Платформа Arduino

- **Arduino** – аппаратная вычислительная платформа, основными компонентами которой являются простая плата ввода/вывода и среда разработки на языке Processing/Wiring. Arduino может использоваться как для создания автономных интерактивных объектов, так и подключаться к программному обеспечению, выполняемому на компьютере

Аппаратные средства Arduino

- Аппаратные средства Arduino включают популярные и доступные комплектующие
изделия. Поэтому принцип работы системы понятен, настройка схемы под требования разработчика проста и обеспечена возможность дальнейшей модификации. Основа контроллер ATmega компании Atmel широко распространенного 8-разрядного семейства AVR. К нему добавляется узел электропитания и последовательный интерфейс.
В последних версиях Arduino имеется USB-интерфейс. Через него происходит загрузка программ пользователя

Возможности Arduino

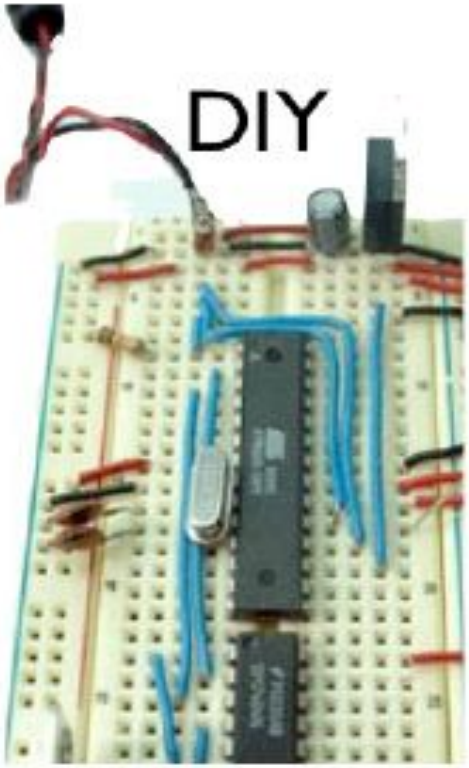
- 16 кБ флэш-памяти программ
- 1 кБ оперативной памяти
- 16 МГц (Apple II: 1 МГц)
- Входы и выходы
  - 13 цифровых входов/выходов
  - 5 аналоговых входов
  - 6 аналоговых выходов*
- Полностью автономна: однажды запрограммированная, не нуждается в компьютере

Разнообразие плат Arduino

LilyPad
(для одежды)

USB

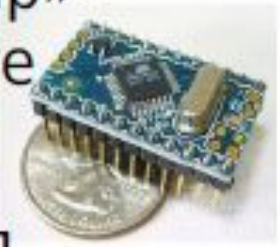
DIY



Bluetooth

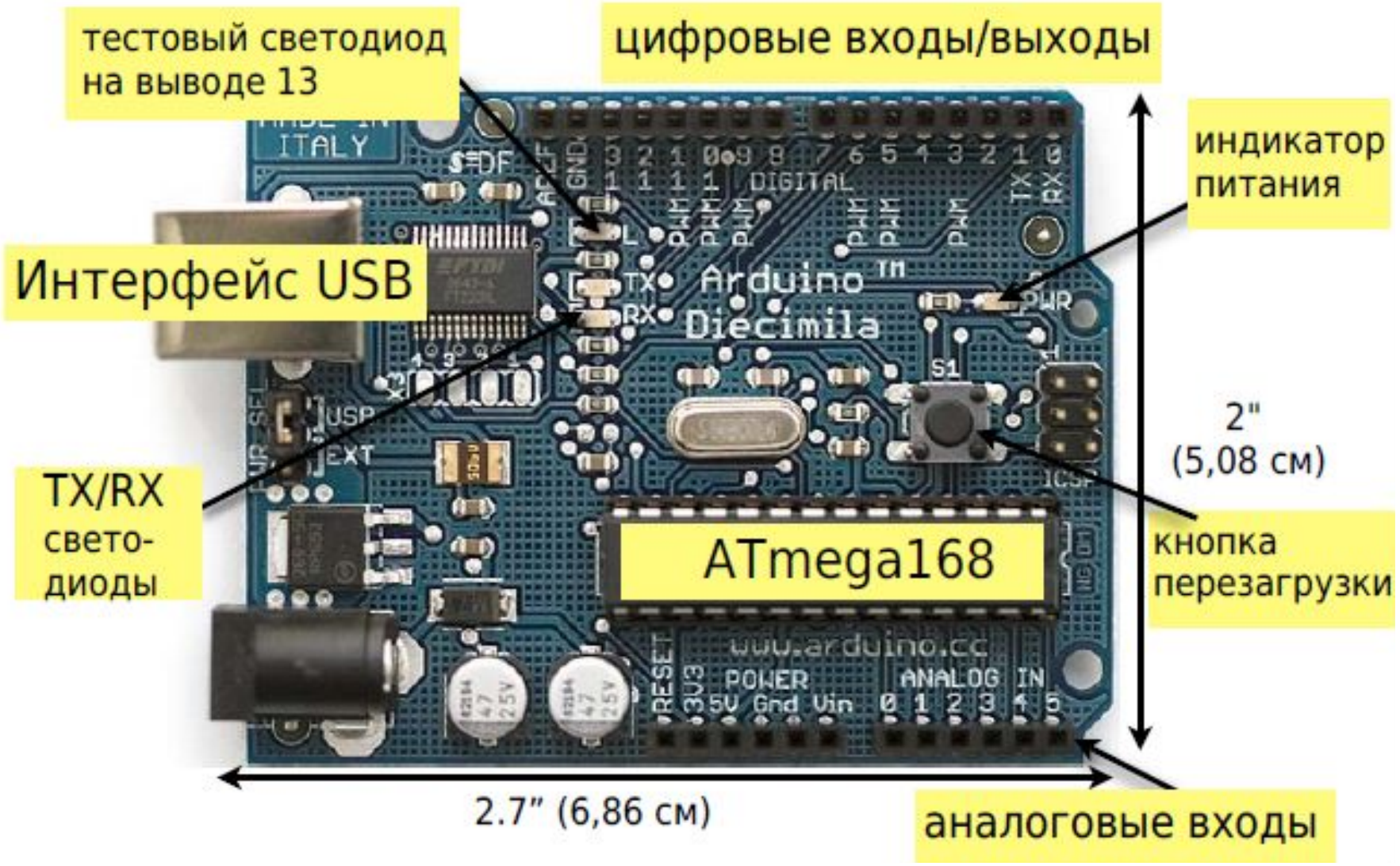
набор Boarduino

в «Stamp»-
формате



множество вариантов для разных нужд

Плата Arduino Diecimila



Язык программирования ARDUINO

- В Arduino-IDE (Integrated Development Environment) — встроенной среде разработки находятся различные инструменты и настройки, которые облегчают общение с программой Arduino.

Arduino alpha

An open project written, debugged and supported
by Massimo Banzi, David Cuartielles, Tom Igoe
Gianluca Martino and David Mellis

Based on processing by Casey Reas and Ben Fry

Структура программы arduino

- Комментарии и описание программы.
- Заголовки файлов и подключенные библиотеки.
- Объявление глобальных переменных.
- Стандартная настройка void setup () (порты и конфигурация).
- Основной цикл void loop ().
- Собственные процедуры

Типы данных
(ознакомится)

- Логический (булевый) тип данных — **boolean.** Может принимать одно из двух значений true или false. boolean занимает в памяти один байт
- **Char** (символ)Переменная типа **char** занимает 1 байт памяти и может хранить один алфавитно-цифровой символ (литеру). При объявление литеры используются одиночные кавычки: 'A' (двойные кавычки используется при объявлении строки символов - тип string: "ABC").
- **Byte -** тип данных byte 8-ми битное беззнаковое целое число, в диапазоне 0..255.
- **Int** (целое)о дин их наиболее часто используемых типов данных для хранения чисел. int занимает 2 байта памяти, и может хранить числа от -32 768 до 32 767
- **unsigned int** - (беззнаковое целое) число, также как и тип **int** (знаковое) занимает в памяти 2 байта. Но в отличие от **int**, тип **unsigned int** может хранить только положительные целые числа в диапазоне от 0 до 65535 (2^16)-1)

- **long** (длинное) используется для хранения целых чисел в расширенном диапазоне от -2,147,483,648 до 2,147,483,647.**long** занимает 4 байта в памяти
- **Unsigned long** (без знака длинное) используется для хранения положительных целых чисел в диапазоне от 0 до 4,294,967,295 (2^32 - 1) изанимает 32 бита (4 байта) в памяти**.**
- **float** (плавающий) служит для хранения чисел с плавающей запятой. Этот тип часто используется для операций с данными, считываемыми с аналоговых входов. Диапазон значений — от -3.4028235E+38 до 3.4028235E+38. Переменная типа **float**занимает 32 бита (4 байта) в памяти
- **Double** (двойной), в отличие от большинства языков программирования, имеет ту же точность, что и тип **float** и занимает также 4 байта памяти

- Базовая структура программы для Arduino довольно проста и состоит, по меньшей мере, из двух частей. В этих двух обязательных частях, или функциях, заключён выполняемый код

```
void setup()
{
  statements;
}

void loop()
{
  statements;
}
```

Где setup() — это подготовка, а loop() — выполнение. Обе функции требуются для работы программы.

Перед функцией setup - в самом начале программы, обычно, идёт, объявление

всех переменных. setup - это первая функция, выполняемая программой, и

выполняемая только один раз, поэтому она используется для установки режима

работы портов (pinMode()) или инициализации последовательного соединения

```
void setup()
{
  pinMode(pin, OUTPUT);      //  устанавливает 'pin' как выход
}
```

Следующая функция loop содержит код, который выполняется постоянно —

читаются входы, переключаются выходы и т.д. Эта функция — ядро всех программ Arduino и выполняет основную работу.

```
void loop()
{
  digitalWrite(pin, HIGH);    // включает 'pin'
  delay(1000);                // секундная пауза
  digitalWrite(pin, LOW);     // выключает 'pin'
  delay(1000);                // секундная пауза
}
```

- Функция — это блок кода, имеющего имя, которое указывает на исполняемый код, который выполняется при вызове функции. Функции void setup() и void loop() уже обсуждались

Могут быть написаны различные пользовательские функции, для выполнения повторяющихся задач и уменьшения беспорядка в программе. При создании функции, первым делом, указывается тип функции. Это тип значения, возвращаемого функцией, такой как 'int' для целого (integer) типа функции. Если функция не возвращает значения, её тип должен быть void. За типом функции следует её имя, а в скобках параметры, передаваемые в функцию.

```
int delayVal()
{
  int v;                    // создаём временную переменную 'v'
  v  = analogRead(pot);     // считываем значение с потенциометра
  v /= 4;                   // конвертируем 0 - 1023 в 0 - 255
  return v;                 // возвращаем конечное значение
}
```

{} фигурные скобки

Фигурные скобки (также упоминаются как просто «скобки») определяют начало и

конец блока функции или блока выражений, таких как функция void loop() или

выражений (statements) типа for и if.

За открывающейся фигурной скобкой { всегда должна следовать закрывающаяся фигурная скобка }. Об этом часто упоминают, как о том, что скобки должны быть «сбалансированы». Несбалансированные скобки могут приводить к критическим, неясным ошибкам компиляции, вдобавок иногда и трудно выявляемым в больших программах.

переменные

- Переменные — это способ именовать и хранить числовые значения для последующего использования программой. Само название - переменные, говорит о том, что переменные - это числа, которые могут последовательно меняться, в отличие от констант, чьё значение никогда не меняется. Переменные нужно декларировать (объявлять), и, что очень важно - им можно присваивать значения, которые нужно сохранить. Следующий код объявляет переменную inputVariable, а затем присваивает ей значение, полученное от 2-го аналогового порта:

```
int inputVariable = 0;          // объявляется переменная и
                                // ей присваивается значение 0
inputVariable = analogRead(2); // переменная получает значение
                                // аналогового вывода 2
```

- Переменные могут быть объявлены в начале программы перед void setup(), локально внутри функций, и иногда в блоке выражений таком, как цикл for. То, где объявлена переменная, определяет её границы (область видимости), или возможность некоторых частей программы её использовать.

- Глобальные переменные таковы, что их могут видеть и использовать любые функции и выражения программы. Такие переменные декларируются в начале программы перед функцией setup(). Локальные переменные определяются внутри функций или таких частей, как цикл for. Они видимы и могут использоваться только внутри функции, в которой объявлены.

```
int value;                          // 'value' видима
                                    // для любой функции
void setup()
{
  // no setup needed
}

void loop()
{
  for (int i=0; i<20;)              // 'i' видима только
  {                                 // внутри цикла for
    i++;
  }
  float f;                          // 'f' видима только
}                                   // внутри loop
```

массивы

```
int myArray[5];     // объявляет массив целых длиной в 6 позиций
myArray[3] = 10;    // присваивает по 4у индексу значение 10
```

Чтобы извлечь значение из массива, присвоим переменной значение по индексу массива:

```
x = myArray[3];     // x теперь равно 10
```

арифметика

```
y = y + 3;
x = x - 7;
i = j * 6;
r = r / 5;
```

операторы сравнения

```
x == y      // x равно y
x != y      // x не равно y
x <  y      // x меньше, чем y
x >  y      // x больше, чем y
x <= y      // x меньше, чем или равно y
x >= y      // x больше, чем или равно y
```

логические операторы

Logical AND:

```
if (x > 0 && x < 5)      // true, только если оба
                         // выражения true
```

Logical OR:

```
if (x > 0 || y > 0)      // true, если любое из
                         // выражений true
```

Logical NOT:

```
if (!x > 0)              // true, если только
                         // выражение false
```

константы

- true/false
- high/low Эти константы определяют уровень выводов как HIGH или LOW и используются при чтении или записи на логические выводы. HIGH определяется как логический уровень 1, ON или 5 вольт(3-5), тогда как LOW — 0, OFF или 0 вольт(0-2)