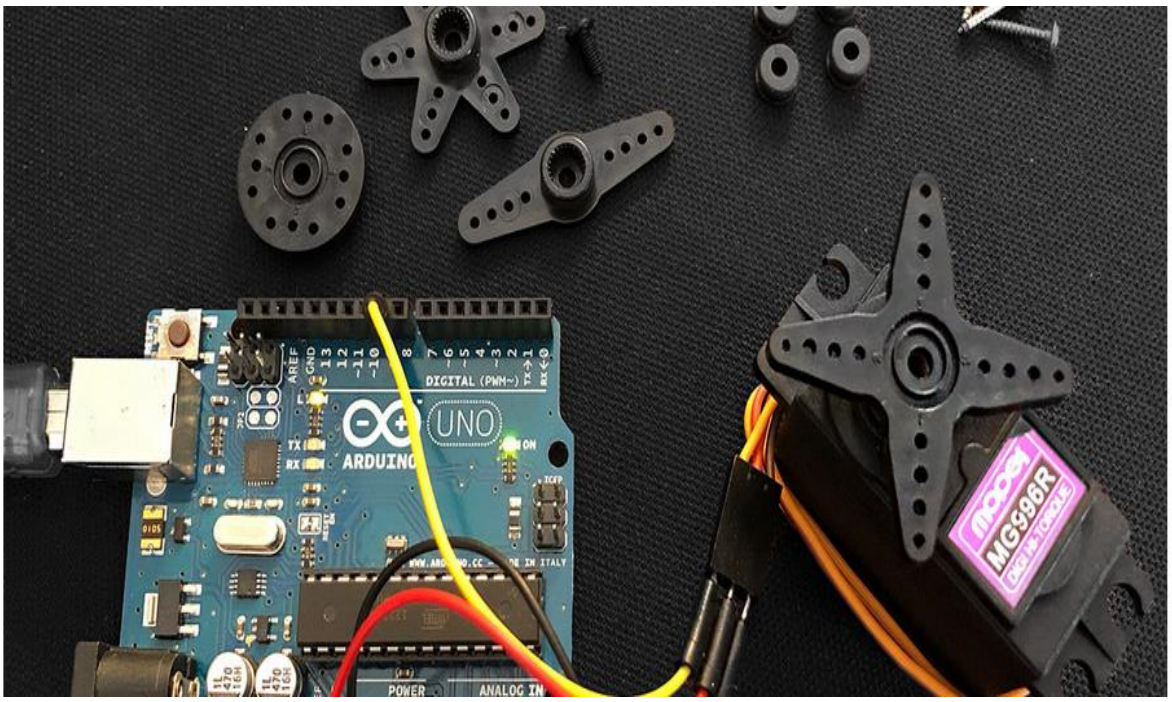
Arduino und Servo Steuerung

https://starthardware.org/servo/

Grundlagen

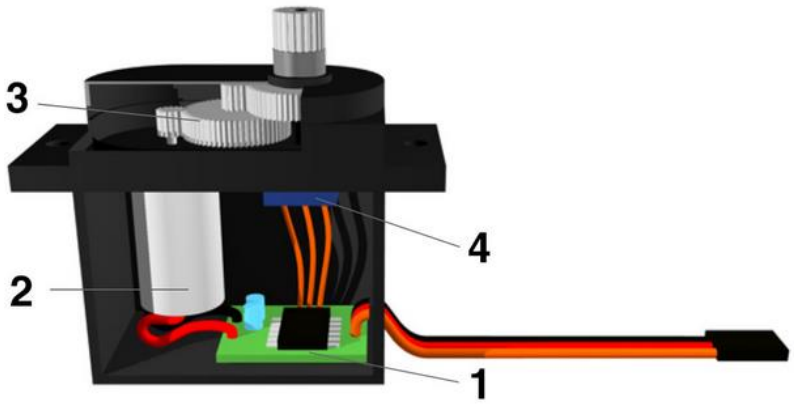
Einen Servo mit Arduino zu steuern ist relativ einfach, wenn man ein paar Dinge beachtet.

- Software-Bibliothek
- Spannungsversorgung
- Mit/ohne Rücksignal



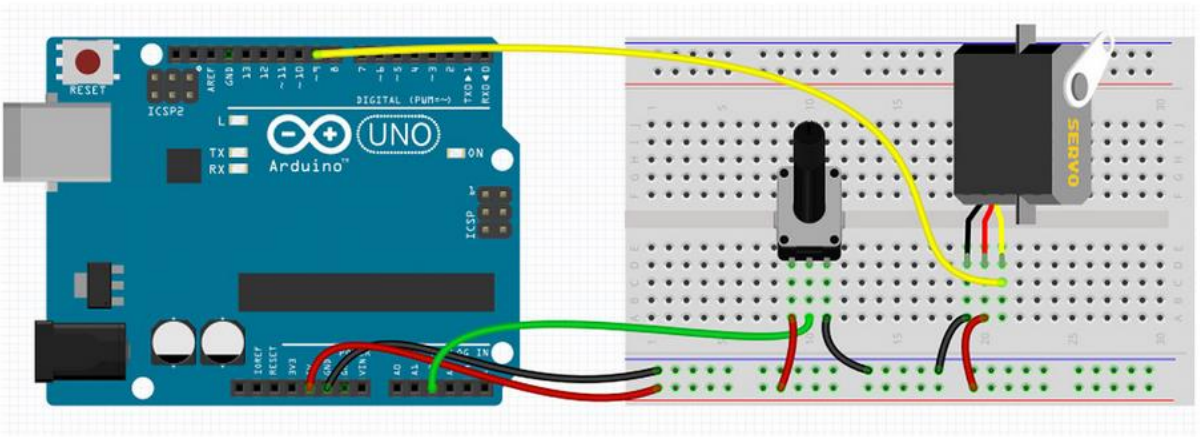
Servo: Aufbau und Anschluss

Ein Servo besteht aus einer Motorsteuerung (1), einem Elektromotor (2), einem Getriebe (3) und einem Potentiometer zur Positionsbestimmung (4). Alle Komponenten sind in einem robusten Gehäuse untergebracht.



Angeschlossen wird er über ein dreipoliges Kabel. Dabei handelt es sich um zwei Versorgungsleitungen (Plus und GND) und um eine Datenleitung.

Basis-Schaltung ohne Rücksignal



Software Beispiel Programm Code

Schaltung und Beispiel-Programm-Code

Im Beispiel ist ein Standard-Servomotor mit dem GND, 5V+ und dem digitalen Pin 9 verbunden. Des Weiteren ist ein Potentiometer am analogen Pin 2 angeschlossen.

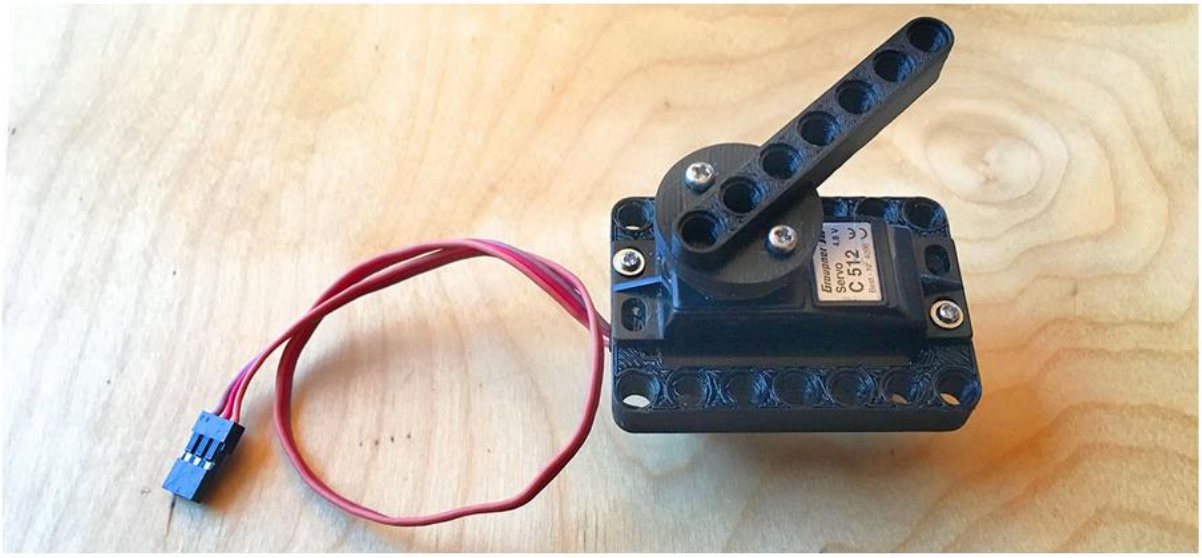
Bei dem Programm handelt es sich um das Knob-Beispiel aus der Arduino Software (Datei > Beispiele > Servo > Knob).

```
#include <Servo.h>

Servo myservo; // erstellt ein Servo-Objekt um einen Servomotor zu steuern

int potpin = 0; // Analog Pin, an dem das Potentiometer angeschlossen ist
int val; // Variable um den Vert des Analogen Pin zwischen zu speichern

void setup() {
myservo.attach(9); // verknüpft den Servomotor an Pin 9 mit dem Servo-Objekt
}

void loop() {
val = analogRead(potpin); // liest das Potentiometer aus (Wert zwischen 0 und 1023)
val = map(val, 0, 1023, 0, 180); // rechnet den Wert in den Wertebereich des Servomotors (zwischen
myservo.write(val); // überträgt die Zielposition an den Servomotors
delay(15); // lässt dem Servomotor Zeit, die Zielposition zu erreichen
}
```

In diesem Beispiel wird der Servomotor auf eine Position gesteuert, die mittels des Potentiometers festgelegt wird. Natürlich ist das nur exemplarisch, um die Funktionsweise zu erklären. Wichtig ist, dass der Befehl myservo.write(val); den Servomotor in eine bestimmte Position zwischen 0 und 180 Grad stellt.

Um diese Kommunikation einzurichten gibt es eine Library von Adafruit, die sich im Handumdrehen einbinden und verwenden lässt.

Alternativ zum Adafruit Robot Servo Shield gibt es noch eine günstigere Variante von SunFounder.



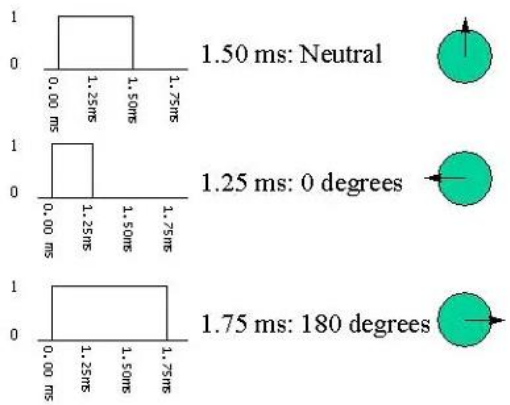
Tipp: Leider passen normale Servos nicht an LEGO Technik. Mit diesem kleinen 3D Modell geht das dann aber doch. Ausgedruckt auf dem Wanhao Duplicator i3.

Viele Servomotoren mit Arduino steuern

Ein Servomotor benötigt für die Bewegungen relativ viel Strom. Will man mehr als einen Servomotor nutzen, empfiehlt die schnell die Nutzung eines Servo-Shields wie dem Adafruit Robot Shield. Dabei handelt es sich um eine, auf das Arduino aufsteckbare, Erweiterung des Arduinos. Mit dem Adafruit Robot Servo lassen sich bis zu 16 Servomotoren gleichzeitig steuern. Ein externes Netzteil ist dafür allerdings erforderlich. Angenehmer Nebeneffekt ist, dass man nicht pro Servomotor einen wertvollen Kanal des Arduinos belegt. Das Shield wird per I2C angesteuert. Dabei handelt es sich um eine digitale Datenverbindung zwischen dem Arduino und dem Servo Shield. Es belegt lediglich zwei Pins des Boards.

Spezialform Servo-Winde

Dabei handelt es sich um einen kräftigeren Servomotor, der sich nicht nur auf 180°, sondern je nach Typ auf 360° oder 720° drehen kann. Winden werden im Modellbau z.B. in Segelschiffen verbaut. Hier kann man sich das Produkt auf Amazon ansehen.



Steuerung ohne Library

Expertenwissen: Steuerung ohne Library

Zum Steuern des Servomotors verwendet man ein sich alle 20 Millisekunden wiederholendes Signal. Es besteht aus einem HIGH-Impuls, der zwischen 1 und 2 Millisekunden lang ist, und einem LOW-Impuls. Die Dauer des HIGH-Impulses bestimmt den Zielwinkel (normaler Weise von 0 bis 180°). In der Arduino-Software kann hierfür der Befehl delayMicroseconds(x); verwenden:

```
digitalWrite(myServo,HIGH);
delayMicroseconds(1500);
digitalWrite(myServo,LOW);
delayMicroseconds(18500);
```

Da der Servo einige Zeit benötigt, um sich auf den gewünschten Zielwinkel einzustellen, muss das Signal mindestens so lange wiederholt werden, bis der Servo die Position erreicht hat.

Servomotoren überprüfen ihren eigenen Stellwinkel mit einem eingebauten Potentiometer. Es kommt hierbei vor, dass eine gewünschte Position nicht exakt erreicht werden kann. Der Servo korrigiert sich dann kontinuierlich selbst, was zu einem »Zittern« des Servos führt. Um das zu verhindern, sollte der Impuls nach erreichen des Zielwinkels auf LOW geschalten werden.

Alternativ zum selbst programmierten Signal, kann man die Servo-Library benutzen. Sie kommt mit der Arduino-Software. Im Beispiel Sweep (File>Examples>Servo>Sweep), wird gezeigt, wie man einen Servo mit der Servo-Library verwendet.

```
// Sweep
// by BARRAGAN

#include Servo myservo;          // erzeugt ein Servomotor-Objekt
// maximal können acht Servomotor-Objekte erzeugt werden

int pos = 0;             // Variable, die die Servoposition (Winkel) speichert

void setup(){
  myservo.attach(9);  // an welchem Pin ist der Servomotor angeschlossen
}

void loop(){
  for(pos = 0; pos < 180; pos += 1) {  // von 0 bis 180 Grad, in Schritten von einem Grad
    myservo.write(pos);                   // sagt dem Servomotor, in welche Position sich drehen s
    delay(15);                            // wartet 15 Millisekunden
  }
  for(pos = 180; pos>=1; pos-=1) {     // und das gleiche zurück
    myservo.write(pos);
    delay(15);
  }
}
```

Steuerung mit Library

https://elektro.turanis.de/html/prj036/index.html

Mit der Library Servo, die schon in der Arduino-IDE vorhanden ist, aber auch über Github bei Arduino/libraries/Servo geladen werden kann, ist eine Ansteuerung des Servos sehr einfach. Hinweis: Die Zahl in servoMotor.write() ist immer der Winkel, der gestellt werden soll. *Wichtig: Die beiden letzten Parameter beim Aufruf von servoMotor.attach() entpsrechenden dem minimalen und maximalen Verzögerungszeit des PWM-Signals und sollten für den Motor entsprechend richtig gewählt werden, damit die Stellwinkel auch wirklich stimmen!*

```
#include <Servo.h>

#define SERVO_SIGNAL_PIN 9

Servo servoMotor;

void setup()
{
    servoMotor.attach(SERVO_SIGNAL_PIN, 900, 1500);
}

void loop()
{
    servoMotor.write(45);
    delay(500);

    servoMotor.write(135);
    delay(500);


    for(int i = 0; i<180; i+=5){
        servoMotor.write(i);
        delay(150);
    }

    delay(1000);

    for(int i = 180; i>0; i-=5){
        servoMotor.write(i);
        delay(150);
    }
}
```

Servormotor direkt ansteuern

Es ist auch möglich ohne Verwendung einer Library einen Servomotor mit einem Arduino anzusteuern. Dazu muss zunächst ein HIGH über die Signalleitung geschickt werden. Anschließend ist eine Pause zwischen 1000µs bis 2000µs einzuhalten, je nachdem welcher Winkel gestellt werden soll. Dann folgt ein LOW, wobei darauf noch eine kurze Pause von ca 20ms folgende sollte, damit der Motor genug Zeit hat, sich vollständig einzustellen. In meinen Versuch ist aber herausgekommen, das es nicht der absolute Winkel ist, der gestellt wird, sondern der relative, d.h. man verstellt durch setServoAngle() immer nur UM einen bestimmten Winkel und nicht AUF einen Winkel. Außerdem ist der Servormotor nach dem stellen manuell frei beweglich und nicht elektrisch gesperrt.

```
#define SERVO_SIGNAL_PIN 8

void setup()
{
    pinMode(SERVO_SIGNAL_PIN, OUTPUT);

    for(int i=0; i<180; i+=5){
        setServoAngle(i);
        delay(100);
    }
}

void loop()
{
}

void setServoAngle(byte angle)
{
    // calculate angle to delayTime
    int delayTime = map(angle, 0, 180, 1000, 2000);

    digitalWrite(SERVO_SIGNAL_PIN, HIGH);
    delayMicroseconds(delayTime);
    digitalWrite(SERVO_SIGNAL_PIN, LOW);
    delay(20); // short pause at the end of the regulating
}
```

360°-Servomotor "DS04-NFC"

Einen speziellen Servo ist das folgende Modell, welches sich um 360° kontinuierlich drehen kann. Laut Hersteller bestitzt es folgende Eigenschaften:

Gewicht:	38g
Abmessungen:	40,8 × 20,0 × 39,5 mm
Geschwindigkeit:	0,22s / 60° (bei 4,8V)
Betriebsspannung:	4,8V - 6V
Betriebsstrom:	<1000mA
Drehmoment:	5,5kg/cm (bei 4,8V)

Sketch

```
#include <Servo.h>

#define SERVO_SIGNAL_PIN  9

// define the duration of the signal in milliseconds
#define STOP 1500
#define ROTATE_CLOCKWISE   1000
#define ROTATE_COUNTERCLOCKWISE  2000

Servo servoMotor;

void setup()
{
    servoMotor.attach(SERVO_SIGNAL_PIN);

    servoMotor.writeMicroseconds(STOP);
    delay(2000);

    servoMotor.writeMicroseconds(ROTATE_CLOCKWISE);
    delay(2000);

    servoMotor.writeMicroseconds(ROTATE_COUNTERCLOCKWISE);
    delay(2000);

    servoMotor.writeMicroseconds(STOP);
}

void loop()
{
}
```