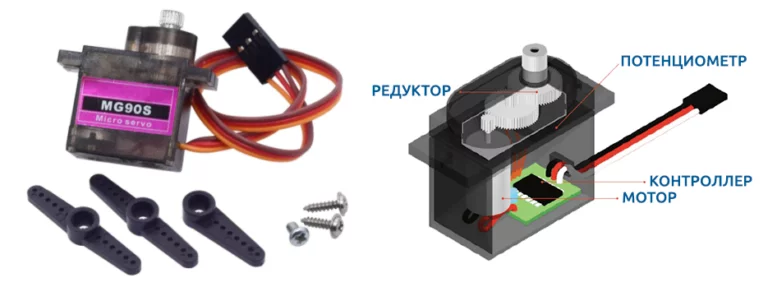
## **Arduino и сервопривод** Описание

Сервопривод, он же сервомашинка, он же просто серво – простейший “модельный” актуатор, использовался в радиоуправляемых моделях ещё до появления Ардуино. Сервопривод поворачивает свой выходной вал в диапазоне ~180 градусов и обладает вполне приличным моментом для перемещения частей механизмов. В маленьком корпусе располагаются:

* Коллекторный моторчик
* Редуктор (пластиковый или металлический)
* Потенциометр обратной связи
* Контроллер (драйвер мотора, обратная связь, управление по интерфейсу PWM)

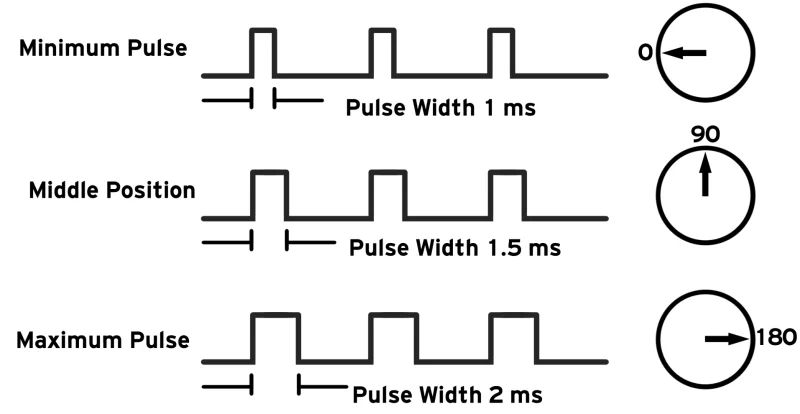
Серво комплектуется набором “качалок” и винтами для крепления



Характеристики 9-граммового сервопривода:

* Напряжение питания: 3.. 7.2V
* Крутящий момент:
  + 1.2кг/см при 4.8V
  + 1.6кг/см при 6.0V
* Рабочий угол: 160.. 180 градусов
* Скорость (без нагрузки): 180 градусов за 0.3с
* Интерфейс: PWM

Сервопривод управляется ШИМ сигналом, точнее длиной импульса: минимальная (0 градусов) и максимальная (~180 градусов) длина импульса колеблется в зависимости от модели и производителя сервопривода.

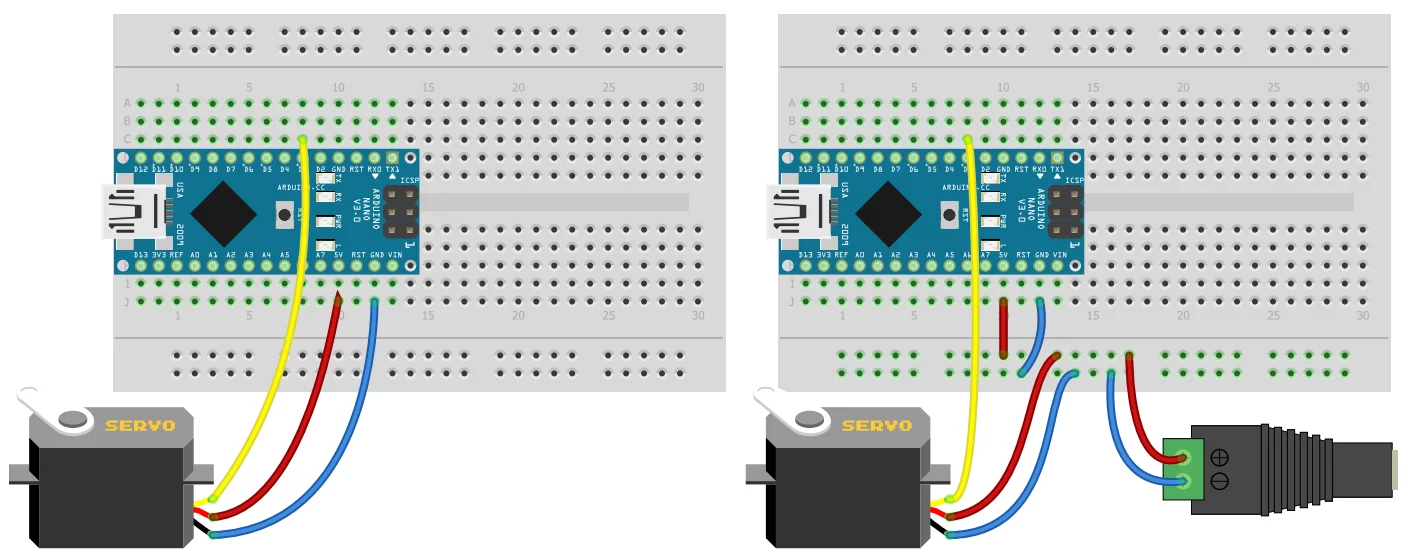


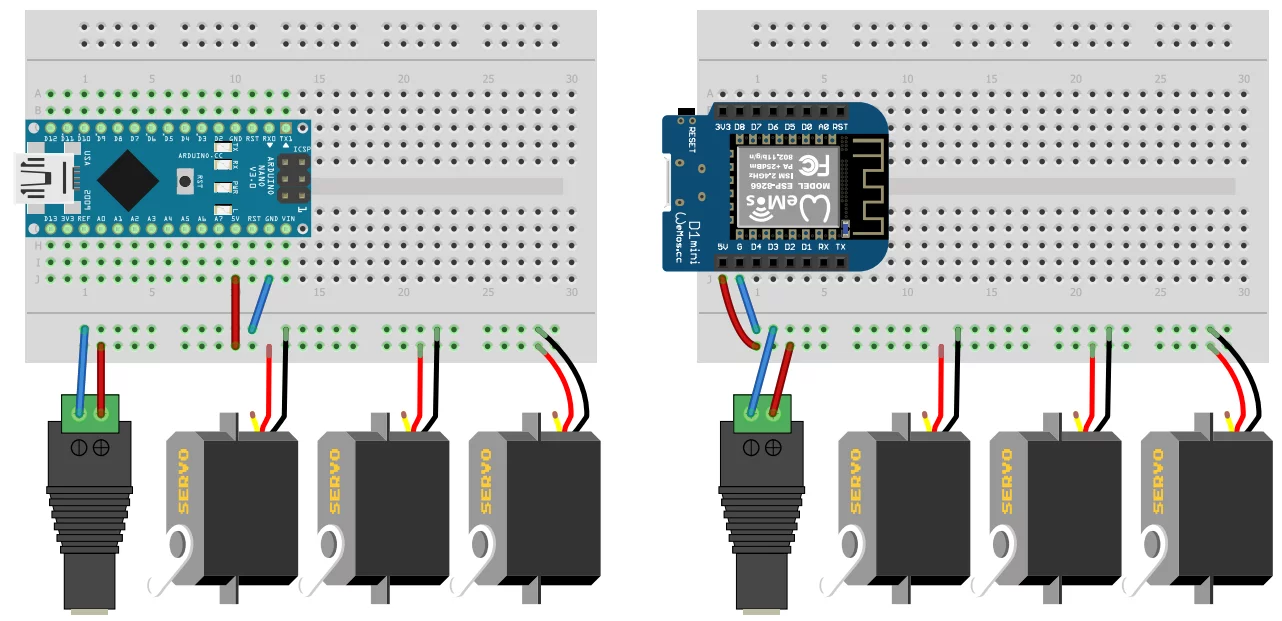
## Подключение

Провода:

* Коричневый: GND
* Красный: VCC
* Жёлтый: цифровой пин

Примечание: сервопривод потребляет довольно приличный ток (стартовый до 1А), поэтому рекомендуется питать его от внешнего источника. При питании от компьютера напряжение может просесть и МК перезагрузится, при высокой нагрузке (с 2 и больше приводов) может выгореть защитный диод на плате Arduino!





## Библиотеки

Для управления сервоприводом можно использовать стандартную библиотеку **Servo**. Эта библиотека отбирает у микроконтроллера первый таймер, поэтому в некоторых случаях будет удобно использовать библиотеку SoftServo. Стандартная библиотека управляет сервоприводом очень резко, на максимальной скорости, поэтому для реальных применений лучше использовать библиотеку плавного движения серво – ServoSmooth.

## Servo.h

Библиотека стандартная и идёт в комплекте с Arduino IDE. Подключаем и создаём объект сервопривода. При создании никакие аргументы не передаются:

#include <Servo.h>

Servo myservo;

Библиотека имеет следующие методы:

uint8\_t attach(int pin); // "подключить" с указанием пина

uint8\_t attach(int pin, int min, int max); // "подключить" с указанием пина и мин. макс. сигнала

void detach(); // отключить

void write(int value); // повернуть на угол в градусах

void writeMicroseconds(int value); // повернуть на длину импульса

При “подключении” серво через

attach(pin)

 диапазон длины импульса устанавливается стандартный: 544.. 2400 мкс (задан в библиотеке). Если есть желание настроить серво идеально, чтобы она работала на весь диапазон – нужно попробовать покрутить её через

writeMicroseconds()

 в крайних значениях диапазона и найти минимум и максимум, при которых серво “упирается”.

**Крутим туда сюда быстро**

Вращаем туда сюда быстро (как blink, только серво):

#include <Servo.h>

Servo myservo;

void setup() {

myservo.attach(3); // подключаем на пин 3

}

void loop() {

myservo.write(0); // поворот на 0 градусов

delay(1000);

myservo.write(180); // поворот на 180 градусов

delay(1000);

}

**Крутим туда сюда плавно**

Поворачиваем туда сюда плавно, блокирующее выполнение:

#include <Servo.h>

Servo myservo;

void setup() {

myservo.attach(3);

}

void loop() {

for (int i = 0; i <= 180; i++) {

myservo.write(i);

delay(15);

}

for (int i = 180; i > 0; i--) {

myservo.write(i);

delay(15);

}

}

**Крутим туда сюда плавно асинхронно**

Поворачиваем туда сюда плавно, асинхронное выполнение:

#include <Servo.h>

Servo myservo;

void setup() {

myservo.attach(3);

}

uint32\_t tmr; // переменная таймера

int val = 0; // переменная яркости

int dir = 2; // скорость и направление яркости

void loop() {

// асинхронный таймер на миллис

if (millis() - tmr >= 20) {

tmr = millis();

val += dir; // п рибавляем скорость

if (val >= 180|| val <= 0) dir = -dir; // разворачиваем

myservo.write(val); // применяем

}

}