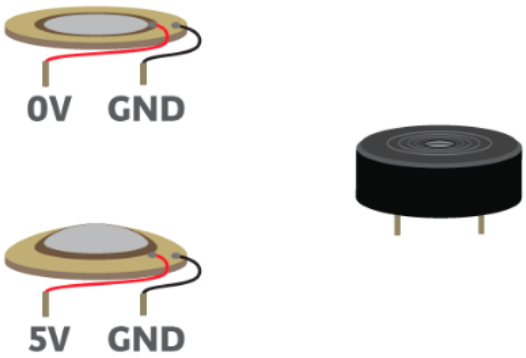
Цель: формирование навыков работы с цифровыми входами при управлении пьезоэлементом и системой светодиодов.

Пьезоэлемент — электромеханический преобразователь, одним из разновидностей которого является пьезоизлучатель звука, который также называют пьезодинамиком, просто звонком или английским buzzer. Пьезодинамик переводит электрическое напряжение в колебание мембраны. Эти колебания и создают звук (звуковую волну).

Данные модули используются для звукового оповещения в тех устройствах и системах, для функционирования которых в обязательном порядке нужен звуковой сигнал. Широко распространены зуммеры в различной бытовой технике и игрушках, использующих электронные платы. Пьезопищалки преобразуют команды, основанные на двухбитной системе счисления 1 и 0, в звуковые сигналы.

Принцип действия зуммера(пьезоэлемента) основан на открытом братьями Кюри в конце девятнадцатого века пьезоэлектрическом эффекте. Согласно ему, при подаче электричества на зуммер он начинает деформироваться. При этом происходят удары о металлическую пластинку, которая и производит “шум” нужной частоты.



Для работы с пьезоэлементом, в основном используют функцию **tone()**, которая генерирует на порту вход/выхода сигнал — прямоугольную "волну", заданной частоты и с 50% рабочим циклом. Длительность может быть задана параметром, в противном случае сигнал генерируется пока не будет вызвана функция noTone().

Синтаксис: **tone(pin, frequency, duration)**

Параметры:

- **pin:** номер порта вход/выхода, на котором будет генерироваться сигнал
- **frequency:** частота сигнала в Герцах
- **duration:** длительность сигнала в миллисекундах

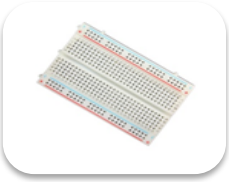
Функции для работы с аналоговыми сигналами:

- **analogRead()** Функция считывает значение с указанного аналогового входа. Большинство плат Arduino имеют 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, это 1024 шага с разрешением 0.0049 Вольт. Считывание значение с аналогового входа занимает примерно 100 микросекунд (0.0001 сек), т.е. максимальная частота считывания приблизительно 10,000 раз в секунду.

- **analogReference(type)** Функция определяет опорное напряжение относительно которого происходят аналоговые измерения. Параметр type: определяет используемое опорное напряжение (DEFAULT, INTERNAL или EXTERNAL).
    - DEFAULT: стандартное опорное напряжение 5 В (на платформах с напряжением питания 5 В) или 3.3 В (на платформах с напряжением питания 3.3 В)
    - INTERNAL: встроенное опорное напряжение 1.1 В на микроконтроллерах ATmega168 и ATmega328, и 2.56 В на ATmega8.
    - INTERNAL1V1: встроенное опорное напряжение 1.1 В (Arduino Mega)
    - INTERNAL2V56: встроенное опорное напряжение 2.56 (Arduino Mega)
    - EXTERNAL: внешний источник опорного напряжения, подключенный к выводу AREF

Оборудование


Кабель USB


Макетная плата


Плата Arduino Uno


Комплект резисторов

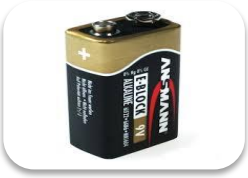

Комплект светодиодов


Комплект соеденительных проводов


Пьезоэлемент

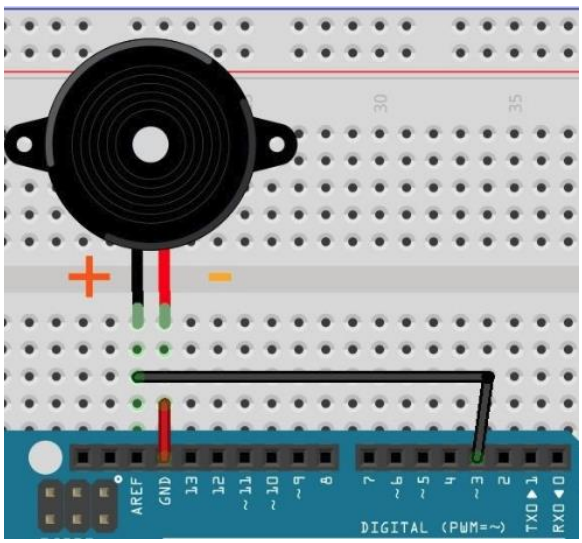

Кнопка

Потенциометр


Батарейка

Упр.1 Подключение пьезоэлемента

- Соберите следущую схему



- Подключите плату Arduino к USB порту компьютера.
- Создайте и сохраните новый проект под названием **Buzzer.ino**.
- Объявим переменную **p** и присвоим ей значение номера пина, на который подключен пьезоэлемент**.**

```
int p = 3;
```

- В области описания функции **Setup()** объявим на пин как выход:

```
void setup()
{
pinMode(p, OUTPUT); //объявляем пин как выход
}
```

- С помощью функции **tone()** будем подавать на выход сигнал различной частоты. Для этого впишите в область описания функции **Loop()** следующий код:

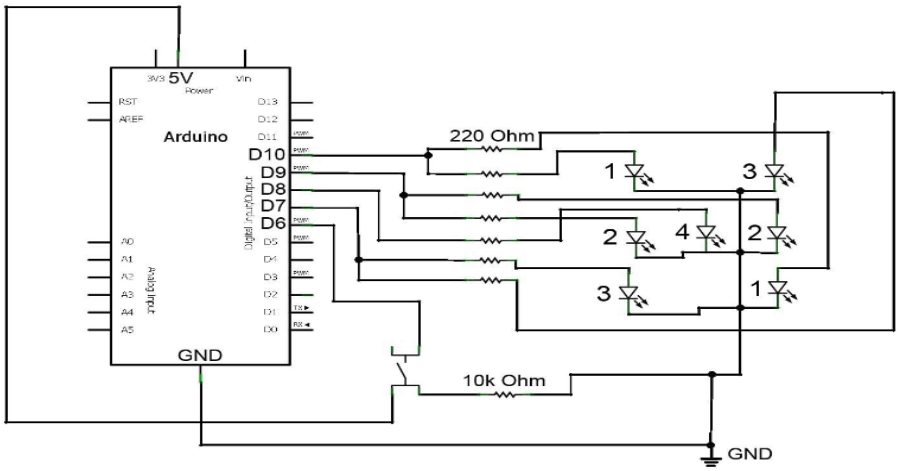
```
void loop()
{
tone (p, 500); //включаем на 500 Гц
delay(100); //ждем 100 Мс
tone(p, 1000); //включаем на 1000 Гц
delay(100); //ждем 100 Мс
}
```

- Для загрузки программы в плату, используем комбинацию клавиш **Ctrl+U**.
- При правильной работе программы зазвучит прерывистый сигнал .

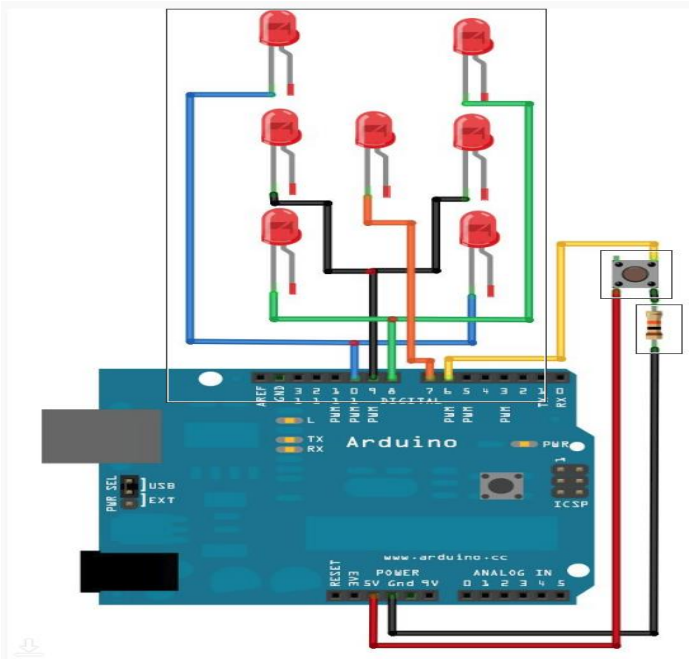
Упр.2 Управление светодиодами при помощи кнопки

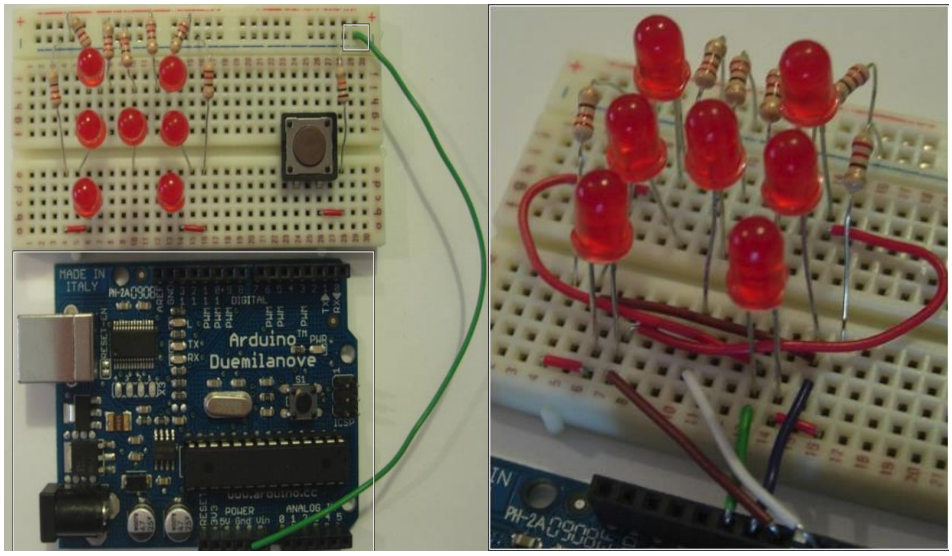
При при помощи системы светодиодов и кнопки, воспроизведем бросок игрального кубика.

- Внимательно рассмотрим следующую схему:



- Чтобы создать все грани кубика, нам понадобится 7 светодиодов, расположенных в форме буквы H, все они связаны с разными входыми Arduino, но большенство обьеденим в пары, чтобы облегчить использование.
- Для создания всех граней, последуем следующим правилам:
  - Для числа 1 игральных костей: загорается светодиод 4
  - За № 2 кости: загорается группа 1
  - Для числа 3 игральных костей: загораются группы 3 и 4
  - Для числа 4 игральных костей: загораются группы 1 и 3
  - Для числа 5 игральных костей: загораются группы 1, 3 и 4
  - Для числа 6 игральных костей: загораются группы 1, 2 и 3
- Запускать «кубик» будем по нажатию на кнопку. Кнопку подключим на плату через резистор с сопротивлением 10 кОм.
- Для реализации проекта, соберите схему следующим образом :





- Подключите плату к компьютеру, запустите среду программирования Arduino IDE.
- Создайте и сохраните новый проект под названием **Dice.ino**.

- Объявим переменные для каждого из светодиодов, кнопки, времени паузы, статуса кнопки (нажата или нет) и хранения показателя кубика.

```
int pinLeds1 = 10;

int pinLeds2 = 9;

int pinLeds3 = 7;

int pinLed4 = 8;

int buttonPin = 6;

int buttonState;

long ran;

int time = 2000;
```

- Опишем использующиеся пины в функции setup() . Кроме того добавим строчку с функцией инициализирующей генератор псевдослучайных чисел RandomSeed(seed). Где seed: параметр, задающий начало выдачи псевдослучайных значений на последовательности

```
void setup ()

{

  pinMode (pinLeds1, OUTPUT);

  pinMode (pinLeds2, OUTPUT);

  pinMode (pinLeds3, OUTPUT);

  pinMode (pinLed4, OUTPUT);

  pinMode (buttonPin, INPUT);

  randomSeed(analogRead(0));

}
```

- В функцию loop(), запишем алгоритм броска «Кубика».

```
void loop()

{

  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH){

    ran = random(1, 7);
```

```
    if (ran == 1){
     digitalWrite (pinLed4, HIGH);
     delay (time);
    }
    if (ran == 2){
     digitalWrite (pinLeds1, HIGH);
     delay (time);
    }
    if (ran == 3){
     digitalWrite (pinLeds3, HIGH);
     digitalWrite (pinLed4, HIGH);
     delay (time);
    }
    if (ran == 4){
     digitalWrite (pinLeds1, HIGH);
     digitalWrite (pinLeds3, HIGH);
     delay (time);
    }
    if (ran == 5){
     digitalWrite (pinLeds1, HIGH);
     digitalWrite (pinLeds3, HIGH);
     digitalWrite (pinLed4, HIGH);
     delay (time);
   }
    if (ran == 6){
     digitalWrite (pinLeds1, HIGH);
     digitalWrite (pinLeds2, HIGH);
     digitalWrite (pinLeds3, HIGH);
     delay (time);
    }
   }
  digitalWrite (pinLeds1, LOW);
  digitalWrite (pinLeds2, LOW);
  digitalWrite (pinLeds3, LOW);
  digitalWrite (pinLed4, LOW);
 }
```

- Загрузим программу в плату Arduino и нажимая на подключенную к плате кнопку, проверим работу построенной системы.

Задания для самостоятельного выполнения

- Измените скетч Buzzer.ino, таким образом чтобы сигнал постепенно нарастал, а затем плавно затухал.
- Добавьте к схеме первого упражнения потенциометр, используя его для регулировки частоты сигнала.
- На основе упражнений, соберите систему, эмитирующую игральные кости. Добавьте в схему пьезоэлемент, который будет проигрывать мелодию при выпадении цифры 12.

Пройдите тест №3 «Свет и звук» на странице «Тесты».