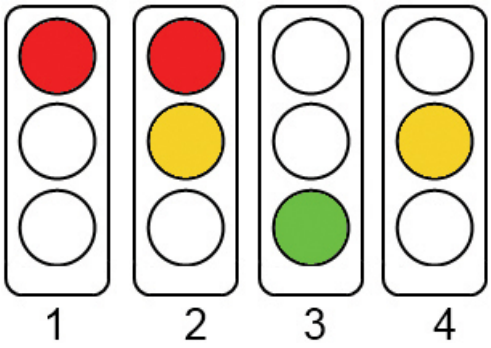


Рис. 1. Состояния светофора

Теперь внутри цикла **loop** нет вызова метода **delay**, значит, мы можем оперативно обрабатывать внешние события, например, нажатие на кнопку.

ПРОЕКТ 2. СВЕТОФОР

Мы уже научились мигать одним светодиодом, теперь попробуем помигать тремя: красного, желтого и зеленого цветов, чтобы получился светофор.

«ПРЕДПОЛЕТНАЯ ПОДГОТОВКА»

Проверьте наличие следующих деталей:

– плата Arduino UNO или ее аналога, USB-кабель;

– макетная плата, соединительные провода,

– светодиоды красного, желтого и зеленого цветов,

– 3 резистора на 150–300 Ом.

«ПОЛЕТ»

Самый простой светофор состоит из четырех состояний, представленных на рис. 1–3 (случай с мигающим зеленым рассмотрим потом):

1. Красный.
2. Красный + желтый.
3. Зеленый.
4. Желтый.

Главный цикл программы описывает переключения между состояниями светофора и время, затраченное на каждое состояние. В первом состоянии будет гореть только красный цвет, все остальные выключены. Во втором состоянии к красному добавляется жёлтый цвет. В третьем – горит только зеленый, в четвертом только жёлтый (листинг 5).

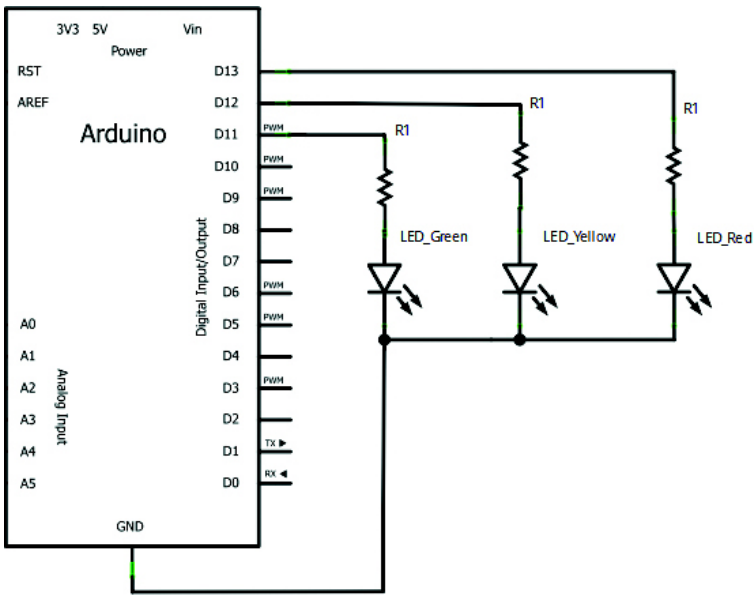


Рис. 2. Принципиальная схема

Листинг 5

```
int ledDelay = 3000; //задержка между переключениями 3 сек.

byte redPin = 13;    //номер вывода для красного светодиода
byte yellowPin = 12; //номер вывода для желтого светодиода
byte greenPin = 11;  //номер вывода для зеленого светодиода

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}

void loop() {
  //состояние 1
  digitalWrite(greenPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(redPin, HIGH);
  delay(ledDelay);
  //состояние 2
  digitalWrite(yellowPin, HIGH);
  delay(ledDelay);
  //состояние 3
  digitalWrite(greenPin, HIGH);
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, LOW);
  delay(ledDelay);
  //состояние 4
  digitalWrite(yellowPin, HIGH);
  digitalWrite(greenPin, LOW);
  delay(ledDelay);
}
```

«РАЗБОР ПОЛЕТА»

Как вы уже заметили, в программе снова используется функция **`delay()`**. Воспользуемся функцией **`millis()`**, чтобы убрать «узкое горлышко» (листинг 6).

Аналогично конструкции **`if`** конструкция **`switch`**...**`case`** позволяет

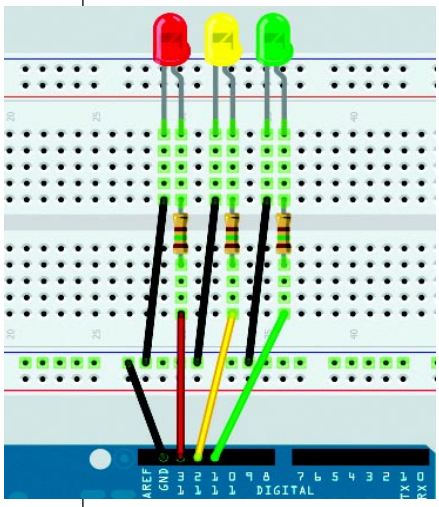


Рис. 3. Макетная плата

Листинг 6

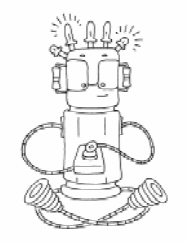
```
int ledDelay = 3000;
byte redPin = 13;
byte yellowPin = 12;
byte greenPin = 11;
byte state = 1;        //переменная, хранящая текущее состояние светофора
long previousTime = 0; //время последней смены состояния

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  setColors(true, false, false);
}

void loop() {
  long time = millis();                    //текущее время
  long difference = time - previousTime;  //время, прошедшее с момента последней смены состояния
  if( difference > ledDelay )  {          //если прошло достаточно времени, то...
    previousTime = time;                  // запомнить текущее время
    switch(state) {                       // и проверить, в каком состоянии находится светофор
      case 1:  //если в 1, то переключить в 2
          state = 2;
          setColors(true, false, false);
        break;
      case 2:  //если в 2, то переключить в 3
          state = 3;
          setColors(true, true, false);
        break;
      case 3:  //если в 3, то переключить в 4
          state = 4;
          setColors(false, false, true);
        break;
      case 4:  //если в 4, то переключить в 1
          state = 1;
          setColors(false, true, false);
        break;
    }
  }
}
//функция непосредственно меняет цвета светофора
void setColors(boolean isRed, boolean isYellow, boolean isGreen){
  digitalWrite(redPin, isRed);
  digitalWrite(yellowPin, isYellow);
  digitalWrite(greenPin, isGreen);
}
```

описывать ветвления в программе в зависимости от различных условий. Оператор **switch** сравнивает значение переменной со значением, определенным в операторах **case**. Когда найден оператор **case**, значение которого равно значению переменной, выполняется соответствующее ветвление кода. Ключевое слово **break** является командой выхода из оператора **case** и обычно используется в конце каждого **case**.

Пример:

```
switch (var) {
  case 1:
   //выполняется, когда var равно 1
     break;
  case 2:
   //выполняется, когда var равно 2
     break;
  default:
   // выполняется, если не выбрана
   //ни одна альтернатива
  }
```

Примечание. Вы можете самостоятельно добавить пятое состояние (между третьим и четвертым) – мигающий зеленый.



ПРОЕКТ 3. ИНТЕРАКТИВНЫЙ СВЕТОФОР

Настало время улучшить наш светофор, добавив сигналы для пешеходов, а также кнопку, нажав на которую пешеход сможет заставить загореться зеленый для перехода дороги.

«ПРЕДПОЛЕТНАЯ ПОДГОТОВКА»

Проверьте наличие следующих деталей:

– плата Arduino UNO или ее аналога, USB-кабель,

– макетная плата, соединительные провода,

– светодиод красного цвета – 2 шт.,

– светодиод зеленого цвета – 2 шт.,

– светодиод желтого цвета – 1 шт.,

– резистор на 150 Ом – 5 шт.

«ПОЛЕТ»

Рис. 4–5, листинг 7.

«РАЗБОР ПОЛЕТА»

В первой статье «Основы на пальцах» подробно рассказывалось про подтяжку выводов микроконтроллера до +5В через резистор. Это необходимо чтобы при считывании логических уровней иметь устойчивый уровень напряжения, когда ключ разомкнут (рис. 6).

Подтяжка вывода, работающего как цифровой вход в Arduino, происходит следующим образом:

```
pinMode(pin, INPUT);
//настраиваем порт как цифровой вход
digitalWrite(pin, HIGH);
//подтяжка до +5В
```

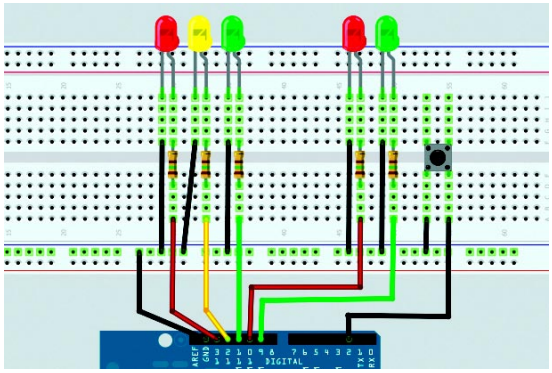


Рис. 4. Платы Arduino

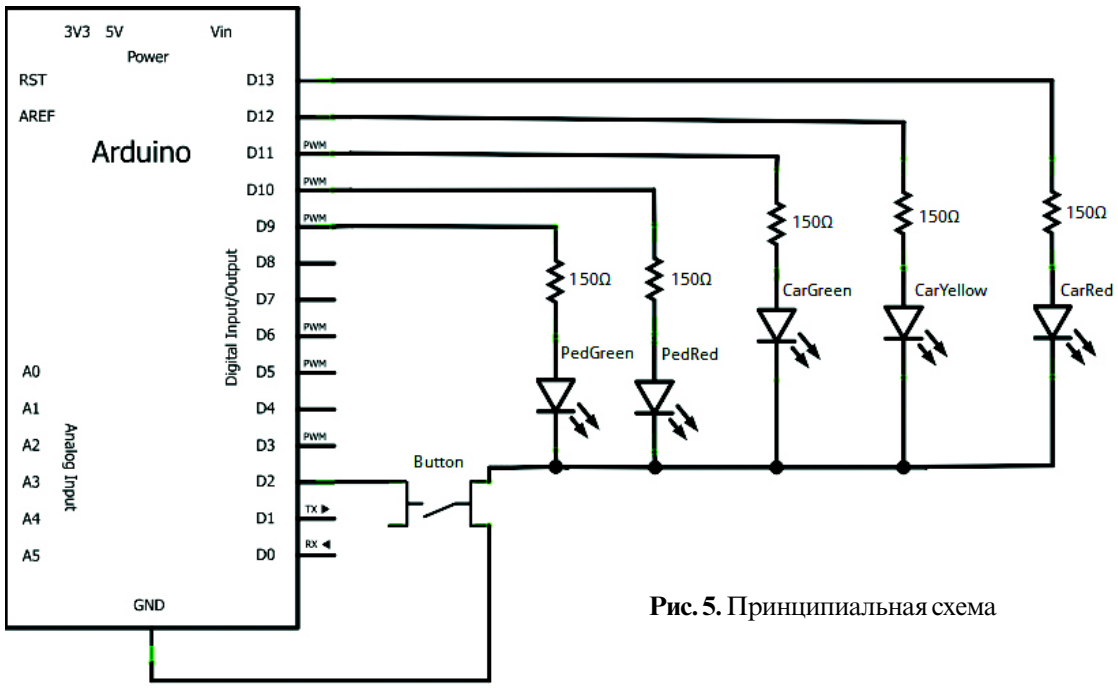


Рис. 5. Принципиальная схема

Листинг 7

```
// Интерактивный светофор
byte carRed = 13;     //красный сигнал светофора для автомобилей
byte carYellow = 12; //желтый сигнал светофора для автомобилей
byte carGreen = 11;  //зеленый сигнал светофора для автомобилей
byte pedRed = 10;     //красный сигнал светофора для пешеходов
byte pedGreen = 9;    //зеленый сигнал светофора для пешеходов
byte button = 2;      //кнопка пешехода
long lastChangeTime; //время с момента последнего зеленого для пешеходов
void setup() {
        pinMode(carRed, OUTPUT);
        pinMode(carYellow, OUTPUT);
        pinMode(carGreen, OUTPUT);
        pinMode(pedRed, OUTPUT);
        pinMode(pedGreen, OUTPUT);
        pinMode(button, INPUT);
        digitalWrite(button, HIGH); //подтяжка
        digitalWrite(carGreen, HIGH);
        digitalWrite(pedRed, HIGH);

}
void loop() {
        int state = digitalRead(button);
       //Если кнопка нажата и прошло больше 10 секунд с момента
       //последнего зеленого для пешеходов
        if (state == LOW && (millis() - lastChangeTime) > 10000){
                // Вызвать функцию чтобы включить зеленый для пешеходов
                changeLights();
        }
}
void changeLights() {
        digitalWrite(carYellow, HIGH); //включаем желтый для машин
        digitalWrite(carGreen, LOW);   //выключаем зеленый для машин
        delay(2000);
        digitalWrite(carRed, HIGH);   //включаем красный для машин
        digitalWrite(carYellow, LOW); //выключаем желтый для машин
        digitalWrite(pedGreen, HIGH); //включаем зеленый для пешеходов
        digitalWrite(pedRed, LOW);    //выключаем красный для пешеходов
        delay(7000);                  //ждем 7 секунд пока пешеходы переходят дорогу
        // мигаем зеленым
        for (int x=0; x < 5; ++x) {
                digitalWrite(pedGreen, HIGH);
                delay(250);
                digitalWrite(pedGreen, LOW);
                delay(250);
        }
        //обратно включаем красный для пешеходов, зеленый для машин
        digitalWrite(pedRed, HIGH);
        digitalWrite(carYellow, HIGH);
        digitalWrite(carRed, LOW);
        delay(2000);
        digitalWrite(carGreen, HIGH);
        digitalWrite(carYellow, LOW);
        //запоминаем время последнего зеленого для пешеходов
        lastChangeTime = millis();
}
```

Теперь функция **`digitalRead(pin)`** будет возвращать **`HIGH`**, если кнопка отпущена, и **`LOW`**, если кнопка нажата.

В главном цикле **`loop`** все время происходит проверка, нажата ли кнопка пешеходом. Если кнопка была нажата и с момента последнего зеленого сигнала для пешеходов прошло больше 10 секунд, вызывается функция **`changeLights()`**, которая меняет состояние светофора.

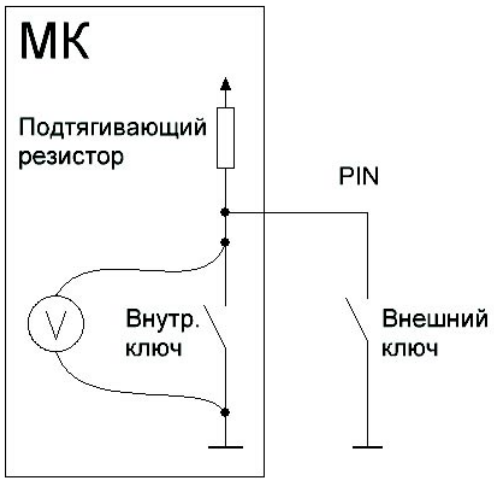


Рис. 6. Вольтметр показывает 5 В, когда внешний ключ разомкнут, и 0 В, когда ключ замкнут

Яркоев Константин Евгеньевич, разработчик фирмы SoftDev SPb.