**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)
Кафедра космического приборостроения и систем связи

УТВЕРЖДАЮ

Проректор по учебной работе

_____ О.Г. Локтионова

«___» _____ 2016 г.

# СРЕДА БЫСТРОГО ПРОТОТИПИРОВАНИЯ ДЛЯ МИКРОКОНТРОЛЛЕРОВ AVR

Методические указания по выполнению лабораторной работы по дисциплине
«Схемотехническое проектирование электронных измерительных средств»
для студентов направления подготовки магистров
11.04.03 «Конструирование и технология электронных средств»

**Курск 2016**

УДК 621.316

Составитель О. Г. Бондарь

Рецензент

Кандидат технических наук, профессор кафедры

информационные системы и технологии В.А. Шлыков

**Среда быстрого прототипирования для микроконтроллеров AVR :** методические указания по выполнению лабораторной работы по дисциплине «Схемотехническое проектирование электронных измерительных средств» / Юго-Зап. гос. ун-т; сост.: О. Г. Бондарь. Курск, 2016. 31 с.: ил. 5. табл. 1. Библиогр.: с.24

Приводится описание аппаратно-программных средств, ускоряющих проектирование микроконтроллерных систем. Описаны базовые языковые средства и приёмы работы. Приведен пример использования и варианты заданий.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по специальностям автоматики и электроники (УМО АЭ).

Предназначены для студентов направления подготовки магистров 11.04.03.

Текст печатается в авторской редакции

Подписано в печать Формат 60 84 1/16.
Усл. печ. л. *1,8.* Уч.-изд. л. *1,63*. Тираж экз.
Заказ . Бесплатно.
Курский государственный технический университет.
305040, г. Курск, ул. 50 лет Октября, 94.

## Оглавление

1 Цель работы ..... 4
2 Краткие сведения о платформе ARDUINO ..... 4
3 Аппаратные средства платформы ARDUINO ..... 7
4 Язык программирования ..... 10
5 Интегрированная среда разработки (IDE) ..... 13
5 Средства коммуникации с персональным компьютером ..... 15
6 Порядок работы с ARDUINO ..... 16
7 Подготовка к работе ..... 19
8 Выполнение работы ..... 21
9 Контрольные вопросы ..... 22
10 Содержание отчёта ..... 23
Литература ..... 24
Приложение ..... 25
Константы ..... 25
Уровни сигналов порта HIGH и LOW ..... 25
Настройка цифровых портов на ввод (INPUT) и вывод (OUTPUT) сигналов ..... 25
Специфичные для Freeduino/Arduino функции и объекты ..... 25
Аналоговый ввод/вывод сигнала ..... 27
Работа со временем ..... 28
Последовательная передача данных ..... 29

## 1 Цель работы

Изучение аппаратных средств платформы ARDUINO.
Изучение открытой среды программирования Wiring.
Изучение средств взаимодействия ARDUINO - ПК.

## 2 Краткие сведения о платформе ARDUINO

Создание проектов на основе микроконтроллеров (**МК**) обычно проводится на основе универсальных аппаратных средств, называемых отладочными платами или платформами прототипирования. Такая плата содержит минимально достаточный набор средств поддержки функционирования. Это система электропитания, интерфейс с системой разработки (обычно ПК), средства программирования, средства тактирования, разъёмы для подключения датчиков, исполнительных устройств, внешних интерфейсов. Дополнительно на плату могут устанавливаться датчики, устройства индикации, некоторое количество кнопок. Производители микроконтроллеров обычно предлагают для разработчиков некоторые начальные средства поддержки. Подобные средства выпускаются и производителем 8-и битовых микроконтроллеров на ядре AVR фирмой Atmel. Atmel предлагает бесплатную интегрированную среду разработки (IDE - Integrated development environment) поддерживающую собственные отладочные средства. Однако эти средства имеют высокую стоимость, а их освоение, особенно на стадии начального изучения, не простая задача. На подобной системе собственной разработки проведен цикл лабораторных работ по изучению базовых особенностей архитектуры

микроконтроллеров AVR и языка ассемблера.

С этой точки зрения в лучшую сторону отличается открытый проект ARDUINO. Платформу ARDUINO можно рассматривать как фреймворк. Фреймворк (англ.) — каркас, структура программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. Фреймворк отличается от понятия библиотеки тем, что библиотека может быть использована в программном продукте просто как набор подпрограмм некой функциональности, не влияя на архитектуру программного продукта и не накладывая на неё никаких ограничений. Каркас же диктует правила построения архитектуры приложения, задавая на начальном этапе разработки поведение по умолчанию в соответствии с задаваемыми им требованиями. Хорошо известным примером является продукт Microsoft для разработки ПО - NET Framework.

Следует отметить, что стандартизация на платформе ARDUINO охватывает не только программную среду, но и аппаратуру. Платформа содержит стандартные средства загрузки прикладного программного обеспечения через последовательный асинхронный интерфейс (в первых моделях RS232, а затем через адаптеры USB-RS232TTL), стандартный разъём внутрисхемного программирования, допускает возможность питания от USB и внешнего источника, унифицирована в использовании аналоговых и цифровых выводов, ориентирована изначально на совместимые по выводам микроконтроллеры ATMEGA8, ATMEGA168, ATMEGA328. В дальнейшем усилиями сообщества номенклатура применяемых

контроллеров была расширена. Более того, её популярность привела к переносу на контроллеры других производителей.

Основное отличие платформы от альтернатив заключается в тесной связке с языком программирования названным Wiring. Это C с некоторыми расширениями из C++ и специализированными библиотеками. Они размещаются в директории установки по пути %Arduino%\hardware\arduino\cores\arduino. Поскольку это фреймворк, то всё заранее сконфигурировано:

- Таймер 0 сконфигурирован как счетчик времени, и в зависимости от типа, выбранного МК/платы, реализован API для доступа к функциям времени;
- Сконфигурированы выводы UART, используемого загрузчиком пользовательского программного обеспечения;
- Создан буфер под хранение данных UART, настроены соответствующие прерывания, опять-таки реализован API;
- Загружена библиотека, включающая в себя функции для работы с символами\строками;
- Имеются библиотеки математических функций;
- Существенно упрощена работа с аналоговыми сигналами благодаря библиотеке, поддерживающей АЦП и ШИМ (никакого дополнительного конфигурирования при этом не требуется);
- Весьма обширны сторонние библиотеки. Имеются типовые примеры использования. При компиляции в конечный код включаются только используемые библиотеки.

## 3 Аппаратные средства платформы ARDUINO

В настоящее время платформа представлена большим количеством оригинальных плат и их клонов. Отличия между ними не носят принципиального характера и состоят в основном в особенностях реализации поддержки и типе микроконтроллера. Изучение платформы будет проводится на примере одной платы – Freeduino 2009 являющейся клоном оригинала Arduino Duemilanove. Внешний вид платы приведен на рисунке 1, а схема на рисунке 2.

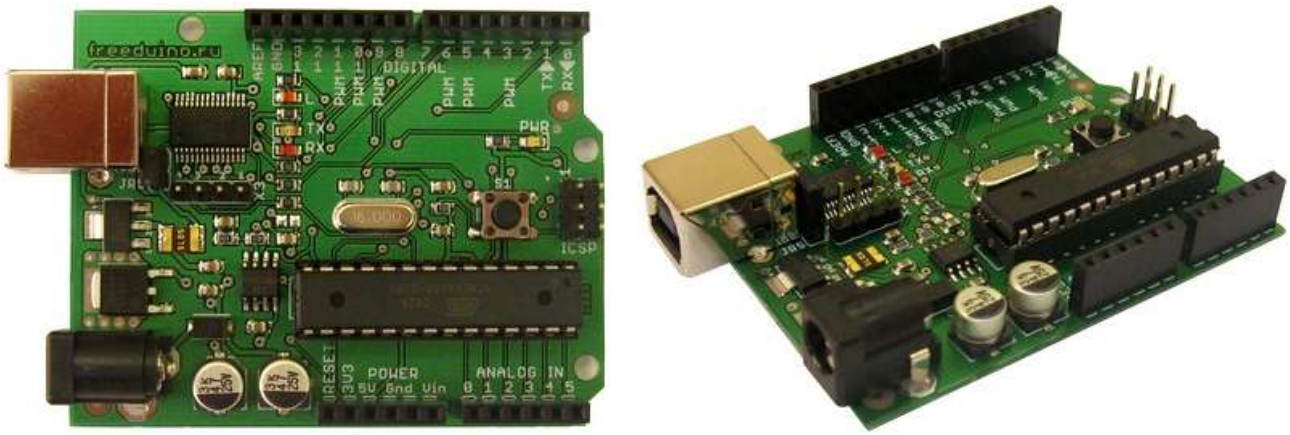


Рисунок 1 – Размещение элементов на плате

Основой платы является микроконтроллер Atmega 168 или 328, в DIP – корпусе, установленный в панельку. Это позволяет осуществить быструю замену МК. На плате имеется линейный стабилизатор напряжения на 5 В и силовой полевой транзистор для автоматической коммутации напряжения питания (USB – внешний источник). Они размещены между разъёмами USB и внешнего питания. Выбор источника питания осуществляется автоматически с помощью компаратора на ИС5, сравнивающего напряжение на

разъёме питания с опорным напряжением, формируемым из напряжения шины USB.

Обмен данными с персональным компьютером реализуется с помощью интерфейса USB. Для этого имеется интегральная микросхема ИС2 FT232RL – конвертора интерфейсов USB-RS232. Она расположена у разъёма USB. Микроконтроллер обменивается данными с конвертором по интерфейсу RS232TL. Специальный драйвер компьютера реализует виртуальный последовательный коммуникационный порт. Помимо обмена данными через разъём USB осуществляется программирование МК. Технология программирования основана на применении загрузчика (BootLoader). Это специальная технология, позволяющая МК AVR одновременно с выполнением программы размещённой в одном блоке программной памяти выполнять загрузку других блоков. В данном случае загрузчик поддерживает интерфейс асинхронный последовательный интерфейс, но возможно применять и другие интерфейсы (I2C, SPI). Помимо этого, имеется возможность программирования микроконтроллера и через стандартный для AVR интерфейс внутрисхемного программирования.

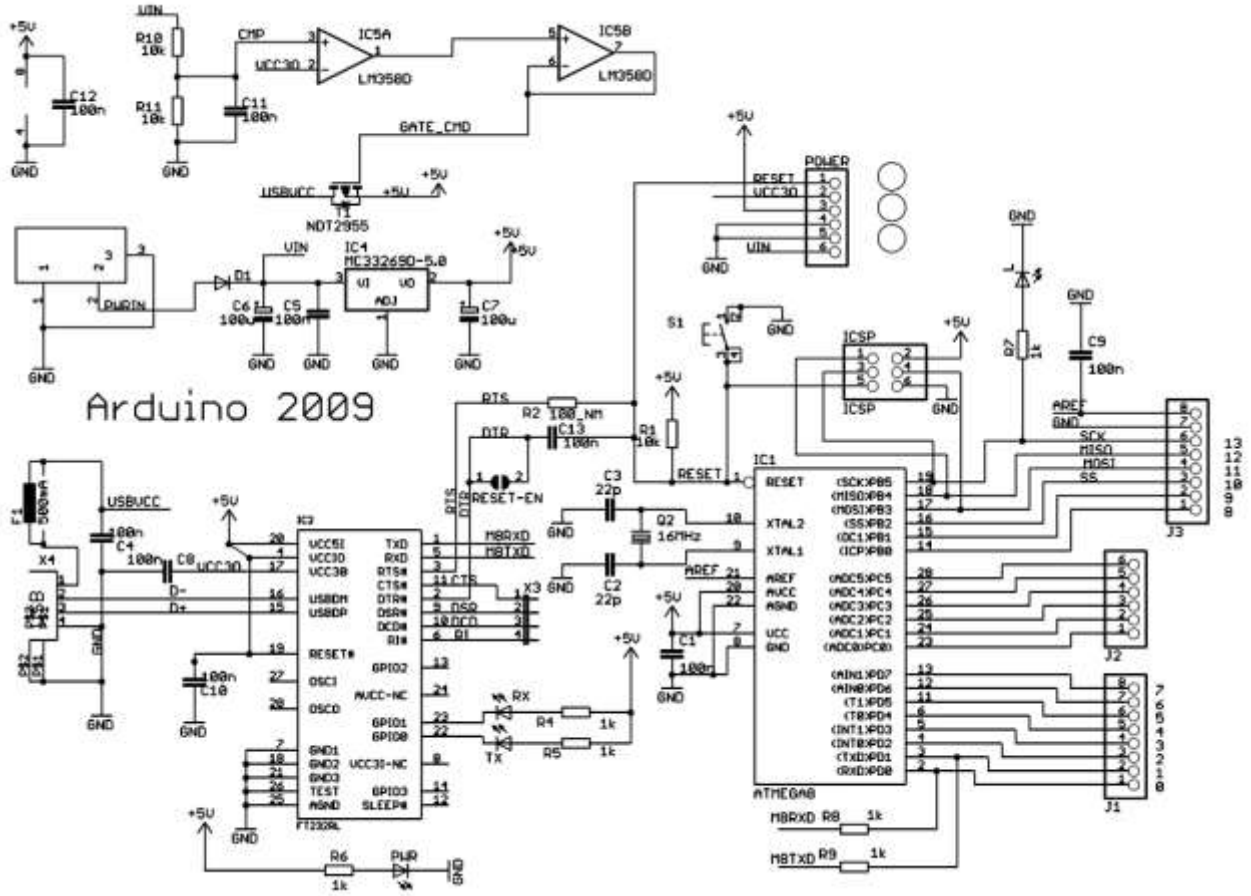


Рисунок 2 – Схема платы Freeduino 2009

Тактирование МК осуществляется с частотой 16 МГц. Частота задаётся кварцевым резонатором Q2. На плате установлено четыре светодиода. Один индикатор напряжения питания, два – индицируют обмен по последовательному интерфейсу (Rx,Tx). Светодиод, подключенный к выводу PB5, используется в простейших демонстрационных проектах. Все внешние устройства подключаются через разъёмы расширения, расположенные по краям платы. Назначение выводов на платах стандартизовано, а их соответствие контактам МК представлено на рисунке 3.

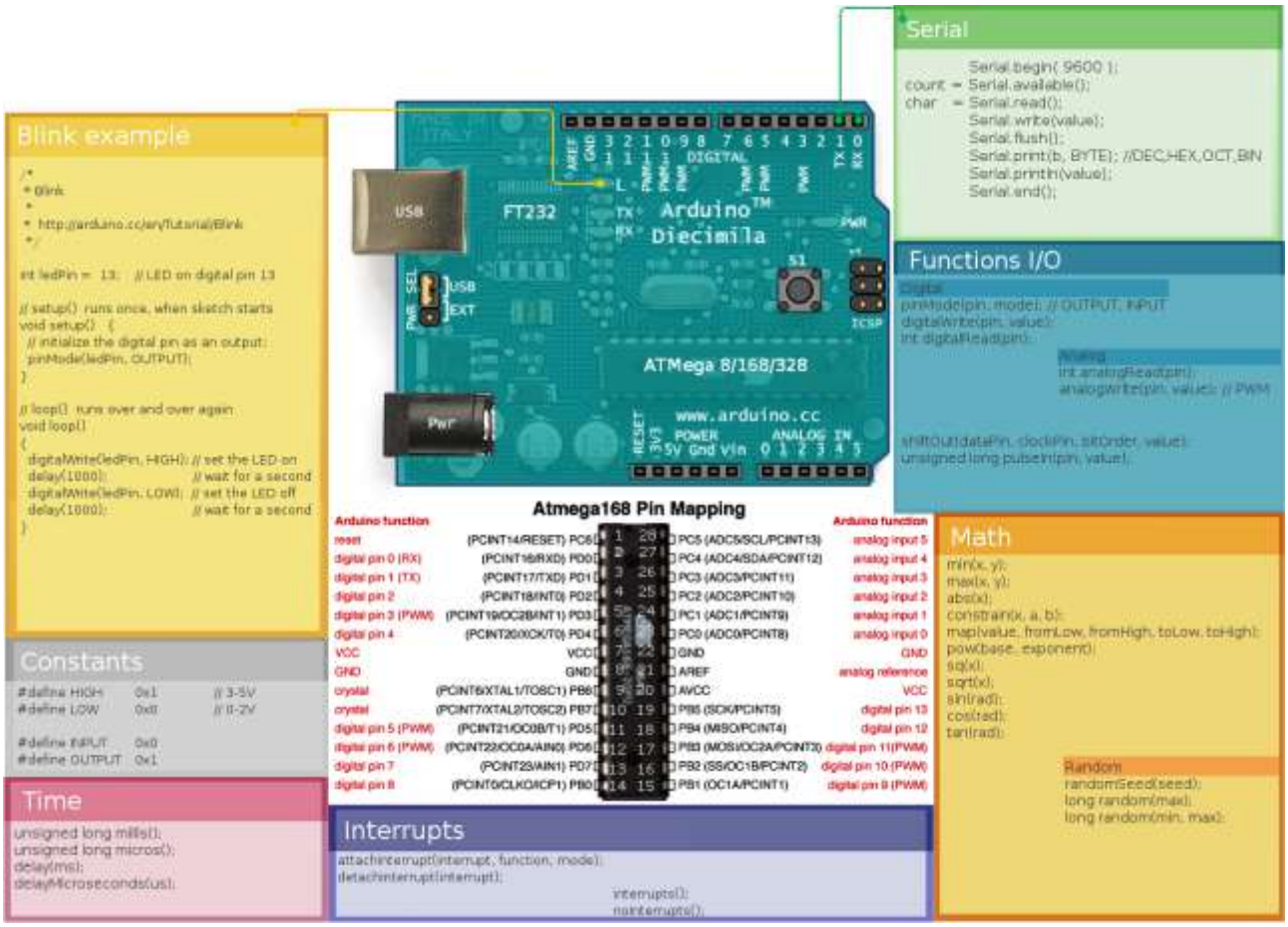


Рисунок 3 – Именование контактов платы Freeduino

На случай аварийной ситуации имеется кнопка сброса S1.

## 4 Язык программирования

Платформа Arduino программируется на языке программирования, основаном на C/C ++, и позволяет использовать любые его функции. Программы для ARDUINO преобразуются в программу на языке C/C++, и затем компилируются компилятором AVR-GCC. Так что фактически, используется специализированный для микроконтроллеров AVR вариант C/C++. Как отмечено, набор

базовых библиотек существенно упрощает доступ к находящейся «на борту» микроконтроллера периферии. Например, для инициализации последовательного порта на скорости 9600 бит в секунду достаточно написать:

```
Serial.begin(9600);
```

При использовании C/C++ это выглядит примерно так:

```
UBRR0H = ((F_CPU / 16 + 9600 / 2) / 9600 - 1) >> 8;
UBRR0L = ((F_CPU / 16 + 9600 / 2) / 9600 - 1);
sbi(UCSR0B, RXEN0);
sbi(UCSR0B, TXEN0);
sbi(UCSR0B, RXCIE0);
```

Структура программы состоит из двух основных функций: setup() и loop(). Первая исполнятся один раз при запуске программы и предназначена для инициализации переменных и установки режимов работы используемого периферийного оборудования. Функция loop() и представляет основной функционал программы, повторяющийся бесконечно.

Типичную структуру программы можно продемонстрировать на примере входящем в набор демонстрационных программ. Текст программы «BLINK» (программа ARDUINO называется «скетч») приведен ниже:

```
/*
  Blink – название программы
  Светодиод включить на одну секунду, выключить на одну секунду, повторять бесконечно. Светодиод подключён к 13 контакту разъёма ARDUINO.
 */

int led = 13; // целое с именем «led» обозначает 13-й контакт платы

// программа инициализации исполняемая при старте
void setup() {
  pinMode(led, OUTPUT);         // цифровой порт «led» на вывод
}
// бесконечный повтор включить-выключить светодиод
void loop() {
  digitalWrite(led, HIGH);      // включить (высокий уровень)
  delay(1000);                  // секундная задержка
  digitalWrite(led, LOW);       // выключить (низкий уровень)
  delay(1000);                  // секундная задержка
}
```

Многострочный комментарий в начале программы заключён в скобки - **/*** ***/**. Для выделения однострочных комментариев используется двойной слеш - //. Завершение строки комментария представлено кодом перехода на другую строку (символ не отображается).

Раздел инициализации переменных сведен к единственному определению номера контакта, к которому подключен светодиод, как целого значения int переменной led. Программа инициализации

ориентирует порт микроконтроллера **PB5**, соответствующий цифровому выводу платы с именем **digital pin13** на вывод.

Основная петля программы – 4 оператора заключённые в фигурные скобки реализуют весь функционал программы. Фигурные скобки определяют начало и конец функции или блока выражений. Каждой открывающей скобке должна соответствовать закрывающая скобка. В среде разработки можно выделить скобку или установить курсор за ней, и пара скобок будет подсвечена. Это облегчает контроль парности скобок.

Следует обратить внимание на то, что каждое выражение завершается **точкой с запятой**. Кроме того, это разделитель для элементов цикла **for**.

Более подробно описание языка приведено в приложении и [Brian W. Evans. Arduino блокнот программиста. - 2007]. Поскольку среда программирования включает в свой состав большое количество демонстрационных примеров, то их изучение наиболее быстрый способ овладения языком.

## 5 Интегрированная среда разработки (IDE)

Вызов среды разработки может быть осуществлён через ярлык программы или щелчком по скетчу прикладной программы. Окно **IDE** представлено на рисунке 4. Его вид стандартен для Windows приложений: строка меню, набор основных инструментов для быстрого вызова основных функций, окно встроенного редактора текстов, область сообщений (внизу).

Рабочая версия, используемая в лабораторном цикле 1.0.5. Программа, написанная в среде Arduino, называется скетч. Скетч пишется в текстовом редакторе, имеющем инструменты вырезки/вставки, поиска/замены текста.

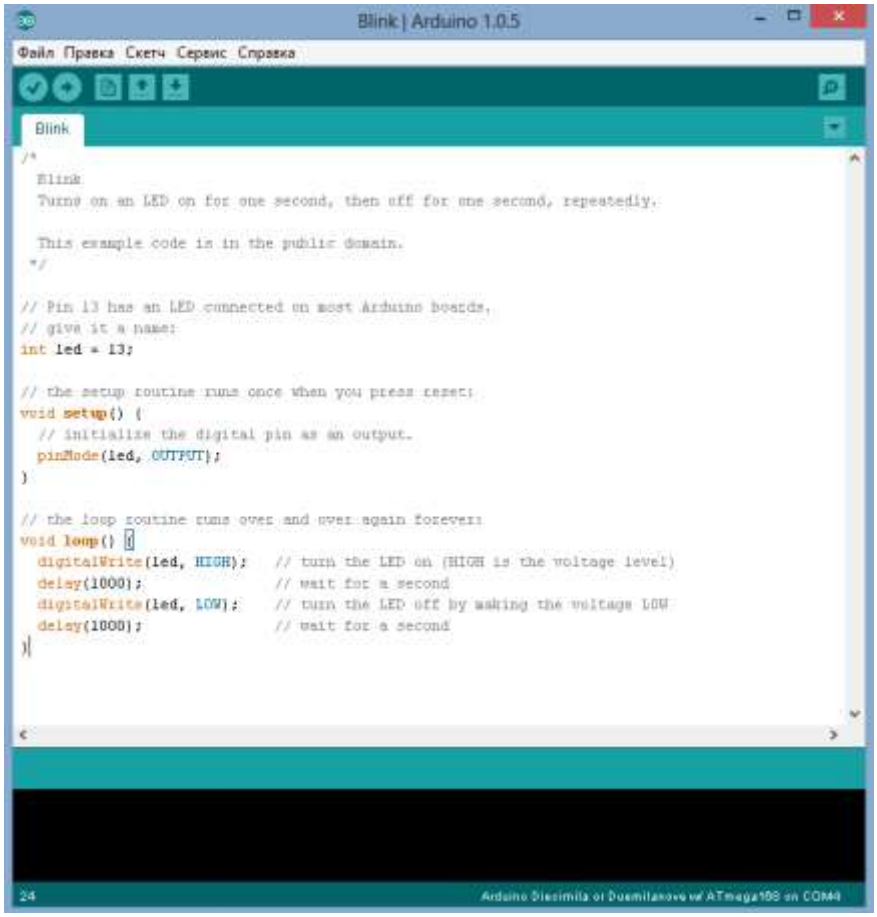


Рисунок 4 – Вид окна среды разработки

Во время сохранения и экспорта проекта в области сообщений появляются пояснения, также могут отображаться возникшие ошибки.

Кнопки панели инструментов позволяют проверить и записать программу, создать, открыть и сохранить скетч, запустить мониторинг последовательного канала обмена данными. Основное

меню и панель инструментов полностью русифицированы. При подведении курсора к инструменту показывается его функция.

Инструмент «**загрузить**» запускает компиляцию скетча и при отсутствии ошибок загружает программу в микроконтроллер.

Инструмент «**проверить**» производит только компиляцию без загрузки.

При первом запуске IDE необходимо осуществить настройку среды через пункт меню «Сервис». Настройка начинается с выбора платформы (пункт меню «**плата**»). Следует выбрать пункт «Arduino Diecimila or Duemilanove w/ATmega 168». Выбор платформы влияет на параметры на основные параметры: скорость ЦП и скорость передачи данных, используемые при компиляции и загрузке скетчей, на настройки загрузчика МК. Некоторые характеристики платформ различаются только настройками загрузчика. Другой важный пункт этого меню - «последовательный порт» позволяет выбрать номер активного виртуального коммуникационного порта, через который будет осуществляться взаимодействие платформы и персонального компьютера. Стандартная скорость при загрузке откомпилированной программы для выбранной платформы 19200 бит/с.

## 5 Средства коммуникации с персональным компьютером

Достоинством платформы является не только полная автоматизация процесса загрузки программы, но и возможность коммуникации через тот же самый интерфейс с персональным компьютером.

Скорость загрузки программы и скорость обмена данными между компьютером и платформой устанавливаются независимо. При этом скорость обмена данными с ПК может быть существенно выше скорости загрузки программы и определяется при инициализации обмена в программе. В этом случае необходимо выбрать такую же скорость обмена данными и в прикладной программе на ПК. В лабораторном цикле для обмена с МК используется терминальная программа **Terminal V1.9b.** Следует отметить, что в пункте меню «сервис» имеется возможность выбора встроенного терминала среды **IDE**. Но предложенная выше терминальная программа обладает лучшей функциональностью. Естественно, что для организации обмена со стороны ПК могут использоваться любые программные продукты, поддерживающие технологию виртуальных коммуникационных портов. Одним из таких эффективных средств является среда LabView National Instrument.

**Важно.** *Для исключения конфликтов между терминальными программами в режиме программирования следует переводить программу Terminal V1.9b. в состояние Disconnect (разъединить)*.

## 6 Порядок работы с ARDUINO

Ниже кратко представлена последовательность работы с платформой.

1. Подключаем плату к разъёму USB персонального компьютера. При этом допускается «горячее подключение». На плате должен светиться зелёный индикатор питания.
2. Запускаем IDE щелчком по ярлыку программы или готовому скетчу.
3. Создаём, копируем, редактируем программу и сохраняем в выбранном месте. Альтернативно через пункт меню файл «примеры» выбираем демонстрационную программу. В первой части лабораторной работы это будет скетч - 01 Basics/BLINC.
4. В пункте меню «**сервис»** выбираем «Плата/ Arduino Diecimila or Duemilanove w/ATmega 168»**.**
5. В пункте меню «**сервис**» выбираем последовательный порт и щёлкаем по выбранному номеру порта. Обычно выбор осуществляется автоматически и предлагается единственный номер порта.
6. Для сохранения скетча в меню «Файл/Сохранить как» выбираем имя и место. Следует иметь в виду, что сохранение произойдёт в одноимённую папку. Имя не должно содержать символов кириллицы.
7. Для загрузки скетча в МК следует нажать на панели инструментов «Загрузить» или в меню выбрать Файл/Загрузить. При этом произойдёт компиляция проекта. Внизу в окне появится сообщение об ошибках или корректном завершении компиляции. Далее произойдёт загрузка сопровождаемая миганием светодиодов приёмника и передатчика

последовательного интерфейса. По окончании программа запустится автоматически.

8. Сообщения МК можно наблюдать с помощью терминала, если они предусмотрены в программе. Это может быть встроенный монитор, или более функциональный Terminal v1.9b. В любом случае монитор должен быть настроен на номер порта и скорость обмена. Скорость обмена задаётся в программе – скетче строкой Serial.begin(115200); цифра в скобках может быть 4800, 9600, 19200, 38400, 56000, 115200.
9. Настройка встроенного монитора осуществляется открытием его окна в меню Сервис/Монитор порта. Внизу справа нажатием на стрелку xxxxbaud открывается выпадающее меню и в нём выбирается скорость.
10. Настройка Terminal v1.9b (рисунок 5) осуществляется вначале выбором активного порта слева вверху в окне COM Port – его номер должен совпадать с выбранным в меню Ардуино. Далее рядом в разделе **Baud rate** следует отметить выбранную скорость. Последнее действие - нажать слева вверху **Connect.** При несовпадении скоростей монитора и платы на экране будут неосмысленные сообщения, противном случае – сообщения, предусмотренные в разработанной программе.
11. Нельзя держать два запущенных монитора. Терминал нужно отключать во время загрузки скетча, нажав в левом верхнем углу **Disconnect.**

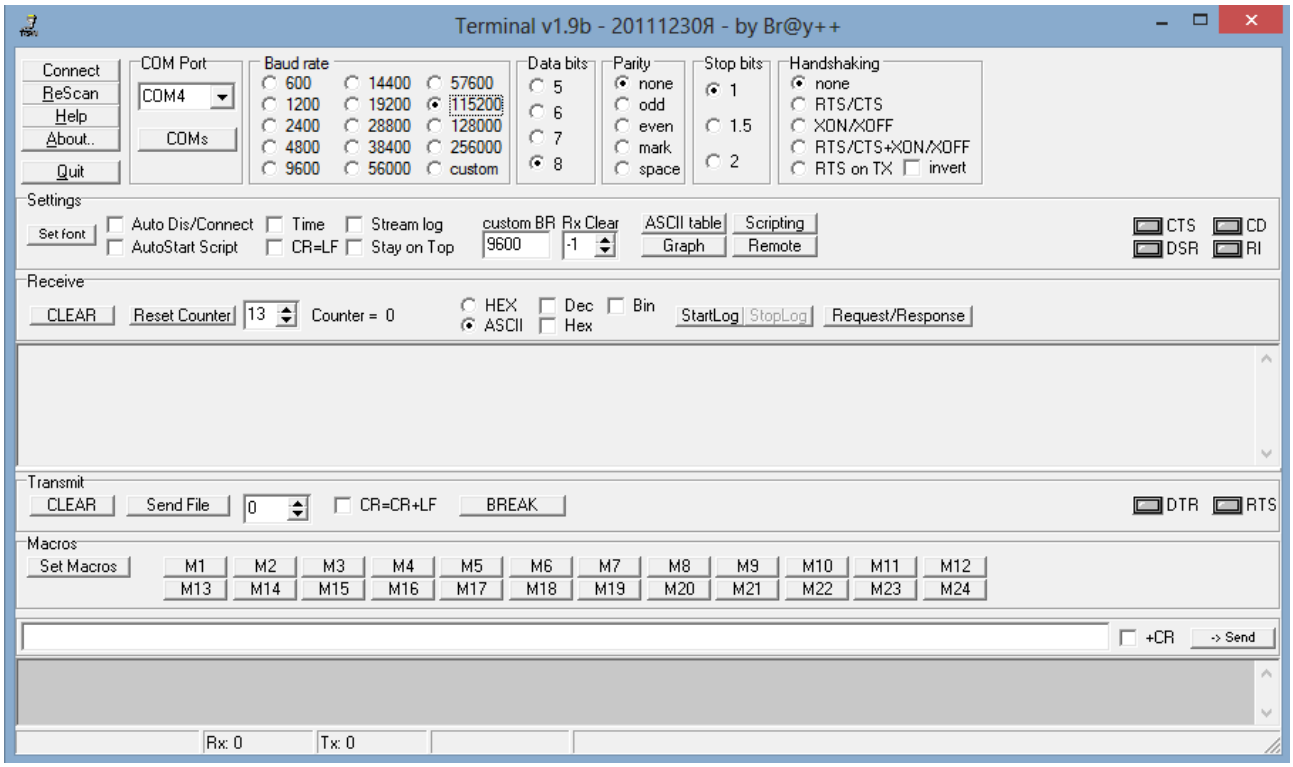


Рисунок 5 – Окно терминальной программы Terminal v1.9b

## 7 Подготовка к работе

Свободно распространяемая среда разработки ARDUINO может быть получена из Интернет или с сервера кафедры. Целесообразно установить среду на домашний компьютер/ноутбук, т.к. это позволит легко усвоить основные приёмы работы со средой.

Познакомиться по описанию с платой Freeduino Duemilanove и основными её характеристиками и характеристиками процессора ATmega 168.

Изучить меню и панель инструментов IDE ARDUINO.

Изучить представленный в методических указаниях текст примера демонстрационной программы. Обратить внимание на особенности синтаксиса языка.

Ознакомиться с кратким описанием языка программирования, представленным в приложении.

Изучить конфигурирование и операторы вывода сообщений в последовательный порт: **Serial.begin, Serial.print, Serial.println**.

Изучить окно терминальной программы. Найти все приведенные в методических указаниях поля и средства управления конфигурацией.

Изменить исходную программу демонстрационного примера в соответствии с вариантом задания.

Таблица 1. Варианты заданий

| № варианта | Последовательность и интервалы вкл/выкл. светодиода | Выдаваемые сообщения |
|---|---|---|
| 1 | вкл -1с, выкл-2с – далее повторять в цикле | LED ON X s LED OFF Y s (время состояния)<br>N=1<br>N=2<br>… 1, 2, 3.. номер цикла |
| 2 | вкл -2с, выкл-1с,<br>вкл 3с, выкл 1с – далее повтор | LED ON X s<br>LED OFF Y s<br>N=1<br>N=2<br>… 1, 2, 3.. номер выключения |
| 3 | вкл -2с, выкл-2с,<br>вкл -1с – далее повтор | ON=X s, OFF=Y s<br>1, 2, 3, 4…. номер шага |
| 4 | вкл -1с, выкл-2с, вкл -1с, вкл – 3с – повтор | ON X s<br>OFF Y s<br>1 2 3 4 … номер цикла |
| 5 | Вкл -3с, выкл-1с – далее повтор | STATUS<br>On<br>Of далее повторять |
| 6 | по варианту 3 | по варианту 1 |

| | | |
|---|---|---|
| 7 | по варианту 4 | по варианту 2 |
| 8 | по варианту 5 | по варианту 3 |
| 9 | по варианту 1 | по варианту 4 |
| 10 | по варианту 2 | по варианту 5 |
| 11 | по варианту 3 | по варианту 1 |
| 12 | по варианту 5 | по варианту 2 |
| 13 | по варианту 4 | по варианту 3 |
| 14 | по варианту 3 | по варианту 4 |
| 15 | по варианту 2 | по варианту 5 |

## 8 Выполнение работы

1. Запустить **IDE** и загрузить скетч примера **BLINC.**
2. Конфигурировать **IDE** в соответствии с разделом 6 МУ.
3. Проверить загрузочный скетч и выписать сообщения **IDE.**
4. Загрузить скомпилированный скетч в МК и убедиться в его работоспособности.
5. Закрыть среду разработки.
6. Открыть её повторно и ввести текст модифицированной программы.
7. Проверить скетч и выписать сообщения **IDE**. При наличии сообщений об ошибках внести исправления в программу.
8. При отсутствии сообщений об ошибках в исправленной программе подготовить терминальную программу для работы. Для этого следует запустить её, установить правильной номер коммуникационного порта и скорость работы, в соответствии с выбранной в программе. Отключить терминальную программу (**Disconnect**).

9. Загрузить скомпилированный скетч в МК.
10. По окончании загрузки включить терминальную программу (**Connect**).
11. Нажать кнопку сброс на плате ARDUINO и наблюдать за сообщениями терминальной программы не менее 15 с.
12. Отключить терминальную программу (**Disconnect**).
13. Скопировать и вставить в отчёт сообщения в поле приёма терминала.
14. Проанализировать сообщения на соответствие заданию.
15. При несоответствии внести коррективы в программу и повторить п.п.7-15.
16. Измерить время цикла программы по выдаваемым сообщениям и проверить его на соответствие варианту задания.

## 9 Контрольные вопросы

1. В чём состоят основные преимущества платформы прототипирования Arduino в сравнении с альтернативными средствами разработки систем?
2. Каковы самые существенные недостатки платформы Arduino?
3. Какие ограничения накладывает платформа на применяемые МК?
4. Какие из перечисленных далее элементов являются обязательными при применении идеологии платформы Arduino? Микроконтроллер с ядром AVR Atmel, стабилизатор напряжения +5В, преобразователь интерфейса USB-RS232TTL, загрузчик программного обеспечения через интерфейс

RS232TTL, устройство программирования через последовательный интерфейс SPI.

5. Предоставляет ли платформа Arduino средства управления скоростью приёма и передачи данных через последовательный порт?
6. С какой дискретностью задаются и измеряются временные интервалы платформой Arduino? Каковы максимальные значения этих интервалов?

## 10 Содержание отчёта

Отчёт должен содержать:

1. Титульный лист
2. Содержание
3. Цель работы
4. Краткие сведения о платформе прототипирования.
5. Схему платы с кратким описанием назначения элементов.
6. Описание изученных операторов работы с коммуникационным последовательным портом.
7. Исходный и изменённый текст программы с комментариями.
8. Сообщения среды разработки при каждой итерации программы.
9. Копии терминальных сообщений, полученных при работе программы.
10. Результаты измерений длительности цикла программы.
11. Ответы на контрольные вопросы.

**Литература**

1. Соммер, У. Программирование микроконтроллерных плат Arduino/Freeduino [Текст] / У. Соммер. - СПб.: БХВ-Петербург, 2012. -256 с.
2. Брайен Эванс. Arduino блокнот программиста [Электронный документ] // Режим доступа - http://robocraft.ru/files/books/arduino_notebook_rus_v1-1.pdf
3. Руководство по освоению Arduino [Электронный документ] // Режим доступа - https://solarbotics.com/download.php?file=1540

# Приложение
# Краткое описание языка программирования

### Константы

Константы – предопределенные значения. Они используются, чтобы делать программы более легкими для чтения. Мы классифицируем константы в группах.

### 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.

### 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
```

### Специфичные для Freeduino/Arduino функции и объекты

#### *Цифровой ввод/вывод*

**pinMode**

Вызов:

pinMode (порт, режим);

Описание:
Конфигурирует указанный порт на ввод или вывод сигнала.
Параметры:
порт – номер порта, режим которого Вы желает установить (значение целого типа от 0 до 13).
режим – либо INPUT (ввод) либо OUTPUT (вывод).
Пример:

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

Примечание:
Аналоговые входы могут использоваться как цифровые входы/выходы, при обращении к ним по номерам с 14 (аналоговый вход 0) по 19 (аналоговый вход 5)

digitalWrite
Вызов:

digitalWrite(порт, значение);

Описание:
Устанавливает высокий (HIGH) или низкий (LOW) уровень напряжения на указанном порте.
Параметры:
порт: номер порта
значение: HIGH или LOW
Пример:

```
digitalWrite(13, HIGH);  // выставляем 13й вывод в «высокое» состояние
```

**digitalRead**
Вызов:

value = digitalRead (порт);

Описание:
Считывает значение на указанном порту
Параметры:
порт: номер опрашиваемого порта
Возвращаемое значение: возвращает текущее значение на порту (HIGH или LOW) типа int
Пример:

```
int val;
val = digitalRead(12);  // опрашиваем 12й вывод
```

Примечание:
Если к считываемому порту ничего не подключено, то функция digitalRead () может беспорядочно возвращать значения HIGH или LOW.

**Аналоговый ввод/вывод сигнала**

**analogRead**
Вызов:

value = analogRead(порт);

Описание:
Считывает значение с указанного аналогового порта. Freeduino содержит 6 каналов, аналого-цифрового преобразователя на 10 битов каждый. Это означает, что входное напряжения от 0 до 5В преобразовывается в целочисленное значение от 0 до 1023. Разрешающая способность считывания составляет: 5 В/1024 значений = 0,004883 В/значение (4,883 мВ). Требуется приблизительно 100 μС (0.0001 С), чтобы считать значение аналогового ввода, так что максимальная скорость считывания - приблизительно 10000 раз в секунду.
Параметры:
порт: номер опрашиваемого аналогового входа
Возвращаемое значение: возвращает число типа int в диапазоне от 0 до 1023, считанное с указанного порта.
Пример:

```
int val;
val = analogRead(0);  // считываем значение на 0м аналоговом входе
```

Примечание:
Аналоговые порты по умолчанию определенны на ввод сигнала и в отличие от цифровых портов их не требуется конфигурировать с помощью вызова функции pinMode.

**analogWrite**
Вызов:

analogWrite(порт, значение);
Описание:

Выводит на порт аналоговое значение. Эта функция работает на: 3, 5, 6, 9, 10, и 11 цифровых портах.
Может применяться для изменения яркости светодиода, для управления двигателем и т.д. После вызова функции analogWrite, соответствующий порт начинает работать в режиме широтно-импульсного модулирования напряжения до тех пор, пока не будет следующего вызова функции analogWrite (или функций digitalRead / digitalWrite на том же самом порте).

Параметры:
порт: номер аналогового выхода
значение: целочисленное между 0 и 255. Значение 0 генерирует 0 В на указанном порте; значение 255 генерирует +5 В на указанном порте. Для значений между 0 и 255, порт начинает быстро чередовать уровень напряжения 0 и +5 В - чем выше значение, тем, более часто порт генерирует уровень HIGH (5 В).
Пример:

```
analogWrite(9, 128);// устанавливает на 9 контакте значение эквивалентное 2,5В
```

Примечание:

Нет необходимости вызвать функцию pinMode, чтобы установить порт на вывод сигналов перед вызовом функции analogWrite.

Частота генерирования сигнала – приблизительно 490 Гц.

**Работа со временем**

**millis**

Вызов:

```
time = millis();
```

Описание:

Возвращает число миллисекунд, с момента исполнения текущей программы. Счетчик переполнится и обнулится приблизительно через 9 часов.

Возвращаемое значение: возвращает значение типа unsigned long

Пример:

```
unsigned long time; // объявление переменной time типа unsigned long
time = millis(); // передача количества миллисекунд
```

**delay**

Вызов:

```
delay(время_мс);
```

Описание:

Приостанавливает программу на заданное число миллисекунд.

Параметры:

время_мс – время задержки программы в миллисекундах

Пример:

```
delay(1000);  //пауза 1 секунда
```

**delayMicroseconds**

Вызов:

```
delayMicroseconds(время_мкс);
```

Описание:

Приостанавливает программу на заданное число микросекунд.

Параметры:

время_мкс – время задержки программы в микросекундах

Пример:

```
delayMicroseconds(500);  //пауза 500 микросекунд
```

**pulseIn**

Вызов:

```
pulseIn(порт, значение);
```

Описание:

Считывает импульс (высокий или низкий) с цифрового порта и возвращает продолжительность импульса в микросекундах.

Например, если параметр «значение» при вызове функции установлен в HIGH, то pulseIn() ожидает, когда на порт поступит высокий уровень сигнала. С момента его поступления начинается отсчет времени до тех пор, пока на порт не поступит низкий уровень сигнала. Функция возвращает длину импульса (высокого уровня) в микросекундах. Работает с импульсами от 10 микросекунд до 3 минут. Обратите внимание, что эта функция не будет возвращать результат, пока импульс не будет обнаружен.

Параметры:

порт: номер порта, с которого считываем импульс

значение: тип импульса HIGH или LOW

Возвращаемое значение: возвращает длительность импульса в микросекундах (тип int)

Пример:

```
int duration; // объявление переменной duration типа int
duration = pulseIn(pin, HIGH); // измеряем длительность импульса
```

**Последовательная передача данных**

Arduino имеет встроенный контроллер для последовательной передачи данных, который может использоваться как для связи между Arduino устройствами, так и для связи с компьютером. На компьютере соответствующее соединение представлено USB COM-портом.

Связь происходит по цифровым портам 0 и 1, и поэтому Вы не сможете использовать их для цифрового ввода/вывода если используете функции последовательной передачи данных.

**Serial.begin**

Вызов:

```
Serial.begin(скорость_передачи);
```

Описание:

Устанавливает скорость передачи информации COM порта битах в секунду для последовательной передачи данных. Для того чтобы поддерживать связь с компьютером, используйте одну из этих нормированных скоростей: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, или 115200. Также Вы можете определить другие скорости при связи с другим микроконтроллером по портам 0 и 1.

Параметры:

скорость_передачи: скорость потока данных в битах в секунду.

Пример:

```
Serial.begin(9600); //устанавливаем скорость 9600 бит/сек
```

**Serial.available**

Вызов:

```
count = Serial.available();
```

Описание:

Принимаемые по последовательному порту байты попадают в буфер микроконтроллера, откуда Ваша программа может их считать. Функция возвращает количество накопленных в буфере байт. Последовательный буфер может хранить до 128 байт.

Возвращаемое значение:

Возвращает значение типа int – количество байт, доступных для чтения, в последовательном буфере, или 0, если ничего не доступно.

Пример:

```
if (Serial.available() > 0) { // Если в буфере есть данные
  // здесь должен быть прием и обработка данных
 }
```

**Serial.read**

Вызов:

```
char = Serial.read();
```

Описание:

Считывает следующий байт из буфера последовательного порта.

Возвращаемое значение:

Первый доступный байт входящих данных с последовательного порта, или -1 если нет входящих данных.

Пример:

```
incomingByte = Serial.read(); // читаем байт
```

**Serial.flush**

Вызов:

```
Serial.flush();
```

Описание:

Очищает входной буфер последовательного порта. Находящиеся в буфере данные теряются, и дальнейшие вызовы Serial.read() или Serial.available() будут иметь смысл для данных, полученных после вызова Serial.flush().

Пример:

```
Serial.flush(); // Очищаем буфер – начинаем прием данных «с чистого листа»
```

**Serial.print**

Описание:

Вывод данных на последовательный порт.

Параметры:

Функция имеет несколько форм вызова в зависимости от типа и формата выводимых данных.

Serial.print(b, DEC) выводит ASCII-строку - десятичное представление числа b.

```
int b = 79;
Serial.print(b, DEC); //выдаст в порт строку «79»
```

Serial.print(b, HEX) выводит ASCII-строку - шестнадцатиричное представление числа b.

```
int b = 79;
Serial.print(b, HEX); //выдаст в порт строку «4F»
```

Serial.print(b, OCT) выводит ASCII-строку - восьмеричное представление числа b.

```
int b = 79;
Serial.print(b, OCT); //выдаст в порт строку «117»
```

Serial.print(b, BIN) выводит ASCII-строку - двоичное представление числа b.

```
int b = 79;
Serial.print(b, BIN); //выдаст в порт строку «1001111»
```

Serial.print(b, BYTE) выводит младший байт числа b.

```
int b = 79;
Serial.print(b, BYTE); //выведет число 79 (один байт). В мониторе
                       //последовательного порта получим символ «O» - его
                       //код равен 79
```

Serial.print(str) если str – строка или массив символов, побайтно передает str на COM-порт.

```
char bytes[3] = {79, 80, 81};    //массив из 3 байт со значениями 79,80,81
Serial.print("Here our bytes:"); //выводит строку «Here our bytes:»
Serial.print(bytes);            //выводит 3 символа с кодами 79,80,81 –
                                //это символы «OPQ»
```

Serial.print(b) если b имеет тип byte или char, выводит в порт само число b.

```
char b = 79;
Serial.print(b); //выдаст в порт символ «O»
```

Serial.print(b) если b имеет целый тип, выводит в порт десятичное представление числа b.

```
int b = 79;
Serial.print(b); //выдаст в порт строку «79»
```

**Serial.println**

Описание:

Функция Serial.println аналогична функции Serial.print, и имеет такие же варианты вызова. Единственное отличие заключается в том, что после данных дополнительно выводятся два символа – символ возврата каретки (ASCII 13, или '\r') и символ новой линии (ASCII 10, или '\n').

Пример 1 и пример 2 выведут в порт одно и то же:

Пример 1:

```
int b = 79;
Serial.print(b, DEC); //выдаст в порт строку «79»
Serial.print("\r\n"); //выведет символы "\r\n" – перевод строки
Serial.print(b, HEX); //выдаст в порт строку «4F»
Serial.print("\r\n");//выведет символы "\r\n" – перевод строки
```

Пример 2:

```
int b = 79;
Serial.println(b, DEC); //выдаст в порт строку «79\r\n»
Serial.println(b, HEX); //выдаст в порт строку «4F\r\n»
```

В мониторе последовательного порта получим:

79

4F