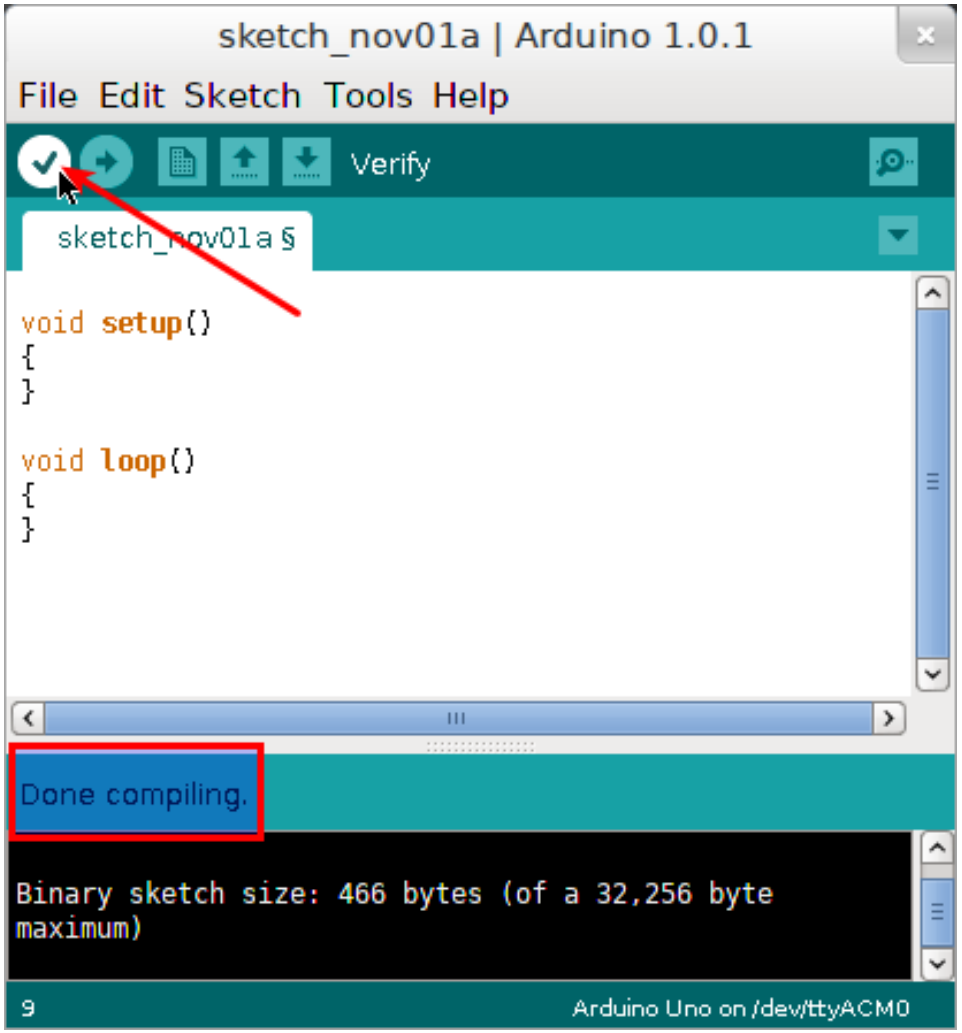
Arduino

С ЧЕМ ЕДЯТ ?

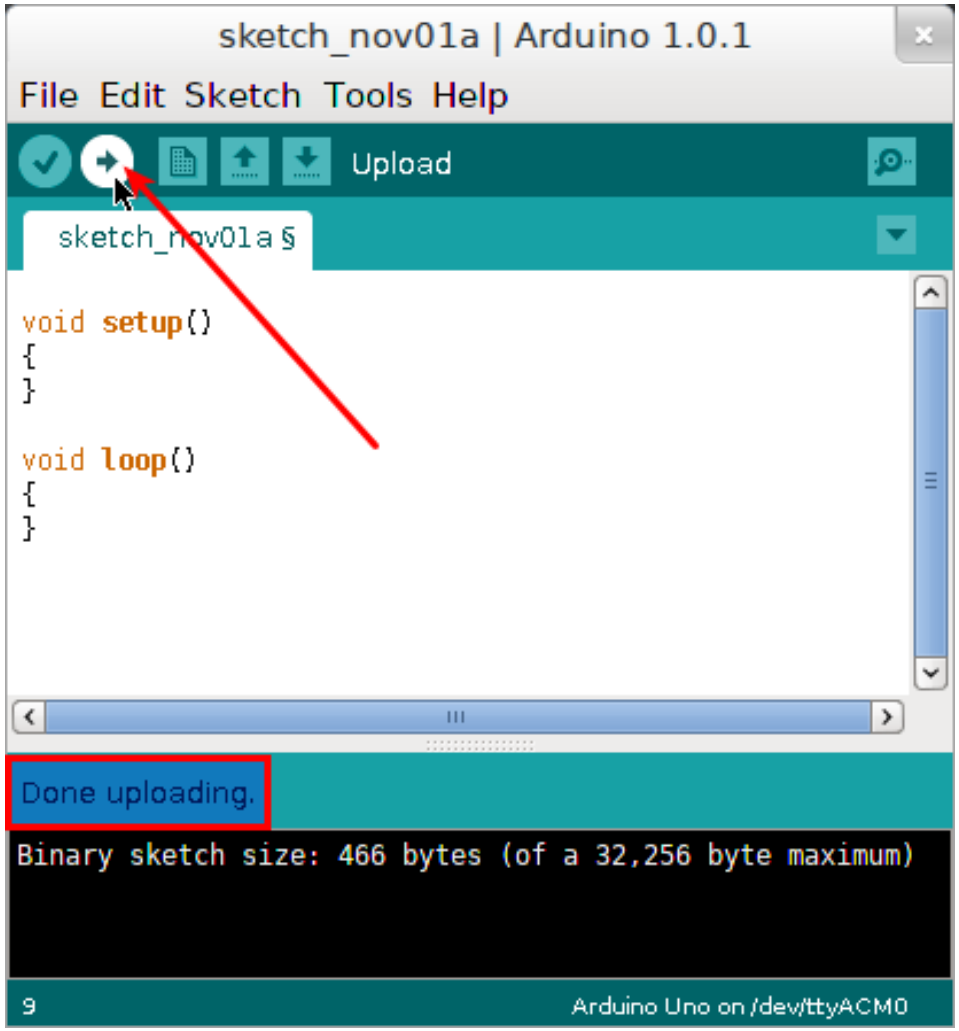
Первая программа

Arduino IDE компиляция



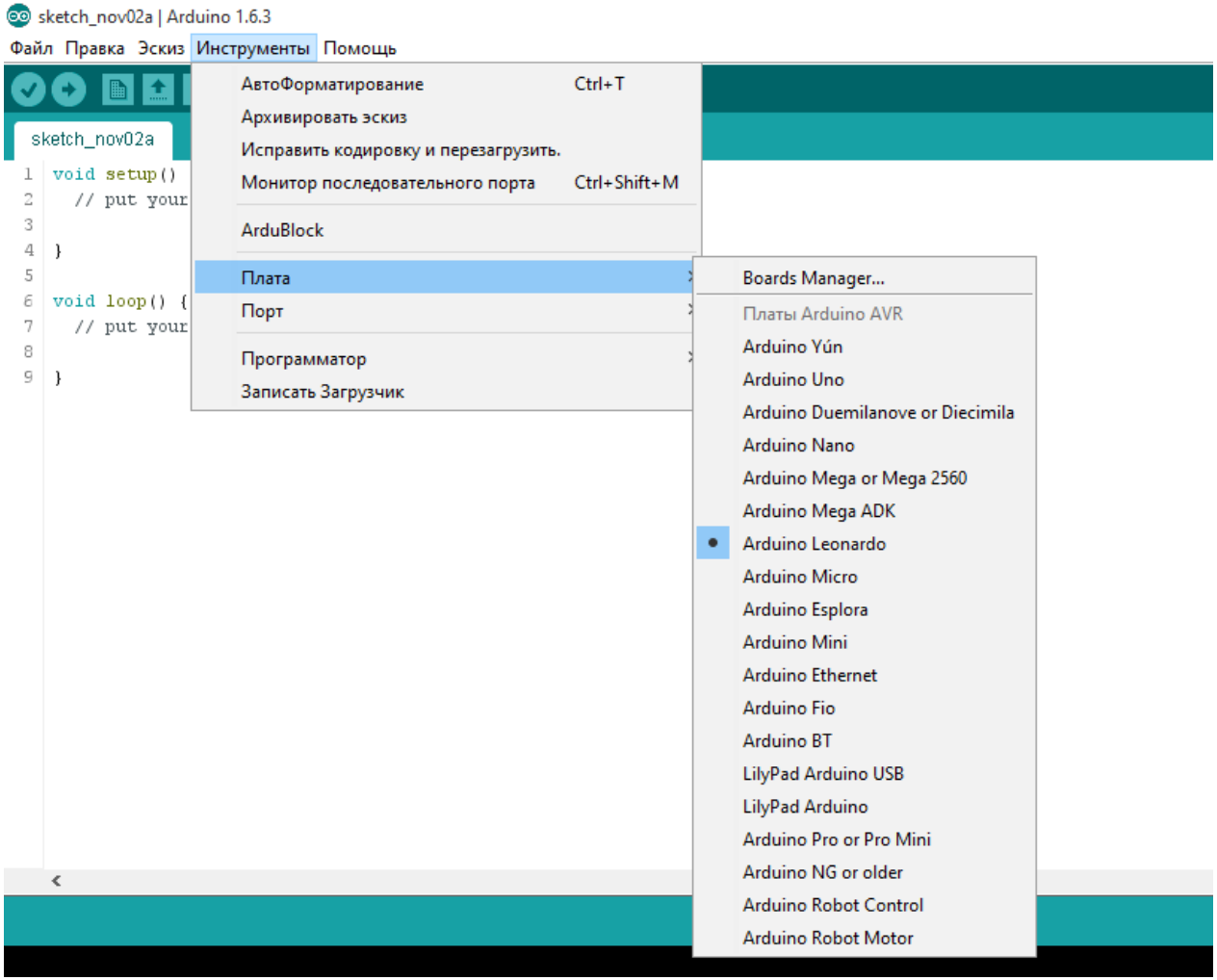
Первая программа

Arduino IDE загрузка



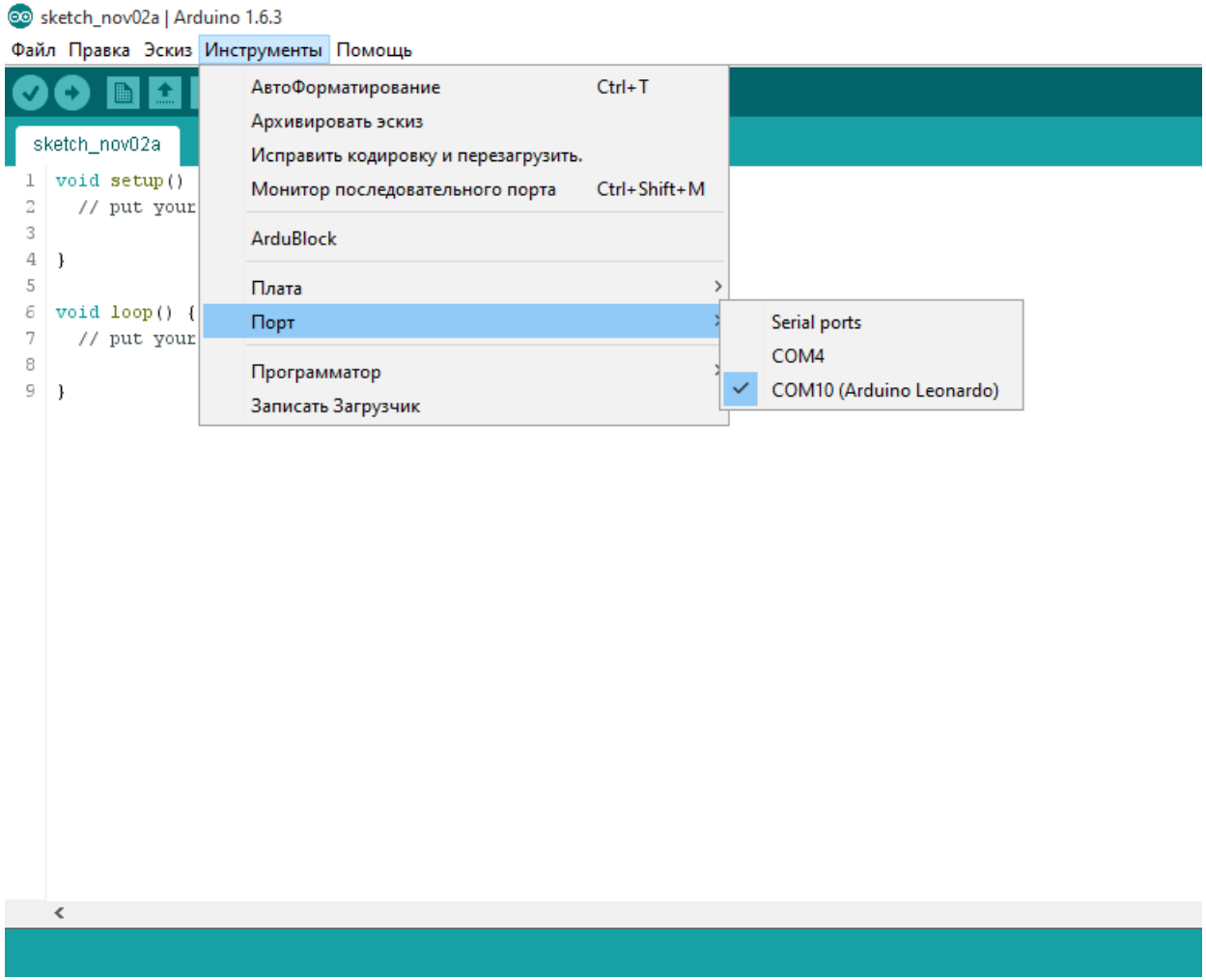
Первая программа

Arduino IDE выбор платы



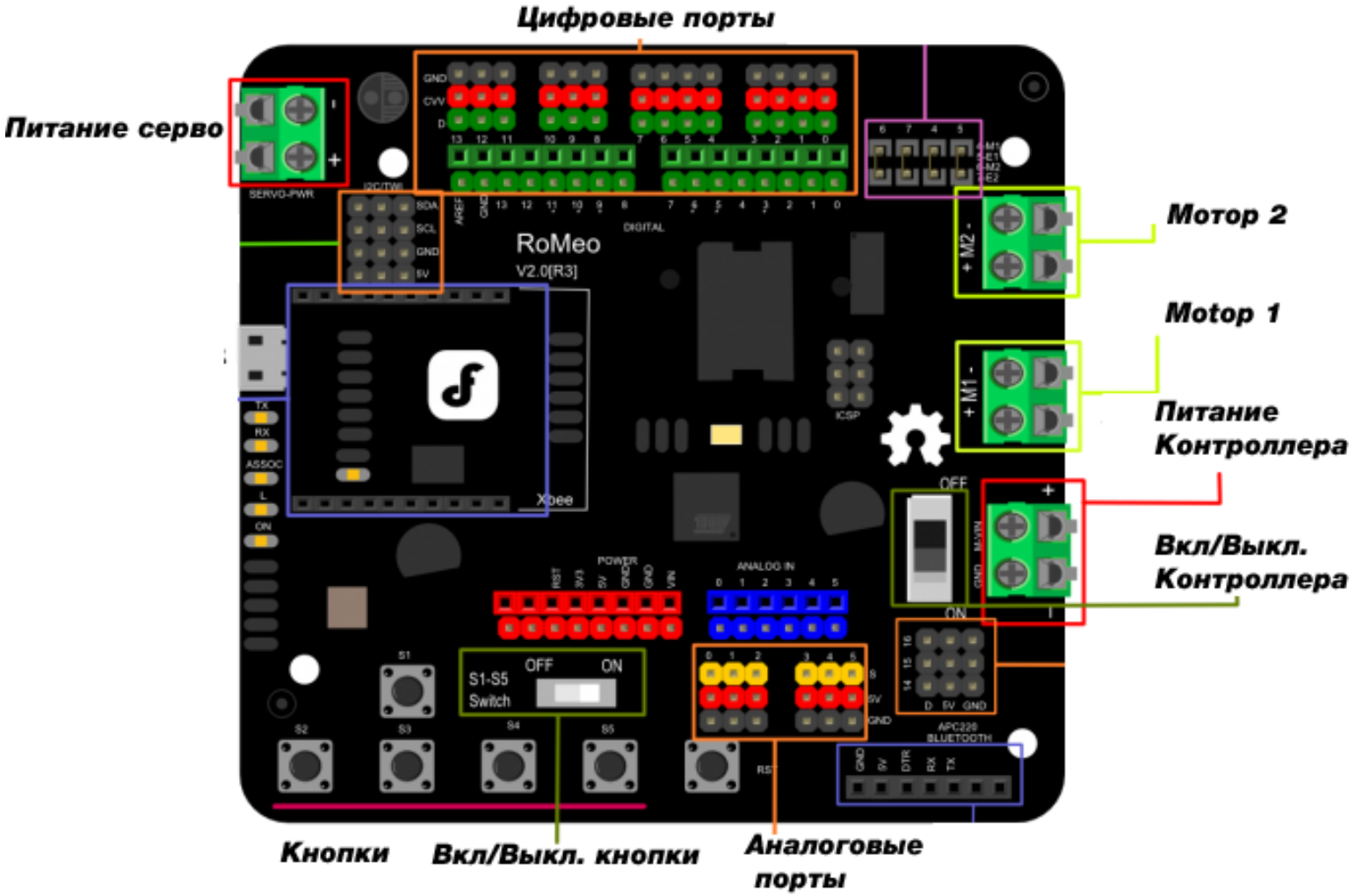
Первая программа

Arduino IDE выбор порта



Немного о контроллере

Arduino ROMEO



Структура программы

```
1  void setup() {
2    // put your setup code here, to run once:
3
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8
9  }
```

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

Мигающий светодиод

```
1
2
3  void setup() {
4
5    pinMode(13, OUTPUT);
6    Serial.begin(115200);
7
8  }
9
10 void loop() {
11
12     digitalWrite(13,LOW);
13     delay(1000);
14     digitalWrite(13,HIGH);
15     delay(1000);
16
17
18 }
```

Структура программы

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


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

Структура программы

{} фигурные скобки

```
type function()
{
  statements;
}
```

; точка с запятой

```
int x = 13;    //  объявляет переменную 'x' как целое 13
```

Переменные

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

```
if (inputVariable < 100) // проверяем, не меньше ли 100 переменная
{
  inputVariable = 100;   // если так, присваиваем ей значение 100
}
delay(inputVariable);    // используем переменную как паузу
```

Границы переменных

```
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
```

Типы переменных

byte

Байт хранит 8-битовое числовое значение без десятичной точки. Он имеет диапазон от ***0*** до ***255***.

```
byte someVariable = 180;    // объявление 'someVariable'
                            // как имеющей тип byte
```

int

Целое — это первый тип данных для хранения чисел без десятичной точки, и хранит 16-битовое значение в диапазоне от ***32767*** до ***-32768***.

```
int someVariable = 1500;    // объявляет 'someVariable'
                            // как переменную целого типа
```

Типы переменных

long

Тип данных увеличенного размера для больших целых, без десятичной точки, сохраняемый в 32-битовом значении с диапазоном от ***2147383647*** до ***-2147383648***.

```
long someVariable = 90000; // декларирует 'someVariable'
                           // типа long
```

float

Тип данных для чисел с плавающей точкой или чисел, имеющих десятичную точку. Числа с плавающей точкой имеют большее разрешение, чем целые и сохраняются как 32-битовые значения в диапазоне от ***3.4028235E+38*** до ***-3.4028235E+38***.

```
float someVariable = 3.14; // объявление 'someVariable'
                           //   как floating-point тип
```

Арифметика

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

Смешанное присваивание

```
x ++          // то же, что x = x + 1, или увеличение x на +1
x --          // то же, что x = x - 1, или уменьшение x на -1
x += y        // то же, что x = x + y, или увеличение x на +y
x -= y        // то же, что x = x - y, или уменьшение x на -y
x *= y        // то же, что x = x * y, или умножение x на y
x /= y        // то же, что x = x / y, или деление x на y
```

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

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

Это Булевы константы, определяющие логические уровни. FALSE легко определяется как 0 (ноль), а TRUE, как 1, но может быть и чем-то другим, отличным от нуля. Так что в Булевом смысле -1, 2 и 200 — это всё тоже определяется как TRUE.

```
if (b == TRUE);
{
  doSomething;
}
```

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).

```
digitalWrite(13, HIGH);
```

Управление программой

if...else

Конструкция if...else позволяет сделать выбор «либо, либо». Например, если вы хотите проверить цифровой вход и выполнить что-то, если он HIGH, или выполнить что-то другое, если он был LOW, вы должны записать следующее:

```
if (inputPin == HIGH)
{
  doThingA;
}
else
{
  doThingB;
}
```

```
if (inputPin < 500)
{
  doThingA;
}
else if (inputPin >= 1000)
{
  doThingB;
}
else
{
  doThingC;
}
```

Управление программой

for

Конструкция for используется для повторения блока выражений, заключённых в фигурные скобки заданное число раз. Наращиваемый счётчик часто используется для увеличения и прекращения цикла. Есть три части, разделённые точкой с запятой, в заголовке цикла for:

```
for (int i=0; i<20; i++)   // декларирует i, проверяет меньше ли
{                          // чем 20, увеличивает i на 1
  digitalWrite(13, HIGH); // устанавливает вывод 13 в ON
  delay(250);              // пауза в 1/4 секунды
  digitalWrite(13, LOW);   // сбрасывает вывод 13 в OFF
  delay(250);              // пауза в 1/4 секунды
}
```

Управление программой

while

Цикл while продолжается, и может продолжаться бесконечно, пока выражение в скобках не станет false (ложно). Что-то должно менять проверяемую переменную, иначе из цикла никогда не выйти. И это должно быть в вашем коде, как, скажем, увеличение переменной, или внешнее условие, как, например, проверяемый сенсор.

```
While (someVariable < 200) // проверяет, меньше ли 200
{
  doSomething;             // выполняет выражение в скобках
  someVariable++;          // увеличивает переменную на 1
}
```

Цифровой ввод/вывод

pinMode (pin, mode)

Используется в void setup () для конфигурации заданного вывода, чтобы он работал на вход (INPUT) или на выход (OUTPUT).

```
pinMode(pin, OUTPUT);      // устанавливает 'pin' на выход
```

digitalRead (pin)

Считывает значение заданного цифрового вывода (pin) и возвращает результат HIGH или LOW. Вывод должен быть задан либо как переменная, либо как константа (0-13).

```
value = digitalRead(Pin);    // задаёт 'value' равным
                             // входному выводу 'Pin'
```

Цифровой ввод/вывод

digitalWrite (pin, value)

Выводит либо логический уровень HIGH, либо LOW (включает или выключает) на заданном цифровом выводе pin. Вывод может быть задан либо как переменная, либо как константа (0-13).

```
digitalWrite(pin, HIGH);
```

Цифровой ввод/вывод

```
int led   = 13;   // соединяем LED с выводом 13
int pin   =  7;   // соединяем кнопку с выводом 7
int value =  0;// переменная для хранения прочитанного значения

void setup()
{
  pinMode(led, OUTPUT);     // задаём вывод 13 как выход
  pinMode(pin, INPUT);      // задаём вывод 7 как вход
}

void loop()
{
  value = digitalRead(pin);  // задаём 'value' равной
                             // входному выводу
  digitalWrite(led, value);  // устанавливаем 'led' в
}                            // в значение кнопки
```

Аналоговый ввод/вывод

analogRead (pin)

Считывает значение из заданного аналогового входа (pin) с 10-битовым разрешением. Эта функция работает только на аналоговых портах (0-5). Результирующее целое значение находится в диапазоне от 0 до 1023.

```
value = analogRead(pin);   // задаёт 'value' равным 'pin'
```

Аналоговый ввод/вывод

analogWtite (pin, value)

Записывает псевдо-аналоговое значение, используя схему с широтно-импульсной модуляцией (PWM), на выходной вывод, помеченный как PWM. Эта функция работает на выводах 3, 5, 6, 9, 10 и 11. Значение может быть задано как переменная или константа в диапазоне 0-255.

```
analogWrite(pin, value);  // записывает 'value' в аналоговый 'pin'
```

Последовательный обмен

Serial.begin (rate)

Открывает последовательный порт и задаёт скорость для последовательной передачи данных. Типичная скорость обмена для компьютерной коммуникации — 9600, хотя поддерживаются и другие скорости.

```
void setup()
{
  Serial.begin(9600);    //  открывается последовательный порт
}                        //  задаётся скорость обмена 9600
```

Serial.println (data)

Передаёт данные в последовательный порт, сопровождая автоматическим возвратом каретки и переходом на новую строку. Команда такая же, что и Serial.print(), но легче для последующего чтения на данных в терминале.

```
Serial.println(analogValue);   //  отправляет значение
                               //  'analogValue'
```

Последовательный обмен

```
void setup()
{
  Serial.begin(9600);                  // задаём скорость 9600 bps
}

void loop()
{
  Serial.println(analogRead(0)); // шлём аналоговое значение
  delay(1000);                         // пауза 1 секунда
}
```

Примеры

Мигающий светодиод

```
3  void setup() {
4
5    pinMode(13, OUTPUT);
6    Serial.begin(115200);
7
8  }
9
10 void loop() {
11
12     digitalWrite(13,LOW);
13     delay(1000);
14     digitalWrite(13,HIGH);
15     delay(1000);
16
17
18 }
```

Примеры

Затухающий светодиод

```
3  void setup() {
4
5    pinMode(13, OUTPUT);
6    Serial.begin(115200);
7
8  }
9
10 void loop() {
11
12     analogWrite(13,255);
13     delay(1000);
14     analogWrite(13,100);
15     delay(1000);
16     analogWrite(13,50);
17     delay(1000);
18     analogWrite(13,25);
19     delay(1000);
20     analogWrite(13,0);
21     delay(1000);
22
23 }
```

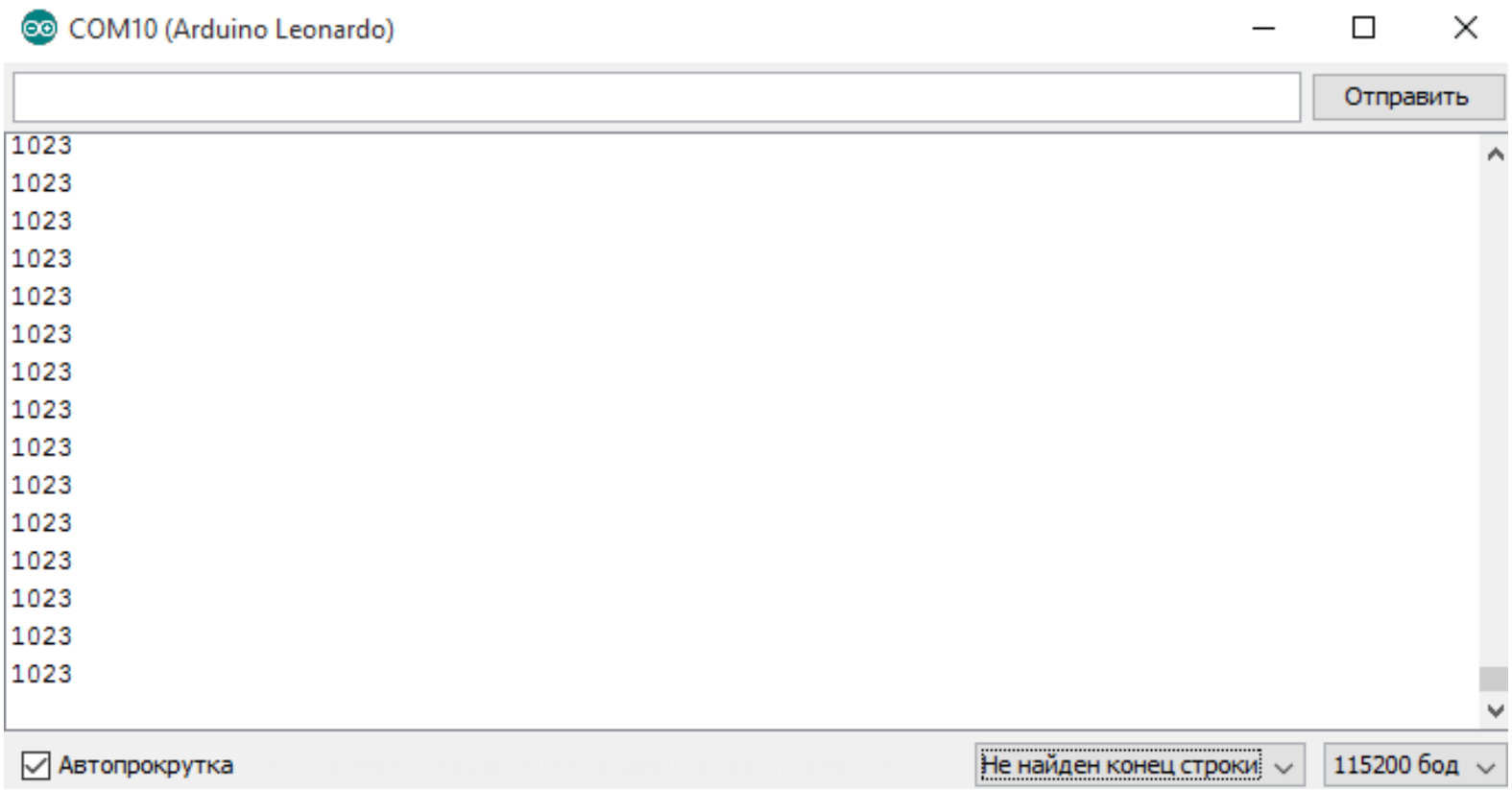
Примеры

Аналоговый вывод

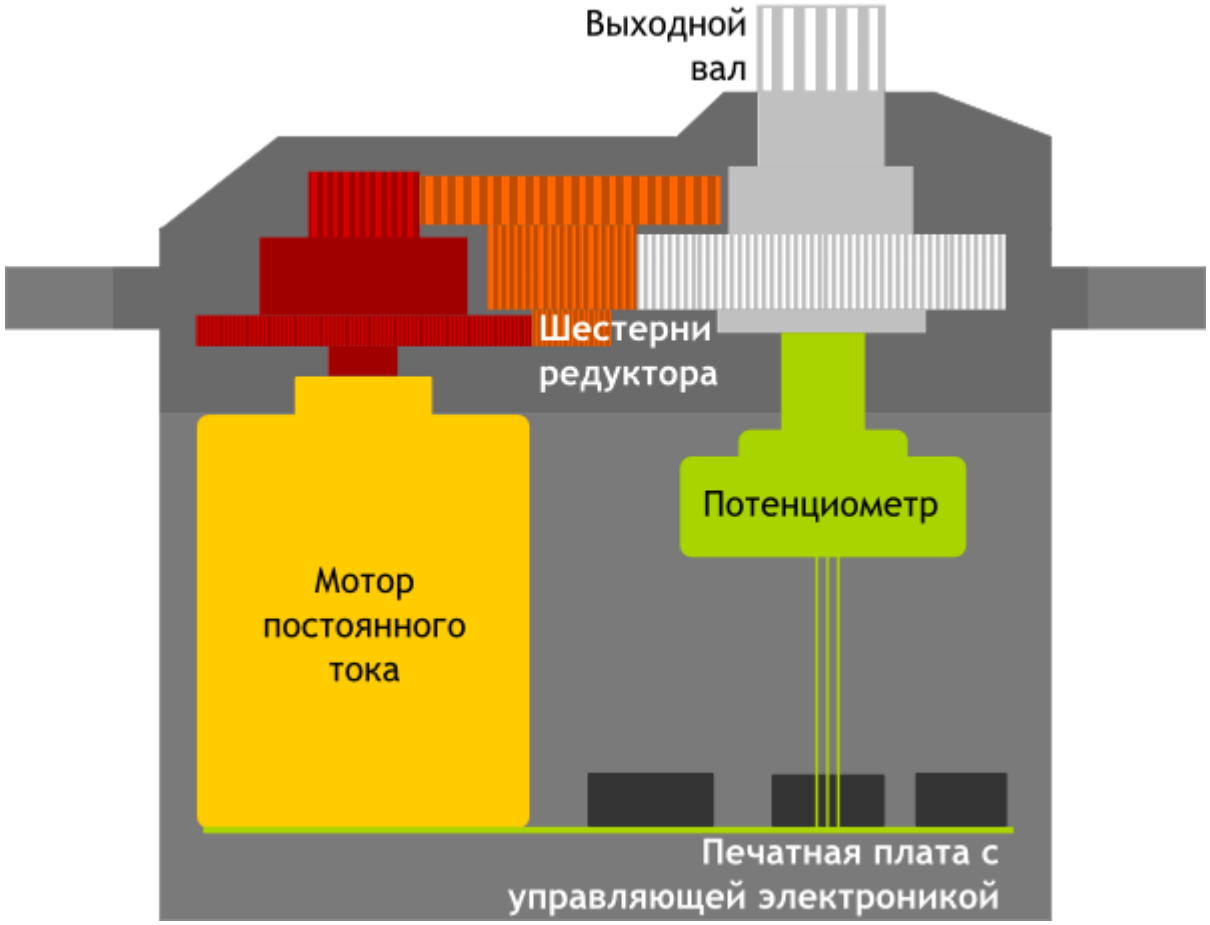
```
2  void setup()
3  {
4    Serial.begin(115200);
5  }
6
7
8  void loop() {
9
.0   int sensorValue = analogRead(A0);
.1   Serial.println(sensorValue);
.2   delay(100);
.3 }
```

Примеры

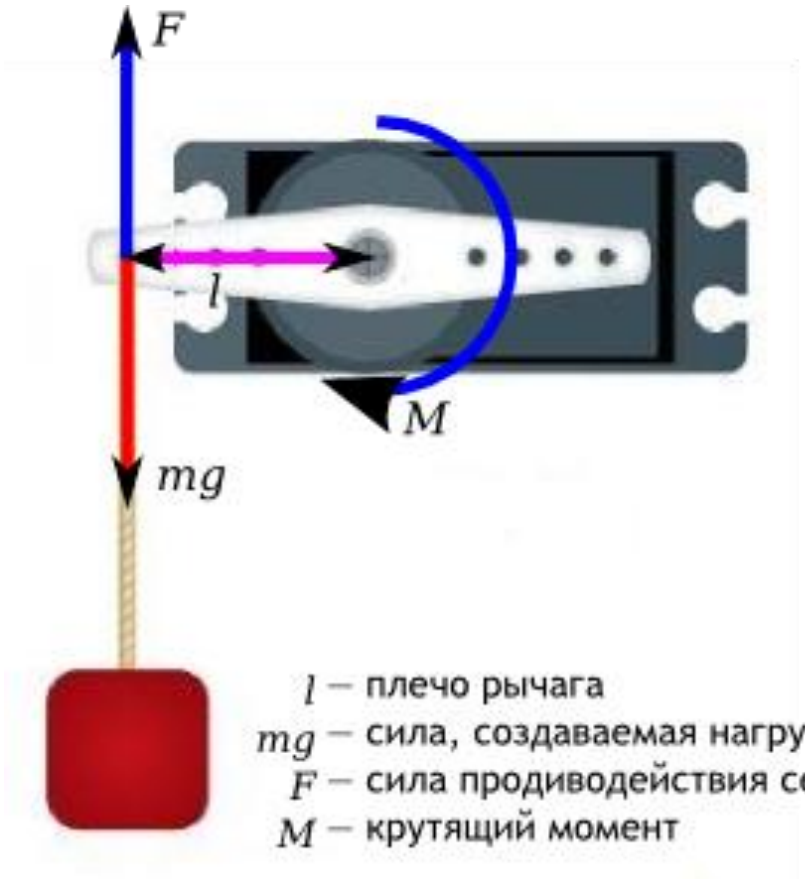
Монитор COM порта



Сервоприводы



Характеристики сервоприводов



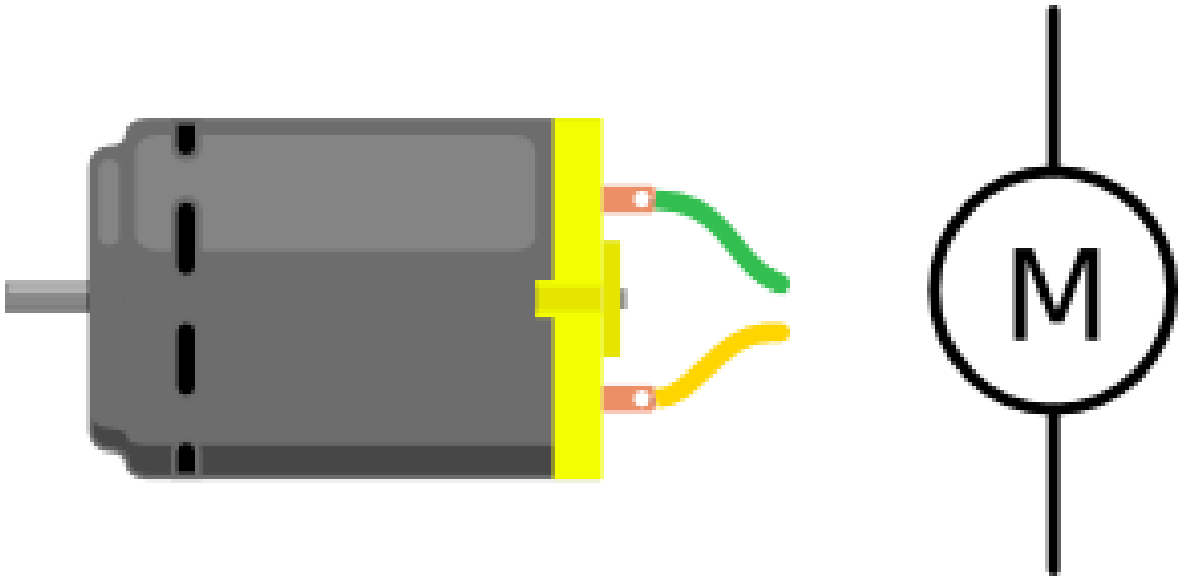
l – плечо рычага
mg – сила, создаваемая нагрузкой
F – сила продиводействия сервопривода
M – крутящий момент

Материалы шестерней

Пример сервопривод

```
3  Servo myservo;
4
5  int pos = 0;
6
7  void setup()
8  {
9    myservo.attach(8);
10 }
11
12 void loop()
13 {
14   for(pos = 0; pos <= 180; pos += 1)
15   {
16     myservo.write(pos);
17     delay(15);
18   }
19   for(pos = 180; pos>=0; pos-=1)
20   {
21     myservo.write(pos);
22     delay(15);
23   }
24 }
```

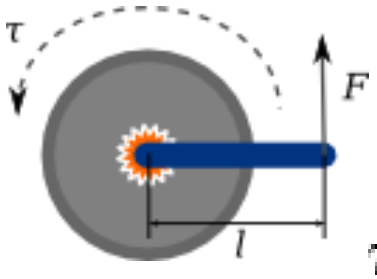
Мотор



Мотор

Основные характеристики

Рекомендуемое (номинальное) напряжение	V	Вольт
Потребляемый ток без нагрузки	I_F	Ампер
Потребляемый ток при блокировке	I_S	Ампер
Скорость вращения без нагрузки	ω	$с^{-1}$
Максимальный крутящий момент	τ	Н×м



Крутящий момент определяет какая сила воздействует на точку рычага на заданном расстоянии от оси вращения.

$$\tau = F \times l$$

Мотор пример

```
#include "DFMobile.h"

DFMobile Robot (4,5,7,6);

void setup()
{
  int i;
  for(i=4;i<=7;i++)
  pinMode(i, OUTPUT);
  Robot.Direction (LOW,HIGH);
  Serial.begin(115200);
}

void loop ()
{
      Robot.Speed (255,255);
      delay(1000);
      Robot.Speed (0,0);
      delay(1000);
      Robot.Speed (-255,-255);
      delay(1000);
}
```

Контакты

ШКОЛЬНЫЙ ТЕХНОПАРК

Сайт: технопарксабы.рф

Почта: technoparksabi@gmail.com

Подписывайтесь на наши группы в социальных сетях