«СОЛОН»

Мамичев Д. И.

ПРОГРАММИРОВАНИЕ НА АРДУИНО

От простого к сложному

РАДИОЭЛЕКТРОНИКА

- составление простых скетчей для ARDUINO UNO
- практическое экспериментирование с ARDUINO UNO
- описание конструкций для повторения

Д. Мамичев

ПРОГРАММИРОВАНИЕ НА АРДУИНО

от простого к сложному

Москва
СОЛОН-Пресс
2018

УДК 621.396.6, 621.86/.87(072)
ББК 32.84, 74.262:32.816
М 22

Мамичев Д.
Программирование на Ардуино. От простого к сложному. — М.: СОЛОН-Пресс, 2018. — 244 с.: ил.

Эта книга написана для человека, только начинающего учиться программированию с использованием платформы Arduino. Многие эксперименты и скетчи могут показаться слишком простыми в повторении, даже для «непосвящённых в азы», но постижение сложного происходит от принятия простого. Чем надежнее вы освоите первые шаги, тем проще будет наращивать потенциал. Книга содержит шесть глав, первые пять рассказывают об экспериментах с платой Arduino UNO. Шестая глава содержит описания законченных конструкций. Книга безусловно будет полезна школьникам, студентам, начинающим электронщикам, желающим самообразовываться в данной области знаний.

Ответственный за выпуск: **В. Митин**
Верстка, обложка: **СОЛОН-Пресс**

ООО «СОЛОН-Пресс»
115487, г. Москва, пр-кт Андропова, дом 38, помещение № 8, комната № 2.
Формат 60×88/16. Объем 12,25 п. л. Тираж 100 экз.

По вопросам приобретения обращаться:
ООО «СОЛОН-Пресс». Тел: (495) 617-39-64, (495) 617-39-65
E-mail: kniga@solon-press.ru, www.solon-press.ru

ISBN 978-5-91359-292-7

ОГЛАВЛЕНИЕ

Ты датчик, отличающий ускорение
от силы притяжения нашел, физег?

Дмитрий Лилик,
программист из Питера

ПРЕДИСЛОВИЕ

Оговоримся, дорогой читатель сразу: эта книга написана в течение двух месяцев — ровно столько её автор знаком с программированием на языке C/C++ применительно к готовой плате Arduino UNO. Почти все скетчи разработаны и опробованы автором лично и рассчитаны исключительно на решение задачи первоначального самообучения программированию человека, без образования в данной области знаний. Однако простота и доступность конструкций не отменяет их практическую значимость. Изложение содержания идёт через анализ работы скетчей, каждый вариант подробно рассмотрен, даны макетные варианты реализации возможного изделия. Книга охватывает лишь малую часть практического применения Ардуино и не претендует на глубину изложения теоритических основ программирования. Может быть весьма полезна начинающим в качестве «катализатора собственных идей» на начальном этапе самообразования.

Схемотехника конструкций выбрана демократичной — большинство изделий ранее изготавливалось без использования микроконтроллеров и программирования. Это позволяет легче адаптироваться читателю, имеющему некоторый практический опыт в конструировании на дискретных элементах РЭА, микросхемах цифровых или аналоговых.

ГЛАВА 1
«АРДУИНО-ОЖИДАНИЕ»

Время неумолимо. Казалось ещё недавно, пайка в любительских условиях являлась неотъемлемой частью хобби. И вот на смену паяльнику пришла программа. Теперь необязательно постоянно перепаивать схему, добиваясь улучшения работы самоделки, достаточно отредактировать скетч и «перезалить его в микроконтроллер». Сегодня на смену отдельным элементам монтажа приходят готовые платки — функциональные модули. Гнёзда, штекера, различные разъёмы вместо пайки — контакт на «сухом трении». Так грустный конец одной истории плавно переходит в веселое начало новой...

Изучив мнение «интернет сообщества» по вопросу «с чего начинать программировать ?» — свой выбор остановил на проекте Arduino. Заказал у китайских коллег на AliExpress готовую плату — OOH R3 MEGA 328P ATMEGA16U2 (рис. 0). Она, как и большинство, ей подобных плат содержит микроконтроллер, USB- интер-

Рис. 0

фейс для его программирования и обмена данных, стабилизатор напряжения с выводами питания, контакты-выводы для ввода-вывода сигналов, встроенный последовательный интерфейс программирования (ICSP).

Посылка с заказом в пути и у меня впереди многие недели ожидания. Что дальше? А дальше погружение в «теорию вопроса». Из обилия «книжных ресурсов» приглянулась работа Джемери Блума «Изучаем Arduino: инструменты и методы технического волшебства», издательства БХВ-Петербург, вышедшая книгой у нас в 2015 году. Изучив первые две главы, понял одно — нужно создавать программы самому, для простых известных конструкций, продвигаясь к более сложным поделкам и изделиям. Их работоспособность решил проверять на компьютерной программе-симуляторе. Среди вариантов выбрал Autodesk с новым продуктом ***123D Circuits***. Это веб-приложение с хорошей имитацией платформы Arduino, которое позволяет в визуальном режиме прямо из браузера редактировать код и строить схемы без паяльника и проводов, что в моём случае (без платы) довольно удобно.

В визуальном режиме можно добавляя соединять провода и различные компоненты на беспаечной макетной плате и подключать к виртуальной «ардуинке». Также есть редактор кода, позволяющий вставлять тексты скетчей. «Исходники» или тексты программ буду набирать в Microsoft Word, копируя их и загружая в симулятор. Итак, приступим...

Эксперимент №1
«МИГАЮЩИЙ СВЕТОДИОД»

Мысленно соберём схему на макетке по рисунку 1. На рисунке 1а приблизительно показано как замысел может быть реализован. Подключим анод светодиода через токоограничительный резистор к выводу 1. К слову сказать, на плате цифровых выводов (пинов), способных работать и как вход и как выход — 14. С нумерацией в диапазоне от 0 до 13. Катод соединим с «–» (GND) источника питания (питание в 5 Вольт подаётся на плату и внешние цепи через сигнальный провод — от компьютера). Теперь рассмотрим работу базовой программы — примера «BLINK». Ниже дано её содержание:

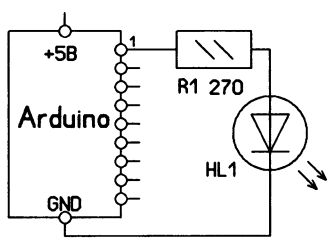

Рис. 1

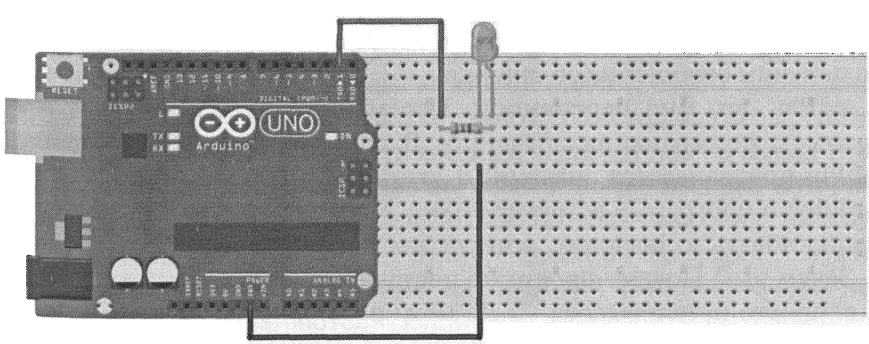

Рис. 1а

```
//LED Blink — вспышки светодиода////////////////////////////////
int ledPin = 1; //вывод Arduino к которому подключен светодиод

void setup()
 {
  pinMode(ledPin, OUTPUT);   //  определение вывода как ВЫХОД
}

void loop()
{
  digitalWrite(ledPin, HIGH);  //зажечь светодиод
  delay(1000);// задержка 1000 мсек (1 сек)
  digitalWrite(ledPin, LOW); //выключить светодиод
  delay(1000);  //ждать 1 сек
}
////////////////////////////////////////////////////////////////////
```

Если программу поставить в симулятор, предварительно «собрав в нём схему» (рис. 1а), то заметим при симуляции, что светодиод будет зажигаться и гаснуть с периодом в 2 секунды.

Дадим ключевые пояснения по коду.

Строки-пояснения, которые начинаются с «//» это комментарии автора к скетчу для лучшего понимания работы данной программы. «Машина» их никак не воспринимает.

Все команды заканчиваются точкой с запятой, если мы их забудем, то получим сообщение об ошибке.

ledPin- это имя переменной. Переменные используются в программах для хранения значений. В данном примере переменной ledPin присваивается значение 1, это номер вывода Arduino. Когда Arduino в программе встретит строку с переменной ledPin , он будет использовать то значение, которое мы указали ранее.

Зачем это? Почему просто не поставить значение туда куда надо? Т.е. просто номер вывода -1. Да можно, но в первом случае нам достаточно поменять вначале значение переменной и оно поменяется в каждой строке, где используется — автоматически, а во втором слу-

чае нам, придётся вручную в каждой команде вносить изменения (а это не всегда удобно).

«int» в первой строке указывает на тип переменной. При программировании Arduino важно всегда объявлять тип переменных.

В итоге запись **«int ledPin = 1;»** трактуем так: переменной данного вида с таким именем присвоить такое значение. То есть вначале любой программы желательно объявить все переменные с типом и указать их начальные значения.

«void setup()» обязательный атрибут любого скетча трактуем как: это функция (и её тип — void), которая вызывается один раз, когда стартует скетч, после каждой подачи питания или сброса платы Arduino. Используется для инициализации переменных, определения режимов работы выводов, запуска используемых библиотек и т.д.

«pinMode(ledPin, OUTPUT);» pinMode ()- эта функция используется в setup () части программы и служит для инициализации выводов, которые мы будем использовать, как вход (INPUT) или выход (OUTPUT).Функция имеет два аргумента: номер вывода, который мы будем использовать (у нас это вывод 1) и его статус — вход, выход. То есть команда читается: выводу 1 присвоить «звание» выход.

«void loop()» — это основная функция (и её тип), которая выполняется после setup(). Фактически это сама программа. Это функция будет выполняться бесконечно по кругу — петле, пока мы не выключим питание.

«digitalWrite(ledPin, HIGH);» digitalWrite() — эта функция служит для того, чтобы задавать состояние выхода. Есть два основных состояния , одно это HIGH, на выходе будет 5В (логическая 1), другое это LOW и на выходе будет 0В (логический 0). Значит, чтобы зажечь светодиод нам надо на 1 выводе , соединенном со светодиодом выставить высокий уровень HIGH. То есть команда читается так: на вывод 1подать 5 Вольт.Функция имеет два аргумента: номер вывода, который мы будем использовать и уровень сигнала на нём — высокий, низкий.

И наконец: **«delay(1000);»** — delay()- функция задержки. Служит для остановки работы программы на заданный в миллисекундах период. При этом сохраняются текущие режимы работы выходов (светодиод горит в течении 1000 мс — одной секунды). Команда в данном месте программы звучит: пусть светодиод горит одну секунду.

Далее, надеюсь, понятно... **«digitalWrite(ledPin, LOW);»** — выключи светодиод (подай логический ноль на первый вывод) и **«delay(1000);»** — пусть будет выключен одну секунду. Затем программа возвращается к void loop() и цикл повторяется бесконечно до отключения питания платы.

Кроме точки с запятой большое значение в правописании программ имеют фигурные скобки **{.....}**. Они должны строго попарно существовать в тексте программы, ограничивая определённые её фрагменты, например внутреннее тело цикла. Проще говоря, число открытых и закрытых скобок должно совпадать.

Остановив работу симулятора, можно редактировать текст программы. Для изменения частоты переключения светодиода и скважности импульсов достаточно менять числа в аргументах delay(). Уменьшаем их значения, и светодиод моргает чаще, увеличиваем — реже.

Поиграв с числами и номиналами токаограничительного резистора, задумался об иных программах мигания светодиода. Ниже даны два моих собственных скетча на «заданную тему», проверенные симулятором.

```
////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа мигания светодиода /
//

unsigned int n = 0;

void setup()
{
    pinMode(1, OUTPUT);
    digitalWrite(1, LOW);
}
```

```
void loop()
{
  for (n = 0; n <= 50000; n++)
    {
     digitalWrite(1, HIGH);
    }
  for (n = 0; n <= 50000; n++)
    {
     digitalWrite(1, LOW);
    }
}

//
// Конец /
//
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа мигания светодиода /
//


void setup()
{
    pinMode(1, OUTPUT);
    digitalWrite(1, LOW);
 }

void loop()
{
digitalWrite(1,! digitalRead (1));   //инвертируем состояние вывода
```

```
delay (1000);
}

//
// Конец /
//
////////////////////////////////////////////////////////////////
```

Попробуем описать иносказательно порядок их работы. Итак, для верхнего скетча:

1Введём положительную целочисленную переменную n и присвоим ей начальное значение 0.

2 Назначим вывод №1 выходом, подадим на него низкий уровень.

3 Запустим бесконечное число раз два последовательно выполняющихся цикла (с числом повторений 50000). Переменная n будет счётчиком с шагом равным +1.

4 В первом цикле светодиод будет 50000 раз получать команду на включение, во втором столько же раз на выключение. 5 Как бы программа быстро не выполнялась, всё равно на такое число повторений требуется время. Поэтому светодиод будет периодически мигать с частотой, заметной глазу.

Теперь поясним некоторые новые управляющие операторы: **«for (n = 0; n <= 50000; n++)»** — цикл с возможной интерпретацией — для n от нуля до 50000 включительно **(<=** оператор сравнения «меньше или равно»**)**, каждый раз к счётчику прибавляй единицу и выполняй действия внутри фигурных скобок (для первого цикла — зажигай светодиод). Меняя числа выполнения циклов можно менять частоту и скважность переключений светодиода. Обратите внимание, в данной программе отсутствует «распоряжение» delay().

Теперь опишем работу нижнего скетча:

1 Назначим вывод №1 выходом, подадим, для начала, на него низкий уровень.

2 А теперь подадим на вывод №1 уровень противоположный текущему, предварительно сняв информацию о нём.

3 Задержим получившиеся значение логического уровня на светодиоде на 1 сек.

4 Будем повторять эти действия (пункта 2-4) бесчисленное число раз.

В скетче появились новые символы — «!» это логический оператор отрицания, говоря проще, в данной программе он меняет состояние выхода светодиода на противоположное (высокий уровень сменится на низкий и наоборот). Функция digitalRead() она считывает значение с заданного входа — HIGH или LOW. В нашем случае аргументом имеет номер вывода — 1.

В итоге запись **«digitalWrite(1,! digitalRead (1));»** можно интерпретировать — на выводе 1 установи уровень сигнала противоположный предыдущему, значение которого мы получили.

Как и в первом примере, редактируя скетч в симуляторе можно, менять частоту переключения светодиода (изменяя аргумент в delay()). Для изменения скважности импульсов (соотношения длительностей свечения и выключения светодиода) «командную связку» в цикле loop придётся продублировать:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа мигания светодиода /
//


void setup()
{
    pinMode(1, OUTPUT);
    digitalWrite(1, LOW);
 }

void loop()
{
digitalWrite(1,! digitalRead (1));   //инвертируем состояние вывода
```

```
delay (1000);                          // задерживаем значение на 1 сек.
digitalWrite(1,! digitalRead (1));     // вновь инвертируем состояние
вывода
delay (500);                           // задерживаем значение на 0,5 сек.


}

//
// Конец /
//
////////////////////////////////////////////////////////////////////////
```

Эксперимент №2 «ПЕРЕКЛЮЧАТЕЛЬ СВЕТОДИОДНЫХ ГИРЛЯНД»

Наверное, одним из первых практически полезных применений платы рационально выбрать автомат световых эффектов. Выберем стандарт числа каналов, пусть их будет четыре. Пока не станем дополнять нашу схему силовыми ключами, ограничимся всего 4 светодиодами. Экспериментальная схема представлена на рис. 2. Вариант макетной реализации на рисунке 2а. Рассмотрим программу переключения светодиодов в режиме «бегущий огонь» с постепенно убывающей частотой бега:

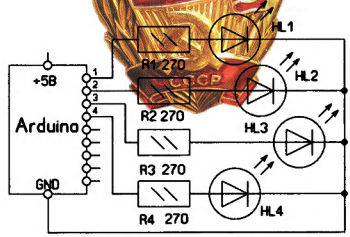

Рис. 2

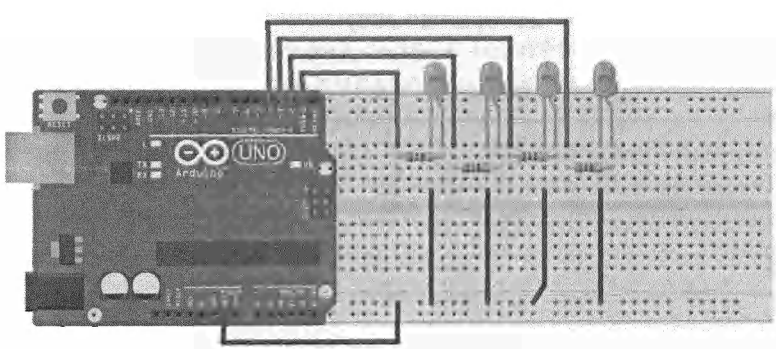

Рис. 2а

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа бегущий огонь на 4 светодиода /
// с переменной частотой переключения /

unsigned int n;
unsigned int k;

void setup()
{

  for ( n = 1; n <= 4; n++)
  {
    pinMode(n, OUTPUT);
    digitalWrite(n, LOW);
  }
}

void loop()
{
     for ( k = 200; k <= 2000; k=k+200)
  {
       for (n = 1; n <= 4; n++)
     {
       digitalWrite(n, HIGH);
       delay(k);
       digitalWrite(n, LOW);
     }
  }
}

//
```

// Конец /
//
//

В ней нет никаких новых «командных связок» и пояснений в строках кода, поэтому рассмотрим подробно смысловое представление скетча:

1 Введём целочисленные переменные n и k (диапазон значений от 0 до 65535 — unsigned int)

2 В цикле от 1 до 4 с шагом +1 присвоим выводам 1-4 состояние: выход

3 В этом же цикле на каждый вывод подадим логический ноль (светодиоды светиться не будут)

4 Перейдём к основной части программы: организуем два цикла (один внутри другого). Внешний цикл, со счётчиком k, меняет длительность свечения светодиодов. Внутренний цикл, со счётчиком n, переключает светодиоды.

5 Основные команды, бесконечно повторяющиеся в циклах, такие: включи светодиод такой то; оставь его включённым на столько то (счётчик k прибавляет в цикле по 200 миллисекунд свечения); выключи такой то светодиод.

6 В итоге, последовательно включаются и гаснут четыре светодиода, постепенно увеличивая время свечения каждого. В момент, когда счётчик k переполнится (станет больше 2000) частота переключений светодиодов скачком, резко увеличится и программа начнёт свой повтор.

Запись «n++» эквивалентна записи «n=n+1». Меняя исходные значения чисел 200, 2000 для k можно регулировать частоты переключений светодиодов и их количество.

Теперь рассмотрим ещё один вариант переключения светодиодов — они циклично ступенчато (одновременно все) увеличивают яркость свечения и гаснут, при этом время нарастания постепенно увеличивается до определённого порога, затем сложный цикл повторяется. Ниже дан скетч:

//
// Arduino UNO

```
//
//////////////////////
//
// Программа плавное зажигание на 4 светодиода /
// с переменной частотой нарастания /

unsigned int k;
unsigned int n;
unsigned int i;


void setup()
{

  for ( n = 1; n <= 4; n++)
  {
    pinMode(n, OUTPUT);
    digitalWrite(n, LOW);
  }
}

void loop()
{
     for ( k = 100; k <= 1000; k=k+100)
  {
       for (n = 1; n <= 4; n++)
     {
         for (i = 1; i <= k; i=i+1)
       {
          digitalWrite(1, HIGH);
          digitalWrite(2, HIGH);
          digitalWrite(3, HIGH);
          digitalWrite(4, HIGH);
           delay(n);
          digitalWrite(1, LOW);
          digitalWrite(2, LOW);
```

```
            digitalWrite(3, LOW);
            digitalWrite(4, LOW);
             delay(5-n);
        }
      }
   }
}

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Поясним его работу по ключевым позициям, отличным от работы ранее рассмотренного скетча. Основное тело программы реализовано в трёх циклах — матрёшке. Внешний (счётчик k) регулирует смену длительности свечения данного уровня яркости светодиодов. Средний цикл (счётчик n) задаёт уровень (ступень, их 4) яркости светодиодов. Внутренний (счётчик i) реализует саму длительность свечения данного уровня яркости. Проверяя работу скетча в симуляторе, можно так же регулировать временные параметры, меняя значения чисел 100, 1000.

Ещё один скетч-режим переключения дан ниже:

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа попарного перемигивания на 4 светодиода /
// с переменной частотой  /

unsigned int k;
unsigned int n;
unsigned int i;
```

```
void setup()
{

  for ( n = 1; n <= 4; n++)
  {
    pinMode(n, OUTPUT);
    digitalWrite(n, LOW);
  }
}

void loop()
{
     for ( k = 200; k <= 2000; k=k+200)
  {
       for (n = 0; n <= 1; n=n+1)
     {
        for (i = 0; i <= 1; i=i+1)
       {
        if (i==1)
         {
          digitalWrite(1, HIGH);
          digitalWrite(3, LOW);
         }
        else
         {
           digitalWrite(1, LOW);
           digitalWrite(3, HIGH);
         }

        if (n==1)
         {
          digitalWrite(2, HIGH);
          digitalWrite(4, LOW);
         }
```

```
        else
         {
           digitalWrite(2, LOW);
           digitalWrite(4, HIGH);
         }
        delay(k);
       }
     }
  }
}

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Функционал программы указан в её заглавии. Как она работает можно посмотреть в симуляторе. Мы остановимся лишь на новых «словах-связках» в программе.

```
        «   if (i==1)
        {
          digitalWrite(1, HIGH);
          digitalWrite(3, LOW);
        }
        else
        {
          digitalWrite(1, LOW);
          digitalWrite(3, HIGH);
        } »
```

Трактуется: если i тождественно равен 1, то включай первый светодиод, выключай третий. А иначе, выключай первый, включай третий.

Конструкция-оператор if..else предоставляет больший контроль над процессом выполнения кода (чем так же допустимая конструкция if...), т. е. по соблюдению или не соблюдению исходного условия реализует два варианта исполнения программы (либо так, либо так).

В заключение реализации этого эксперимента скажем: практически интересен код который одновременно проигрывает все три программы переключений светодиодов. Именно он наиболее зрелищный. Вариант такой объединённой программы дан ниже:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа 3х режимного переключателя на 4 светодиода /
// с переменной частотой переключения (задействованы выводы 1-4)/

unsigned int n;
unsigned int k;
unsigned int i;

void setup()
{

  for ( n = 1; n <= 4; n++)
  {
    pinMode(n, OUTPUT);
    digitalWrite(n, LOW);
  }
}

void loop()
{
     for ( k = 200; k <= 2000; k=k+200)
  {
       for (n = 1; n <= 4; n++)
     {
       digitalWrite(n, HIGH);
       delay(k);
```

```
      digitalWrite(n, LOW);

    }
}
    for ( k = 100; k <= 1000; k=k+100)
{
      for (n = 1; n <= 4; n++)
    {
        for (i = 1; i <= k; i=i+1)
      {
          digitalWrite(1, HIGH);
          digitalWrite(2, HIGH);
          digitalWrite(3, HIGH);
          digitalWrite(4, HIGH);
           delay(n);
          digitalWrite(1, LOW);

          digitalWrite(2, LOW);
          digitalWrite(3, LOW);
          digitalWrite(4, LOW);
           delay(5-n);
      }
    }
}
 for ( k = 200; k <= 2000; k=k+200)
{
      for (n = 0; n <= 1; n=n+1)
    {
        for (i = 0; i <= 1; i=i+1)
      {
        if (i==1)
         {
          digitalWrite(1, HIGH);
          digitalWrite(3, LOW);
         }
        else
```

```
          {
            digitalWrite(1, LOW);
            digitalWrite(3, HIGH);
          }

          if (n==1)
          {
            digitalWrite(2, HIGH);
            digitalWrite(4, LOW);
          }
          else
          {
            digitalWrite(2, LOW);
            digitalWrite(4, HIGH);
          }
          delay(k);
        }
      }
    }

}
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Эксперимент №3 «ИГРА — УГАДАЙ: КРАСНЫЙ ИЛИ ЗЕЛЁНЫЙ?»

На страницах радиолюбительской литературы, в разделах для начинающих эта игра известна много лет. До появления электроники люди её знают под именем «орёл или решка?». Она имитирует случайное выпадение одной из сторон монеты. Схема электронной версии для платы АРДУИНО представлена на рисунке 3. Макет на рисунке 3а. К новым элементам здесь относится управляющая кнопка SB1. Догадываетесь, что один из выводов платы будет работать в «режиме ВХОД» (вывод под номером 12). Работает схема просто — при нажа-

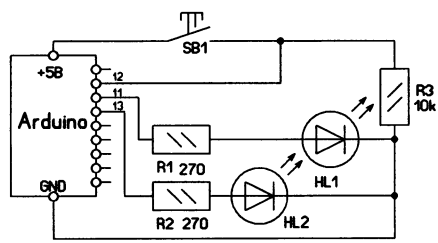

Рис. 3

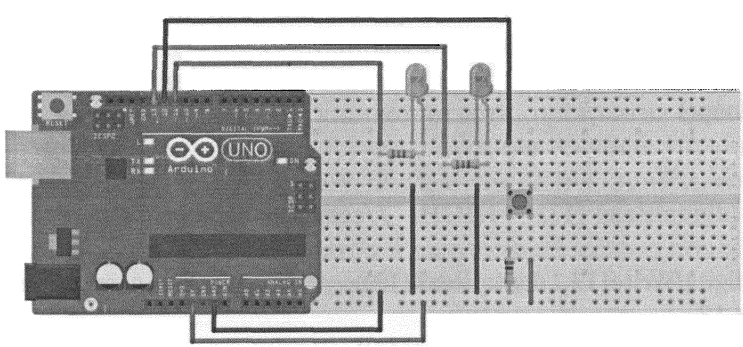

Рис. 3а

той кнопке зеленый и красный светодиод мерцают, после отпускания остаётся горящим лишь один «случайный светодиод», «выпавший» красный или зелёный. Резистор R3 в исходном состоянии (контакты кнопки разомкнуты) задаёт на входе (вывод 12) логический 0. При замыкании контактов на входе устанавливается логическая единица. Следует отметить, что переход с одного уровня на другой сопровождается дребезгом контактов — т.е. кратковременным, многочисленным «перескакиванием» логических уровней 0-1. Для корректной работы программы в её код вносят временную задержку — delay(25); — на время пережидания дребезга. Итак, ниже дан код:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//Игра  угадай красный или зелёный?//
void setup()
  {
     pinMode(11, OUTPUT); //красный светодиод вых. 11 /
     pinMode(13, OUTPUT); //зелёный светодиод вых.13 /
     pinMode(12,INPUT);  //кнопка выв.12 вход/
     digitalWrite(13,LOW);
     digitalWrite(11,HIGH);
   }

  void loop()

    {
      if (digitalRead (12)==HIGH)  //если кнопка нажата ...
       {
        digitalWrite(13,! digitalRead(13));  //инвертируем состояние
вывода 13
        digitalWrite(11,! digitalRead(13));  //инвертируем состояние
вывода 11 по отн. 13
```

```
      delay(25);        //небольшая защита от «дребезга» контактов
кнопки

      }

   }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Команду **«pinMode(12,INPUT);»** читаем так: выводу №12 «присвоить звание вход». Понимание работы программы:

1 Обозначить 11 и 13 выводы как выходы, 12 вывод как вход; вначале включить красный светодиод, зелёный выключить.

2 Всё время работы бесконечного цикла «loop» проверять уровень сигнала на кнопке и если вдруг этот уровень тождественен единице (кнопку нажали), то выполнять нижеследующие действия.

3 Узнав в каком состоянии находится зелёный светодиод — менять его состояние на противоположное. А следом инвертировать состояние красного светодиода по отношению к новому значению уровня зелёного светодиода.

4 Установившиеся изменения сохранить неизменными в течении 25 миллисекунд. Затем поменять изменения на противоположные значения и снова ждать 25 миллисекунд.

5 Пока кнопка нажата светодиоды будут быстро перемигиваться, обмениваясь логическими состояниями.

6 Если уровень сигнала на кнопке низкий, программа постоянно будет пропускать активные действия, сохраняя текущие уровни сигнала на светодиодах. По-другому, после отпускания кнопки, ярко горит «выпавший цвет светодиода».

Записав программу в симулятор и запустив её можно поэкспериментировать со значениями аргумента delay().

Эксперимент №4
«ИГРОВОЙ КУБИК НА ШЕСТЬ ГРАНЕЙ»

На рисунке 4 представлена схема электронного «шестигранного» игрового кубика. Число возможных выпадающих баллов от 1 до 6. В исходном состоянии схемы программа «быстро перебирает все грани кубика» — светодиоды мерцают. Нажимая и быстро отпуская кнопку, мы фиксируем выпавший результат на 5 секунд. Например, при выпадении тройки, светят светодиоды HL2, HL5, HL7. Рассмотрим программу работы игрушки:

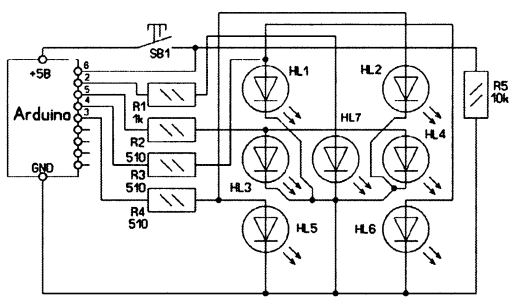

Рис. 41

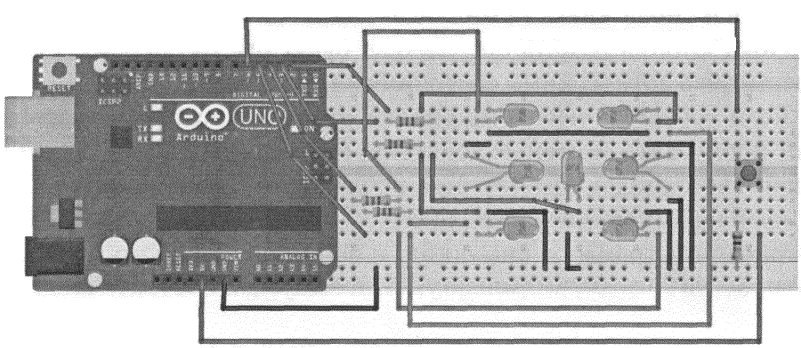

Рис. 4а

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//Кубик игровой на шесть граней//
void setup()
  {
     pinMode(2, OUTPUT); //центральный светодиод вых. 2 /
     pinMode(3, OUTPUT); //диагональ светодиодов вых. 3 /
     pinMode(4, OUTPUT); //диагональ светодиодов  вых. 4 /
     pinMode(5, OUTPUT); //горизонталь светодиодов вых.5 /
     pinMode(6,INPUT);  //кнопка выв.6 стоп игра/
     digitalWrite(2,LOW); //Зажигаем грань 6/
     digitalWrite(3,HIGH);
     digitalWrite(4,HIGH);
     digitalWrite(5,HIGH);
   }

  void loop()
   {
   for ( int k = 2; k <= 5; k=k+1)  // Зажигаем грань 1
       {
         digitalWrite(k,! digitalRead(k));
       }
         delay(10);    //небольшая защита от «дребезга» контактов кнопки


      if (digitalRead (6)==HIGH)  //если кнопка нажата, останавливаем
грань
      {
          delay(5000);         //Фиксация выпавшей грани 1 на 5 сек.
       }
       digitalWrite(2,! digitalRead(2));  // Зажигаем грань 2
       digitalWrite(3,! digitalRead(3));
```

```
      delay(10);
     if (digitalRead (6)==HIGH)  //если кнопка нажата, останавлива-
ем грань
      {
        delay(5000);        //Фиксация выпавшей грани 2 на 5 сек.
       }
        digitalWrite(2,! digitalRead(2)); // Зажигаем грань 3
        delay(10);
       if (digitalRead (6)==HIGH)  //если кнопка нажата, останавли-
ваем грань
      {
        delay(5000);        //Фиксация выпавшей грани 3 на 5 сек.
       }
      digitalWrite(2,! digitalRead(2)); // Зажигаем грань 4
      digitalWrite(4,! digitalRead(4));
       delay(10);
      if (digitalRead (6)==HIGH)  //если кнопка нажата, останавлива-
ем грань
      {
        delay(5000);        //Фиксация выпавшей грани 4 на 5 сек.
       }
       digitalWrite(2,! digitalRead(2)); // Зажигаем грань 5
        delay(10);
        if (digitalRead (6)==HIGH)  //если кнопка нажата, останавли-
ваем грань
      {
        delay(5000);        //Фиксация выпавшей грани 5 на 5 сек.
      }
      digitalWrite(2,! digitalRead(2)); // Зажигаем грань 6
      digitalWrite(5,! digitalRead(5));
       delay(10);
      if (digitalRead (6)==HIGH)  //если кнопка нажата, останавлива-
ем грань
      {
        delay(5000);        //Фиксация выпавшей грани 6 на 5 сек.
      }
```

```
}
//
// Конец /
//
//////////////////////////////////////////////////////////////////
```

Эта программа содержит много комментариев и в подробных пояснениях не нуждается. Последовательность основного цикла следующая: высвечивается грань кубика «единица» и «опрашивается кнопка», высвечивается грань кубика «двойка» и «опрашивается кнопка» и так далее до бесконечности, пока контакты кнопки не замкнут. После замыкания цикл перебора граней остановится на «текущей грани» на 5 секунд.

Принципиальное отличие конструкций на микроконтроллерах от изделий на цифровых микросхемах в том, что для перемены функции устройства часто достаточно лишь изменить код. Рассмотрим, как это работает в данном примере. Перепишем программу для шестигранного кубика с гранью «пусто» (зеро). Монтажную схему переделывать не нужно, достаточно лишь исключить из работы горизонтальную пару светодиодов HL 3, HL4. В этом случаи количество баллов на гранях меняется от 0-5. Тогда код примет вид:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//Кубик игровой на шесть граней с зеро//
void setup()
  {
    pinMode(2, OUTPUT); //центральный светодиод вых. 2 /
    pinMode(3, OUTPUT); //диагональ светодиодов вых. 3 /
    pinMode(4, OUTPUT); //диагональ светодиодов  вых. 4 /
    pinMode(5, OUTPUT); //горизонталь светодиодов вых.5 /
    pinMode(6,INPUT);  //кнопка выв.6 стоп игра/
```

```
    digitalWrite(2,HIGH); //Зажигаем грань 5/
    digitalWrite(3,HIGH);
    digitalWrite(4,HIGH);
    digitalWrite(5,LOW);
  }

 void loop()
  {
  for ( int k = 2; k <= 4; k=k+1)  // Зажигаем грань 0
      {
        digitalWrite(k,! digitalRead(k));
      }
        delay(10);   //небольшая защита от «дребезга» контактов кнопки


     if (digitalRead (6)==HIGH)  //если кнопка нажата, останавливаем
грань
     {
         delay(5000);        //Фиксация выпавшей грани 0 на 5 сек.
       }
       digitalWrite(2,! digitalRead(2));  // Зажигаем грань 1

        delay(10);
       if (digitalRead (6)==HIGH)  //если кнопка нажата, останавлива-
ем грань
          {
            delay(5000);        //Фиксация выпавшей грани 1 на 5 сек.
          }
            digitalWrite(2,! digitalRead(2));  // Зажигаем грань 2
            digitalWrite(3,! digitalRead(3));
            delay(10);
           if (digitalRead (6)==HIGH)  //если кнопка нажата, останавли-
ваем грань
          {
            delay(5000);        //Фиксация выпавшей грани 2 на 5 сек.
          }
```

```
        digitalWrite(2,! digitalRead(2)); // Зажигаем грань 3
            delay(10);
        if (digitalRead (6)==HIGH)  //если кнопка нажата, останавлива-
ем грань
         {
           delay(5000);        //Фиксация выпавшей грани 3 на 5 сек.
          }
          digitalWrite(2,! digitalRead(2)); // Зажигаем грань 4
          digitalWrite(4,! digitalRead(4));
           delay(10);
           if (digitalRead (6)==HIGH)  //если кнопка нажата, останавли-
ваем грань
         {
           delay(5000);        //Фиксация выпавшей грани 4 на 5 сек.
         }
        digitalWrite(2,! digitalRead(2)); // Зажигаем грань 5
           delay(10);
        if (digitalRead (6)==HIGH)  //если кнопка нажата, останавлива-
ем грань
         {
           delay(5000);        //Фиксация выпавшей грани 5 на 5 сек.
         }
}
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Вот и все изменения! Однако достаточно напутать с аргументами в скобках, поставив вместо 2 -3 или 4 -2 и вся «гармония» рухнет. В этом легко убедиться на симуляторе.

Эксперимент №5
«ПЕСОЧНЫЕ ЧАСЫ»

Микроконтроллер очень точно умеет считать время (благодаря кварцевому резонатору). Рассмотрим в этом примере конструкцию светодиодных «песочных часов», например, с интервалом счёта в одну минуту. Только вместо песка у нас будет из чаши в чашу переливаться свет. Принципиальная схема дана на рисунке 5, вариант макетной конструкции на рисунке 5а. В схему добавлены транзистор-

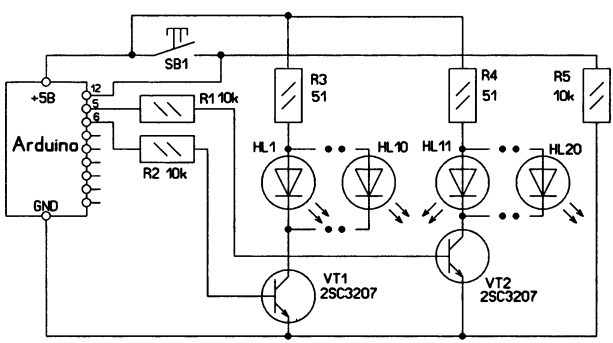

Рис. 1

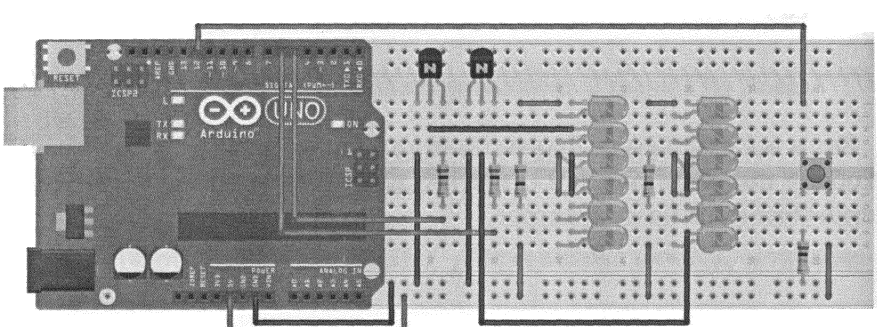

Рис. 1а

ные ключи, ибо выходы контроллера не в состоянии питать сразу десяток параллельно включённых светодиодов (они должны быть одинаковыми). Две группы светодиодов в реальной конструкции располагают в форме пары треугольников по 6 или 10 штук в каждой. Образованный индикатор напоминает по форме бабочку. Внешне часы работают так: вначале ярко светят светодиоды нижнего треугольника — светодиоды верхнего не горят. После кратковременного нажатия кнопки верхний треугольник загорается, нижний гаснет. Начинается отсчёт времени. Каждые 4 секунды яркость верхнего сегмента убывает, а нижнего нарастает. Через минуту часы остановятся — верхняя группа светодиодов погаснет, весь свет будет в нижней части индикатора. После повторного нажатия кнопки цикл отсчёта минуты повторится. Ниже написан скетч для данной конструкции:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//Песочные часы на 1 минуту//
unsigned int  n=0;
unsigned  int  i=0;

void setup()
  {
     pinMode(5, OUTPUT); //нижняя группа светодиодов /
     pinMode(6, OUTPUT); /верхняя группа светодиодов /
     pinMode(12,INPUT);  //кнопка запуска часов/
     analogWrite(5,n);
     analogWrite(6,238- n);
  }
     void loop()
    {
       if (digitalRead (12)==HIGH)  //если часы запущены ...
         {
```

```
    for (n = 0; n <= 14; n=n+1)
     {
      i=17*n;
            analogWrite(5,i);
            analogWrite(6,238- i);
            delay(4000);        //задержка свечения данной яркости на 4
сек.
             }
       }
  }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Программу можно интерпретировать так:

1 Введём две целочисленные переменные и присвоим им начальное значение 0.
2 Выводы 5 и 6 обозначим как выходы, вывод 12 как вход, на 5 вывод подадим минимальное напряжение (0 В), на 6 вывод максимальное (менее 5 В).
3 Будем постоянно проверять уровень сигнала на кнопке, если он окажется высоким — запустим цикл на 15 кругов.
4 При каждом его прохождении уровень яркости светодиодов, подключённых через ключ к выводу 5, будет нарастать, а к выводу 6 убывать.
5 Время свечения фиксированной яркости будет 4 секунды.
6 Спустя минуту (после отпускания кнопки) цикл for прервётся — время минуты истечёт.

Среди строк программы нам встречаются «новые символы». Функция analogWrite(). У неё два аргумента — номер вывода и уровень напряжения сигнала. В нашем скетче, например, **«analogWrite(6,238-i);»** читается так: на 6 вывод подать напряжение, составляющее 238-i/255 части от максимума при напряжении в 5 Вольт.

Условно (в числах) меняется от 0 до 255. Выдает аналоговую величину (точнее ШИМ волну) на порт вход/выхода. Функция может

быть полезна для управления яркостью подключенного светодиода или скоростью вращения вала электродвигателя. После вызова analogWrite() на выходе будет генерироваться постоянная прямоугольная волна с заданной шириной импульса до следующего вызова analogWrite (или вызова digitalWrite или digitalRead на том же порту вход/выхода).

На большинстве плат Arduino (на базе микроконтроллера ATmega168 или ATmega328) ШИМ поддерживают порты 3, 5, 6, 9, 10 и 11.

Для вызова analogWrite() нет необходимости устанавливать тип вход/выхода функцией pinMode().

Остановимся на этом, читатель. Можно ещё много придумать и написать скетчей, затем проверить их на симуляторе, но давай дождёмся самой платы и подключим её к компьютеру...

ГЛАВА 2
«АРДУИНО — НАЧАЛО»

Посылка-пакетик пришла достаточно быстро — через три недели. Вскрыв, и убедившись, что в дороге платка не пострадала, подключил её к компьютеру, предварительно установив программное обеспечение согласно рекомендациям в книге Блума.

На начальном этапе экспериментирования решил ограничиться в работе всего тремя «кнопками программы». Итак, последовательность следующая:

1. Собираем на макетке необходимую схему.
2. После тщательной проверки соединений подключаем разъём к компьютеру.
3. Открываем Arduino IDE.
4. В появившемся окне (рис. 1) нажимаем кнопку «Новый».
5. Открывается новое окно с заготовкой под свежий скетч (можно прямо набирать программу в этом окошке).

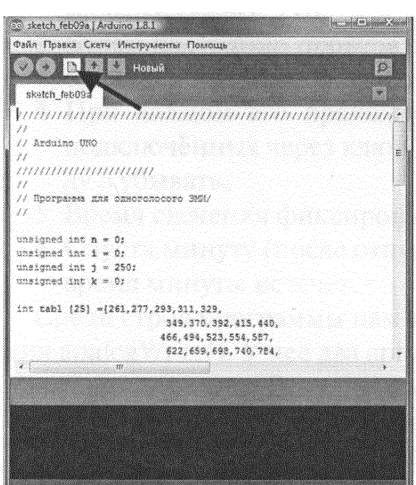

Рис. 1

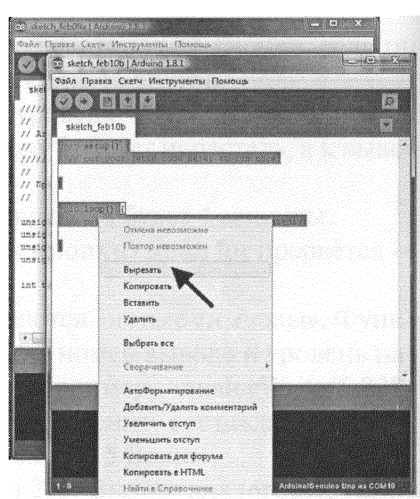

Рис. 2

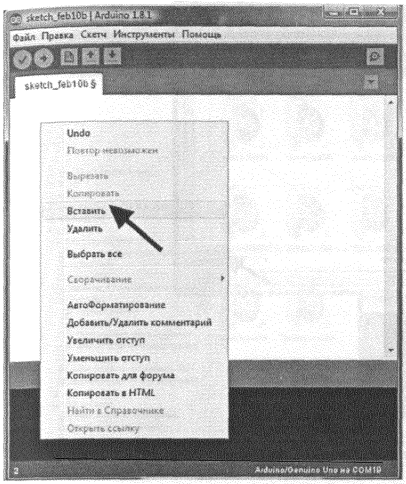

Рис. 3

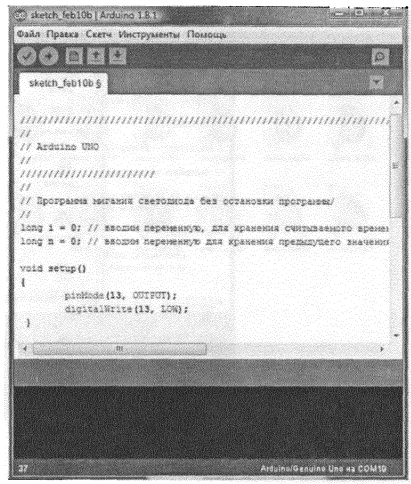

Рис. 4

6. Выбираем иной путь, как кажется автору более удобный...
7. Выделяем текст заготовки и удаляем его (работаем левой и правой кнопкой мыши) — рисунок 2.
8. Далее параллельно открываем документ, в котором храним текст скетча, выделяем его, копируем.
9. Вставляем текст программы в Arduino IDE — рисунок 3.
10. Через несколько секунд текст появится в окошке (рис. 4).
11. Убедимся что в скетче мы не допустили никаких ошибок — нажмём кнопку «Проверить» — рисунок 5.
12. Сразу появится дополнительное окошко, в котором программа предложит нам выбрать место сохранения пап-

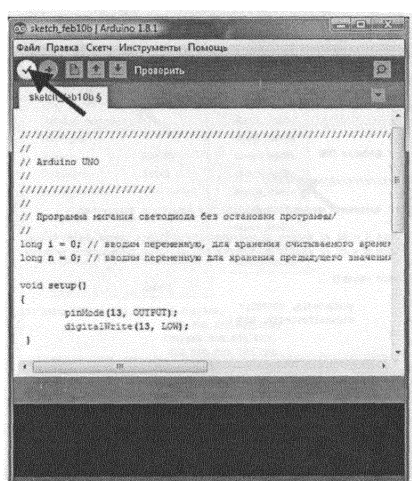

Рис. 5

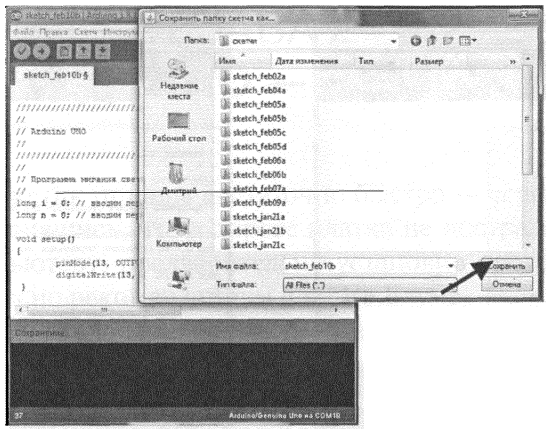

Рис. 6

ки со скетчем на нашем компьютере. Выберем это место и сохранимся — рисунок 6.

13. После этого окошко исчезнет, а процесс проверки-компиляции продолжится (в нижней части окна появится таймер реализации).

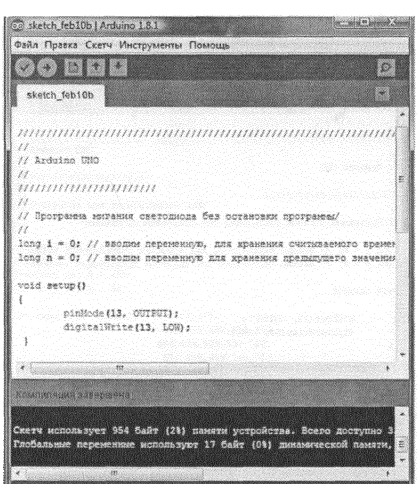

Рис. 7

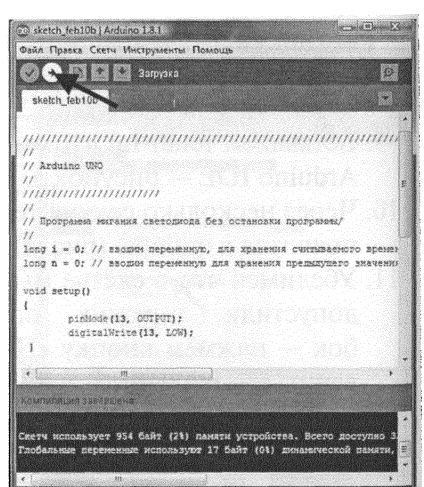

Рис. 8

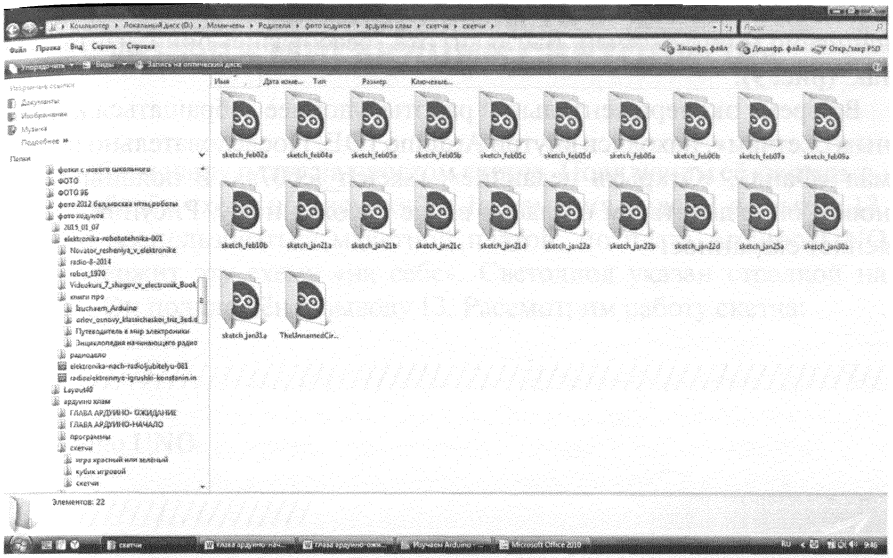

Рис. 9

14. Спустя некоторое время (десятки секунд), при успешной проверке, внизу окошка появляется надпись «Компиляция завершена» и высвечивается информация о «энергоёмкости» скетча — рисунок 7.
15. Далее нажимаем кнопку «Загрузка», ждём время... и наслаждаемся работой загруженного скетча в макете (железе) конструкции.

После завершения работы все окна можно закрывать, скетчи сохраняться в указанной папке. Следует обратить внимание, что Arduino IDE сохраняет каждый скетч в отдельной папке с указа-

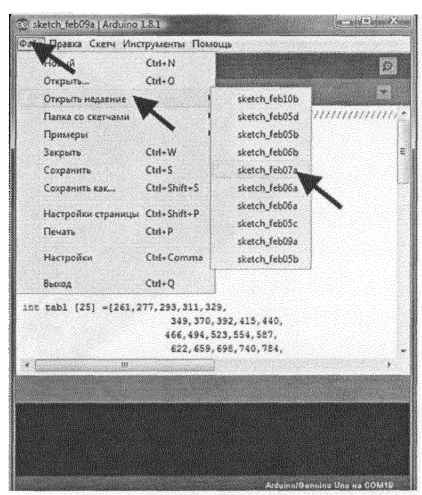

Рис. 10

нием даты его написания и буквенными индексами, если в этот день написано много скетчей. Выглядит это «безобразие» приблизительно так: (рис. 9).

Во время экспериментальной работы, удобнее «обращаться к нужным скетчам», находясь внутри Arduino IDE. Последовательно нажимая «Файл», «Открыть недавние», «sketch_feb07a». В появившимся новом окне действуем согласно выше изложенному. Рисунок 10 поясняет сказанное.

Эксперимент №6
«ВНОВЬ МИГАЮЩИЙ СВЕТОДИОД»

Опробуем нашу плату на простом скетче. Подключим её к компьютеру. Соберём схему по рисунку 1 и 1а, заменив вывод 1 на вывод 13. Можно воспользоваться макетной платой, но плата Arduino UNO уже содержит эту схему «на себе». Светодиод указан стрелкой на рис. 11. Он подключён к выводу 13. Рассмотрим работу скетча:

```
//////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////////
//
// Программа мигания светодиода без остановки программы/
```

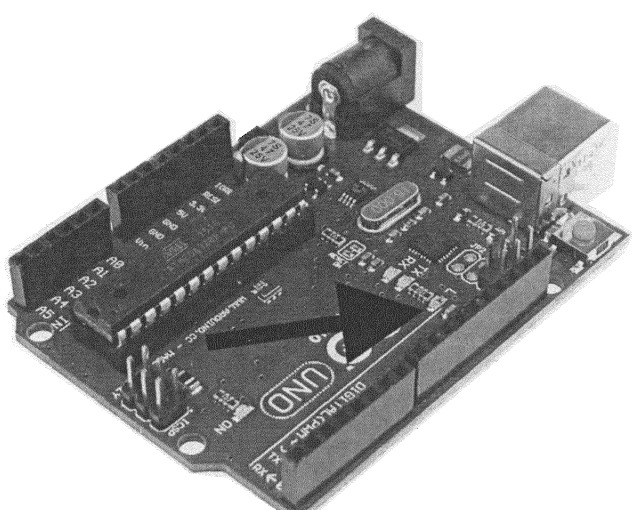

Рис. 11

```
//
long i = 0; // вводим переменную, для хранения считываемого времени
long n = 0; // вводим переменную для хранения предыдущего значения
времени

void setup()
{
    pinMode(13, OUTPUT);
    digitalWrite(13, LOW);
}

void loop()
  {
      i= millis(); //присваиваем переменной текущее время
        if (i — n > 500)  // если разница предыдущего значения и теку-
щего...
          {
   digitalWrite(13,! digitalRead (13));   //инвертируем состояние вывода
           n=i;  // заменяем предыдущее значение текущим

    }
}
//
// Конец /
```

Но вначале несколько слов о том, зачем опять «мигать светодиодом». Мы рассматривали в предыдущей главе пример того, как можно реализовать программно «светодиодный генератор», используя функцию delay(). Основной недостаток такого подхода заключается в том, что во время выполнения delay() программа «останавливается» и уже не может выполнять других операций. Получается мигание светодиодом ради мигания. В более трудных примерах светодиод должен мигать «отдельно», и программа выполняться тоже «отдельно». Такой подход можно реализовать, используя функцию **«millis()»**. Она возвращает количество миллисекунд с момента запуска текущей программы на плате Arduino. Это количество нарастает при непрерыв-

ной работе и сбрасывается в ноль, вследствие переполнения значения, приблизительно через 50 дней. Конечно, максимальное число очень большое и имеет тип unsigned long.

Теперь о программе. Её работу можно интерпретировать так:

1. Создадим две переменные для хранения и сравнения значений времени при вызове функции millis().
2. Вначале каждого возвращения к «void loop()» будем присваивать переменной i значение текущего времени и сравнивать его со значением, записанным в переменной n (стартовое значение её 0).
3. Если разница окажется больше полсекунды, мы поменяем логическое состояние выхода 13 на противоположное (включим или выключим светодиод), а также присвоим переменной n значение этого момента времени (так программа начнёт новый отсчёт следующей полсекунды).
4. Числовые значения i и n будут неуклонно нарастать со временем работы программы, а их разница будет практически неизменной.

Длительность свечения светодиода и время паузы можно регулировать заменой числа 500.

Эксперимент №7 «ОСМЫСЛЕННО МИГАЮЩИЙ СВЕТОДИОД или СПАСИТЕ НАШИ ДУШИ»

Эксперименты с одним светодиодом могут показаться бесполезными с точки зрения практического приложения. Но это не так. Всем, наверное, хорошо известна азбука Морзе, точнее, зачем эта азбука нужна. Из Википедии: код Мо́рзе — способ знакового кодирования, представление букв алфавита, цифр, знаков препинания и других символов последовательностью сигналов: длинных («тире») и коротких («точек»). За единицу времени принимается длительность одной точки. Длительность тире равна трём точкам. Пауза между элементами одного знака — одна точка, между знаками в слове — 3 точки, между словами — 7 точек. Назван в честь американского изобретателя и художника Сэмюэля Морзе. Код является довольно устойчивым к помехам при радиопередаче. Порой единственной возможностью быть услышанным при бедствии. Сигналы могут быть любыми, например, вспышки светодиода. Рассмотрим ниже программу, позволяющую передавать световыми вспышками международный сигнал SOS — сигнал бедствия. Текст скетча:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа передачи светодиодом сигнала SOS /
//

unsigned int n = 0;
```

```
int tabl [18] ={1,1,1,1,1,3,3,1,3,1,3,3,1,1,1,1,1,7}; // зашифрованный
сигнал

void setup()
{
    pinMode(13, OUTPUT);
    digitalWrite(13, LOW);
}

void loop()
{
  for (n = 0; n <18; n++)
    {
    digitalWrite(13,! digitalRead (13));   //инвертируем состояние
вывода

    delay(250*tabl[n]);  // временную задержку сигнала определяем
данными табл.
    }

}

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Здесь появляется новый элемент языка программирования — массив данных (мне, почему то хочется вместо слова массив употребить слово таблица). Массивы это именованный набор однотипных переменных, с доступом к отдельным элементам по их индексу. В нашей программе он объявлен так: «**int tabl [18] ={1,1,1,1,1,3,3,1,3,1,3,3,1,1, 1,1,1,7};**». Читаем — массив по имени tabl содержит 18 однотипных элементов, в фигурных скобках указаны через запятую все эти элементы. Доступ к элементам имеет небольшую особенность, начинается с номера 0. В нашей таблице (массиве) последний элемент имеет

номер соответственно 17. То есть, что бы присвоить, например, x значение 7 (из массива) в программе мы должны написать: «x=tabl[17];».

Изюминка данной программы и ей подобных состоит в том, что меняя содержание элементов массива и их количество легко можно шифровать любые сообщения, передаваемые вспышками светодиода азбукой Морзе.

Обратимся к рисунку 12. Сигнал состоит из трёх точек, трёх тире, трёх точек. Пусть точка это короткая вспышка светодиода, тире продолжительная вспышка. Пауза это выключенный светодиод. Тогда, согласно правилам азбуки, условно приняв единичный интервал времени за единицу, элементы массива будут иметь следующий смысл: включи светодиод на один интервал, выключи на один, включи на один, выключи на один, включи на один, выключи на три, включи на три, выключи на один, включи на три, выключи на один, включи на три, выключи на три, включи на один, выключи на один, включи на один, выключи на один, включи на один, выключи на семь.

Счётчик n перебирает элементы массива, в цикле переключая светодиод. Значения элементов задают время данного состояния светодиода. Каждое из них умножается на 250. Так получается временная задержка состояния на 250, 750 или 1750 миллисекунд.

Сигнал SOS передаётся непрерывно, до отключения питания платы.

Как и в предыдущем примере темп передачи сигнала можно менять, подбирая коэффициент 250 в пределах 50-500.

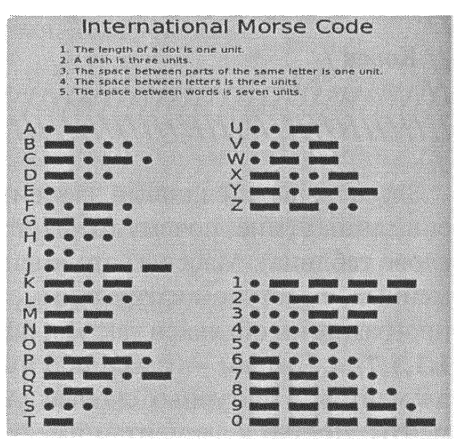

Рис. 12

Эксперимент №8
«ПЕСОЧНЫЕ ЧАСЫ С СЕКУНДНЫМ ОТСЧЁТОМ»

В эксперименте под номером 5 мы симулировали работу программы «песочных часов» с отсчётом времени в 1 минуту. Попробуем реализовать конструкцию в «железе». Изготовить индикатор (рис. 13) проще из отрезков листовой пластмассы толщиной 2-3мм. Светодиоды (1) располагаются в отверстиях диаметром 5мм, образующих пару треугольников. Подставка (2) и крышка индикатора (3) сажаются на секундный клей. Четыре соединительных провода (4) располагаются с обратной стороны индикатора. Монтаж выводов светодиодов (согласно рис. 5 предыдущей главы) навесной. Реализация макета показана на рисунке 14.

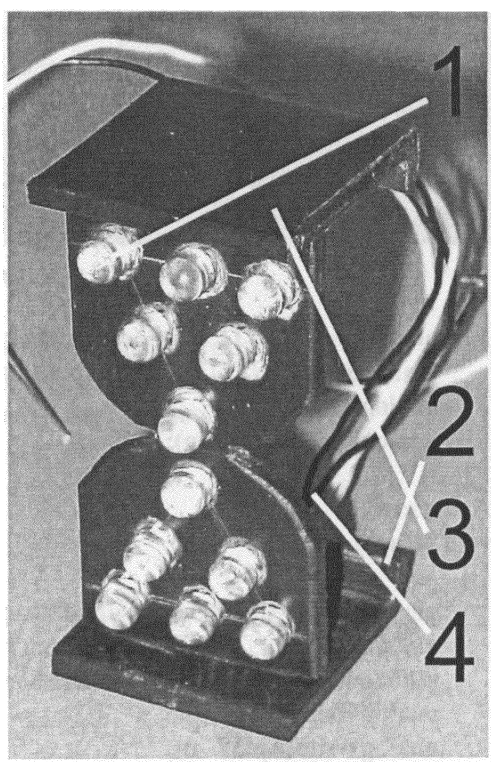

Рис. 13

Его работа по скетчу показала несколько существенных недостатков. Во-первых, в программе есть недочет, связанный с неправильным указанием начальных уровней сигнала на светодиодах. В программе записано:

```
void setup()
  {
    pinMode(5, OUTPUT); //
нижняя группа светодиодов /
```

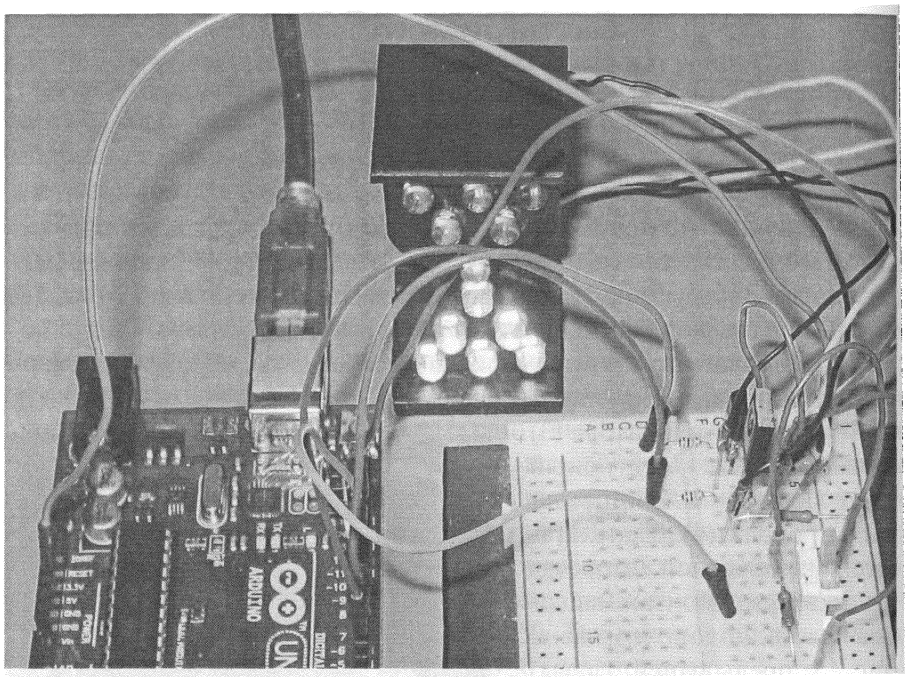

Рис. 14

```
    pinMode(6, OUTPUT); //верхняя группа светодиодов /
    pinMode(12,INPUT);  //кнопка запуска часов/
    analogWrite(5,n);
    analogWrite(6,238- n);
  }
```

А должно:

```
void setup()
  {
    pinMode(5, OUTPUT); //нижняя группа светодиодов /
    pinMode(6, OUTPUT); //верхняя группа светодиодов /
    pinMode(12,INPUT);  //кнопка запуска часов/
    analogWrite(6,n);
    analogWrite(5,238- n);
  }
```

Конечно, на работу программы это не влияет, но в начальный момент времени светят светодиоды верхнего треугольника, а должны нижнего. Но самое неприятное, что световой динамики на часах практически нет. Она присутствует в начале и в конце отсчёта времени. Для исправления положения можно ввести дополнительно отсчёт секунд, указывая его гашением всех светодиодов, например, на четверть секунды. Вариант исправленного скетча дан ниже:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//Песочные часы на 1 минуту с отсчётом секунд//
unsigned int  n=0;
unsigned  int  i=0;
unsigned  int  k;


void setup()
  {
     pinMode(5, OUTPUT); //нижняя группа светодиодов /
     pinMode(6, OUTPUT); //верхняя группа светодиодов /
     pinMode(12,INPUT);  //кнопка запуска часов/
     analogWrite(6,n);
     analogWrite(5,238- n);
  }
     void loop()
    {
       if (digitalRead (12)==HIGH)  //если часы запущены ...
          {
      for (n = 0; n <= 14; n=n+1)
       {
       i=17*n;
            for (k = 1; k <= 4; k=k+1)
```

```
                {
            digitalWrite(5, LOW);
            digitalWrite(6, LOW);

            delay(250);
            analogWrite(5,i);
            analogWrite(6,238- i);
            delay(750);        //задержка свечения данной яркости на 4
сек. за цикл
                }
             }
        }
   }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Здесь внутрь основного цикла смены яркости свечения светодиодных групп каждые 4 секунды добавлен четырёхкратный цикл их гашения на 250 миллисекунд. То есть каждую секунду светодиоды мигают, каждые четыре секунды после запуска яркость верхней группы убывает, нижней нарастает.

Такой вариант более информативен, но убывание-нарастание яркости по- прежнему очень «статично». Небольшого улучшения можно добиться, управляя изменением яркости в каждой группе на различные величины ступеней. Вариант скетча ниже:

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//Песочные часы на 1 минуту с отсчётом секунд — улучшенные//
unsigned int  n=0;
```

```
unsigned int i=0;
unsigned int k;
unsigned int j=0;
void setup()
  {
     pinMode(5, OUTPUT); //нижняя группа светодиодов /
     pinMode(6, OUTPUT); //верхняя группа светодиодов /
     pinMode(12,INPUT);  //кнопка запуска часов/
     analogWrite(6,n);
     analogWrite(5,238- n);
  }
     void loop()
    {
       if (digitalRead (12)==HIGH)  //если часы запущены ...
         {
      for (n = 0; n <= 14; n=n+1)
       {

          i=5*n;
          j=17*n;
            for (k = 1; k <= 4; k=k+1)
                 {
              digitalWrite(5, LOW);
              digitalWrite(6, LOW);

              delay(250);
              analogWrite(5,i);
              analogWrite(6,238- j);
              delay(750);        //задержка свечения данной яркости на 4
сек. за цикл
                  }
               }
         }
   }
//
// Конец /
```

```
//
////////////////////////////////////////////////////////////////////////
```

Изменить существенно динамику смены свечения можно изменив конструкцию индикатора. Но об этом позже...

Эксперимент №9
«7 СЕГМЕНТНЫЙ СВЕТОДИОДНЫЙ ИНДИКАТОР»

Индикатор предназначен для отображения цифробуквенных символов. Он состоит из семи основных сегментов — штрихов и восьмого сегмента — точки. Каждый сегмент это светодиод. Их группа, объединённая по одному из выводов в общий анод или катод, образует индикатор. В нашем эксперименте это будет индикатор с общим анодом (каждый сегмент зажигается логическим 0). На рисунке 16 дана схема включения индикатора и платы, на рисунке 17 вариант макетной реализации (следует отметить, что в зависимости от типа индикатора он имеет различную «цоколёвку» выводов). Рассмотрим работу программы, последовательно включающую цифры от 0 до 9 с задержкой свечения каждой в одну секунду. Для переделки программы под индикатор с общим катодом в ней достаточно убрать два «восклицательных знака» — операторы отрицания. Смотрим скетч.

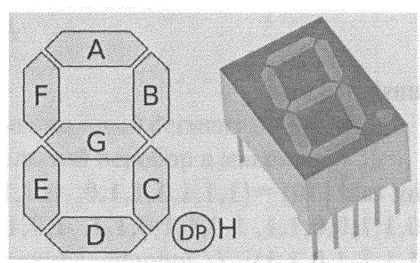

Рис. 15

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа перебора цифр 7 сегментного  индикатора /
//

unsigned int k = 0;
```

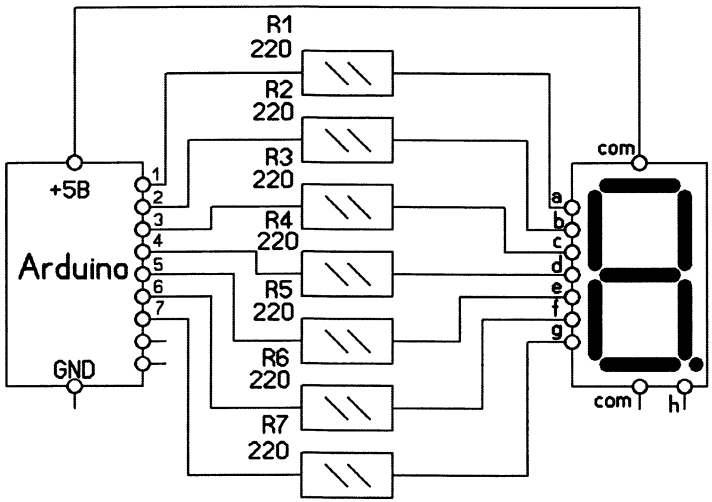

Рис. 16

```
unsigned int i ;


int tabl [70] ={1,1,1,1,1,1,1,0,  0,1,1,0,0,0,0,0,  1,1,0,1,1,0,1,  1,1,1,1,1,0,0,1,
0,1,1,0,0,1,1, 1,0,1,1,0,1,1,  1,0,1,1,1,1,1,  1,1,1,0,0,0,0,  1,1,1,1,1,1,1,
1,1,1,1,0,1,1}; // зашифрованная последовательность цифр

void setup()
{
  for (i = 1; i <=7; i++)
      {
      pinMode(i, OUTPUT);
      digitalWrite(i,! LOW);
      }
 }

void loop()
{
```

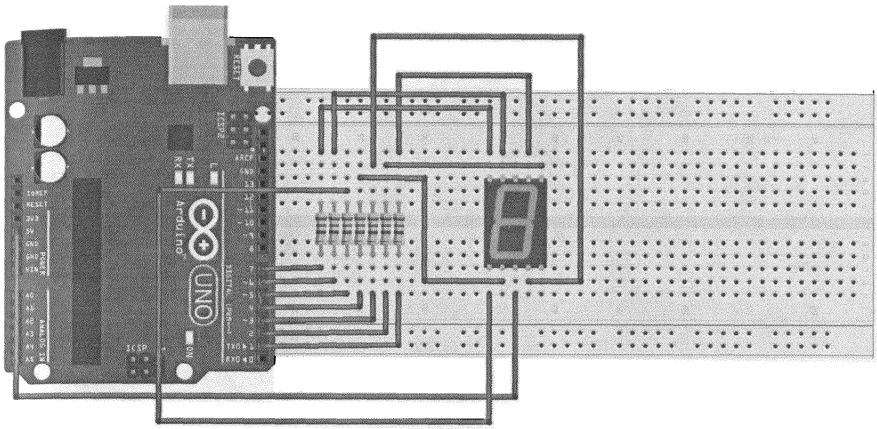

Рис. 17

```
 for (i = 1; i <=7; i++)
  {
   digitalWrite(i,! tabl [k]);   //включаем каждый сегмент по коду
   k=k+1; // производим последовательный перебор в массиве
     if (k==70)
      {
        k=0;
       }
  }
 delay(1000);
}


//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

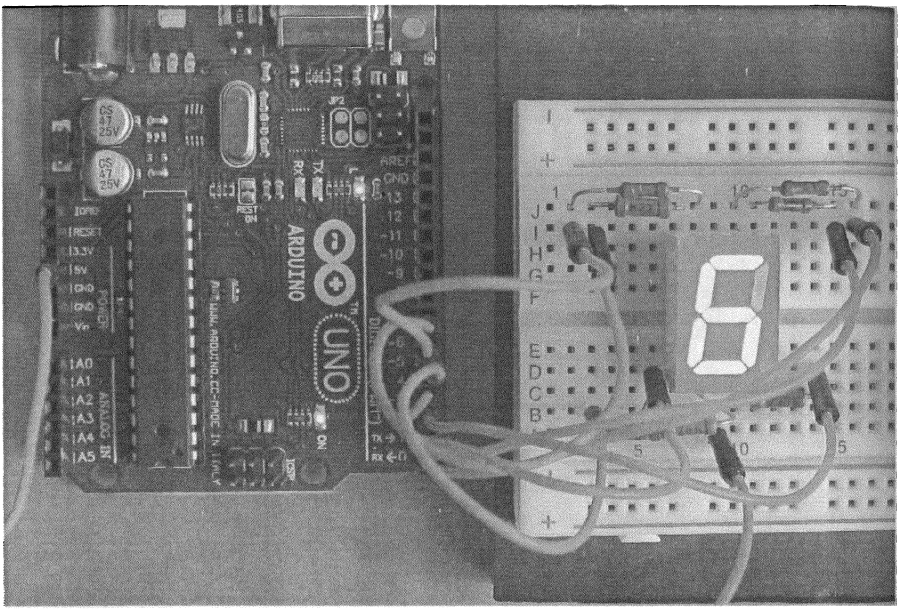

Рис. 18

В массив занесены значения логических уровней, последовательно подаваемые на сегменты a,b,c,d,e,f,g индикатора (рис. 15). Так первая группа элементов массива **1,1,1,1,1,1,0,** на индикаторе высветит цифру 0, последняя **1,1,1,1,0,1,1** — цифру 9.

Счётчик i перебирает выводы, подающие сигналы на сегменты, счётчик k элементы массива — логические уровни, выводимые на них. Дополнительное условие — **«if (k==70)»** — не даёт счётчику «покинуть границы массива», следующей командой обнуляя его. Время свечения цифры определяет аргумент функции delay(). Его значение рационально выбирать в пределах 250-2000 мс. Вид на макет эксперимента поясняет рисунок 18. Меняя содержание таблицы-массива можно задавать любые последовательности и виды знаков на индикаторе.

Эксперимент №10 «СЕКУНДОМЕР НА ПРЕДЕЛ ИЗМЕРЕНИЯ — 10 СЕКУНД»

В этом эксперименте рассмотрим наиболее очевидное практическое применение индикатора, создадим макет секундомера с отсчётом и фиксацией результата в пределах десяти секунд. Добавим в схему кнопку с функцией «запуск-сброс». Вариант схемы дан на рисунке 19, макетное расположение элементов на рисунке 20. Нажимая и фиксируя кнопку, мы отсчитываем время, отпуская — фиксируем текущее значение количества прошедших секунд. Вновь нажимая кнопку, обнуляем показания индикатора, начиная новый отсчёт секунд с нуля. Программа видоизменится следующим образом:

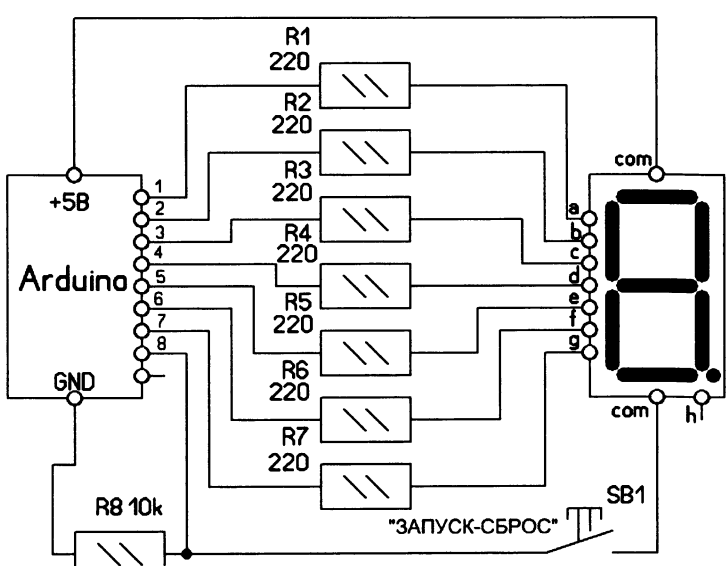

Рис. 19

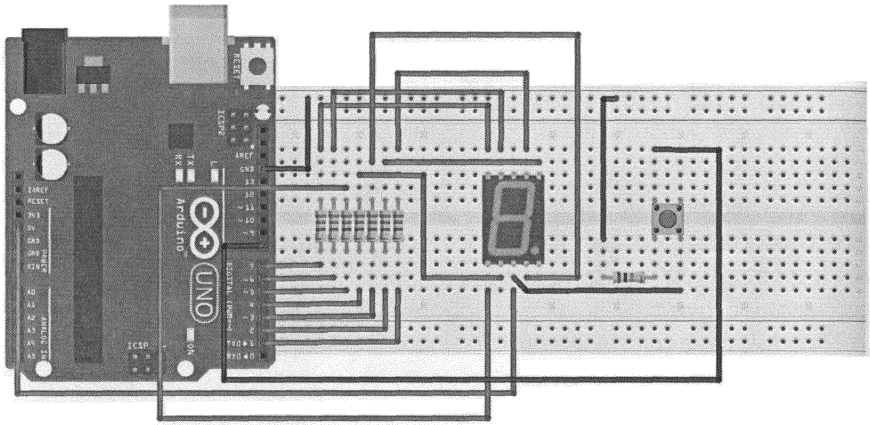

Рис. 20

```
//////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Секундомер на 10 секунд /
//

unsigned int k = 0;
unsigned int i ;


int tabl [70] ={1,1,1,1,1,1,0,0,1,1,0,0,0,0,1,1,0,1,1,0,1,1,1,1,1,0,0,1,0,1
,1,0,0,1,1,1,0,1,1,0,1,1,1,0,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,
0,1,1};  // зашифрованная последовательность цифр

void setup()
{
    pinMode(8,INPUT);  //кнопка запуска, остановки секундомера
  for (i = 1; i <=7; i++)
```

```
    {
    pinMode(i, OUTPUT);
    digitalWrite(i,! LOW);
    }
}

void loop()
{
      if (digitalRead (8)==LOW)  // если кнопка отпущена...
      {
         k=0;
       }
      if (digitalRead (8)==HIGH)  //если кнопка нажата ...
       {

  for (i = 1; i <=7; i++)
   {
    digitalWrite(i,! tabl [k]);   //включаем каждый сегмент по коду
    k=k+1; // производим последовательный перебор в массиве
      if (k==70)
       {
         k=0;
        }
   }
  delay(1000);
      }
}

//
// Конец /
//
```

Трактовать её содержание можно так:

1. Введём две переменные и массив, указав все его 70 элементов.
2. Обозначим вывод 8 как вход, в цикле обозначим выводы с 1 по 7 как выходы, присвоим им через отрицание высокий уровень (сегменты гореть не будут).

3. В бесконечном цикле void loop() сначала будем проверять уровень сигнала на кнопке (вывод 8).
4. Если он высокий запустится перебор сегментов каждой цифры от 0 до 9 с задержкой на каждой в 1 сек.
5. При этом на каждом элементе массива будет проверяться условие, предотвращающее выход значения счётчика k за границы допустимого.
6. При отпускании кнопки значение счётчика k обнулится и цикл void loop() будет выполняться в границах отрезка:

```
if (digitalRead (8)==LOW)  // если кнопка отпущена...
{
    k=0;
}
```

Индикатор при этом будет высвечивать последнюю цифру, при которой произошло нажатие кнопки. Следует отметить, что из-за несовершенства программы погрешность измерения данным прибором интервала времени будет составлять не менее одной секунды.

Эксперимент №11 «СЕКУНДОМЕР НА ПРЕДЕЛ ИЗМЕРЕНИЯ — 99 СЕКУНД»

Конечно, гораздо практичнее будет конструкция, позволяющая измерять большее число секунд с момента начала отсчёта. Как это сделать, не прибегая к использованию дополнительного индикатора и выводов платы? Можно использовать принцип динамического вывода информации на индикатор. Рассмотрим схему на рисунке 21 и возможный вариант макета на рисунке 22. В схему секундомера добавлена ещё одна управляющая кнопка и задействован восьмой сегмент — точка. Программно реализован следующий порядок работы секундомера: после подачи питания при отключенных кнопках на индикаторе высвечивается 0 и мигает точка — секундомер готов к ра-

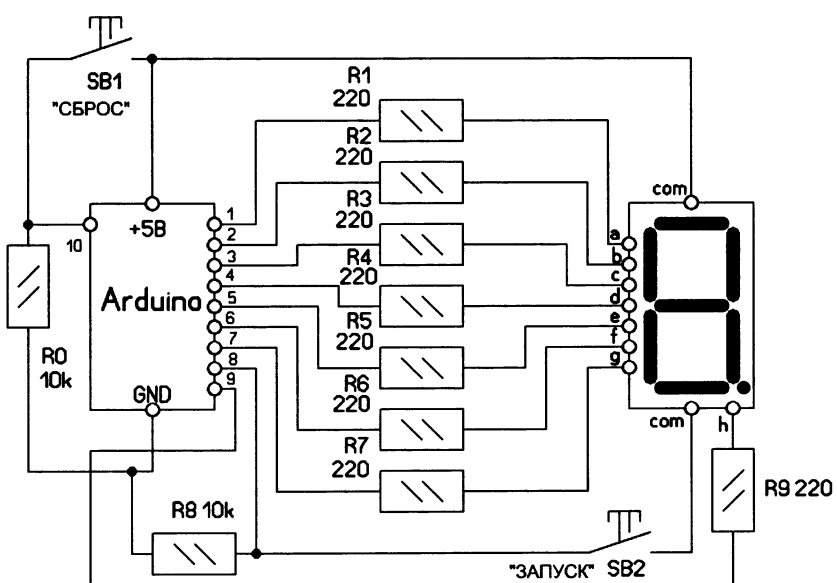

Рис. 21

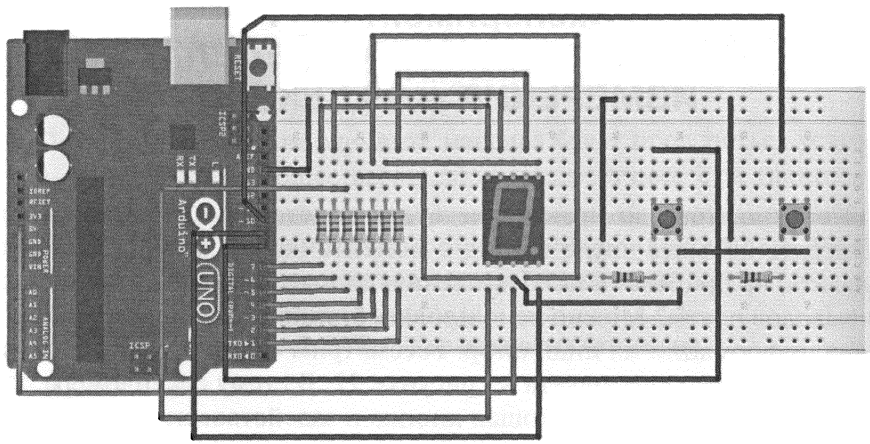

Рис. 22

боте. Если нажать и зафиксировать кнопку «ЗАПУСК» индикатор начнёт высвечивать текущие секунды 0-9 и повторно 0-9 и т.д. Если кнопку отпустить — на индикаторе попеременно будет высвечиваться число десятков секунд (сопровождается свечением точки) и единиц секунд (точка не горит). Таким образом, индикатор информирует нас о количестве секунд прошедшем между нажатием и отпусканием кнопки (SB2). Если повторно нажать кнопку отсчёт времени продолжится с ранее зафиксированного и до нового отпускания — считывания показаний. Для сброса зафиксированных значений нужно ненадолго нажать кнопку «СБРОС». На индикаторе вновь появится ноль. Максимальное значение, измеренное секундомером равно 99 секунд. Как и в предыдущем эксперименте, данный вариант имеет тоже погрешность порядка секунды. Ниже расположен скетч для платы.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
```

```
//
// Секундомер 99 секунд /
//

unsigned int k=7;
unsigned int i=0 ;
unsigned int r=0 ;
unsigned int n=0 ;
unsigned int f=0 ;


int tabl [70] ={1,1,1,1,1,1,1,0,  0,1,1,0,0,0,0,0,  1,1,0,1,1,0,1,  1,1,1,1,0,0,1,
0,1,1,0,0,1,1, 1,0,1,1,0,1,1,  1,0,1,1,1,1,1,1,  1,1,1,0,0,0,0,0,  1,1,1,1,1,1,1,1,
1,1,1,1,0,1,1};  // зашифрованная последовательность цифр

void setup()
{
    pinMode(10,INPUT);  //кнопка сброса показаний секундомера
    pinMode(8,INPUT);  //кнопка запуска секундомера
    pinMode(9, OUTPUT);
    digitalWrite(9,! LOW);

  for (i = 1; i <=7; i++)
     {
     pinMode(i, OUTPUT);
     digitalWrite(i,! LOW);
     }

}

void loop()
{
    if (digitalRead (10)==HIGH)  //если кнопка сброса нажата
      {
        k=0;
        n=0;
```

```
}

if (digitalRead (8)==LOW)  //если кнопка запуска отпущена
{
    f=7*n;
    r=k-7;
 for (i = 1; i <=7; i++)
   {
     digitalWrite(i,! tabl [f]);
     f=f+1;
   }
  digitalWrite(9,! HIGH);
    delay(250);



  for (i = 1; i <=7; i++)
    {
       ' italWrite(i,! tabl [r]);
      r=r+1;
    }
     digitalWrite(9,! LOW);

      delay(250);

 }
 if (digitalRead (8)==HIGH)  //если кнопка запуска нажата ...
  {

for (i = 1; i <=7; i++)
 {
  digitalWrite(i,! tabl [k]);   //включаем каждый сегмент по коду
  k=k+1; // производим последовательный перебор в массиве
    if (k==70)
     {
       k=0;
```

```
          n=n+1;
        if (n==10)
        {
          n=0;
        }
        }
    }
  delay(1000);
        }
}

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Функционирование программы аналогично ранее рассмотренным вариантам и будет полезнее, если читатель сам попробует разобраться в интерпретации её работы.

Эксперимент №12
«СВЕТОДИОДНЫЙ КУБ 3*3*3»

В интернете можно найти очень много описаний конструкций светодиодного LED_ куба с различным числом светодиодов в гранях. Они очень зрелищные и интересные, позволяют создавать трёхмерные световые эффекты. Чем больше светодиодов, тем интереснее, но одновременно и сложнее конструкция изделия. В нашем эксперименте рассмотрим самый простой и доступный в повторении вариант — из трёх светодиодов в ребре куба с общим числом 27. На рисунке 23 дано его изображение. Для успешного изготовления куба надо использовать одинаковые (желательно из одной партии) светодиоды. Что бы куб получился в форме куба, нужна матрица (1). Её проще сделать из куска листовой пластмассы, насверлив в ней отверстия диаметром 5 мм с определённым шагом. Значение его зависит от дли-

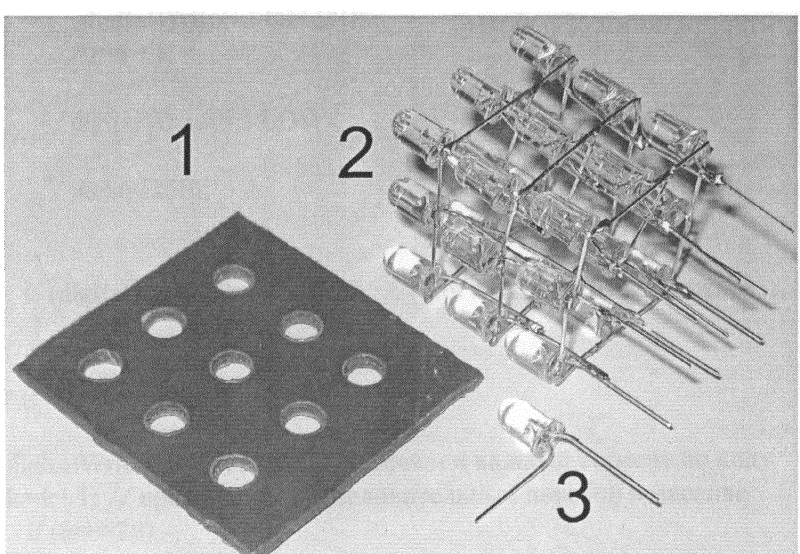

Рис. 23

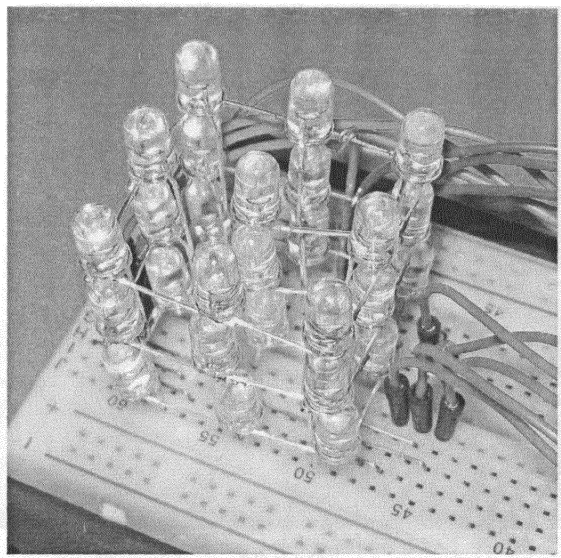

Рис. 24

ны выводов имеющихся светодиодов. В авторском варианте шаг составил 13мм (светодиоды попались «коротконогие»). Выводы изгибают под прямым углом (3). В нашем схемном решении гнуть нужно вывод катода светодиода (анод изгибать). Светодиоды вставляют в матрицу, катоды объединяют «обходя по кругу» и спаивают между собой. Так образуется слой-грань из 9 светодиодов. После спайки трёх слоёв, их собирают и соединяют пайкой в единый куб (2).

Такой компактный индикатор, возможно, разместить на одной макетной беспаечной плате (рис. 24). С помощью перемычек вывести каждый контакт на свободное пространство, рядом с кубом. И с помощью проводников-перемычек соединить с выводами Arduino (рис. 25). На рисунке 26 изображена схема подключения куба к плате. Подавая высокий логический уровень на выходы (10,11,12), например, 10 открываем транзистор VT1 и соответственно подключаем к «минусу» питания всю первую вертикаль светодиодов (эти светодиоды образуют первый слой). Теперь, подав на вывод 1 логический высокий уровень, мы включим светодиод HL1. Если добавить единицу

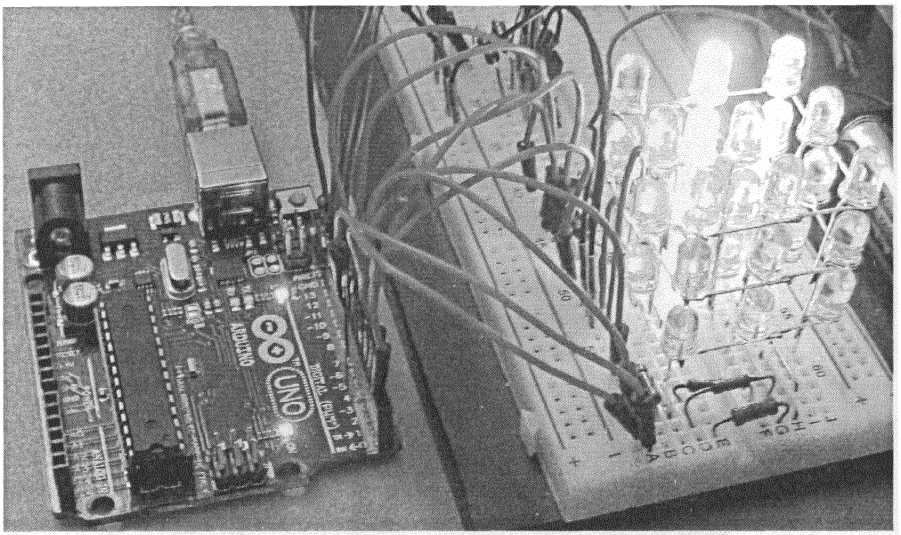

Рис. 25

на вывод 2 включится дополнительно светодиод HL4 и т.д. «Играя» на выводах логическими уровнями можно управлять свечением светодиодов куба. Вариант макетного исполнения (без изготовления куба) дан на рисунке 27. Взаиморасположение светодиодов в кубе относительно подключения номеров выводов платы дано на рисунке 28. Рассмотрим работу нескольких программ управления «лед кубом».

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// LED КУБ  змейка столбцов /
//

unsigned int i ;
```

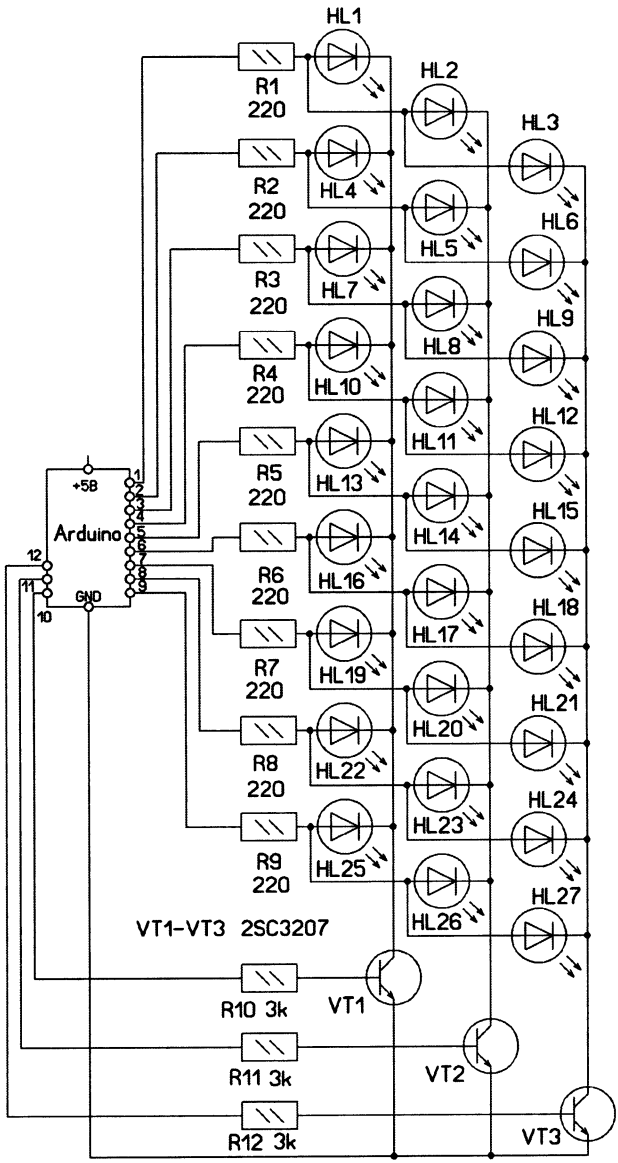
HL1
HL2
HL3
HL4
HL5
HL6
HL7
HL8
HL9
HL10
HL11
HL12
HL13
HL14
HL15
HL16
HL17
HL18
HL19
HL20
HL21
HL22
HL23
HL24
HL25
HL26
HL27
R1 220
R2 220
R3 220
R4 220
R5 220
R6 220
R7 220
R8 220
R9 220
+5В
Arduino
GND
VT1–VT3 2SC3207
R10 3k
R11 3k
R12 3k
VT1
VT2
VT3

Рис. 26

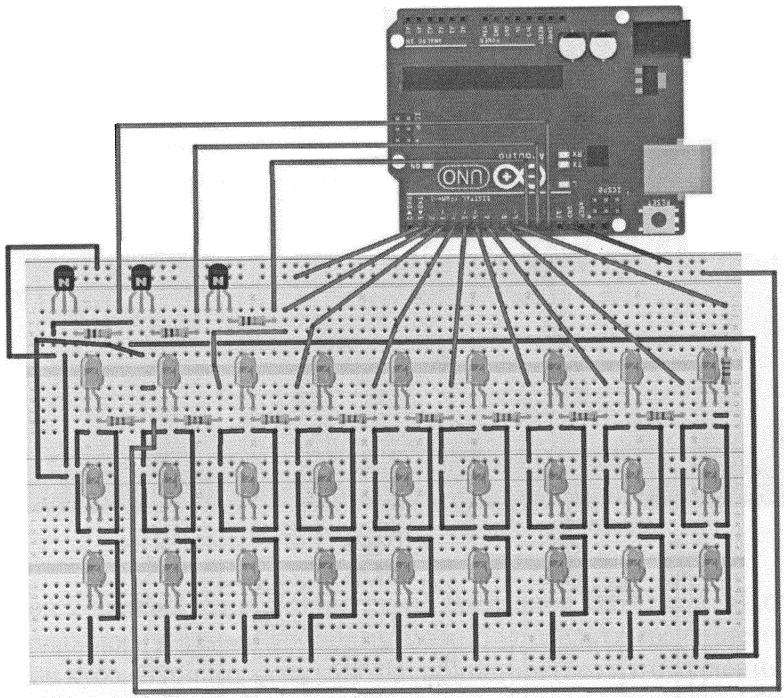

Рис. 27

```
unsigned int k ;

void setup()
{
 for (k = 10; k <=12; k++)
      {
      pinMode (k, OUTPUT);
      digitalWrite (k, HIGH );
      }
  for (i = 1; i <=9; i++)
      {
      pinMode (i, OUTPUT);
```

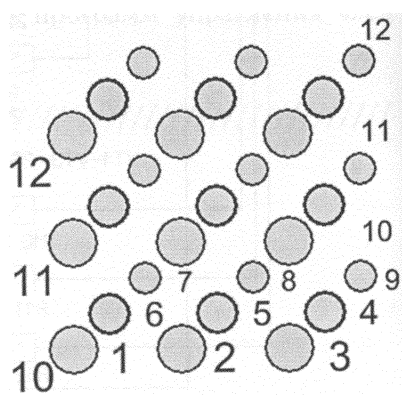

Рис. 28

```
    digitalWrite (i, LOW);
    }

}

void loop()
{
  for (i = 1; i <=9; i++)
    {
    digitalWrite (i, HIGH);
    delay (500);
    digitalWrite (i, LOW);

    }
}

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

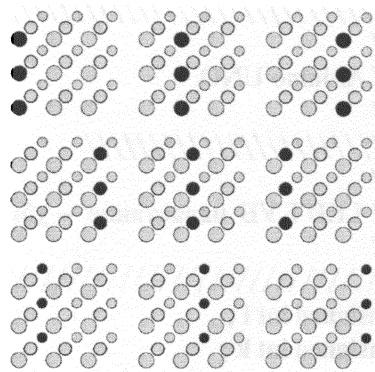

Рис. 29

Режим переключений поясняет рисунок 29 (порядок чтения — слева направо, сверху вниз). Скетч можно «расшифровать»:

1. Введём две переменные — счётчики выходов платы.
2. В цикле присвоим сначала выводам 10-12 статус выход и состояние — высокий уровень сигнала; затем выводам 1-9 статус выход и состояние — низкий уровень.
3. В бесконечном цикле введём цикл пересчёта выводов с 1 по 9.
4. Будем там последовательно присваивать соответствующему выводу высокий уровень, задерживать свечение столбца на полсекунды и выключать соответствующий вывод (при этом гаснет весь столбец)

Следующий пример позволяет последовательно наращивать светящиеся слои куба, а затем в обратном порядке их отключать. На рисунке 30 дано визуальное представление работы скетча:

```
//////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// LED КУБ  нарастание — убывание слоёв /
//

unsigned int i ;
unsigned int k ;

void setup()
{
 for (k = 10; k <=12; k++)
      {
      pinMode (k, OUTPUT);
      digitalWrite (k, LOW );
      }
  for (i = 1; i <=9; i++)
      {
      pinMode (i, OUTPUT);
      digitalWrite (i, HIGH );
      }

 }

void loop()
{
  for (k = 10; k <=12; k++)
      {
      digitalWrite (k, HIGH);
      delay (500);
      }
  for (k = 12; k >=10; k=k-1)
      {
```

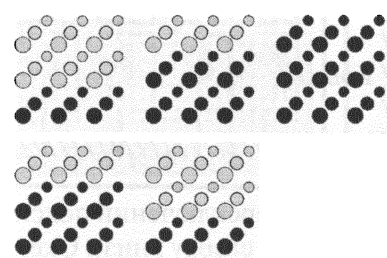

Рис. 30

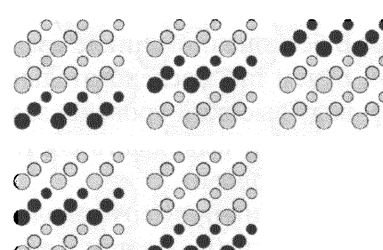

Рис. 31

```
    digitalWrite (k, LOW);
    delay (500);
    }
}

//
// Конец /
//
////////////////////////////////////////////////////////////////
```

Скетч можно «расшифровать»:

1. Введём две переменные — счётчики выходов платы.
2. В цикле присвоим сначала выводам 10-12 статус выход и состояние — низкий уровень сигнала; затем выводам 1-9 статус выход и состояние — высокий уровень.
3. В бесконечном цикле введём два последовательных цикла пересчёта выводов с 10 по 12 и в обратном порядке с 12 по 10.
4. Будем там последовательно присваивать соответствующему выводу высокий уровень, задерживать свечение слоя на полсекунды и переходить в цикле к следующему выводу — слою (предыдущий слой при этом остаётся включённым).
5. В следующем цикле наоборот будем отключать слои, задерживая каждый режим на полсекунды.

Ещё один режим свечения поясняет рисунок 31. «Бегущие слои с реверсом» так можно назвать этот режим. Программа имеет вид:

```
////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// LED КУБ  бегущие слои с реверсом /
//

unsigned int i ;
```

```
unsigned int k ;

void setup()
{
 for (k = 10; k <=12; k++)
      {
      pinMode (k, OUTPUT);
      digitalWrite (k, LOW );
      }
  for (i = 1; i <=9; i++)
      {
      pinMode (i, OUTPUT);
      digitalWrite (i, HIGH );
      }

}

void loop()
{
  for (k = 10; k <=12; k++)
      {
      digitalWrite (k, HIGH);
      delay (500);
      digitalWrite (k, LOW);
       }
  for (k = 12; k >=10; k=k-1)
      {
      digitalWrite (k, HIGH);
      delay (500);
      digitalWrite (k, LOW);

      }
}

//
// Конец /
```

```
//
//////////////////////////////////////////////////////////////////
```

Работа скетча похожа на работу предыдущих вариантов программ и не требует дополнительных пояснений.

Конечно, свечение куба будет гораздо более интересным, если скетч меняет режимы переключений, и скорости переключений автоматически с течением времени. Такой вариант управления кубом приведён ниже:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// LED КУБ  общая программа /
//

unsigned int i ;
unsigned int k ;
unsigned int j=100 ; // регулировщик частоты переключений

void setup()
{
 for (k = 10; k <=12; k++)
     {
     pinMode (k, OUTPUT);
     digitalWrite (k, LOW );
     }
  for (i = 1; i <=9; i++)
     {
     pinMode (i, OUTPUT);
     digitalWrite (i, HIGH );
     }
```

```
}

void loop()
{
  for (k = 10; k <=12; k++) // начало  фрагмента №3
      {
      digitalWrite (k, HIGH);
      delay (j);
      digitalWrite (k, LOW);
       }
  for (k = 12; k >=10; k=k-1)
      {
      digitalWrite (k, HIGH);
      delay (j);
      digitalWrite (k, LOW);

      }
  for (i = 1; i <=9; i++) // подготовка к фрагменту №1
      {
      digitalWrite (i, LOW);
      }

 for (k = 10; k <=12; k++) // начало фрагмента №1
      {
           digitalWrite (k, HIGH );
      }
  for (i = 1; i <=9; i++)
      {
      digitalWrite (i, HIGH);
      delay (j);
      digitalWrite (i, LOW);
       }
for (k = 10; k <=12; k++) // подготовка к  фрагменту №2
      {
           digitalWrite (k, LOW );
      }
```

```
  for (i = 1; i <=9; i++)
     {
          digitalWrite (i, HIGH);
     }
    for (k = 10; k <=12; k++) // начало фрагмента №2
     {
     digitalWrite (k, HIGH);
     delay (j);
     }
  for (k = 12; k >=10; k=k-1)
     {
     digitalWrite (k, LOW);
     delay (j);
     }
      if (j==500) // меняем частоту переключений от 100 мс до 500 мс
соответственно
           {
            j=100;
           }
          else
           {
             j=j+50;
           }

}
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Программа содержит все три режима переключений с изменяемой частотой их движения.

Наблюдательный читатель заметил, что во всех этих примерах светятся и гаснут либо столбцы, либо слои. А как быть если хочется зажечь, например, диагональ боковой грани? Здесь может помочь универсальная программа переключения. Рассмотрим на примере эф-

фекта «Вспышка» — рисунок 32. Итак у нас 8 картинок, где 27 светодиодов должны «внятно знать» какой уровень сигнала на них в данный момент «свечения картинки» подан. Воспользуемся для хранения логических уровней 9 каналов светодиодов в каждом слое массивом. Тогда 8 картинок можно зашифровать в 216 элементов. Остаётся лишь грамотно написать программу опроса данного массива-таблицы. Вот она:

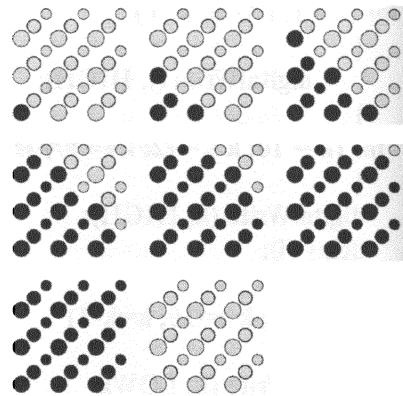

Рис. 32

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// LED КУБ универсальная программа — пример вспышка /
//

unsigned int i ;
 int k ;
unsigned int j ;
unsigned int n ; // в таблицу-массив ниже вносим данные для этого
примера
int tabl [216]= {1,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,
                 1,1,0,0,0,1,0,0,0,  1,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,
                 1,1,1,0,1,1,1,0,0,  1,1,0,0,0,1,0,0,0,  1,0,0,0,0,0,0,0,0,
                 1,1,1,1,1,1,1,1,0,  1,1,1,0,1,1,1,0,0,  1,1,0,0,0,1,0,0,0,
                 1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,0,  1,1,1,0,1,1,1,0,0,
                 1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,0,
                 1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,
```

```
0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0 };

void setup()
{
 for (k = 10; k <=12; k++) // объявляем все выходы и задаём начальное
состояние
     {
     pinMode (k, OUTPUT);
     digitalWrite (k, LOW );
     }
  for (i = 1; i <=9; i++)
     {
     pinMode (i, OUTPUT);
     digitalWrite (i, LOW);
     }

}

void loop()
{
     k=-1;
  for (n = 1; n <=8; n++) // определяет перебор картинок — пример 8
картинок
     {
     for (j = 1; j <=25; j++) // определяет длительность свечения 1
картинки
     {
          digitalWrite (10, HIGH);
        for (i = 1; i <=9; i++)
         {
            k=k+1;
           digitalWrite (i, tabl [k]);

         }
           delay(5);
           digitalWrite (10, LOW);
```

```
            digitalWrite (11, HIGH);
        for (i = 1; i <=9; i++)
         {
             k=k+1;
            digitalWrite (i, tabl [k]);
          }
            delay(5);
            digitalWrite (11, LOW);
            digitalWrite (12, HIGH);
        for (i = 1; i <=9; i++)
         {
             k=k+1;
            digitalWrite (i, tabl [k]);

          }
            delay(5);
            digitalWrite (12, LOW);
            k=k-27;
       }
           k=k+27;
   }
}

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Теперь можно создавать свои собственные эффекты, рисовать картинки по образцу (рис. 32) и шифровать их в массив данных. Например, первая картинка по данному рисунку в массиве занимает первую строчку: 1,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0, а последняя соответственно так же — 0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0.

Эксперимент №13
«ЗВУКОВОЙ СИГНАЛ SOS»

Ещё одной полезной функцией языка программирования Arduino можно считать функцию tone(). Она формирует меандр с заданной частотой и подаёт его на выбранный номер вывода. У неё три аргумента — номер вывода, на который подаётся волна, частота генерируемого звука, длительность (в мсек) звучания. Если последнее значение не указано, звук отключают дополнительной командой noTone(). Особенность функции tone() такова что после её вызова программа будет выполняться дальше, а звук будет носить фоновый характер.

Рассмотрим её практическое применение на примере подачи звукового сигнала SOS. Ранее мы рассматривали пример подачи такого сигнала световым источником. Соберём схему по рисунку 33. Переменный резистор в роли потенциометра регулирует громкость звучания динамика B1. Сопротивление его обмотки 8 Ом, мощность 0,5 Вт. Резистор R1 токаограничительный. Вариант макетки для симулятора или реального эксперимента дан на рисунке 34. В авторском исполнении макетирование имеет вид по рисунку 35. Рассмотрим скетч:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
```

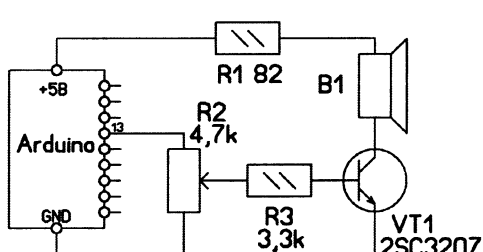

Рис. 33

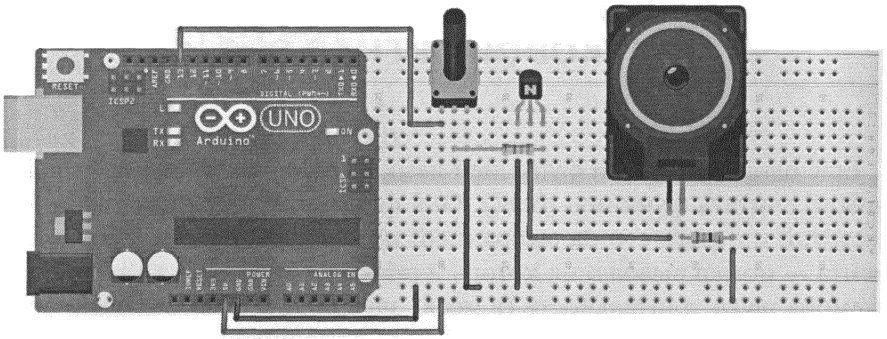

Рис. 34

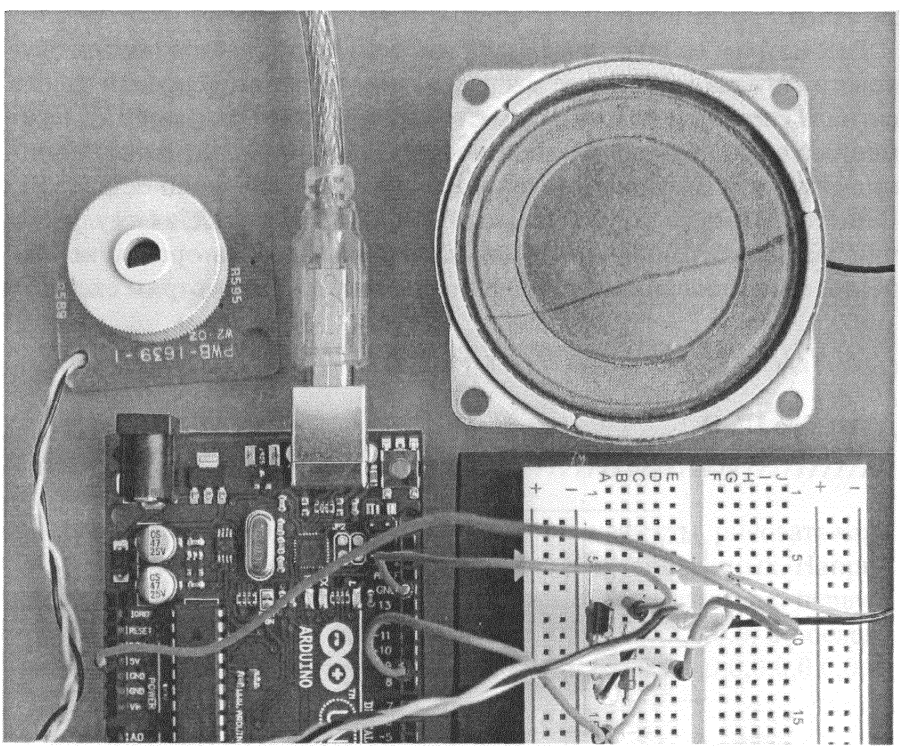

Рис. 35

```
//
//////////////////////
//
// Программа передачи звукового  сигнала SOS /
//

unsigned int n = 0;
unsigned int k = 0;
unsigned int j = 0;
unsigned int r = 0;

int tabl [18] ={1,1,1,1,1,3,3,1,3,1,3,3,1,1,1,1,1,7}; // зашифрованный
сигнал

void setup()
{
      pinMode(13, OUTPUT);
      digitalWrite(13, LOW);
 }

void loop()
{
   for (n = 0; n <18; n=n+2)
     {
      k=tabl[n];
      j=tabl[n+1];
     tone (13, 700, 100*k) ;
     r=100*(k+j);
     delay(r);
     }
}

//
// Конец /
//
////////////////////////////////////////////////////////////////
```

В массиве зашифрованы длительности звуков и пауз между ними. Счётчик k определяет длительность звука, присваивая себе соответствующее значение элемента массива. Счётчик j определяет длительность паузы, работая аналогично счётчику k. Счётчик r управляет длительностью звука и паузы, следующей за ним. Строку программы — **«tone (13, 700, 100*k);»** трактуем: подать на вывод 13 звук частотой 700Гц на время k раз по 100 миллисекунд. Команда delay (r) «замораживает» программу на время звука и паузы после него. Счётчик n перебирает все элементы массива.

Эксперимент №14
«МОЯ МЕЛОДИЯ»

Интересным практическим приложением рассмотренной выше функции может быть программа-проигрыватель мелодий. Зная соответствия частоты звука и ноты в октаве, а так же их последовательность перебора в мелодии (с учётом времени пауз) можно заполнять массивы-таблицы. Перебирая программно элементы этих массивов можно проигрывать несложные мелодии. Таблица нот-частот дана на рисунке 36. Ниже дан скетч для исполнения авторской мелодии (схема конструкции проигрывателя рис. 33 — неизменная). Дополнительно (в сравнении с предыдущим скетчем) в программу введён счётчик h, меняющий темп звучания мелодии. Он трёхступенчатый.

1 октава	Обозначение	Частота, Гц	2 октава	Обозначение	Частота, Гц
до	C	261	до	c	523
до-диез	C#(R)	277	до-диез	c#(r)	554
ре	D	293	ре	d	587
ре-диез	D#(S)	311	ре-диез	d#(s)	622
ми	E	329	ми	e	659
фа	F	349	фа	f	698
фа-диез	F#(T)	370	фа-диез	f#(t)	740
соль	G	392	соль	g	784
соль-диез	G#(U)	415	соль-диез	g#(u)	830
ля	A	440	ля	a	880
си-бимоль	B	466	си-бимоль	b	932
си	H	494	си	h	988

Рис. 36

```
//////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Универсальная программа проигрывания мелодий — пример моя
мелодия/
//

unsigned int n = 0;
unsigned int k = 0;
unsigned int j = 0;
unsigned int r = 0;
unsigned int s = 0;
unsigned int h = 0;

int tabl [189] ={ 532,6,3,  532,6,3,  440,6,3,
                  523,12,3,  440,6,3,  523,12,3,
                  440,6,3,  494,12,6,  494,6,3,
                  494,6,3,  415,6,3,  494,12,3,
                  415,6,3,  494,12,3,  415,6,3,
                  440,12,6,  659,6,3,  659,6,3,
                  554,6,3,  659,12,3,  554,6,3,
                  659,12,3,  554,6,3,  587,12,6,
                  698,6,3,  587,6,3,  494,6,3,
                  523,12,3,  440,6,3,  415,6,3,
                  370,6,3,  415,6,3,  440,12,24,
                  523,4,2,  523,4,2,  494,4,2,
                  440,4,2,  494,4,2,  523,4,2,
                  587,8,2,  523,4,2,  587,12,2,
                  587,8,2,  523,4,2,  587,12,2,
                  523,8,2,  494,4,2,  523,12,2,
                  523,4,2,  523,4,2,  494,4,2,
                  440,4,2,  494,4,2,  523,4,2,
                  587,8,2,  523,4,2,  587,12,2,
```

```
                587,8,2,  523,4,2,  494,4,2,
                 440,4,2,  415,4,2,  440,12,24}; // зашифрованные данные
нот, длительности нот и пауз

void setup()
{
    pinMode(13, OUTPUT);
    digitalWrite(13, LOW);
}

void loop()
{
  for (h = 20; h <=40; h=h+10) // смена темпа звучания
    {

      for (n = 0; n <189; n=n+3)
       {
          k=tabl[n]; // нота-частота звука
          j=tabl[n+1]; // условная длительность звучания
          s=tabl[n+2]; // условная длительность паузы

         tone (13, k, h*j) ;
         r=h*(j+s);
         delay(r);
    }
  }
}

//
// Конец /
//
////////////////////////////////////////////////////////////////
```

В массиве каждые три подряд элемента «отвечают» за звучание ноты. Первый задаёт частоту звука, второй условную длительность звучания ноты, третий — условную длительность паузы. Реальную дли-

тельность исполнения-паузы каждой ноты задаёт в программе строка-выражение: «**r=h*(j+s);**».

Теперь, используя таблицу рисунок 36 и зная набор нот своей мелодии можно заполнять массив в программе под свои музыкальные пристрастия. И озвучивать их с помощью данной программы.

Эксперимент №15 «ОДНОГОЛОСЫЙ ЭМИ или ARDUINO-ДУДКА»

Теперь наши познания в программировании контроллера платы позволяют нам создать простенький музыкальный инструмент (забегая вперёд — очень напоминающий тональный набор телефонного номера). Итак, обратимся к рисунку 37. Основное место схемы на нём занимает матрица кнопочной клавиатуры 5 на 5. Всего инструмент позволяет исполнять при нажатии по очереди каждой из кнопок 25 нот. Длительность звучания ноты постоянная при кратковременном нажатии и сколь угодно долгая при удержании контактов кнопки в замкнутом положении. Выводы с 6 по 10 запрограммированы как выходы, с 1 по 5 как входы. Программа отслеживает, какая кнопка в данный момент нажата и «включает» соответствующую ей ноту. Диоды VD1-VD5 развязывающие, они защищают выходы платы от случайного замыкания при одновременном нажатии нескольких клавиш. В программе (ниже) вы увидите значок «&» (встречается и двойное «&&»). Это логический оператор И — он позволяет объединять два условия одновременное выполнение которых даёт команду на дальнейшее исполнение программы.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа для одноголосого ЭМИ/
//

unsigned int n = 0;
unsigned int i = 0;
unsigned int j = 500;
```

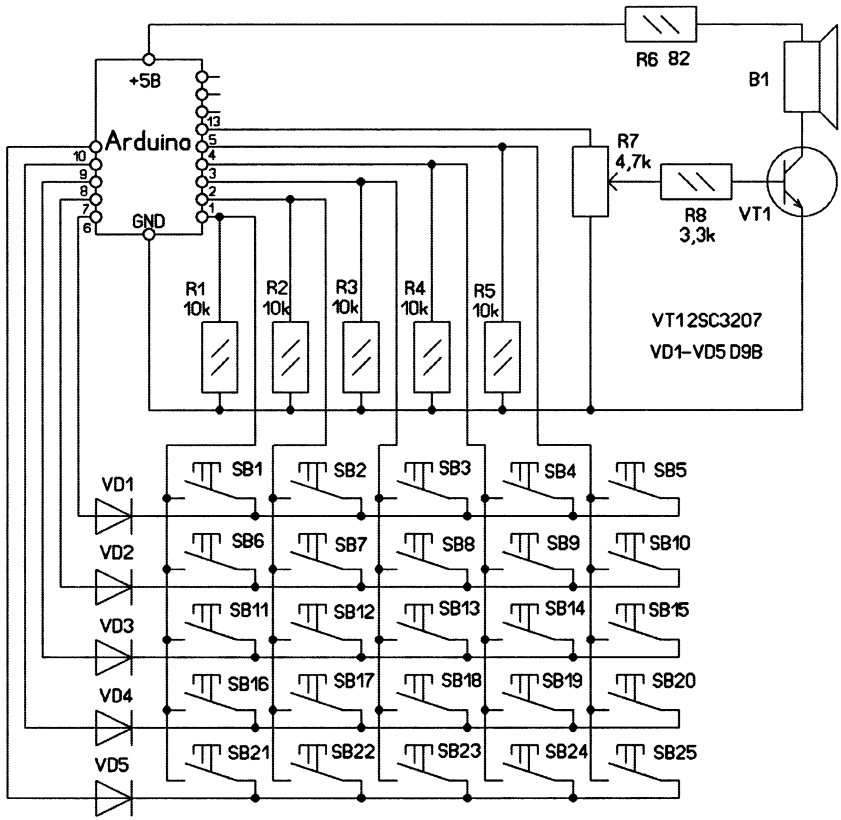

Рис. 37

```
unsigned int k = 0;

int tabl [25] ={261,277,293,311,329,
                349,370,392,415,440,
                466,494,523,554,587,
                622,659,698,740,784,
                830,880,932,988,1046 }; // зашифрованные данные
нот — их частоты
```

```
void setup()
{
      pinMode(13, OUTPUT);
      digitalWrite(13, LOW);
      for (i =6 ; i <=10; i=i+1)
       {
         pinMode(i, OUTPUT);
        digitalWrite(i, LOW);
       }
      for (i =1 ; i <=5; i=i+1)
       {
         pinMode(i, INPUT);
       }

}
```

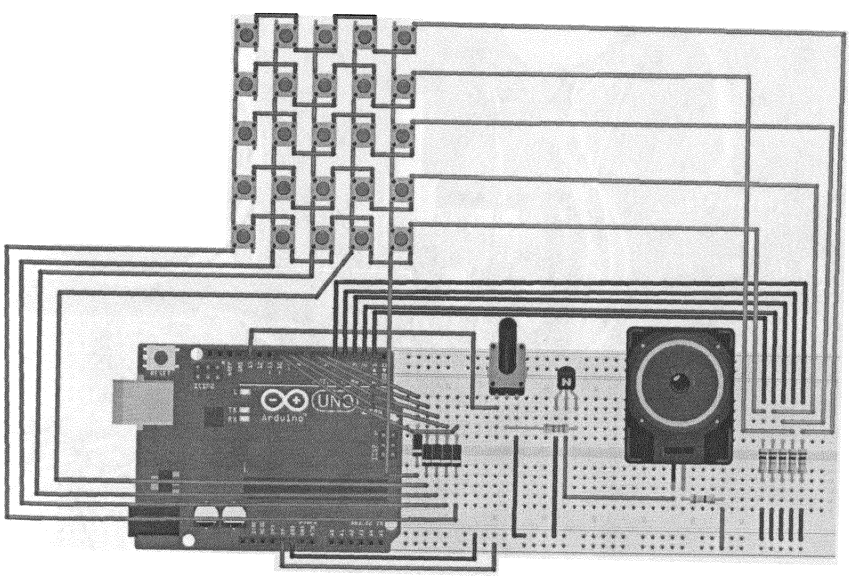

Рис. 38

```
void loop()
{
      for (i =6 ; i <=10; i=i+1) //последовательно-попеременная подача
на выходы логической единицы
          {
          digitalWrite(i, HIGH);
          for (n =1 ; n <=5; n=n+1) // перебор входов
          {
              if (digitalRead (i)==HIGH & digitalRead (n)==HIGH) //
проверка замкнута ли кнопка ?
                {
                    k=5*(i-6) + (n-1);
                    tone (13, tabl[k],j); // исполнение вызванной ноты
                    delay (j);
```

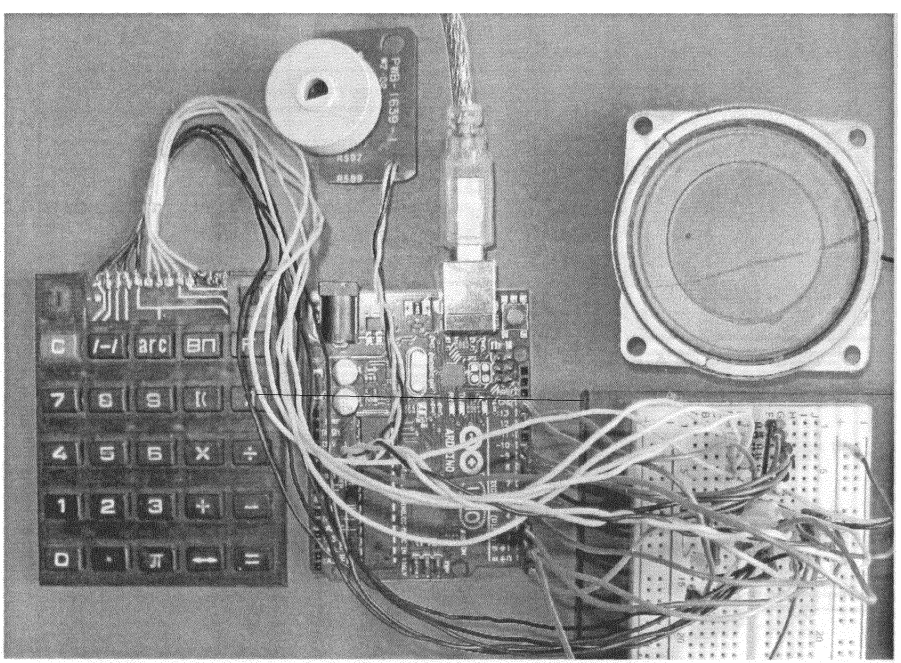

Рис. 39

```
            }
          }
          digitalWrite(i, LOW);


        }

}

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Макетные вариации для повторения приведены на рисунках 38, 39. А на этом в данной главе остановимся, так сказать для осмысления и понимания процесса самостоятельного программирования...

ГЛАВА 3.
«АРДУИНО-ПРОБЫ»

Вот и настал момент узнать практическое назначение выводов A0-A5 на плате. При «штатном использовании» это аналоговые входы. Наша плата Arduino имеет 6 каналов с 10-битным аналого-цифровым преобразователем (АЦП). Напряжение, поданное на аналоговый вход, обычно от 0 до 5 вольт будет преобразовано в значение от 0 до 1023, что составляет 1024 шага с разрешением 0.0049 Вольт. Разброс напряжения и шаг может быть изменен функцией **analogReference()**. Но о ней в другой раз.

Рассмотрим работу функции **analogRead()** на примере регулировки яркости свечения светодиода (рис. 1). Она позволяет преобразовывать значение величины входного напряжения в соответствующее число из интервала 0-1023. Итак:

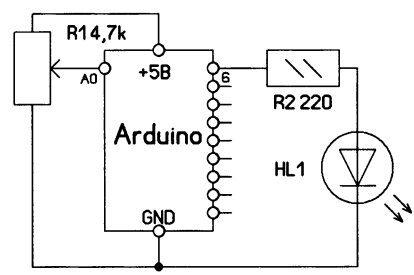

Рис. 1

Эксперимент №16
«ОПЯТЬ СВЕТОДИОД»

На рисунке 2 дан вариант макетной реализации схемы. Светодиод плавно набирает яркость свечения, затем плавно уменьшает — далее цикл повторяется. Изменяя положение ручки потенциометра можно менять скорость нарастания и убывания яркости свечения. Рассмотрим скетч.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//Плавное переключение яркости свечения светодиода с регулятором частоты//
unsigned int  n=0;
 int  k=1;
unsigned int  x=0;
unsigned int  j=0;
```

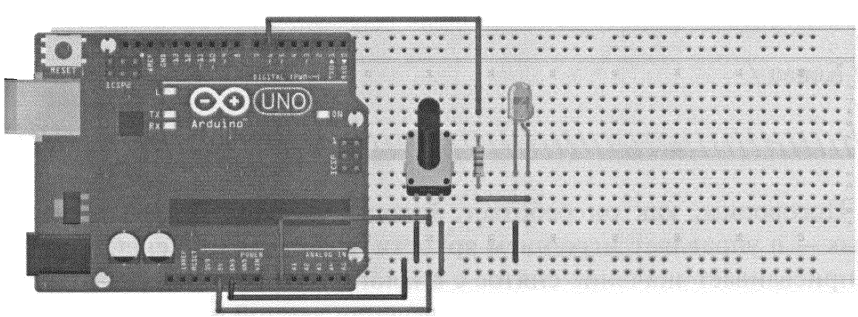

Рис. 2

```
void setup()
  {
     pinMode(6, OUTPUT); //канал сигнала для светодиода /
     analogWrite(6,0);
     pinMode(A0, INPUT); // регулятор частоты/
  }
     void loop()
    {

        analogWrite(6,n);
        delay(j);
        x=analogRead(A0);
        j=map(x, 0, 1023, 20, 2);
        n=n+k;

            if ( n>=255)
            {
              k= -1 ;
             }
             if ( n<=0)
            {
              k= 1 ;
             }

    }

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Поговорим, как он работает. Вначале вводим четыре переменных — n управляет значением яркости, k счётчик ступеней яркости, x присваивает значение снятое с аналогового входа A0, j определяет (в миллисекундах) длительность свечения данной яркости светодиода. Смысл строки **«x=analogRead(A0);»** — x присвой значение в диа-

пазоне 0-1023 сообразно напряжению на аналоговом входе (от 0 до 5 Вольт). Строкой ниже видим новую математическую функцию map(). Функция пропорционально переносит значение из текущего диапазона значений в новый диапазон, заданный параметрами. То есть строка **«j=map(x, 0, 1023, 20, 2);»** по смыслу читается — переменной j присвой соответствующее значение x, при этом учти, что x принимает значения от 0 до 1023, значит j сообразно этому диапазону, может принимать значения от 20 до 2.

Фрагмент скетча **«n=n+k; if (n>=255) { k= -1 ; } if (n<=0){k= 1 ;}»** имеет трактовку: счётчику n увеличивай значение на единицу пока оно не дойдёт до 255, потом уменьшай на единицу, пока не обратится в ноль. Далее повторяй по кругу.

Как это всё работает и взаимосвязано можно посмотреть, перезагружая скетч в плату, меняя несколько раз значения 1023 (в сторону уменьшения), значения 20 в сторону увеличения. Применим «каркас данного скетча» в двух ниже следующих конструкциях.

Эксперимент №17
«НОЧНИК — ТРИ ЦВЕТА»

Об этой конструкции, реализованной на таймере 555, я уже рассказывал на страницах своей книги «Роботы своими руками. Игрушечная электроника». Рассмотрим её практическую реализацию на Ардуино. Схемное решение дано на рисунке 3. Плавная ШИМ-модуляция создаёт относительно медленное перетекание «цвета света» в трубочке от основного (красный, зелёный, синий) к общему фоновому (белый). Макет и его внешний вид представлен на рисунках 4 и 5. Основа изделия три трубки молочного цвета для коктейля, в торцы которых вставлены по два светодиода. Трубочки длиной около 10 см. Снаружи головки вставленных светодиодов, желательно обмотать чёрной изолентой — для исключения нежелательной засветки боковых поверхностей. Между трубочек можно проложить картонные

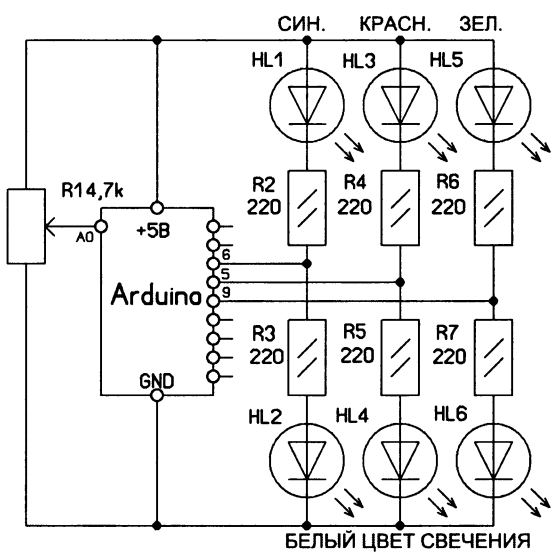

Рис. 3

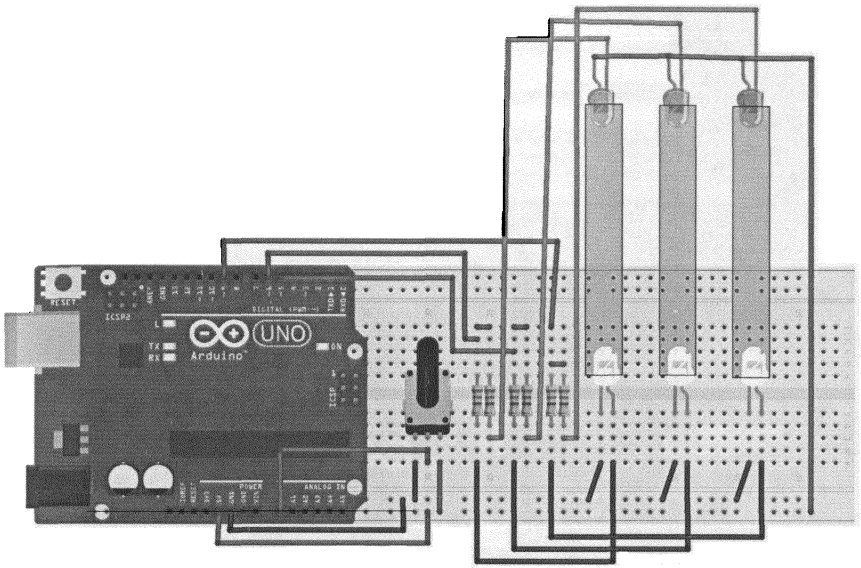

Рис. 4

вкладыши, также уменьшающие засветку. Программа для конструкции обеспечивает четыре смены динамики свечения, переменный резистор регулирует амплитуду перелива света и его скорость (частоту смены). Реализовано два основных режима — синхронное нарастание цвета в трубочке (выдавливание белого цвета красным, зелёным, синим с реверсом) и поочерёдное выдавливание в каждой трубке (перелив цветов). Между этими режимами — статичное, пятисекундное, синхронное свечение каждого (основного) цвета в трубочке. Программа представлена ниже.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
```

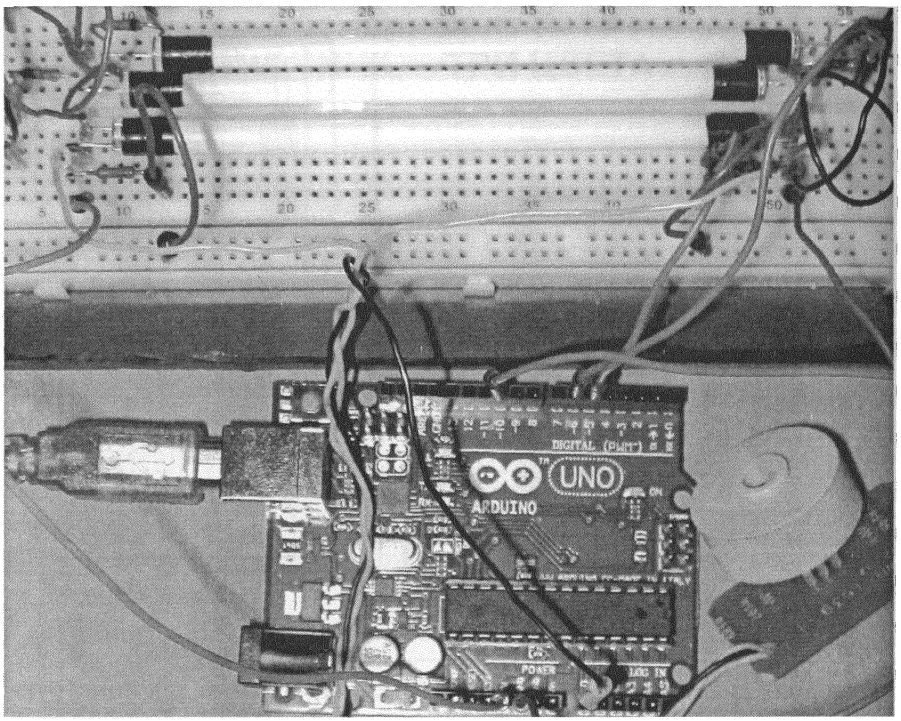

Рис. 5

```
//Свечение трёх трубочек с регулятором амплитуды перелива света и
частоты//
unsigned int  n1=0;
unsigned int  n2=0;
unsigned int  n3=0;

 int  k1=1;
 int  k2=1;
 int  k3=1;
 int  i=0;

long  y1=0;
```

```
long y2=0;
unsigned int x=0;
unsigned int j=0;
unsigned int z=0;
unsigned int r=0;

void setup()
  {
     pinMode(6, OUTPUT); // канал сигнала для 1 трубки /
     analogWrite(6,0);
     pinMode(5, OUTPUT); // канал сигнала для 2 трубки /
     analogWrite(5,0);
     pinMode(9, OUTPUT); // канал сигнала для 3 трубки /
     analogWrite(9,0);
     pinMode(A0, INPUT); // регулятор амплитуды и частоты/
  }
     void loop()
    {

        x=analogRead(A0); y2=millis();
        z=map(x, 0, 1023, 0, 127); // переменная, ограничивающая min
яркости
        j=map(x, 0, 1023, 20, 2); // переменная, определяющая длитель-
ность задержки
        r=map(x, 0, 1023, 255, 128); // переменная, ограничивающая
max яркости
        n1=n1+k1; n2=n2+k2; n3=n3+k3; //счётчики ступеней измене-
ния яркости

      if ( n1>=r) { k1= -1 ;} if ( n1<=z) {k1= 1 ; }//условия смены на-
правления счёта
      if ( n2>=r) { k2= -1 ;} if ( n2<=z) {k2= 1 ; }//ступеней изменения
яркости
      if ( n3>=r) { k3= -1 ;} if ( n3<=z) {k3= 1 ; }
      if (y2-y1>=5000&i==0){n1=z+1;n2=127-z;n3=r;y1=y2;i=1;}//
условия смены
```

```
    if (y2-y1>=30000&i==1){ i=2;y1=y2;}                    //програм-
мы переключения
    if (y2-y1>=5000&i==2){n1=r;n2=r;n3=r;y1=y2;i=3;}
    if (y2-y1>=30000&i==3){ i=0;y1=y2;}

    if(i==0 || i==2){analogWrite(6,25); analogWrite(5,25);
analogWrite(9,25);}
    else
    {
     analogWrite(6,n1);
     analogWrite(5,n2);
     analogWrite(9,n3);
     }
     delay(j);

  }

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Разобраться досконально с работой программы удобнее экспериментируя с ней, например, меняя числовые параметры задания интервала времени смены программ (числа 30000 и 5000).

При настройке конструкции, возможно, понадобиться подобрать номиналы резисторов R2-R7 до получения желаемой «гаммы цветов».

Эксперимент №18 «ЦВЕТОВЫЕ ЧАСЫ С СЕКУНДНЫМ ОТСЧЁТОМ»

В предыдущей главе нам не удалось существенно улучшить динамику свечения «песочных часов». Заменим один индикатор на другой — индикатор ночника, и трансформируем управляющий скетч. Добавим в схему кнопку управления запуска часов и уберём потенциометр согласно схеме по рисунку 6. Вариант внешнего вида макета дан на рисунке 7. В итоге:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
```

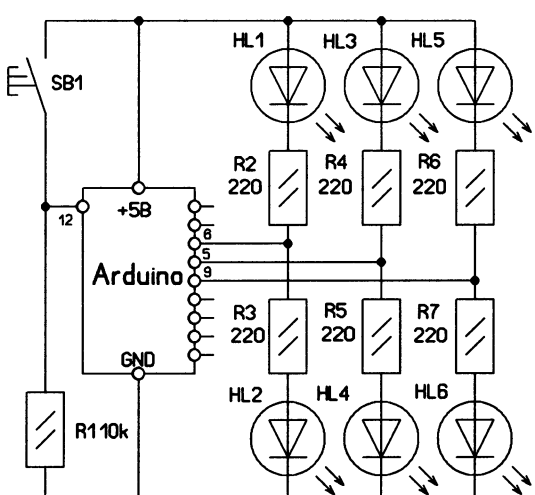

Рис. 6

```
//
//Цветовые часы на 1 минуту с отсчётом секунд//
unsigned int  n=0;
unsigned  int  i=0;
unsigned  int  k;


void setup()
  {
     pinMode(5, OUTPUT); //трубка красного свечения /
     pinMode(6, OUTPUT); // трубка синего свечения /
     pinMode(9, OUTPUT); // трубка зелёного свечения /
     pinMode(12,INPUT);  // кнопка запуска часов/

     analogWrite(6,n);
     analogWrite(5,n);
     analogWrite(9,n);
  }
     void loop()
    {
       if (digitalRead (12)==HIGH)  //
если часы запущены ...
```

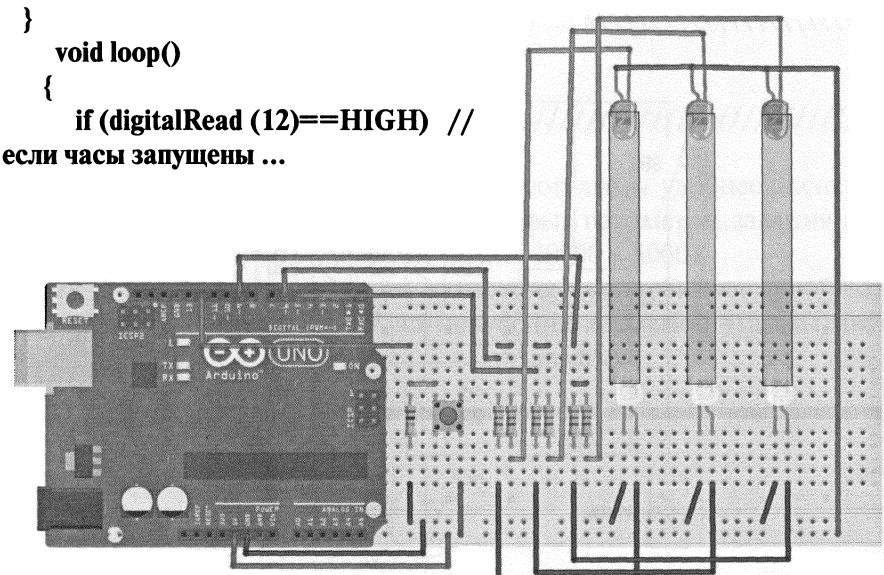

Рис. 7

```
        {

      for (n = 0; n <= 14; n=n+1)
       {
       i=17*n;
            for (k = 1; k <= 4; k=k+1)
                 {
                digitalWrite(5, HIGH);
                digitalWrite(6,HIGH);
                digitalWrite(9,HIGH);
                delay(150);
                analogWrite(5,i);
                analogWrite(6,i);
                analogWrite(9,i);
                delay(850);        //задержка свечения данной яркости на 4
сек. за цикл
                    }
                 }
            }
       }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Как и в исходном варианте, отсчёт времени минуты поделен на 15 отрезков по 4 секунды в каждом. Соответственно яркость нарастания белого цвета и убывания красного — зелёного — синего имеет 15 ступеней. Каждую секунду все «цветные» светодиоды гаснут на 150 миллисекунд, информируя о её начале. Запуск часов осуществляется кратковременным нажатием кнопки. При окончании отсчёта времени мигания прекращаются, все трубки светят белым светом. При повторном запуске, все трубки окрашиваются в «свой цвет» и происходит его плавное убывание в течении минуты до «всеобъёмного» белого.

Эксперимент №19 «ДЛИННОМЕР»

Рассмотренные ниже два примера носят чисто обучающий характер и к практической составляющей конструирования относятся слабо. На рисунке 8 приведена схема «длинномера» — прибора, предназначенного для измерения длины. Правда интервал измерений невелик — несколько десятков миллиметров и результат выводится не на цифровое табло, а на «аналоговый микроамперметр». Сведущий читатель спросит — «А зачем между потенциометром и микроамперметром Ардуино?» Отвечу — «Чтобы не подбирать номинал резистора R1».

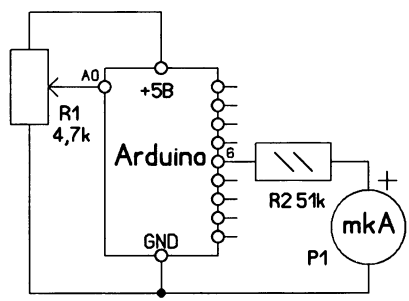

Рис. 8

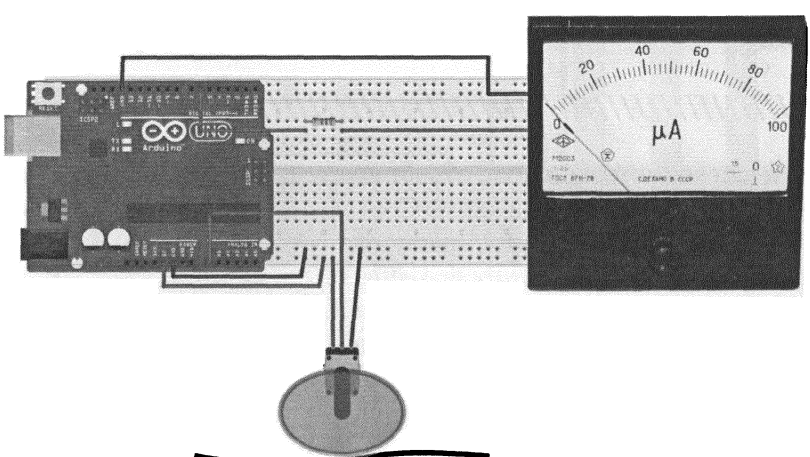

Рис. 9

Шутки в сторону, разберем, как это работает. Итак, R1 теперь полноценный датчик — устройство, предназначенное для измерения (контроля) и преобразования одной физической величины в другую. Макет изделия, представленный на рисунке 9, пропорционально преобразует пройденный колёсиком потенциометра путь в величину силы тока, протекающего через микроамперметр. Рассмотрим работу программы данной конструкции.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//Длинномер: программа работы //
 unsigned int  n=0;
 unsigned int  x=0;

void setup()
  {
     pinMode(6, OUTPUT); //канал сигнала для микроамперметра /
     pinMode(A0, INPUT); // датчик перемещения/
  }
     void loop()
    {
       x=analogRead(A0);
       n=map(x, 0, 1023, 0, 170);
       analogWrite(6,n);
     }

//
// Конец /
//
//////////////////////////////////////////////////////////////////
```

В зависимости от положения ротора резистора напряжение на аналоговом входе A0 будет меняться в пределах от 0 до 5 Вольт. Соответственно переменная x в программе будет меняться в пределах от 0 до 1023. Величина «усреднённого тока» на выходе 6 (это то что будет фиксировать микроамперметр) будет находиться в пределах от 0 до 60 мкА. Ключевым числом, позволяющим сопоставлять, пройденные колёсиком миллиметры длины в микроамперы показаний прибора является число — 170 (строка «**n=map(x, 0, 1023, 0, 170);**»). Поясним, как это сделать на практике. Итак, берём произвольное колесо, диаметром около 2-3 см. Одеваем его на вал потенциометра (имеющегося переменного резистора), прокручиваем по линейке от одного крайнего положения до другого крайнего положения ротора. Запоминаем длину. Пусть это будет 60 мм. Следовательно, на шкале микроамперметра максимальный ток должен быть 60 мкА, а не 100 как ограничивает это значение резистор R2. Как быть? Просто уменьшаем максимально возможное «усреднённое по времени» значение напряжения на выходе 6 с 5 В до 3В. Для чего в выше упомянутой строке пишем 170, а не 255. Иными словами практически — программно подбираем это значение под имеющееся колесо, резистор и микроамперметр. Финальное исполнение возлагаем на строку программы — **«analogWrite(6,n);».**

Конечно, точность такого прибора не велика по ряду причин. Резистор не линейный попался. Ротор вращается по краям на контактах, требуемый номинал резистора R2 зависит от сопротивления головки прибора и т.д. Но наша цель увидеть как легко, благодаря простому скетчу можно переводить показания одной шкалы в показания другой...

Эксперимент №20
«КОДОВЫЙ ЗАМОК»

Еще одна конструкция шуточная — кодовый замок. Схемное и макетное решение дано на рисунках 10 и 11 соответственно. Комбинаций шифра всего 1000 (от 000 до 999). Исполняющее устройство — соленоид, втягивающий сердечник. Нагрузка такая может быть силь-

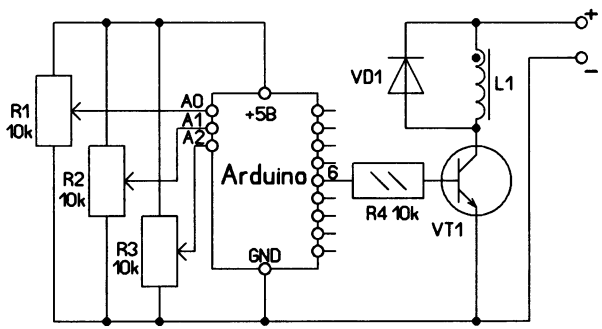

Рис. 10

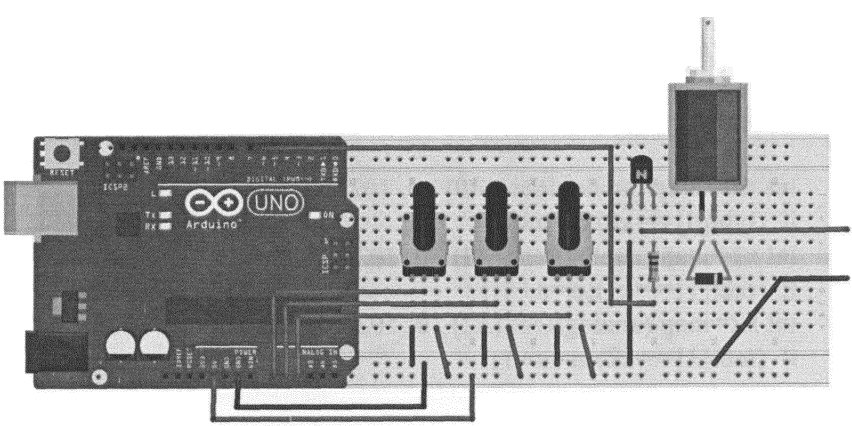

Рис. 11

ноточной, поэтому лучше питать её от отдельного источника. Работает замок так. Выкручивая на цифровых шкалах потенциометров нужную комбинацию, добиваемся срабатывания сердечника соленоида. Замок открывается на 5 секунд. Если за это время сменить комбинацию он закроется до вновь правильно набранного шифра. В скетче ниже эта комбинация — шифр 005.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//Кодовый замок — игрушка //
unsigned int n = 0;
int tabl [3] ={0,0,5};// комбинация цифр ключа
void setup()
  {
     pinMode(6, OUTPUT); //канал сигнала для лампы /
     digitalWrite(6, LOW);

    pinMode(A0, INPUT); // набор 1 цифры/
    pinMode(A1, INPUT); // набор 2 цифры/
    pinMode(A2, INPUT); // набор 3 цифры/
  }
     void loop()
    {
        int  x0=map (analogRead(A0) , 0, 1023, 0, 9);
        if  (tabl[0]==x0){n=n+1;}
        int x1=map (analogRead(A1) , 0, 1023, 0, 9);
        if  (tabl[1]==x1){n=n+1;}
        int x2=map (analogRead(A2) , 0, 1023, 0, 9);
        if  (tabl[2]==x2){n=n+1;}


       if  (n==3){digitalWrite(6, HIGH);}
```

```
        delay(5000); digitalWrite(6, LOW);n=0;
    }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

В основной части программа постоянно сравнивает каждую, «снятую с аналогового входа цифру» с «правильной» цифрой ключа. При совпадении счётчик n увеличивается на единицу. При накоплении значения в 3 на вывод 6 подаётся логическая единица — замок открыт на 5 секунд. Далее счётчик обнуляется, замок закрывается, идёт повторная проверка кода. Если код верный происходит повторное открывание замка. На практике, при выставленном коде, замок остаётся открытым постоянно, замыкается лишь при его смене на потенциометрах R1-R3. Разрядность шифра замка можно увеличивать, увеличивая число переменных резисторов, аналогично добавляя строки в программу, увеличивая массив для хранения ключа и максимальное значение счётчика n. Для удобства экспериментирования резисторы можно вынести на отдельную панель, снабжённую шкалами цифр на подобии (рис. 12).

Рис. 12

Эксперимент №21
«ПРОБА ВХОД-ВЫХОД?»

Эксперимент этого пункта может показаться весьма странным. В нём мы попробуем сделать «невозможное». Совместить вход и выход по одному выводу в одно целое.... Рассмотрим схему на рисунке 13. Представим, что A0 выход микросхемы. Тогда возможно два крайних значения напряжения на выводе — 5 Вольт и 0. Светодиод при таком режиме в первом случае горит, во втором нет. Резистор R3 защищает выход от короткого замыкания при логической единице и положении центрального контакта внизу (по схеме). Теперь представим, что вывод A0 это вход. Напряжение на нём определяется положением ручки потенциометра и максимальным значением напряжения, падающем на цепочке светодиод — диоды. Резистор R3 на эти значения практически не оказывает влияния. Ясно, что интервал значений больше 0, но меньше 5 Вольт. Соединим элементы, как показано на макетке, по рисунку 14. Попробуем создать программу, которая позволяет регулировать частоту миганий светодиода с помощью потенциометра. Для этого, как минимум, придётся отказаться от объявления статуса вывода в части **void setup()** . Программа дана ниже.

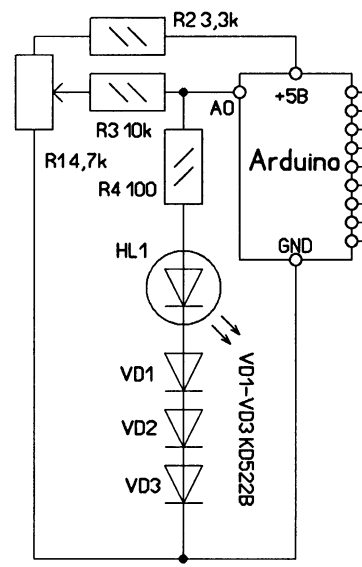

Рис. 13

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
```

```
//
//////////////////////
//
//программа ПРОБА 1 //
unsigned int  n=0;
 unsigned int  x=0;

void setup()
  {
  }
     void loop()
    {
     pinMode(A0,
INPUT);x=analogRead(A0);
n=map(x, 0, 700, 100, 2000);
     pinMode(14, OUTPUT);digital
Write(14,HIGH);delay(n);
     digitalWrite(14,LOW);delay(n);
     }

//
// Конец /
//
//////////////////////////////////////////////////////////////////
```

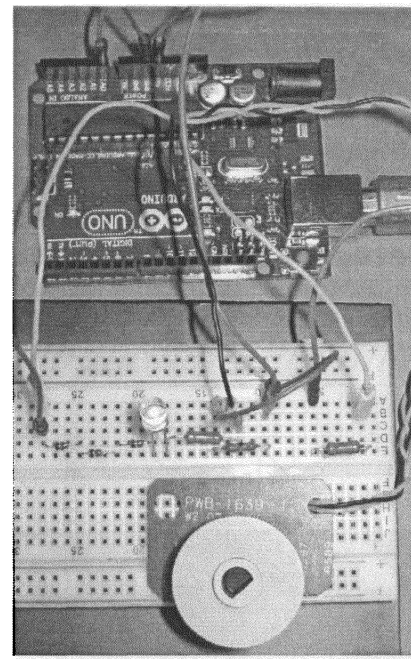

Рис. 14

Убедившись в её работоспособности, интерпретируем содержание строк части **void loop()**. Выводу A0 (он же вывод 14) присвоить статус вход. Переменной x присвоить значение, снятое со входа. Переменной n присвоить значение сообразное значению x из диапазона от 100 до 2000. Эта переменная определяет длительность свечения и гашения светодиода в миллисекундах. Теперь выводу присвоить статус выход. Включить светодиод на n миллисекунд, выключить настолько же. Вновь повторять проделанное сколь угодно число раз.

Для понимания работы скетча можно изменять значения чисел 700, 100, 2000. Число 700 можно уменьшать, не меняя номинал резистора R2. Далее действия можно повторить, увеличивая его номинал.

Эксперимент №22
«ЭМИ — ВАРИАНТ ВТОРОЙ»

В главе второй мы рассматривали электромузыкальный инструмент на два десятка нот пары октав. В этом эксперименте рассмотрим ещё варианты одноголосого (одно нотного) инструмента с меньшим числом нот. Отличие схемы в клавиатуре, «нагруженной» всего на один вывод платы (рис. 15). Резисторы R1-R7 образуют ступенчатый

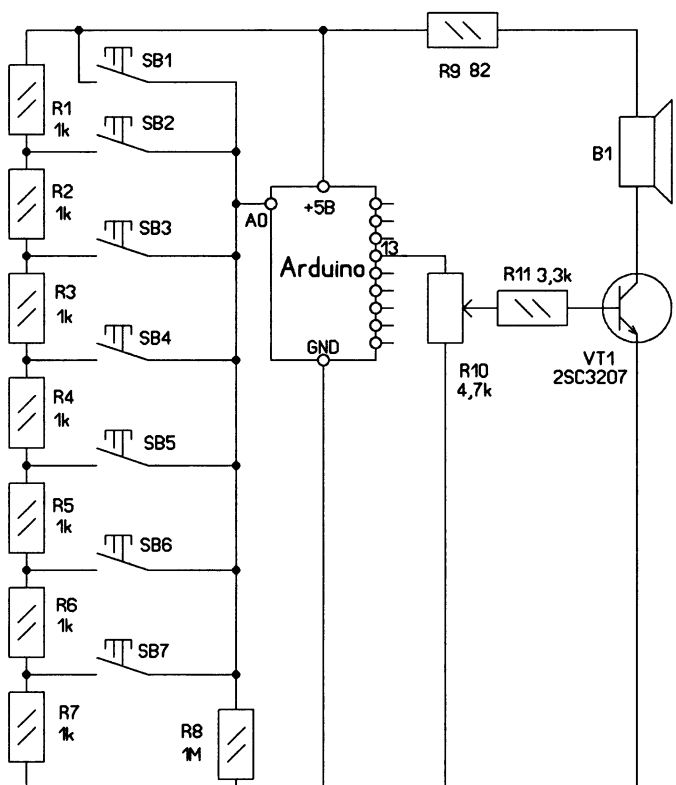

Рис. 15

делитель напряжения, поочерёдное нажатие кнопок SB1-SB7 подает это напряжение на вход A0 аналого-цифрового преобразователя. Так если делитель получает в питании 5 Вольт, при нажатии кнопки SB7 на входе величина близкая к значению 0,7 Вольт, нажатие кнопки SB2 увеличивает значение до 4,3 Вольта. Одновременное нажатие кнопок нежелательно, ибо нарушит логику работы программы. Резистор R8 (его номинал можно варьировать от 1 М до 220 кОм) нужен для ликвидации помех на входе A0 при разомкнутых контактах кнопок (он формирует цифровое значение — 0). Программа имеет вид:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
```

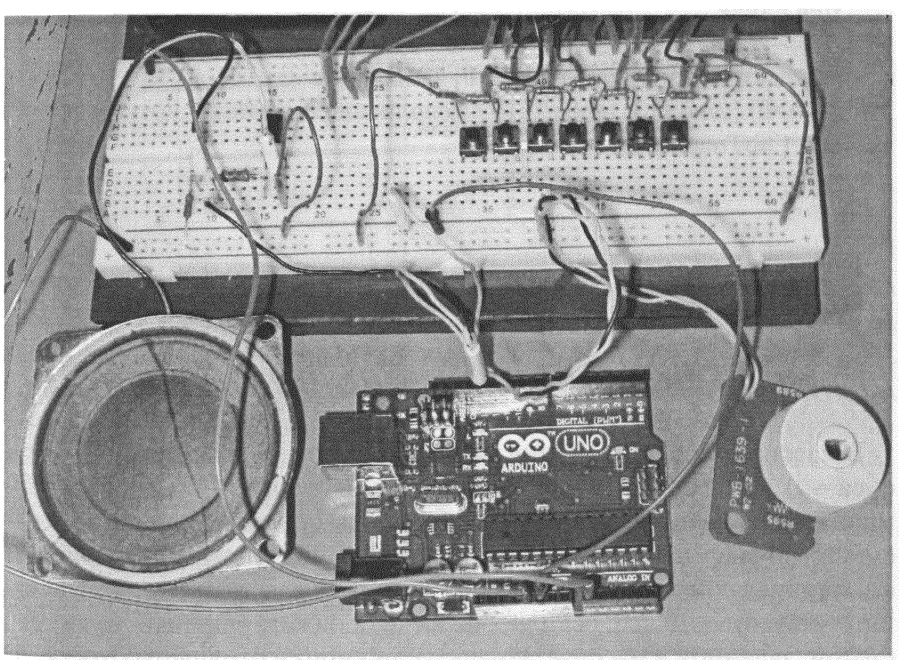

Рис. 16

```
//////////////////////
//
//программа  ЭМИ вариант 2 //
unsigned int  n=0;
 unsigned int  x=0;
int tabl [8] ={0,261,293,329,349,392,
                440,494  };  // зашифрованные данные нот — их частоты


void setup()
  {
      pinMode(13, OUTPUT);
      digitalWrite(13, LOW);
      pinMode(A0, INPUT);
  }
     void loop()
    {
     x=analogRead(A0); n=map(x, 0, 1025, 0, 8);
     if ( n>0) {tone (13, tabl[n]);} // исполнение вызванной ноты
     else { noTone(13) ;}

    }

//
// Конец /
//
//////////////////////////////////////////////////////////////////
```

В массив внесены значения частот семи нот, первый элемент массива — 0. Он необходим для того чтобы при отпущенных кнопках команда tone() не исполнялась. Логику работы основного цикла трактуем: сними значение напряжения на входе (в числовом коде), присвой его переменной x, определи номер элемента массива, соответствующий значению. Далее, если любая кнопка нажата «включи звук», частотой, заданной в массиве. Иначе (если кнопки отпущены) отключи звук. При настройке программы столкнулся с тем, что не все кнопки

срабатывали на нужные ноты. Пришлось (смотрим скетч) поменять начальные значения 1023 на 1025, а 7 на 8 (строка с «map»). Так же пришлось уменьшить номинал резистора R8. Макет изделия дан на рисунке 16.

Музицируя (балуясь) с ЭМИ заметил что совершенно не могу отмерять временные интервалы игры нот. Для временного (ударение на «о») ориентира добавил пару светодиодов с регулятором частоты миганий, подключив всё на один вывод. Слова поясняет схема рисунка 17. Регулятор громкости убран, добавлен регулятор частоты миганий светодиодов (резистор R12). Скетч:

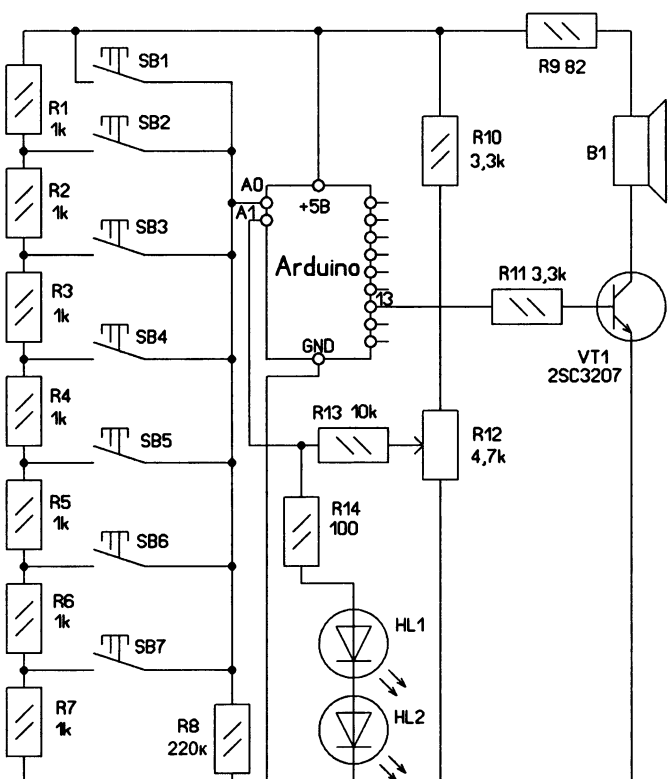

Рис. 17

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа  ЭМИ вариант  со светодиодами //
unsigned int  n=0;
 unsigned int  x=0;
unsigned int  n1=0;
 unsigned int  x1=0;
long  y1=0;
long  y2=0;
int i=0;
int tabl [8] ={0,261,293,329,349,392,
                440,494  };  // зашифрованные данные нот — их частоты


void setup()
  {
      pinMode(13, OUTPUT);
      digitalWrite(13, LOW);
      pinMode(A0, INPUT);
  }
     void loop()
    {
     x=analogRead(A0); n=map(x, 0, 1025, 0, 8);
     if ( n>0) {tone (13, tabl[n]);} // исполнение вызванной ноты
     else { noTone(13) ;}

 /////////////// ////////////////////////////// мигающие светодио-
ды ////////////////////////////////////////////// y2=millis();
pinMode(A1, INPUT);x1=analogRead(A1); n1=map(x1, 0, 700, 100,
2000);
```

```
pinMode(15, OUTPUT);if (i==1){digitalWrite(15,HIGH);}
else{digitalWrite(15,LOW);} if ( y2-y1>=n1&i==1) {y1=y2;i=0;} if (
y2-y1>=n1&i==0) {y1=y2;i=1;}
//////////////////////////////////////////////////////////////////
    }
//
// Конец /
//
//////////////////////////////////////////////////////////////////
```

Фрагмент кода, касающийся работы светодиодов, выделен. В нём полностью исключена команда delay(). Почему? Ранее мы это обсуждали.

Поработав с этим вариантом, читатель, тебе непременно захочется улучшений. Я пошёл таким путём: увеличил число играемых нот, и запретил звучать инструменту при одновременном нажатии более одной кнопки, убрал светодиоды. Схема варианта дана на рис. 18. Добавлены кнопки, резисторы и конденсатор — нужен для подавления помех на входе. Кстати интересный эффект получается, если его номинал увеличить до 0,1 мкф. Но здесь умолчим об этом.

Поясним особенность отслеживания программой нажатия двух и более кнопок. «Следит» за этим отдельный аналоговый вход A1, снимая напряжение с резистора R10. В чём его особенность? В идеале, при отпущенных кнопках (разомкнутых контактах) напряжение на нём имеет значение, близкое к 0,5 В. При нажатии одной любой кнопки значение меняется несущественно. А вот если нажать пару... например седьмую и восьмую кнопку. Мы закорачиваем выводы резистора R7. Теперь напряжение на десятом резисторе скачком увеличивается на 0,6 Вольта. При трёх нажатых кнопках ещё больше. Именно это и можно отслеживать программой. Изменённый скетч:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
```

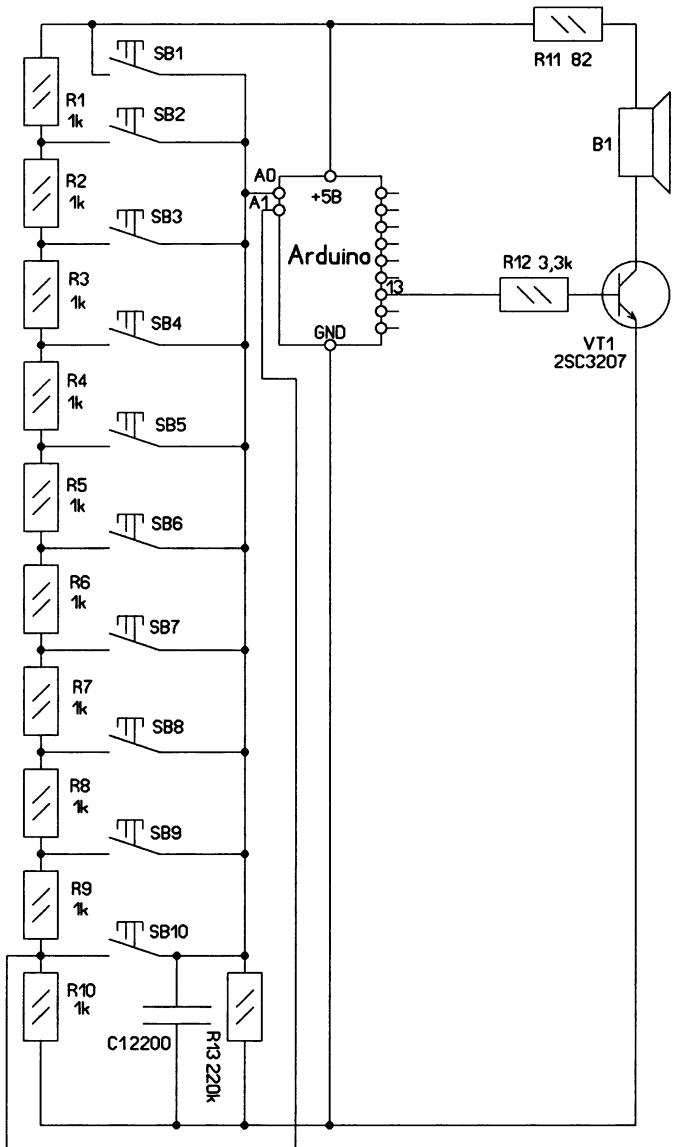
SB1
SB2
SB3
SB4
SB5
SB6
SB7
SB8
SB9
SB10
R1 1k
R2 1k
R3 1k
R4 1k
R5 1k
R6 1k
R7 1k
R8 1k
R9 1k
R10 1k
C12200
R13 220k
A0
A1
+5B
Arduino
13
GND
R11 82
B1
R12 3,3k
VT1
2SC3207

Рис. 18

```
//
//программа  ЭМИ вариант 3 с запретом нажатия более 1 кнопки //
unsigned int  n=0;
 unsigned int  x=0;
unsigned int  x1=0;
int tabl [11] ={0,261,293,329,349,392,
                 440,494,523,587,659};  // зашифрованные данные нот —
их частоты


void setup()
  {
      pinMode(13, OUTPUT);
      digitalWrite(13, LOW);
      pinMode(A0, INPUT); //вход, следящий какая кнопка нажата
      pinMode(A1, INPUT); //вход, следящий сколько кнопок нажато

  }
     void loop()
    {
     metka:
     x1=analogRead(A1);
     if ( x1>=106) {goto metka;}// блокировка при нажатии более 1
кнопки
     x=analogRead(A0); n=map(x, 0, 1025, 0, 11);
     if ( n>0) {tone (13, tabl[n],10);} // исполнение вызванной ноты
     else { noTone(13);}
     }

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Не смотря на то что конструкция и программа рабочие, автора терзают сомнения. Мне кажется что не хватает развязывающего диода (диодов). Но продолжим.

В фрагменте: **metka:**

x1=analogRead(A1);

if (x1>=106) {goto metka;}// блокировка при нажатии более 1 кнопки

появляется новая команда «goto metka;». Это управляющий оператор, совершающий «перемещение» выполнения программы к метке-указателю. В нашем примере происходит возврат в начало void loop() при условии x1>=106. В общем случае, пропускается весь код до метки и выполняется после. То есть, если напряжение на входе A1 будет больше 0,56 Вольт программа зациклится внутри фрагмента, до отпускания «лишнёй» кнопки или кнопок. Напомню, что число 106 — экспериментальное (теоритическое вычисление даёт 102-103).

В связи с этим иногда при отладке простых программ удобно пользоваться монитором порта. Более подробно о его работе написано в книге Блума. Мы рассмотрим эту «полезность» на примере ещё одного варианта скетча для ЭМИ:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа  ЭМИ вариант 3 с запретом нажатия более 1 кнопки и
выводом данных на монитор порта //
unsigned int  n=0;
 unsigned int  x=0;
unsigned int  x1=0;
long y1 = 0;
long y2 = 0;

int tabl [11] ={0,261,293,329,349,392,
                440,494,523,587,659};  // зашифрованные данные нот —
их частоты
```

```
void setup()
  {
       pinMode(13, OUTPUT);
       digitalWrite(13, LOW);
       pinMode(A0, INPUT);
       pinMode(A1, INPUT);
       Serial.begin(9600);
  }
      void loop()
     {
      metka:
       y2=millis();
       if (y2-y1>=2000)
       {y1=y2; Serial.println("x,x1,n");Serial.println(x);Serial.
println(x1);Serial.println(n);}

      x1=analogRead(A1);
      if ( x1>=106) {goto metka;}// блокировка при нажатии более 1
кнопки
      x=analogRead(A0); n=map(x, 0, 1025, 0, 11);
      if ( n>0) {tone (13, tabl[n],10);} // исполнение вызванной ноты
      else { noTone(13);}


     }

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Фрагмент программы:

```
 y2=millis();
  if (y2-y1>=2000)
```

{y1=y2; Serial.println(“x,x1,n”);Serial.println(x);Serial.println(x1);Serial.println(n);}

можно «перевести» — каждый раз, через две секунды печатай в мониторе порта надпись «x,x1,n» и числовые значения переменных x, x1, n.

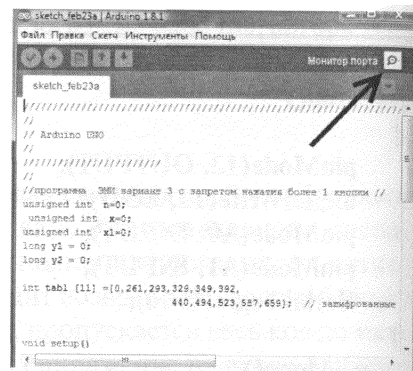

Рис. 19

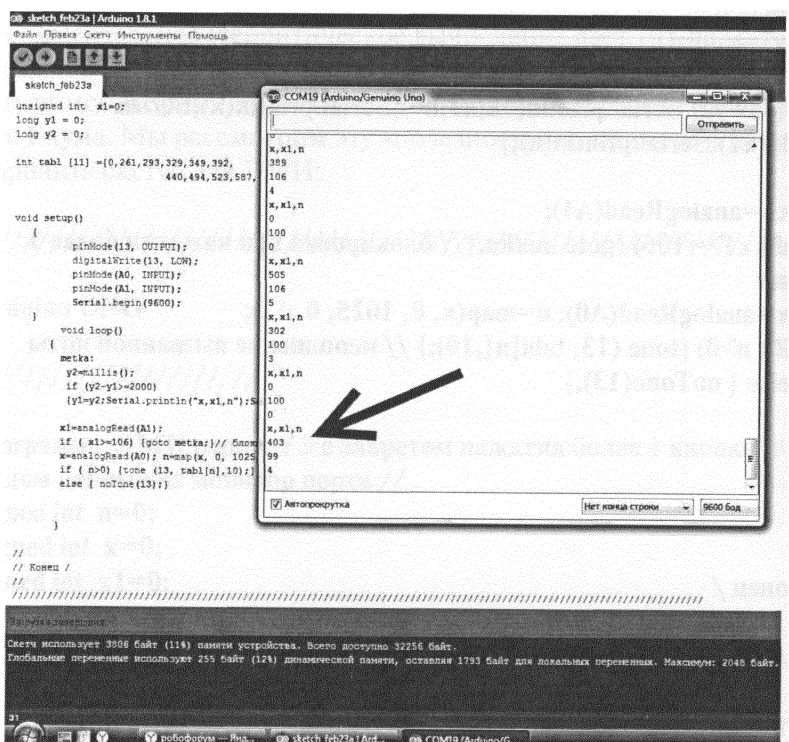

Рис. 20

Теперь, музицируя, можно отслеживать какие значения принимают переменные при различных комбинациях нажатых кнопок.

Для вызова монитора пользуемся «кнопкой Монитор порта» — поясняет рисунок 19. Внешний вид информации на мониторе, при работе скетча даёт рисунок 20. Столбец-строки обновляются снизу вверх.

Функция Serial.begin(9600);,прописанная в части void setup() инициирует последовательный интерфейс обмена данными между платой и компьютером. Число 9600 определяет скорость обмена информацией в единицах — бод. После загрузки скетча, во время его работы можно заметить мигание светодиода (Tx) на плате каждые две секунды. Этот индикатор показывает, что Ардуино передаёт данные на компьютер. Передача идёт через последовательный USB-интерфейс.

Эксперимент №23 «ПРОБА — МОТОР»

Поэкспериментируем с мотором. Соберём схему по рисунку 21. Внешний вид макета и его реализация даны на рисунках 22 и 23 соответственно. В качестве испытуемого применим двигатель от DVD-

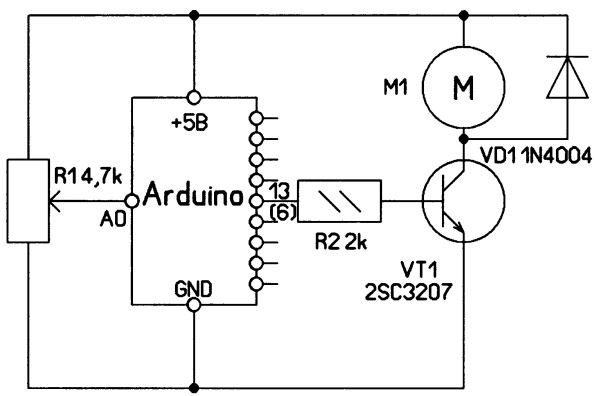

Рис. 21

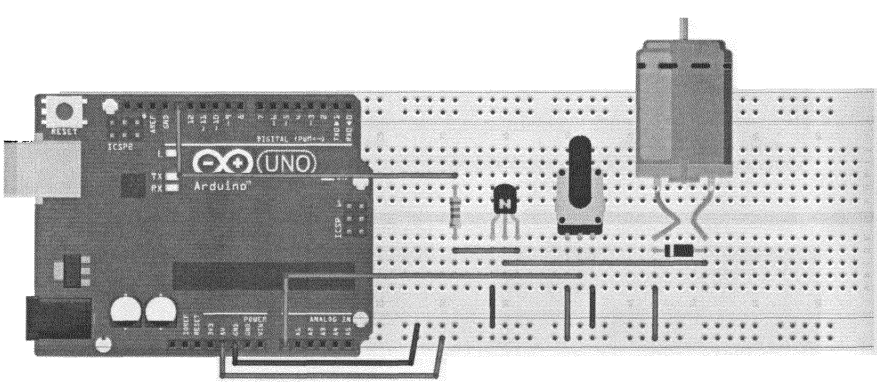

Рис. 22

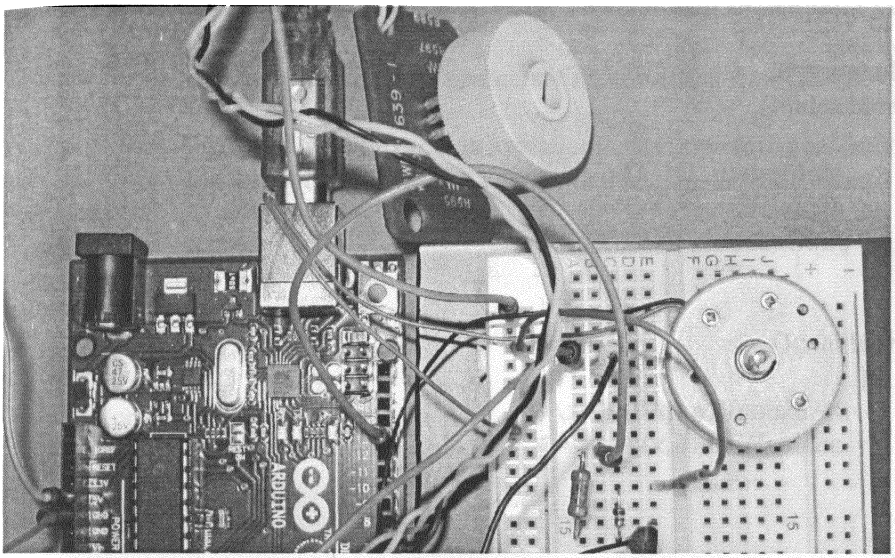

Рис. 23

привода. Потребляемый им ток под нагрузкой, обычно не превышает 100-120 мА. Это обстоятельство позволяет нам использовать для его питания соответствующие выводы платы. Транзисторный ключ позволит нам регулировать «усреднённое значение напряжения» на моторе методом ШИМ. Диод защищает схему от нежелательных импульсов напряжения обратной полярности, возникающих при коммутации двигателя. К аналоговому входу A0 подключим потенциометр для регулировки скорости вращения вала или круговой частоты. В процессе проб будем менять номера выводов платы на мотор. Итак первый скетч:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
```

```
// Программа регулировки частоты вращения вала мотора /
//
long x = 0;
void setup()
{
     pinMode(6, OUTPUT);
     digitalWrite(6, LOW);
    pinMode(A0, INPUT);
 }

void loop()
  {
    x=analogRead(A0);x=x/4;
     analogWrite(6,x);

 }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Всё предельно просто. В основном цикле переменная x принимает числовое значение, пропорциональное напряжению на центральном выводе потенциометра (от 0 до 1023). Затем это значение уменьшается в 4 раза. И наконец, на вывод №6 подаётся ШИМ волна, скважность импульсов которой регулируется текущим значением x. Вращаем ручку резистора и убеждаемся в работоспособности скетча.

Но может получиться так, что вывод. Поддерживающий ШИМ занят. Как регулировать скорость вращения вала на «цифровом выходе»? Ведь команда, например, «analogWrite(13,x);» уже не сработает. Переключим транзистор на вывод №13. Рассмотрим работу программы:

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
```

```
//////////////////////
//
// Программа регулировки частоты вращения вала мотора /
//
long y1 = 0; // вводим переменную, для хранения считываемого времени
long y2 = 0; // вводим переменную для хранения предыдущего значения
времени
long x = 0;
void setup()
{
    pinMode(13, OUTPUT);
    digitalWrite(13, LOW);
   pinMode(A0, INPUT);
}

void loop()
  {
     x=analogRead(A0);
      y2= micros(); //присваиваем переменной текущее время
      if (y2-y1 >=x) { digitalWrite(13,HIGH); }//определяет длитель-
ность низкого уровня сигнала
      if (y2-y1 >=1023) { digitalWrite(13,LOW);y1=y2; }// запускает
новый период отсчёта
 }
//
// Конец /
//
//////////////////////////////////////////////////////////////////////
```

В ней появляется «новое слово» — **micros()**. Её работа (функции) очень похожа на работу функции «**millis()**». Она возвращает количество микросекунд с момента запуска текущей программы на плате Arduino. Это количество нарастает при непрерывной работе и сбрасывается в ноль, вследствие переполнения значения, приблизительно через 70 минут. Конечно, максимальное число очень большое и имеет тип unsigned long. Возможность отсчитывать не миллисекунды,

а микросекунды существенно расширяет возможности творческого программирования.

Итак, 1023 микросекунд определяет суммарную длительность высокого и низкого уровня сигнала на выходе 13 за один цикл изменения этих уровней. А «снятое с потенциометра число x» есть число микросекунд длительности присутствия на выходе низкого уровня сигнала. Вращая ротор резистора, мы меняем соотношения длительностей присутствия логических уровней на выходе, то есть добиваемся управляемой ШИ модуляции на частоте около 977 Гц. Действительно, период «колебания» логических уровней (в микросекундах) можно определить по формуле: d=1000000/n, где n частота, а d длительность «колебания». В итоге мы плавно регулируем «усреднённую силу тока», протекающую через обмотку мотора.

Эксперимент №24
«ПРОБА — НОТЫ БЕЗ TONE»

Недостаток этой функции в том, что нельзя одновременно, например, на разные динамики «исполнять» разные ноты. Как избавиться от этого неудобства? Воспользуемся результатами предыдущего эксперимента, привлекая в программы «новые слова». Схема эксперимента на рисунке 24, вариант макета на рис. 25. Ниже дана серия скетчей:

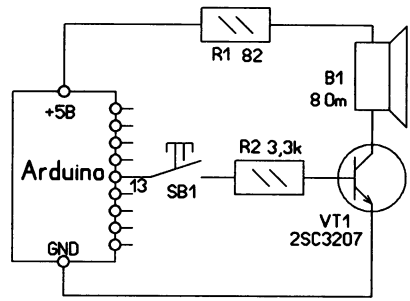

Рис. 24

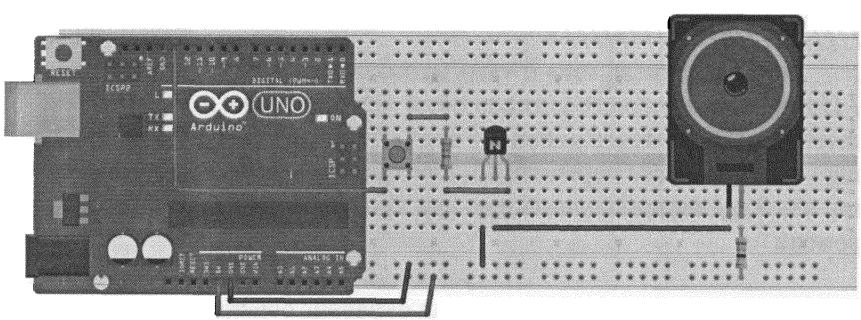

Рис. 25

```
//LED Blink — вспышки звука в динамике ////////////////////////
////////////////////////////////
int ledPin = 13; //номер вывода Arduino к которому подключен теперь
динамик

void setup()
 {
```

```
 pinMode(ledPin, OUTPUT);  // определение вывода как ВЫХОД
}

void loop()
{
 digitalWrite(ledPin, HIGH); //подать высокий уровень сигнала на
динамик
 delayMicroseconds (760);// задержка 760 мксек
 digitalWrite(ledPin, LOW); //выключить динамик
 delayMicroseconds (760);//ждать 760 микросек
}
//////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа звукового сигнала ноты «ми» 2 октавы /
//
long i = 0; // вводим переменную, для хранения считываемого времени
long n = 0; // вводим переменную для хранения предыдущего значения
времени

void setup()
{
    pinMode(13, OUTPUT);
    digitalWrite(13, LOW);
 }

void loop()
  {
      i= micros(); //присваиваем переменной текущее время
         if (i — n >=760)  // если разница предыдущего значения и теку-
щего...
```

```
          {
           digitalWrite(13,! digitalRead (13));   //инвертируем состояние
вывода
            n=i;  // заменяем предыдущее значение текущим

           }
}
//
// Конец /
//
////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа звуковой сирены (ноты «ми» и «до» разных октав)/
//
long i1 = 0; // вводим переменную, для хранения считываемого времени

long n1 = 0; // вводим переменные для хранения предыдущего значения
времени
long n2 = 0;
long k = 760;//начальное значение длительности уровня сигнала ноты
«ми»

void setup()
{
    pinMode(13, OUTPUT);
    digitalWrite(13, LOW);
 }

void loop()
  {
```

```
      i1= micros(); //присваиваем переменной текущее время

      if (i1-n2>=1000000&k==760){k=1916;n2=i1;}// условия смены
частоты звука
      if (i1-n2>=1000000&k==1916){k=760;n2=i1;}//через каждую
секунду


     if(i1-n1>=k)//если разница предыдущего значения и текущего...
      { digitalWrite(13,! digitalRead (13)); n1=i1; }//инвертируем состоя-
ние вывода
     // заменяем предыдущее значение текущим


   }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Рассмотрим их работу. В первом скетче новая команда delayMicroseconds(). Она останавливает выполнение программы на заданное в параметре количество микросекунд (1 000 000 микросекунд в 1 секунде).

Для остановки выполнения программы более чем на несколько тысяч микросекунд рекомендуется использовать функцию delay(). Фактически обе эти функции работают одинаково, отличаясь лишь смыслом значений аргументов. Аналогичные рекомендации относятся и к функциям micros(), millis().

Частота переключений (длительность задержки в примере 760 микросекунд) определяется выражением n=500000/d (d=760). Нажимая и отпуская кнопку, на слух, можно определить соответствие ноты «ми».

Второй скетч работает совершенно аналогично, но не содержит команду delayMicroseconds(). Нижний скетч превращает схему в сирену «на две ноты». Звучание нот «ми» и «до» разных октав сменяют друг друга каждую секунду.

Эксперимент №25
«ПРОБА — GOTO»

В заключение изложения главы рассмотрим пример скетча, позволяющего «независимо проигрывать» три отдельных фрагмента программ. Такая структурная схема может быть полезна, например, при разработке мобильного робота, с «жестким, последовательным сценарием поведения». Скетч проверим по схеме рис. 26, переключая звуковые частоты через заданный промежуток времени (в данном случае это 2 секунды). Цветом в программе выделены части кода, вместо которых можно вставлять свои варианты управления исполняющими устройствами.

```
//////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа проба команды goto /
//
long y1 = 0;
long y2 = 0;

void setup()
{
    pinMode(13, OUTPUT);
    digitalWrite(13, LOW);

}

void loop()
  {
```

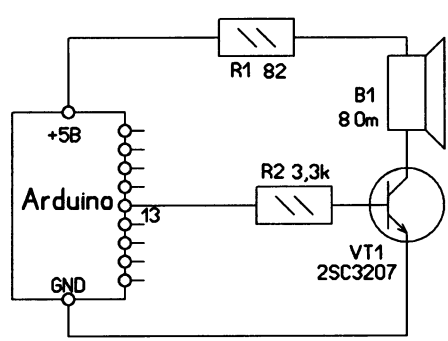

Рис. 26

```
    metka0: // 1 фрагмент программы
    y2=millis();
    if (y2-y1<=2000)
   {tone(13,200); //фрагмент программы// goto metka0;}
    else {y1=y2;goto metka1;}

    metka1: // 2 фрагмент программы
    y2=millis();
    if (y2-y1<=2000)
   {tone(13,400); // фрагмент программы// goto metka1;}
    else {y1=y2;goto metka2;}

    metka2: // 3 фрагмент программы
    y2=millis();
    if (y2-y1<=2000)
   {tone(13,600);// фрагмент программы//  goto metka2;}
    else {y1=y2; goto metka0;}
  }
//
// Конец /
//
////////////////////////////////////////////////////////////////
```

Строка «if (y2-y1<=2000)» определяет длительность работы каждого фрагмента. Увеличивая или уменьшая число 2000, мы меняем время исполнения данной части скетча. Строка «else {y1=y2;goto metka1;}» осуществляет переход к новому фрагменту исполнения общей программы.

Ну вот, читатель, пройдя успешно три главы, теперь можем приступить к реализации первого проекта. Знаний для этого вполне достаточно. Пусть это будет робот по образу и подобию БЕАМ роботов.

ГЛАВА 4
«АРДУИНОБИМ»

Эту главу начнём с описания конструкции мобильного робота, в качестве исполняющих элементов, имеющего пару двигателей постоянного тока, зуммер и два светодиода. Для упрощения реализации исключим из неё любые датчики, редукторы. Основные элементы представлены на рисунке 1. Это батарейный отсек (1) на четыре элемента питания типа AAA (солевые батарейки), плата Arduino UNO (2), «односторонняя под пайку» макетная плата (3), скоба с закреплёнными моторами (4), пластмассовое основание (5) и три крепёжных винта с гайками (6).

Снизу на боковую поверхность батарейного контейнера приклеено пластмассовое основание. В нём просверлено отверстие для крепления к общему шасси робота. Плата (3) имеет три крепёжных отвер-

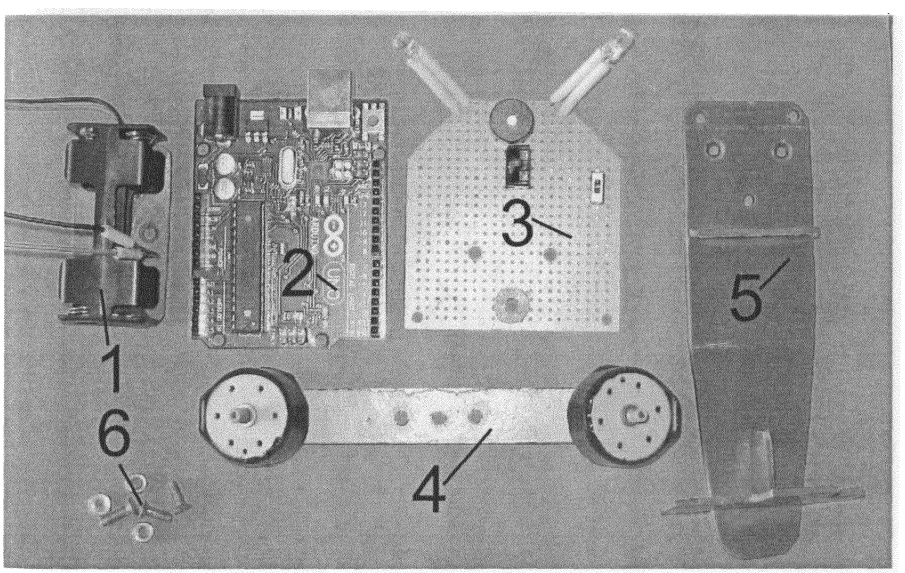

Рис. 1

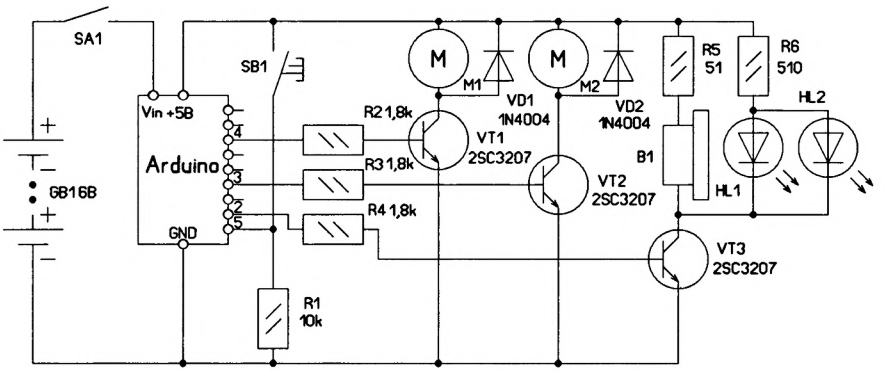

Рис. 2

стия (образуют треугольник) и пару резервных. На неё монтируют кроме платы и контейнера все остальные элементы схемы (рис. 2). Моторы закреплены на стальной П-образной планке посредством отрезков изоляционной ленты. Для более прочного крепления можно использовать пластиковые стяжки для проводов, подходящей длины. На валы моторов посажены отрезки резиновой изоляции (на-

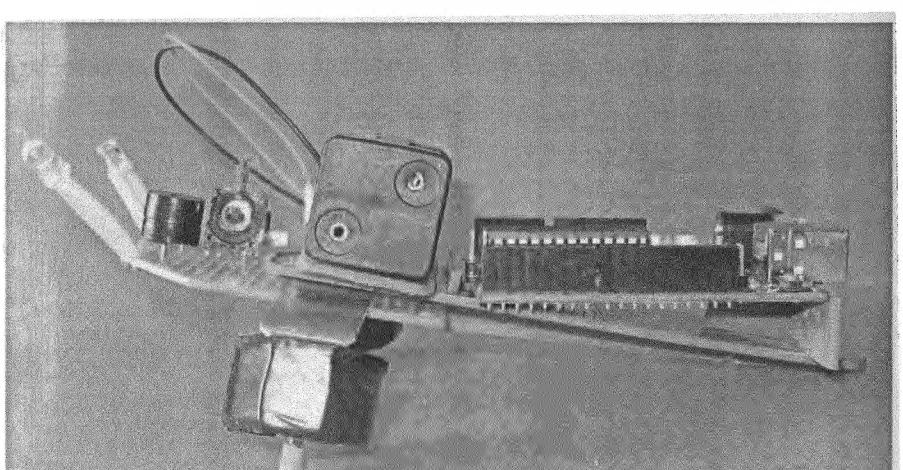

Рис. 3

пример, изоляция шнура компьютерной мыши). Чтобы при длительной эксплуатации ходовой части «колёсики» не съезжали на корпус моторов, предварительно на валы можно сажать отрезки термоусадочной трубки, а уже сверху «колёсики». Пластмассовое основание (5) состоит из 4 элементов, соединённых между собой при помощи «секундного клея». Элементы нужны для создания съёмного крепления платы.

Сборку совершают послойно — сажая на стержни винтов макетку, скобу, основание, контейнер. Конструктивный концепт робота дан на рис. 3. Вид сверху на рисунке 4. Внешний вид готовой конструкции изображён на рисунке 5. Дополнительно плата (около разъёмов) крепится к основанию канцелярской резинкой в два оборота.

Для работы робота (рис. 2) задействовано 4 «пин — вывода» платы. Выводы 3 и 4 для управления моторами, вывод 2 для управления «звуком и светом» и вывод №5 задействован для обеспечения функционирования «пусковой кнопки робота». Питание подаётся на вывод

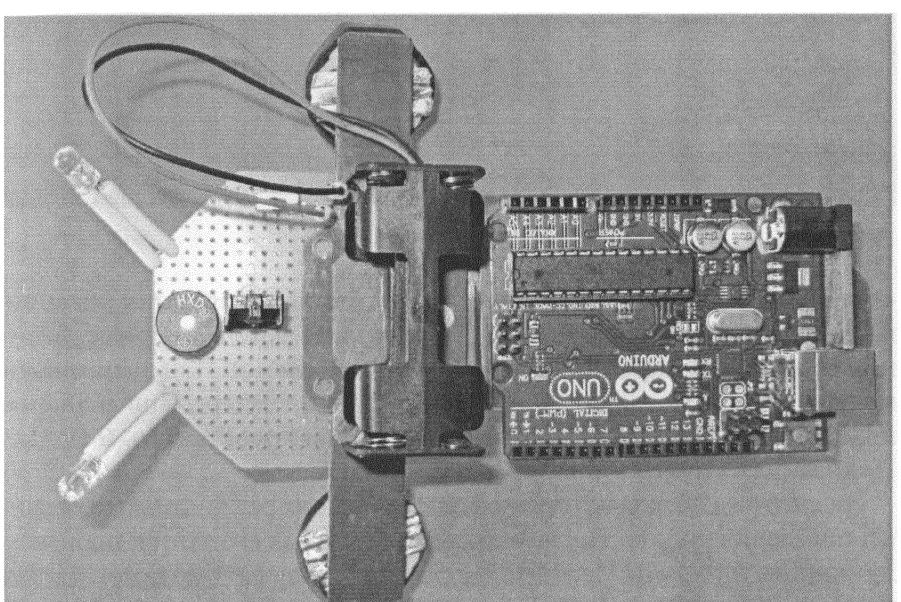

Рис. 4

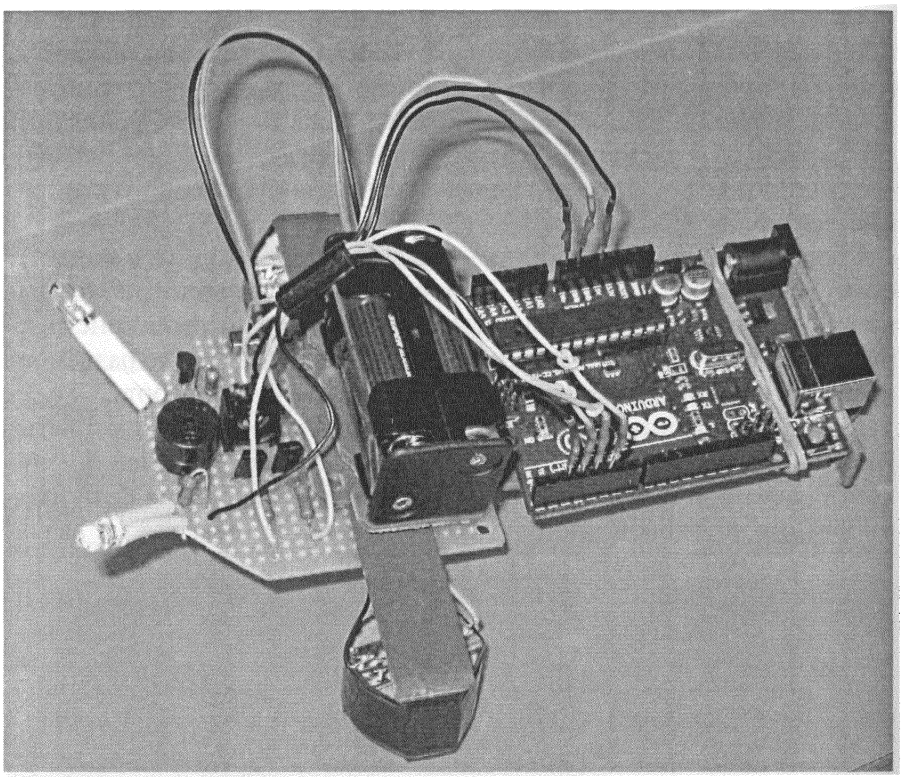

Рис. 5

«Vin». Вообще-то он предназначен для снятия напряжения питания на внешние исполняющие устройства с платы Ардуино при её питании от внешнего источника (рис. 6 –верхняя схема). При этом минимально допустимое напряжение может быть 6,5 Вольт. При «нашем» подключении минимальное напряжение может составлять 6 Вольт. Полная схема платы приведена на рисунке 7.

Рассмотрим несколько скетчей, управляющих различными сюжетами поведения робота. После включения питания его запуск производит кнопка SB1. ВНИМАНИЕ! Не следует одновременно подавать напряжение на плату через USB разъём и от батареек, дабы исключить её поломку (при подключении платы к компьютеру SA1 — выключен).

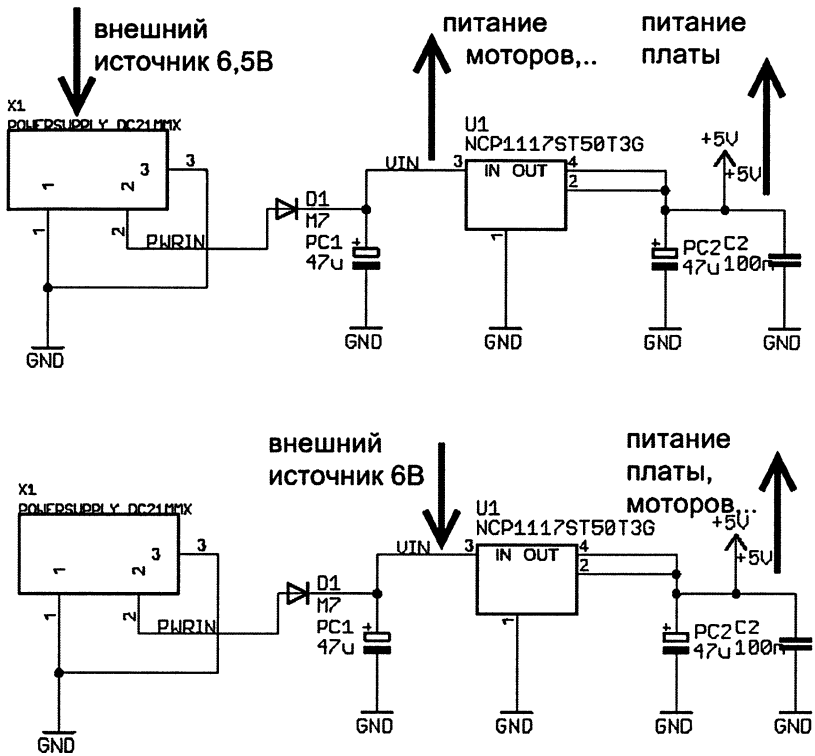

Рис. 6

Сюжет первый: после пуска робот сердито «бурчит на своём языке», гордо разворачивается, и «отходит» от «обидчика», после становится в дежурный режим, до нового беспокойства. Программа:

```
//////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа №1 для управления роботом /
```

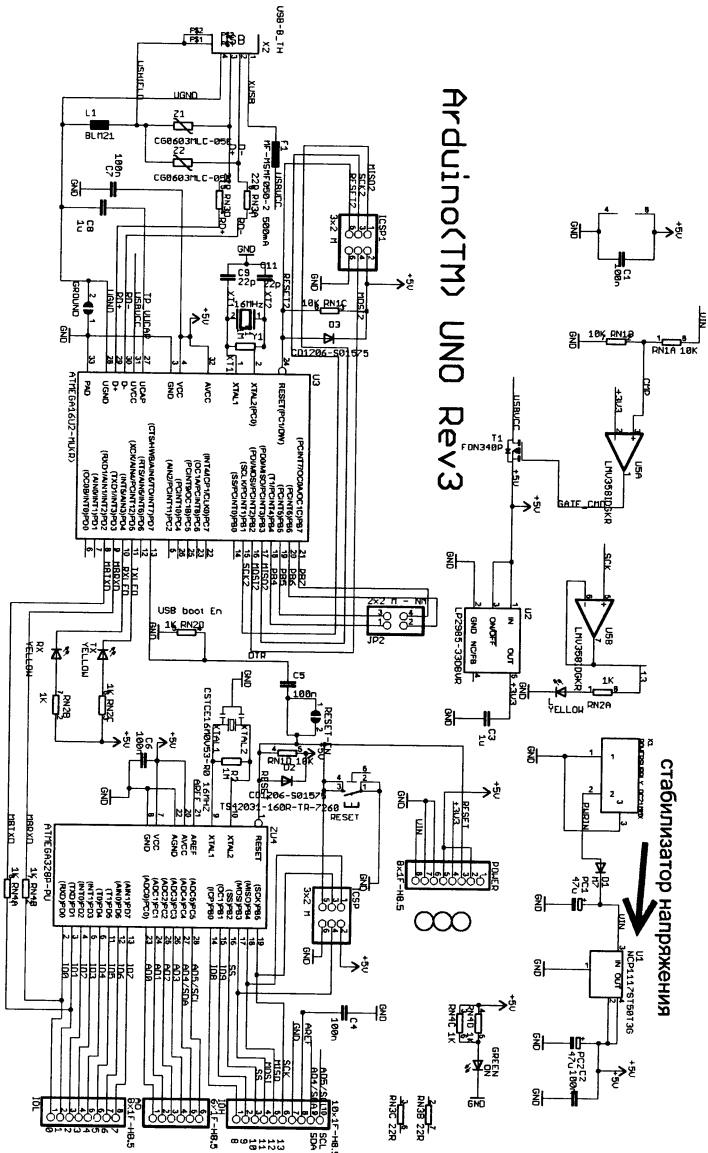

Рис. 7

```
//
long y1 = 0;
long y2 = 0;
long y3 = 0;
int x;
int x1;
long y4 = 0;
void setup()
{
    pinMode(2, OUTPUT);//канал передачи звуковых и световых
эффектов
    digitalWrite(2, LOW);
    pinMode(3, OUTPUT);//канал управления левым мотором
    digitalWrite(3, LOW);
    pinMode(4, OUTPUT);// канал управления правым мотором
    digitalWrite(4, LOW);
    pinMode(5, INPUT);// кнопка управления пуском
}

void loop()
  {
    metka:
    if (digitalRead (5)==LOW){y2=millis();y1=y2;goto metka;}  //
условие блокировки запуска работы скетча

    metka0: // 1 фрагмент программы — звуки
    y2=millis();
    if (y2-y1<=4000)
   { y4=millis();
   if (y4-y3 >=250){tone(2, 2500+500*x,100*x1);y3=y4;x=random(1,5);
x1=random(1,3);}goto metka0;}
    else {y1=y2;goto metka1;}

    metka1: // 2 фрагмент программы — поворот
    y2=millis();
    if (y2-y1<=2000)
```

```
    {
    digitalWrite(3, HIGH);delay(800);digitalWrite(3, LOW);delay(1500);

    goto metka1;}
    else {y1=y2;goto metka2;}

    metka2: // 3 фрагмент программы — хождение
    y2=millis();
    if (y2-y1<=6300)
    {digitalWrite(4, HIGH);delay(270);
    digitalWrite(3, HIGH);digitalWrite(4, LOW);delay(170);
    digitalWrite(3, LOW);delay(500);goto metka2;}
    else {y1=y2; goto metka;}
  }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Основа скетча — пример из предыдущей главы на применение команды goto. Строки — **metka:**

if (digitalRead (5)==LOW){y2=millis();y1=y2;goto metka;}

проверяют, нажата ли кнопка, если нет, то зацикливают программу на себе и присваивают переменным y1, y2 значение текущего времени после запуска. После нажатия последовательно запускается исполнение 3 фрагментов. Первый фрагмент отвечает за «речь робота». Чтобы она была «разнообразной» используется команда random(). Функция random() возвращает псевдослучайное число. Случайное число между min и max-1. (long). В данной программе для **x=random(1,5);x1=random(1,3)** переменная x может быть равна 1, 2, 3, 4, переменная x1 — 1 или 2. Одна из них определяет значение частоты звука, другая его длительность. Условие **if (y2-y1<=4000)** определяет длительность исполнения данного фрагмента (4 секунды).

Второй фрагмент «включает» левый двигатель на 0,8 секунд, отключает, делает небольшую временную паузу. Так происходит «гордое отворачивание робота».

Третий фрагмент «управляет» ходьбой робота в течение времени около 6 секунд — левым, правым мотором по очереди. Из–за разброса характеристик моторов и управляющих транзисторов равную длину шага приходится осуществлять, корректируя время включения того или иного двигателя (в авторском варианте конструкции правый мотор «слабее» левого). После исполнения всех фрагментов робот останавливается, до нового нажатия кнопки.

Сюжет второй: робот имитирует звуки сирены, затем вращается на месте в одну сторону, останавливается. Даёт сирену другой тональности, вновь вращается на месте в другую сторону и наконец, остановившись, завершает своё выступление, сиреной «третьей тональности». Этакая скорая помощь, сразу и на месте...

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа №2 для управления роботом /
//
long y1 = 0;
long y2 = 0;

void setup()
{
    pinMode(2, OUTPUT);//канал передачи звуковых и световых
эффектов
    digitalWrite(2, LOW);
    pinMode(3, OUTPUT);//канал управления левым мотором
    digitalWrite(3, LOW);
    pinMode(4, OUTPUT);// канал управления правым мотором
    digitalWrite(4, LOW);
    pinMode(5, INPUT);// кнопка управления пуском
}
```

```
void loop()
  {
    metka:
    if (digitalRead (5)==LOW){y2=millis();y1=y2;goto metka;}  //
условие блокировки запуска работы скетча

    metka0: // 1 фрагмент программы — сирена1
    y2=millis();
    if (y2-y1<=3000)
    {tone(2, 600,200);delay(250); tone(2, 1000,200);delay(250);goto
metka0;}
    else {y1=y2;goto metka1;}

    metka1: // 2 фрагмент программы — вращение по часовой стрелке
    y2=millis();
    if (y2-y1<=4200)
    {
    digitalWrite(3, HIGH);

    goto metka1;}
    else {digitalWrite(3, LOW);y1=y2;goto metka2;}

    metka2: // 3 фрагмент программы — сирена2

    y2=millis();
    if (y2-y1<=3000)
   {tone(2, 900,200);delay(250); tone(2, 1300,200);delay(250);goto
metka2;}
    else {y1=y2; goto metka3;}

    metka3: // 4 фрагмент программы — вращение против часовой
стрелки
    y2=millis();
    if (y2-y1<=6000)
    {
    digitalWrite(4, HIGH);
```

```
    goto metka3;}
    else {digitalWrite(4, LOW);y1=y2;goto metka4;}

    metka4: // 5 фрагмент программы — сирена3

    y2=millis();
    if (y2-y1<=3000)
   {tone(2, 500,200);delay(250); tone(2, 700,200);delay(250);goto
metka4;}
    else {y1=y2; goto metka;}
  }
//
// Конец /
//
//////////////////////////////////////////////////////////////////
```

Структура программы схожа со структурой предыдущего скетча. Разница в количестве фрагментов-эпизодов. Здесь их пять. Временные интервалы реализации составляют 3 сек., 4,2 сек., 3 сек., 6сек., 3 сек. соответственно. После реализации сюжета робот «утихает» до повторного запуска программы.

Сюжет третий: поиски «спасителя» и взывание к помощи. Робот метается в разные стороны и зовёт на помощь, подавая сигнал SOS. Скетч работы дан ниже:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа №3 для управления роботом /
//
long y1 = 0;
long y2 = 0;
```

```
long y3 = 0;
int x;
int x1;
long y4 = 0;
unsigned int n = 0;
unsigned int k = 0;
unsigned int j = 0;
unsigned int r = 0;
unsigned int n1 = 0;

int tabl [18] ={1,1,1,1,1,3,3,1,3,1,3,3,1,1,1,1,1,7}; // зашифрованный
сигнал sos


void setup()
{
    pinMode(2, OUTPUT);//канал передачи звуковых и световых
эффектов
    digitalWrite(2, LOW);
    pinMode(3, OUTPUT);//канал управления левым мотором
    digitalWrite(3, LOW);
    pinMode(4, OUTPUT);// канал управления правым мотором
    digitalWrite(4, LOW);
    pinMode(5, INPUT);// кнопка управления пуском
}

void loop()
  {
    metka:
    if (digitalRead (5)==LOW){y2=millis();y1=y2;goto metka;n1=0;}  //
условие блокировки запуска работы скетча

    metka0: // 1 фрагмент программы — сигнал sos
    y2=millis();
    if (y2-y1<=4000)
   { for (n = 0; n <18; n=n+2)
```

```
    {
     k=tabl[n];
     j=tabl[n+1];
     tone (2, 700, 100*k) ;
     r=100*(k+j);
     delay(r);
     }

    goto metka0;}
     else {y1=y2;goto metka1;}

     metka1: // 2 фрагмент программы — поворот
     y2=millis();
     if (y2-y1<=2000)
     {
     digitalWrite(3, HIGH);delay(600);digitalWrite(3, LOW);delay(1500);

     goto metka1;}
     else {y1=y2;goto metka2;}

     metka2: // 3 фрагмент программы — прямохождение
     y2=millis();
     if (y2-y1<=800)
     {digitalWrite(4, HIGH);
     digitalWrite(3, HIGH);
     goto metka2;}
     else {digitalWrite(3, LOW);digitalWrite(4, LOW);y1=y2;while(n1<3)
{n1=n1+1; goto metka0;}goto metka;}
   }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Программа состоит из трёх частей — подача сигнала SOS в течении 4 секунд, разворота в течении 0,6 секунд с паузой-остановкой, и ры-

вок вперёд (оба мотора работают одновременно 0,8 секунды), затем повтор действий. Количество повторов программы регулирует переменная n1. Её функция скрыта в последней командной строке: «**else {digitalWrite(3, LOW);digitalWrite(4, LOW);y1=y2;while(n1<3) {n1=n1+1; goto metka0;}goto metka;}**». Каждый программный проход значение n1 возрастает на единицу и достигнув трёх, сбрасывается в ноль. Строка реализации : «**if (digitalRead(5)==LOW) {y2=millis();y1=y2;goto metka;n1=0;}**» . Таким образом робот совершает три подачи сигнала бедствия. Изменяя количество повторов можно увеличивать время реализации сюжета. Управляющий оператор –«**while()**» — задаёт цикл повторов, количество которых ограничено условием в скобках. Применительно к нашему примеру часть строки — «**while(n1<3){n1=n1+1; goto metka0;}**» можно трактовать так: пока n1 будет меньше трёх, увеличивай её значение на единицу и переходи к исполнению скетча в месте расположения metka0: (повтор сигнала SOS).

Количество возможных игровых сюжетов ограничивается только фантазией конструктора и его ленью. Возможные составляющие действий робота сведены в функциональную таблицу, количество фрагментов скетча может быть и больше 3-5. А количество повторов программы больше трёх.

Действия робота	**Реализация действия (возможная)**	**Время реализации (примерное), сек.**
Движение-рывок вперёд	Включены одновременно оба мотора	0,5-2
Шагание вперёд (шажок)	Попеременно включаются и выключаются оба мотора приблизительно на одинаковое время	0,3-07
Разворот на месте (вращение)	Включён один мотор	0,3-3

Действия робота	Реализация действия (возможная)	Время реализации (примерное), сек.
Плавное движение по дуге (требуется использование ШИМ управления выходом)	Включены оба мотора, но с разным «усреднённым» значением напряжения питания	1-6
«Несвязная речь»	Команда tone(), генератор случайных чисел	3-8
Разговор на языке азбуки Морзе	Массивы, циклы, команда tone()	4-12
Звуки сирены	Циклы, команда tone()	3-10
Исполнение музыкальных фрагментов	Массивы, циклы, команда tone()	5-20

Одним из «неудобств» конструкции является «включатель программ» SB1. Рациональнее управлять роботом на дистанции, не применяя в простом случае никаких приспособлений для этого. Как это сделать? Например, используя звук. Хлопок в ладоши или щелчок пальцами и робот начинает движение по программе. А после завершения ждёт новой команды...

Рассмотрим, как это реализуется на нескольких вспомогательных конструкциях. Соберём звуковой датчик на электретном микрофоне (рис. 8), присоединим его к плате и попробуем включать и выключать светодиод HL2, щелкая пальцами. Плата самодельного готового датчика дана на рисунке 9. Резистор R4 (на плате подстроечный, на макете удобнее пользоваться переменным) позволяет регулировать чувствительность датчика к звукам. Светодиод HL1 помогает визуально контролировать работу датчика. Правильно отстроенный режим схемы даёт возможность управлять светодиодом с расстояния в 7-10 метров. В момент щелчка — HL1 ярко вспыхивает, а светодиод HL2 меняет своё состояние на противоположное состояние (загорается или

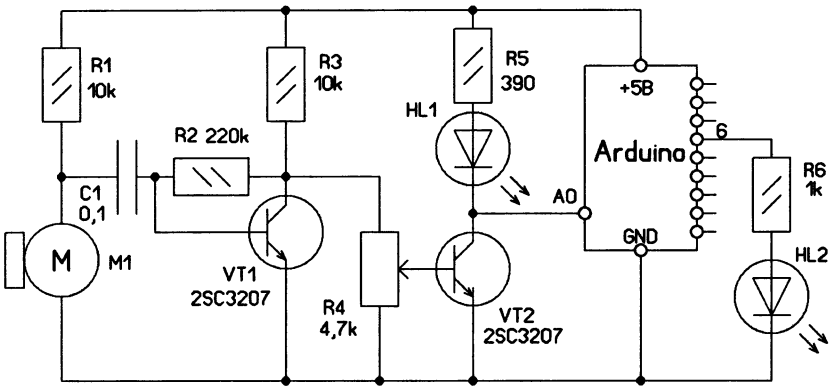

Рис. 8

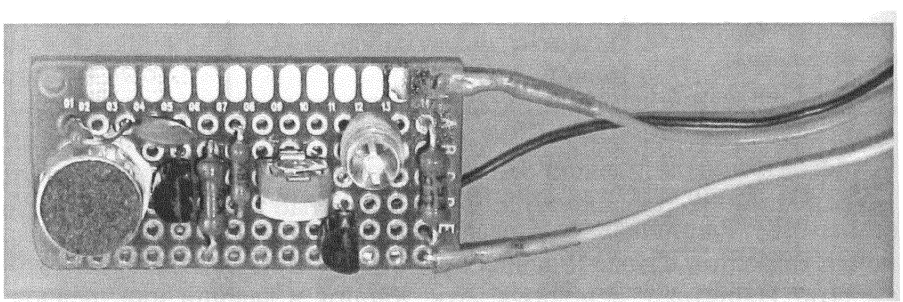

Рис. 9

гаснет). Транзистор VT1усиливает сигнал, снимаемый с микрофона, VT 2, своего рода ключ, фиксирующий наличие или отсутствие звукового сигнала (включает первый светодиод при высоком уровне сигнала и отключает при низком уровне). Изменения напряжения на его коллектор-эмиттерном переходе «отслеживает» аналоговый вход A0. Если напряжение резко уменьшается (при хлопке), программа переключает второй светодиод. Рассмотрим управляющий скетч:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
```

```
//
//////////////////////
//
//программа микрофон как акустический датчик — реакция 1 хлопок//
unsigned int  x=0;

void setup()
  {
     pinMode(6, OUTPUT); //канал сигнала для светодиода /
    digitalWrite(6,LOW);
     pinMode(A0, INPUT); // подключение датчика/
  }
     void loop()
    {

 x=analogRead(A0);
if (x<=350){ digitalWrite(6,! digitalRead (6)); delay (500);} //инвертируем
состояние вывода

     }

//
// Конец /
//
//////////////////////////////////////////////////////////////////
```

Программа постоянно «сканирует» напряжение на входе A0 и если оно вдруг в какой то момент становится меньше 1,7 Вольт меняет уровень сигнала на выводе 6 на противоположное значение. Далее делает паузу в полсекунды. То есть, щёлкая пальцами каждую секунду можно управлять светодиодом как «послушной мигалкой». Можно отдавать и голосовые команды, например, короткое «а». Если дать очень протяжное «ааааа» светодиод начнёт постоянно мигать с частотой переключения в 1 Гц. Датчик реагирует на любые громкие звуки. Ещё один вариант скетча, управляет светодиодом при двух последовательных быстрых хлопках:

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа микрофон как акустический датчик — реакция 2 хлопка
подряд//
unsigned int  x=0;
unsigned int  i=1;
void setup()
  {
     pinMode(6, OUTPUT); //канал сигнала для светодиода /
    digitalWrite(6,LOW);
     pinMode(A0, INPUT); // подключение датчика/
  }
     void loop()
    {

 x=analogRead(A0);
     if (x<=650)//если произошёл первый хлопок
      {
       delay (250);//ждём окончания первого хлопка
       for (i=1;i<=500;i++)//в цикле ожидаем и проверяем наличие
второго хлопка
        {
         x=analogRead(A0);//снимаем данные с датчика
         delay (1); //короткая временная пауза
         if (x<=650)//если произошёл второй хлопок
          {
            digitalWrite(6,! digitalRead (6));//меняем состояние выхода на
противоположное
            return;
          }
        }
     }
```

```
  }

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Этот скетч позволяет второму светодиоду не реагировать на одиночные хлопки — своего рода простейший «фильтр звука». Однако протяжное «ааааааа» вновь портит всю картину. Если всё-таки хочется отделять короткие звуковые сигналы от длинных сигналов (хотя бы частично) можно поэкспериментировать с таким вариантом программы:

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа акустического переключения RGB светодиода//
unsigned int  x=0;
unsigned int  n=0;
unsigned int  k=0;
long y1=0;
long y2=0;
void setup()
  {
     pinMode(6, OUTPUT); //канал сигнала для красного светодиода /
      digitalWrite(6,LOW);
      pinMode(7, OUTPUT); //канал сигнала для  зелёного светодиода
/
      digitalWrite(7,LOW);
       pinMode(8, OUTPUT); //канал сигнала для синего светодиода /
      digitalWrite(8,LOW);
     pinMode(A0, INPUT); // подключение датчика/
```

```
  }
     void loop()
    {

 x=analogRead(A0);
if (x<=350){y1=millis();}
if (x>=750){y2=millis();}
if (y1-y2>=200){
 digitalWrite(6,! digitalRead (6));n=n+1;k=k+1;
if (n==2)
 {
 digitalWrite(7,! digitalRead (7));n=0;
 }
 if (k==4)
 {
 digitalWrite(8,! digitalRead (8));k=0;
 }
 y1=y2;delay (300);
 }

    }

//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Для визуальной объективности оценки работы надо добавить ещё два светодиода или поставить в схему один трёхцветный (рис. 10). Относительную избирательность программы к коротким и протяжным звукам можно проверить, распевая или выкрикивая «а». Короткие сигналы будут переключать цвета свечения светодиода, а протяжные блокировать (иногда не сразу) переключения. Строка «**y1=y2;delay (300);**» ограничивает «скорость переключений» в 0,3 сек., а строка «**if (y1-y2>=200){....}**» определяет временной порог, отличающий корот-

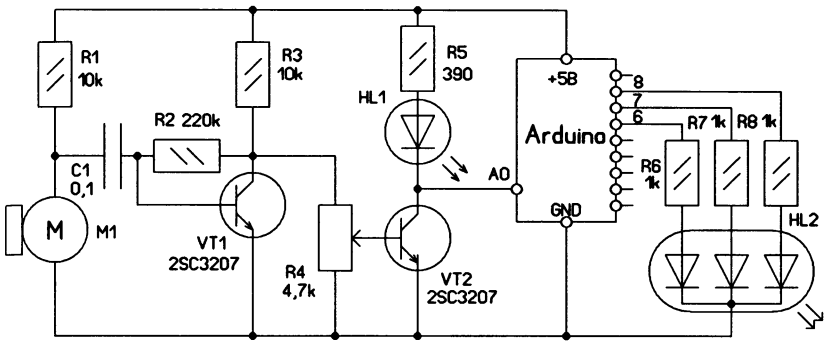

Рис. 10

кие звуки от длинных звуков (0,2 сек). Переменные n и k «определяют» очерёдность переключения кристаллов светодиода.

Довольно интересным практическим приложением для звукового датчика может стать самодельный светодиодный индикатор, напоминающий контур человека (рис. 11). Для его реализации понадобится всего 14 светодиодов (13 прямоугольных и один круглый по форме корпуса) произвольного цвета свечения. Принципиальная схема индикатора дана на рисунке 12. Номиналы резисторов подбираются экспериментально и их значения могут находиться в диапазоне от 200 Ом до 2 кОм. Светодиоды HL1-HL6 «статичные» и горят постоянно, имитируя неподвижность головы и тела человечка. Остальные разбиты на четыре пары — имитируют движения ног и рук человечка. Так, если на входе Л.Н. низкий уровень светит 10 светодиод (левая нога фигурки подогнута). При наличии высокого уровня загорается 9 светодиод — нога распрямляется.

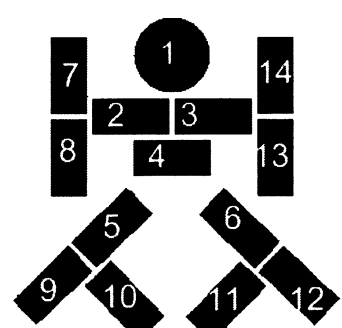

Рис. 11

Конструктивно индикатор реализован на плате квадратной формы со стороной 5-6 см. На (рис. 13)

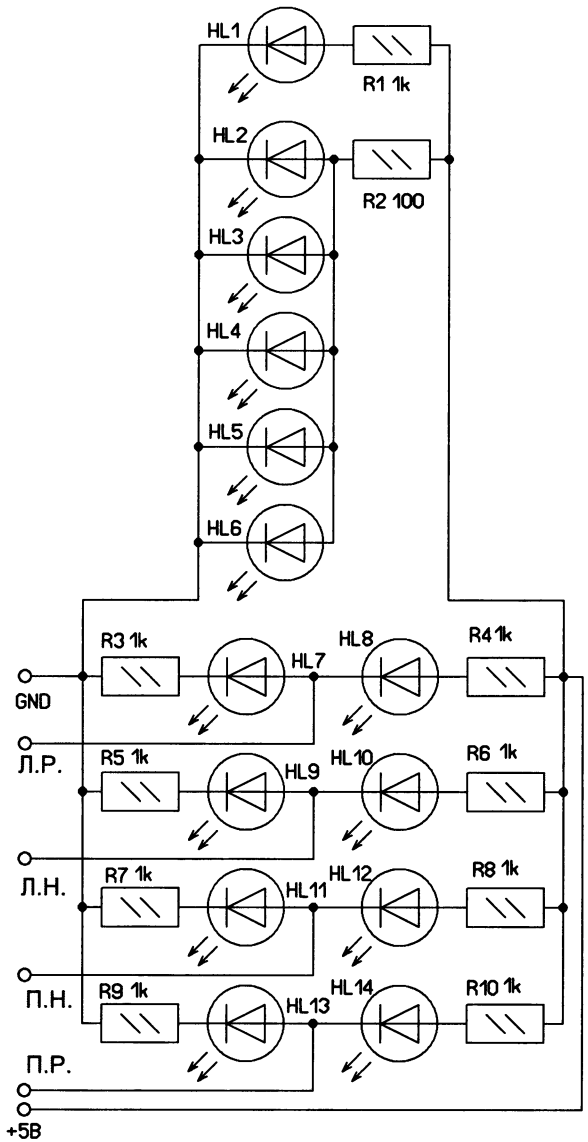
HL1
R1 1k
HL2
R2 100
HL3
HL4
HL5
HL6
R3 1k
HL7
HL8
R4 1k
GND
Л.Р.
R5 1k
HL9
HL10
R6 1k
Л.Н.
R7 1k
HL11
HL12
R8 1k
П.Н.
R9 1k
HL13
HL14
R10 1k
П.Р.
+5В

Рис. 12

представлен его внешний вид. Для того чтобы исключить взаимозасвечивание светодиодов боковую поверхность некоторых из них можно закрасить чёрной краской. Само основание платы под фигуркой также можно покрыть слоем краски.

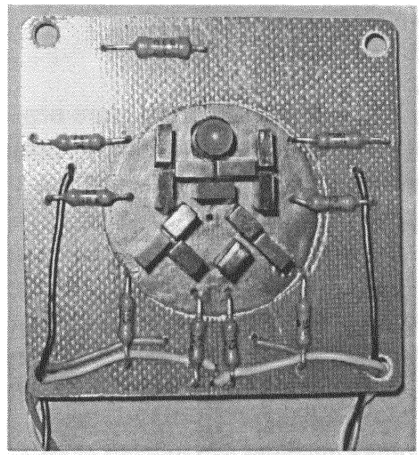

Рис. 13

Схема для экспериментирования изображена на рисунке 14. Возможные варианты поз фигурки даны на рис. 15- всего их возможно в данном варианте индикатора 16. Если на вход микрофона подавать короткие громкие звуки музыки (подойдут ритмичные композиции «на пианино», с барабанами или сольное пение с «акцентами» на слова) человечек начнёт танцевать под музыку, витиевато выбирая движения...

Вариант простого скетча дан ниже.

```
//////////////////////////////////////////////////////////////////////
//
// Arduino UNO
```

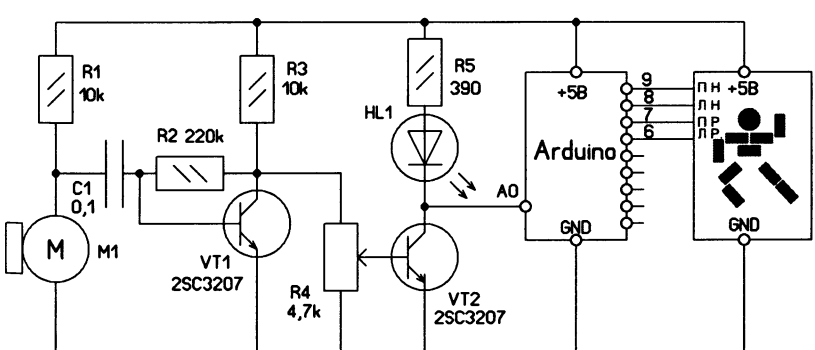

Рис. 14

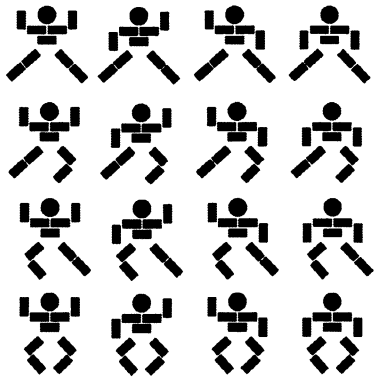

Рис. 15

```
//
//////////////////////
//
//Программа акустического пере-
ключения танцующего человечка//
unsigned int  x=0;

void setup()
  {
       pinMode(6, OUTPUT); //
канал сигнала для левой руки /
       digitalWrite(6,LOW);
       pinMode(7, OUTPUT); //
канал сигнала для правой руки /
       digitalWrite(7,LOW);
       pinMode(8, OUTPUT); //канал сигнала для левой ноги /
       digitalWrite(8,LOW);
       pinMode(9, OUTPUT); //канал сигнала для правой ноги /
       digitalWrite(9,LOW);

      pinMode(A0, INPUT); // подключение датчика/
  }
      void loop()
    {

 x=analogRead(A0);
if (x<=650)//если звуковой сигнал получен...
{
 digitalWrite(6,random (0,2));//псевдослучайно меняем логическое
состояние выходов...
 digitalWrite(7,random (0,2));
 digitalWrite(8,random (0,2));
 digitalWrite(9,random (0,2));
 delay(300);
 }
```

```
}

//
// Конец /
//
////////////////////////////////////////////////////////////////
```

При каждом относительно громком звуке происходит смена поз человечка случайным образом, благодаря команде random. Она возвращает одно из возможных значений 0 или 1, которое подаётся «в виде логического уровня» на соответствующий выход платы. Далее поза фиксируется на время в 0,3 секунды или до нового громкого звука.

Внимательный читатель, повторив конструкцию, заметит, что иногда фигурка не реагирует на громкий звук. Такое может, происходить если генератор случайных чисел выбрасывает четыре одинаковых (по отношению к предыдущим значениям) числа. Редкое, но меткое явление. Чтобы избежать такой «неприятности» программу можно немного переделать, добавив одно условие. Если все четыре числа, выброшенные генератором совпадают с предыдущими значениями — поменяй состояние вывода 9 на противоположное (принудительная смена позиции человечка). Программа:

```
////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа акустического переключения танцующего человечка//
unsigned int x=0;
unsigned int y1=0;
unsigned int y2=0;
unsigned int z1=0;
unsigned int z2=0;
unsigned int i1=0;
unsigned int i2=0;
```

```
unsigned int  j1=0;
unsigned int  j2=0;
void setup()
  {
      pinMode(6, OUTPUT); //канал сигнала для левой руки /
      digitalWrite(6,LOW);
      pinMode(7, OUTPUT); //канал сигнала для правой руки /
      digitalWrite(7,LOW);
      pinMode(8, OUTPUT); //канал сигнала для левой ноги /
      digitalWrite(8,LOW);
      pinMode(9, OUTPUT); //канал сигнала для правой ноги /
      digitalWrite(9,LOW);

     pinMode(A0, INPUT); // подключение датчика/
  }
     void loop()
    {

 x=analogRead(A0);
if (x<=650)//если звуковой сигнал получен...
{
  y1=y2;z1=z2;i1=i2;j1=j2;
  y2=random (0,2); z2=random (0,2);i2=random (0,2);j2=random (0,2);

if(y1==y2&z1==z2&i1==i2&j1==j2){j2=!j2;}// условие принудитель-
ной смены позы человечка

 digitalWrite(6,y2);//псевдослучайно меняем логическое состояние
выходов...
 digitalWrite(7,z2);
 digitalWrite(8,i2);
 digitalWrite(9,j2);
 delay(200);
 }

    }
```

```
//
// Конец /
//
//////////////////////////////////////////////////////////////////////
```

В остальном работа скетча не меняется. Теперь, завершая опыты с роботом, вооружим его акустическим включателем. Оговоримся сразу. Датчик позволяет только управлять запуском программы. Во время исполнения скетча он ловит много посторонних сигналов от моторов и зуммера, но это влияния на работу робота не оказывает — после запуска основной части скетча аналоговый вход A0 не опрашивается. Итак, схема модернизации дана на рисунке 16, внешнее расположение датчика на рисунке 17. Программа:

```
//////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
// Программа №4 для управления роботом /
```

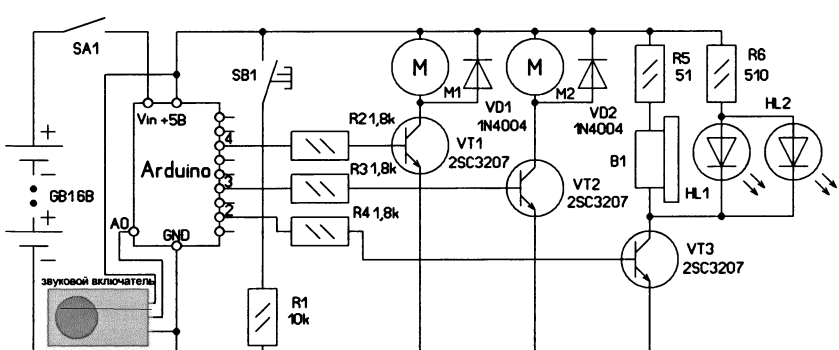

Рис. 16

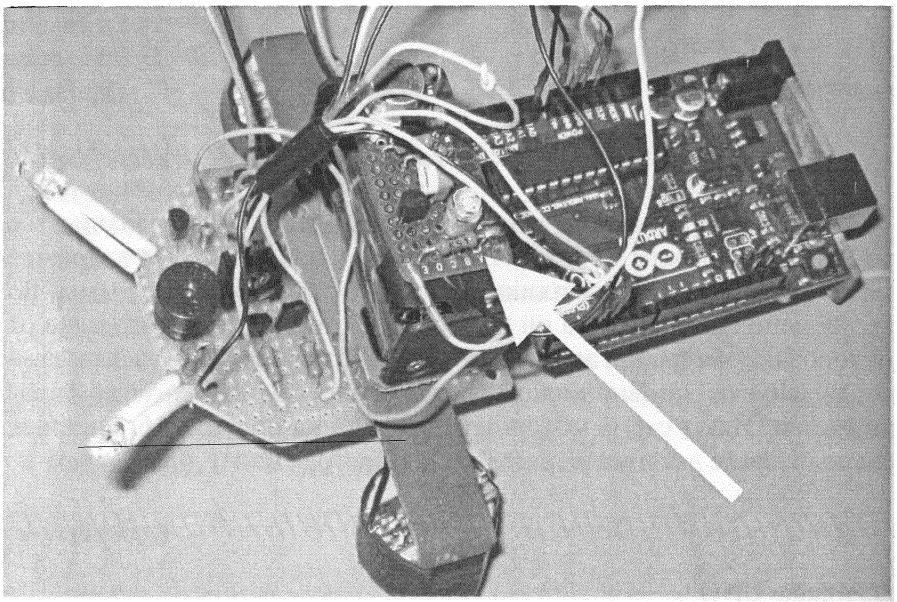

Рис. 17

```
//
long y1 = 0;
long y2 = 0;
long y3 = 0;
int x;
int x1;
int x2;
long y4 = 0;
void setup()
{
    pinMode(2, OUTPUT);//канал передачи звуковых и световых
эффектов
    digitalWrite(2, LOW);
    pinMode(3, OUTPUT);//канал управления левым мотором
    digitalWrite(3, LOW);
    pinMode(4, OUTPUT);// канал управления правым мотором
```

```
    digitalWrite(4, LOW);
    pinMode(A0, INPUT);// звуковой датчик управления пуском
}

void loop()
  {
    metka:
    x2=analogRead(A0);
    if (x2>=650){y2=millis();y1=y2;goto metka;}  //условие блокировки
запуска работы скетча

    metka0: // 1 фрагмент программы — звуки
    y2=millis();
    if (y2-y1<=4000)
   { y4=millis();
   if (y4-y3 >=250){tone(2, 2500+500*x,100*x1);y3=y4;x=random(1,5);
x1=random(1,3);}goto metka0;}
    else {y1=y2;goto metka1;}

    metka1: // 2 фрагмент программы — поворот
    y2=millis();
    if (y2-y1<=2000)
    {
    digitalWrite(3, HIGH);delay(800);digitalWrite(3, LOW);delay(1500);

    goto metka1;}
    else {y1=y2;goto metka2;}

    metka2: // 3 фрагмент программы — хождение
    y2=millis();
    if (y2-y1<=6300)
    {digitalWrite(4, HIGH);delay(270);
    digitalWrite(3, HIGH);digitalWrite(4, LOW);delay(170);
    digitalWrite(3, LOW);delay(500);goto metka2;}
    else {y1=y2; goto metka;}
  }
```

```
//
// Конец /
//
//////////////////////////////////////////////////////////////////
```

По щелчку робот «ругается», отворачивается и отходит, после повторного щелчка цикл работы повторяется. Важно чтобы после выполнения программы робот останавливался и замолкал (на момент выполнения строки «**else {y1=y2; goto metka;}**»), иначе возможны «несанкционированные повторы сценария».

На этом наш рассказ о роботе прервём...

ГЛАВА 5
АРДУИНО И ВРЕМЯ

Любая программа, изложенная в данной книге, так или иначе, постоянно реализуется при работе во времени. Оно является весьма важным инструментом при её настройке и функционировании. Иногда возникает необходимость срочно изменить ход её выполнения, например, по команде извне (нажатие кнопки, срабатывание датчика) или по времени. Такое событие программа может не заметить (хотя бы из-за наличия команды delay()), если не использовать её прерывания.

На плате Arduino UNO имеется два вывода (выводы 2 и 3), способные вызывать внешние прерывания. При его возникновении исполнение программы обрывается, она переходит к выполнению обработки прерывания и лишь, затем возвращается к исполнению прерванных команд.

Для вызова прерывания есть функция **attachInterrupt()**. Её первый аргумент — идентификатор прерывания (для вывода 2 это 0, вывода 3 — 1), второй аргумент — имя выполняемой в прерывании функции (выбирается произвольно). Третий аргумент задаёт условие запуска прерывания. Это может быть переход сигнала на входе с одного уровня на другой или заданный уровень сигнала. Практически, чаще используют три — CHANGE, RISING, FALLING (переход по спаду и переднему фронту сигнала и только по фронту или спаду).

Эксперимент №26
«ТРЁХЦВЕТНЫЙ RGB СВЕТОДИОД»

Рассмотрим ниже пример, позволяющий «оперативно» менять цвет свечения мигающего светодиода. Воспользуемся для наших целей трёхцветным светодиодом (три «разноцветных кристалла» в одном корпусе).

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа проверки аппаратного прерывания на RGB светодиоде//
const volatile int  x=0;
int  k=11;

void setup()
{
 pinMode(2, INPUT);//кнопка прерываний
 pinMode(13, OUTPUT);//канал красного кристалла
 digitalWrite(13,LOW);
pinMode(12, OUTPUT);//канал синего кристалла
 digitalWrite(12,LOW);
pinMode(11, OUTPUT);//канал зелёного кристалла
 digitalWrite(11,LOW);
attachInterrupt (x,fun,RISING);
}
 void fun()//переключение светодиода на прерывании
 {
  k=k+1;if(k==14){k=11;}//переключение цвета свечения по кругу
 }
void loop()//переключение светодиода каждые 0,05 секунды
```

```
{
digitalWrite(11,LOW);
digitalWrite(12,LOW);
digitalWrite(13,LOW);
 delay (50);
 digitalWrite(k, HIGH);//включение кристалла данного цвета
 delay (50);
 }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Схема для программы дана на рисунке 1. Введение в неё для управления кнопкой конденсатора C1 позволяет бороться с дребезгом контактов. Резистор R1 ограничивает броски тока при коммутации контактов.

В части void loop() происходит постоянное выключение всех кристаллов светодиода и включение кристалла под номером k. Переключение производится с временными задержками в 50 миллисекунд. Присвоение переменной k номера конкретного вывода происходит при прерываниях программы в части void fun() (выделено цветом). В

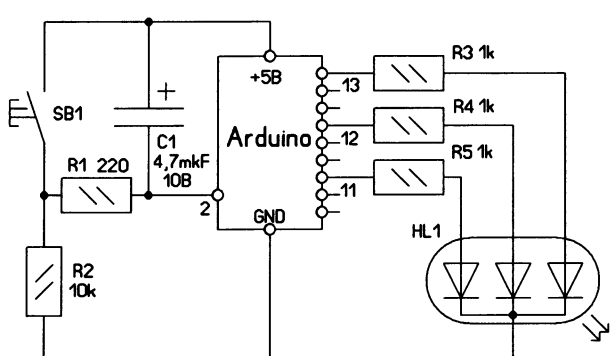

Рис. 1

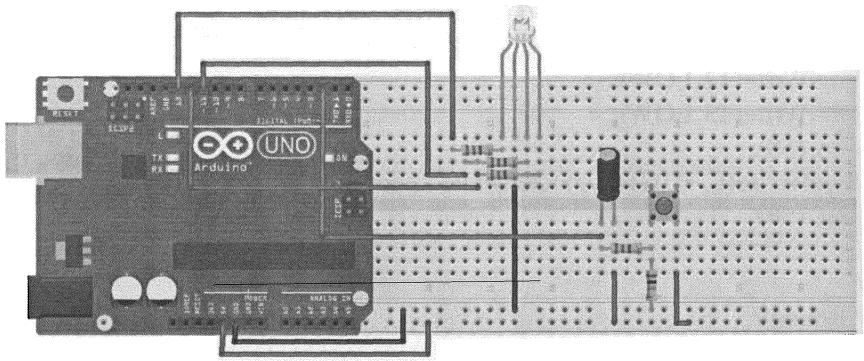

Рис. 2

ней по каждому нажатию кнопки производится перебор возможных значений — 11,12,13. Вызывая каждый раз прерывания, мы меняем цвет мигания светодиода, перебирая зелёный, синий, красный. Часть кода, относящаяся к реализации прерывания, выделена цветом.

Вариант макета дан на рисунке 2. Практически схему и скетч можно применить при проектировании светодиодного ночника-светильника. Правда рациональней будет «добавить в скетч возможность» перебора комбинаций цветов. Например, красный — синий, красный — зелёный, зелёный — синий.

Эксперимент №27
«СНОВА ЭЛЕКТРОННЫЙ КУБИК»

Время бежит быстро. От первых экспериментов до этих строк прошло уже два месяца. Вернёмся вновь к конструкции электронного кубика и рассмотрим подробно ещё варианты изготовления данной конструкции — закрепим так сказать пройденный материал.

Основой кубика служит самодельный светодиодный индикатор (рис. 3), соединённый проводниками с платой Ардуино. Принципиальная схема устройства приведена на рисунке 4. Всего в работе кубика задействовано 5 выводов платы, четыре управляют работой светодиодов (выводы 6-9 запрограммированы как выходы), один предназначен для кнопки SB1 (вывод 2 «прописан» как вход). С её помощью фиксируют выпавшую грань кубика. Резисторы R1-R4 тока ограни-

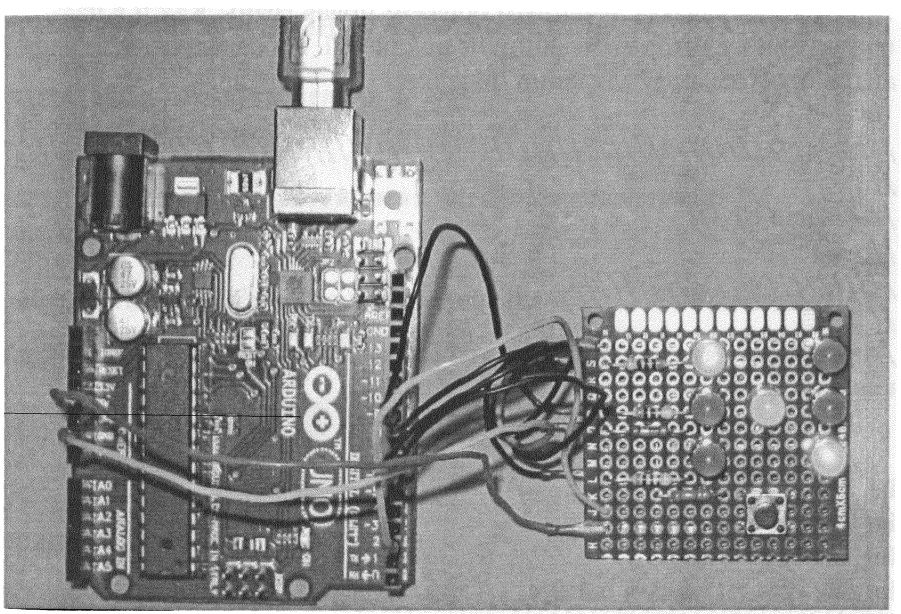

Рис. 3

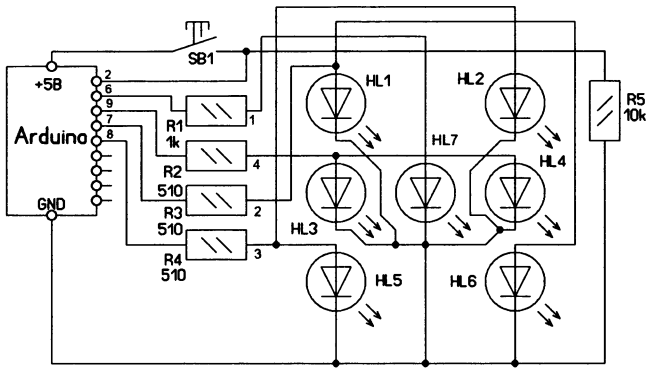

Рис. 4

чительные. В исходном состоянии, после запуска скетча светодиоды не горят. После кратковременного нажатия кнопки, высвечивается выпавшая грань, от одного до шести. При каждом следующем нажатии будет выпадать случайным образом новая грань. Как и в настоящем кубике, повтор выпадений не исключён. Ниже дан один из вариантов программы реализации поделки.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа для игрового кубика на 6 граней с использованием опроса кнопки в цикле//


int i=0;
int k=0;
int z=0;
int tabl [24] ={1,0,0,0, 0,1,0,0, 1,1,0,0, 0,1,1,0, 1,1,1,0, 0,1,1,1}; // значения логических уровней каналов на каждую грань
```

```
void setup()
{
 pinMode(2, INPUT);//кнопка выпадения грани
 pinMode(6, OUTPUT);//1 канал
 digitalWrite(6,LOW);
 pinMode(7, OUTPUT);//2 канал
 digitalWrite(7,LOW);
 pinMode(8, OUTPUT);//3 канал
 digitalWrite(8,LOW);
 pinMode(9, OUTPUT);//4 канал
 digitalWrite(9,LOW);

}

void loop()
{
 i=random(0,6);//псевдослучайный перебор граней
 if (digitalRead (2)==HIGH)  //если кнопка нажата ...
 {
 delay (200);//пауза временнАя
 if (digitalRead (2)==LOW)  //и отпущена ...
 {
  z=4*i;//выпавшая грань (номер в массиве 1канала грани)
 for(k=6;k<=9;k++)
 {
  digitalWrite(k,tabl[z]); //включение канала
  z++;//перебор каналов грани кубика
 }
 }
 }
 }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Разберём последовательно, как работают команды в скетче. Вначале вводится три переменные **i,k,z** (их тип **int**) с присвоением начальных значений 0. Строка **int tabl [24] ={1,0,0,0, 0,1,0,0, 1,1,0,0, 0,1,1,0, 1,1,1,0, 0,1,1,1}** именованный набор однотипных переменных, с доступом к отдельным элементам по их индексу, это массив. Вначале его указан тип переменных, далее название д массива (даём его произвольно). В квадратных скобках указано количество элементов — 24. После знака равенства, в фигурных скобках идёт перечисление переменных. В нашей программе они указывают значения логических уровней каналов для каждой грани. Например, первые четыре элемента «указывают грань 1». Индексация массива начинается с 0. Это значит, что для массива с 24- элементами, индекс 23 будет последним.

В части **void setup()** идёт назначение входов-выходов, присвоение начальных состояний.

Часть **void loop()** начинается с присвоения переменной **i** значения, отданного функцией **random()**. Переменная «случайным перебором» может получать целочисленные значения от 0 до 5 включительно. До нажатия кнопки данный процесс происходит постоянно — переменная «очень, очень быстро» меняет свои значения — номера возможно выпадающих граней.

Одновременно с этим процессом программа постоянно проверяет на истинность условие **if (digitalRead (2)==HIGH)**. Функция digitalRead(), она считывает значение с заданного входа — HIGH или LOW. В нашем случае аргументом имеет номер вывода — 2. Условие звучит: если значение логического уровня на входе 2 тождественно единице, то... Далее следует выполнение команд после проверки истинности условия. Команда **delay (200)** останавливает выполнение программы на 200 миллисекунд (время необходимое для отпускания пальцем кнопки). Затем повторная проверка на наличие низкого уровня сигнала и выполнение части программы по «фиксации выпавшей грани». Такая реализация отслеживания работы кнопки позволяет программе реагировать выпадением граней лишь на кратковременные её нажатия. Ели нажать и зафиксировать кнопку, смены выпавших граней происходить не будет.

Итак, после кратковременного нажатия пойдёт выполнение команды **z=4*i** — присвой переменной **z** значение переменной **i** (которое она

имеет в данный момент времени сразу после нажатия кнопки), умноженное на 4. Таким образом, она указывает номер элемента в массиве. Этот элемент (0 или 1) есть значение логического уровня первого канала выпавшей грани. Например, при i=4, z=16 выпавшая грань «5», логический уровень 1 канала единица (элемент массива под номером 16).

Далее идёт строка **for(k=6;k<=9;k++)**. Поясним некоторые новые управляющие операторы. Это цикл с возможной интерпретацией — для переменной **k** от шести до девяти включительно (**<=** оператор сравнения «меньше или равно»), каждый раз к счётчику (**k**) прибавляй единицу и выполняй действия внутри фигурных скобок (для данного цикла — включай, выключай светодиоды согласно «коду» из массива). Счётчик **z** в каждом «круге цикла» увеличивается на 1. Например, для грани «5» в цикле последовательно будет выполнено: digitalWrite(6, HIGH); digitalWrite(7, HIGH); digitalWrite(8, HIGH); digitalWrite(9, LOW);... Для высокого уровня HIGH тождественно 1, низкого LOW тождественно 0. То есть, по схеме, зажгутся светодиоды HL1, HL2, HL5, HL6, HL7. Вторым аргументом в функции digitalWrite() является элемент массива, его мы вызываем командой: присвой аргументу значение элемента массива под номером **z** (**digitalWrite(k,tabl[z]);**). Затем программа вновь вернётся к перебору значений **i** и проверки состояния кнопки, до нового нажатия.

Элементы индикатора размещены на макетной квадратной плате со стороной 35...50 мм, для её изготовления можно применить гетинакс, текстолит или другой диэлектрический материал. Постоянные резисторы — МЛТ, С2-23. Светодиоды — маломощные с соответствующей формой корпуса и конечно одинакового цвета. Питать устройство можно от стабилизированного зарядного устройства сотового телефона или компьютера через USB-разъём. Её монтаж можно осуществить внутри пластикового контейнера, подходящего размера и использовать как приставку к компьютеру. Номиналы резисторов в индикаторе (0,2...3 кОм) можно подобрать экспериментально, по желаемой яркости свечения светодиодов. Но не следует забывать, что максимальный выходной ток выходов Arduino — 40 мА. В программе можно изменить значение константы, задающий задержку (200). Соединив платы вместе согласно схеме и подключив Arduino к компьютеру, можно загружать программу, используя соответствующую среду разработки Arduino IDE.

Программу можно легко модернизировать, сделав кубик «виртуальным» — на 8 граней (добавить 7 и зеро — 0). Для этого в массив добавляем последовательность **0,0,0,0, 1,1,1,1** , меняем его размер (заменяем 24 на 32), меняем аргумент в **random(0,8)**.

Ещё один вариант скетча, работающий на прерываниях, с возможностью подавления дребезга контактов программно (переменная n) дан ниже:

```
////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа для игрового кубика на 6 граней с использованием прерываний//
const int  x=0;
volatile int  n=1;
int  i=0;
int  k=0;
volatile int  z=0;
int tabl [24] ={1,0,0,0, 0,1,0,0, 1,1,0,0, 0,1,1,0, 1,1,1,0, 0,1,1,1};
void setup()
{
 pinMode(2, INPUT);
 pinMode(6, OUTPUT);
 digitalWrite(6,LOW);
 pinMode(7, OUTPUT);
 digitalWrite(7,LOW);
 pinMode(8, OUTPUT);
 digitalWrite(8,LOW);
 pinMode(9, OUTPUT);
 digitalWrite(9,LOW);
 attachInterrupt (x,fun,RISING);
}
 void fun()//функция выбора выпадающей грани на прерывании
```

```
 {
  if(n==1)//условие, устраняющее дребезг контакта
  {
  n=0;
   z=4*i;//фиксация выпавшей грани
   }
 }
void loop()
{
    i=random(0,6);

   if (n==0)
    {
      for(k=6;k<=9;k++)
       {
        digitalWrite(k,tabl[z]);
        z++;// перебор каналов грани кубика
       }

 delay (300);
 n=1;

    }
}
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Завершить эксперименты начального уровня обучения программированию мне хочется на примере готового изделия — прибора учёта расхода важного, очень важного для человека ресурса — времени. Поговорим о конструкции часов на Ардуино. Попробуем сделать их без применения дополнительных плат и модулей, жертвуя точностью хода в угоду простоте реализации. Но вначале небольшая разминка.

Эксперимент №28 «СНОВА ЭЛЕКТРОННЫЙ СЕКУНДОМЕР»

Для реализации воспользуемся микроамперметром с током полного отклонения 100 микроампер. Соберём схему по рисунку 5. Для отсчёта секунд применим микроамперметр. А для выполнения соответствия между величиной тока через микроамперметр и отсчитанными секундами применим ШИМ. Подобным мы уже пользовались в эксперименте с длинномером. Внешний вид макета дан на рисунке 6.

Резисторы R3, R4 тока ограничительные. Последний позволяет подстраивать ток полного отклонения стрелки в значение близкое к 100 микроампер (регулировка точности показаний прибора). Подавление дребезга управляющей кнопки — аппаратное (схемное) с помощью конденсатора.

Предел измерения отрезков времени от нуля до ста секунд. Такой интервал позволяет не делать дополнительную фальшь — шкалу для измерительного прибора. Всё управление секундомером осуществляется одной кнопкой SB1. Последовательно нажимая после подачи напряжения на плату, мы запускаем секундомер, останавливаем секундомер, обнуляем его показания. Управляющий цикл состоит из трёх режимов — запуск, остановка, обнуление показаний. Скетч дан ниже.

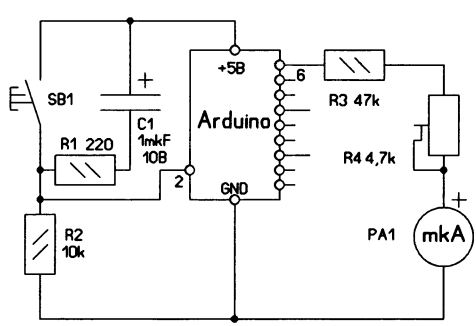

Рис. 5

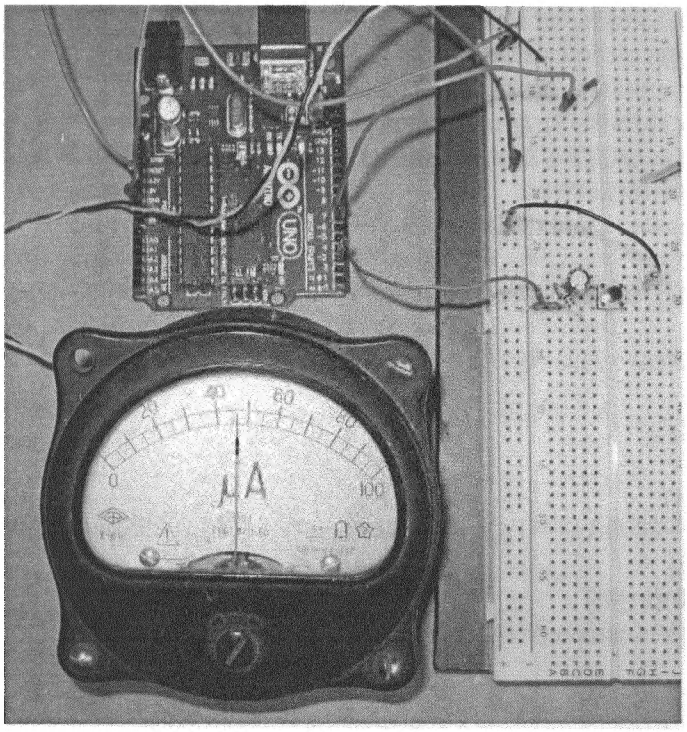

Рис. 6

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа секундомера на 100 секунд на микроамперметре//
const int  x=0;
int  k=0;
int  x1=0;
int  n=0;
long y1=0;
```

```
long y2=0;
void setup()
{
 pinMode(2, INPUT);//кнопка прерываний
 pinMode(6, OUTPUT);//канал для микроамперметра
 digitalWrite(6,LOW);
attachInterrupt (x,fun,RISING);
}
 void fun()//управление секундомером на прерывании
 {
  n=n+1;if(n>=3){n=0;}//переключение режимов секундомера по кругу
 }
void loop()
{
y2=millis();
if(y2-y1>=1000&n==1){k=k+1;x1=map(k,0,100,0,255);analogWrite(6,x
1);y1=y2;}
if(k>=100){k=0;}
if(n==0){k=0;analogWrite(6,0);}
 }
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

В основной части программы постоянно вызывается функция millis(), переменной y2 присваивается её значение. Через каждый интервал времени в 1000 миллисекунд при соблюдении режима отсчёта после запуска (n=1) происходит выполнение команд: **k=k+1;x1=map (k,0,100,0,255);analogWrite(6,x1);y1=y2**. Переменная k увеличивает своё значение на единицу (число прошедших секунд), x1принимает значение пропорциональное данному числу в интервале 0-255. На выход 6 подаётся «усреднённое напряжение» пропорциональное k, x1 в интервале 0-5 Вольт. Именно его фиксирует микроамперметр, работающий в режиме вольтметра (добавочные резисторы R3, R4). Переменной y1 присваивается текущее значение y2 — так происходит

запуск нового отсчёта следующей секунды. При повторном нажатии кнопки (n=2) в части loop() ничего не происходит кроме нарастания значения y2 — стрелка прибора показывает «остановленное время». При ещё одном нажатии кнопки выполняется строка кода: **if(n==0) {k=0;analogWrite(6,0);}** — стрелка секундомера сбрасывается в ноль. Прибор готов к повторному запуску нового измерения интервала времени. В программе изначально заложена «небольшая» погрешность измерения времени. При повторном запуске секундомера практически сразу значение k равно 1, а должно быть равно 0. Как это исправить, внимательный читатель ты догадаешься сам.

Эксперимент №29
«АНАЛОГОВЫЕ ЧАСЫ»

Назовём условно данный эксперимент так. Изготовление конструкции удобнее начинать с изготовления фальшь — шкалы прибора (рис. 7). Для этого головку прибора аккуратно разбирают, снимают шкалу. Её контур обводят на листе белой бумаги, наносят карандашом линии — продолжения рисок и по ним разграничивают на бумажном контуре новые шкалы. Верхняя шкала-линейка предназначена для отсчёта часов в интервале 0 — 24, нижняя для минут от 0 до 60 (числа нарисованы от руки). Шкалу крепят на поверхность настоящей шкалы при помощи скотча, закрывая бумагу прозрачным слоем сверху и заворачивая за края, собирают прибор.

Схема часов дана на рисунке 8. Это пробный вариант, его основная цель проверить работоспособность предполагаемого готового изделия. Светодиоды HL1, HL2 нужны как указатели для считывания показаний прибора. При свечении верхнего (по схеме) считываем число минут, нижнего — число часов. Программа-пробник дана ниже.

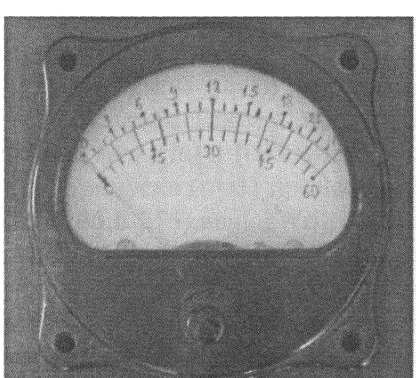

Рис. 7

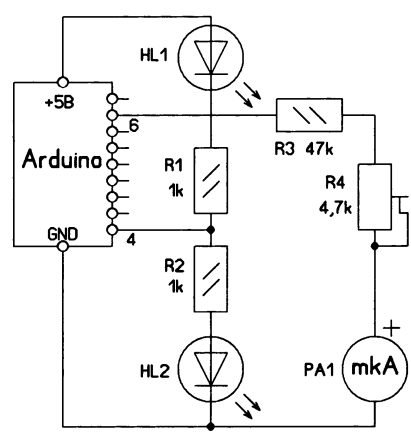

Рис. 8

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа №1 для часов на микроамперметре//
int  n=0;
int  k=50;//изначально выставленные минуты
int  k1=0;
int  j=12;//изначально выставленные часы
int  j1=0;
long y1=0;
long y2=0;
long y3=0;
void setup()
{
 pinMode(6, OUTPUT);//канал для отсчёта часов и минут
 digitalWrite(6,LOW);
 pinMode(4, OUTPUT);//канал — указатель индикации
 digitalWrite(4,LOW);
}

void loop()
{
y2=millis();

if(y2-y1>=60000){k++;y1=y2;}//условие отсчёта минут
if(k==60){k=0;j++;}//условие отсчёта часов
if(j==24){j=0;}//условие обнуления числа часов за сутки

if(y2-y3>=10000&n==1){n=0;y3=y2;k1=map(k,0,60,0,255);analogWrite
(6,k1);}
//условие индикации числа минут
if(y2-y3>=10000&n==0){n=1;y3=y2;j1=map(j,0,24,0,255);analogWrite(
6,j1);}
```

```
//условие индикации числа часов
digitalWrite(4,n);//указатель для индикации часов, минут
}
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Переменная k считает минуты, переменная j считает часы, переменная n отвечает за переключения режимов индикации часов, принимая значения 0 или 1. Одновременно с этим осуществляет «попеременную коммутацию» светодиодов-указателей.

Меняя начальные значения переменных k и j, перезагружая и запуская скетч на разные промежутки времени можно отсматривать «равномерность нарисованных шкал», при необходимости корректируя их.

После этого этапа можно собирать основную конструкцию. Вариант внешнего вида дан на рисунке 9. Схемное решение дано на рисунке 10. Конденсатор C1 нужен для прекращения нежелательного «пищания микроамперметра», которое может возникнуть при ШИМ сигнала на выходе 6. Кнопка SB1перезапускает программу, обнуляя текущее время. Кнопка SB2 позволяет выставлять значение текущего

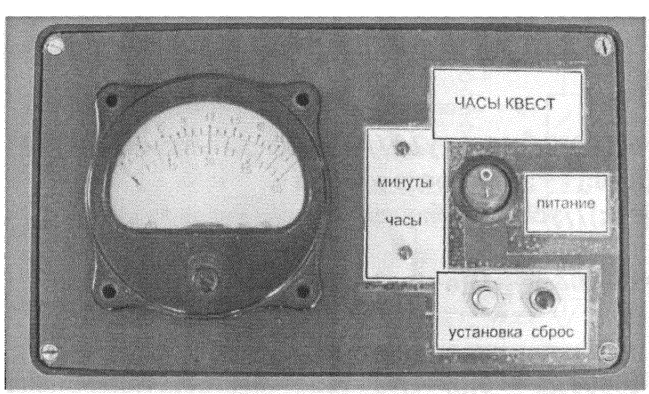

Рис. 9

времени при запуске часов. Когда горит первый светодиод, выставляется количество минут, когда второй светодиод светится, выставляется число часов. Переключение светодиодов происходит каждые 10 секунд. Питать конструкцию можно от комплекта батареек, суммарным напряжением 9 В. Потребляемый ток (в зависимости от вида платы) составляет 25-45 мА. ~~Вариант программы работы~~ дан ниже. Часы в течении суток работы «не ушли из границ» единицы измерения.

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа №2 для часов на микроамперметре//
int n=0;
int k=0;
int k1=0;
int j=0;
int j1=0;
long y1=0;
```

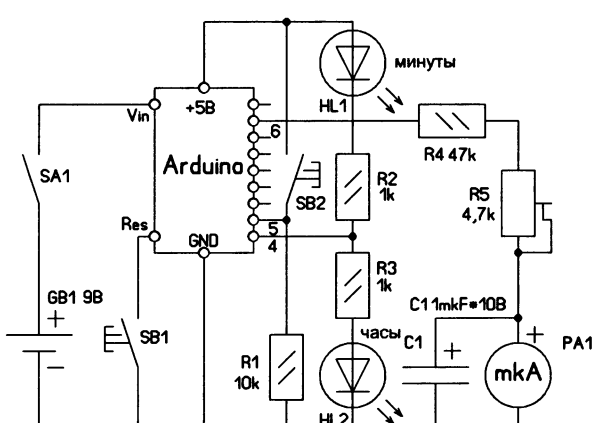

Рис. 10

```
long y2=0;
long y3=0;
void setup()
{
 pinMode(6, OUTPUT);//канал для отсчёта часов и минут
 digitalWrite(6,LOW);
 pinMode(4, OUTPUT);//канал- указатель индикации
 digitalWrite(4,LOW);
 pinMode(5, INPUT);//канал начальных установок времени
}

void loop()
{
y2=millis();

if(y2-y1>=60000){k++;y1=y2;}//условие отсчёта минут
if(k==60){k=0;j++;}//условие отсчёта часов
if(j==24){j=0;}//условие обнуления числа часов за сутки

if(y2-y3>=10000&n==1){n=0;y3=y2;k1=map(k,0,60,0,255);analogWrite
(6,k1);}
//условие индикации числа минут

if(y2-y3>=10000&n==0){n=1;y3=y2;j1=map(j,0,24,0,255);analogWrite(
6,j1);}
//условие индикации числа часов

digitalWrite(4,n);//указатель для индикации часов, минут

if(digitalRead(5)==HIGH&n==1){j++;if(j==24){j=0;}
j1=map(j,0,24,0,255);analogWrite(6,j1);delay(200);}
//процедура настройки начального времени — число часов

if(digitalRead(5)==HIGH&n==0){k++;if(k==60){k=0;}
k1=map(k,0,60,0,255);analogWrite(6,k1);delay(200);}
//процедура настройки начального времени — число минут
```

```
}
//
// Конец /
//
////////////////////////////////////////////////////////////////////////
```

Интерпретируем финальную часть кода, отвечающую за процедуру настройки начального значения времени. Итак, часть:

if(digitalRead(5)==HIGH&n==1){j++;if(j==24){j=0;}
j1=map(j,0,24,0,255);analogWrite(6,j1);delay(200);}

Если на входе 5 высокий уровень (кнопка 2 кратковременно нажата) и переменная n равна1 — увеличь число часов (j) на единицу, проверь не стало ли оно равно 24. Если стало, обнули его. Подай напряжение пропорциональное числу выставленных часов на выход 6 и сделай временную паузу в 0,2 секунды, чтобы палец успел отпустить кнопку.

Такую же интерпретацию имеет и часть, относящаяся к настройке минут:

if(digitalRead(5)==HIGH&n==0){k++;if(k==60){k=0;}
k1=map(k,0,60,0,255);analogWrite(6,k1);delay(200);}

После установки времени часы ведут его отсчёт до новой установки или отключения питания.

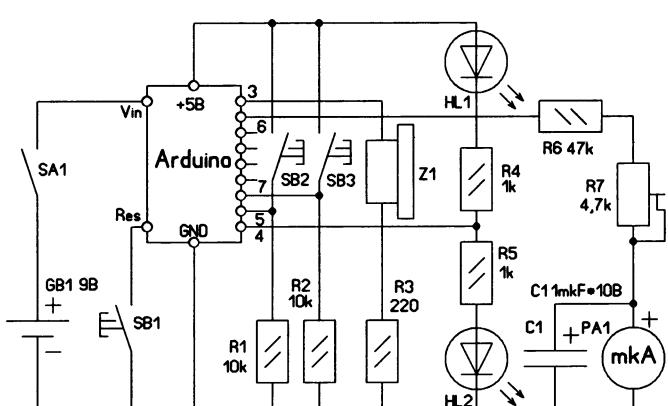

Рис. 11

В заключение ещё вариант схемной и программной модернизации — установка будильника. Вариант на размышление читателю — схема рис. 11. Скетч ниже.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа №3 для часов на микроамперметре//
int  n=0;
int  n1=1;
unsigned int  k=0;
int  k1=0;
unsigned int  k2=0;
unsigned int  j=0;
int  j1=0;
unsigned int  j2=0;
long y1=0;
long y2=0;
long y3=0;
void setup()
{
 pinMode(6, OUTPUT);//канал для отсчёта часов и минут
 digitalWrite(6,LOW);
 pinMode(4, OUTPUT);//канал- указатель индикации
 digitalWrite(4,LOW);
 pinMode(5, INPUT);//канал начальных установок времени
 pinMode(3, OUTPUT);//канал для зуммера будильника
 digitalWrite(3,LOW);
 pinMode(7, INPUT);//канал управления будильником
}

void loop()
{
```

```
y2=millis();

if(y2-y1>=60000){k++;y1=y2;}//условие отсчёта минут
if(k==60){k=0;j++;}//условие отсчёта часов
if(j==24){j=0;}//условие обнуления числа часов за сутки
if(y2-y3>=10000&n==1){n=0;y3=y2;k1=map(k,0,60,0,255);analogWrite
(6,k1);}
//условие индикации числа минут
if(y2-y3>=10000&n==0){n=1;y3=y2;j1=map(j,0,24,0,255);analogWrite(
6,j1);}
//условие индикации числа часов
digitalWrite(4,n);//указатель для индикации часов, минут
if(digitalRead(5)==HIGH&n==1){j++;if(j==24){j=0;}j1=map(j,0,24,0,2
55);analogWrite(6,j1);delay(200);}
//процедура настройки начального времени число часов
if(digitalRead(5)==HIGH&n==0){k++;if(k==60){k=0;}k1=map(k,0,60,
0,255);analogWrite(6,k1);delay(200);}
//процедура настройки начального времени число минут

////////////////////////// будильник — небольшая добавка к скетчу
////////////////////////////////
if(k2==k&j2==j){tone(3,600,1000);}//условие срабатывания зуммера
if(digitalRead(7)==HIGH&n1==1){noTone(3);k2=-1;j2=-
1;n1=0;delay(200); }
//условие отключения звука зуммера
if(digitalRead(7)==HIGH&n1==0){k2=k;j2=j;n1=1;delay(200); }
//запись-фиксация времени срабатывания будильника
}
//
// Конец /
//
//////////////////////////////////////////////////////////////////
```

Кнопка 3 управляет его работой. При одном нажатии происходит отключение будильника, при другом включение с записью времени срабатывания зуммера Z1.

Последовательность работы с часами такая: после включения часов они обнулены, зуммер пищит. Нажимаем кнопку 3 — обнуляем будильник. Выставляем кнопкой 2 желаемое время его срабатывания, нажимаем кнопку 3 — зуммер звучит (указывая время, когда это произойдёт). Далее кнопкой 2 выставляем текущее время — зуммер замолкает. Начинается его отсчёт. При повторном совпадении значений текущего и выставленного времени будильник сработает (звучать будет в течении минуты или пока не нажмут кнопку3). В остальном работа часов аналогична предыдущей версии.

Эксперимент №30 «ПЕСОЧНЫЕ ЧАСЫ НА СВЕТОДИОДНЫХ МАТРИЦАХ»

Чаще всего для конструирования с помощью Ардуино требуются дополнительные компоненты, собранные на отдельных платах. На рисунке 12 изображен такой модуль — плата со светодиодной матрицей с количеством светодиодов 8 на 8 (64 штуки) и микросхемой управления (MAX7219). А для управления ими, в свою очередь, нужны дополнительные функции. Библиотека — это набор функций, предназначенных для того, чтобы максимально упростить работу с различными датчиками, ЖК-экранами, модулями и пр. Например, встроенная библиотека LiquidCrystal позволяет легко взаимодействовать с символьными LCD-экранами. Существуют десятки полезных библиотек, которые можно скачать в Интернете. Стандартные библиотеки Ардуино и ряд наиболее часто используемых дополнительных библиотек перечислены ниже в справке. Но перед тем, как использовать данные библиотеки, необходимо сперва установить их...

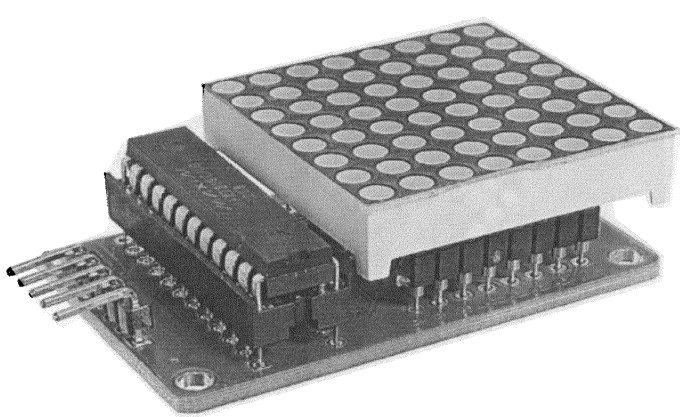

Рис. 12

EEPROM — чтение и запись в энергонезависимую память микроконтроллера

Ethernet — работа с сетью через плату расширения Arduino Ethernet

Firmata — взаимодействие Arduino и компьютера

GSM — соединение с сетью GSM/GRPS через GSM-плату расширения.

LiquidCrystal — вывод на жидкокристаллический тестовый дисплей.

SD — для чтения и записи данных на SD-карту памяти.

Servo — для управления сервоприводами.

SPI — взаимодействие Arduino с периферией по последовательному периферийному интерфейсу (SPI)

SoftwareSerial — реализация последовательных интерфейсов на любых цифровых контактах.

Рассмотрим практическое применение библиотек и дополнительных модулей к основной плате на примере конструкции матричных песочных часов (рис. 13) и библиотеки LedControl.h. Подробнее об установке данной библиотеки на свой компьютер можно узнать, задав в поисковике строку — «установка библиотеки LedControl .

Сначала опробуем работоспособность модуля — подключим его к плате, согласно рисунку 14 и попробуем анимировать изображение сердечка (рис. 15). Рассмотрим работу скетча.

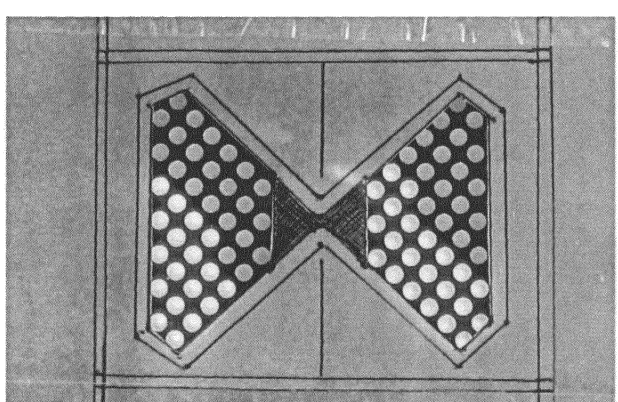

Рис. 13

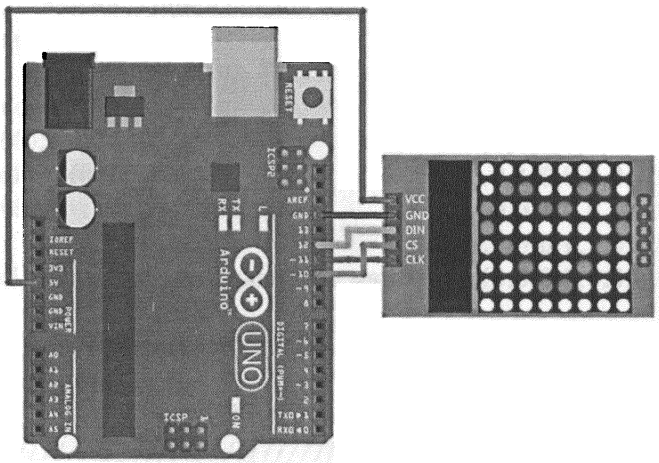

Рис. 14

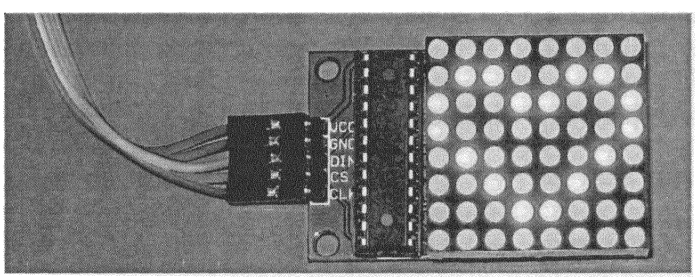

Рис. 15

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа для светодиодной матрицы 8*8 и max7219 //

#include «LedControl.h»//подключаем данную библиотеку
```

```
int i=0;int j=0;
int tabl[8][8]={0,0,0,0,0,0,0,0,
                0,1,1,0,0,1,1,0,
                1,0,0,1,1,0,0,1,
                1,0,0,0,0,0,0,1,
                0,1,0,0,0,0,1,0,
                0,0,1,0,0,1,0,0,
                0,0,0,1,1,0,0,0,
                0,0,0,0,0,0,0,0};// храним картинку в массиве

LedControl LC = LedControl(12, 11, 10, 1);// создаём объект класса для
1 индикатора
// при этом выводы 12-DIN 11-CLKC 10-CS //

void setup()

 LC.shutdown(0, false);//выключаем энергосберегающий режим
 LC.setIntensity(0, 4);// устанавливаем интенсивность в 4 единицы
между 0 и 15
 LC.clearDisplay(0);//очищаем дисплей
}

void loop()
{
 for (j=0;j<=7;j++)//цикл в цикле для перебора элементов массива
 {
 for (i=0;i<=7;i++)
  {
 LC.setLed(0, i, j, true);//последовательное включение каждого свето-
диода матрицы
 delay(150);
 LC.setLed(0, i, j, tabl[i][j]);//включение, оставление включёнными
только светодиодов контура картинки
  }
 }
  delay(1000);
```

```
  for (j=0;j<=5;j++)//цикл биения сердца — переключение изображения картинки
   {
   LC.shutdown(0, true);
    delay(300);
    LC.shutdown(0, false);
    delay(1000);
   }
   LC.clearDisplay(0);
 }
 //
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Строка **#include «LedControl.h»** указывает на необходимость использования данной библиотеки. Ниже задан двухмерный массив **int tabl[8][8]={...};** с количеством элементов 64. В нём хранятся логические уровни для светодиодов матрицы, необходимые в построении изображения картинки. В нашем примере это контур сердечка. Строка **LedControl LC = LedControl(12, 11, 10, 1);** создаёт в программе объект класса для единственного индикатора. Аргументы в скобках задают номера выходов платы и соответственно порядок подключения входов модуля. Первый это DIN, второй CLKC, третий CS. Четвёртый аргумент указывает число используемых индикаторов (в нашем примере он один). Таким образом, можно использовать на три выводы платы до восьми индикаторов. А подключая к иным выводам дополнительные объекты класса, число индикаторов можно увеличивать ещё.

Далее идут команды **LC.shutdown(0, false);** — выключает индикатор под номером 0 (нумерация начинается с 0 и заканчивается цифрой 7) из режима экономии энергии, **LC.setIntensity(0, 4);** — устанавливает интенсивность свечения в 4 единицы (условно интенсивность разбита на 16 уровней с нумерацией от 0 до 15 по возрастанию). Команда **LC.clearDisplay(0);** очищает экран, гасит все пиксели матрицы под номером 0.

Далее в основной части цикла происходит постоянный перебор элементов массива и включение-выключение перебираемых точек матрицы. Два счётчика j и i обеспечивают смену опроса элементов матрицы по схеме «цикл в цикле». Команда **LC.setLed(0, i, j, true);** включает светодиод столбца под номером j и строки под номером i (нумерация строк и столбцов также идёт от 0 до7). Такое состояние сохраняется в течении 150 миллисекунд (команда **delay(150);**), а далее выполняется команда **LC.setLed(0, i, j, tabl[i][j]);**- оставление включёнными только светодиодов контура картинки (значение **tabl[i][j]**) может быть либо 0, либо 1 — выключить, включить соответствующий светодиод). То есть бегущая последовательно по столбцам точка-светодиод зажигает контур сердца. Далее небольшая пауза в 1 секунду и сердце начинает пульсировать — шесть раз гаснет и вспыхивает. Выполнение команд **LC.shutdown(0, true);** — включить экономный режим (отключить свечение матрицы) и **LC.shutdown(0, false);**- выключить экономный режим, повторённые в цикле обеспечивают данный отрезок анимации. В конце программы происходит очистка экрана, и цикл повторяется вновь. Таким образом, сердце вырисовывается, пульсирует и исчезает, чтобы появиться вновь.

Вернёмся вновь к песочным часам (рис. 13). В данной конструкции использовано два индикатора, расположенные «по диагонали» отно-

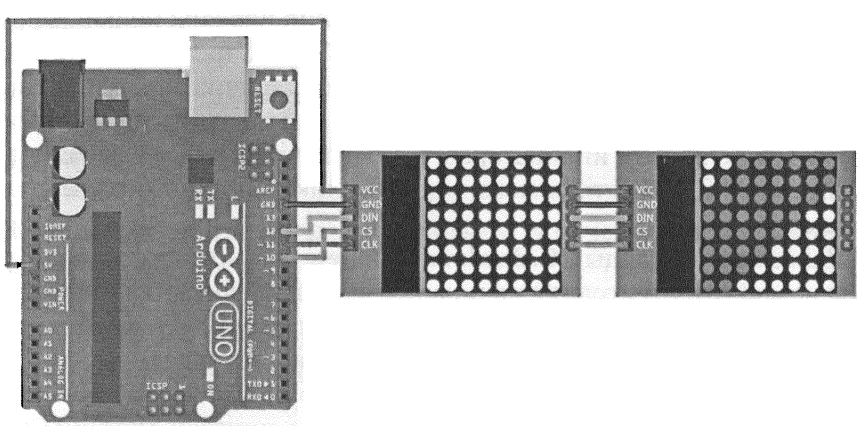

Рис. 16

сительно друг друга. В течении одной минуты песок-светодиоды пересыпаются из верхней чаши в нижнюю. При этом каждую секунду, индикаторы меняют яркость свечения, отмеряя отдельные секунды. После минутного цикла наступает небольшая пауза в пять секунд и цикл повторяется. На рисунке 16 дана «схема» подключения индикаторов и платы. Ниже дан скетч.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа для светодиодной матрицы 8*8 и max7219 песочные часы
//

#include «LedControl.h»//подключаем данную библиотеку
int i=0;int j=0;int k=1;int k1=0;int z=0;
int tabl[8][8]={0,0,1,1,1,1,1,1,
          0,1,1,1,1,1,1,1,
          1,1,1,1,1,1,1,0,
          1,1,1,1,1,1,0,0,
          1,1,1,1,1,0,0,0,
          1,1,1,1,0,0,0,0,
          1,1,1,0,0,0,0,0,
          1,1,0,0,0,0,0,0};// храним картинку в массиве

LedControl LC = LedControl(12, 11, 10, 2);// создаём объект класса для
2 индикаторов
// при этом выводы 12-DIN 11-CLK 10-CS //

void setup()
{
 LC.shutdown(0, false);//выключаем энергосберегающий режим
 LC.setIntensity(0, 8);// устанавливаем интенсивность в 8 единиц
между 0 и 15
```

```
 LC.clearDisplay(0);//очищаем матрицу 1
 LC.shutdown(1, false);//выключаем энергосберегающий режим
 LC.setIntensity(1, 8);// устанавливаем интенсивность в 8 единиц
между 0 и 15
 LC.clearDisplay(1);//очищаем матрицу 2
}

void loop()
{
 ////////// зажигаем верхнюю чашу ////////////////////
  for (i=7;i>=0;i--) //цикл в цикле для перебора элементов массива
 {
 for (j=7;j>=0;j--)
  {
   LC.setLed(1, j, i, tabl[j][i]);
  }
  }
     delay(5000);
  /////////////// «пересыпаем песок» в нижнюю чашу
/////////////////
 for (i=7;i>=0;i--) //цикл в цикле для перебора элементов массива
 {
 for (j=7;j>=0;j--)
  {

 LC.setLed(0, j, i, tabl[j][i]);
 if (tabl[j][i]==1){LC.setLed(1, j, i, !tabl[j][i]); z=k;k=k1;k1=z;LC.
setIntensity(0, 8*k);LC.setIntensity(1, 8*k);delay(500);LC.setIntensity(0,
8*k1);LC.setIntensity(1, 8*k1);delay(500);
 LC.setIntensity(0, 8*k);LC.setIntensity(1, 8*k);delay(500);}//включение
и соответственно выключение каждого светодиода контура картинки с
задержкой в 1,5 сек.

  }
 }
```

```
  LC.clearDisplay(0);//очищаем матрицу 1

}
 //
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Меняя «рисунок элементов массива» (рисунок 17) можно менять внешний вид индикатора песочных часов. Для управления часами желательно добавить кнопку и резистор, подключив их, например, к выводу 9 аналогично тому, как мы делали в схеме предыдущего эксперимента. Видоизменённый скетч дан ниже:

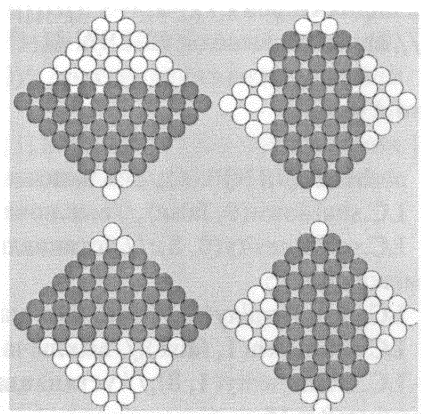

Рис. 17

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа для светодиодной матрицы 8*8 и max7219 песочные часы
//

#include «LedControl.h»//подключаем данную библиотеку
int i=0;int j=0;int k=1;int k1=0;int z=0;
int tabl[8][8]={0,0,1,1,1,1,1,1,
          0,1,1,1,1,1,1,1,
```

```
        1,1,1,1,1,1,1,0,
        1,1,1,1,1,1,0,0,
        1,1,1,1,1,0,0,0,
        1,1,1,1,0,0,0,0,
        1,1,1,0,0,0,0,0,
        1,1,0,0,0,0,0,0};// храним картинку в массиве

LedControl LC = LedControl(12, 11, 10, 2);// создаём объект класса для
2 индикаторов
// при этом выводы 12-DIN 11-CLK 10-CS //

void setup()
{
 pinMode(9,INPUT);//подключение управляющей кнопки
 LC.shutdown(0, false);//выключаем энергосберегающий режим
 LC.setIntensity(0, 8);// устанавливаем интенсивность в 8 единиц
между 0 и 15
 LC.clearDisplay(0);//очищаем матрицу 1
 LC.shutdown(1, false);//выключаем энергосберегающий режим
 LC.setIntensity(1, 8);// устанавливаем интенсивность в 8 единиц
между 0 и 15
 LC.clearDisplay(1);//очищаем матрицу 2
}

void loop()
{
 ////////// зажигаем верхнюю чашу ////////////////////
  for (i=7;i>=0;i--) //цикл в цикле для перебора элементов массива
 {
 for (j=7;j>=0;j--)
  {
   LC.setLed(1, j, i, tabl[j][i]);
  }
  }
  metka:
  if (digitalRead (9)==LOW){goto metka;}  //пока кнопка не нажата
```

```
  /////////////// «пересыпаем песок» в нижнюю чашу
/////////////////
 for (i=7;i>=0;i--) //цикл в цикле для перебора элементов массива
 {
 for (j=7;j>=0;j--)
  {

 LC.setLed(0, j, i, tabl[j][i]);
 if (tabl[j][i]==1){LC.setLed(1, j, i, !tabl[j][i]); z=k;k=k1;k1=z;LC.
setIntensity(0, 8*k);LC.setIntensity(1, 8*k);delay(500);LC.setIntensity(0,
8*k1);LC.setIntensity(1, 8*k1);delay(500);// посекундная смена яркости
свечения индикаторов
 LC.setIntensity(0, 8*k);LC.setIntensity(1, 8*k);delay(500);}//включение
и соответственно выключение каждого светодиода контура картинки с
задержкой в 1,5 сек.

  }
 }

   LC.clearDisplay(0);//очищаем матрицу 1

}
 //
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Работа скетча аналогична работе разобранного варианта с сердечком и в особых комментариях не нуждается.

Конструкция часов может быть произвольной. Фальш — панель изготовлена из отрезка прозрачного оргстекла. С обратной стороны лицевой части под скотч наклеена бумажная вырезка, имеющая пару окошек в форме чаш для песка. Всего задействовано 40 светодиодов с периодом переключения в 1,5 секунды.

На этом читатель закончим прохождение нашего нулевого уровня обучения программированию на примере Ардуино.

ГЛАВА 6
АРДУИНО-ПРАКТИЧЕСКОЕ ПРИЛОЖЕНИЕ

Эта глава своего рода послесловие к книге. Завершив начальный этап самообразования, спустя некоторое время, захотелось применить навыки в некоторых простых поделках, описания которых, позднее были реализованы в виде журнальных статей. Ниже приведены данные описания нескольких изделий и тексты управляющих скетчей (без пояснений к ним). Если содержание книги, как модно теперь говорить, «вошло в читателя», данная глава будет полезна как итог работы с ней при реализации собственных конструкций.

Светодиодный куб 4x4x4

В Интернет-магазинах по доступной цене можно приобрести модуль

управления светодиодной матрицей 8x8, собранный на микросхеме MAX7219 (**рис. 1**). Этот модуль можно использовать для совместной работы с Arduino. Такой модуль, используя всего три сигнальных вывода платы, позволяет управлять светодиодной матрицей из 64 (8x8) светодиодов.

Дополнительные функции по управлению такими модулями даёт библиотека LedControl, автором которой является Эбехард Фэйл. Совместное использование библиотеки и модулей расширяет возможности Arduino UNO. Библиотека поддерживает работу до восьми последовательно соединённых модулей, позволяя параллельно подключать несколько таких цепочек. Это широко используется любителями при построении различных информационных табло и бегущих строк. Однако не менее интересным является создание с помощью этого модуля и светодиодов трёхмерных анимационных эффектов. Об управлении с помощью модуля на MAX7219 светодиодной 3D-матрицей 4x4x4 и пойдёт речь в статье.

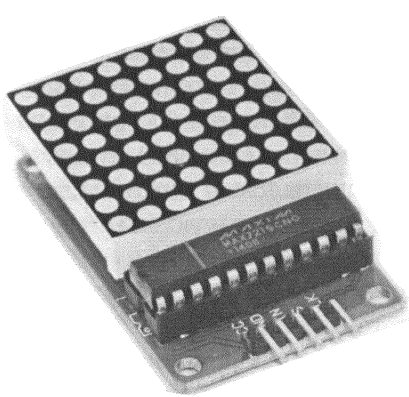

Рис. 1

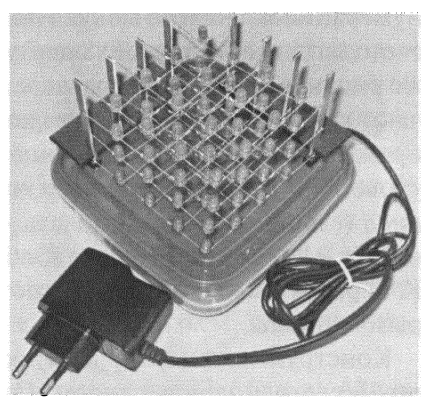

Рис. 2

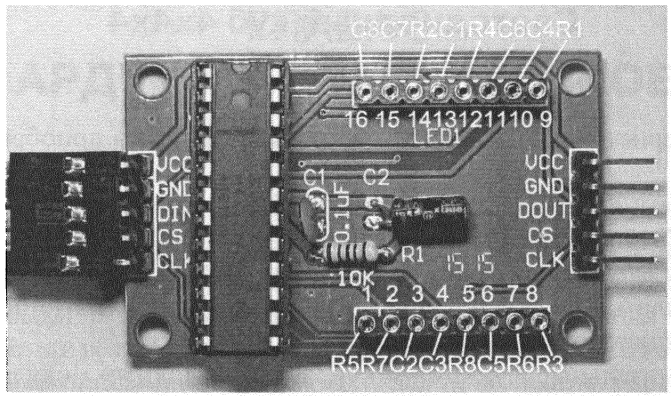

Рис. 3

На **рис. 2** изображена конструкция светодиодного автоматического переключателя световых эффектов. Автомат перебирает в цикле восемь программ переключения светодиодов. Использование Arduino позволяет гибко менять их число и содержание, открывая простор для творчества.

Изготовление устройства удобнее начать с самого куба и его подключения к модулю. Сначала с платы модуля надо удалить светодиодную матрицу, для этого её надо аккуратно поддеть отвёрткой. В результате, освободятся разъёмы для подключения матрицы. Условная нумерация контактов модуля показана на **рис. 3**. Именно через них на светодиодную матрицу 8x8 поступает питающее напряжение. На этом же рисунке указаны выводы светодиодного куба, к которым подключают контакты модуля. Через контакты C1—C8 высокий уровень напряжения поступает на столбцы матрицы, а через (R1—R8) — низкий уровень на строки. Например, чтобы включить левый нижний светодиод (см. рис. 1) нужно подать на вывод C1 логическую единицу, на вывод R1 — ноль. Выводы R образуют ряды, выводы C — столбцы. К первым выводам подключаются все катоды светодиодов, ко вторым — аноды.

Конструктивно куб (**рис. 4**) состоит из четырёх одинаковых слоёв по 16 светодиодов в каждом. Схема одного слоя показана на **рис. 5**. В авторском варианте применены индикаторные светодиоды красно-

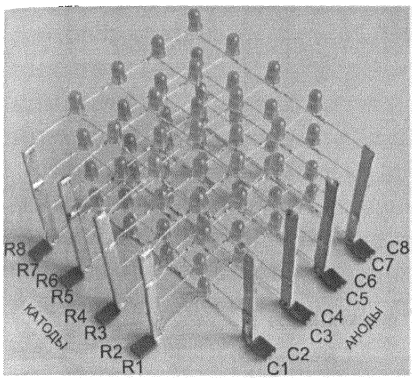

Рис. 4

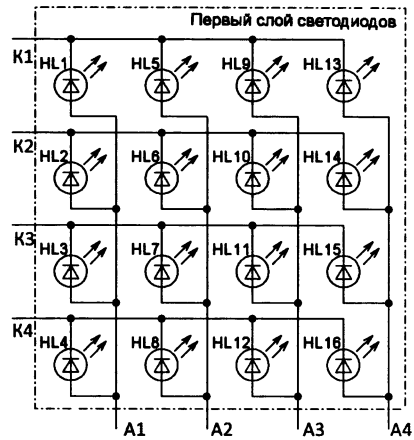

Рис. 5

го цвета свечения с диаметром корпуса 5 мм. Каждый слой монтируется с помощью пластины—шаблона. Она изготовлена из листовой пластмассы толщиной 2...3 мм или толстого картона, её размеры — 80x80 мм. В ней просверлены 16 отверстий диаметром 5 с шагом 20 мм. В отверстия вставляются светодиоды, аккуратно изгибают выводы (**рис. 6**) и производят их пайку в соответствии со схемой слоя. Удобнее производить соединения, например, в такой последовательности HL1, HL5, HL9, HL13. После последовательной сборки слоёв приступают к монтажу куба. Для этого потребуется восемь стоек из двустороннего фольгированного стеклотекстолита шириной 5 и длиной 60 мм каждая. Они будут обеспечивать механическое крепление и электрическое соединение слоёв в единый куб. Выводы катодов верхних двух слоёв (рис. 4) попарно припаивают к внешней стороне (по рис. 4) каждой стойки, нижних двух слоёв — к внутренней стороне. Аноды (выводы) первого и третьего слоя (отсчёт сверху) соединяют с внешней стороной стоек, второго и четвёртого слоя — с внутренней. Внизу, к каждой стойке припаивают двухконтактный разъём, для соединения с платой модуля в соответствии с рис. 3, рис. 4 и рис. 5.

Далее собирают устройство в соответствии с **рис. 7**. Модуль А2 соединяют с кубом, состоящим из четырёх слоёв А3—А6. Дополнитель-

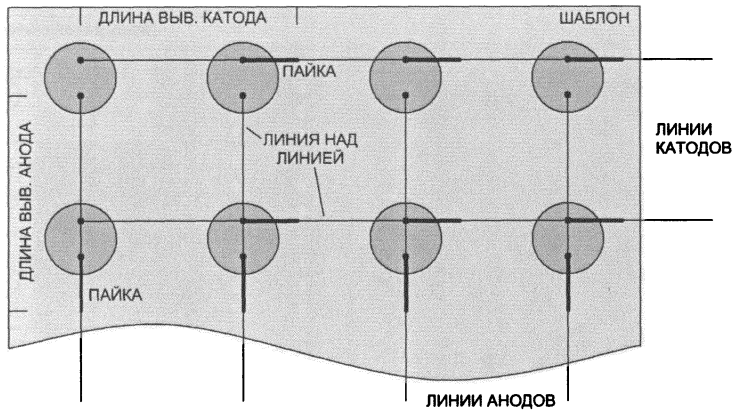
ДЛИНА ВЫВ. КАТОДА
ШАБЛОН
ПАЙКА
ЛИНИЯ НАД ЛИНИЕЙ
ЛИНИИ КАТОДОВ
ДЛИНА ВЫВ. АНОДА
ПАЙКА
ЛИНИИ АНОДОВ

Рис. 6

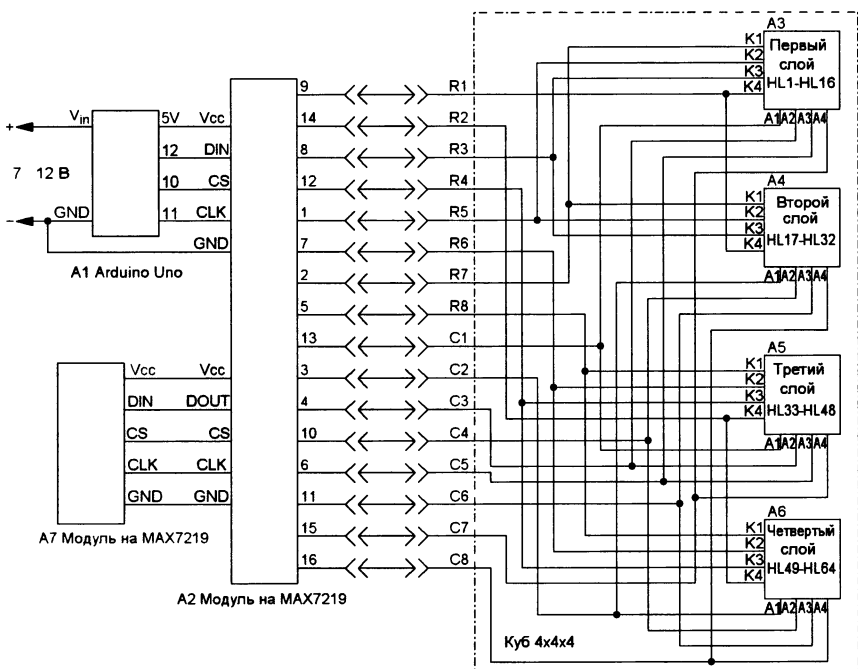
A3
K1
K2
K3
K4
Первый слой
HL1-HL16
A1 A2 A3 A4
A4
Второй слой
HL17-HL32
A5
Третий слой
HL33-HL48
A6
Четвертый слой
HL49-HL64
Vin
+
-
7 12 В
GND
5V Vcc
12 DIN
10 CS
11 CLK
GND
A1 Arduino Uno
Vcc Vcc
DIN DOUT
CS CS
CLK CLK
GND GND
A7 Модуль на MAX7219
9 R1
14 R2
8 R3
12 R4
1 R5
7 R6
2 R7
5 R8
13 C1
3 C2
4 C3
10 C4
6 C5
11 C6
15 C7
16 C8
A2 Модуль на MAX7219
Куб 4x4x4

Рис. 7

ный модуль A7 (модуль на MAX7219 со светодиодной матрицей 8x8) позволяет визуально контролировать при самостоятельном программировании соответствие двухмерной и трёхмерной картинки свечения светодиодов в кубе и матрице. После окончания программирования этот модуль удаляют. Питают конструкцию при настройке от компьютера, при эксплуатации — от внешнего источника напряжением 7...12 В при токе до 500 мА.

Библиотека LedControl не встроена в Arduino IDE, поэтому её нужно найти и установить на свой компьютер. Для этого в поисковую строку записывают "Библиотека LedControl, ZIP архив скачать" и устанавливают её после скачивания из Arduino IDE (**рис. 8**). Её название должно появиться внизу выпадающего списка.

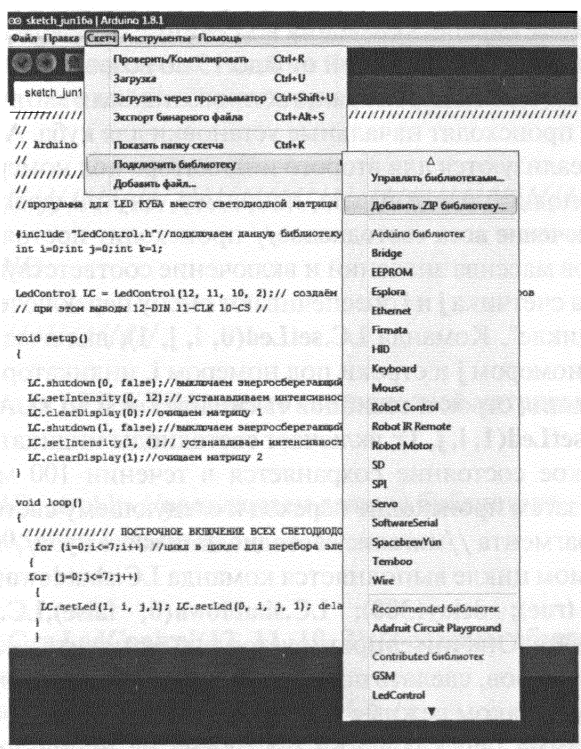

Рис. 8

Далее рассмотрим основные моменты использования библиотеки и функции применительно к скетчу для светодиодного куба. Строка **#include «LedControl.h»** в начале скетча указывает на необходимость использования данной библиотеки. Строка **LedControl LC = LedControl(12, 11, 10, 2);** создаёт в программе объект класса для двух индикаторов — куба и плоской матрицы. Аргументы в скобках задают номера выходов платы и соответственно порядок подключения входов модуля. Первый — DIN, второй —CLKC, третий — CS. Четвёртый аргумент указывает число используемых индикаторов (в нашем примере их два).

Затем следует команда **LC.shutdown(0, false);**, которая выключает индикатор под номером 0 (нумерация начинается с 0 и заканчивается цифрой 7) из режима экономии энергии. Команда **LC.setIntensity(0, 12);** устанавливает яркость свечения в 12 единиц (яркость условно разбита на 16 уровней с нумерацией от 0 до 15 по возрастанию). Команда **LC.clearDisplay(0);** очищает экран, гасит все пиксели матрицы под номером 0. Так происходят начальные установки для куба. Аналогичные процедуры реализуются для второго индикатора под номером 1.

Далее в основной части цикла, например, внутри фрагмента **//построчное включение всех светодиодов//** происходит постоянный перебор элементов массива значений и включение соответствующих точек матрицы. Два счётчика **j** и **i** обеспечивают смену переключения по схеме "цикл в цикле". Команда **LC.setLed(0, i, j, 1);** включает светодиод столбца под номером **j** и строки под номером **i**, индикатора под номером 0 (нумерация строк и столбцов также идёт от 0 до 7). Аналогично, команда **LC.setLed(1, i, j, 1);** включает светодиоды индикатора под номером 1. Такое состояние сохраняется в течении 100 мс (команда **delay(100);**), затем происходит переход к следующему светодиоду.

Внутри фрагмента **//мигание светодиодами всего куба//** в пятикратно повторяемом цикле выполняется команда **LC.shutdown(0, true);LC.shutdown(1, true); delay(300); LC.shutdown(0, false);LC.shutdown(1, false); delay(300);**. Она интерпретируется так: включить экономичный режим индикаторов, сделать паузу, выключить экономичный режим, сделать паузу. В таком режиме все светодиоды куба будут мигать.

Вся программа переключений построена на использовании трёх основных функций **LC.setLed(); LC.shutdown(); LC.clearDisplay();**.

Это — включение/выключение заданного светодиода, включение/выключение индикатора с сохранением данных и очистка экрана с выключением всех светодиодов с потерей данных по их предыдущему состоянию.

После макетирования и программирования, приступают к сборке устройства. В пластмассовом контейнере подходящих размеров (см. рис. 2) размещают платы Arduino и модуля MAX7219. В крышке вдоль стоек прорезают щели и пропускают в них соединительные провода к контактным разъёмам рядов и столбцов куба. Сверху щели закрывают пластмассовыми съёмными пластинами с крепёжными прорезями под стойки. Сам куб приклеивают к поверхности крышки через пластмассовые брусочки-переходники размерами 5x5x10 мм, приклеенные к основаниям 4 крайних стоек. В боковой поверхности контейнера сверлят отверстие для кабеля источника питания.

Содержательная часть скетча дана ниже. На FTP- сервере журнала РАДИО можно его найти в приложениях к публикациям за 2017 год в номере 11.

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа для LED КУБА вместо светодиодной матрицы 8*8, и
max7219 //

#include «LedControl.h»//подключаем данную библиотеку
int i=0;int j=0;int k=1;int n=1;


LedControl LC = LedControl(12, 11, 10, 2);// создаём объект класса для
2 индикаторов
// при этом выводы 12-DIN 11-CLK 10-CS //

void setup()
```

```
{

 LC.shutdown(0, false);//выключаем энергосберегающий режим
 LC.setIntensity(0, 12);// устанавливаем интенсивность в 12 единиц
между 0 и 15
 LC.clearDisplay(0);//очищаем матрицу 1
 LC.shutdown(1, false);//выключаем энергосберегающий режим
 LC.setIntensity(1, 4);// устанавливаем интенсивность в 4 единиц
между 0 и 15
 LC.clearDisplay(1);//очищаем матрицу 2
}

void loop()
{
 /////////////// ПОСТРОЧНОЕ ВКЛЮЧЕНИЕ ВСЕХ СВЕТОДИО-
ДОВ ///////////////////
  for (i=0;i<=7;i++) //цикл в цикле для перебора элементов матрицы
  {
  for (j=0;j<=7;j++)
   {
    LC.setLed(1, i, j,1); LC.setLed(0, i, j, 1); delay(100);
   }
   }
   /////////////// МИГАНИЕ СВЕТОДИОДАМИ ВСЕГО КУБА
//////////////////////
  for (j=0;j<=4;j++)
   {
   LC.shutdown(0, true);LC.shutdown(1, true); delay(300); LC.shutdown(0,
false);LC.shutdown(1, false); delay(300);
   }
   /////////////// ПОСТРОЧНОЕ РЕВЕРС-ВЫКЛЮЧЕНИЕ ВСЕХ
СВЕТОДИОДОВ //////////
    for (i=7;i>=0;i--) //цикл в цикле для перебора элементов матрицы
  {
  for (j=7;j>=0;j--)
   {
```

```
  LC.setLed(1, i, j,0); LC.setLed(0, i, j, 0); delay(50);
 }
 }
 /////////////// ПОСЛОЙНОЕ ВКЛЮЧЕНИЕ И ВЫКЛЮЧЕ-
НИЕ ВСЕХ СВЕТОДИОДОВ ////////////////////
 for (i=0;i<=7;i++) //цикл в цикле для перебора элементов матрицы
 {
 for (j=0;j<=7;j++)
 {
  LC.setLed(1, i, j,1); LC.setLed(0, i, j, 1);
  if (i==1&j==7){delay(500);}
  if (i==3&j==7){delay(500);}
  if (i==5&j==7){delay(500);}
  if (i==7&j==7){delay(500);}

 }
 }
 for (i=0;i<=7;i++) //цикл в цикле для перебора элементов матрицы
 {
 for (j=0;j<=7;j++)
 {
  LC.setLed(1, i, j,0); LC.setLed(0, i, j, 0);
  if (i==1&j==7){delay(500);}
  if (i==3&j==7){delay(500);}
  if (i==5&j==7){delay(500);}
  if (i==7&j==7){delay(500);}

 }
 }
 /////////////// ПОСЛОЙНОЕ ВКЛЮЧЕНИЕ И ВЫКЛЮЧЕНИЕ
С ПАУЗОЙ СВЕТОДИОДОВ СЛОЁВ ////////////////////
 for (k=1;k<=2;k++)//2 кратное повторение перебора
 {
 for (i=0;i<=7;i++) //цикл в цикле для перебора элементов матрицы
 {
 for (j=0;j<=7;j++)
```

```
  {
   LC.setLed(1, i, j,1); LC.setLed(0, i, j, 1);
   if (i==1&j==7){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==3&j==7){delay(200);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==5&j==7){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==7&j==7){delay(400);LC.clearDisplay(0); LC.clearDisplay(1);}

  }
  }
  }
  /////////////// ПОРЯДНОЕ ВКЛЮЧЕНИЕ И ВЫКЛЮЧЕНИЕ С
ПАУЗОЙ СВЕТОДИОДОВ СЛОЯ ///////////////////
  for (k=1;k<=2;k++)//2 кратное повторение перебора
  {
  for (i=0;i<=7;i++) //цикл в цикле для перебора элементов матрицы
 {
 for (j=0;j<=7;j=j+2)
  {
   LC.setLed(1, i, j,1); LC.setLed(0, i, j, 1);
   if (i==0&j==6){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==1&j==6){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==2&j==6){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==3&j==6){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==4&j==6){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==5&j==6){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==6&j==6){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==7&j==6){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}

  }
  for (j=1;j<=7;j=j+2)
  {
   LC.setLed(1, i, j,1); LC.setLed(0, i, j, 1);
   if (i==0&j==7){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==1&j==7){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==2&j==7){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
   if (i==3&j==7){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
```

```
  if (i==4&j==7){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==5&j==7){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==6&j==7){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==7&j==7){delay(300);LC.clearDisplay(0); LC.clearDisplay(1);}

 }
 }
 }
 /////////////// ПОСТОЛБЦОВОЕ И ПОРЯДНОЕ ВКЛЮЧЕНИЕ И ВЫКЛЮЧЕНИЕ С ПАУЗОЙ СВЕТОДИОДОВ СЛОЁВ
/////////////////
 for (k=1;k<=2;k++)//2 кратное повторение перебора
 {
 for (i=0;i<=7;i++) //цикл в цикле для перебора элементов матрицы
 {
 for (j=0;j<=7;j=j+2)
 {
  LC.setLed(1, j, i,1); LC.setLed(0, j, i, 1);LC.setLed(1, i, j,1); LC.
setLed(0, i, j, 1);
  if (i==0&j==6){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==1&j==6){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==2&j==6){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==3&j==6){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==4&j==6){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==5&j==6){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==6&j==6){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==7&j==6){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}

 }
 for (j=1;j<=7;j=j+2)
 {
  LC.setLed(1, j, i,1); LC.setLed(0, j, i, 1);LC.setLed(1, i, j,1); LC.
setLed(0, i, j, 1);
  if (i==0&j==7){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==1&j==7){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==2&j==7){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
```

```
  if (i==3&j==7){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==4&j==7){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==5&j==7){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==6&j==7){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}
  if (i==7&j==7){delay(100);LC.clearDisplay(0); LC.clearDisplay(1);}

 }
 }
 }
  for (k=1;k<=2;k++)//2 кратное повторение перебора
  {
 for (i=0;i<=7;i=i+2) //цикл в цикле для перебора элементов матрицы
 {
for (j=0;j<=7;j=j+2)
 {
  LC.setLed(1, i, j,1); LC.setLed(0, i, j, 1);
  LC.setLed(1, i, j+1,1); LC.setLed(0, i, j+1, 1);
  LC.setLed(1, i+1, j,1); LC.setLed(0, i+1, j, 1);
  LC.setLed(1, i+1, j+1,1); LC.setLed(0, i+1, j+1, 1);
  delay(250);LC.clearDisplay(0); LC.clearDisplay(1);
 }
}
 }
 /////////////// ПСЕВДОСЛУЧАЙНОЕ ВКЛЮЧЕНИЕ/ВЫКЛЮ-
ЧЕНИЕ 4 ПАР (КУБИК) СВЕТОДИОДОВ ////////////////////
 for (k=1;k<=20;k++)//20 кратное повторение перебора
 {
i=random(0,6);j=random(0,5);
  LC.setLed(1, i, j,1); LC.setLed(0, i, j, 1);
  LC.setLed(1, i, j+1,1); LC.setLed(0, i, j+1, 1);
  LC.setLed(1, i, j+2,1); LC.setLed(0, i, j+2, 1);
  LC.setLed(1, i, j+3,1); LC.setLed(0, i, j+3, 1);
  LC.setLed(1, i+2, j,1); LC.setLed(0, i+2, j, 1);
  LC.setLed(1, i+2, j+1,1); LC.setLed(0, i+2, j+1, 1);
  LC.setLed(1, i+2, j+2,1); LC.setLed(0, i+2, j+2, 1);
  LC.setLed(1, i+2, j+3,1); LC.setLed(0, i+2, j+3, 1);
```

```
  delay(random (100,250));
  LC.clearDisplay(0);//очищаем матрицу 1
   LC.clearDisplay(1);//очищаем матрицу 2
  }
  /////////////// КОНТУР КУБА — ПОЯВЛЕНИЕ/ИЗЧЕЗНОВЕ-
НИЕ ////////////////////
  int ta
bl1[32]={0,0,0,0,2,4,6,6,6,6,4,2,0,1,1,3,5,7,7,7,7,5,3,1,1,1,6,7,0,1,6,7}
;
  int ta
bl2[32]={0,2,4,6,6,6,6,4,2,0,0,0,1,0,1,1,1,1,3,5,7,7,7,7,5,3,7,6,7,6,1,0}
;
   for (k=0;k<=31;k++)
    {
     LC.setLed(1, tabl1[k],tabl2[k] ,1); LC.setLed(0, tabl1[k],tabl2[k]
,1);delay(250);

    }
  delay(2000);
   for (k=31;k>=0;k--)
    {
     LC.setLed(1, tabl1[k],tabl2[k] ,0); LC.setLed(0, tabl1[k],tabl2[k]
,0);delay(250);

    }
  delay(2000);
  ////////////////////////////ПЕРЕХОД КАПЕЛЬ СВЕТА С ПЛО-
СКОСТИ НА ПЛОСКОСТЬ//////////////////////////////////
  for (k=1;k<=2;k++)//2 кратное повторение перебора
  {
  for (i=0;i<=1;i++) //цикл в цикле для перебора элементов матрицы
 {
 for (j=0;j<=7;j++)
  {
   LC.setLed(1, i,j ,1); LC.setLed(0, i,j ,1);
  }
```

```
  }
   delay(200);
   for (i=0;i<=1;i++) //цикл в цикле для перебора элементов матрицы
  {
  for (j=0;j<=7;j++)
   {
   LC.setLed(1, i,j ,0); LC.setLed(0, i,j ,0);delay(50);
   for (n=2;n<=6;n=n+2){LC.setLed(1, i+n,j ,1); LC.setLed(0, i+n,j
,1);delay(50);LC.setLed(1, i+n,j ,0); LC.setLed(0, i+n,j ,0);}
   LC.setLed(1, i+6,j ,1); LC.setLed(0, i+6,j ,1);delay(50);
   }
   }
  LC.clearDisplay(0);LC.clearDisplay(1);
  }
   /////////////// ПАДАЮЩИЕ КАПЛИ СВЕТА ////////////////////
    for (k=0;k<=15;k++)
     {
     i=random(0,4);j=random(0,4); i=2*i;j=2*j;
     LC.setLed(1, i,j ,1); LC.setLed(0, i,j ,1);delay(200); LC.
clearDisplay(0);LC.clearDisplay(1);
     LC.setLed(1, i,j+1 ,1); LC.setLed(0, i,j+1 ,1);delay(100); LC.
clearDisplay(0);LC.clearDisplay(1);
     LC.setLed(1, i+1,j ,1); LC.setLed(0, i+1,j ,1);delay(75); LC.
clearDisplay(0);LC.clearDisplay(1);
      LC.setLed(1, i+1,j+1 ,1); LC.setLed(0, i+1,j+1 ,1);delay(50); LC.
clearDisplay(0);LC.clearDisplay(1);
     }
      for (k=0;k<=20;k++)
    {
     i=random(0,4);j=random(0,4); i=2*i;j=2*j;
     LC.setLed(1, i,j ,1); LC.setLed(0, i,j ,1);delay(200);
     LC.setLed(1, i,j+1 ,1); LC.setLed(0, i,j+1 ,1);delay(100);
     LC.setLed(1, i+1,j ,1); LC.setLed(0, i+1,j ,1);delay(75);
      LC.setLed(1, i+1,j+1 ,1); LC.setLed(0, i+1,j+1 ,1);delay(50);n++;
      if (n>=5){delay(500);LC.clearDisplay(0);LC.clearDisplay(1);n=0;}
     }
```

```
  ////////////////////////////// ПОВОРОТ ПЛОСКОСТЕЙ //////
//////////////////////////
    for (k=1;k<=8;k++)//8 кратное повторение перебора
  {
  for (i=0;i<=1;i++) //цикл в цикле для перебора элементов матрицы
  {
  for (j=0;j<=7;j++)
   {
    LC.setLed(1, i,j ,1); LC.setLed(0, i,j ,1);
   }
   }
  delay(150);LC.clearDisplay(0);LC.clearDisplay(1);
  for (i=0;i<=6;i=i+2) //цикл для перебора элементов матрицы
   {
   LC.setLed(1, i,i ,1); LC.setLed(0, i,i ,1);LC.setLed(1, i+1,i ,1); LC.
setLed(0, i+1,i ,1);
   LC.setLed(1, i,i+1 ,1); LC.setLed(0, i,i+1 ,1);LC.setLed(1, i+1,i+1 ,1);
LC.setLed(0, i+1,i+1 ,1);
    }
  delay(150);LC.clearDisplay(0);LC.clearDisplay(1);
  for (i=0;i<=1;i++) //цикл в цикле для перебора элементов матрицы
  {
  for (j=0;j<=7;j++)
   {
    LC.setLed(1, j,i ,1); LC.setLed(0, j,i ,1);
   }
   }
  delay(150);LC.clearDisplay(0);LC.clearDisplay(1);
  for (i=0;i<=6;i=i+2) //цикл  для перебора элементов матрицы
   {
   LC.setLed(1, i,7-i ,1); LC.setLed(0, i,7-i ,1);LC.setLed(1, i+1,7-i ,1);
LC.setLed(0, i+1,7-i ,1);
   LC.setLed(1, i+1,6-i ,1); LC.setLed(0, i+1,6-i ,1);LC.setLed(1, i,6-i
,1); LC.setLed(0, i,6-i ,1);
    }
  delay(150);LC.clearDisplay(0);LC.clearDisplay(1);
```

```
 for (i=6;i<=7;i++) //цикл в цикле для перебора элементов матрицы
 {
 for (j=0;j<=7;j++)
  {
   LC.setLed(1, i,j ,1); LC.setLed(0, i,j ,1);
  }
  }
  delay(150);LC.clearDisplay(0);LC.clearDisplay(1);
   for (i=0;i<=6;i=i+2) //цикл для перебора элементов матрицы
   {
   LC.setLed(1, i,i ,1); LC.setLed(0, i,i ,1);LC.setLed(1, i+1,i ,1); LC.
setLed(0, i+1,i ,1);
   LC.setLed(1, i,i+1 ,1); LC.setLed(0, i,i+1 ,1);LC.setLed(1, i+1,i+1 ,1);
LC.setLed(0, i+1,i+1 ,1);
    }
  delay(150);LC.clearDisplay(0);LC.clearDisplay(1);
   for (i=6;i<=7;i++) //цикл в цикле для перебора элементов матрицы
 {
 for (j=0;j<=7;j++)
  {
   LC.setLed(1, j,i ,1); LC.setLed(0, j,i ,1);
  }
  }
  delay(150);LC.clearDisplay(0);LC.clearDisplay(1);
  for (i=0;i<=6;i=i+2) //цикл  для перебора элементов матрицы
   {
   LC.setLed(1, i,7-i ,1); LC.setLed(0, i,7-i ,1);LC.setLed(1, i+1,7-i ,1);
LC.setLed(0, i+1,7-i ,1);
   LC.setLed(1, i+1,6-i ,1); LC.setLed(0, i+1,6-i ,1);LC.setLed(1, i,6-i
,1); LC.setLed(0, i,6-i ,1);
    }
  delay(150);LC.clearDisplay(0);LC.clearDisplay(1);
  }
  ////////////////////// ОЧИСТКА МАТРИЦ ПЕРЕД НОВЫМ
ПОВТОРОМ ПОЛНОГО ЦИКЛА ////////////////
   LC.clearDisplay(0);//очищаем матрицу 1
```

```
    LC.clearDisplay(1);//очищаем матрицу 2

}
 //
// Конец /
//
////////////////////////////////////////////////////////////////////////
```

Тренажёр-игрушка алфавит «Павлин»

Тренажёр, описание конструкции которого приведено далее, позволяет закрепить первичные навыки чтения в игровой форме. Её основой является механическое табло (**рис. 9**). По командам программы металлическая стрелка указывает последовательность букв, из которых читающий составляет слова или короткие фразы.

Схема устройства показана на **рис. 10**. После подачи питающего напряжения кратковременно нажимая на кнопку SB2 "Выбор" мы случайным образом выбираем заготовленную фразу из заранее созданного массива. Исполняющим устройством игрушки является сервопривод SG90, управляет которым плата Arduino UNO. Для обеспечения устойчивой работы устройства применено раздельное питание платы и сервопривода. Для питания Arduino UNO применена батарея типоразмера 6F22 ("Крона", "Корунд" или аналогичная с выходным током около 40 мА). Сервопривод подключён к батарее из четырёх элементов типоразмера АА.

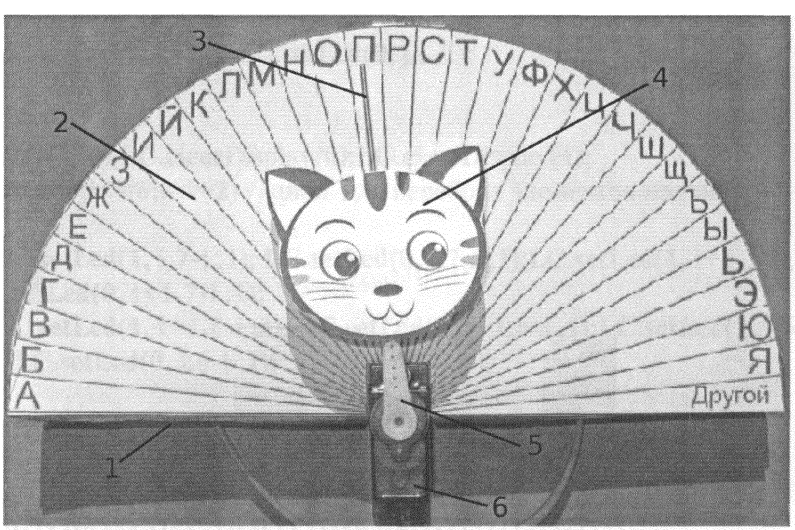

Рис. 9

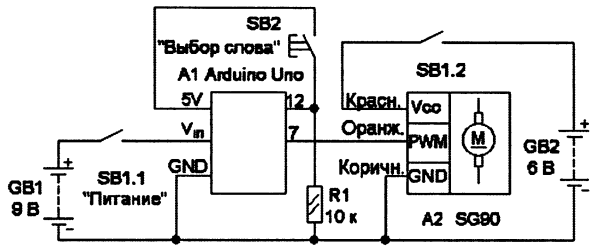

Рис. 10

Рассмотрим работу программы (скетча). В её начале находится таблица соответствия угла поворота стрелки и длительности (в микросекундах) управляющего позиционированием импульса. Его возможные значения заключены между 540 (пробел или "другой") и 2235 мкс (буква "А"). Каждой букве соответствует свой угол поворота коромысла сервопривода в интервале от 0 до 180 градусов. Длительность пауз между импульсами не критична и может изменяться от 15 до 25 мс (15000—25000 мкс). В данной программе длительность составляет 20000 мкс, за вычетом длительности управляющего импульса. Под буквы и пробел выделено 33 угловых сектора на бумажной фальшь панели алфавита.

Ниже, в программе находится массив данных (имя **tabl**) в котором можно хранить короткие слова и словосочетания в форме длительностей импульсов. Массив содержит сто элементов типа int, разделённых в десять строк по десять символов в каждом. Первая строка, например, содержит слово "привет" и четыре пробела:

int tabl[10][10]={540,1410,1355,1795,2130,1960,1245,540,540,540,

Далее вводятся три переменные **x, y** и **i** типа int. Первая указывает строки массива, вторая перебирает в цикле его столбцы. Значение i определяет длительность удержания угла поворота сервопривода, то есть длительность индикации текущей буквы слова. В следующих двух строках определяется вывод 7 платы как выход — **pinMode(7,OUTPUT);**, и его начальное состояние как логический ноль — **digitalWrite(7,LOW);**. Затем определяется вывод 12 как вход — **pinMode(12,INPUT);**.

В основном цикле скетча void loop() производится проверка состояния контактов кнопки SB2, случайный выбор словосочетания при её замыкании, последовательная индикация его букв в заданном темпе и ожидание повторного нажатия кнопки. При условии, что на выводе 12 присутствует высокий логический уровень — **if (digitalRead (12)==HIGH)** переменной **x** присваивается случайное значение от 0 до 9, переменной **y** — значение 0 и выдерживается пауза в 50 мс — **{x=random (0,10);y=0;delay(50);}**. Таким образом, программа готова считывать из массива выпавшую фразу. Если чтение уже произошло, строка **if(y>=10){goto metka;}** возвращает её выполнение на строку **metka:** и программа ждёт повторного нажатия кнопки для воспроизведения на табло другого словосочетания.

В заключительной части скетча в цикле из 75 повторений на выходе 7 формируются управляющие импульсы с требуемой для индикации конкретной буквы длительностью и выполняется пауза. Их суммарная продолжительность составляет 20 мс. За счёт 75 повторов задаётся продолжительность индикации одной буквы (20·75 мс). Далее переменная **y** увеличивает своё значение на единицу и происходит переход к чтению из массива следующую буквы.

В устройстве можно применить постоянный резистор любого типа выключатель питания — переключатель любого типа на два положения и два направления, кнопка SB2 — с самовозвратом любого типа, например КМ1-1.

Основание шкалы (см. рис. 9) изготовлено из листа пластмассы 1 толщиной 2...3 мм в форме полукруга радиусом 7...8 см. В центре имеется прямоугольный вырез для сервопривода 6. Его крепят с помощью миниатюрного винта-самореза входящего в комплект. С обратной стороны основания, напротив центральной части алфавита приклеена прямоугольная пластмассовая пластинка. Она и дно сервопривода образуют опорные точки шкалы для её размещения на лицевой панели корпуса с помощью клея. Шкала-алфавит 2 — бумажная, напечатана с помощью компьютера на принтере. Её крепят к основанию отрезками липкой ленты. Стрелка-указатель 3 сделана из отрезка стальной проволоки диаметром 1...2 и длиной 50...60 мм. Она крепится с помощью "секундного" клея к внутренней поверхности коромысла 5 сервопривода 6. К стрелке, в свою очередь, прикреплена

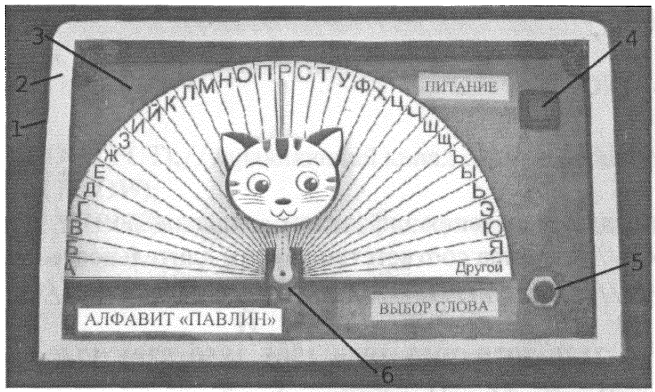

Рис. 11

декоративная картинка 4 на картонном основании. Для её фиксации можно применить термоклей.

Все узлы устройства (**рис. 11**) размещены внутри пластмассового корпуса-футляра 1 подходящего размера. Лицевая стенка 3 изготовлена из отрезка ДВП и утоплена внутрь корпуса на высоту сервопривода 6. На ней расположено табло, кнопочный выключатель питания 4 на две группы контактов и кнопка SB2 5. Напротив декоративной картинки в стенке 3 просверлено отверстие, в которое пропущен шлейф проводов сервопривода. Информационные надписи напечатаны на бумаге и зафиксированы на поверхностях с помощью полосок прозрачной липкой ленты. Во время хранения футляр закрывают от пыли прозрачной тонкой пластиковой крышкой 2. Она согнута из листовых фрагментов от упаковки хозтоваров. Внутри по её краю наклеен бумажный кант.

Налаживание сводится к регулировке углов поворота качалки-коромысла сервопривода до совпадения расположения конца стрелки и нужной буквы алфавита. Для этого вместо фраз в массив данных следует записать подряд все буквы алфавита и, варьируя числовые значения длительностей импульсов, подобрать требуемый для каждой буквы угол поворота стрелки. Это удобнее сделать на макете, до окончательной сборки тренажёра.

Ниже дан текст управляющего скетча.

```
////////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа для управления алфавитом — тренажёром //
/////////////// таблица соответствия букв и длительностей импульса
//////////////////////////////////////////////////////
///А Б В Г Д Е Ж З И Й
//2235 2180 2130 2070 2015 1960 1905 1850 1795 1740
 //К Л М Н О П Р С Т У
///1685 1630 1575 1520 1465 1410 1355 1300 1245 1190
 //Ф Х Ц Ч Ш Щ Ъ Ы Ь Э
///1135 1080 1025 970 915 860 805 750 710 675
 // Ю Я ДРУГОЙ (ПРОБЕЛ)
///620 580 540
////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////

int tabl[10][10]={540,1410,1355,1795,2130,1960,1245,540,540,540,//
привет
        540,1575,2235,1575,2235,540,1795,1355,2235,540,//мама
Ира
        1410,2235,1410,2235,540,2015,1795,1575,2235,540,//папа
Дима
        540,540,1300,1465,1630,1520,1025,1960,540,540,//солнце
        1190,1245,1355,1465,540,1520,1465,970,710,540,//утро ночь
        540,1575,2235,2070,2235,1850,1795,1520,540,540,//магазин
        540,540,2015,1355,1190,1905,2180,2235,540,540,//дружба
        540,540,1685,1465,1245,1465,1410,1960,1300,540,//котопес
        540,1575,750,915,710,540,1300,750,1355,540,//мышь сыр
        540,540,1685,1465,1355,1465,2130,2235,540,540};//корова
//ДАННЫЕ СЛОВ
int x=0;int y=0;
```

```
int i=1;//счётчик,max значение определяет длительность удержания
угла поворота
void setup()
{
 pinMode(7,OUTPUT);//управляющий канал сервопривода
 digitalWrite(7,LOW);
 pinMode(12,INPUT);  //кнопка выв.12 вход — перебор словосочета-
ний/
}

void loop()
{
   metka:
   if (digitalRead (12)==HIGH)  //если кнопка нажата ...
   {x=random (0,10);y=0;delay(50);} //случайный выбор словосочетания
   if(y>=10){goto metka;}//условие завершения чтения словосочетания
до нового нажатия кнопки
   for (i=1;i<=75;i=i+1)
     {
      digitalWrite(7, HIGH);
      delayMicroseconds(tabl[x][y]);//длительность импульса
      digitalWrite(7, LOW);
      delayMicroseconds(10000-tabl[x][y]);//длительность паузы
      delayMicroseconds(10000);
     }
    y++;//переход на следующую букву

}
//
// Конец /
//
////////////////////////////////////////////////////////////////
```

Танцующий человечек Next

Схема устройства показана на **рис. 12**. Оно содержит микрофонный усилитель на транзисторе VT1, электронный ключ на транзисторе VT2, плату Arduino UNO (A1), модуль на MAX7219 (A2) и самодельный светодиодный индикатор A3. Схема индикатора показана на **рис. 13**. Светодиоды соединены в матрицу 5x7, чтобы их нумерация была более удобной, был пронумерован и светодиод HL12, которого на самом деле в устройстве нет. Катоды образуют семь вертикальных линий, аноды — пять горизонтальных. Расположение и форма светодиодов в индикаторе показаны на **рис.14** . Они образуют контуры трех человечков с возможностью индикации 16 поз в крайних и 32 в центральном контуре.

Усиленный транзистором VT1 сигнал микрофона поступает на регулятор чувствительности — переменный резистор R4. Когда напряжения на базе транзистора окажется достаточно для его открывания, появится коллекторный ток, напряжение на нём уменьшится. В результате включает индикаторный светодиод HL1, который помогает визуально контролировать работу устройства. Изменение напряжения на коллекторе транзистора VT2 "отслеживает" аналоговый вход

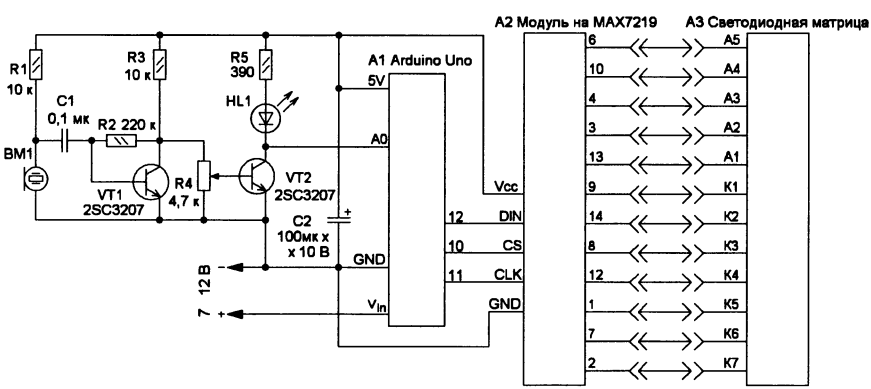

Рис. 12

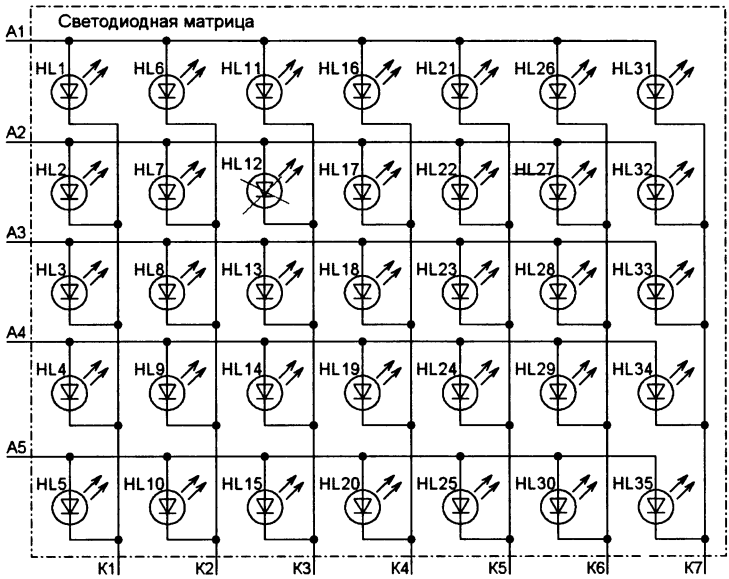

Рис. 13

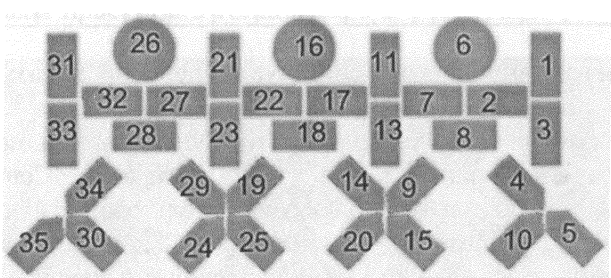

Рис. 14

A0 платы Arduino UNO. Если напряжение резко уменьшается (при достаточно громкой музыке), происходит переключение позы человечка.

Кратко рассмотрим работу программы. Она выполняет следующие задачи: определение уровня напряжения входного сигнала (на выво-

де A0), установку яркости индикатора по полученным данным с возможной сменой значения каждые 30 мс, смену позы человечка по временным интервалам, зависящим от громкости звукового сигнала (120...210 мс) и псевдослучайный выбор части исполняемого танца.

Вначале программы находится двухмерный массив-библиотека, в котором хранятся координаты светодиодов, включаемых в каждом такте смены позы человечка. Каждая строка это контур позы, состоящей из десяти светящихся светодиодов: двадцать цифр, координат. Первые двенадцать задают контур тела, последние восемь — контуры рук и ног. Соответствие светодиодов в индикаторе и их координат для команды **LC.setLed()** поясняет **таблица** и рис. 14.

№ светодиода	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	18	19
i	0	0	0	0	0	1	1	1	1	1	2	2	2	2	3	3	3	3
j	0	1	2	3	4	0	1	2	3	4	0	2	3	4	0	1	2	3

№ светодиода	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
i	3	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
j	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4

В данном варианте скетча используется перебор 24 возможных поз человечка.

Далее рассмотрим ключевые моменты использования библиотеки и функции применительно к скетчу. Строка **#include «LedControl.h»** в начале скетча указывает на необходимость использования данной библиотеки. Строка **LedControl LC = LedControl(12, 11, 10, 1);** создаёт в программе объект класса для одного индикатора. Аргументы в скобках задают номера выходов платы и соответственно порядок подключения к ним входов модуля на MAX721. Первый — вход DIN, второй — вход CLKC, третий — вход CS. Четвёртый аргумент указывает число используемых индикаторов (в нашем примере он один). Таким образом, можно использовать на три выводы платы Arduino UNO до восьми модулей.

Затем идут команды **LC.shutdown(0, false);**, которая выключает индикатор под номером 0 (нумерация начинается с 0 и заканчивается 7)

из режима экономии энергии, **LC.setIntensity(0, 12);** — устанавливает яркость свечения в 12 единиц (условно яркость разбита на 16 уровней с нумерацией от 0 до 15 по возрастанию). Команда **LC.clearDisplay(0);** очищает экран, гасит все пиксели матрицы под номером 0. Так происходят начальные установки для индикатора.

В основной части программы через каждые 30 мс (команда **delay(30);**) переменной x присваивается значение, полученное с вывода A0 АЦП — **x=analogRead(A0);**. Для полученных значений от 150 до 650 по условию **if()** назначается уровень яркости индикатора **LC.setIntensity(0, 10);** (например 10) и нарастание значение счётчика j. При превышении его значении 14 происходит смена позы человечка — последовательность команд **LC.clearDisplay(0); for (n=0;n<=18;n=n+2) {LC.setLed(0, tabl[k][n] ,tabl[k][n+1],1);}** очищают индикатор и включают светодиоды по координатам текущей строки массива. Счётчик i при достижении значения в пять по условию **if (i==5)** производит случайный выбор строки в массиве, обеспечивая относительное разнообразие движений человечка.

Нумерация выводов модуля на MAX7219 показана на **рис. 15**. Чертёж для установки и монтажа светодиодов показан на **рис. 16**. Далее о конструкции индикатора, которую поясняет **рис. 17**. Её основа — пластмассовая пластина 5 толщиной 3 и размерами 50x110 мм вы-

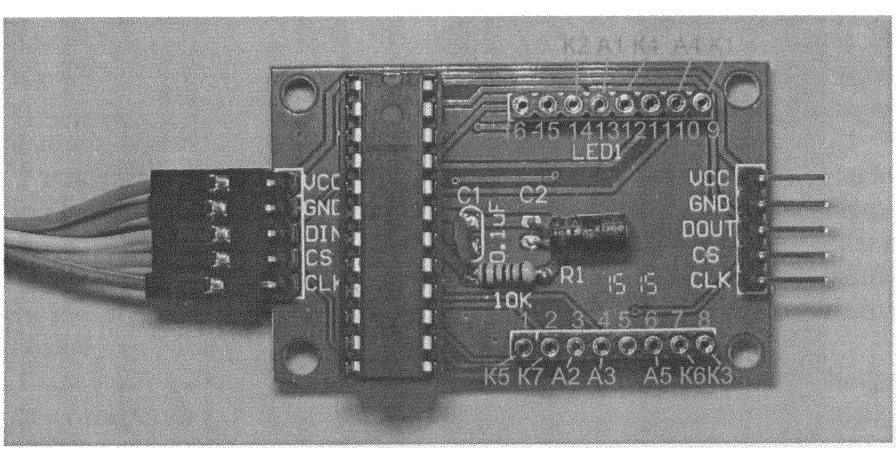

Рис. 15

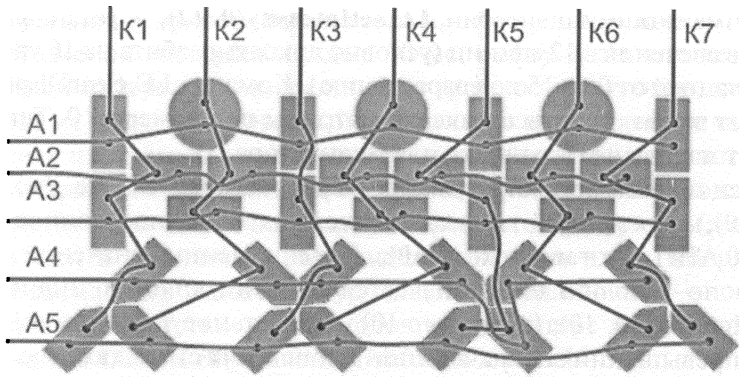

Рис. 16

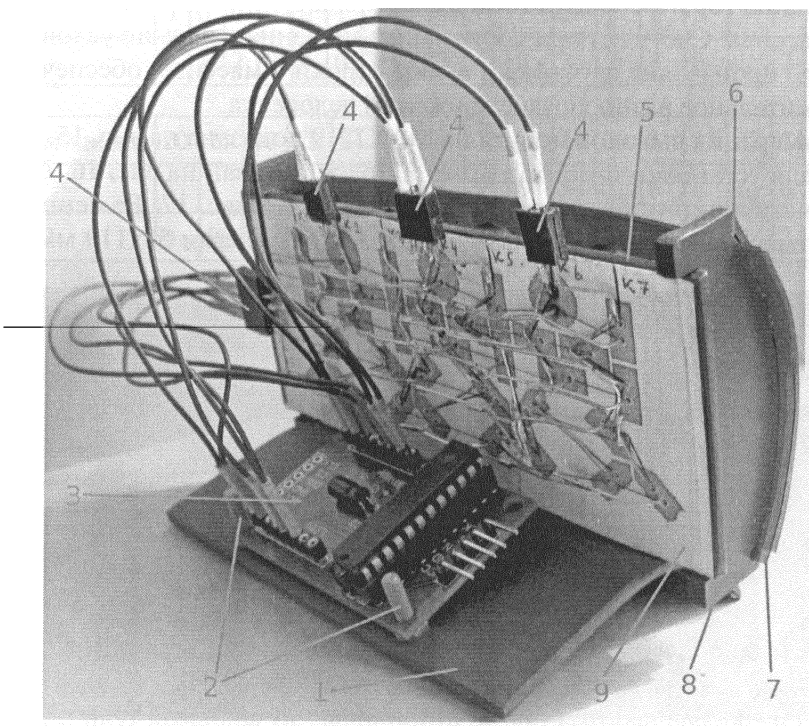

Рис. 17

пилена из корпуса старого телевизора. На обратную сторону наклеена бумажная распечатка 9, соответствующая рис. 16. Шилом намечены, а затем просверлены отверстия под выводы светодиодов. Плата модуля 3 надета на винты 2, прикрученные к подставке 1, которая изготовлена из того же материала, что и пластина. Основание 1 и пластина 5 склеены между собой. Для усиления прочности в торце основания вклеены в глухие отверстия два стальных штифта, изготовленных из металлической скрепки. Они вставляются в подставку сквозь отверстия.

Снаружи индикатор закрыт съёмным светофильтром. Он состоит из двух одинаковых боковых планок 8 и пары пластин-светофильтров 7 (отрезки пластика от бутылок из под кваса). Пластины 7 изогнуты и вставлены с небольшим усилием в щели 6, выпиленные ручным лобзиком в каждой планке 8. Они подогнаны по размеру так, чтобы с небольшим усилием надеваться на основание 5. Для соединения с модулем на MAX7219 использованы гнезда 4 и гибкие монтажные провода со штырями.

В светодиодном индикаторе применены 34 шт. светодиодов повышенной яркости красного свечения в круглом корпусе диаметром 10 мм, например КИПМ15М10-К4-П5. Боковые поверхности светодиодов, за исключением трёх, обточены на точильном камне до получения двух параллельных плоскостей. В результате линза светодиода приобретает форму прямоугольника. Корпуса "ножных" светодиодов дополнительно заточены на угол. Сверху линзы зашлифованы мелкой наждачной бумагой, чтобы кристалл светодиода давал рассеянный свет.

Боковые поверхности всех светодиодов закрашивают сначала жёлтой (или белой) нитрокраской, а затем чёрной. Основание светодиодов и пластина, на которой они смонтированы, также покрашены в чёрный цвет. Это сделано для исключения подсветки соседних светодиодов и поверхностей. Внешний вид индикатора без светофильтра показан на **рис. 18**, а с фильтром — на **рис. 19**.

Светодиоды на пластину монтируют в следующем порядке. Сначала в отверстия вставляют выводы (с соблюдением полярности) верхнего правого светодиода по рис. 18 (или верхний слева на рис. 16) и сгибаем вывод катода вверх. Затем вставляют второй (нижний) свето-

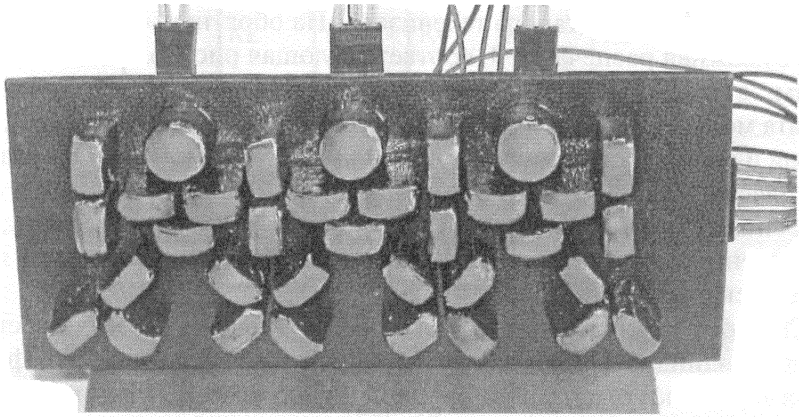

Рис. 18

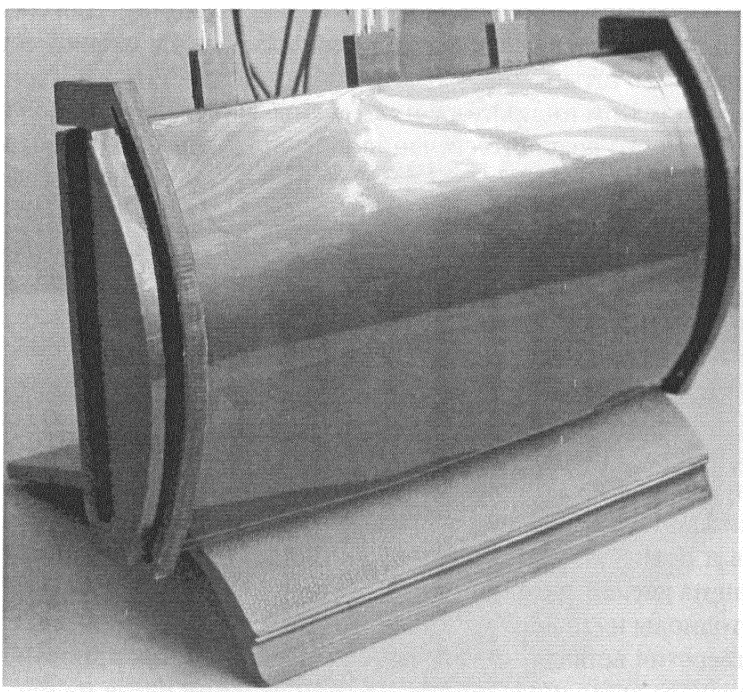

Рис. 19

диод. Сгибают его катод к катоду первого и спаиваем между собой. Аналогично монтируют весь вертикальный ряд (столбец). Далее отгибают вправо все выводы анодов светодиодов, оставив зазор от плоскости пластины в 2...3 мм. Затем монтируют второй ряд-столбец, в конце соединяя аноды в "строки", и т. д. Так последовательно получим индикатор. Следует напомнить, что светодиода HL12 нет.

Питать устройство можно от стабилизированного зарядного устройства сотового телефона или компьютера через USB-разъём. Устройство можно разместить внутри пластикового контейнера подходящего размера и использовать как приставку к компьютеру во время прослушивания музыки.

Скетч конструкции представлен ниже.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа для танцующего человечка вместо светодиодной матрицы
8*8, и max7219 //

#include «LedControl.h»//подключаем данную библиотеку
int x=0;int i=0;int k=1;int n=1;int j=1;
int tabl[24][20]={0,1,1,2,1,1,1,0,0,3,1,3, 0,0,2,0,0,4,3,4,
          0,1,1,2,1,1,1,0,0,3,1,3, 0,2,2,2,1,4,2,4,
          0,1,1,2,1,1,1,0,0,3,1,3, 0,2,2,0,0,4,2,4,
          0,1,1,2,1,1,1,0,0,3,1,3, 0,0,2,2,1,4,3,4,
          0,1,1,2,1,1,1,0,0,3,1,3, 0,0,2,0,1,4,2,4,
          0,1,1,2,1,1,1,0,0,3,1,3, 0,2,2,2,0,4,3,4,
          0,1,1,2,1,1,1,0,0,3,1,3, 0,2,2,0,0,4,2,4,
          0,1,1,2,1,1,1,0,0,3,1,3, 0,0,2,2,1,4,3,4,
          3,1,3,2,4,1,3,0,2,3,3,3, 2,2,4,2,2,4,4,3,
          3,1,3,2,4,1,3,0,2,3,3,3, 2,0,4,0,4,4,1,3,
          3,1,3,2,4,1,3,0,2,3,3,3, 2,2,4,2,2,4,4,3,
          3,1,3,2,4,1,3,0,2,3,3,3, 2,0,4,0,5,3,3,4,
```

```
        5,1,5,2,6,1,5,0,5,3,6,3, 4,0,6,2,4,3,5,4,
        5,1,5,2,6,1,5,0,5,3,6,3, 4,0,6,0,4,4,5,4,
        5,1,5,2,6,1,5,0,5,3,6,3, 4,2,6,0,4,3,5,4,
        5,1,5,2,6,1,5,0,5,3,6,3, 4,0,6,2,4,4,5,4,
        5,1,5,2,6,1,5,0,5,3,6,3, 4,2,6,2,4,4,5,4,
        5,1,5,2,6,1,5,0,5,3,6,3, 4,2,6,0,4,3,6,4,
        5,1,5,2,6,1,5,0,5,3,6,3, 4,0,6,0,4,4,5,4,
        5,1,5,2,6,1,5,0,5,3,6,3, 4,2,6,2,4,3,6,4,
        3,1,3,2,4,1,3,0,2,3,3,3, 2,2,4,0,2,4,4,3,
        3,1,3,2,4,1,3,0,2,3,3,3, 2,0,4,0,4,4,1,3,
        3,1,3,2,4,1,3,0,2,3,3,3, 2,2,4,0,2,4,4,3,
        3,1,3,2,4,1,3,0,2,3,3,3, 2,0,4,0,5,3,3,4};//массив-таблица для
хранения поз человечка


 LedControl LC = LedControl(12, 11, 10, 1);// создаём объект класса
для 1 индикатора
// при этом выводы 12-DIN 11-CLK 10-CS //

void setup()
{
  pinMode(A0, INPUT); // подключение датчика/
  LC.shutdown(0, false);//выключаем энергосберегающий режим
  LC.setIntensity(0, 0);// устанавливаем интенсивность в 0 единиц
между 0 и 15
  LC.clearDisplay(0);//очищаем матрицу 1

}

void loop()
{
  x=analogRead(A0);
 if (x<=650)//если звуковой сигнал получен...
 {
  if(x>=150&x<=450){LC.setIntensity(0, 10);j=j+4;}//если сигнал
сильный...
```

```
 if(x>450&x<=550){ LC.setIntensity(0, 3);j=j+3;}//если сигнал сред-
ний...
 if(x>550){ LC.setIntensity(0, 0);j=j+2;}//если сигнал слабый...

 if (j>=14)//условие смены позы человечка
 {
  k++; i++;j=1;LC.clearDisplay(0);//очищаем матрицу 1
  if (i==9){k=random(0,24);i=1;}if (k==23){k=0;}// условие псевдослу-
чайной смены части танца
 for (n=0;n<=18;n=n+2)//включение контура одной позы человечка
  {
   LC.setLed(0, tabl[k][n] ,tabl[k][n+1],1);
  }

  }
  delay(30);
  }
  }
//////////////////////
// Конец /
//
////////////////////////////////////////////////////////////////////
```

Ночной светильник

Применение совместно с Arduino Uno ультразвукового датчика-дальномера HC-SR04 (**рис. 20**) позволило реализовать устройство, описание конструкции которого приводится далее. HC-SR04 — один из самых распространённых и доступных датчиков, он позволяет измерять расстояние до объекта в интервале от 2 до 400 см. В предлагаемой конструкции с его помощью будет проводиться измерение расстояние от светильника до ладони. Датчик имеет небольшие габариты и простой интерфейс. Внешний вид светильника показан на **рис. 21**. Изменяя расстояние между ладонью и светильником от 5 до 45 см можно переключать цвет свечения или выключить его совсем. В программе-скетче заложено пять вариантов работы светильника: белый свет, выключено, красный, зелёный и синий. Изменив программу, число вариантов можно сделать больше.

Схема устройства показана на **рис. 22**. Датчик HC SR04 имеет четыре вывода. Два из них Vcc и Gnd — для подачи питания, а два Trig и Echo — цифровые вход и выход соответственно. Для проведения измерения расстояния на вход Trig необходимо подать запускающий импульс длительностью около 10 мкс. По спаду импульса датчик излучает в пространство перед собой пачку ультразвуковых сигналов

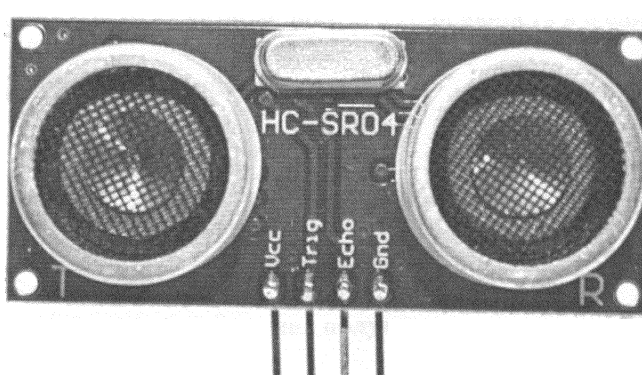

Рис. 20

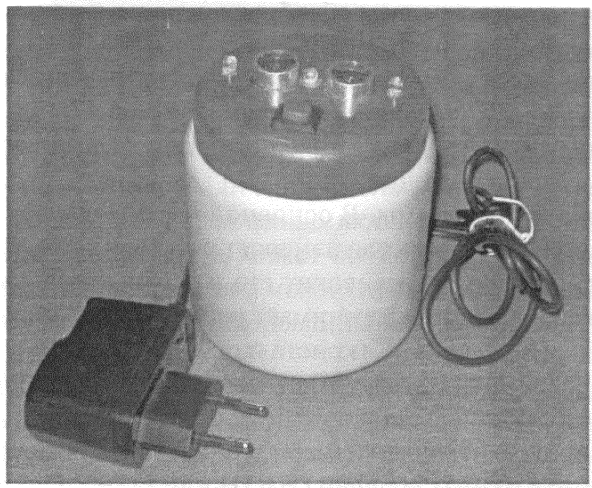

Рис. 21

частотой 40 кГц. После приёма отраженного сигнала на выходе Echo формируется импульс, длительность которого пропорциональна расстоянию до объекта, от которого отразился сигнал. Следующий запускающий импульс рекомендуется подавать не ранее чем через

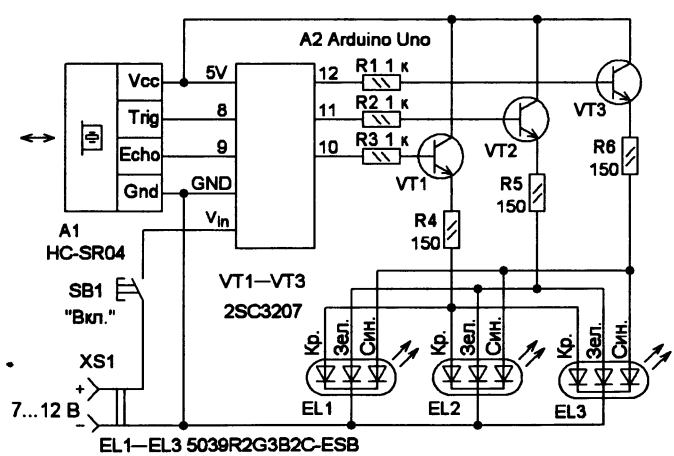

Рис. 22

50 мс, что связано со временем обработки первого импульса. Взаимодействуют с входом и выходом датчика два вывода платы Ардуино — вывод 8 — как выход и вывод 9 — как вход. Напряжение питания на светодиоды EL1—EL3 поступает через транзисторы VT1—VT3. Логические уровни для управления транзисторами формируются на выходах 10, 11, 12 платы Arduino Uno.

Далее о работе программы. В основной части реализованы операции. Каждые 100 мс устройство измеряет расстояние до предполагаемого положения ладони, переводит его в сантиметры, сравнивает с граничными значениями и принимает решение по включению соответствующих светодиодов (кристаллов светодиодов). Команды **digitalWrite(trigPin, HIGH); delayMicroseconds(15);** и **digitalWrite(trigPin, LOW);** формируют запускающий импульс для датчика A1. По команде **pulseIn()** микроконтроллер измеряет длительность сигнала на указанном входе для разных уровней (HIGH или LOW). Например, если задано считывание состояния HIGH , микроконтроллер ожидает пока на указанном входе не появится высокий уровень (HIGH). При его появлении включается счётчик-таймер, который будет остановлен, когда уровень сигнала сменится на низкий (LOW). Измерение производится в микросекундах. Функция **pulseIn()** имеет три аргумента: наименование или номер вывода, тип ожидаемого сигнала, длительность ожидания сигнала (по умолчанию 1000 мс). Таким образом, команда **x=pulseIn(echoPin, HIGH,4000);** означает следующее: измерение длительности сигнала (переменная x) с высоким уровнем на выводе echoPin (вывод номер 9) и присвоение переменной x значения 0 если сигнал не вернулся в течении 4 мс. Команда **x1 = x / 58;** переводит измеренное значение длительности в сантиметры. Команды, начинающиеся с условия **if()** производят включение светодиодов в зависимости от расстояния до ладони. Например, командная строка **if(x1<=55&x1>=45){digitalWrite(10, HIGH);digitalWrite(11, HIGH);digitalWrite(12, HIGH);}** трактуется так: если расстояние до ладони больше или равно 45 см, но меньше или равно 55 см включаются все светодиоды до выполнения нового условия.

Далее о конструкции светильника. Её основа — пластмассовая банка от геля. Светодиоды, датчик и кнопочный выключатель установлены на крышке банки. Для этих элементов в крышке высверли-

вают отверстия соответствующих диаметров. Датчик и светодиоды крепят с помощью термоклея. Резисторы и транзисторы монтируют на макетной гетинаксовой плате, которую размещают внутри банки.

Применены резисторы МЛТ, С2-23, транзисторы — любые маломощные структуры n-p-n. Светодиоды — любые маломощные трёхцветные с общим катодом или одноцветные в каждом канале, но важно, что бы параметры параллельно включенных светодиодов были одинаковыми. Выключатель может быть любого типа малогабаритный. Для питания светильника можно применить сетевой блок питания, в том числе и нестабилизированный напряжением 7...12 В и выходным током до 300 мА.

Налаживание сводится к корректировке программы для установки желаемых расстояний при срабатывание переключений.

Скетч изделия приведён ниже.

```
//////////////////////////////////////////////////////////////////
//
// Arduino UNO
//
//////////////////////
//
//программа для ультразвукового датчика — включатель светодиодов RGB //
const int echoPin=9;
const int trigPin=8;
unsigned int x,x1;

void setup()
{

 pinMode(trigPin, OUTPUT);
 pinMode(10, OUTPUT);// "красный канал"
 pinMode(11, OUTPUT); //"зелёный канал"
 pinMode(12, OUTPUT); //"синий канал"
 pinMode(echoPin, INPUT);
 digitalWrite(trigPin, LOW);
```

```
  digitalWrite(10, LOW);
   digitalWrite(11, LOW);
    digitalWrite(12, LOW);
}

void loop() {

 digitalWrite(trigPin, HIGH);//излучение звука в течении 15 микросе-
кунд
 delayMicroseconds(15);
 digitalWrite(trigPin, LOW);
 x = pulseIn(echoPin, HIGH,4000);//подсчёт длительности отражённо-
го сигнала с ограничением в 4 мс
 x1 = x / 58;//дистанция до ладони в сантиметрах
 if(x1<=55&x1>=45){digitalWrite(10, HIGH);digitalWrite(11,
HIGH);digitalWrite(12, HIGH);}//условие включения всех кристаллов
светодиодов
 if(x1<=45&x1>=35){digitalWrite(10, LOW);digitalWrite(11,
LOW);digitalWrite(12, LOW);}//условие выключения светодиодов
 if(x1<=15&x1>=5){digitalWrite(10,HIGH);digitalWrite(11,
LOW);digitalWrite(12, LOW);}//условие включения красных кристал-
лов
 if(x1<=25&x1>=15){digitalWrite(11,HIGH);digitalWrite(10,
LOW);digitalWrite(12, LOW);}//условие включения зелёных кристаллов
 if(x1<=35&x1>=25){digitalWrite(12,HIGH);digitalWrite(10,
LOW);digitalWrite(11, LOW);}//условие включения синих кристаллов
 delay(100);// временная задержка

}
//
// Конец /
//
////////////////////////////////////////////////////////////////////
```