

Тема 2

Ветвления программы

Операторы ветвлений

Оператор выбора (проверки условия)

if

```
если → if (хорошее_поведение)
      {
        поход_в_зоопарк;
        покупка_мороженого;
      }
      иначе → else
      {
        стояние_в_углу;
      }
```

Оператор выбора (проверки условия)

if

краткие формы

```
if (Условие)
{
    Команда1;
    Команда2;
    ...
}
```

```
if (Условие)
    Команда1;
```

```
if (Условие)
{
    Команда1;
    Команда2;
    ...
} else
    Команда3;
```

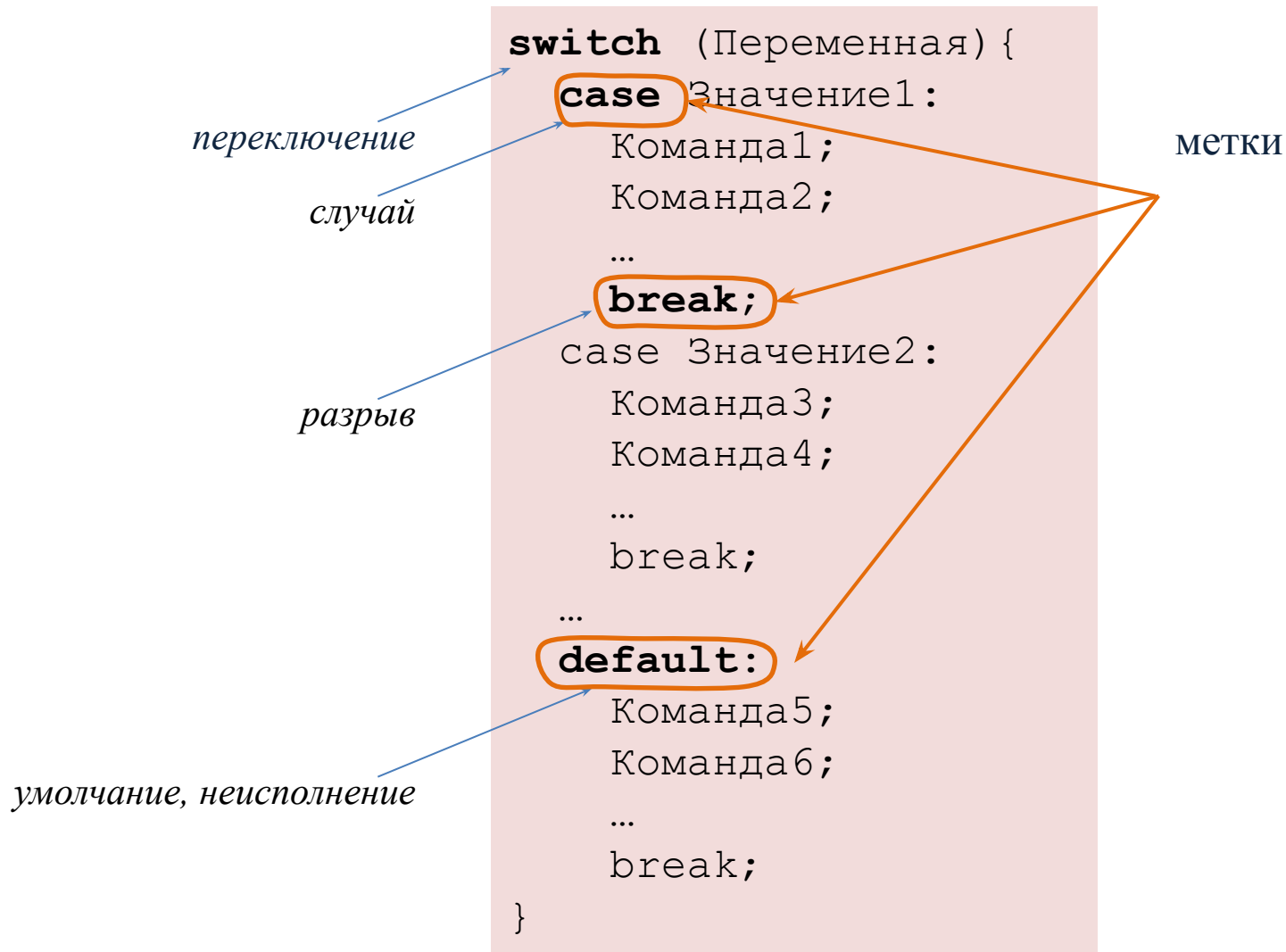
```
if (Условие)
    Команда1;
else
    Команда2;
```

Арифметические операции для
проверки условия в операторе
if

<code>==</code>	равно
<code>!=</code>	не равно
<code>></code>	больше
<code>>=</code>	больше или равно
<code><</code>	меньше
<code><=</code>	меньше или равно

Оператор выбора (проверки условий)

switch



Оператор выбора (проверки условий)

switch

```
switch (оценка_по_математике) {  
    case 1:  
        стояние_в_углу;  
        break;  
    case 2:  
        сидение_дома;  
        мытьё_полов;  
        break;  
    case 3:  
        сидение_дома;  
        мытьё_посуды;  
        break;  
    default:  
        поездка_на_рыбалку;  
        break;  
}
```

Замена оператора **switch** на **if**

```
switch (оценка_по_математике)
{
    case 1:
        стояние_в_углу;
        break;
    case 2:
        сидение_дома;
        мытьё_полов;
        break;
    case 3:
        сидение_дома;
        мытьё_посуды;
        break;
    default:
        поездка_на_рыбалку;
        break;
}
```

```
if(оценка_по_математике == 1)
    стояние_в_углу;
else
{
    if(оценка_по_математике == 2)
    {
        сидение_дома;
        мытьё_полов;
    } else
    {
        if(оценка_по_математике == 3)
        {
            сидение_дома;
            мытьё_посуды;
        } else
            поездка_на_рыбалку;
    }
}
```

```
if(оценка_по_математике == 1)
    стояние_в_углу;
else {
    if(оценка_по_математике == 2){
        сидение_дома;
        мытьё_полов;
    } else {
        if(оценка_по_математике == 3){
            сидение_дома;
            мытьё_посуды;
        } else
            поездка_на_рыбалку;
    }
}
```

В программировании различают знак
присваивания "="
и знак **сравнения** "=="

Истина и **Ложь** в программировании – это значения, которые могут принимать переменные специального типа **boolean**

логический

Переменные типа **boolean** могут иметь только два значения:

true и **false**

истина

ложь

Операторы циклов

Оператор цикла с предусловием

while

```
while (условие)
{
    Команда1;
    Команда2;
    Команда3;
    ...
}
```

пока →

```
координаты_1 = дом;
координаты_2 = школа;
while (координаты_1 != координаты_2)
{
    шаг;
    пересчет_координаты_1;
}
```

Оператор цикла **for**



```
i=i+1; → ++i;
```

```
i=i-1; → --i;
```

```
for(int i=1;i<3;++i)
{
    digitalWrite(lamp1+1, HIGH);
    digitalWrite(lamp1+3, HIGH);
    delay(500);
    digitalWrite(lamp1+1, LOW);
    digitalWrite(lamp1+3, LOW);
    delay(500);
}
```

Оператор цикла с постусловием **for**

```
координаты_2 = школа;  
for(координаты_1 = дом; координаты_1 != координаты_2; шаг)  
{  
    пересчет_координаты_1;  
}
```

```
координаты_1 = дом;  
координаты_2 = школа;  
while(координаты_1 != координаты_2)  
{  
    шаг;  
    пересчет_координаты_1;  
}
```

Собственные процедуры

Процедура – это часть кода, которому назначено какое-то имя

Процедуры используют для:

- 1) сокращения кода;
- 2) удобства;
- 3) наглядности.

```
void помыть _руки()  
{  
    зайти_в_ванную()  
    включить_кран()  
    намылить_руки()  
    смыть_с_рук_мыло()  
    выключить_кран()  
}
```

Аргументы – это некоторые переменные, которые передаются в процедуру

процедура

```
void имяПроцедуры(ТипАргумента1 имяАргумента1,  
ТипАргумента2 имяАргумента2, ...)  
{  
  Команда1;  
  Команда2;  
  Команда3;  
  ...  
}
```

```
void setupMotorShield()  
{  
  pinMode(leftDirPin, OUTPUT);  
  pinMode(leftSpeedPin, OUTPUT);  
  pinMode(rightDirPin, OUTPUT);  
  pinMode(rightSpeedPin, OUTPUT);  
}
```

функция

```
ТипВозвращаемогоЗначения имяФункции(  
ТипАргумента1 имяАргумента1,  
ТипАргумента2 имяАргумента2, ...)  
{  
  Команда1;  
  Команда2;  
  Команда3;  
  ...  
  return возвращаемаяПеременная;  
}
```

```
int sum(int a, int b)  
{  
  int c = a + b;  
  return c;  
}
```

Массив – это *индексированный*, то есть пронумерованный, список элементов.

Массив является **переменной**, которую нужно объявлять (инициализировать).

```
типЭлементовМассива имяМассива [количествоЭлементов] =  
(Элемент1, Элемент2, ...);
```

```
int myPins[3] = {2, 4, 8};
```

```
char команда[3] = {Иванов, Петров, Сидоров};  
                  0          1          2
```

```
X = команда[2];
```

```
X = Сидоров
```

индекс

Инициализация массива

полная инициализация массива

```
int mySensVals[6] = {2, 4, -8, 3, 2, 1};
```

инициализация массива без указания его длины

```
int myPins[] = {2, 4, 8, 3, 6, 2};
```

инициализация массива без указания значений элементов

```
int myInts[6];
```


Пример задания режимов работы пинов с помощью оператора **for**

```
int ledPin[4]={10,4,6,2};  
int i;  
  
void setup ()  
{  
  for(i=0;i<4;++i)  
    pinMode(ledPin[i],OUTPUT);  
}
```

i	i<4	результат проверки	Индекс массива	
i=0	0<4	true	ledPin[0]	устанавливаем 10 ПИН
i=1	1<4	true	ledPin[1]	устанавливаем 4 ПИН
i=2	2<4	true	ledPin[2]	устанавливаем 6 ПИН
i=3	3<4	true	ledPin[3]	устанавливаем 2 ПИН
i=4	4<4	false	выходим из цикла	

Строка – это массив букв.

123

Слово «**dog**» - это массив элементов с **ТИПОМ** «**СИМВОЛ**».

Тип «символ» при инициализации обозначается как **char**.

```
char theword[4] = {'d', 'o', 'g', 0x00};
```

терминальный ноль

```
int myPins[3] = {2, 4, 8};
```

количество элементов будет вычислено автоматически

```
char theword[ ] = "dog";
```

двойные кавычки

К символам строки можно обращаться отдельно, т.е. по индексу.

```
char theword[ ] = "dog";  
theword[0] = 'f';
```

"fog"

Таблица символов в кодировке **ASCII**

десятичный код символа

символ

шестнадцатеричный код символа

Таблица, ставящая в соответствие каждому символу свой код, называется **кодировкой**.

!	33	0x21	A	65	0x41	a	97	0x61
"	34	0x22	B	66	0x42	b	98	0x62
#	35	0x23	C	67	0x43	c	99	0x63
\$	36	0x24	D	68	0x44	d	100	0x64
%	37	0x25	E	69	0x45	e	101	0x65
&	38	0x26	F	70	0x46	f	102	0x66
'	39	0x27	G	71	0x47	g	103	0x67
(40	0x28	H	72	0x48	h	104	0x68
)	41	0x29	I	73	0x49	i	105	0x69
*	42	0x2A	J	74	0x4A	j	106	0x6A
+	43	0x2B	K	75	0x4B	k	107	0x6B
,	44	0x2C	L	76	0x4C	l	108	0x6C
-	45	0x2D	M	77	0x4D	m	109	0x6D
.	46	0x2E	N	78	0x4E	n	110	0x6E
/	47	0x2F	O	79	0x4F	o	111	0x6F
0	48	0x30	P	80	0x50	p	112	0x70
1	49	0x31	Q	81	0x51	q	113	0x71
2	50	0x32	R	82	0x52	r	114	0x72
3	51	0x33	S	83	0x53	s	115	0x73
4	52	0x34	T	84	0x54	t	116	0x74
5	53	0x35	U	85	0x55	u	117	0x75
6	54	0x36	V	86	0x56	v	118	0x76
7	55	0x37	W	87	0x57	w	119	0x77
8	56	0x38	X	88	0x58	x	120	0x78
9	57	0x39	Y	89	0x59	y	121	0x79
:	58	0x3A	Z	90	0x5A	z	122	0x7A
;	59	0x3B	[91	0x5B	{	123	0x7B
<	60	0x3C	\	92	0x5C		124	0x7C
=	61	0x3D]	93	0x5D	}	125	0x7D
>	62	0x3E	^	94	0x5E	~	126	0x7E
?	63	0x3F	_	95	0x5F			
@	64	0x40	`	96	0x60			

ASCII – это американский стандартный код для обмена информацией.

ASCII представляет собой **кодировку** для представления десятичных цифр, латинского и национального алфавитов, знаков препинания и управляющих символов.

Каждый символ имеет свой **числовой код** в диапазоне от 0 до 255 (*один байт*).

ASCII **используется в программировании** для определения кодов нажатых символов на клавиатуре, либо кодирования/декодирования, экранирования, анализа данных.

Коды служебных символов в кодировке **ASCII**

переход на новую строку

Символ	Десятичный код	Шестнадцатеричный код
Терминальный ноль	0	0x00
Табуляция	9	0x09
Перенос строки (LF)	10	0x0A
Возврат каретки (CR)	13	0x0D
Пробел	32	0x20

переход на начало строки

Слово «**dog**», записанное при помощи кодов

```
char theword[4];  
theword[0] = 0x64;  
theword[1] = 0x6F;  
theword[2] = 0x67;  
theword[3] = 0x00;
```

Пьезоизлучатель звука переводит переменное напряжение в колебание мембраны, которая в свою очередь создаёт звуковую волну.

buzzer – звонок, звуковой сигнал, гудок



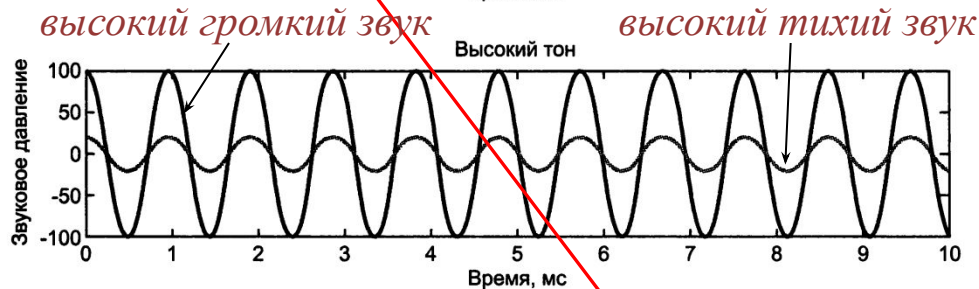
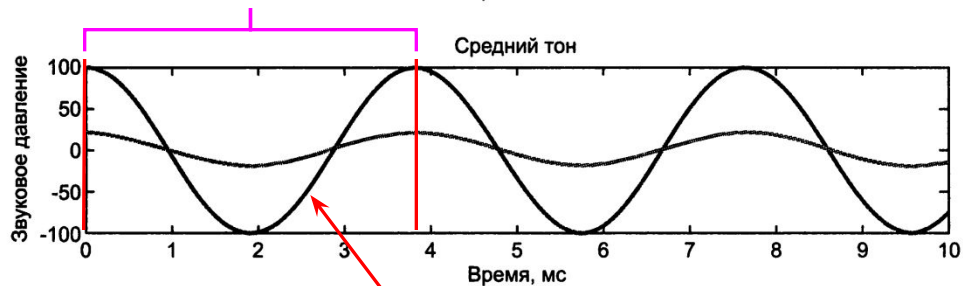
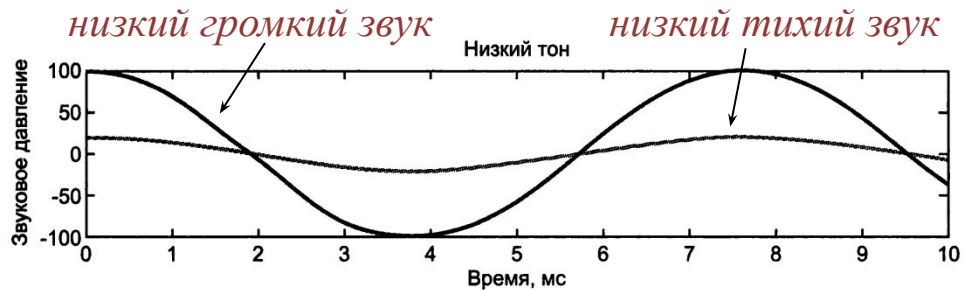
Внешний вид
пьезоэлемента



Обозначение на
схемах пьезоэлемента

Пьезодинамик – это **конденсатор**, который звучит при зарядке и разрядке.

Пьезоэлемент изменяет свой размер, когда на него подаётся напряжение, и возвращается к первоначальному размеру, если напряжение снять.

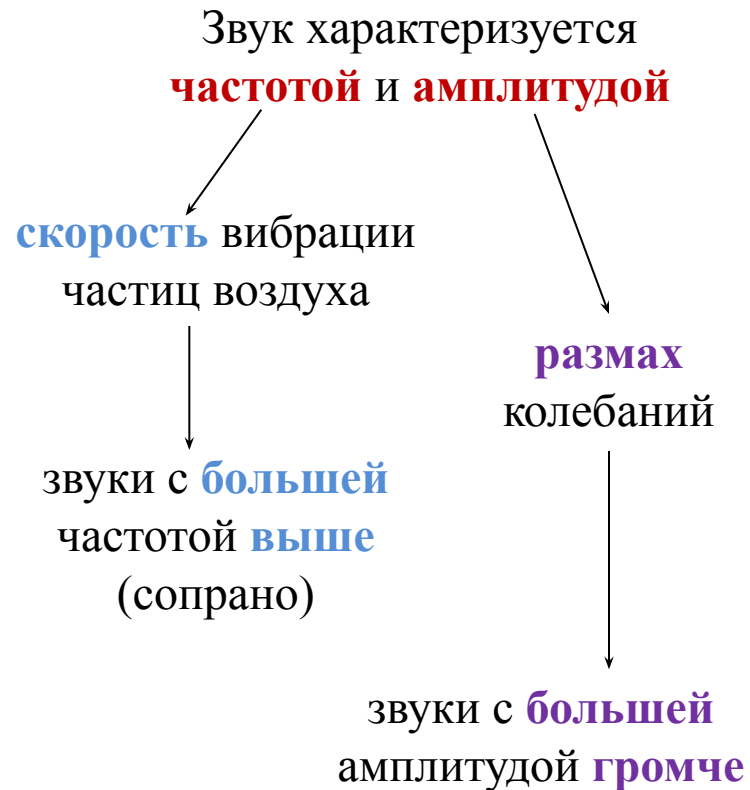


Звуковые волны с различной частотой и амплитудой

Нота **До** первой октавы

частота – **261,63** Гц (столько колебаний в секунду)

период – $1/261,63 = 3,822$ мс (полное колебание)



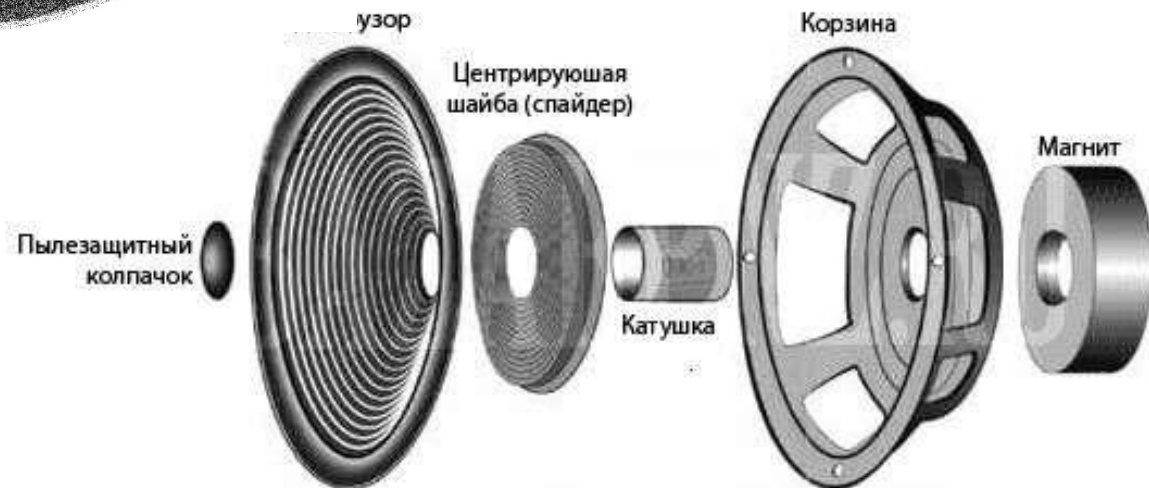
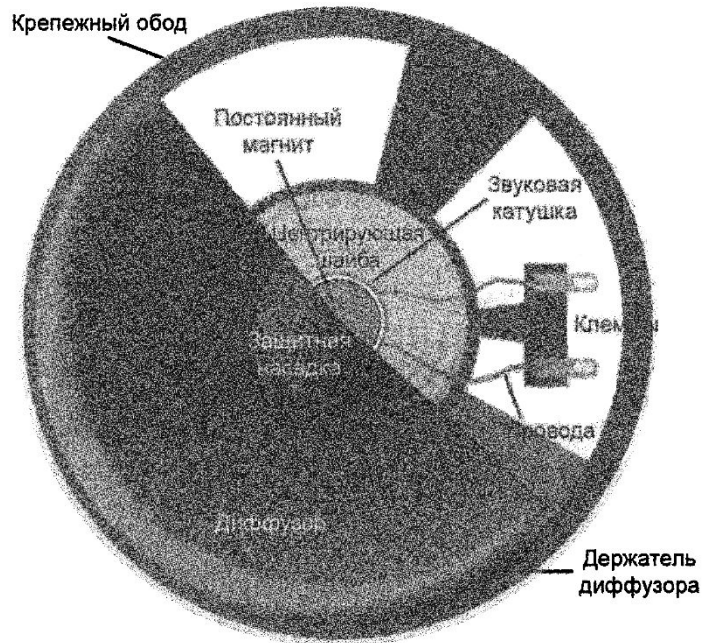
Основные характеристики пьезоэлемента

Рекомендуемое (номинальное) напряжение	V	Вольт
Громкость (на заданном расстоянии)	P	Децибелл
Пиковая частота	f_P	Герц
Ёмкость	C	Фарад

Звук – это периодическое сгущение и разряжение воздуха.

Высота звука – это частота этих сгущений и разряжений.

Устройство динамика



Встроенная функция **tone** используется для генерации звуков произвольной частоты

tone (аргумент1, аргумент2, аргумент3)

номер пина Arduino для генерации волны

частота сигнала

продолжительность звучания (необязательный аргумент)

Встроенная функция **strlen** подсчитывает количество символов в строке до терминального нуля

strlen (аргумент1)

строка

Встроенная функция

millis() возвращает текущее время
(с момента включения
Arduino) в миллисекундах

Арифметическая операция

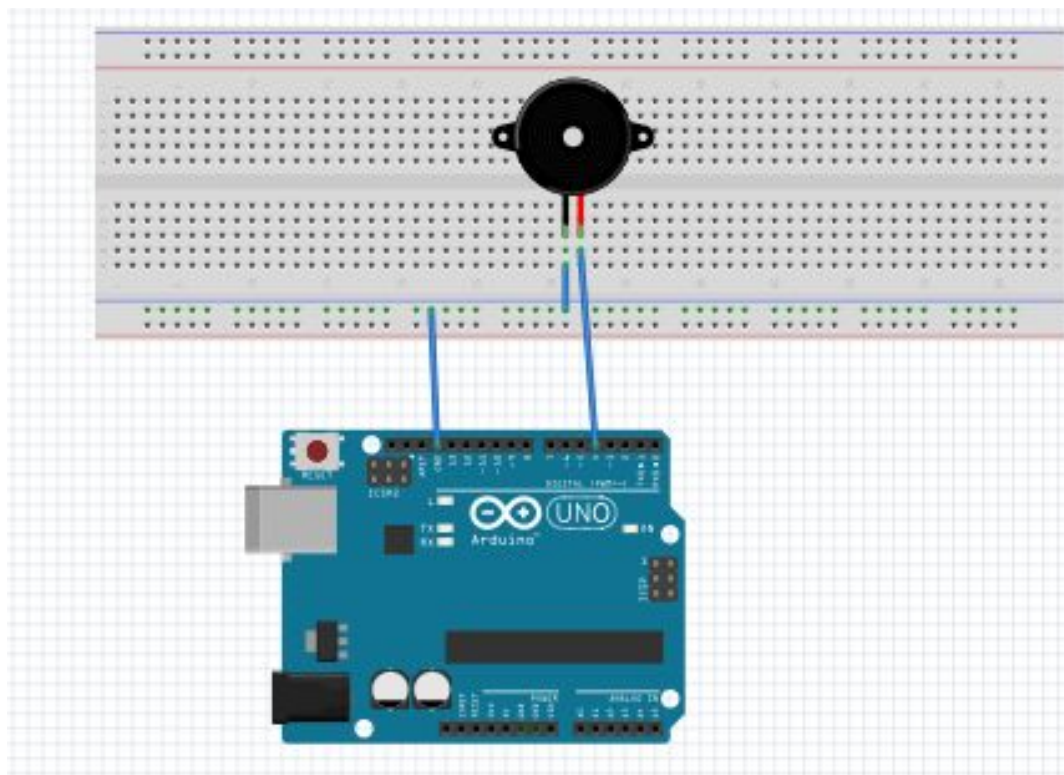
% остаток от деления двух
операндов
x = y % 2;

Тип данных

long длинный целый
Занимает 4 байта памяти
Диапазон значений:

-2 147 483 648..2 147 483 647

Схема подключения пьезоэлемента



Задание

Соберите на макетной плате схему, состоящую из **5-ти светодиодов и пьезоэлемента.**

Запрограммируйте контроллер на включение нужного количества светодиодов одновременно в зависимости от значения **переменной**:

- если значение переменной находится в диапазоне **от 1 до 5**, то одновременно загорается столько светодиодов, чему равно значение переменной;

- если значение переменной **меньше 1**, то мигает 1-й светодиод;

- если значение переменной **больше 5**, то мигает 5-й светодиод.

Каждый из режимов работы должен сопровождаться звучанием одной из нот 1 октавы.

При программировании необходимо использовать операторы **for** (или **while**), **if**, **switch**. При задании номера пинов необходимо использовать массив.

Частоты звучания нот

Частота, Гц		1	2
Нота		октава	октава
До	C	261.63	523.25
До-диез	C	277.18	554.36
Ре	D	293.66	587.32
Ре-диез	D	311.13	622.26
Ми	E	329.63	659.26
Фа	F	349.23	698.46
Фа-диез	F	369.99	739.98
Соль	G	392.00	784.00
Соль-диез	G	415.30	830.60
Ля	A	440.00	880.00
Си-бемоль	B	466.16	932.32
Си	H	493.88	987.75