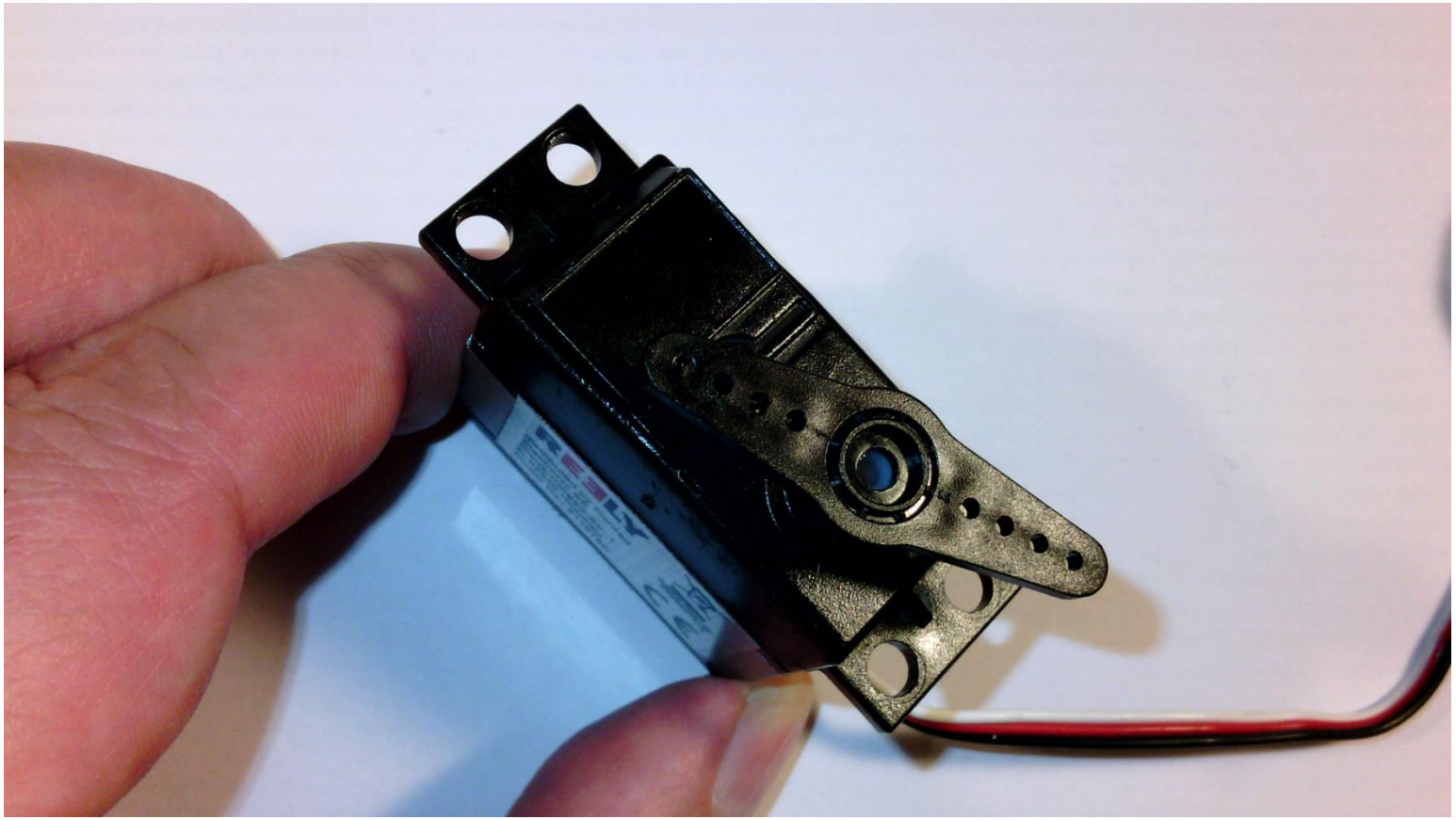
EBW Hannover

- Servo hacken
- Potentiometer auf 90°justieren
- Software Potentiometer auf 90°
- Aufbau Servo
- Pulsweite entspricht Servo-Position
- DC-Motor vs. Servo
- PWM Arduino vs. Servo
- Software Lösungs-Varianten
- Pulsweiten ermitteln
- 180° Servo Arbeitsweise
- 360° Servo Arbeitsweise
- 360° Servo Programm

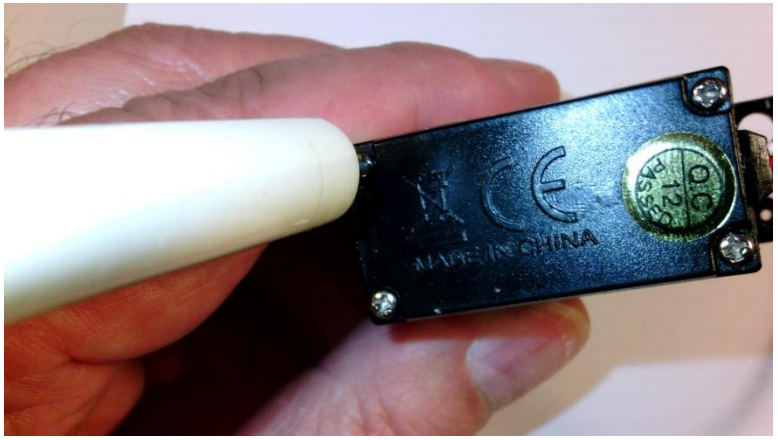
Enno Klatt

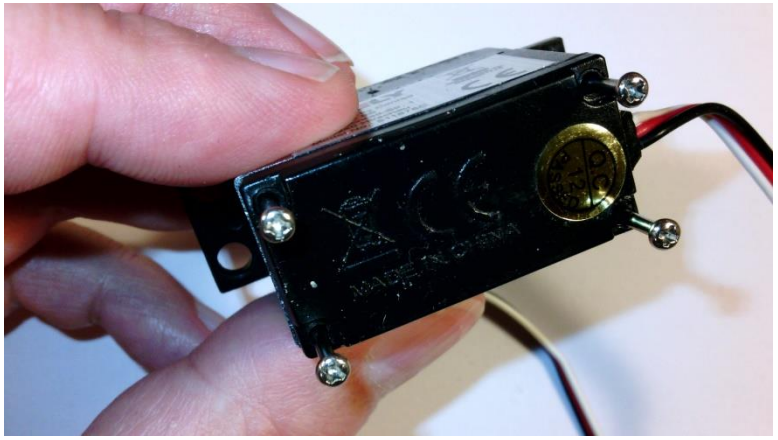
Servo RS-303 für Modellbau

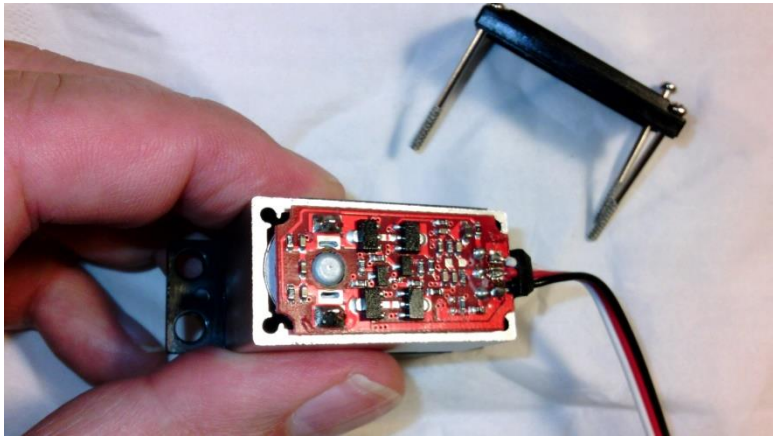
Servo RS-303 Servoarm



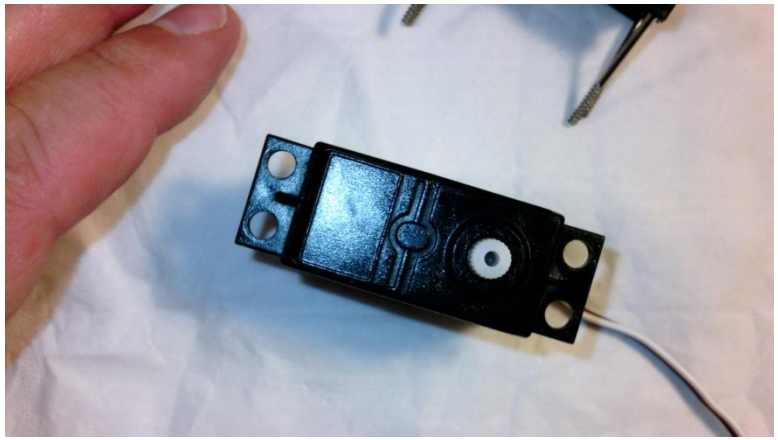
Gehäuse unten entfernen

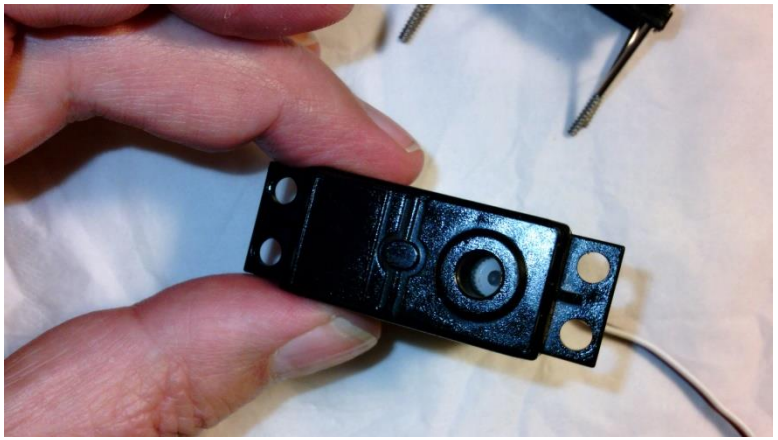


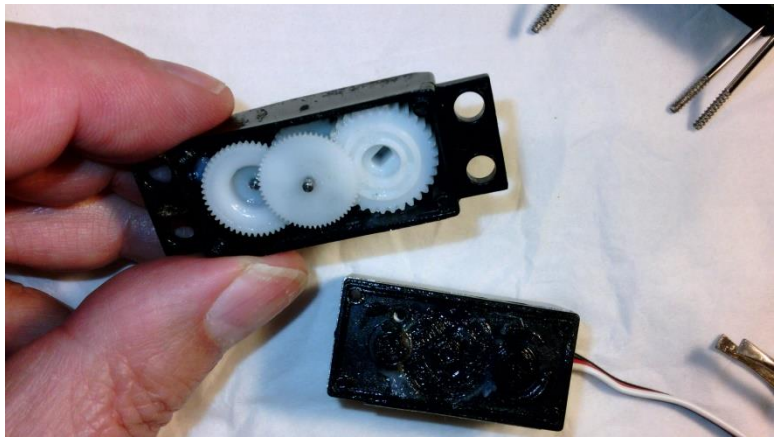


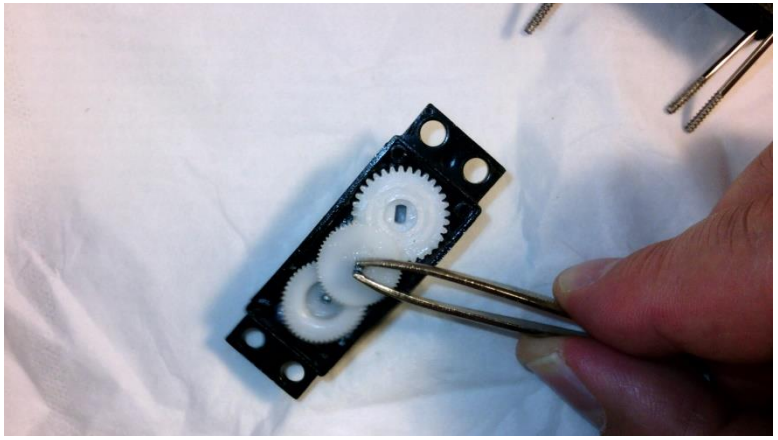


Gehäuse oben entfernen

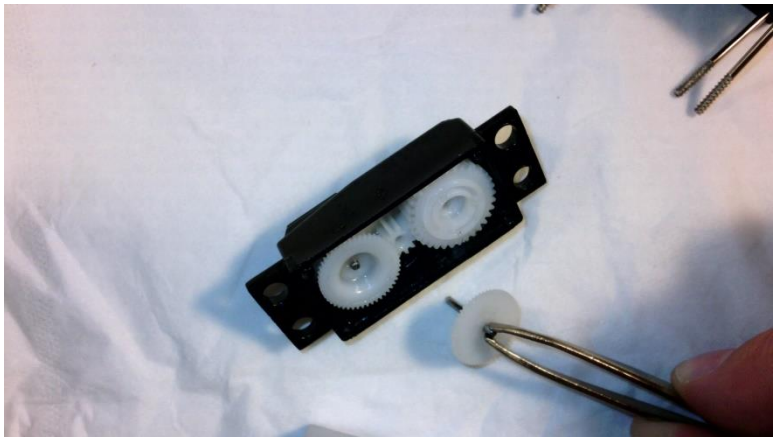


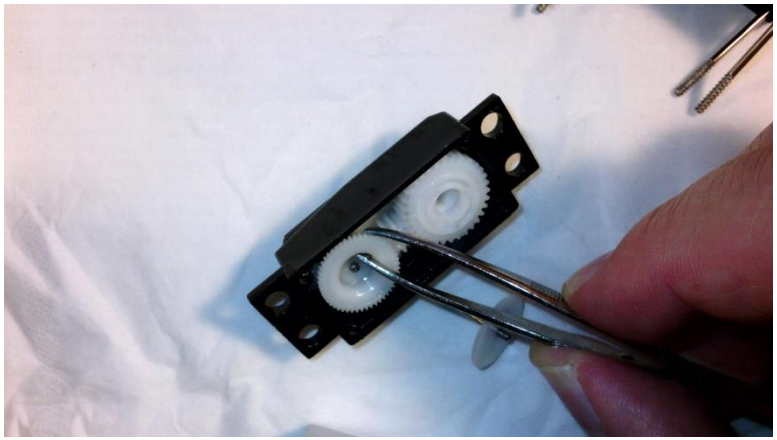


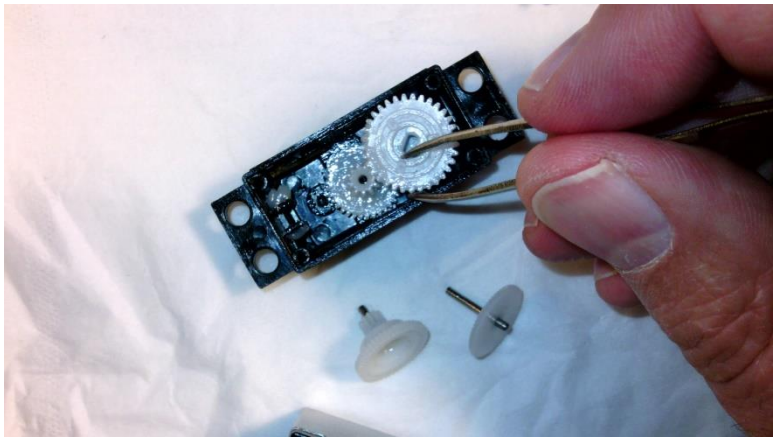




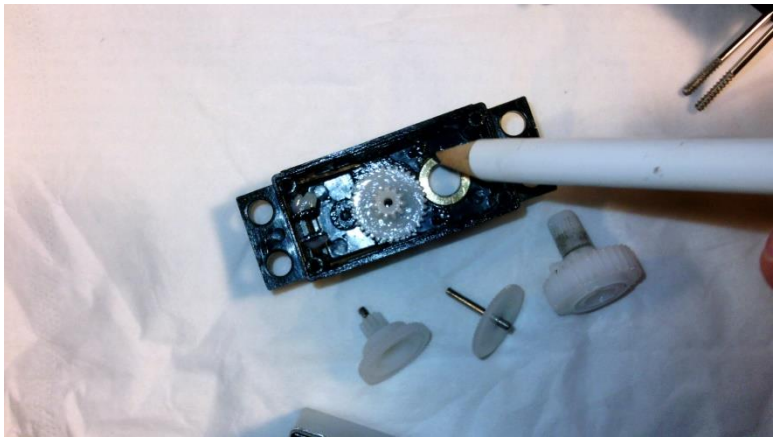
Getriebe demontieren

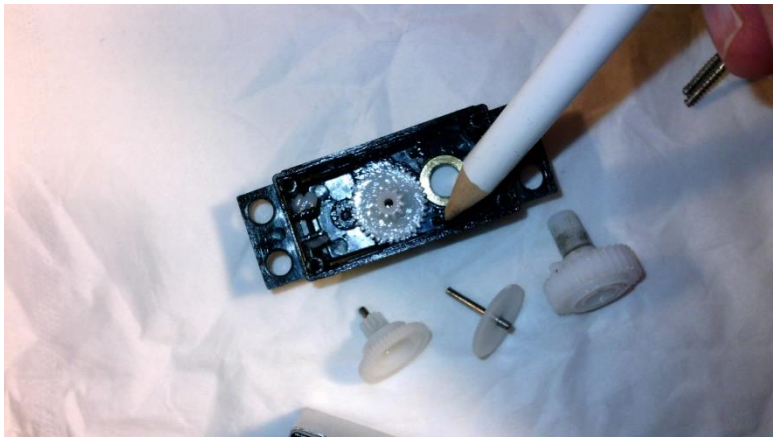




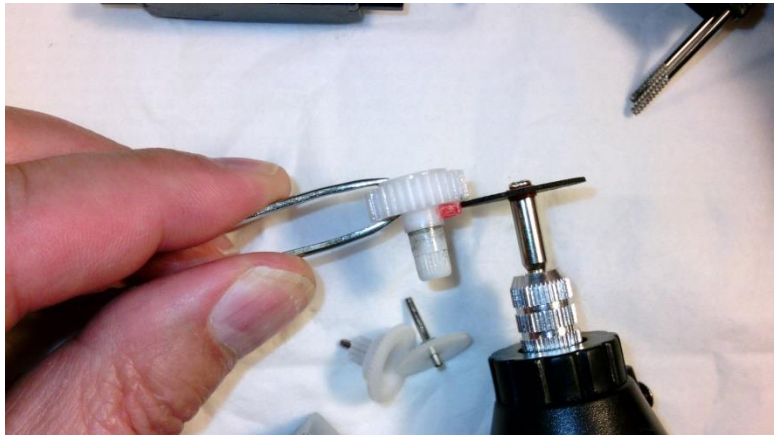


Mechanische Drehbegrenzung

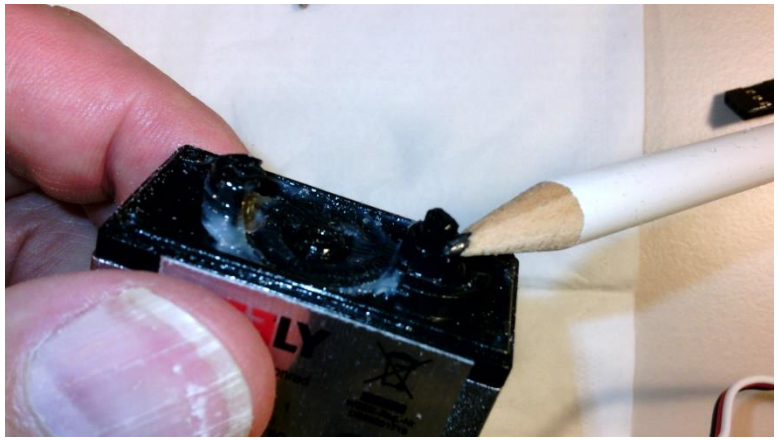


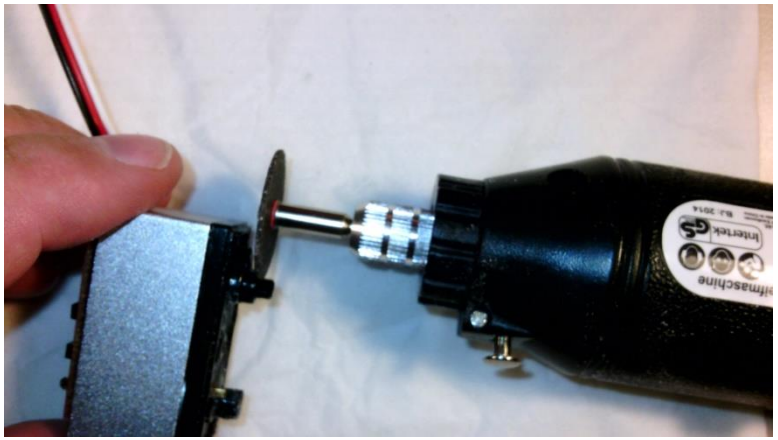


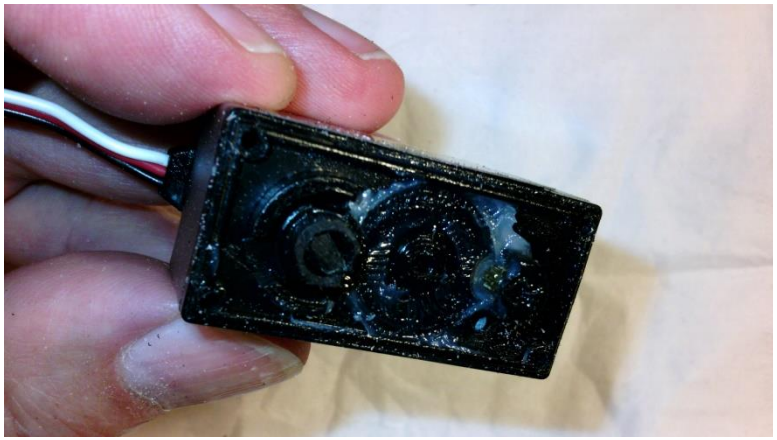
Nocken am Zahnrad entfernen



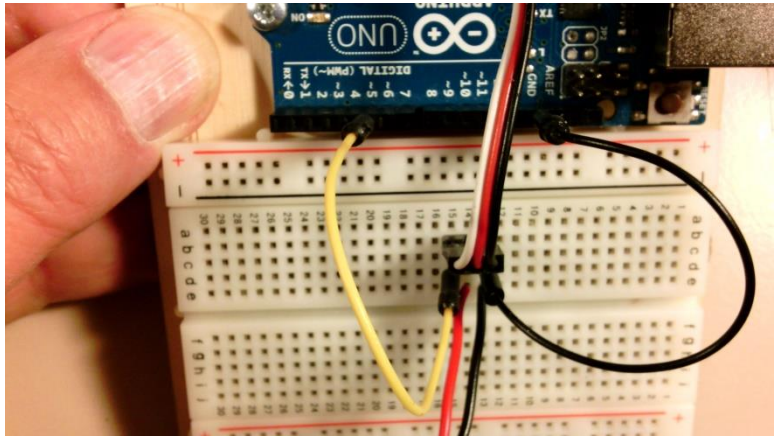
Nocken am Potentiometer entfernen

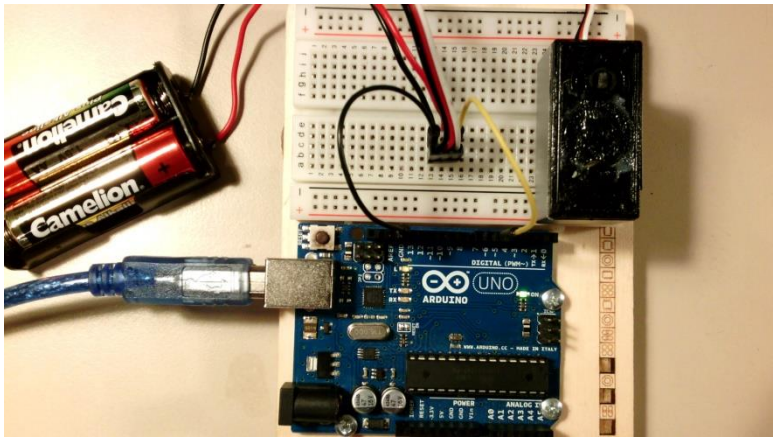


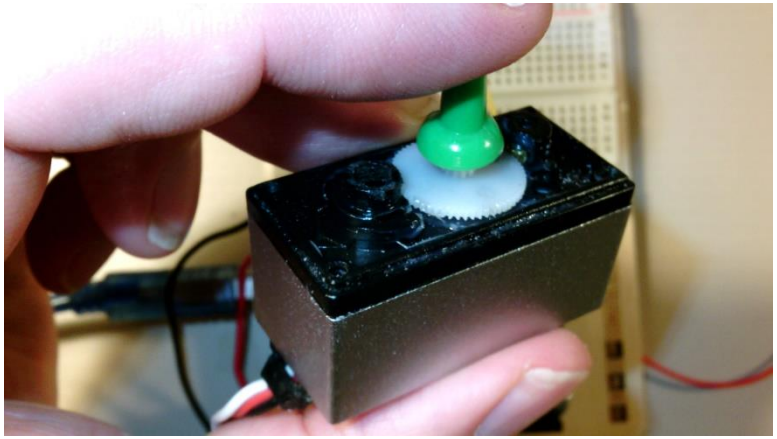




Potentiometer auf 90° justieren



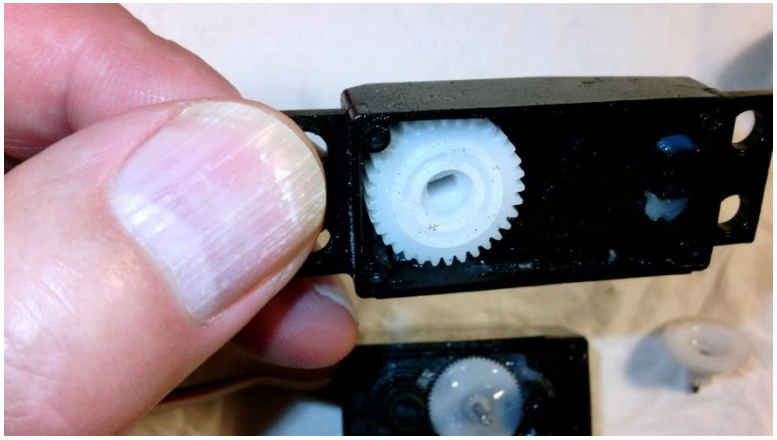


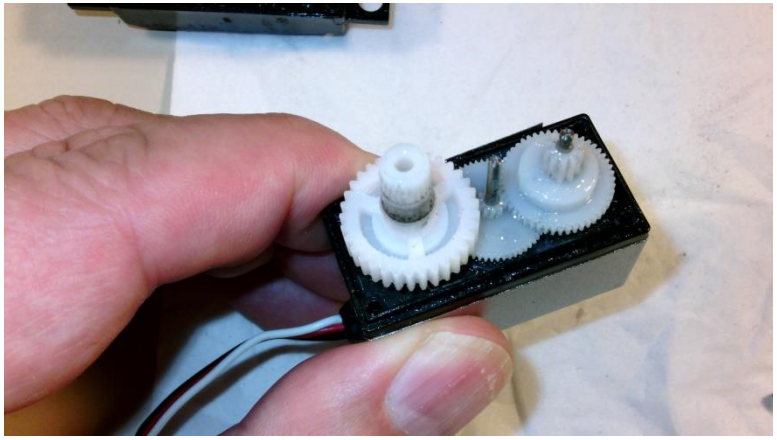


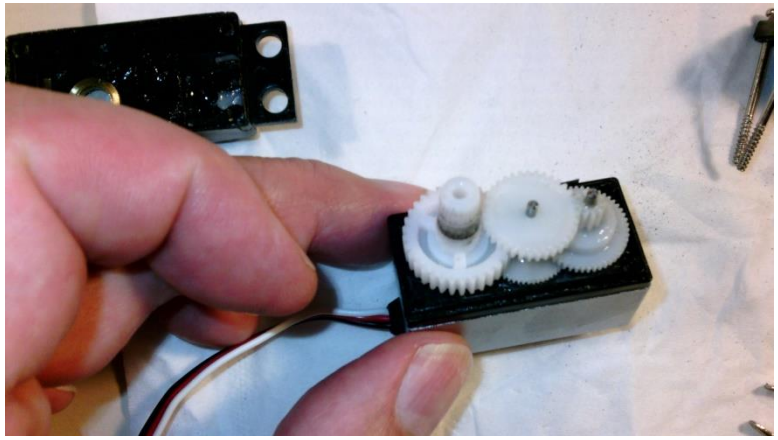
Software Potentiometer auf 90°

```
#include <Servo.h>              // Bibliothek Servo einbinden

int intServoPin = 5;            // Digital-Pin 5 für Servo
int intPotVal = 90;             // Setzen der 90° Position
Servo objServo;                 // Objekt (objServo) der Klasse
                                // Servo erzeugen

void setup() {
  objServo.attach(intServoPin);  // Das Servoobjekt (objServo)
                                 // verbinden mit Digital-Pin 5
}

 void loop() {
  objServo.write(intPotVal);
}
```

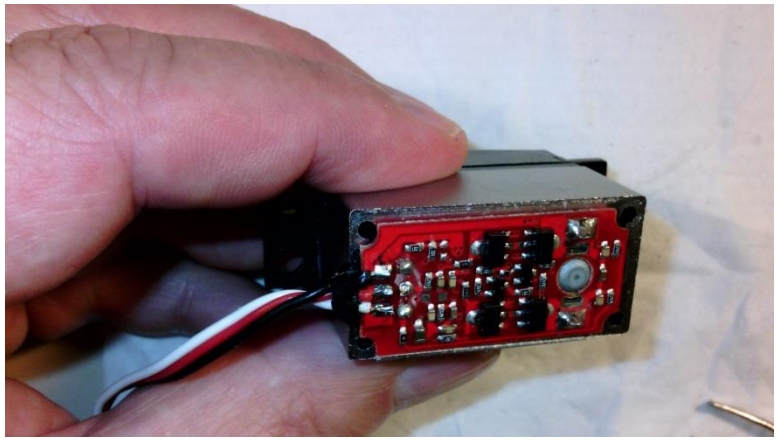
Zahnrad auf freie Drehung prüfen

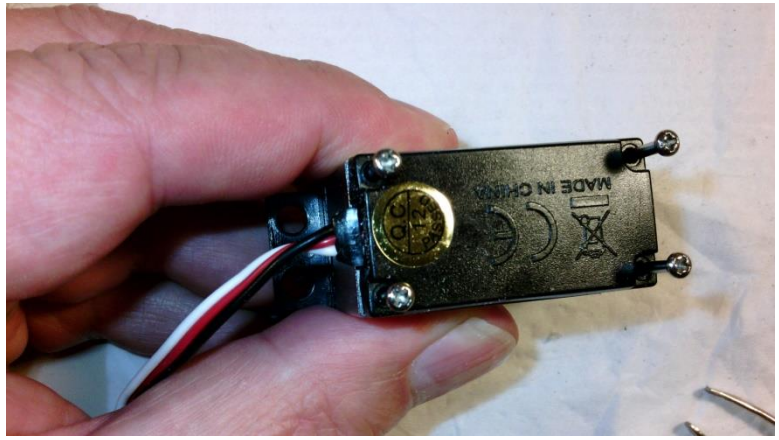


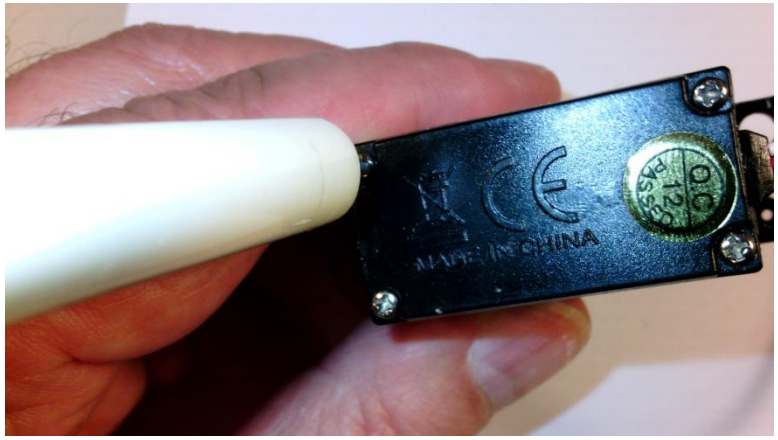


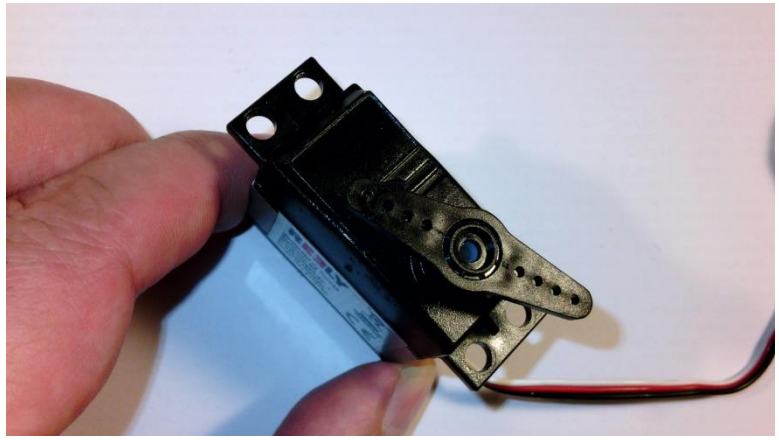


Unterteil montieren

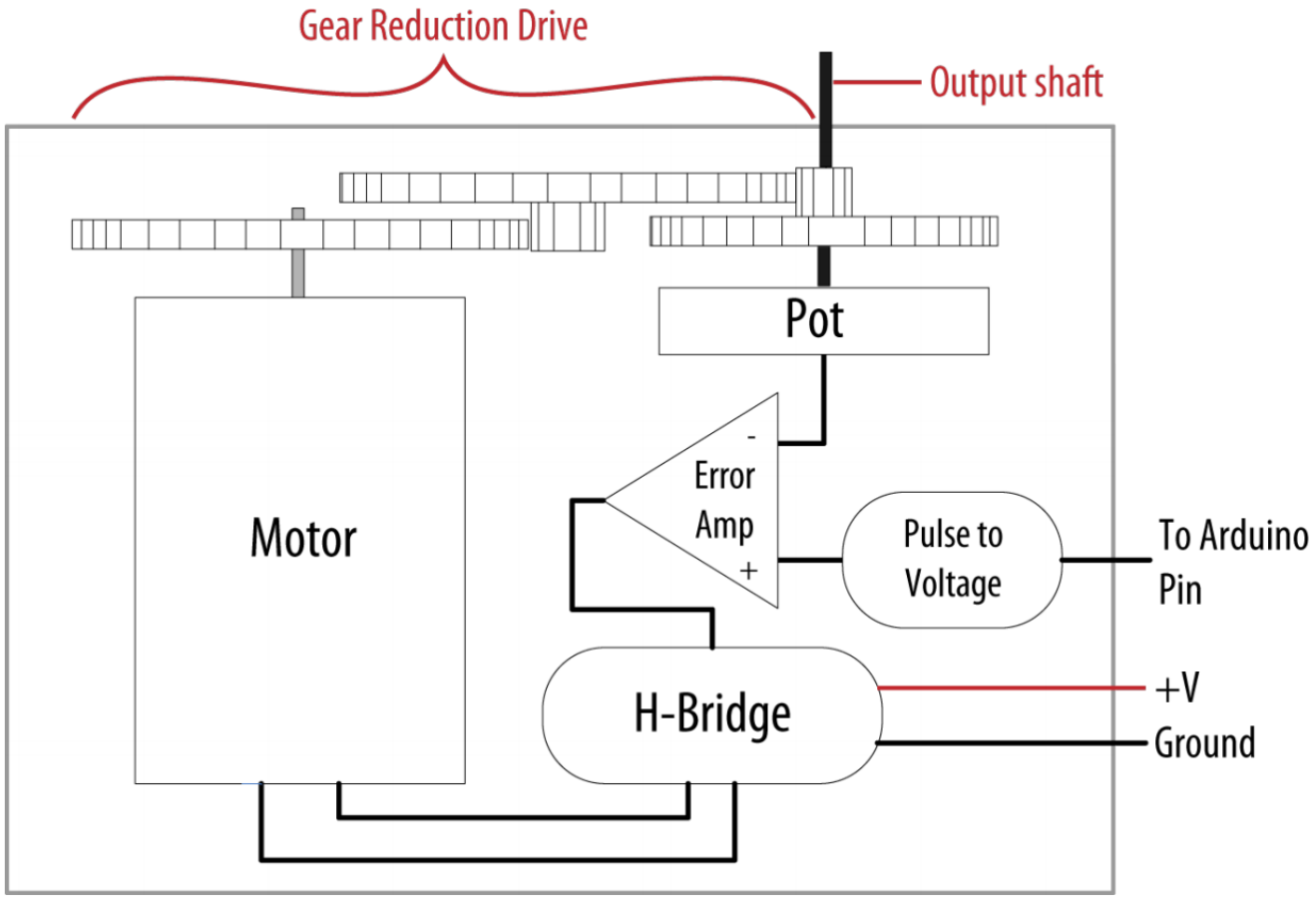






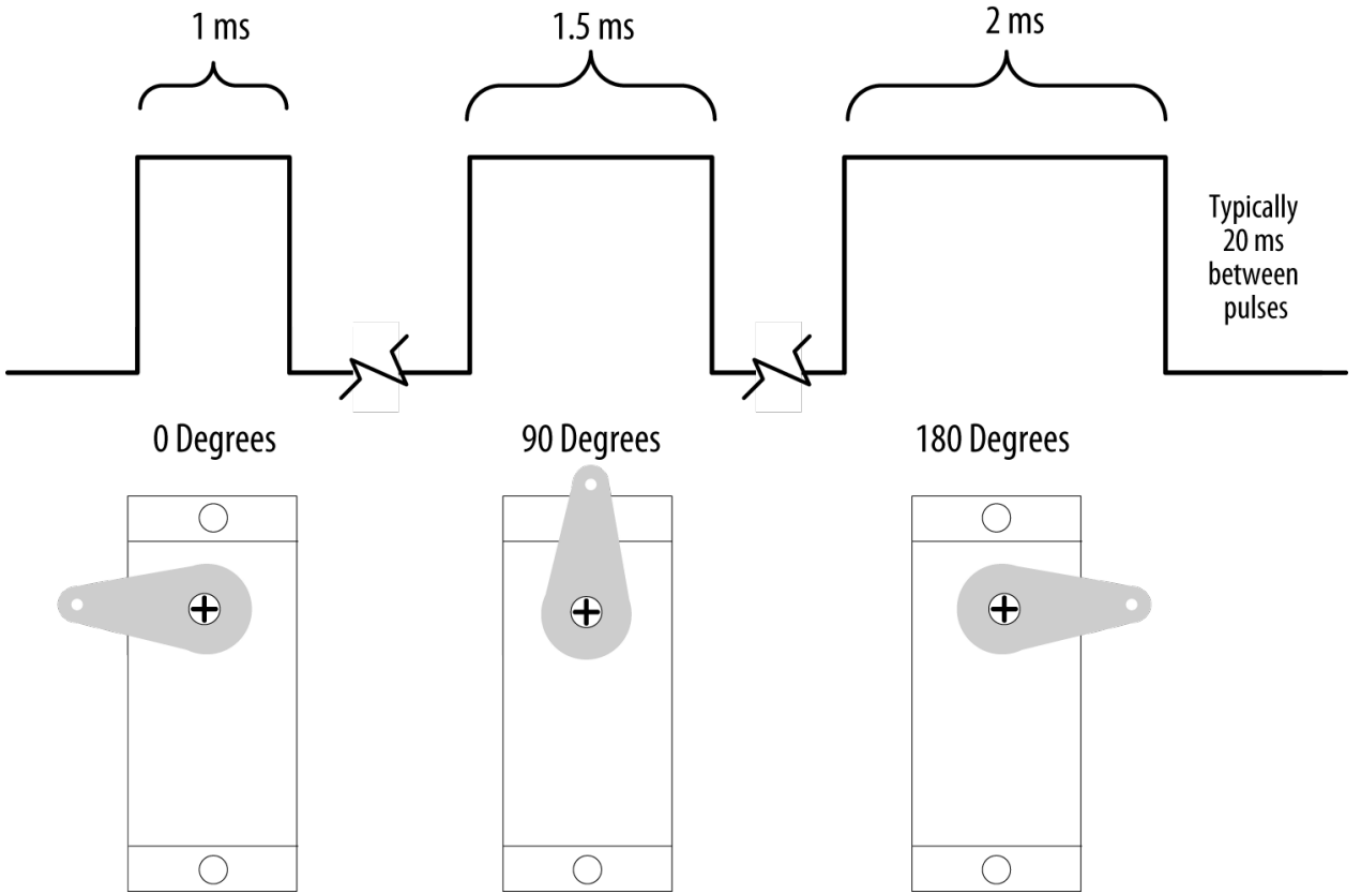


Aufbau Servo



aus Arduino cookbook

Pulsweite entspricht Servo-Position



DC-Motor vs. Servo

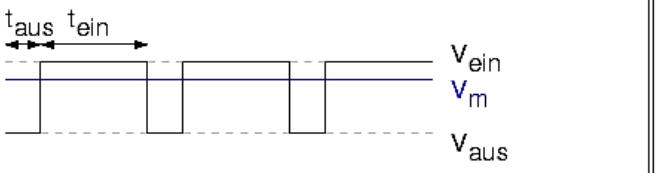
DC-Motor

- Steuerung der Drehzahl
- Frequenz (5-15kHz), Tastverhältnis
- PWM-Signal ergibt mittlere Spannung
- Konstante Frequenz
- Dauer der Pulse => Motorlaufzeit
- Tastverhältnis = 75%
  Um = 3,75 V

Beispiel 1

$U_{ein} = 5\,\mathrm{V}$
$U_{aus} = 0\,\mathrm{V}$
$t_{ein} = 3\,\mathrm{ms}$
$t_{aus} = 1\,\mathrm{ms}$

$$U_m = 0\,\mathrm{V} + (5\,\mathrm{V} - 0\,\mathrm{V}) \cdot \frac{3\,\mathrm{ms}}{3\,\mathrm{ms} + 1\,\mathrm{ms}} = 3,75\,\mathrm{V}$$



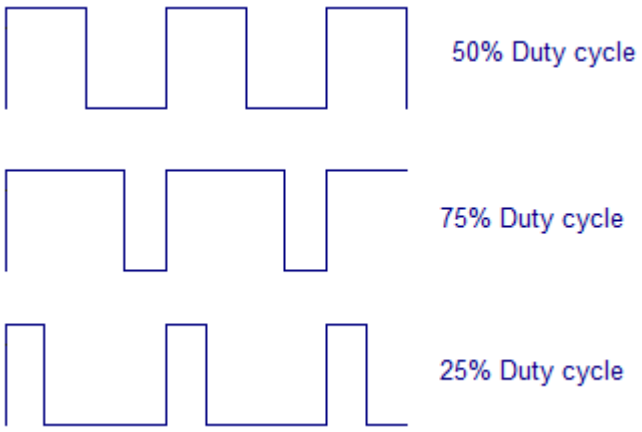
Servo

- Steuerung der Position
- Frequenz 50 Hz, Pulsweite
- Pulsweite ergibt Position
- Konstante Frequenz
- Anzahl der Pulse =>
  Trigger für Servo-Elektronik, die schrittweise zur Server-Position führt

PWM Arduino vs. Servo

PWM Arduino

- Frequenz 490 Hz / 980 Hz je nach Digital-Pin für PWM
- analogWrite(pin, value)
- value 0 bis 255
- geeignet für DC-Motoren



Servo

- Frequenz 20 ms / 50 Hz
- Arduino PWM nicht geeignet
- (Es sei denn Mikrokontroller-Ebene AVR-Bibliotheken)
- Arduino-Bibliothek „Servo.h“
- Standardwerte:
  MIN_WIDTH = 544
  MAX_WIDTH = 2400
  DEFAULT_WIDTH = 1500
- Funktionen:
  attach(Pin)
  attach(Pin, MIN_WIDTH, MAX_WIDTH)
  write(Position)
  writeMicroseconds(Mikrosekunden)

Software-Lösungs-Varianten

1. Variante

- „Servo.h“, attach(Pin), write(Position)
- „Servo.h“, attach(Pin, MIN_WIDTH, MAX_MIN_WIDTH), write(Position)

2. Variante

- „Servo.h“, attach(Pin), writeMicroSeconds(Mikrosekunden)
- „Servo.h“, attach(Pin, MIN_WIDTH, MAX_MIN_WIDTH), writeMicroSeconds(Mikrosekunden)

3. Variante

- Eigene Funktion mit einem Digital-Pin: servoMove(Pin, Position)

```
void servoMove( int Pin, int PosTime)
{
 Serial.println(Pos);
 for (int intI = 0; intI <= 50 ; intI++)
 {
  digitalWrite(Pin, HIGH);
  delayMicroseconds(PosTime);
  digitalWrite(Pin, LOW);
  delay(20);
 }
}
```

Pulsweiten ermitteln

- Die Arduino-Bibliothek „Servo.h“ führt zu genauen Servo-Positionen, wenn die Pulsweiten des jeweiligen Servos bekannt wären.
- Folgende Pulsweiten passen für den SM-S2309S

0 Grad, Standard	1 ms	Gefunden 490 µs	MIN_WIDTH
90 Grad, Standard	1,5 ms	Gefunden 1207 µs	DEFAULT_WIDTH
180 Grad, Standard	2 ms	Gefunden 2225 µs	MAX_WIDTH

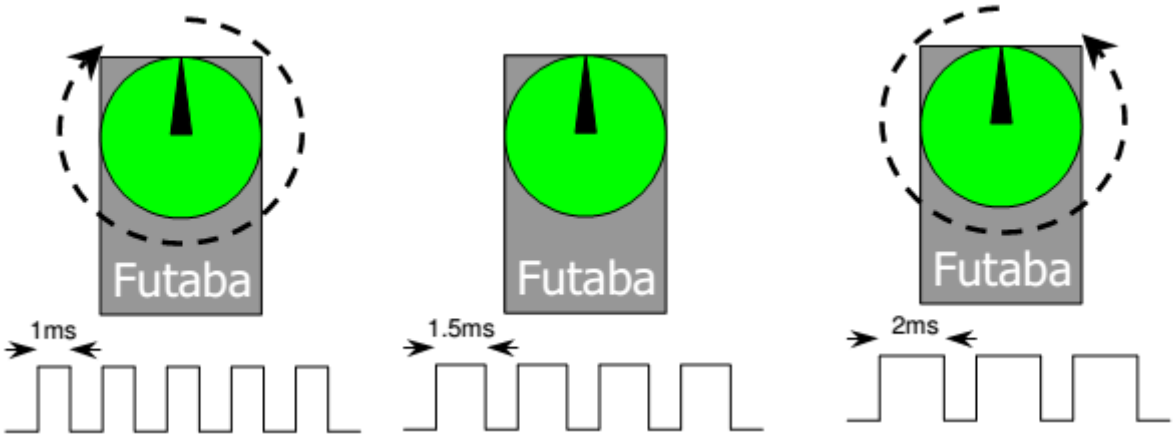
- Mit der Funktion „attach(pin, MIN_WIDTH, MAX_WIDTH)“ können die Standardwerte überschrieben werden.
- Mit „write(Position)“, wobei Position der Winkel in Grad ist, können die Positionen angefahren werden.
- Alternativ kann mit writeMicroseconds(Mikrosekunden), wobei „Mikrosekunden“ die o.g. Pulsweiten sind, können die Positionen angefahren werden.

180° Servo Arbeitsweise

- Der Servoarm hat eine beliebige Position, z.B. 100°.
- Bekommt die Servoelektronik Pulse für einen Sollwert von z.B. 45°, dreht der Servomotor in Richtung 45°.
- Der Servoarm ist mit einem Potentiometer verbunden, der stets einen Ist-Wert liefert, hier z.B. 80°.
- Der Servomotor dreht solange bis der Ist-Wert den Soll-Wert erreicht.
- Die Anzahl der Pulse muss groß genug sein, damit die jeweilige Drehung erfolgen kann.
- Die Drehung wird durch die mechanischen Anschläge zusätzlich begrenzt.

360° Servo Arbeitsweise

- Das Potentiometer muss manuell auf ca. 90° eingestellt werden.
- In der 90° Position steht der Servo still.
- Schickt man nun Pulse von einem von 90° nach unten oder oben abweichenden Winkel beginnt der Servo zu drehen.



- Linkes Bild mit z.B. 110° (90° + 20°) im Uhrzeigersinn.
- Rechtes Bild mit z.B. 70 ° (90° - 20°) gegen den Uhrzeigersinn.
- Die Drehgeschwindigkeit wächst mit dem Abstandswert zu 90°.

360° Servo Programm

```
#include <Servo.h>           // Bibliothek Servo
                             // einbinden

int intServoPin = 5;         // Digital-Pin 5 für
                             // Servo
int intServoVal;             // Winkel, daraus folgt
                             // Drehrichtung
                             // Geschwindigkeit Servo
int intServoCenter = 88;     // Ermittelte
                             // Mittelstellung

Servo objServo;              // Objekt (objServo) der
                             // Klasse Servo erzeugen

void setup()
{
  objServo.attach(intServoPin);  // Das Servoobjekt
                                 // (objServo)
                                 // verbinden mit
                                 // Digital-Pin 5
  Serial.begin(9600);
  Serial.print("Centerwert: ");
  Serial.println(intServoCenter);
  Serial.println("Moegliche Werte -20 H und +20 L");
                          // High => im Uhrzeigersinn
                          // Low  => entgegen
                          // Uhrzeigersinn
  intServoVal = intServoCenter;
}
```

```
void loop()
{
   if (Serial.available() > 0) {
    // lese den Integerwert
    intServoVal = Serial.parseInt();
    intServoVal = intServoCenter + intServoVal;
    Serial.print("Drehwinkel: ");
    Serial.println(intServoVal, DEC);
  }
  objServo.write(intServoVal);   // Servo auf den
                                 // einglesenen Wert
                                 // setzen
  delay(500);
}
```