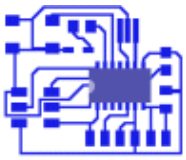


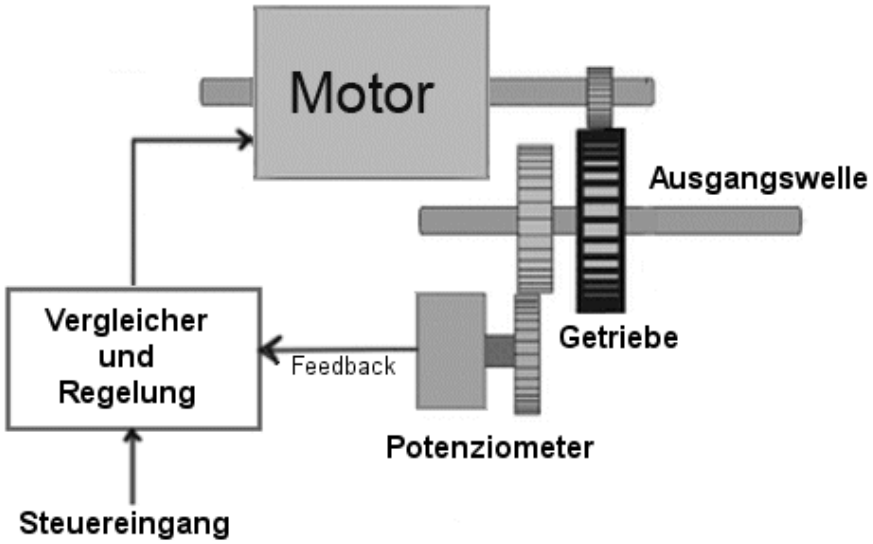
Arduino

von Prof. Jürgen Plate

Servosteuerung

Allgemeines

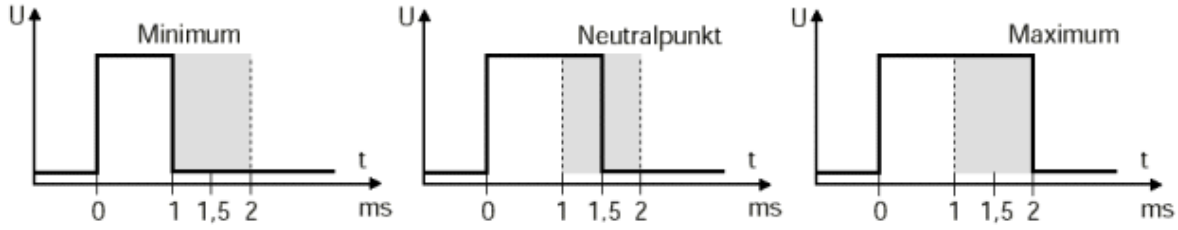
Servos aus dem Modellbaubereich ("Rudermaschinen") sind trotz der Betriebsspannung von 5 Volt besonders kräftige Antriebe. Ihr hohes Drehmoment wird durch ein Untersetzungsgetriebe erreicht. Ein Servoantrieb besteht aus einem Motor, einem auf der Achse angebrachten Positions- oder Winkelsensor (im einfachsten Fall ist das ein Potentiometer) und einer Regelelektronik mit Sollwert-Eingang. Diese vergleicht den eingegebenen Sollwert mit dem Istwert des Sensors. Stimmen beide nicht überein, so lässt die Regelung den Motor zu der Position laufen, bei der Istwert und Sollwert gleich sind. Diese elektronische Motorsteuerung ermöglicht nicht nur ein sehr feinfühliges und genaues Stellen des Antriebs in eine bestimmte Position, sie sorgt auch für ein kräftiges Gegenmoment gegen Rückstellversuche der Last und hält damit die gewünschte Position.



Je nach verwendetem Servo sind Stellzeiten bis herab auf 0,08 s über einen Stellweg von 60 Grad erreichbar. Die modernste Version, der Digitalservo, ist statt der bei analog arbeitenden Servos passiv arbeitenden Servosteuerung mit einem Mikroprozessor bestückt. Durch eine hohe Taktfrequenz kann der Antriebsmotor besonders schnell und in allen Lagen mit vollem Drehmoment arbeiten, was extrem kurze Reaktionszeiten bei gleichmäßiger Kraftentfaltung erlaubt. Die Analog-Servos dagegen verlieren oft gegen Ende des Stellwegs an Drehmoment und auch Drehgeschwindigkeit. Ein weiterer Vorteil des Digitalservos ist das aktive Gegensteuern durch den Prozessor bei Rückstellversuchen der Last. Nachteilig bei beiden Typen ist der relativ kleine Stellweg und das laute Arbeitsgeräusch. Die Stellkraft (etwa 10 Newton) ist, gemessen an der Größe, beachtlich. Sie sollten aber berücksichtigen, dass die Getriebe bei vielen Modellen aus Plastikzahnrädern bestehen, die bei Überlast sehr schnell verschleißen (gegebenenfalls auf die etwas teureren Modelle mit Metallgetriebe ausweichen). Ein Servo besitzt üblicherweise drei Leitungen: Masse (meistens braun oder schwarz), +5 V (meist rot) und die Steuerleitung (weiß, grün, gelb, orange ...). Da ein Servo verhältnismäßig viel Strom zieht, sollte er nicht über den Arduino mit Strom versorgt werden. Sonst besteht die Gefahr, dass der Arduino überlastet und zerstört wird. Außerdem kann es sein, dass der Ardino von Störsignalen des Motors aus dem Takt gebracht wird. Insofern lohnt sich eine separate Stromversorgung. **Bei getrennter Stromversorgung dürfen Sie nicht vergessen, die Masseanschlüsse (GND, Minuspol) von Arduino und Servo-Stromversorgung zu verbinden, um ein gemeinames Bezugspotenzial zu haben.**

Die Ansteuerung vom Arduino aus ist bei allen Servo-Typen gleich; die Sollwert-Vorgabe erfolgt über einen

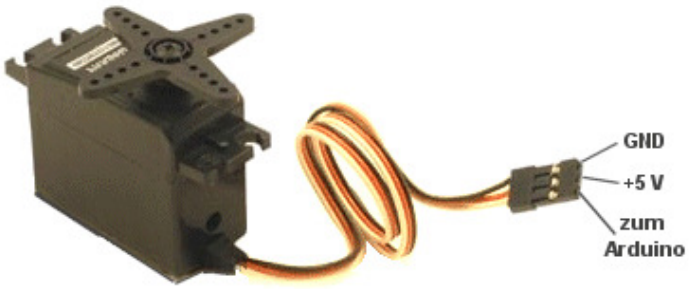
längenmodulierten Impuls (Pulsweitenmodulation, PWM). Der Servo wird alle 20 ms mit kurzen Impulsen von ein bis zwei Millisekunden Dauer angesteuert und damit die Position seines Antriebs bestimmt. Die Impulse müssen im 20-ms-Abstand wiederholt werden, damit ein Servo seine Position beibehält. Praktisch alle Servos erwarten übrigens einen positiven Impuls.



Alle genannten Eigenschaften prädestinieren den Modellbau-Servo als Antrieb für andere Verwendungen. Sie lassen sich für beliebige Betätigungsfunktionen verwenden, z. B. das Betätigen eines Riegels, das Aktivieren einer Fütterungsautomatik für Fische, das Öffnen und Schließen von Lüftungsklappen, die computerisierte Einzelbild-Steuerung für Langzeitaufnahmen einer 16-mm-Kamera oder das Schwenken einer Überwachungskamera. Modellbau-Servos werden heutzutage auch gerne bei experimentellen Robotern eingesetzt. Häufig findet man in einem Roboter eine größere Anzahl davon. Ein "Käfer" mit sechs Beinen benötigt in der Regel mindestens drei Servos pro Bein, und ein einfacher Roboterarm hat mitunter sechs bis sieben.

Ansteuerung per Software

Nachdem Servo und Arduino mit Strom versorgt sind kann die Steuerleitung des Servos mit einem der Arduino-Pins verbunden werden (Sie haben auch an die gemeinsame Masseverbindung gedacht?). Die Servo-Bibliothek unterstützt bis zu 12 Servos auf den meisten Arduino-Boards und 48 auf dem Arduino Mega. Beachten Sie auch die Einschränkung: Solange ein Pin für einen Servo verwendet wird, können die Pins 9 und 10 keine pulsweitenmodulierten Signale (PWM-Signale) ausgeben (die Servo-Library verwendet den Timer 1). Auf dem Mega können bis zu 12 Servos verwendet werden, ohne die PWM-Funktion zu beeinträchtigen. Die Verwendung von 12 bis 23 Servos deaktiviert PWM auf den Pins 11 und 12.



Zur Programmierung von Servos gibt es eine fix-und-fertige Bibliothek. Im Hauptmenü der IDE finden Sie unter "Sketch" den Befehl "Import Library". Hier wählt man einfach die Servo-Bibliothek aus und im Sketch erscheint die Include-Zeile, in desem Fall:

```
#include <Servo.h>
```

Im folgenden Beispiel wird eine `write()`-Methode genutzt, um den Servo von 1 bis 179 Grad und wieder zurück zu bewegen (das Beispiel findet sich auch in der IDE unter "File" → "Examples" → "Servo"). Beachten Sie auch, dass bei langsameren Bewegung des Servos Wartezeiten eingeschoben werden müssen, weil sonst eventuell die am Servo hängende Mechanik nicht "mitkommt":

```
// Headerfile für die Servosteuerung einbinden
#include <Servo.h>

// Erzeugen eines Servo-Objekts
```

```
Servo myservo;

int pos = 0; // Speichern der Position

void setup()
  { myservo.attach(9); } // Servo-Datenleitung an Pin 9

void loop()
  {
  // mit mittlerer Geschwindigkeit nach rechts drehen
  for(pos = 1; pos < 180; pos++)
    {
    myservo.write(pos);
    delay(15);
    }
  // mit mittlerer Geschwindigkeit nach links drehen
  for(pos = 179; pos > 0; pos--)
    {
    myservo.write(pos);
    delay(15);
    }
  }
```

Die `write()`-Methode verträgt Steuerwerte zwischen 0 und 180, was bei einen 180-Grad-Servo exakt dem Winkel der Servoachse entspricht. Im folgenden Beispiel wird eine per A/D-Wandler gelesene Eingangsspannung in eine Servoauslenkung transponiert.

```
#include <Servo.h>

Servo myservo;   // neues Servo-Objekt erzeugen
int potpin = 0;  // Analogpin, an dem das Potenziometer haengt
int val;         // Hilfsvariable pin

void setup()
  {
  myservo.attach(9);  // ordnet Pin 9 der Servo-Steuerleitung zu
  }

void loop()
  {
  val = analogRead(potpin);          // Poti einlesen (0 - 1023)
  val = map(val, 0, 1023, 0, 179);  // Skalierung auf zulaessige
                                     // Servo-Werte (max. 180)
  myservo.write(val);                // Servo positionieren
  delay(15);                         // etwas warten
  }
```

Die Bibliothek enthält neben den oben verwendeten Methoden `attach()` und `write()` noch weitere Methoden. Im FOlgenden sind alle aufgelistet:

- **attach(Pin)**
- **attach(Pin, Minimum, Maximum)**
  Mit diesem Aufruf wird ein Pin des Arduino mit dem Servo-Objekt verknüpft. Die erste Version ist der minimale Code. Die zweite Variante enthält zwei sehr wichtige, optionale Parameter, welche die minimale und maximale Pulsbreite festlegen. Auf diese Weise können auch "exotische" Servos bequem angesteuert werden. In der Servobibliothek sind die Standard-Minimum- und Maximum-Einstellungen als Konstante eingetragen:

  ```
  #define MIN_PULSE_WIDTH       544     // the shortest pulse sent to a servo
  #define MAX_PULSE_WIDTH      2400     // the longest pulse sent to a servo
  #define DEFAULT_PULSE_WIDTH  1500     // default pulse width when servo is attached
  ```

  Wenn ein 180-Grad-Servo auf Impulse zwischen 1000 µs und 2000 µs reagiert, interpretiert es 1000 µs als 0° und 2000 µs als 180°. Mit dem Standardimpulsbreitenbereich von 544 µs bis 2400 µs, sendet der Arduino ein Signal von ca. 1000 µs für einen Winkel von 44°. Das Servo könnte nur einem Winkel von 90° überstreichen, statt der vollen 180°. Dies und andere potenzielle Probleme können vermieden werden, wenn Mikrosekunden anstelle von Winkeln verwendet werden (siehe unten: tt>servo.writeMicroseconds(1500);) oder wenn die optionalen Minima und Maxima im Pin-Setup für jeden Servo definiert werden.

Beispiel: `servo.attach(9);`

- **write(Wert)**
  Dies dreht die Servoachse auf einen bestimmten Zielwert. Der angegebene Wert entspricht in der Regel dem Winkel in Grad, den der Servo anfahren soll. Da die Bewegung eine gewisse Zeit in Anspruch nimmt, sollte man anschließend eine entsprechende Pause einlegen(z. B. mit `delay()`).
  Beispiel: `servo.write(60);`
- **writeMicroseconds(Wert)**
  Sendet den angegebenen Wert für die Impulsbreite an den Servo. Bei Standard-Servos bedeutet ein Wert von 1000 das Minimum, 2000 das Maximum. Die Auflösung ist hier wesentlich höher als bei `write()` und es lassen sich auch 270-Grad-Servos gut ansteuern.
  Beispiel: `servo.writeMicroseconds(1500);`
- **read()**
  Gibt die Stellung des Servos zurück. Dabei wird jedoch nur der zuletzt von `write()` verwendete Wert zurückgegeben.
  Beispiel: `stellung = servo.read();`
- **attached()**
  Ergibt entweder "true" oder "false", je nachdem, ob das Servo-Objekt mit einem Pin des Arduino verknüpft ist oder nicht.
  Beispiel: `if (servo.attached()) { ... }`
- **detach()**
  Löst die Verbindung des Servos-Objekts zu einem Pin. Wenn die Verbindungen aller Pins aufgehoben wurden, können die Pins 9 und 10 wieder für die Erzeugung pulsweitenmodulierter Signale verwendet werden.
  Beispiel: `servo.detach();`

Literatur/Links

- Arduino-Referenz: Servo
- Anwendungen mit Servo Motoren

Letzte Aktualisierung: 05/01/2017 18:06:33