Сервопривод – это такой вид привода, который может точно управлять параметрами движения. Другими словами, это двигатель, который может повернуть свой вал на определенный угол и поддерживать непрерывно это положение.

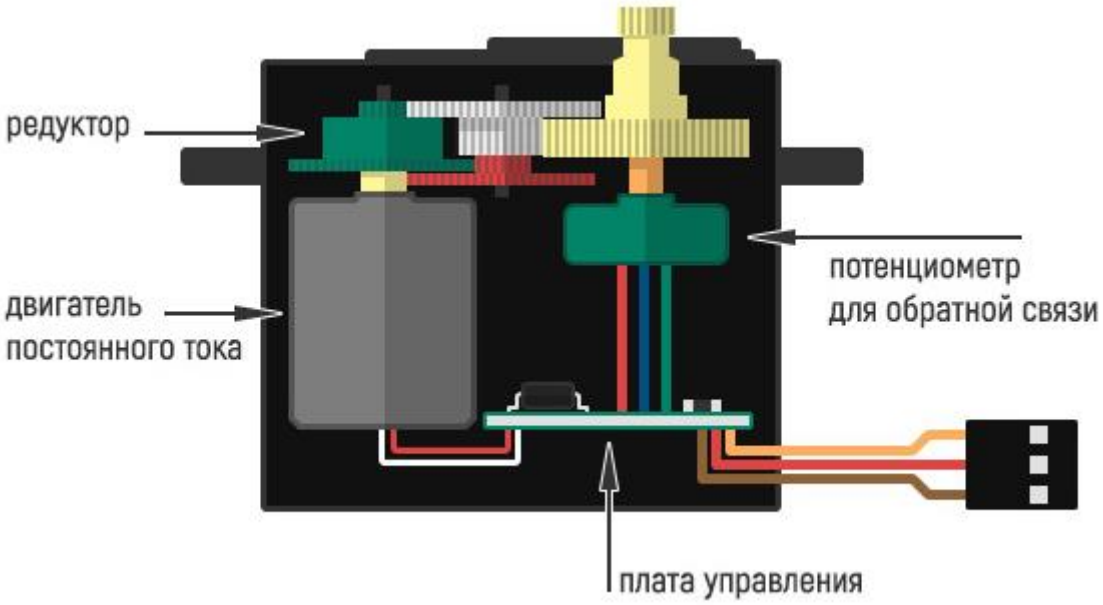
Сервоприводы могут быть использованы для управления перемещением:

частей робота (рук, ног, головы);

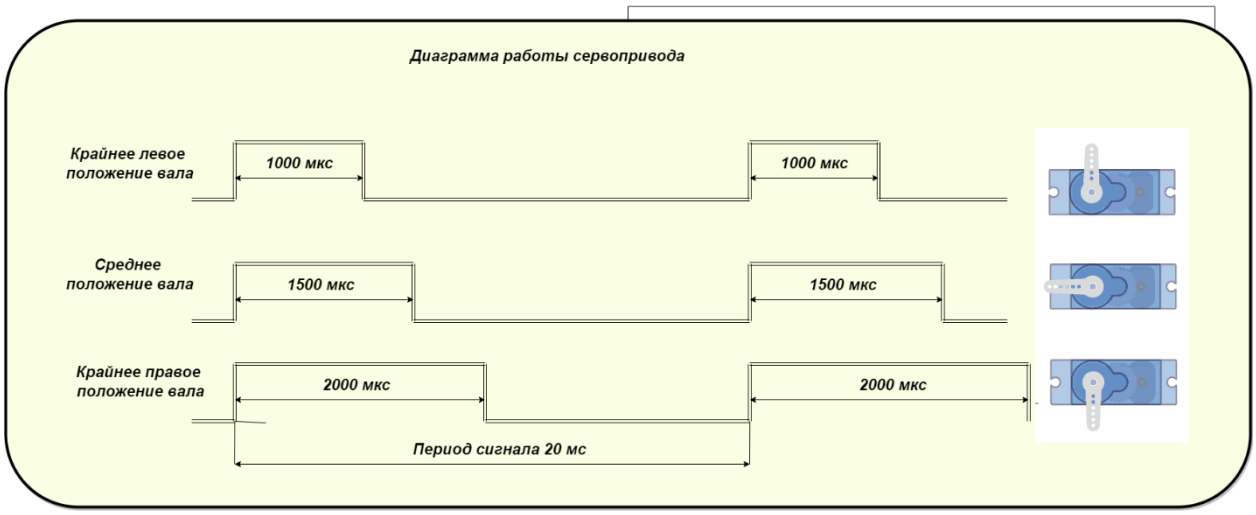
элементов управления модели транспортного средства — самолета, вертолета, автомобиля, катера;

поворотного устройства для видеокамеры;

различных механизмов.



Для управления сервоприводом используется сигнал специальной формы. Временная диаграмма сигнала приведена на рисунке ниже. Частота повторения сигнала составляет 20 мс. Длина импульса соответствует заданному углу поворота вала сервопривода. Импульс в 1000 мкс соответствует крайнему левому положению, 2000 мкс — крайнему правому. 1500 мкс соответствует среднему положению.



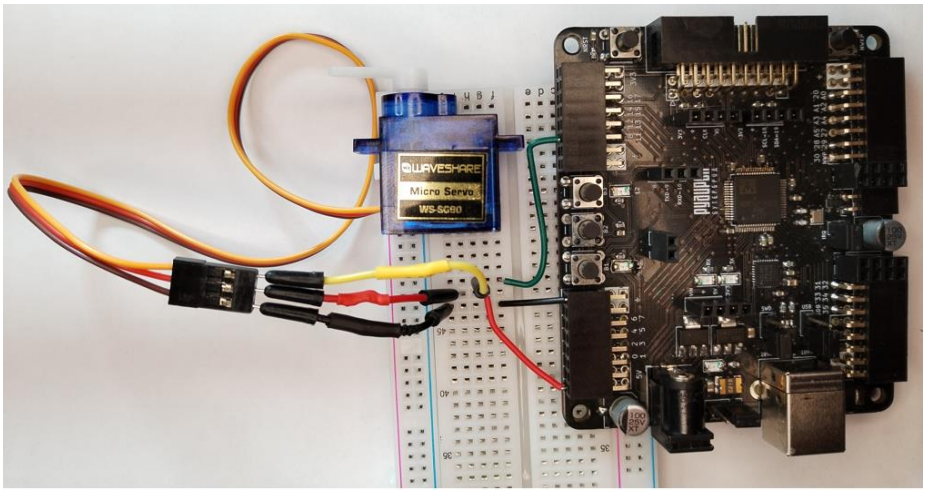
Параметры сервопривода SG90S

№	Наименование	Значение	Назначение
1	Тип сервопривода	аналоговый	Управляется широтно-импульсной модуляцией
2	Напряжение питания	4.8 ~ 5 В	
3	Усилие на валу: 4.8В	1.4 кг/см	
4	Скорость: 4.8В	0.12 сек/60°	
5	Размеры (мм)	22.8×12.2×28.5	
6	Разъем	JR / Spektrum / Dupont	

Подключение к сервоприводу

Вход модуля	Назначение	Значения
Коричневый провод	Земля Ground	подключается к порту GND на плате
Красный провод	Питание +5 V	подключается к порту 5V на плате
Желтый провод	Сигнал управления	подключается к цифровому порту работающий в режиме ШИМ (PWM)

Схема подключения:



В соответствии с правилами подключения сервоприводы Коричневый провод соединен с разъемом земля контроллера. Красный провод соединен с разъемом +5 В контроллера. Желтый провод соединяем с выводом 8 так как он может генерировать ШИМ сигнал в соответствии с таблицей GPIO. Питание поступает от разъема USB пока проводится отладка программы.

```
#include "Arduino.h"
#include "Servo.h"          // подключаем дополнительную программу по управлению сервоприводом

#define SERVO_PIN 8 // даем имя порту 8 по управлению сервоприводом

Servo servo; // создаем переменную типа сервопривод

int delay_ms; // создаем переменную для хранения значения паузы в миллисекундах

void setup(){

pinMode(SERVO_PIN, OUTPUT);

servo.attach(SERVO_PIN, 450,2500); // устанавливаем параметры сервы порт, и значения мин и макс заполнения длительности импульсов ШИМ

}

 void loop()
{
```

```
delay_ms = 500;
      servo.write(0); // устанавливаем 0 градусов на сервоприводе
      delay(delay_ms);
      servo.write(90); // устанавливаем 90 градусов на сервоприводе
      delay(delay_ms);
      servo.write(180); // устанавливаем 180 градусов на сервоприводе
      delay(delay_ms);
       servo.write(127); // устанавливаем 270 градусов на сервоприводе
      delay(delay_ms);
       servo.write(45); // устанавливаем 45 градусов на сервоприводе
      delay(delay_ms);

}
```

В данном примере мы каждые 500 миллисекунд поворачиваем вал на определенный градус. От 0 до 180 градусов и обратно. Затем естественно опять с нулевого значения по циклу.
Порядок наших действие:

1. Подключаем программу управления сервоприводом *#include "Servo.h"*. Эта программа будет встроена в исполняемый файл для микроконтроллера на этапе сборки проекта на основании файла CMake.
2. Определяем имя для вывода 8 контроллера. #define SERVO_PIN 8
3. Установили режим на вывод pinMode(SERVO_PIN, OUTPUT).
4. Вызываем внешнюю программу *Servo*, которой сообщаем на каком выходе контроллера подключен сервопривод. Так же указываем минимальную длину импульсов шим и максимальную. Для каждого типа сервопривода параметр подбирается индивидуально. servo.attach(SERVO_PIN, 450, 2500).
5. Управляем углом поворота сервопривода командой servo.write(90), указав команде на какой угол необходимо повернуть вал.

Как видите не сложно так как непосредственно программу правильно настраивающая микроконтроллер для работы с сервоприводом написали за нас. Мы этой программой пользуемся внутри нашей. Нам надо только программу подключить строкой (называется – директива) #include Servo.h. А когда мы будем запускать сборку проекта, то программа будет включена внутрь исполняемого файла для нашего микроконтроллера.