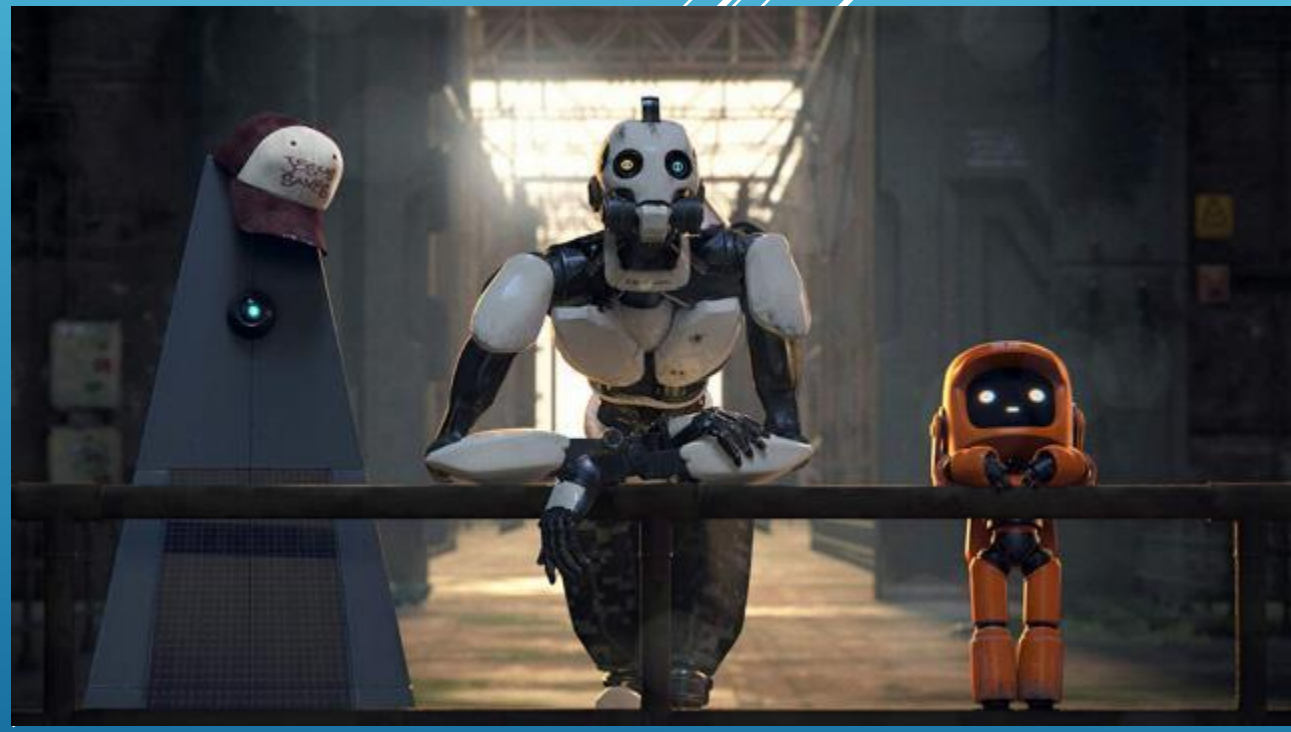
ОБРАЗОВАТЕЛЬНАЯ РОБОТОТЕХНИКА

ARDUINO™

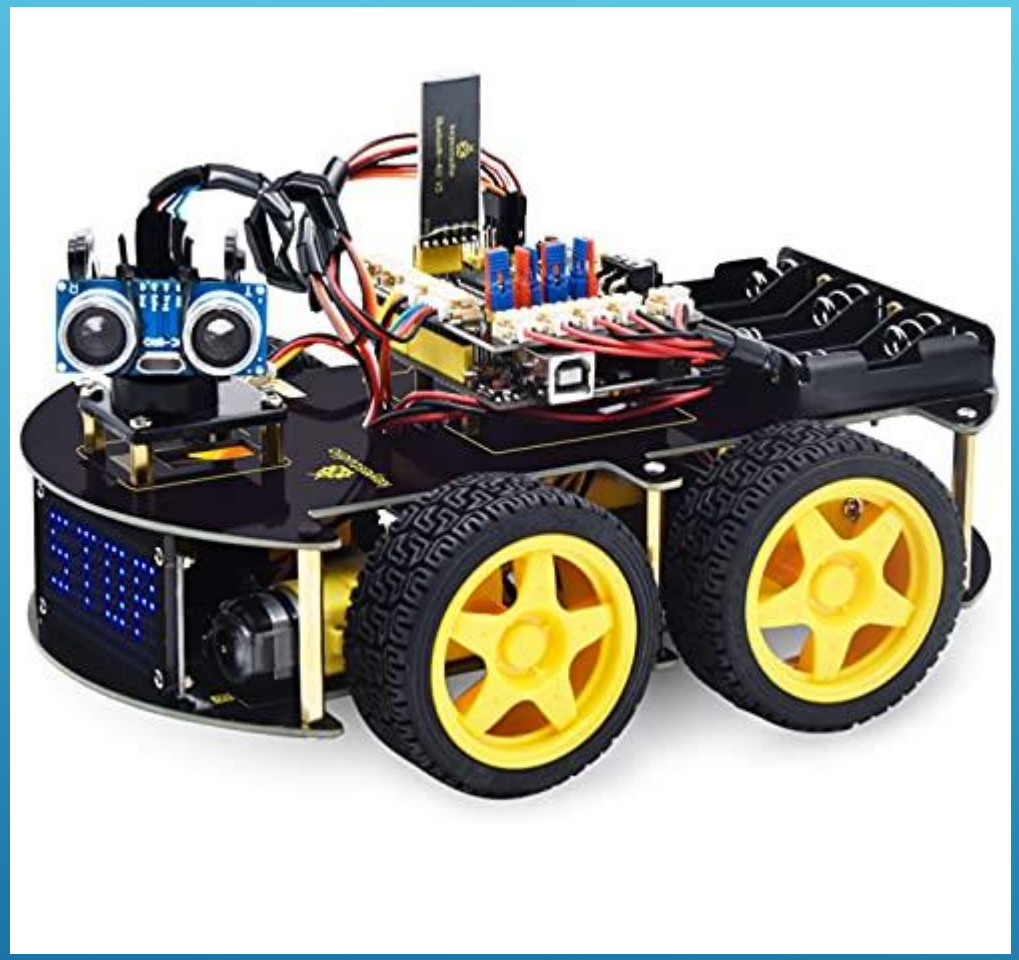
Лекция 2. Обзор платформы Arduino. Приёмы программирования.

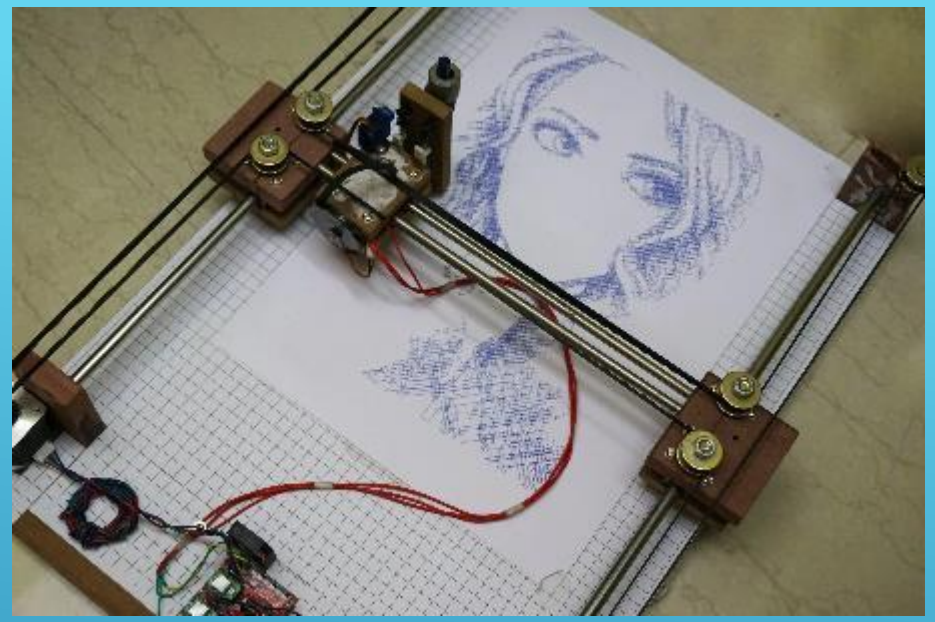


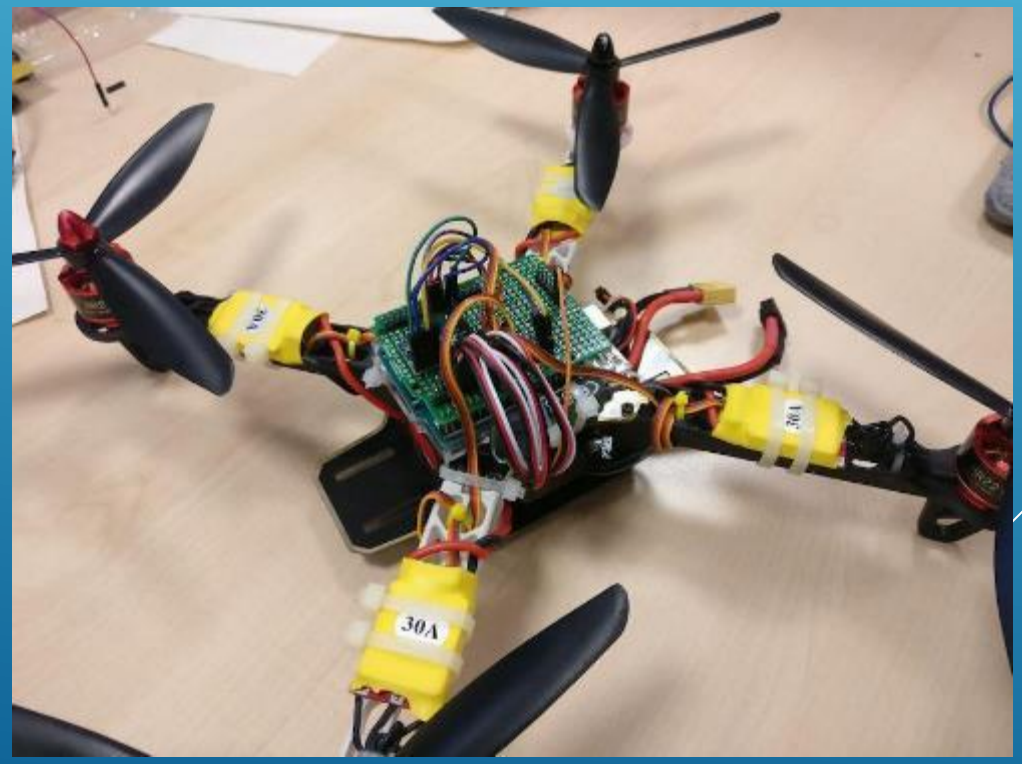
Пикулев В. Б.

http://sandbox.scilink.ru

УСТРОЙСТВА НА ARDUINO





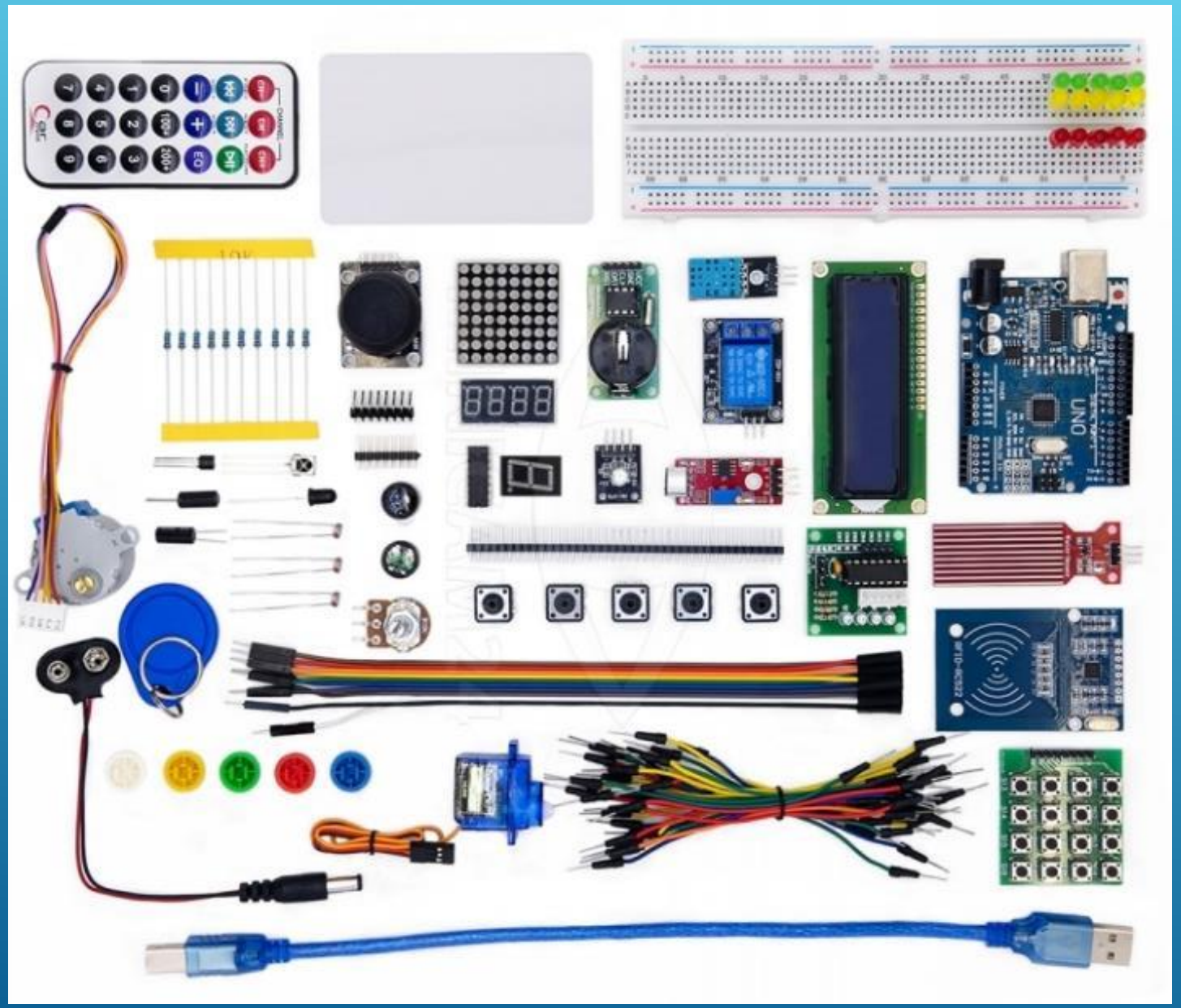


Arduino – это инструмент для создания различных устройств автоматики и робототехники, ориентированный на непрофессиональных пользователей.

ПЛАТФОРМА ARDUINO

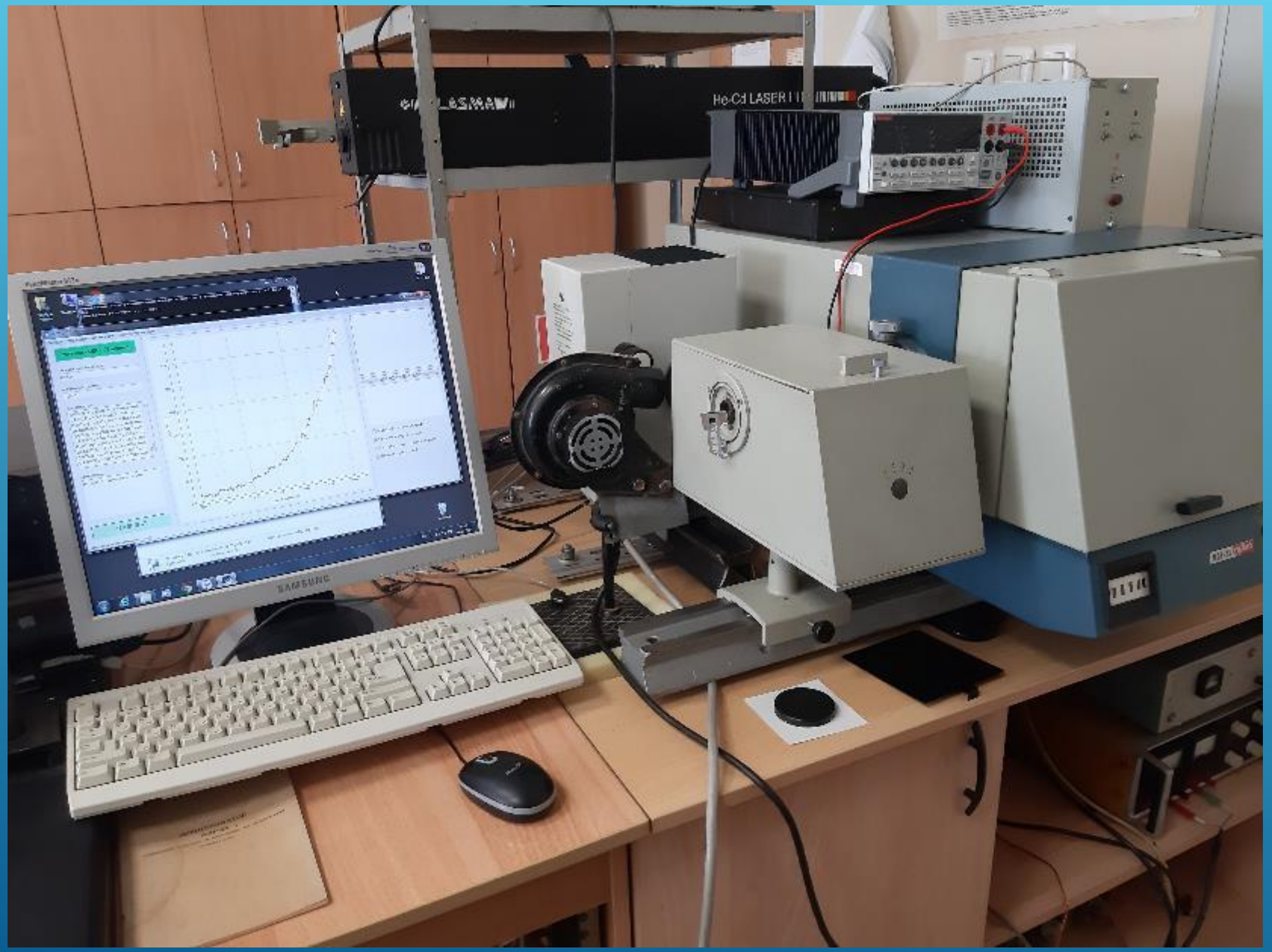
- Arduino — это открытая программно-аппаратная платформа для макетирования и разработки электронных автоматизированных устройств на базе микроконтроллеров Atmel. Используя датчики, приводы, динамики, платы расширения с набором стандартных интерфейсов, можно использовать Arduino в качестве управляющего устройства для различных систем управления.
- История Arduino началась с 2003 года, когда Массимо Банци, Дэвид Меллис и Давид Куартилье делают удачный вариант программно-аппаратной платформы для начального обучения программированию. Проект Arduino был с восторгом встречен схемотехниками во всем мире, на базе микросхем с простым языком программирования было разработано огромное количество вариантов различных электронных устройств. Имя для платформы было взято с вывески паба в итальянском городе Иврея.

ARDUINO KIT



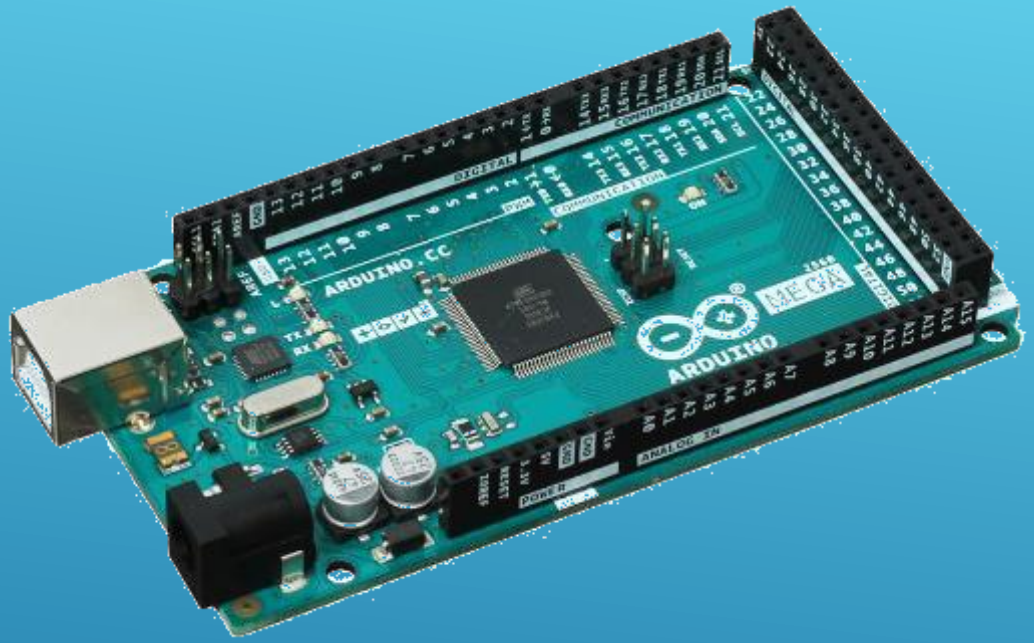
"Конструктор" Arduino, помимо платы контроллера, обычно содержит жидкокристаллический дисплей для вывода информации, набор различных датчиков, контроллер шаговых двигателей вместе с соответствующими двигателями, сервоприводы, светодиоды, динамик, часы, блок реле, платы коммуникации (GSM-модуль, IR-модуль), макетную плату, провода и прочее.

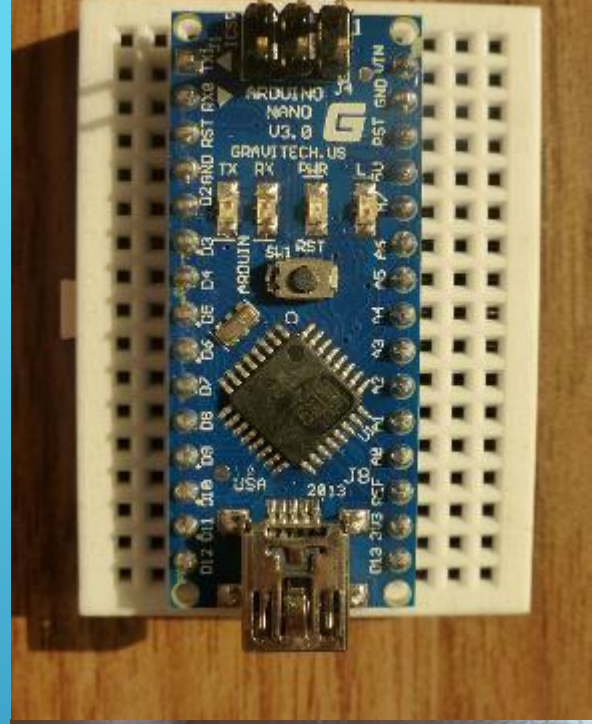
ARDUINO В НАУЧНЫХ ЛАБОРАТОРИЯХ



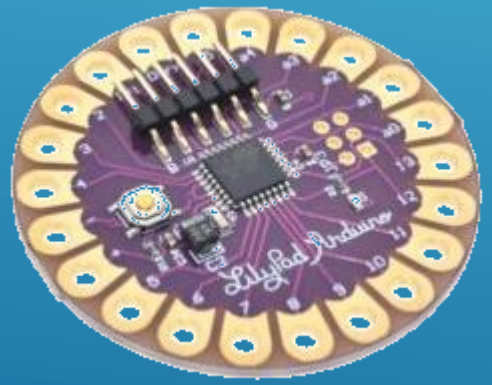
Автоматизированная установка для измерения спектров фотолюминесценции различных материалов, управляемая через контроллер Arduino и выводящая результаты на плату сбора данных компании National Instruments.

РАЗЛИЧНЫЕ ЛИНЕЙКИ ARDUINO

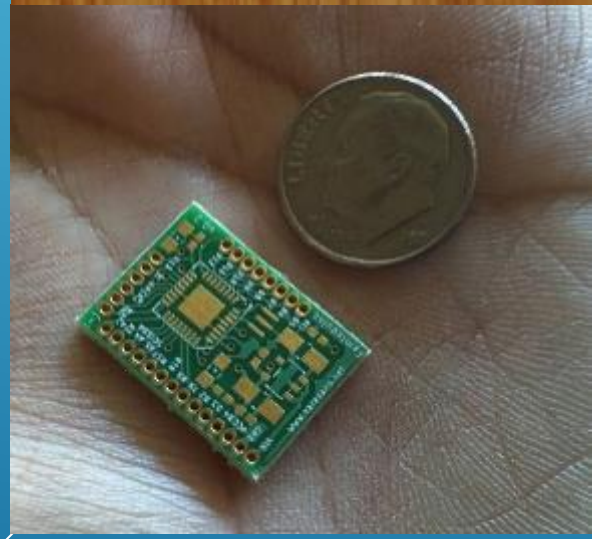




- Существует несколько версий платформ Arduino. Версия Leonardo базируется на микроконтроллере ATmega 32u4. Uno, Nano построены на 8-разрядном микроконтроллере ATmega 328. Arduino Mega построена на микроконтроллере ATMega 2560. Arduino Due разработана на базе 32-разрядного микропроцессора Cortex.



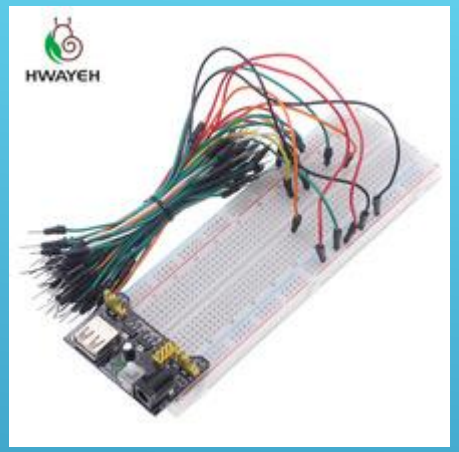
LilyPad

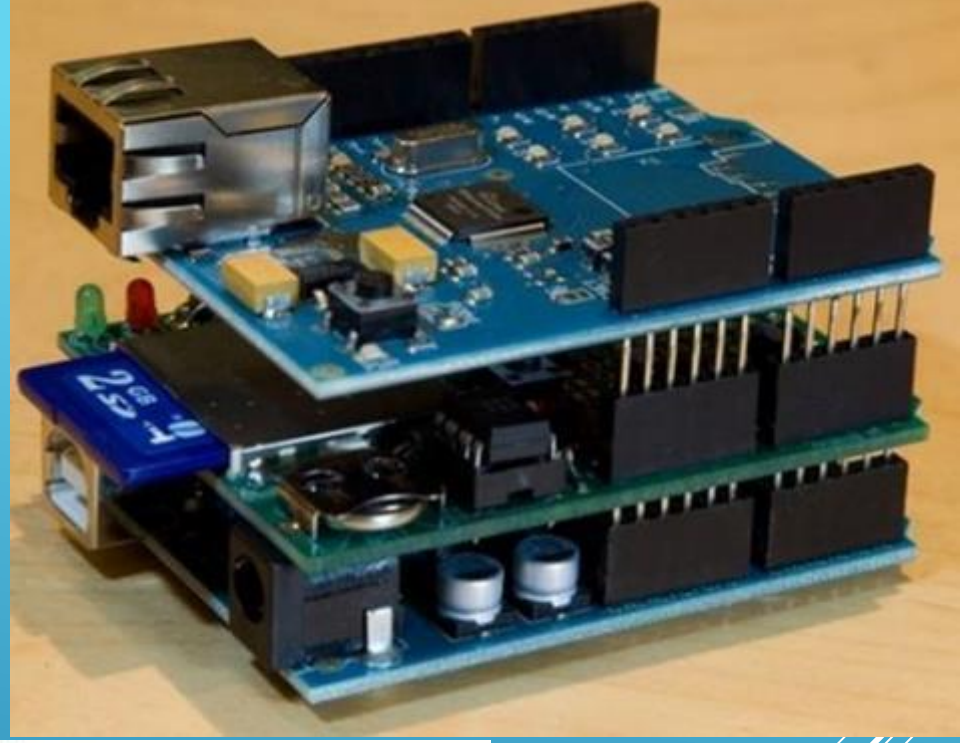


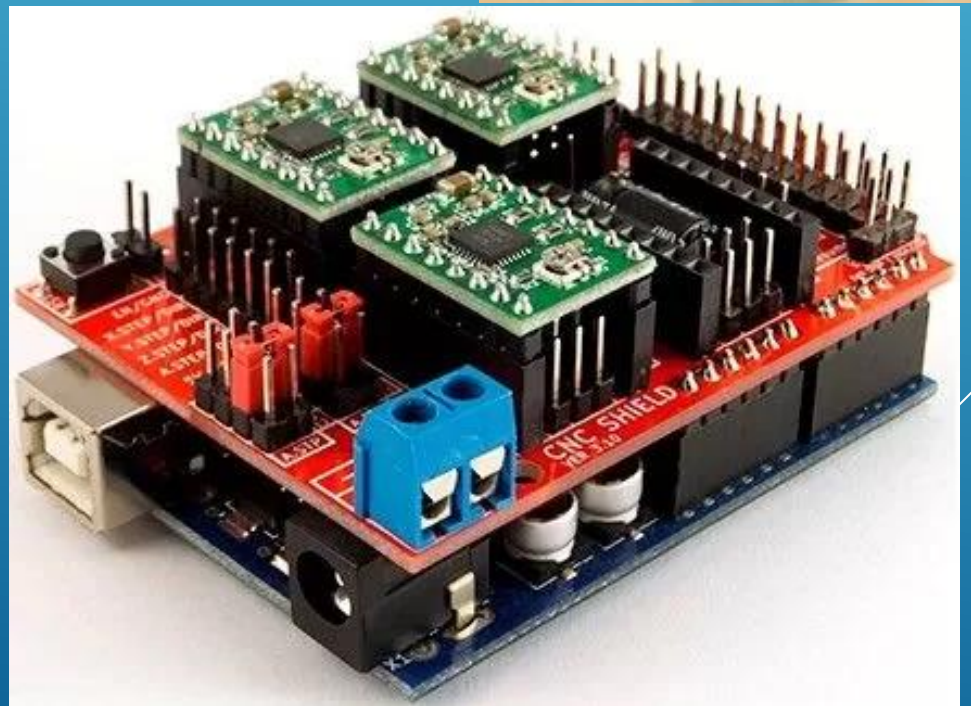
нано- и фемтодуино

ПРОЕКТИРОВАНИЕ УСТРОЙСТВ

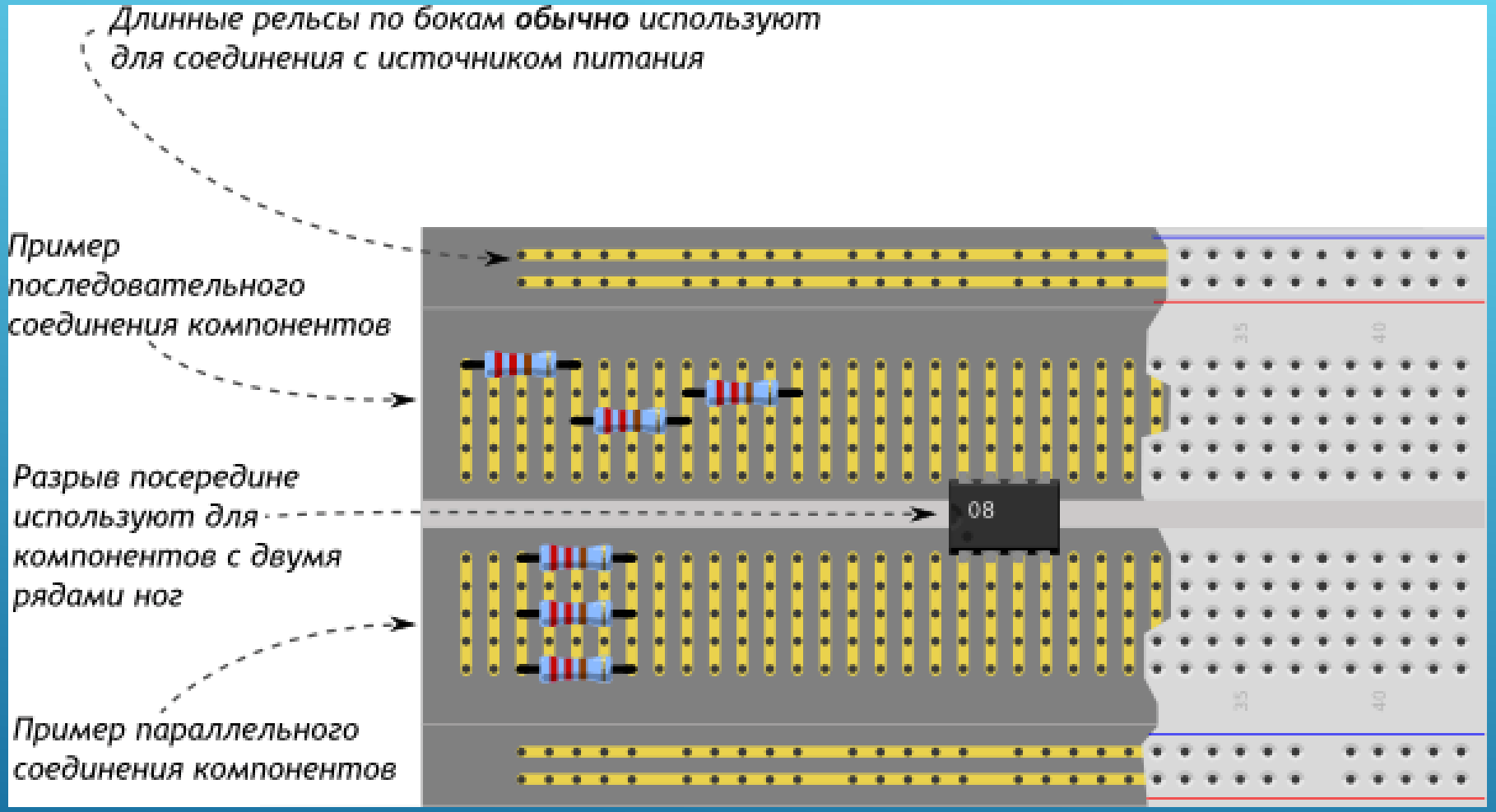
- Подключение к контактам (пинам) микроконтроллера выполняется через штыревые линейки, распаянные по обе стороны платы. Таким образом, разработчик может связать ATMega с внешними устройствами при помощи макетных проводов. Также под топологию Arduino создано огромное количество плат расширения (шилдов), обеспечивающих дополнительный функционал путём их каскадного включения. Такой подход позволяет значительно ускорить процесс создания прототипов тех или иных устройств, превращая рутинную работу в непринуждённую сборку электронного конструктора. Существуют шилды с набором датчиков, шилды-клавиатуры, шилды-экраны, шилды-расширители портов, радио-шилды и многое другое.







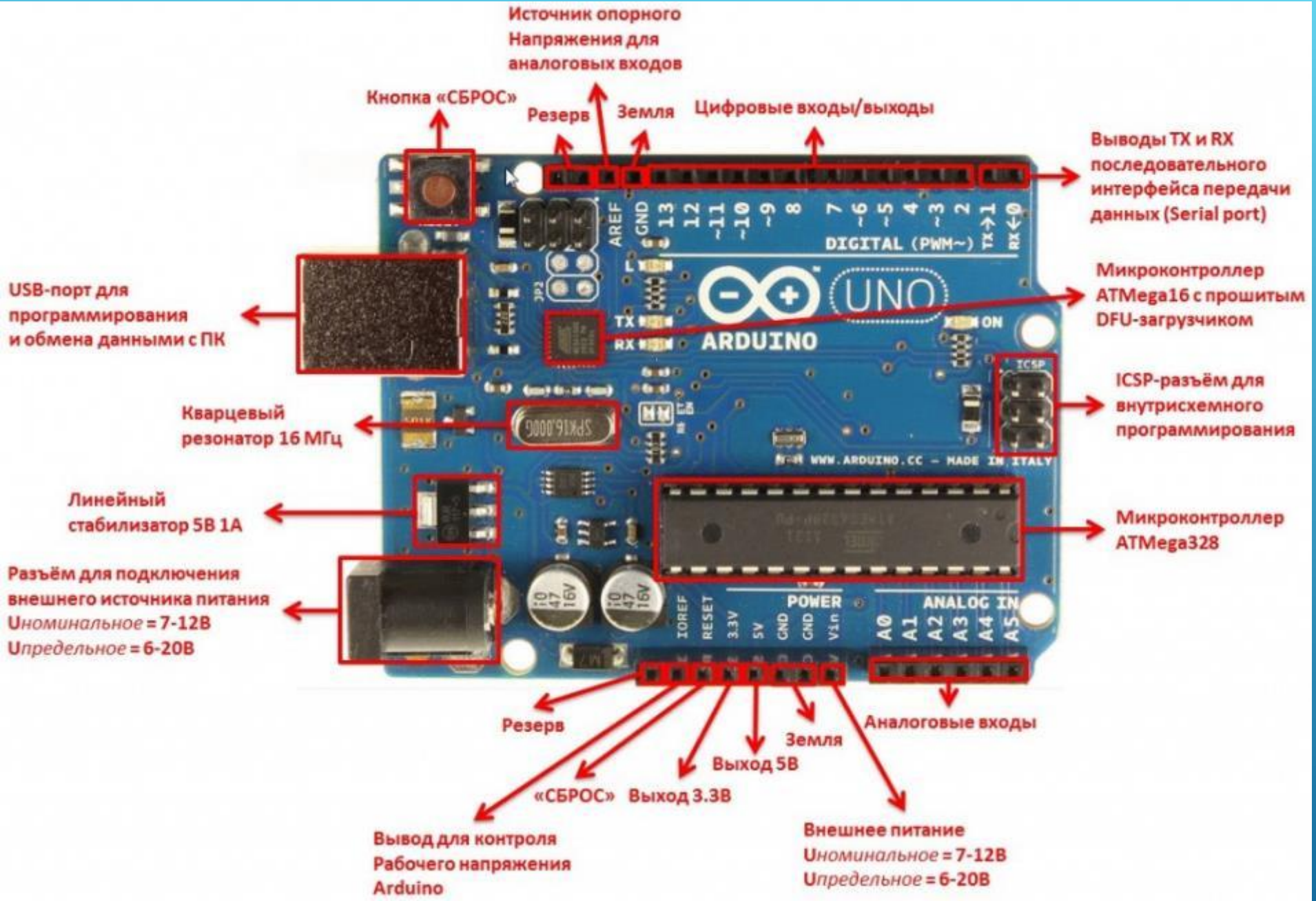
МАКЕТНАЯ ПЛАТА



Не обязательно ограничиваться одной платой. На многих монтажных платах предусмотрены специальные пазы и выступы по бокам, с их помощью можно соединить несколько плат.

Беспаечная макетная плата (breadboard) для Arduino используется при быстрой сборке схем без необходимости пайки радиоэлементов и проводов для соединения.

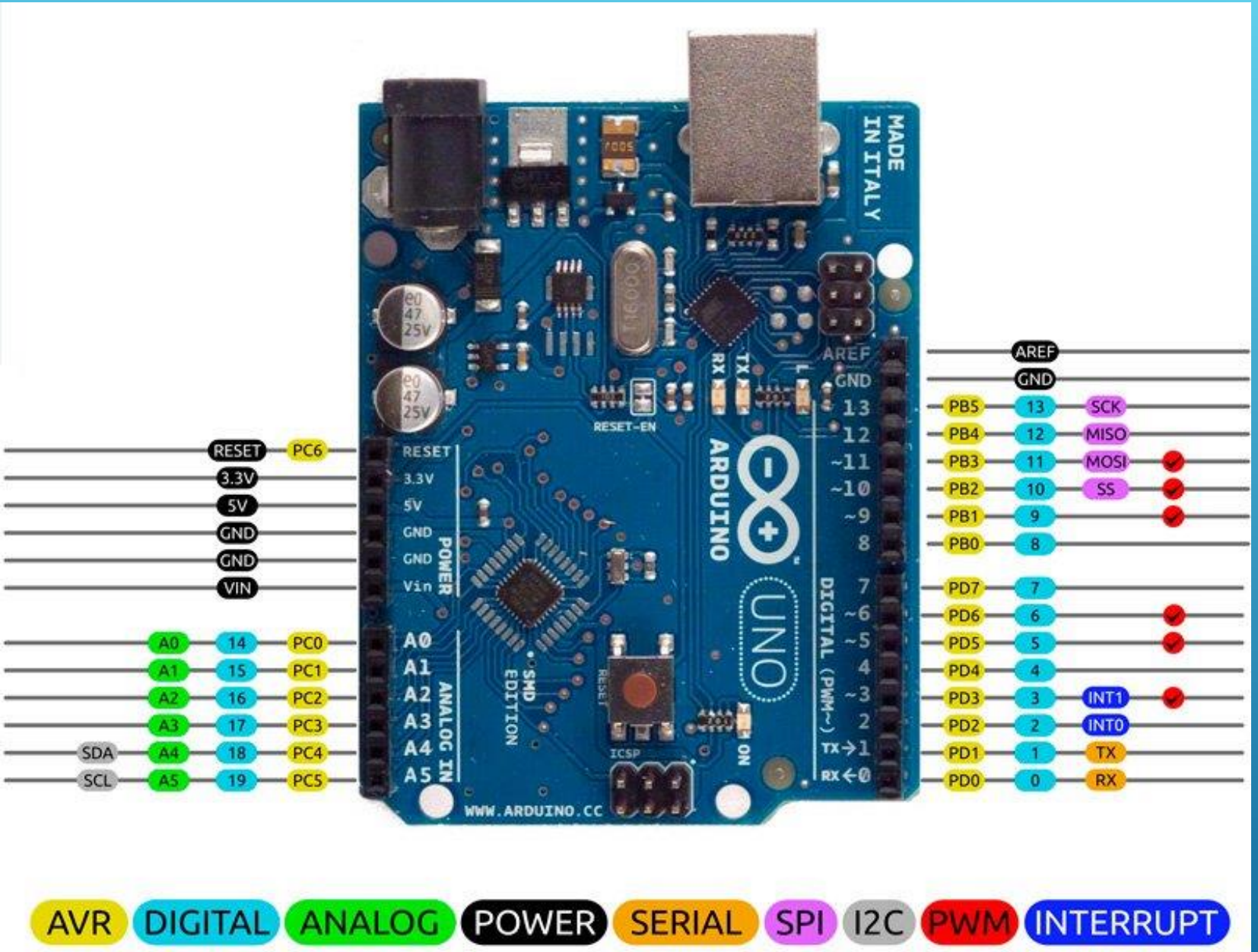
ПЛАТА ARDUINO UNO



Характеристики Arduino Uno:

- Контроллер – Atmega 328
- Рабочее напряжение - 5В
- Напряжение питания - 7-12В
- Максимальный ток одного вывода - 40 мА
- Тактовая частота - 16 МГц
- Длина - 68,6 мм
- Ширина - 53,4 мм

ПЛАТА ARDUINO UNO - РАСПИНОВКА



AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT

Группы контактов:

- Аналоговые входы (6 контактов)
- Последовательный интерфейс (2 контакта, протокол UART)
- Внешнее прерывание (2 контакта)
- Аналоговые выходы (6 контактов, ШИМ)
- Интерфейс SPI (4 контакта)
- Интерфейс I2C (2 контакта)
- Контакт 13 подключён к светодиоду на плате
- Контакт RESET дублирует кнопку сброса на плате
- Цифровые входы/выходы – 19 контактов

По умолчанию все внешние контакты Arduino сконфигурированы как входы. Если необходимо использовать контакт как выход, нужно его переконфигурировать, подав соответствующую команду микроконтроллеру.

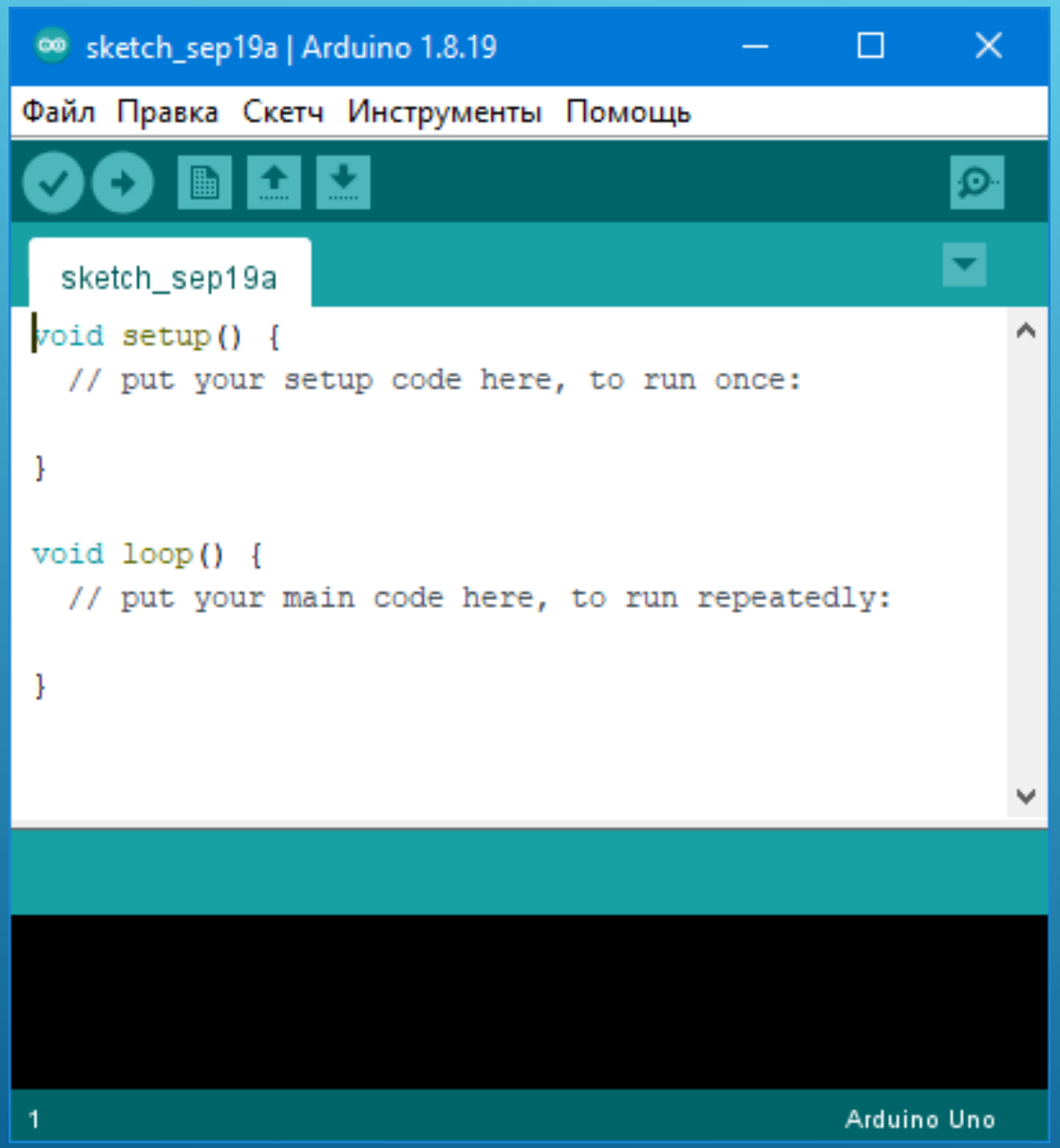
ПРОГРАММИРОВАНИЕ ARDUINO

- Для разработки программ для контроллера используется собственная, официально бесплатная среда разработки Arduino IDE, которая предназначена для написания, компиляции и загрузки пользовательских программ в память микроконтроллера, установленного на плате Arduino-совместимого устройства.
- Среда разработки устанавливается на персональный компьютер под Linux, MacOS либо Windows, и состоит из встроенного текстового редактора программного кода, области сообщений, окна консоли виртуального последовательного порта, панели инструментов с кнопками часто используемых команд. Для загрузки программ и организации обмена информацией среда разработки подключается к плате Arduino через USB-соединение.
- Основой среды разработки является язык Processing/Wiring – очень похожий на язык C++, дополненный простыми и понятными функциями для управления вводом/выводом на контактах.

Память в Arduino Uno:

- Flash – 32 кБ. Это основное хранилище для команд. 2кБ из данного пула памяти отводится на bootloader.
- Оперативная SRAM память - 2 кБ. Здесь хранятся переменные и объекты, создаваемые в ходе работы программы.
- Энергонезависимая память (EEPROM) объемом 1кБ. Процедура записи и считывания EEPROM требует использования дополнительной библиотеки.

ПРОГРАММИРОВАНИЕ ARDUINO

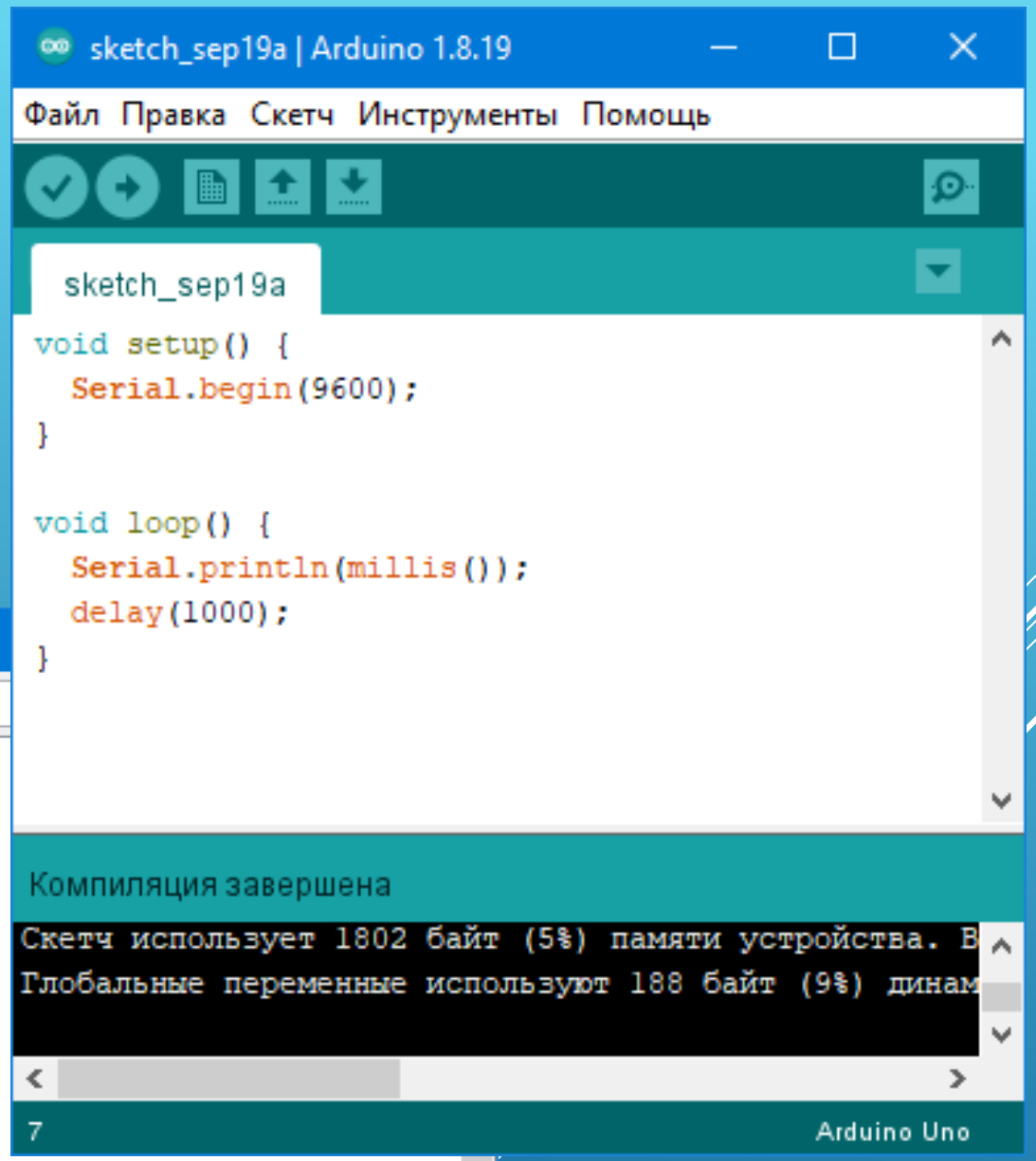


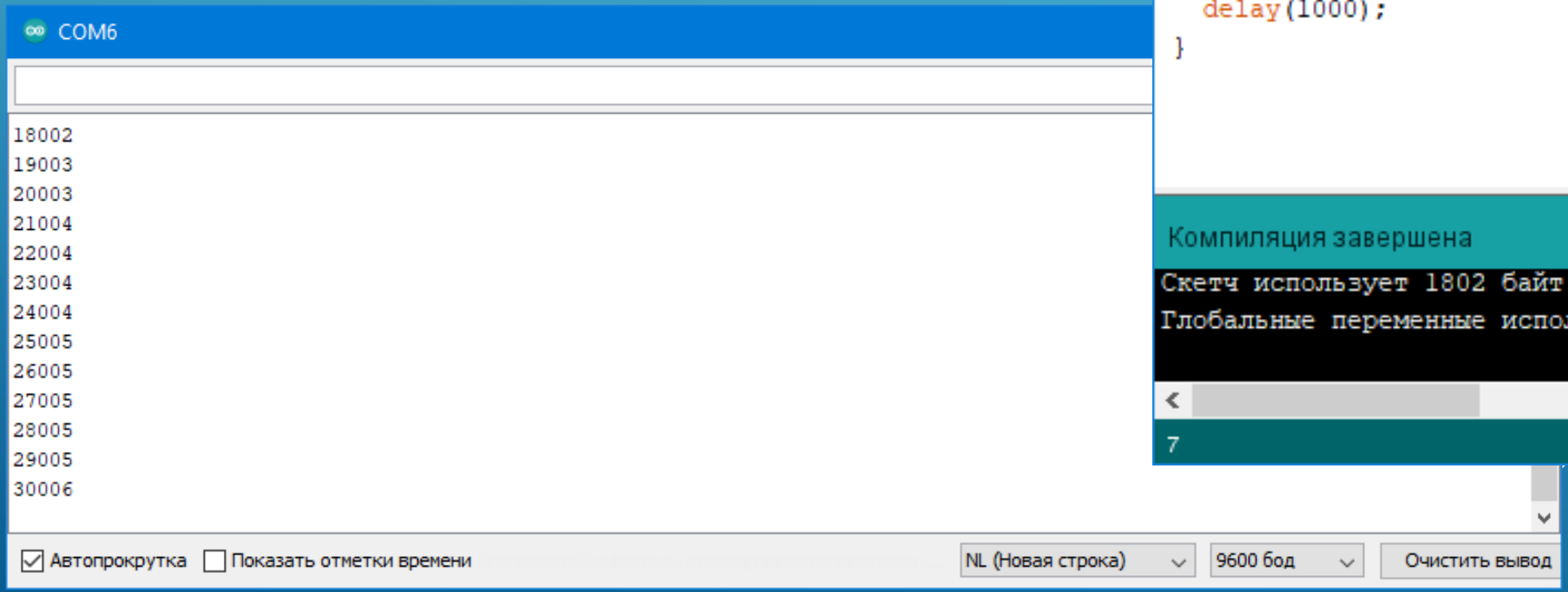
Разрабатываемые программы называются "скетчами" и при загрузке в Arduino автоматически преобразуются в инструкции микропроцессора. Широкий набор библиотек, которые могут быть легко найдены и загружены из Интернет, добавляет дополнительную функциональность скетчам, упрощая работу с датчиками и различными периферийными устройствами. Многие библиотеки снабжаются примерами, расположенными в папке examples каталога установки Arduino.

Базовая структура программы для Arduino состоит из двух обязательных частей: функций **setup()** и **loop()**. Перед функцией setup() идет объявление переменных, подключение библиотек. Функция setup() запускается один раз после каждого включения питания или сброса платы Arduino. Она используется для инициализации переменных, установки режима работы портов и прочих подготовительных действий. Функция loop() выполняет основной рабочий (бесконечный) цикл: последовательно исполняет команды, которые описаны в теле этой функции.

ПРИЁМ ДАННЫХ ОТ ARDUINO

Монитор последовательного порта (Serial Monitor) отображает данные, посылаемые в плату и получаемые от платы Arduino через порт USB (эмулятор COM-порта). Для отправки данных необходимо ввести в соответствующее поле текст и нажать кнопку "Send" или клавишу <Enter>. Скорость передачи последовательного порта настраивается командой Serial.begin. Повторное открытие окна порта автоматически делает рестарт программе.

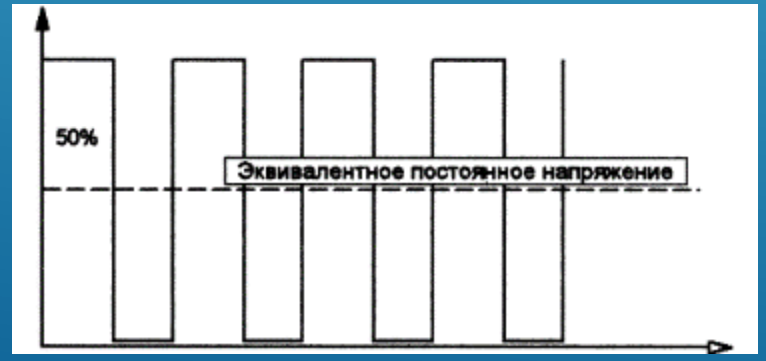


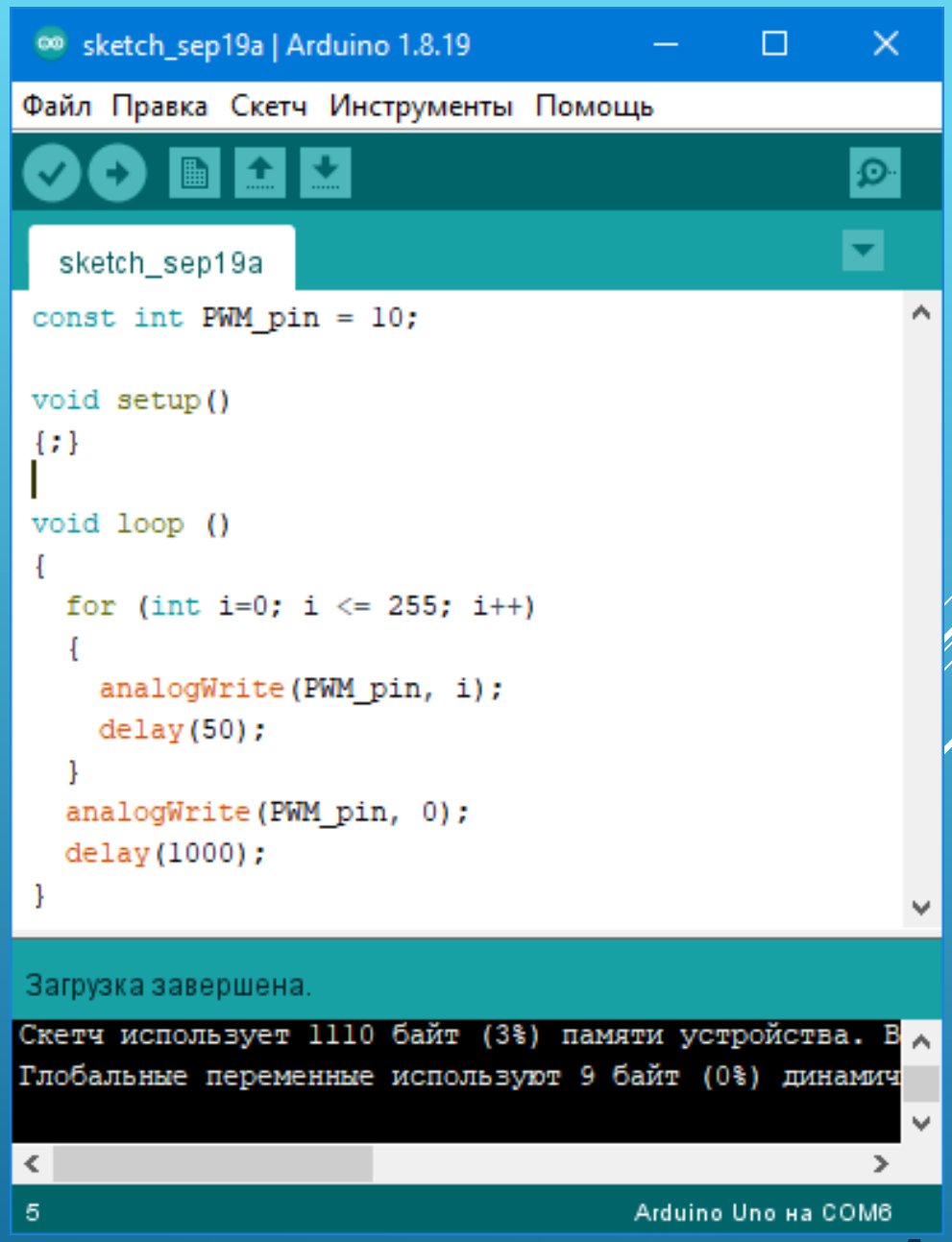


АНАЛОГОВЫЙ ВЫХОД

К 10 выводу платы подключен светодиод последовательно с резистором 330 Ом, второй вывод светодиода заземлён. Светодиод периодически медленно разгорается и быстро гаснет.

Вывод 10 на самом деле цифровой, на который функция analogWrite() подаёт ШИМ-сигнал. Суть широтно-импульсной модуляции: подавая на выход периодический сигнал, состоящий из высоких и низких уровней разной длительности, можно эмулировать среднее значение напряжения между максимальным значением (5 В) и минимальным (0 В).





ПЕРЕМЕННЫЕ, КОНСТАНТЫ И МАССИВЫ

- Переменная должна быть объявлена до её использования.
- Переменные могут быть объявлены в начале программы перед void setup() – это будут **глобальные** переменные, **локальные** – объявляются внутри функций или в блоке кода { }, таком как цикл for. Место, где объявлена переменная, определяет ее область видимости.
- Локальные переменные могут быть объявлены как **static** – такая переменная всегда сохраняет своё значение, оставаясь локальной.
- Константы могут вводиться с помощью ключевого слова **const** либо с помощью директивы **#define** (прямая замена имени числом).

```
int sensor = 10;
byte val1, val2, val3 = 10;

#define BANG_PIN 10

int myPins[] = {2, 4, 8, 3, 6};
int X = myPins[0];
myPins[4] = 11;

float sens[3] = {0.2, 0.4, -8.5};
byte myTable[][2] = {{10, 11}, {15, 16}};
myTable[1][0] = 77;

char message[6] = "Hello";
char c = message[1];
char newMessage[] = "Good bye!"

String info = "Run, rabbit...";

char* myABC[] = {"Alpha","Beta","Gamma"};
```

ТИПЫ ДАННЫХ, ПРЕОБРАЗОВАНИЯ ТИПОВ

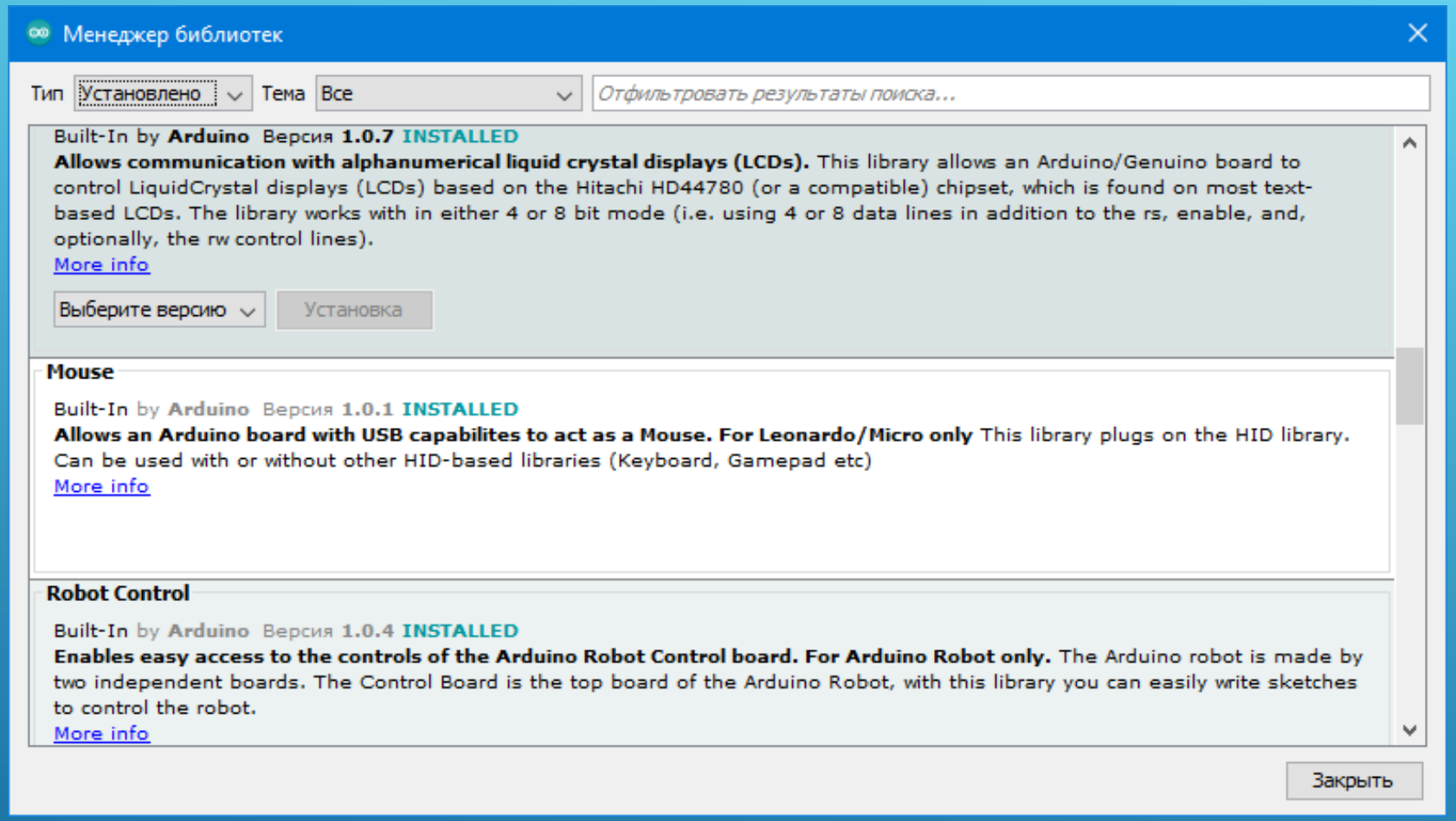
- **boolean** – true либо false
- **char** – 'A' либо от -128 до 127
- **byte** – от 0 до 255
- **int** – от -32768 до 32767
- **unsigned int** – от 0 до 65535
- **long** – от -2147483648 до 2147483647
- **unsigned long** – от 0 до 4 294 967 295
- **float** - от -3,4028235E+38 до 3,4028235E+38.
- **double** – только для совместимости, аналогичен float
- **String** – очень старые версии не поддерживают. String s = "строка";
- ***void*** – ключевое слово со смыслом "отсутствие значения"
- ***** - указатель, способ динамической адресации к переменным и массивам

```
char myval = 65; // символ 'A'
Serial.println(myval);

String myStr = "111";
int i = myStr.toInt();
float varF = myStr.toFloat();
// float "тяжёл" для Arduino

byte val = 100;
int res = (int)val;

String varH = String(res, HEX);
String varB = String(res, BIN);

char[] chArray = "пример";
String str(chArray);

long val = 35 * 1000L;
int X1 = i++;
int X2 = ++i;
X1 += X2;
```

ПОЛЕЗНЫЕ ОПЕРАЦИИ И ВСТРОЕННЫЕ ФУНКЦИИ

- **true** и **HIGH** есть **1**, **false** и **LOW** есть **0**, но пользоваться этим надо осторожно
- **%** - остаток от деления
- **PI** – константа *π*
- **min(x, y)**, **max(x, y)** - наименьшее и наибольшее из x и y
- **abs(x)** – модуль числа x, **round (x)** – округление x до ближайшего целого
- **constrain(x, a, b)** – ограничение вывода областью допустимых значений [a, b]
- **bit(x)** – возводит 2 в степень x
- **radians(deg)** – перевод градусов в радианы, обратный **degrees(rad)**
- **sin(x)**, **cos(x)**, **tan(x)** – триг. ф. от x, заданного в радианах (math.h)
- **pow(base, power)** – возведение числа base в степень power (math.h)
- **sqrt(x)** – квадратный корень числа x, **cbrt(x)** – кубический корень (math.h)
- **log(x)** – натуральный логарифм x (math.h)

УСТАНОВКА БИБЛИОТЕК




```
#include <Wire.h>
#include <OneWire.h>
#include <LiquidCrystal_I2C.h>
#include <DS3231.h>
#include "SparkFunHTU21D.h"

#define A6_serial Serial1 // моду
#define A6_speed 115200
#define PC_speed 115200

byte i_addr[8];
```

Полезные примеры по применению возможностей библиотеки обычно находятся в каталоге **\examples** внутри папки с библиотекой. Заголовочный файл библиотеки **имя.h** несёт основную информацию о её возможностях.

Библиотеку можно разместить в каталоге **\Arduino\libraries** либо в папке со скетчем. Последнее полезно, если в код библиотеки вносятся изменения, необходимые только для этого проекта. В этом случае рядом с файлом скетча **.ino** должны лежать файлы библиотеки, а в скетче подключать библиотеку необходимо через двойные кавычки.

СИНТАКСИС ПРОГРАММ: ЦИКЛЫ

- Оператор **for**:
  - служит для повторения блока операторов, заключенных в фигурные скобки. Цикл завершается, когда проверка условия даёт false.
  - **for (initialization; condition; increment) { тело цикла; }**
- Оператор **while**:
  - Оператор while будет вычислять в цикле непрерывно и бесконечно до тех пор, пока выражение в круглых скобках не станет равно false.
  - **while (выражение) { тело цикла; }**
- Оператор **do...while:**
  - Цикл do работает так же, как и цикл while, за исключением того, что условие проверяется не в начале, а в конце цикла. То есть цикл do выполнится хотя бы один раз.
  - **do { тело цикла } while (выражение);**
- Оператор **break:**
  - Оператор break используется для принудительного выхода из циклов do, for или while, не дожидаясь завершения цикла по условию.
  - внутри цикла **if (условие) { операторы; break; }**
- Оператор **continue:**
  - Оператор continue пропускает оставшиеся операторы в текущем шаге цикла. Вместо них выполняется проверка условного выражения цикла.
  - внутри цикла **if (условие) { операторы; continue; }**

СИНТАКСИС ПРОГРАММ: ПРИМЕРЫ ЦИКЛОВ И SWITCH

```
for (int x = 0; x < 255; x++) {
  digitalWrite(PWMpin, x);
  sens = analogRead(sensorPin);
  if (sens > threshold) {
    x = 0;
    break;
  }
delay(50);
}
```

```
while(true){
    Serial.println("Waiting…");
}
```

```
switch (var)
 {
 case label1:
 // код для выполнения
 break;
 case label2:
 // код для выполнения
 break;
 case label3:
 // код для выполнения
 break;
 default:
 // код для выполнения
 break;
 }
```

```
switch (inByte) {
case 'a':
  digitalWrite(2, HIGH);
  break;
case 'b':
  digitalWrite(3, HIGH);
  break;
case 'c':
  digitalWrite(4, HIGH);
  break;
case 'd':
  digitalWrite(5, HIGH);
  break;
case 'e':
  digitalWrite(6, HIGH);
  break;
default:
  for (int thisPin = 2;
thisPin < 7; thisPin++) {
digitalWrite(thisPin, LOW);
}
```

```
do {
delay(50);    // подождать, пока датчики стабилизируются
x = readSensors(); // проверить датчики
} while (x < 100);
```

ФУНКЦИИ

```
void setup() {
  Serial.begin(9600);
  int c = sumFunction(10, 20);
  int d = sumFunction(11, 22, 33);
  Serial.println(c);
  Serial.println(d);
}

void loop() {
}

int sumFunction(int paramA, int paramB) {
  return (paramA + paramB);
}

int sumFunction(int paramA, int paramB, int paramC) {
  return (paramA + paramB + paramC);
}
```

- Функция – логически выделенный именованный блок кода, предназначенный обычно для многократного вызова из основной программы. Программа может строиться из нескольких функций, каждая из которых выполняет свою задачу.
- Функции могут вообще не возвращать результат, могут возвращать любые типы значений, включая структуры, либо изменять переменные по ссылке.

```
#define sum(x, y) ((x)+(y)) // пример описания встраиваемой функции-макроса
```

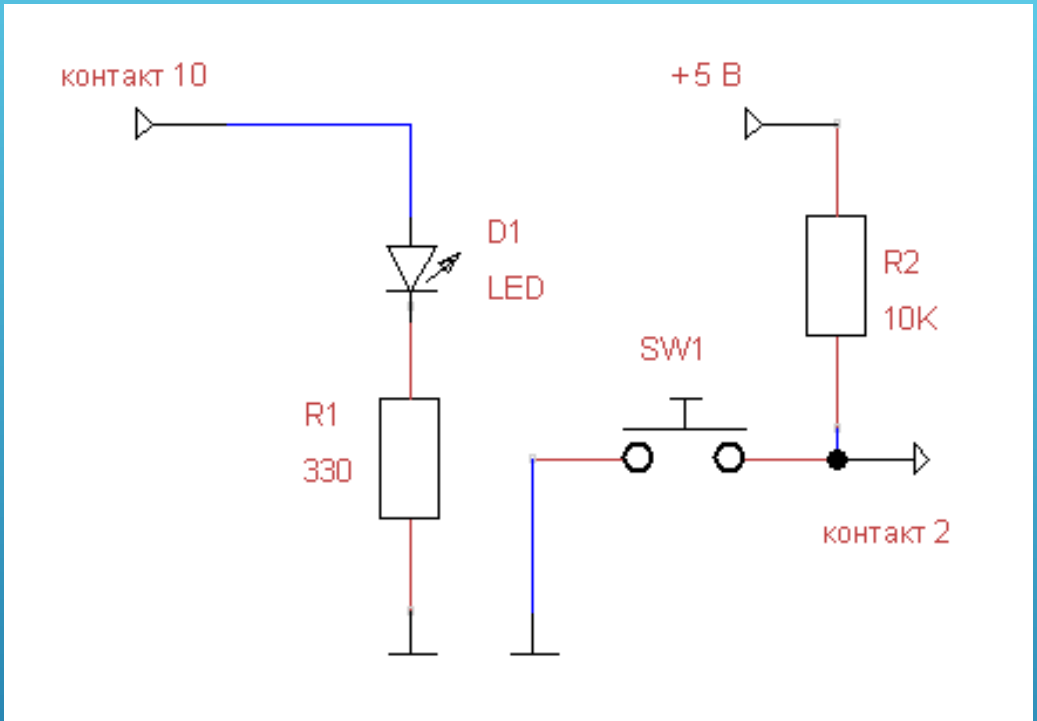
УПРАВЛЕНИЕ ЗАДЕРЖКАМИ

- **delay(time)** - задержка на указанное количество миллисекунд (от 1 до 4 294 967 295 мс ~ 50 суток). Работает с использованием системного таймера. Неудобна тем, что в линейно организованных программах блокирует выполнение команд и не работает в прерываниях.
- **delayMicroseconds(time)** – задержка на указанное количество микросекунд (от 4 до 16383 мкс). Работает методом пропуска тактов процессора.
- **millis()** - возвращают время в миллисекундах, прошедшее с момента запуска программы
- **micros()** - возвращают время в микросекундах, прошедшее с момента запуска программы

Эти функции позволяют организовать программу практически любой сложности с любым количеством параллельно выполняющихся по таймеру задач.

```
#define LED_RED     10
#define LED_GREEN   9

void setup() {
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
}

void loop() {
  digitalWrite(LED_RED, HIGH);
  delay(1000);
  digitalWrite(LED_RED, LOW);
  delay(1000);
}

void yield() {
  digitalWrite(LED_GREEN, HIGH);
  delay(220);
  digitalWrite(LED_GREEN, LOW);
  delay(220);
}
```

ЦИФРОВОЙ ВХОД И ВЫХОД



На выходных контактах обычно ставят токоограничивающий резистор (ток через светодиод не более 20 мА, падение напряжения на нём – 2 В), на входных – "подтягивающий" резистор. При нажатии на кнопку вход 2 будет заземлён, в соответствии с логикой программы светодиод при этом погаснет. Если кнопку отпустить, светодиод загорится. Программа простая и неэффективная.

sketch_sep20b | Arduino 1.8.19

Файл Правка Скетч Инструменты Помощь

sketch_sep20b §

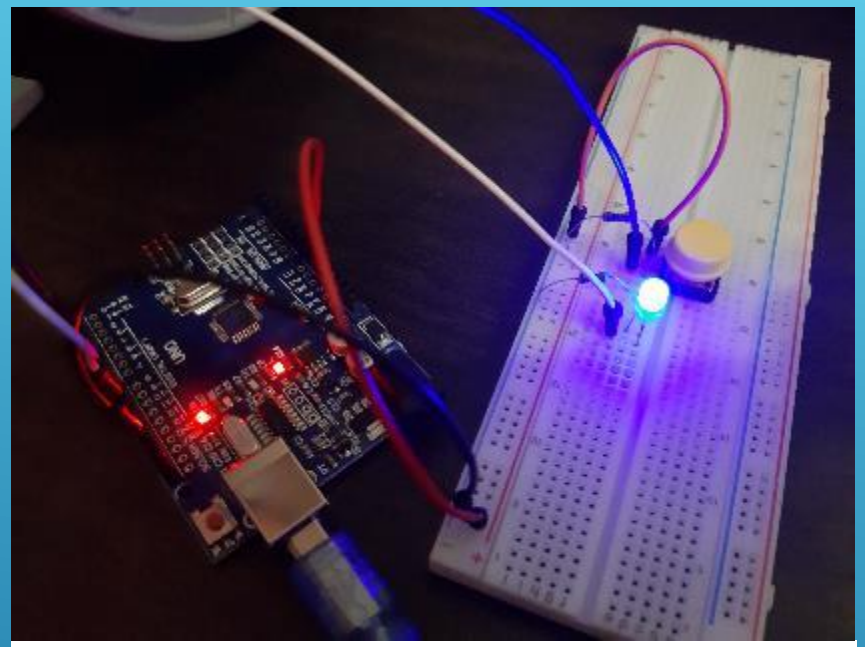
```
const int LED=10; // Контакт для подключения светодиода
const int BUTTON=2; // Контакт для подключения кнопки
void setup()
{
  pinMode (LED, OUTPUT); // выходной контакт
  pinMode (BUTTON, INPUT); // входной контакт
}
void loop()
{
  if (digitalRead(BUTTON) == LOW)
  {
    digitalWrite(LED, LOW);
  }
  else
  {
    digitalWrite(LED, HIGH);
  }
}
```

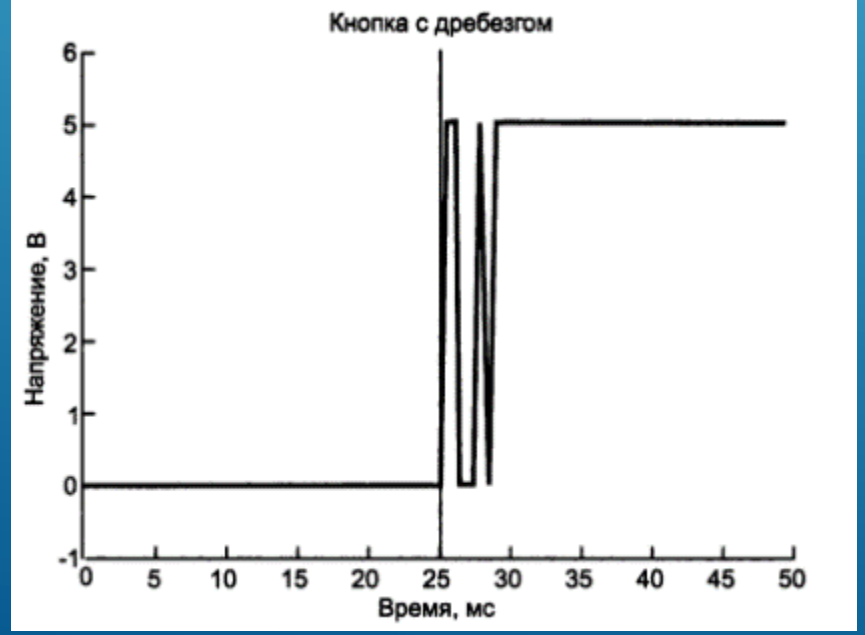
Загрузка завершена.

```
Скетч использует 892 байт (2%) памяти устройства. Всего доступно 3225
Глобальные переменные используют 9 байт (0%) динамической памяти, оста
```

19 Arduino Uno на COM6

LED-ТРИГГЕР





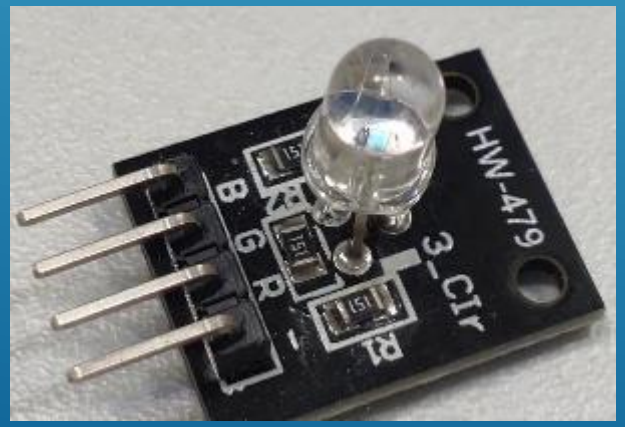
sketch_sep22a

```
const int LED=10;
const int BUTTON=2;
boolean lastButton = LOW; // предыдущее состояние кнопки
boolean currentButton = LOW; // текущее состояние кнопки
boolean ledOn = false; // текущее состояние светодиода

void setup() {
  pinMode (LED, OUTPUT);
  pinMode (BUTTON, INPUT);
}

boolean myButton(boolean last) {
  boolean current = digitalRead(BUTTON);
  if (last != current) {
    delay(5);
    current = digitalRead(BUTTON);
  }
  return current;
}

void loop() {
  currentButton = myButton(lastButton);
  if (lastButton == HIGH && currentButton == LOW) {
    ledOn = !ledOn;
  }
  lastButton = currentButton;
  digitalWrite(LED, ledOn);
}
```

Основной цикл программы переключает состояние светодиода при каждом нажатии кнопки. Предыдущее состояние кнопки запоминается в переменной lastButton. Функция myButton корректно читает состояние сработавшей кнопки (устраняет дребезг контактов).

RGB-LED И ГЕНЕРАТОР СЛУЧАЙНЫХ ЧИСЕЛ

- Каждое нажатие на кнопку меняет цвет светодиода
- Функция **random** возвращает псевдослучайное число между min и max-1: random(max), random(min, max)
- Можно использовать функцию **RandomSeed()**, которая инициализирует генератор псевдослучайных чисел. Генерируемая последовательность случайных чисел очень длинная, и всегда одна и та же. Точка в этой последовательности, с которой начинается генерация чисел, зависит от параметра seed.




```
const int R_LED=9;
const int G_LED=10;
const int B_LED=11;
const int BUTTON=2;
boolean lastButton = LOW; // предыдущее состояние кнопки
boolean currentButton = LOW; // текущее состояние кнопки

void setup() {
  pinMode (R_LED, OUTPUT);
  pinMode (G_LED, OUTPUT);
  pinMode (B_LED, OUTPUT);
  pinMode (BUTTON, INPUT);
}

void loop() {
  currentButton = myButton(lastButton);
  if (lastButton == HIGH && currentButton == LOW) {
    analogWrite(R_LED, random(255));
    analogWrite(G_LED, random(127));
    analogWrite(B_LED, random(127));
  }
  lastButton = currentButton;
}
```

УКАЗАТЕЛИ

- Указатель - это переменная, которая содержит адрес начала области данных (переменной, структуры, объекта, функции и т.п.) в памяти микроконтроллера. Зная адрес начала блока данных и тип соответствующих данных, можно получить контроль над данными, расположенными по этому адресу. Функция тоже имеет свой адрес в памяти, по которому к ней можно обратиться. По указателю функцию можно просто вызвать, а можно передать её адрес в качестве аргумента в другую функцию.
- **&** – возвращает адрес в памяти
- ***** – обращение (чтение, запись) по указанному адресу
- **->** – оператор косвенного обращения к членам и методам класса или структуры.

```
byte b;      // просто переменная типа byte
b = 10;      // b теперь 10
byte* ptr;   // ptr – переменная "указатель на объект типа byte"
ptr = &b;    // указатель ptr хранит адрес переменной b
*ptr = 22;   // b теперь равна 22 (записываем по адресу &b)
byte s;      // переменная s
s = *ptr;    // s теперь тоже равна 22 (читаем по адресу &b)
void (*ptrF)(byte a); // указатель на функцию
ptrF = goWork;        // присваиваем указателю адрес функции goWork
```

НЕКОТОРЫЕ ПРИМЕРЫ ПРОГРАММИРОВАНИЯ

```
// изысканный таймер
boolean LEDflag = false;
uint32_t myTimer;

void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  if (millis() - myTimer >= 1000) {
    myTimer = millis();
    digitalWrite(13, LEDflag);
    LEDflag = !LEDflag;
  }
}
```

```
// переменное количество аргументов
void func(int num, ...) {
  va_list valist;
  va_start(valist, num);
  for (int i = 0; i < num; i++) {
    Serial.print(va_arg(valist, int));
    Serial.print(',');
  }
  va_end(valist);
  Serial.println();
}
```

```
// ссылка
void setup() {
  Serial.begin(9600);
  int value = 7
  square(value);
  Serial.println(value);
}
void square(int &val) {
  val = val * val;
}
```

КЛАССЫ

led.h

```
class LED {
  public:
    LED(byte pin, uint16_t prd) {
      _pin = pin;
      _prd = prd;
      pinMode(_pin, OUTPUT);
    }

  void blink() {
      if (millis() - _tmr >= _prd) {
        _tmr = millis();
        _flag = !_flag;
        digitalWrite(_pin, _flag);
      }
    }

  private:
    byte _pin;
    uint32_t _tmr;
    uint16_t _prd;
    bool _flag;
};
```

скетч

```
#include "LED.h"

LED led1(13, 500);
LED led2(12, 1000);

void setup() {
}

void loop() {
  led1.blink();
  led2.blink();
}
```

Несколько асинхронно мигающих светодиодов

- uint16_t - unsigned int (0...65535)
- uint32_t – unsigned long (0...4294967295)
- конструктор имеет такое же имя, как и сам класс
- blink() – метод класса, функция мигания

СПАСИБО ЗА ВНИМАНИЕ!