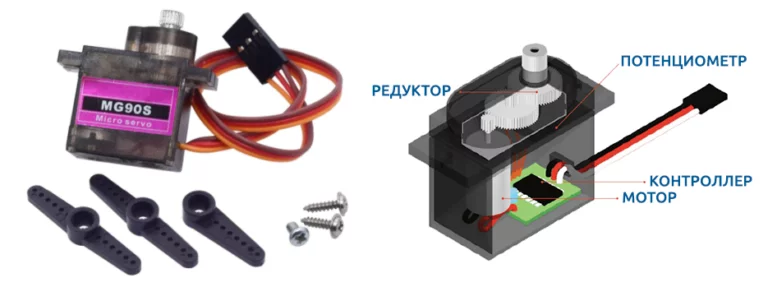
**Сервопривод на 180 градусов**

**Сервопривод** – простейший “модельный” актуатор.

В маленьком корпусе располагаются:

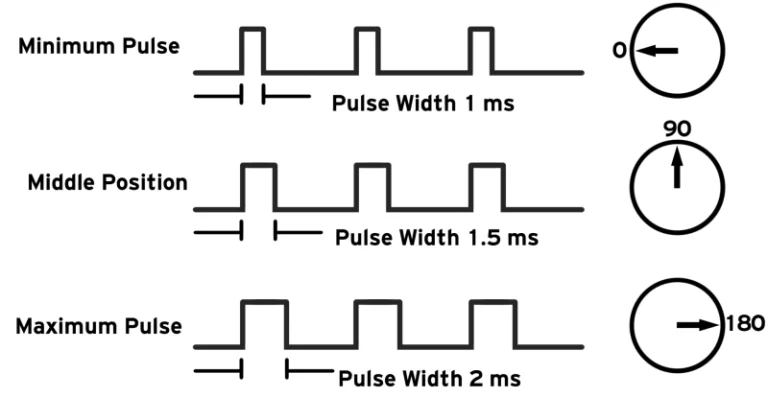
* Коллекторный моторчик;
* Редуктор (пластиковый или металлический);
* Потенциометр обратной связи;
* Контроллер (драйвер мотора, обратная связь, управление по интерфейсу PWM).

Сервопривод комплектуется набором “качалок” и винтами для крепления



## Характеристики 9-граммового сервопривода:

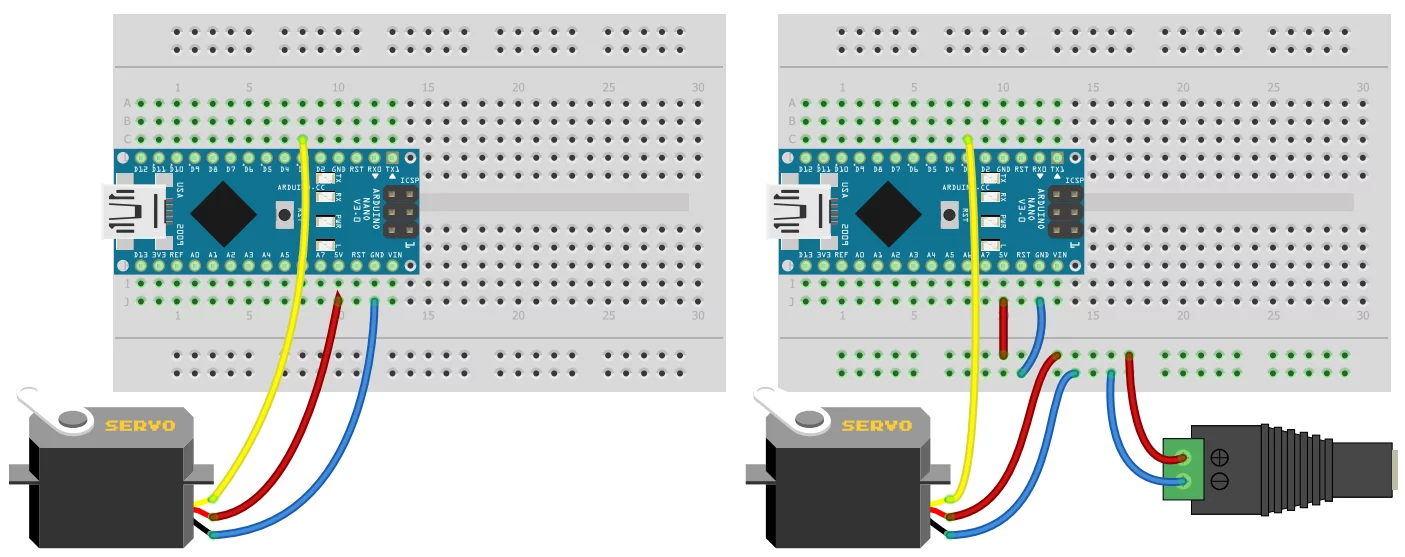
* Напряжение питания: 3-7.2V
* Крутящий момент:
  + 1.2кг/см при 4.8V
  + 1.6кг/см при 6.0V
* Рабочий угол: 160-180 градусов
* Скорость (без нагрузки): 180 градусов за 0.3с
* Интерфейс: PWM

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, может выгореть защитный диод на плате Arduino!**

**Библиотеки**

Стандартная библиотека Servo.h управляет сервоприводом очень резко, на максимальной скорости, поэтому для реальных применений лучше использовать библиотеку плавного движения ServoSmooth.

**Servo.h**

**Подключение библиотеки:**

#include <Servo.h>

Servo myservo;

**Библиотека Servo.h имеет следующие методы:**

uint8\_t attach(int pin); - подключить сервопривод 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). Чтобы серво работал на весь диапазон – нужно попробовать покрутить его через

writeMicroseconds()

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

**ServoSmooth.h**

#include "ServoSmooth.h" - подключили библиотеку

ServoSmooth servo; - создали объект

ServoSmooth servo(270); - создали с указанием максимальным углом сервопривода

Управления сервоприводом:

По инициализации attach()

  есть несколько вариантов:

* attach(pin);

 - подключит серво на указанный pin, угол поворота будет установлен на **0** градусов. Длина импульса будет стандартная, 500-2400 мкс

* attach(pin, target);

 - подключит серво на указанный pin, угол поворота будет установлен на target градусов. Длина импульса будет стандартная, 500-2400 мкс

* attach(pin, min, max);

 - подключит серво на указанный pin, угол поворота будет установлен на **0** градусов. Длина импульса\* будет установлена min и max соответственно.

* attach(pin, min, max, target);

 - подключит серво на указанный pin, угол поворота будет установлен на target градусов. Длина импульса\* будет установлена min и max соответственно.

### Плавный пуск сервопривода

Сервопривод не имеет обратной связи по углу (для программы), поэтому при запуске будет "резко" повёрнут на стартовый угол (0 по умолчанию или на указанный в

attach(pin, target)

Есть два варианта избежать резких рывков в механизме при запуске программы:

* Заранее знать, на какой угол физически повёрнут привод при запуске и передать его в

attach(pin, target)

. Как узнать? Зависит от конкретной задачи и логики работы программы. Можно запоминать положение сервы в ЕЕПРОМ и восстанавливать при запуске, можно устанавливать серво в один и тот же угол перед выключением/перезагрузкой системы, и т.д.

* Воспользоваться фичей

smoothStart()

, которая появилась в версии 3.2 данной библиотеки. Работает она очень просто: аттачит и детачит сервопривод с периодом в пару десятков миллисекунд, таким образом привод плавно движется до заданного угла из любого начального положения. Вызывать

smoothStart()

 нужно однократно (при старте программы) сразу после

attach(pin, target)

 в блоке

setup()

. **Внимание!** Функция блокирующая, выполнение занимает 900 миллисекунд. Период "рывка" сервопривода выбран минимальный, при котором серво начинает понимать, чего от неё хотят. Период довольно большой, поэтому движение к заданной позиции происходит рывками, но в целом гораздо плавнее, чем без

smoothStart()

. В массивном   механизме рывки практически незаметны!

servo.attach(2, 35); // стартовый угол 35 градусов

servo.smoothStart(); // "плавно" движемся к нему

### Управление

Движение серво происходит автоматически в методе

tick()

, нам нужно всего лишь вызывать его как можно чаще в

loop()

 (

tick()

 имеет встроенный таймер на 20 миллисекунд). Также есть метод

tickManual()

, который поворачивает серву на следующий "шаг" при каждом вызове (тот же

tick()

, но не имеет своего таймера). Оба метода

tick()

 возвращают

**false**

, пока серво движется, и

**true**

, когда серво достигла установленного угла, это можно использовать. Также серво автоматически отключается от управления при достижении заданного угла поворота (это уменьшает жужжание серво в простое). Эту функцию можно отключить, вызвав

setAutoDetach(**false**)

. Инструменты для управления движением привода:

* setTarget(длина);

 - устанавливает целевую позицию для серво в величине длина импульса, мкс (~500-2400)

* setTargetDeg(угол);

 - устанавливает целевую позицию для серво в градусах (0-180)

* setSpeed(скорость);

 - установка максимальной скорости (больше нуля) в градусах в секунду

* setAccel(ускорение);

 - установка ускорения (float числа 0.01 - 1.0). Можно больше 1, будет ещё резче. Если установить ускорение

0

 - оно будет отключено и серво будет двигаться по профилю постоянной скорости (с бесконечным ускорением)

* + Если передавать ускорение в целых числах (**с версии 3.7** библиотеки) - ускорение будет установлено в градусах/сек/сек. Рабочий диапазон ускорений

1 - 1500

, чем больше - тем резче. При значении

0

 ускорение будет отключено.

* start();

 - автоматический attach + разрешает работу tick - серво движется к заданной позиции

* stop();

 - detach + запрещает работу tick - серво останавливается

Полезные вспомогательные методы для различных ситуаций:

* setDirection(напр);

 - принимает

NORMAL

 (false) или

REVERSE

 (true), меняет направление серво

* setCurrent(длина);

 - установка текущей позиции в мкс (500 - 2400). Может пригодиться в ситуации, когда мы знаем реальный угол серво и хотим сообщить о нём программе, чтобы алгоритм не дёргал привод.

* setCurrentDeg(угол);

 - установка текущей позиции в градусах (0-180). Зависит от min и max.

* getCurrent();

 - получение текущей позиции в мкс (500 - 2400)

* getCurrentDeg();

 - получение текущей позиции в градусах (0-180). Зависит от min и max

* getTarget();

 - получение целевой позиции в мкс (500 - 2400)

* getTargetDeg();

 - получение целевой позиции в градусах (0-180). Зависит от min и max

* setMaxAngle();

 - установка макс. угла серво, по умолчанию

180

. Позволяет удобно работать с разными сервами (на 270 и 360 градусов)