ROBOTER
Bausatz-Shop

SG90 9g Micro Servomotor

Anleitung mit Schalt- und Programmierbeispielen

Schwenken

Schwenkt die Welle eines RC-Servomotors um 180 Grad hin und her.

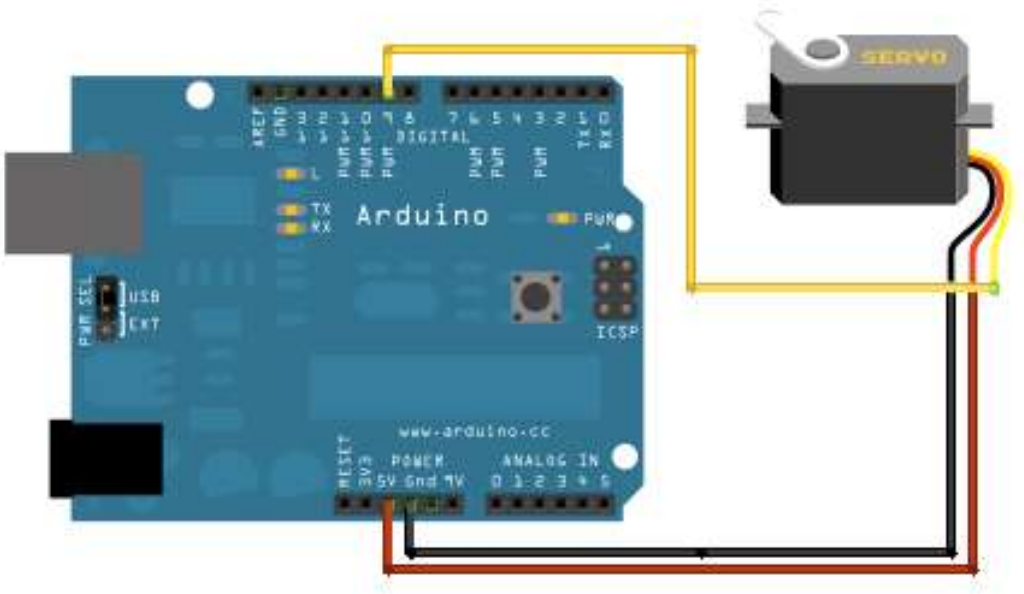
In diesem Beispiel wird die Arduino-Servobibliothek verwendet.

Erforderliche Hardware

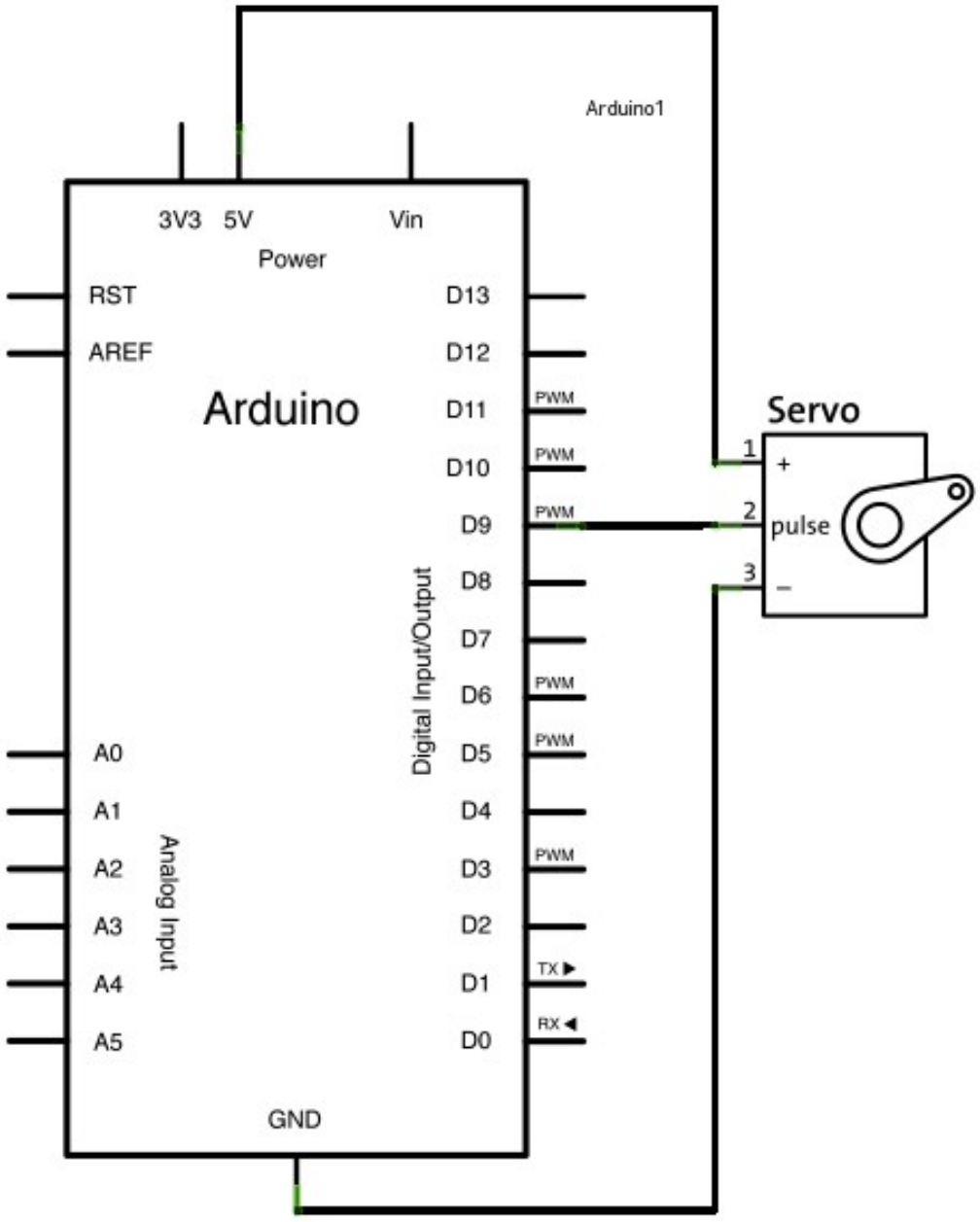
- Arduino- oder Genuino-Vorstand
- Servomotor
- Anschlussdrähte

Schaltung

Servomotoren haben drei Drähte: Strom, Masse und Signal. Der Stromversorgungsdraht ist normalerweise rot und sollte an den 5V-Pin auf der Arduino- oder Genuino-Platine angeschlossen werden. Der Massedraht ist typischerweise schwarz oder braun und sollte mit einem Erdungspin auf der Platine verbunden werden. Der Signal-Pin ist typischerweise gelb, orange oder weiß und sollte mit Pin 9 auf der Platine verbunden werden.



Bilder wurden mit Fritzing entwickelt. Weitere Schaltungsbeispiele finden Sie auf der Fritzing-Projektseite http://fritzing.org/projects/


Arduino1
3V3
5V
Vin
Power
RST
AREF
Arduino
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
PWM
TX
RX
Digital Input/Output
A0
A1
A2
A3
A4
A5
Analog Input
GND
Servo
1
2
3
+
pulse
–

Code

```
/* Sweep
 by BARRAGAN <http://barraganstudio.com>
 This example code is in the public domain.

 modified 8 Nov 2013
 by Scott Fitzgerald
 http://www.arduino.cc/en/Tutorial/Sweep
*/

#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

attach()

Erklärung

Befestigen Sie den Servo an einem Pin. Beachten Sie, dass in Arduino 0016 und früher die Servo-Bibliothek nur Servos auf nur zwei Pins unterstützt: 9 und 10.

Syntax

servo.attach(pin)
servo.attach(pin, min, max)

Parameter

servo: ein variable Servotyp

pin: die Nummer des Pins, mit dem der Servo verbunden wird.

min (optional): die Impulsbreite in Mikrosekunden, die dem minimalen (0-Grad-) Winkel auf dem Servo entspricht (Standardwert 544)

max (optional): die Impulsbreite in Mikrosekunden, die dem maximalen Winkel (180 Grad) auf dem Servo entspricht (Standardeinstellung: 2400)

Beispiel

```
#include <Servo.h>

Servo myservo;

void setup()
{
  myservo.attach(9);
}

void loop() {}
```

write()

Erklärung

Schreibt einen Wert in den Servo und steuert die Welle entsprechend. Bei einem Standardservo wird dadurch der Winkel der Welle (in Grad) eingestellt und die Welle in diese Ausrichtung bewegt. Bei einem Servo mit kontinuierlicher Drehung wird dadurch die Geschwindigkeit des Servos eingestellt (wobei 0 für volle Geschwindigkeit in einer Richtung, 180 für volle Geschwindigkeit in der anderen Richtung und ein Wert nahe 90 für keine Bewegung steht).

Syntax

servo.write(angle)

Parameter

servo: ein variable Servotyp

angle: ein Wert zw. 0 und 180, um den Servo einzustellen

Beispiel

```
#include <Servo.h>

Servo myservo;

void setup()
{
  myservo.attach(9);
  myservo.write(90);  // set servo to mid-point
}

void loop() {}
```

map()

[Math]

Erklärung

Neuzuordnung einer Nummer von einem Bereich zu einem anderen. Das heißt, ein Wert von fromLow würde auf toLow abgebildet werden, ein Wert von fromHigh auf toHigh, Werte dazwischen auf Werte dazwischen, usw.

Beschränkt die Werte nicht auf Werte innerhalb des Bereichs, da Werte außerhalb des Bereichs manchmal beabsichtigt und nützlich sind. Die Funktion constrain() kann entweder vor oder nach dieser Funktion verwendet werden, wenn eine Begrenzung der Bereiche erwünscht ist.

Beachten Sie, dass die "unteren Grenzen" beider Bereiche größer oder kleiner als die "oberen Grenzen" sein können, so dass die map()-Funktion verwendet werden kann, um einen Zahlenbereich umzukehren, z.B.

```
y = map(x, 1, 50, 50, 1);
```

Die Funktion kann auch mit negativen Zahlen gut umgehen, so dass dieses Beispiel`y = map(x, 1, 50, 50, -100);`

auch gültig ist und gut funktioniert.

Die map()-Funktion verwendet die Ganzzahlmathematik, erzeugt also keine Brüche, wenn die Mathematik möglicherweise anzeigt, dass sie dies tun sollte. Restbruchteile werden abgeschnitten und nicht gerundet oder gemittelt.Syntax

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Parameter

`value`: die Nummer für die Zuordnung.

`fromLow`: die untere Grenze des aktuellen Wertebereichs.
`fromHigh`: die obere Grenze des aktuellen Wertebereichs.
`toLow`: die untere Grenze des Zielbereichs des Wertes.
`toHigh`: die obere Grenze des Zielbereichs des Wertes.

Returns

Der zugeordnete Bereich.

Beispiel Code

```
/* Map an analog value to 8 bits (0 to 255) */
void setup() {}

void loop() {
  int val = analogRead(0);
```

```
  val = map(val, 0, 1023, 0, 255);
  analogWrite(9, val);
}
```

Anhang

Für die mathematisch Interessierten ist hier die ganze Funktion

```
long map(long x, long in_min, long in_max, long out_min, long out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Notizen und Warnungen

Wie bereits erwähnt, verwendet die map()-Funktion Ganzzahlmathematik. Dadurch werden möglicherweise Brüche unterdrückt. Zum Beispiel werden Brüche wie 3/2, 4/3, 5/4 alle als 1 von der map()-Funktion zurückgegeben, trotz ihrer unterschiedlichen tatsächlichen Werte. Wenn Ihr Projekt also präzise Berechnungen erfordert (z.B. Spannung auf 3 Dezimalstellen genau), sollten Sie map() vermeiden und die Berechnungen selbst manuell in Ihrem Code implementieren.

Knopf

Steuern Sie die Position eines RC (Hobby)-Servomotors mit Ihrem Arduino und einem Potentiometer.

In diesem Beispiel wird die Arduino-Servobibliothek verwendet.

Erforderliche Hardware

- Arduino- oder Genuino-Vorstand

- Servomotor

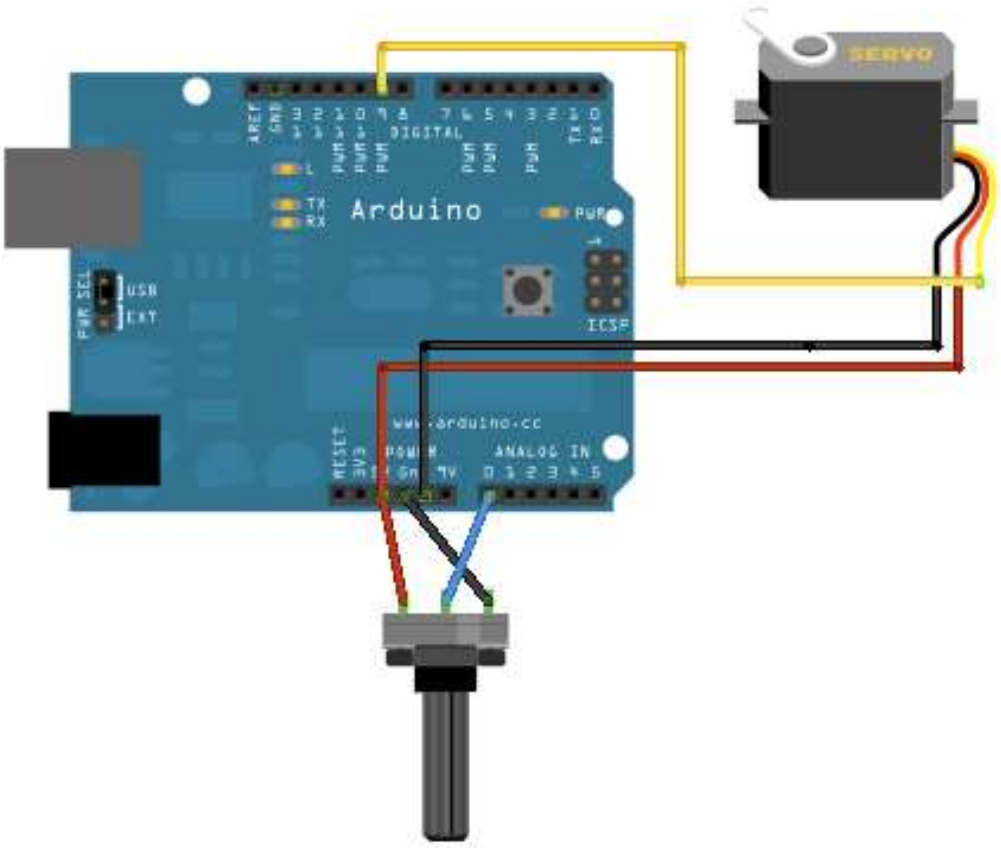
- 10k Ohm-Potentiometer

- Anschlussdrähte

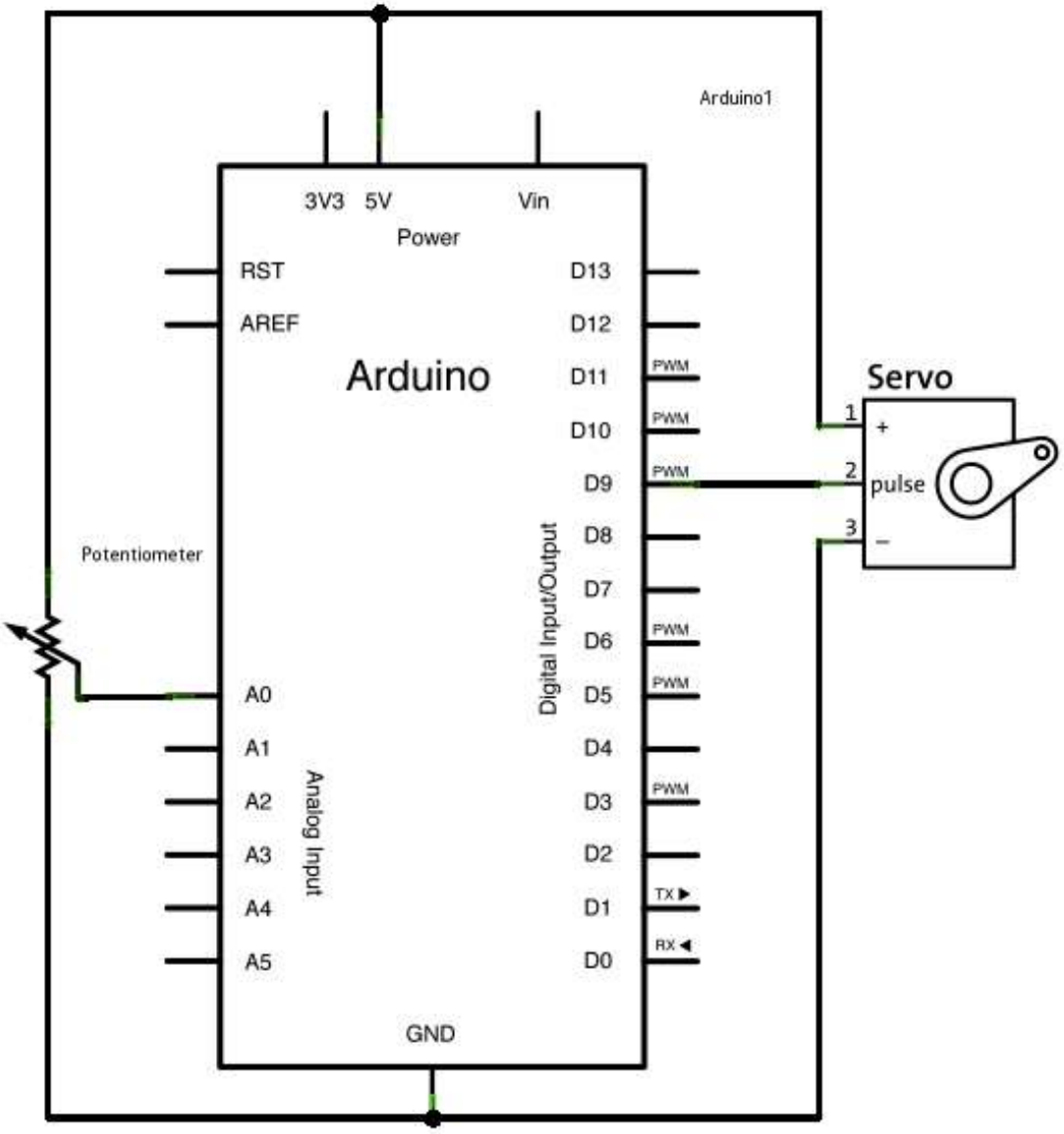
Schaltung

Servomotoren haben drei Drähte: Strom, Masse und Signal. Der Stromversorgungsdraht ist normalerweise rot und sollte an den 5V-Pin auf der Arduino- oder Genuino-Platine angeschlossen werden. Der Massedraht ist typischerweise schwarz oder braun und sollte mit einem Erdungspin auf der Platine verbunden werden. Der Signal-Pin ist typischerweise gelb oder orange und sollte mit Pin 9 auf der Platine verbunden werden.

Das Potentiometer sollte so verdrahtet werden, dass seine beiden äußeren Pins mit Strom (+5V) und Masse verbunden sind und sein mittlerer Pin mit Analogeingang 0 auf der Platine verbunden ist.


SERVO
Arduino
PWR SEL
USB
EXT
PWR
ICSP
www.arduino.cc
ANALOG IN

Anschlussschema



Code

```
/*
 Controlling a servo position using a potentiometer (variable resistor)
 by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

 modified on 8 Nov 2013
 by Scott Fitzgerald
 http://www.arduino.cc/en/Tutorial/Knob
*/

#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin);           // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```