Ардуино 2 – Первые шаги.

1. Установка драйвера Arduino . Проверка установки и определение номера виртуального COM порта для Arduino.

Итак, надеюсь, что Вы приобрели «Arduino» и Вам не терпится начать с ней работать. Модель «Arduino» может быть любой, например «Ardino-Nano», «Arguino Uno», «Freeduino» или вообще какая-нибудь экзотическая индийская « …duino». Чтобы начать работать с «Arduino» сначала необходимо установить в компьютер драйвер этого устройства. Для тех кто не знает что такое драйвер – объясняю. Драйвер это небольшая программ, которая позволяет компьютеру определить какое устройство к нему подключено. Кроме того в драйвере находится инструкция для компьютера как работать с данным устройством. Наверняка Вы уже устанавливали драйверы для принтера, сканера, видеокамеры для «Skype» и т.д. Поэтому, надеюсь, что у Вас никаких трудностей при установке драйвера к «..duino» не возникнет.
Теперь вопрос – а где взять драйвер ? А вот где – в программе, которая называется, так же как и устройство – «Arduino». Зайдите в любую поисковую систему и введите : «Скачать программу «Arduino». В окне поисковика Вам будет предложено много сайтов с которых можно скачать эту программу. Скачайте последнюю версию «Arduino 1.0.4» или «Arduino 023», распакуйте и перепишите ее в корневую директорию диска С. Никакой инсталляции для этой программы не требуется..

Итак, Вы скачали программу «Arduino» и поместили ее на диск С. Теперь подключаете микроконтроллер «Arduino» к порту USB к порту USB компьютера. «Как только **МК « Arduino »** будет подсоединен к USB порту компьютера, Вы услышите короткий звуковой сигнал, и в нижнем правом углу экрана монитора появится сообщение операционной системы Windows об обнаружении нового USB устройства (рис.1).

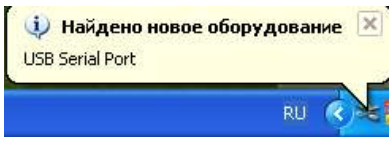


Рис. 1.

Тут же появится окно мастера установки нового оборудования, который будет заниматься установкой драйвера. Он предложит начать поиск драйвера в Интернете. Нам такой вариант не требуется, так как драйвер для «Arduino» находится в ее паке на диске С, поэтому выбирайте пункт "**Нет, не в этот раз**" и нажимайте кнопку "**Далее**".

Далее необходимо выбрать пункт **"Установка из указанного места"** и нажать на кнопку **"Далее"**. Появится окно выбора способа поиска драйвера. Нажмите на кнопку **"Обзор".** Укажите путь к папке **« FTDI USB Drivers»**. Драйвер находится в папке **« Arduino-0023/drivers/FTDI USB Drivers »**, в ней же находится и необходимая установочная информация. Нажмите на кнопку **"Ok".** Нажмите на кнопку **"Далее"**.
Начнется процесс копирования драйверов и регистрации их в системе.

По окончании установки нажмите кнопку "**Готово**". После этого должно появится дополнительное сообщение операционной системы об успешности установки устройства и его готовности к использованию (рис. 2).

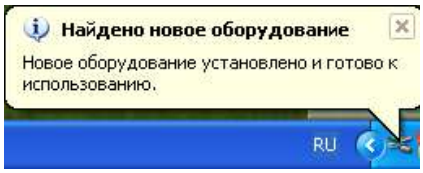
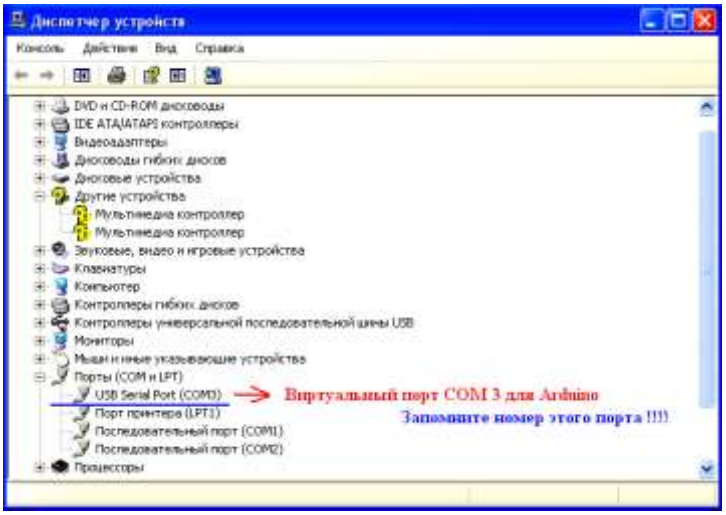


Рис. 2.

По окончании установки, модулю будет присвоен номер виртуального COM порта. Именно через этот виртуальный порт будет возможен обмен информацией с модулем. Как же узнать номер виртуального COM порта, с которым будет работать Ваша «Arduino». Имейте в виду, что это очень важно !



Чтобы определить этот номер откройте диспетчер устройств, например, через ***Пуск -> Настройка -> Панель управления -> Система -> Оборудование -> Диспетчер устройств.***

В ветке **"Порты (COM и LPT)"** должна появиться запись о новом устройстве – **USB Serial Port (COM 3).** Виртуальному порту может быть присвоен номер **1 - 9 (COM1 - COM9).**

У нас он имеет номер **COM 3** (рис.3). Запомните этот номер, он вам пригодится во время работы с **МК Arduino.**

Рис. 3.

Загрузка и настройка программы «Arduino».

Итак, драйвер к **МК Arduino** успешно установлен. Можно начинать работу. Но сначала надо загрузить программу **«Arduino».** Для этого зайдите на диск С и откройте папку **« Arduino-0023»** (или папку с другой весиеей **«Arduino»,** например **«Arduino 1.0.4»**. Найдите в этой папке файл **arduino.exe** и запустите его. Это исполняемый файл, поэтому обычной инсталляции программ для Windows в данном случае не потребуется. На экране монитора появится рабочий стол программы **«Arduino»** (рис.4).

Рис. 4.

Теперь программе **«Arduino»** необходимо сообщить программе с какой моделью **Arduino** мы будем работать и с помощью какого виртуального COM порта будет работать **МК «Arduino»** с компьютером..

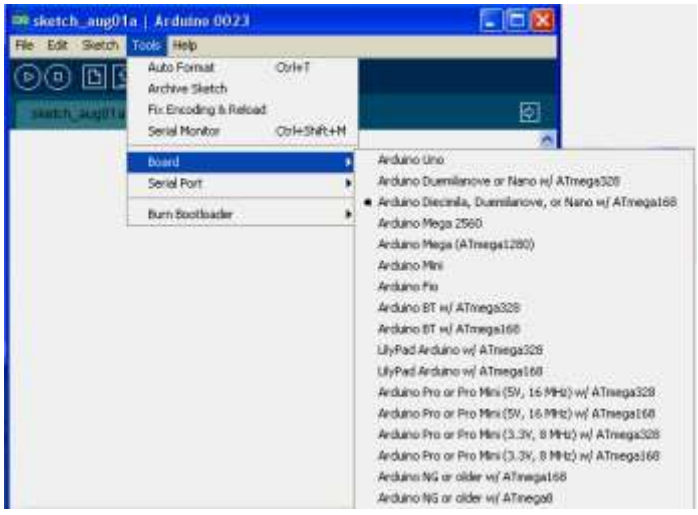


Рис.5.

Чтобы выбрать в программе нашу модель МК **«Arduino»,** откроем меню **Tools**. В открывшемся меню наведем курсор на строку **Board.**
Откроется список различных моделей МК Arduino. Выбираем, ту модель, которая у Вас имеется.
Мы работаем с моделью **«Ardino-2009».** Она в списке имеет название – **«Arduino Diecmila, Duemilanone, jr Nano w/ Atmega168»** (рис. 5).

У Вас может быть другая модель, ничего страшного, методом проб и ошибок Вы найдете и Вашу модель.

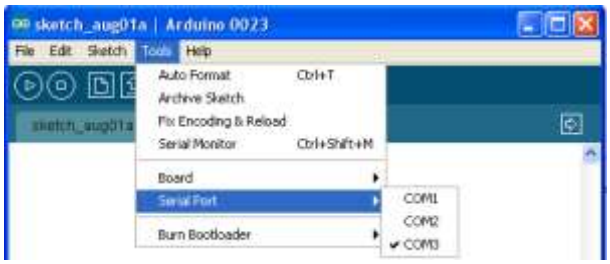


Рис. 6.

Теперь необходимо сообщить программе номер виртуального COM порта. Для этого в меню **Tools** открываем подменю **Serial Port.** Откроется список всех COM портов, которые отыскала программа при ее запуске (рис. 6). Как уже говорилось выше, при установке драйвера ПК зарезервировал для МК **«Arduino» COM3** (у Вас может быть другой). Выбираем COM3 и закрываем меню. **МК «Arduino»** и программа полностью готовы к работе.

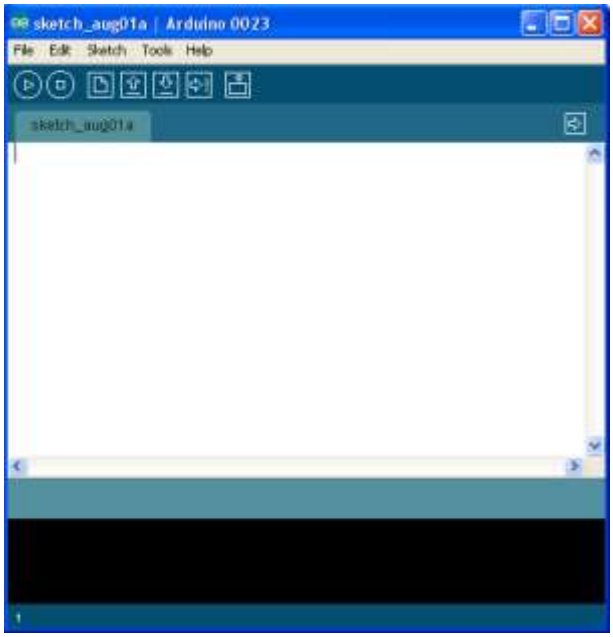


Рис. 7.

Ну что же, поехали ! Но чтобы ехать надо знать, как рулить. Поэтому познакомимся с основными элементами управления программой Arduino.

В верхней части программы под строкой **Меню** находится панель элементов управления, состоящая из семи элементов находится под панелью меню (рис. 7)
Она состоит из следующих элементов (рис 8) :

Рис. 8.


-компиляция, перевод программы с языка Wiring в машинный язык


- стоп,


- новый скетч (программа),


открыть скетч (программу),


- сохранить скетч (программу),


- загрузить скетч (программу) в Arduino,


- включить монитор COM порта.

Самыми важными для нашей будущей работы являются первая  **и две последние кнопки**  **.**

Ниже белого поля рабочего стола программы **«Arduino»** находится зеленая строка состояния, в которой будет выводиться информация о операциях выполняемых программой, а в самом низу расположен черный экран монитора последовательного COM порта. На этот экран будет выводиться информация о работе модуля МК **«Arduino».**

К моменту написания статьи вышла версия программы **«Arduino 1.0.6».** Это очень приятная вещь. На многих компьютерах она устанавливается автоматически. Эта же версия опубликована и для других операционных систем «MAC OC», «Linux». Работать в этой программе еще проще. Кстати все версии «Arduino» полностью совместимы.

Скетчи (программы) для Freeduino.

Практически во всех языках программирования текстовый перечень инструкций для компьютера называется программой. А вот в **«Arduino»** они называются не программами, а **скетчами.**

Для создания **скетчей** в программе **« Arduino»** использован язык программирования **Wiring**. Можно считать что это урезанный С плюс-плюс.

Вы не знаете С плюс-плюс ? И я не знаю, поэтому будем учиться вместе. Главное, имейте в виду, что все это несложно и абсолютно доступно любому «чайнику».

Скетчи для **«Arduino»** могут быть простыми и сложными. Но в каждом скетче обязательно должны присутствовать две основных функции: **setup()** и **loop().**

Функция **setup()** вызывается один раз, после каждого включения питания или сброса платы **«Arduino».** В ней инициализируются переменные, устанавливается режимы работы портов и т.д.

Функция **loop()** – это функция бесконечного цикла – она последовательно раз за разом исполняет команды, которые описаны в ее теле. Т.е. после завершения функции снова произойдет ее вызов и все повторяется сначала..

Познакомимся с «мини-шаблоном» для любого скетча.

```
int ... = ... ;

void_setup()
{
pinMode(... , OUTPUT);
pinMode(... , INPUT);
}

void_loop()
{
.......
}
```

1. Назначения «ножек» и описания переменных.

2. Настройка портов ввода и вывода в фигурных скобках { } в функции «void setup()».

3. Написание последовательности команд в фигурных скобках { } в функции «void loop().

Внимание !!! В дальнейшем, все Ваши скетчи (программы) будут состоять из этих трех блоков.

Теперь можно начинать создавать настоящий работающий скетч, но предварительно давайте познакомимся с уровнями сигналов, с которыми работают порты **«Arduino»** и настройкой цифровых портов на вход и выход.

Уровни сигналов порта HIGH и LOW

При чтении или записи к цифровому порту применимо только два возможных значения – порт может быть установлен как **HIGH** (высокий уровень) или **LOW** (низкий уровень). Уровень **HIGH** соответствует 5 вольтам на выходе. При чтении значения на цифровом порте, начиная с 3 вольт и выше, микропроцессор воспримет это напряжение как **HIGH.** Эта константа представлена целым числом 1.

Уровень **LOW** соответствует 0 вольтам на выходе порта. При чтении значения на цифровом порте, начиная с 2 вольт и меньше, микропроцессор воспримет это напряжение как **LOW**. Эта константа представлена целым числом 0.
Таким образом, оба следующих вызова будут эквивалентны:

digitalWrite(13, HIGH); // можно так,
digitalWrite(13, 1); // а можно и так

Считается, что первый вариант более нагляден.

Функция - **digitalWrite() -** устанавливает на цифровом пине (контакте) высокий или низкий уровень напряжения. Если пин был сконфигурирован как выходной, то на нем устанавливается + 5 В для высокого уровня напряжения **(HIGH)** и 0 В для низкого уровня **(LOW).**

Настройка цифровых портов на ввод (INPUT) и вывод (OUTPUT) сигналов

Цифровые порты могут использоваться на ввод или вывод сигналов.
Изменение порта с ввода на вывод производится при помощи функции pinMode().
Порты, сконфигурированные на ввод сигналов, имеют большое входное сопротивление, что позволяет подключать к ним источник сигнала, и порт не будет потреблять большой ток.
Порты, сконфигурированные на вывод сигналов, имеют малое выходное сопротивление. Это означает, что такие порты могут обеспечивать подключенные к ним элементы электроэнергией. В этом состоянии порты поддерживают положительное или отрицательное направление тока до 40 мА (миллиампер) на другие устройства или схемы. Это позволяет подключить к ним какую-либо нагрузку, например светодиод (через резистор, ограничивающий ток). Порты, сконфигурированные как выводы, могут быть повреждены, если их замкнуть накоротко на «землю» (общая шина питания), на источник питания +5 В, или подсоединить к мощной нагрузке с малым сопротивлением.
Пример:

```
pinMode(13, OUTPUT);  //13й вывод будет выходом
pinMode(12, INPUT);   //а 12й – входом
```

Функция pinMode() - устанавливает типа порта – входной **(INPUT)** или выходной **(OUTPUT).**

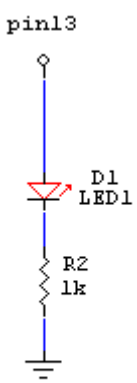
И вот настал счастливый момент – сейчас мы будем писать настоящий работающий скетч. Сначала определимся – а что он должен делать ? Если Вы когда-нибудь учились программировать на каком либо языке программирования, то вероятно знаете, что первая программа на всех языках программирования одинаковая. Она выводит на экран такую фразу : « **Здравствуй Паскаль ! »** (или C++, Delphi, и т.д.).

Так вот для Arduino или Freeduino первый скетч позволит управлять зажиганием и гашением светодиода. Светодиод это полупроводниковый прибор, который начинает светиться при подаче на него напряжения от 2 до 5 В.

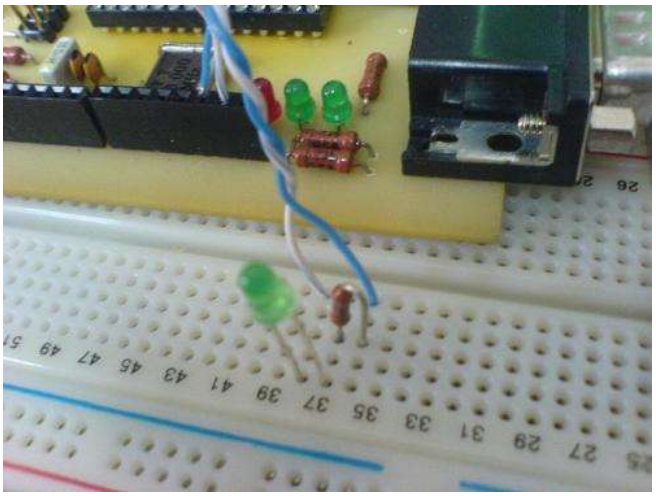
Скетч 1. Мигание светодиода.

Процесс создания программы для МК **«Arduino»** состоит из следующих этапов : написание кода (текста скетча) – компиляция кода – загрузка кода в МК **«Arduino»**.

Сначала соберем несложную схему, для которой нам понадобятся **«Arduino»,** макетная плата, светодиод **D1 LED 1** любого цвета, резистор на 750 Ом – 1 к и два соединительных провода (рис.9). Макетную плату лучше использовать беспаечную (они продаются в Интернет магазинах или на радио-рынках) (рис.9).



Но это не принципиально – можно собирать и на обычных платах методом пайки. Обратите внимание на подключение светодиода. У него есть плюс (длинная ножка) и минус (короткая ножка). Плюс светодиода подключается к контакту № 13 МК **«Arduino»,** а минус – через токо- ограничительный резистор 750 Ом – 1 к к контакту GND (земля). Если резистор не подключить, то может перегореть светодиод или испортится МК **«Arduino» .**
Рис. 9.



А вот так выглядит эта схем на беспаечной макетной плате (рис. 10).

Рис.10.

Итак, пишем наш первый скетч :

```
int ledPin = 13;                  //  назначаем рабочий контакт 13 и присваиваем его
                                  // значение  переменной ledPin
void setup()                      //функция начальных установок – вызывается
                                  //при включении микроконтроллера
{                                 //фигурная скобка открывает тело функции
  pinMode( ledPin , OUTPUT);      //устанавливаем 13 контакт в режим вывода OUTPUT
}                                 //фигурная скобка закрывает  функции начальных установок
void loop()                       //основная функция – осуществляет замкнутый цикл
                                  //в процессе работы микроконтроллера
{                                 // фигурная скобка открывает тело функции
  digitalWrite(ledPin, HIGH);     //включение светодиода на 13 выходе
  delay(1000);                    //задержка 1000 миллисекунд (1 сек)
  digitalWrite(ledPin, LOW);      //выключение светодиода на 13 выходе
  delay(1000);                    //задержка 1000 миллисекунд (1 сек)
}                                 // фигурная скобка закрывает  тело функции
                                  // По завершении она снова  будет вызвана
```

// - два слеша пишем для комментирования строчек программы. При выполнении скетча весь текст после // игнорируется.

Чтобы Вам было все понятно, разберем исходный текст скетча по подробнее :

Первая строка - int ledPin = 13 - описание переменной, которую мы назовем **ledPin.**
int – означает, что переменная **ledPin – целочисленная,** она занимает 2 байта **(1 байт – 8 бит)** и может хранить значения от -32 768 до 32767. Этой переменной присваиваем значение **13.** Знак равенства в этой строке – это оператор присваивания. Значение 13 означает, что светодиод подключается к цифровому контакту № 13.

Вторая строка - void setup() объявляет обязательную функцию инициализации **(setup)** – она будет вызвана при включении микроконтроллера. v**oid** – обязательное служебное слово.

Третья строка - в ней происходит вызов стандартной функции **pinMode**, с помощью которой 13-й контакт микроконтроллера переводится в режим вывода, чтобы затем можно было управлять напряжением на этом контакте. Функция **pinMode() - у**станавливает :

- **pin -** номер конфигурируемого порта (у нас он **ledPin = 13)** и тип порта
- **Mode - INPUT** или **OUTPUT**, входной или выходной (у нас он выходной – **OUTPUT).**

Таким образом, у нас функция **pinMode,** имеет следующий вид :

- **pinMode(ledPin , OUTPUT);**

Четвертая строка – в ней объявляется функция **loop()** – она будет постоянно вызываться в бесконечном цикле в процессе работы. **void** – обязательное служебное слово

Последние четыре строки – тело цикла **loop() - з**десь мы последовательно выставляем на 13-м контакте «высокое» значение – это + 5 Вольт, делаем паузу в тысячу миллисекунд (одну секунду), выставляем «низкое» значение – это 0 Вольт, и опять делаем паузу в одну секунду.

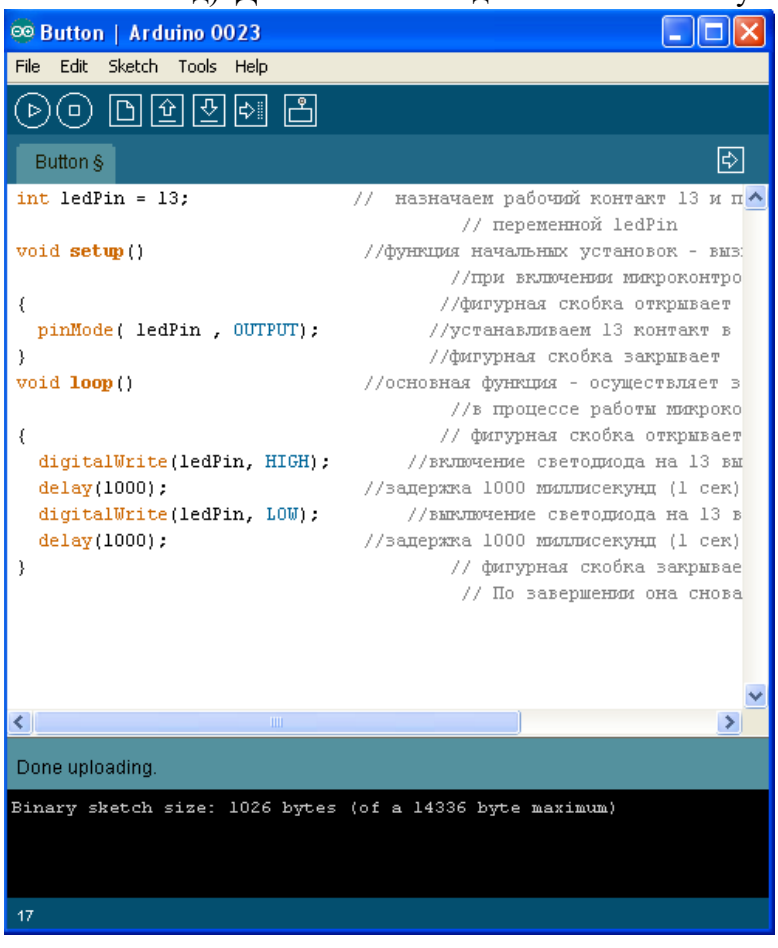
После завершения функции **loop() - эта функция** будет вызвана снова бесконечное количество раз.

Итак, мы получили скетч, переключающий 13-й контакт с + 5 В на 0 В каждую секунду. С учетом того, что к 13-му контакту на плате подключен тестовый индикатор (светодиод), получается, что он будет мигать с периодом 2 секунды.

Поместим нашу программу (можно без комментариев) на рабочий стол **«Arduino»**

Теперь подключим МК «Arduino» к USB порту ПК. О его работоспособности можно судить по загоревшемуся зеленому светодиоду PWR (индикатор питания на плате **«Arduino»** горит постоянно не мигая) и, по дружески подмигивающему красному светодиоду, находящемуся напротив 13 контакта ««**Arduino»**.

Итак, текст скетча готов, теперь его надо скомпилировать (преобразовать в машинный код). Для этого необходимо нажать кнопку компиляции на панели инструментов.



Начнется процесс компиляции. В строке состояния зеленого цвета, она находится ниже окна кода программы) появится надпись **Compiling.**

После удачного окончания процесса компиляции в строке состояния появится надпись **«Done compiling»**), т.е. компиляция прошла без ошибок (рис.11.).

Кроме этого на черном экране монитора последовательного COM порта появится сообщение о размере программы (скетча_ в двоичном коде, в данном случае размер скетча 1018 бит.

Чтобы загрузить скомпилированный скетч в микроконтроллер нажмите кнопку на панели инструментов:

Рис. 11.

В процессе загрузки скетча сначала в строке состояния появится сообщение «**Uploading to I/O Board** », потом будут загораться индикаторы чтения и записи (RX и TX) на плате ««**Arduino**». Затем произойдет программный сброс микроконтроллера, один раз вспыхнет тестовый индикатор, в строке состояния появится сообщение « **Done Uploading**» и затем начнет выполняться наша программа. После этого Вы увидите, что Ваш светодиод будет мигать с периодом 2 с.

Подобный скетч есть и в стандартных примерах – ее можно загрузить через меню **File\Examples\1.Basics\Blink**. И при разработке других скетчей мы будем пользоваться готовыми примерами-заготовками. Но я счел полезным, написание первого Вашего скетча сделать самостоятельным, начиная с чистого листа.

А теперь сохраним Ваш шедевр. Войдите в меню **File**, Нажмите **Save As**.. . Откроется окно с заранее автоматически созданной папкой **«Arduino»** (она находится в папке «Мои документы») Напишите имя Вашего скетча (**только на латинице !!!)** и сохраните его. Проверьте – сохранилась ли он. Зайдите в папку **«Arduino»** и в ней Вы должны увидеть папку с названием Вашего файла, а в нем - сам файл с расширением **.pde.**

Впрочем, светодиод даже не обязательно вставлять в разъём – на плате уже есть сигнальный светодиод, подключённый к 13-му цифровому порту **«Arduino»** через ограничительный резистор.

Теперь немножко переделаем наш скетч таким образом, чтобы плата сообщала нам через USB – порт, когда светодиод горит, а когда нет.

Для этой цели в языке **«Arduino»** есть функция **Serial –** которая и**с**пользуется для связи между платформой **«Arduino»** и компьютером или другими устройствами.

Для выполнения поставленной задачи нужно в скетч добавить всего три строчки кода:

Первая строка : Serial.begin(9600); // инициализация работы с COM-портом

Функция **Serial.begin(int speed) - у**станавливает скорость обмена данными в бит в секунду (бод). Для связи с компьютером определены следующие скорости: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, или 115200. Для связи с устройствами можно установить любую скорость обмена данными. Мы установим стандартную скорость для COM порта 9600 бод.

Вторая строка : Serial.print("H"); // светодиод горит – пишем H

Третья строка : Serial.println("L"); // светодиод погасили – пишем L

Функция **Serial.print(data) - п**ечатает данные в порт. **Data** целочислительные переменные, включая char, floats. Печать чисел с плавающей точкой поддерживается только до двух знаков после запятой.

Таким образом наш скетч приобретает следующий вид :

```
int ledPin = 13;
void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);     // инициализация работы с COM-портом
}
void loop()
```

```
{
  digitalWrite(ledPin, HIGH);
  Serial.print("H");      // светодиод горит – пишем H
  delay(1000);
  digitalWrite(ledPin, LOW);
  Serial.println("L");    // светодиод погасили – пишем L
  delay(1000);
}
```

Компилируем и загружаем скетч в МК.
Светодиод мигает, а сообщения от платы мы можем увидеть на мониторе последовательного порта.
На экране монитора периодически появляются буквы :
H – высокий уровень (HIGH) -светодиод горит,
L – низкий уровень (LOW) - светодиод не горит (рис. 12).

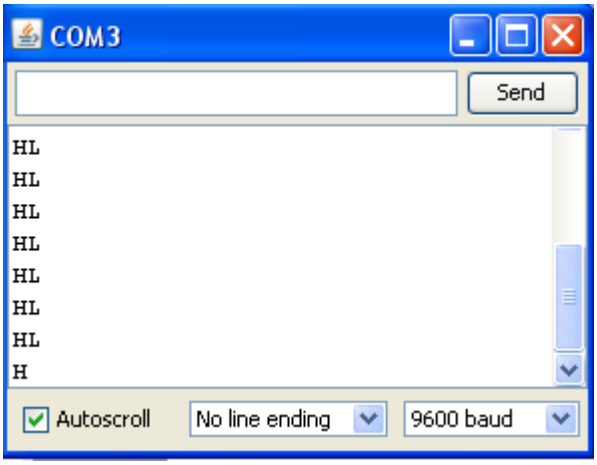


Рис. 12.

Вот и подошло к концу наше второе занятие. Какое продолжение может иметь скетч по загоранию и гашению светодиода. Очень большое – все будет зависеть от Вашей фантазии. К различным цифровым выходам от 0 до 13 можно подключить четырнадцать светодиодов и написав несложный скетч для последовательного зажигания светодиодов и их гашения можно сделать «бегущие огни», которые можно повесть на елочку или использовать в качестве элементов дизайна.

Хочу Вас предупредить – подключать к цифровым выходам обычные лампы накаливания нельзя, так как цифровые выходы рассчитаны на ток до 20 мА, так что если Вы захотите с помощью «Arduino» управлять мощными лампами, электромоторами, моделями роботов и т.д., то для этих целей можно использовать мощные транзисторы, базы которых подключаются к цифровым выходам «Arduino». Но об этом я Вам расскажу в следующий раз.

Небольшое, но нужное дополнение к **Serial.print()** . Оно отвечает на вопрос – Как правильно пользоваться этой функцией.

Serial.print() У этой функции два параметра :

1. Передает данные через последовательный порт как ASCII текст. Эта функция может принимать различные типы данных. Так целые числа выводятся соответствующими им символами ASCII. Вещественные выводятся с помощью двух ASCII символов, для целой и дробной части. Байты передаются как символ с соответствующим номером. Симоволы и строки отсылаются как есть. Пример:

   Serial.print(78) передается как "78"
   Serial.print(1.23456) передается как "1.23"
   Serial.print(byte(78)) передается как "N" (т.к. в таблице ASCII "N" под 78 номером)
   Serial.print('N') передается как "N"
   Serial.print("Hello world.") передается как "Hello world."

2. С помощью второго опционально параметра можно задать базис (систему счисления) для чисел. Допустимые значения BYTE, BIN (двоичный), OCT (восьмиричный), DEC (десятеричный), HEX (шестнадцатеричный). Для вещественных (дробных) чисел второй параметр задает количество знаков после запятой. Пример:

Serial.print(78, BYTE) выводит "N"
Serial.print(78, BIN) выводит "1001110"
Serial.print(78, OCT) выводит "116"
Serial.print(78, DEC) выводит "78"
Serial.print(78, HEX) выводит "4E"
Serial.println(1.23456, 0) выводит "1"
Serial.println(1.23456, 2) выводит "1.23"
Serial.println(1.23456, 4) выводит "1.2346"

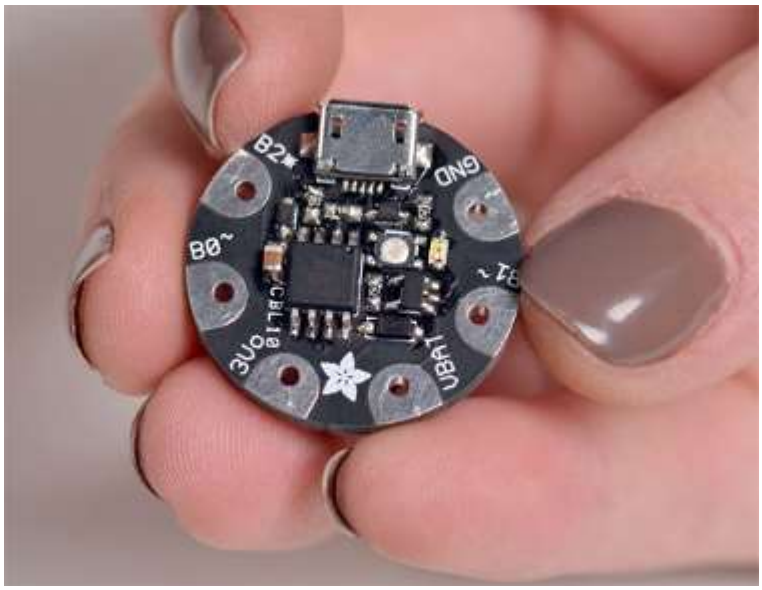
Кроме обычных светодиодов красного, зеленого, желтого и синего есть, так называемые RGB светодиоды. По сути дела – это обычный светодиод, в котором одновременно смонтированы 3 светодиода.

Обратите внимание на то, как подключаются эти светодиоды – общий провод – у них + !

В скетче надо инициализировать 3 светодиода, соответственно пинам подключения и работать как с обычными светодиодами.

На базе Ардуино, светододов и различных датчиков можно создать новое оригинальное направление – Электронный текстиль (Е – текстиль).

Для этого чаще всего применяют плату Ардуино LilyPad - Ардуино для одежды.



Вариант одежды из Е текстиля с RGB светодиодами представлен на фото.

Из вышесказанного можно сделать вывод, что применение светодиодов на Ардуино безгранично и позволяет, имя фантазию, создавать уникальные конструкции.

Практические работы :

1. Мигание светодиода.

2. Бегущие гни.
3. RGB светодиод.
4. Светофор.
5. Азбука Морзе.

Домашнее задание :

1. Что такое Ардуино. Виды Ардуино.
2. Назначение основных устройств Ардуино UNO и MEGA.
3. Общие сведения о параметрах Ардуино. Аналоговые и цифровые каналы.
4. Аналоговый и цифровой сигналы. Дискретизация. Частота дискретизации.
5. Загрузка среды Ардуино. Установка рабочей платы Ардуино.
6. Загрузка драйвера Ардуино. Определение работающего COM порта.
7. Среда Ардуино. Язык Ардуино.
8. Управление средой Ардуино.
9. Программа на Ардуино. Запуск программы.
10. Работа программы с Serial Monitor.
11. Применение Serial Print.
12. Работа Ардуино с цифровыми устройствами.
13. Детали : плата прототипирования, светодиод, резистор.
14. Работа Ардуино с внешним светодиодом.
15. RGB светодиоды.
16. Применение Ардуино и светодиодов. Носимая электроника.