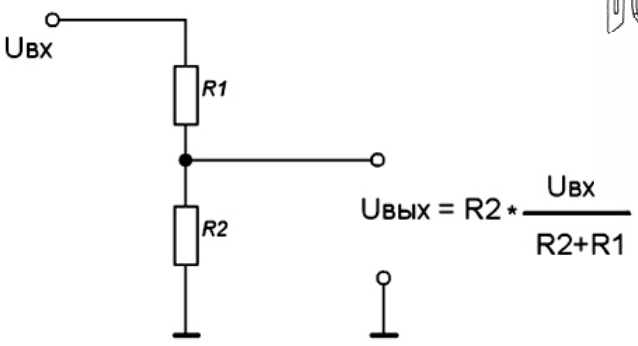
Яркоев Константин Евгеньевич

ПРАКТИЧЕСКАЯ МИКРОЭЛЕКТРОНИКА. ЗАНЯТИЕ 4. ШИРОТНО-ИМПУЛЬСНАЯ МОДУЛЯЦИЯ

На этом занятии мы познакомимся с аналоговыми портами вывода Ардуино, узнаем что такое ШИМ, сделаем программную эмуляцию цифро-аналогового преобразования, а также сделаем бегущую волну из нескольких светодиодов.

На прошлом занятии мы использовали цифровые порты и написали простейший светофор. Цифровые порты полезны, когда мы имеем дело с дискретными состояниями, например включено-выключено. Но что если мы хотим не просто включать/выключать свет, а менять напряжение плавно, чтобы яркость тоже менялась плавно? У нас есть источник тока, напряжением 5 В и мы хотим получить произвольное напряжение в интервале от 0 до 5 В. Самый простой вариант – делитель напряжения (рис. 1, см. Занятие 1):



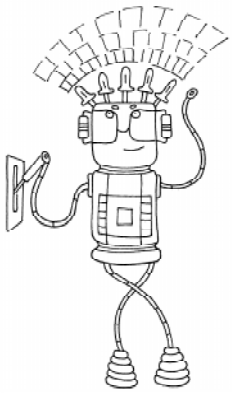
$$U_{вых} = R_2 \cdot I_{общ} = R_2 \cdot \frac{U_{вх}}{R_{общ}} = R_2 \cdot \frac{U_{вх}}{R_1 + R_2} = U_{вх} \cdot \frac{R_2}{R_1 + R_2}$$

Рис. 1. Делитель напряжения из двух резисторов

Меняя R_1 и R_2, можно плавно менять $U_{вых}$: $0 \le U_{вых} \le U_{вх}$.

Для этой цели существует специальное устройство – потенциометр, который изображен на рис. 2. У него есть три ножки-вывода, к крайним выводам подключается земля и $U_{вх}$, а с центрального вывода и земли снимается $U_{вых}$.

Однако у потенциометра низкий КПД: часть энергии тратится на нагрев резисторов. Кроме того, микроконтроллер не имеет возможности крутить «ручку» потенциометра. МК может подавать лишь 0 или 5 В.



Но как при помощи цифрового выхода, дающего нули и единицы, получить плавно меняющуюся величину?

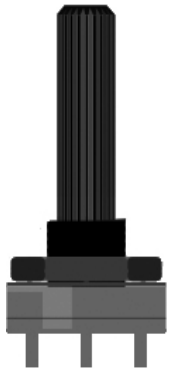


Рис. 2. Потенциометр (делитель напряжения)

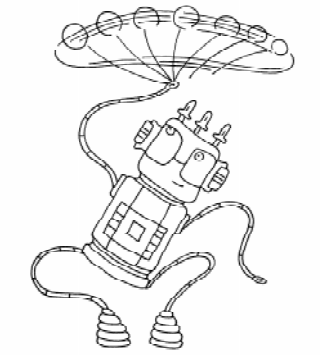
Дело в том, что все приборы имеют предел измерения или чувствительность. То же самое относится и к органам человека. Наш слух измеряет частоту в диапазоне 20Гц–20 кГц, глаз воспринимает мерцание до 60–70 Гц. Можно провести следующий эксперимент: возьмите камень или маленький мячик, привяжите его к веревке и начинайте вращать. Сначала вы будете наблюдать камень отдельно, но по мере увеличения частоты вращения камень превратится в окружность. Точнее, камень сам ни во что не превратится и будет по-прежнему камнем, но мы будем *наблюдать* его «размазанным» по окружности. При большой частоте вращения мы не сможем сказать, где конкретно находится камень в данный момент времени. Точно так же электрон составляет электронное облако вокруг атома.

Материальные тела обладают инерцией. Возьмем, например, массивный электромотор и начнем подавать на него номинальное напряжение. Мотор раскрутится до максимальных оборотов и будет вращаться с частотой ω. Если напряжение убрать, мотор постепенно остановится. Но если подавать напряжение короткими импульсами, мотор будет не успевать раскручиваться до частоты, в то же время не будет успевать останавливаться. Регулируя продолжительность импульсов, можно регулировать среднюю частоту вращения. Например, если каждую секунду подавать импульс длительностью 1/10 секунды, то мотор в силу инерции будет воспринимать данные импульсы как напряжение в 10 % от номинального (рис. 3).

Если каждую секунду подавать импульс длительностью 9/10 секунды, то мотор в силу инерции будет воспринимать данные импульсы как напряжение в 90 % от номинального (рис. 4).

Итак, вернемся к Ардуино. Микроконтроллер может подавать на вывод 0 или 5 В, соответствующие нули или единице. Мы можем сгенерировать периодический сигнал, состоящий из нулей и единиц (рис. 5). *Периодический* – означает полностью повторяющийся через равные промежутки времени. Период повторения обозначим за T, а время, в течение которого подается напряжение (5 В), обозначим за τ. Величину равную $D = \frac{\tau}{T}$, где $0 \le D \le 1$ называют *коэффициентом заполнения* или *рабочий цикл* (*Duty Cycle*). Иногда D считают в процентах, тогда $0\% \le D \le 100\%$.

Меняя τ при постоянном периоде T, мы меняем рабочий цикл $0\% \le D \le 100\%$, что будет восприниматься электродвигателем как плавное изменение напряжение от 0 до 5 В. То же самое произойдет с вольтметром, стрелка которого тоже инертна: вольтметр на рис. 6 покажет 4 В при D = 80 % или 1 В при D = 20 %.



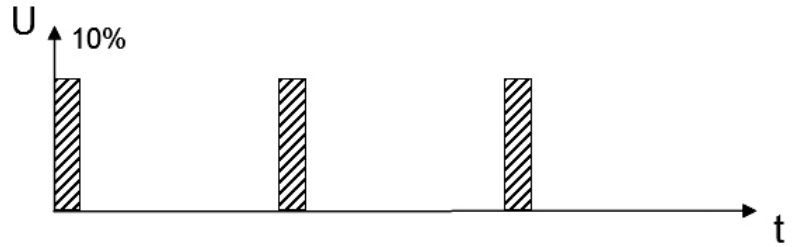


Рис. 3. Импульс продолжительностью 1/10 секунды

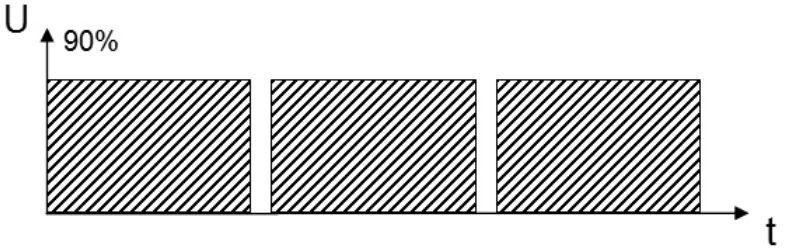


Рис. 4. Импульс продолжительностью 9/10 секунды

Если мы увеличим период T до 0.1 с, стрелка вольтметра начнет заметно дрожать, а при $T = 0.5$ с стрелка начнет ощутимо колебаться.

Мы выяснили, что приборы в силу своих конечных пределов измерения, а также предметы в силу своей инерции воспринимают дискретные сигналы высокой частоты как один сплошной непрерывный сигнал (рис. 7).

Широтно-импульсная модуляция (ШИМ) – это способ приближения желаемого выходного сигнала с помощью бинарных сигналов (с двумя уровнями вкл/выкл) так, что в среднем, за некоторый промежуток времени, их значения равны. Иначе говоря, с помощью ШИМ мы просто «обманываем» приборы, пользуясь их пределами измерения, а предметы – пользуясь их инерцией! Важно помнить, что частота должна быть достаточно высокой, иначе обман не удастся: стрелка вольтметра начнет вибрировать, двигатель работать неравномерно, а лампочка начнет моргать.

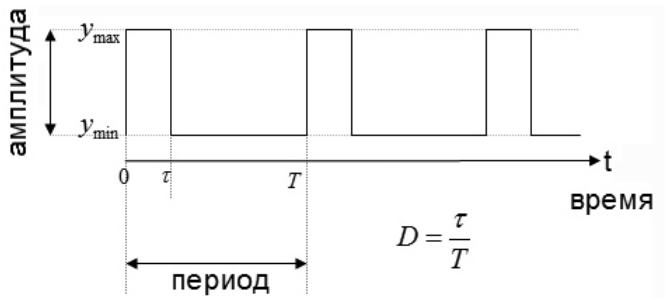


Рис. 5. Прямоугольная волна с параметрами: амплитуда A, период T, рабочий цикл D

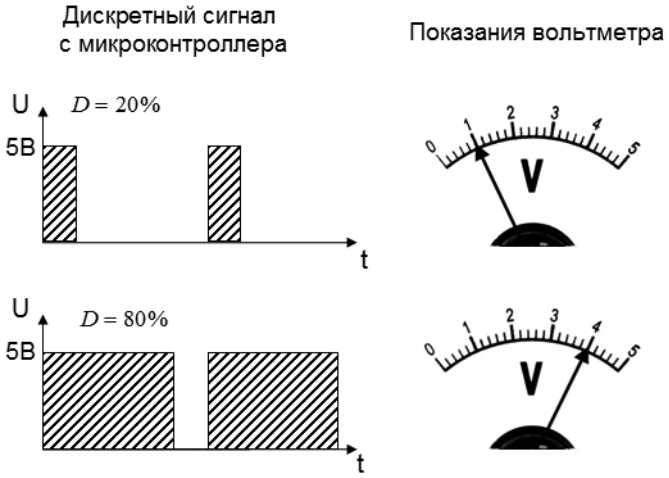


Рис. 6. Показания вольтметра при $A = 5$ В, $T = 0.01$ с

ПРОЕКТ 3. ШИМ

Попробуем на практике реализовать метод широтно-импульсной модуляции, заставив светодиод менять яркость в зависимости от рабочего цикла D.

«Предполетная подготовка»

Проверьте наличие следующих деталей:

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

– макетная плата, соединительные провода,

– светодиод,

– резистор на 150–300 Ом.

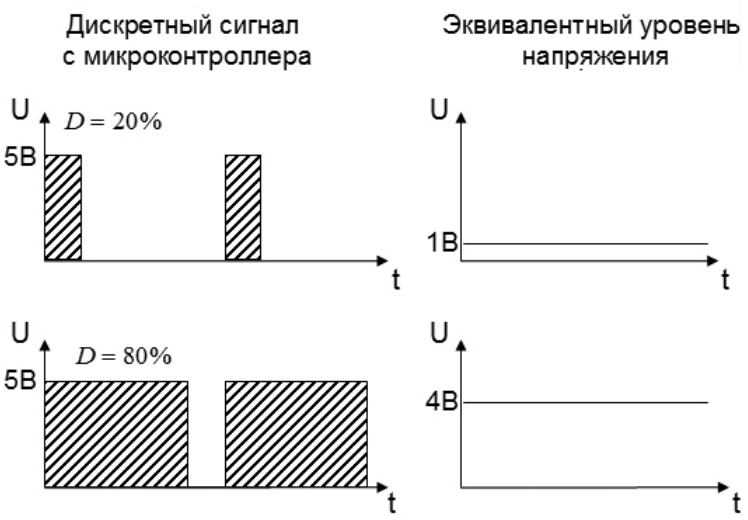


Рис. 7. Эквивалентное напряжение в зависимости от рабочего цикла

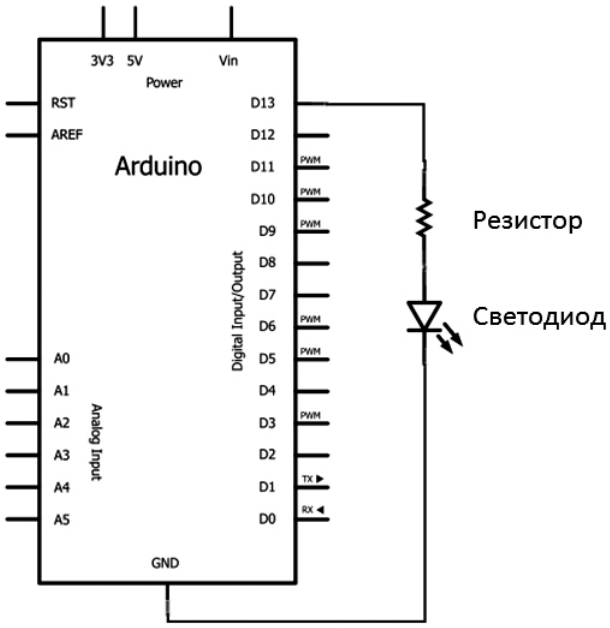


Рис. 8. Принципиальная схема

«Полет» (рис. 8, 9)

Чтобы сгенерировать прямоугольный сигнал для данного периода T и рабочего цикла D надо определить τ. Если $T = 1$ с, то $\tau = D$. Однако с такой

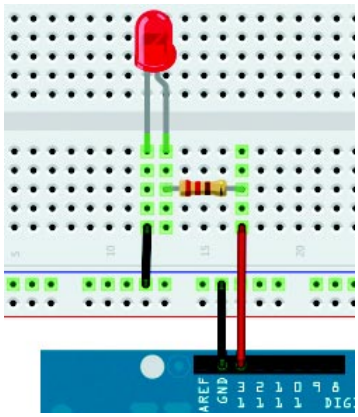


Рис. 9. Макетная плата

Листинг 1

```
/*
  PWM (Pulse-Wide modulation) - Широтно-Импульсная модуляция
  Реализация ШИМ. Можно управлять периодом и рабочим циклом.
  Включает диод на промежуток времени t=D*T, затем выключает
  на время (1-D)*T
 */

// Объявление переменной, хранящей номер вывода.
byte ledPin = 13;

// Инициализация.
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
   PWM(10, 0.5); //эмулируем один прямоугольный сигнал ШИМ в бесконечном цикле
}

//T - период в миллисекундах, D - рабочий цикл, D принадлежит [0;1]
void PWM(int T, float D)
{
  int t = T * D;
  digitalWrite(ledPin, HIGH); //включить светодиод
  delay(t);                   //подождать время t
  digitalWrite(ledPin, LOW);  //выключить светодиод
  delay(T-t);                 // подождать время T-t = (1-D)*T
}
```

частотой эффекта не достигнуть. Посчитаем чему равно τ для произвольного T' (рис. 10, листинг 1).

$$\frac{|AD|}{|AT|} = \frac{|A'D'|}{|A'T'|} \Rightarrow \frac{|AD|}{1} = \frac{\tau}{|A'T'|} \Rightarrow$$

$$\Rightarrow \frac{D}{1} = \frac{\tau}{T'} \Rightarrow \tau = D \cdot T'$$

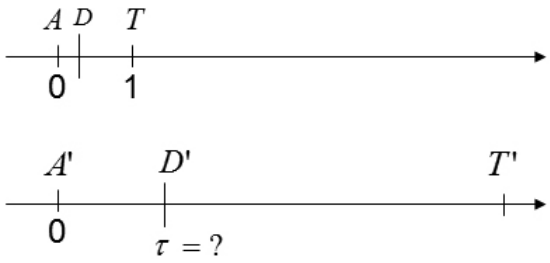


Рис. 10. Линейное преобразование отрезка AT в $A'T'$ (пропорции сохраняются)

«Разбор полета»

Мы сгенерировали бесконечный прямоугольный сигнал ШИМ. Яркость светодиода при $D = 50\,\%$ равна половине яркости при $D = 100\,\%$. Ведь у светодиода малая инерционность, в отличие от лампы накаливания, которая не успевает остыть при частоте 50 Гц. Светодиод действительно затухает и вспыхивает даже при большой частоте. Но почему мы не замечаем этого? Дело в том, что наш глаз не различает мерцание больше 60–70 Гц. То есть предельное количество вспышек, которое мы можем различить – 70 раз в секунду. Значит $T = 1/70 = 0.014$ с = 14 мс (миллисекунд). Лучше взять 10 мс, с запасом, чтобы даже самый зоркий глаз не заметил подвоха. Вы можете поэкспериментировать с периодом T и узнать частоту, на которой ваши глаза перестает видеть мерцание.

А теперь давайте заставим светодиод плавно менять яркость. Это можно легко сделать, меняя рабочий цикл D (см. листинг 2).

Мы получили немного неожиданный результат – светодиод стал менять яркость очень быстро. Это происходит потому, что

Листинг 2

```
/*
  PWM
  Реализация ШИМ.
  Плавно меняем напряжение (яркость диода), изменяя рабочий цикл ШИМ
 */

// Объявление переменной, хранящей номер вывода.
byte ledPin = 13;

// Инициализация.
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  //меняем рабочий цикл от 0 до 100% с шагом 10%
  for( float d = 0.0; d <= 1.0; d += 0.10 )
  {
    PWM(10, d);
  }
  //меняем рабочий цикл от 100% до 0 с шагом 10%
  for( float d = 1.0; d >= 0.0; d -= 0.10 )
  {
    PWM(10, d);
  }
}
```

```
//T - период в миллисекундах, D - рабочий цикл, D принадлежит [0;1]
void PWM(int T, float D)
{
  int t = T * D;
  digitalWrite(ledPin, HIGH); //включить светодиод
  delay(t);                   //подождать время t
  digitalWrite(ledPin, LOW);  //выключить светодиод
  delay(T-t);                 // подождать время T-t = (1-D)*T
}
```

Листинг 3

```
/*
  PWM. Реализация ШИМ.
  Плавно меняем напряжение (яркость диода), изменяя рабочий цикл ШИМ
 */

// Объявление переменной, хранящей номер вывода.
byte ledPin = 13;

// Инициализация.
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  //меняем рабочий цикл от 0 до 100% с шагом 10%
  for( float d = 0.0; d <= 1.0; d += 0.10 )
  {
    for( int i = 0; i < 20; i++)
    {
      PWM(10, d);
    }
  }
  //меняем рабочий цикл от 100% до 0 с шагом 10%
  for( float d = 1.0; d >= 0.0; d -= 0.10 )
  {
    for( int i = 0; i < 20; i++)
    {
      PWM(10, d);
    }
  }
}

//T - период в миллисекундах, D - рабочий цикл, D принадлежит [0;1]
void PWM(int T, float D)
{
  int t = T * D;
  digitalWrite(ledPin, HIGH); //включить светодиод
  delay(t);                   //подождать время t
  digitalWrite(ledPin, LOW);  //выключить светодиод
  delay(T-t);                 // подождать время T-t = (1-D)*T
}
```

весь цикл от 0 % до 100 % светодиод проходит за 10 итераций · 10 мс = 100 мс, то есть за одну десятую секунды. Попробуем не менять параметр *D* в течение нескольких циклов (см. листинг 3).

Как вы уже заметили, для генерации прямоугольного сигнала используется функция `delay()`. Конечно можно переписать программу, используя функцию `millis()`, чтобы избежать ненужных простаиваний. Однако есть вариант лучше: использовать аппаратные возможности Ардуино! В Ардуино есть несколько выводов, которые могут генерировать волну ШИМ с частотой примерно 490 Гц. Эти выводы являются цифровыми выводами № 3, 5, 6, 9, 10, 11 и обозначены на плате Arduino UNO волной ~ (рис. 11).

Синтаксис функции аппаратного ШИМ:

```
analogWrite(pin, value)
```

Параметр `pin` – номер вывода, на который подается сигнал ШИМ;

Параметр `value` – рабочий цикл. Значение меняется от 0 до 255, причем 0 соответствует полностью выключено (*D* = 0 %), а 255 – полностью включено (*D* = 100 %). Это ограничение микроконтроллера ATmega 328, где ШИМ имеет 8 бит (2^8 = 256 значений) (листинг 4).

ПРОЕКТ 4. БЕГУЩАЯ ВОЛНА

«Предполетная подготовка»

Проверьте наличие следующих деталей:

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

– макетная плата, соединительные провода,

– 6 светодиодов,

– 6 резисторов на 150–300 Ом.

Если светодиодов будет меньше, то волна будет короче.

«Полет» (рис. 12, 13, листинг 5)

«Разбор полета»

В данной программе используются три массива: массив списка контактов светодиодов `LedPins`, массив значений рабочего цикла `Values` каждого светодиода, массив направлений приращений `Directions` для каждого светодиода. Эти массивы используются для хранения данных о текущем состоянии *i*-го светодиода.

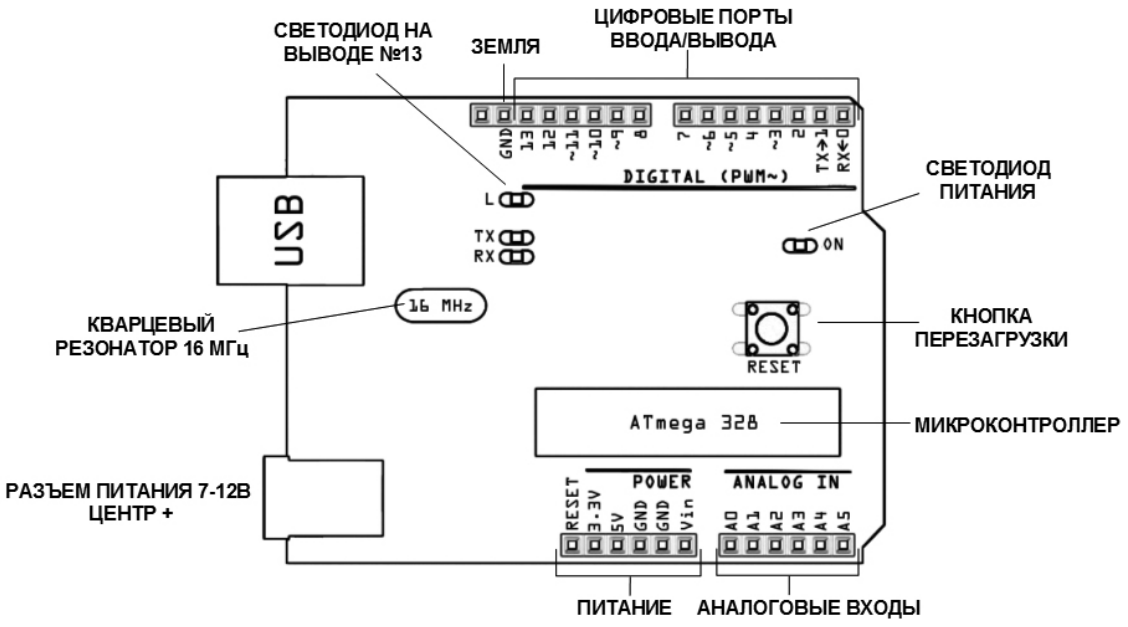


Рис. 11. Выводы с аппаратным ШИМ (PWM) обозначены на плате символом волны ~

Листинг 4

```
/*
  PWM.
  Плавно меняем напряжение (яркость диода),
  используя аппаратный ШИМ - функцию analogWrite()
 */

// ACHTUNG & WARNING!!! используем только выводы PWM!
byte ledPin = 11;

// Инициализация.
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  //t меняем в промежутке [0; 255] т.к. analogWrite
  //имеет 8-битную разрядность - имеет всего 2^8 = 256 значений
  for( int t = 0; t < 256; t += 1 )
  {
    analogWrite(ledPin, t);
    delay(10);
  }
  for( int t = 255; t >= 0; t -= 1 )
  {
    analogWrite(ledPin, t);
    delay(10);
  }
}
```

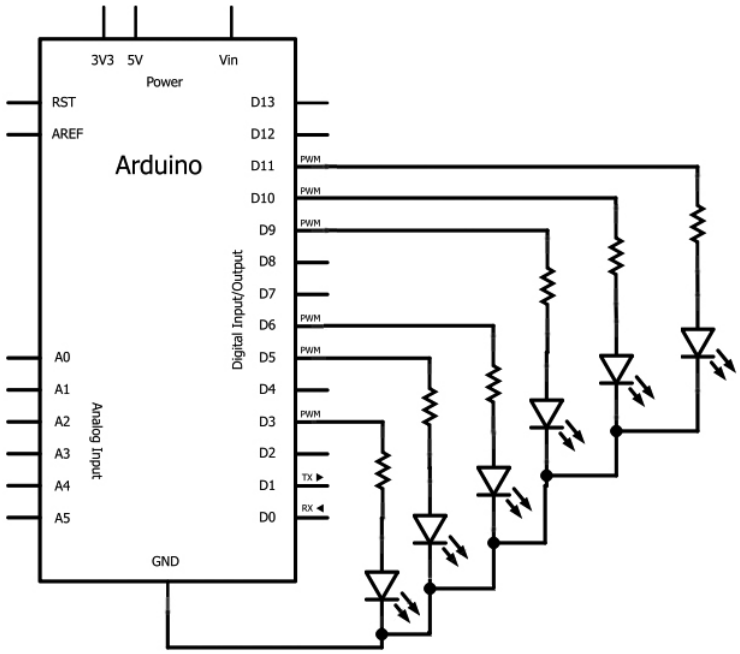


Рис. 13. Принципиальная схема

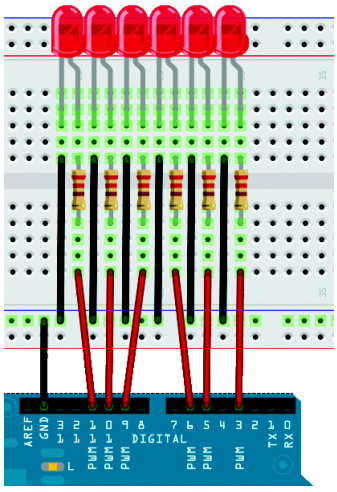


Рис. 12. Макетная плата

Листинг 5

```
/*
  "Бегущая волна" из светодиодов.
  Для задания яркости используется аппаратный ШИМ
 */

int LedPins[6] = { 3, 5, 6, 9, 10, 11 };  //список контактов светодиодов
byte Values[6] = { 0, 85, 170, 255, 170, 85 }; //начальные значения рабочего цикла
int Directions[6] = { +1, +1, +1, +1, -1, -1 }; //направление приращения: +1 или -1


void setup()  {
}

void loop()  {

  for(byte i = 0; i < 6; i++) //по всем светодиодам
  {
    byte pin = LedPins[i]; //берем номер контакта i-го светодиода
    byte value = Values[i]; //значение рабочего цикла i-го светодиода
    int direction =  Directions[i]; //направление приращения для i-го светодиода

    if( direction == 1 && value == 255) //если дошли до 100%
    {
      direction = -1; //надо уменьшать яркость
    }
    else if( direction == -1 && value == 0 ) //если дошли до 0%
    {
      direction = 1; //надо увеличивать яркость
    }

    Values[i] = value + direction;
    Directions[i] = direction;
    analogWrite(pin, Values[i]); //меняем яркость i-го светодиода
  }
  delay(2);  //задержка чтобы волна бежала не очень быстро
}
```

В главном цикле идет перебор по всем шести светодиодам. Если у *i*-го светодиода приращение +1 и значение рабочего цикла достигло 100 %, то приращение становится –1. Если приращение –1 и значение рабочего цикла достигло 0 %, то приращение становится +1.

Затем обновляем значение рабочего цикла и приращение *i*-го светодиода в массивах и устанавливаем соотв. напряжение на *i* светодиоде с помощью `analogWrite`.

Яркоев Константин Евгеньевич,
разработчик фирмы SoftDev SPb.