Работа со временем

```
pulseIn(), millis(), micros(),
 delay(), delayMicroseconds()
```

pulseIn()

Функция pulseIn() считывает длину сигнала на заданном порту (**HIGH** или **LOW**).

Например, если задано считывание **HIGH** функцией pulseIn() , функция ожидает пока на заданном порту не появиться **HIGH**. Когда **HIGH** получен, включается таймер, который будет остановлен когда на порту вход/выхода будет **LOW**.

pulseIn()

Синтаксис:

pulseIn(pin, **value**)

pulseIn(pin, **value**, **timeout**)

Параметры:

pin: номер порта вход/выхода, на котором будет ожидаться сигнал.

value: тип ожидаемого сигнала — **HIGH** или **LOW**.

timeout (опционально): время ожидания сигнала (таймаут) в микросекундах; по умолчанию - одна секунда. (unsigned long)

Возвращаемое значение:

Длина сигнала в микросекундах (мкс) или 0, если сигнал не получен до истечения таймаута. (unsigned long)

1 с = 1000 мс = 1 000 000 мкс

pulseIn()

```
int pin = 7;
unsigned long duration; //длительность

void setup()
{
  pinMode(pin, INPUT);
}

void loop()
{
  duration = pulseIn(pin, HIGH);
}
```

delay()

Функция delay() останавливает выполнение программы на заданное в параметре количество миллисекунд (1000 миллисекунд в 1 секунде).

Замечания по использования функции

Не рекомендуется использовать эту функцию для событий длиннее 10 миллисекунд, т.к. во время останова, **не могут** быть произведены манипуляции с портами, **не могут** быть считаны сенсоры или произведены математические операции. В качестве альтернативного подхода возможно контролирование времени выполнения тех или иных функций с помощью millis().

Большинство активности платы останавливается функцией delay(). Тем не менее работа прерываний не останавливается, продолжается запись последовательно (serial) передаваемых данных на RX порту, ШИМ сигнал (analogWrite) продолжает генерироваться на портах.

delayMicroseconds()

Функция delayMicroseconds() останавливает выполнение программы на заданное в параметре количество микросекунд (1 000 000 микросекунд в 1 секунде).

```
delayMicroseconds(1000) = delay(1)
```

millis()

Функция millis() возвращает количество миллисекунд с момента начала выполнения текущей программы на плате Arduino. Это количество сбрасывается на ноль, в следствие переполнения значения, приблизительно через 50 дней.

Возвращаемое значение: количество миллисекунд с момента начала выполнения программы. (unsigned long) //до 4,294,967,295 секунд

millis()

Пример:

```
unsigned long time; //переменная для millis() нужна обязательно!


void setup(){
  Serial.begin(9600);
}
void loop(){
time = millis(); //выводит количество миллисекунд с момента начала выполнения программы
Serial.println(time); // ждет секунду, перед следующей итерацией (т.е., повторения) цикла.
delay(1000);
}
```

micros()

Функция micros() возвращает количество микросекунд с момента начала выполнения текущей программы на плате Arduino.

Значение переполняется и сбрасывается на ноль, приблизительно через 70 минут. На 16MHz платах Ардуино (Duemilanove и Nano) функция micros() имеет разрешение 4 микросекунды (возвращаемое значение всегда кратно

Возвращаемое значение: микросекунд с момента начала выполнения программы. (unsigned long) //до 4,294,967,295 секунд (как и у millis)

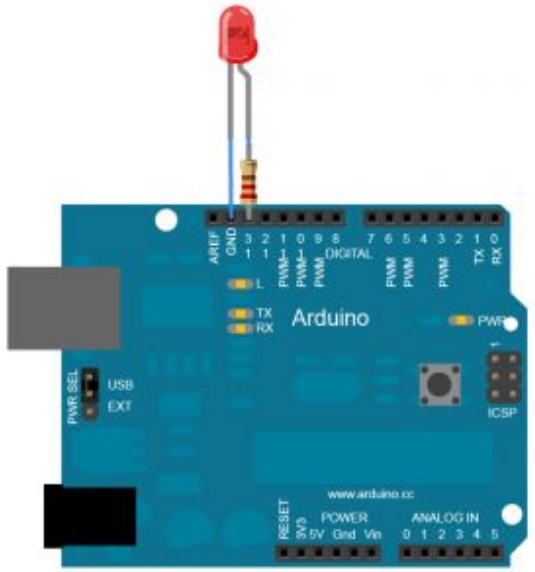
micros()

Пример:

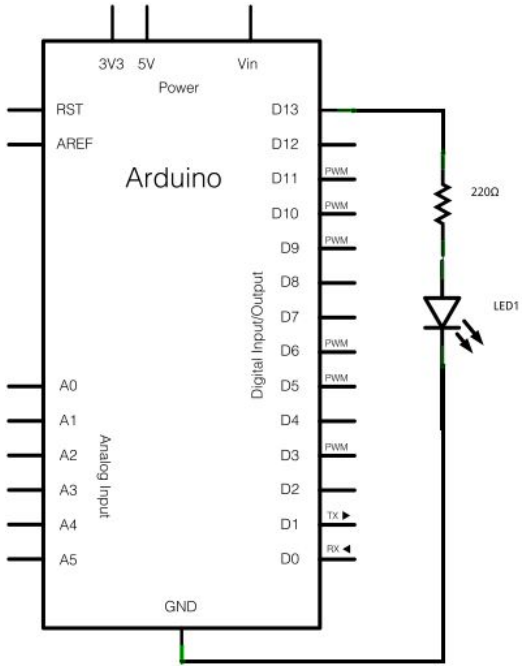
```
unsigned long time; //переменная для micros() тоже нужна обязательно!


void setup(){
  Serial.begin(9600);
}
void loop(){
time = micros(); //выводит количество миллисекунд с момента начала выполнения программы
Serial.println(time); //выводит количество микросекунд с момента начала выполнения программы
delay(1000); // ждет секунду, перед следующей итерацией (т.е., повторения) цикла.
}
```

Мигаем светодиодом без delay()

В этом примере без delay(), программа будет запоминать время когда был включен или выключен светодиод и в каждом цикле loop() будет проверять не прошло ли достаточно времени для переключения светодиода.



При сборке схем не забывайте
про токоограничивающий резистор
для каждого светодиода!

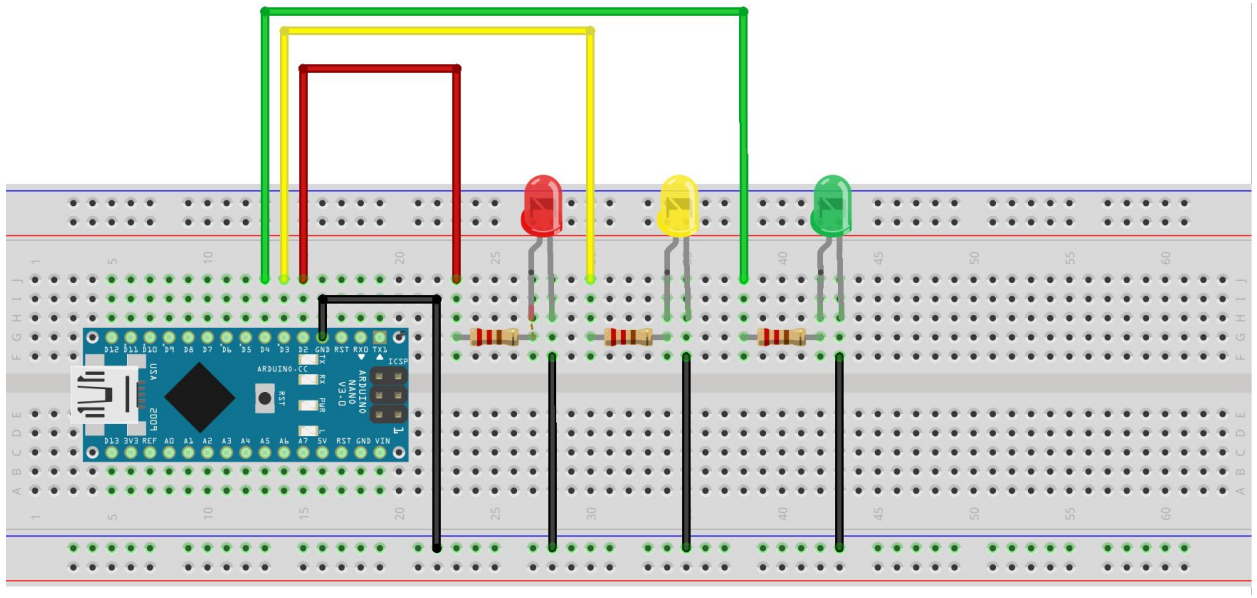


Мигаем светодиодом без delay()

```
int ledState = LOW;             // этой переменной устанавливаем состояние светодиода
long previousMillis = 0;        // храним время последнего переключения светодиода
long interval = 1000;           // интервал между включение/выключением светодиода (1 секунда)
void setup() {
pinMode(13, OUTPUT); }  // задаем режим выхода для порта, подключенного к светодиоду

void loop() {
unsigned long currentMillis = millis();
if(currentMillis - previousMillis > interval) {      //проверяем не прошел ли нужный интервал, если прошел то…
   previousMillis = currentMillis;                   // сохраняем время последнего переключения
if (ledState == LOW)                                 // если светодиод не горит…
   ledState = HIGH;                                  // …то зажигаем, и наоборот
   else
     ledState = LOW;
   digitalWrite(13, ledState);                       // устанавливаем состояния выхода, чтобы включить или выключить светодиод
  }
}
```

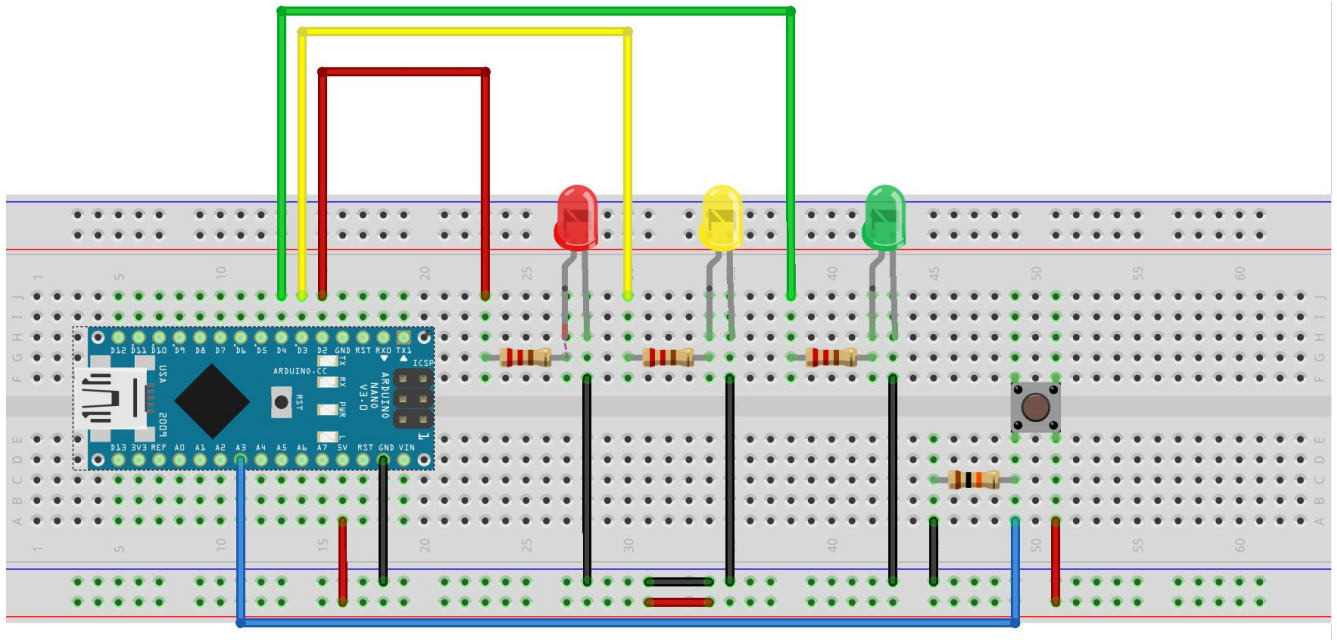
Контрольное задание

- Соберите схему (можно онлайн)
- Напишите единый скетч для мигания светодиодов: красным частотой 1 раз в 2 секунды, жёлтый – 1 раз в 3 секунды, зелёный 1 раз в 5 секунд. Длительность горения светодиода выберите самостоятельно от 50 до 500 мс.



Контрольное задание повышенного уровня

- Соберите схему на макетной плате
- Напишите скетч для «бегущего огня» из светодиодов, где время горения всех светодиодов будет настраиваться зажатием кнопки.



Например, если подержали кнопку 1 секунду, то светодиоды горят по 1 секунде).

Паузу между включениями выдержать в пределах 50-100 мс.

Ограничьте программно время свечения светодиодов от 50 до