

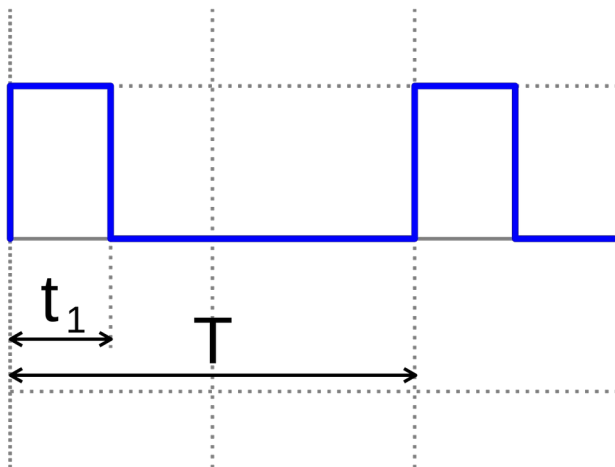
## Willkommen!

Und herzlichen Dank für den Kauf unseres MG995 Modellbau-Servo! Auf den folgenden Seiten gehen wir mit dir gemeinsam die ersten Schritte von der Einrichtung bis zur ersten Bewegung. Viel Spaß!



Beim AZ-Delivery MG995 Servo handelt es sich um einen Modellbauservoantrieb mit hoher Stell- und Haltekraft, sowie schneller Stellgeschwindigkeit. Im Inneren befinden sich ein Getriebe aus stabilen Metallzahnradern, ein analoges Potentiometer, sowie eine Platine zur Umsetzung der Steuerbefehle in Bewegung.

Im Gegensatz zu industriellen Servo-Einheiten werden Modellbauservoantriebe meist mithilfe der Pulsweitenmodulation (PWM) angesteuert, bzw. angesprochen.



Über die Breite der Pulse wird der Winkel, auf den der Servoarm gestellt werden soll, gesteuert.

Gängig ist ein 50-Hz-Signal (20 ms Periodenlänge), welches zwischen 500 Mikrosekunden (linker Anschlag, 0 Grad) und 2500 Mikrosekunden (rechter Anschlag, 180 Grad) auf High-Pegel und den Rest der Periodenlänge auf Low-Pegel ist.

Viele Servos haben in diesem Wertebereich jedoch nicht ihre volle Bewegungsfreiheit ausgenutzt und/oder können sich zwischen anderen Winkeln bewegen.

Zur Positionsregelung befindet sich im Servo ein Potentiometer, das mit der Ausgangswelle verbunden ist. Über dieses Potentiometer ermittelt die Servoelektronik den Ist-Winkel der Ausgangswelle

Dieser wird mit dem Soll-Winkel verglichen, der aus dem PWM-Signal ermittelt wird. Bei einer Abweichung zwischen Ist- und Soll-Winkel regelt die Elektronik über den Motor und das Getriebe den Winkel der Ausgangswelle nach.

## Die wichtigsten Informationen in Kürze

» **Abmessungen:** je nach Ausführung

» **Verbindung:**

VCC	4,8 V - 7.2V
GND	Masse
Data	PWM-Signal

» **Temperaturbereich:** -10 - +100 °C

» **Maximale Leistungsaufnahme:** bis 1600 mA

» **Programmierung über PWM-Signal,** Bibliothek in Arduino IDE

Auf den nächsten Seiten findest du Informationen zur

» ***Einrichtung der Hardware***

Diese Anleitung setzt voraus, dass du weißt, wie du Sketche auf einen Arduino hochlädst und die IDE verwendest!

## Alle Links im Überblick

### » **Datenblatt:**

[https://cdn.shopify.com/s/files/1/1509/1638/files/Servo\\_M\\_G995\\_Datenblatt.pdf?17227628845248709399](https://cdn.shopify.com/s/files/1/1509/1638/files/Servo_M_G995_Datenblatt.pdf?17227628845248709399)

### » **Bibliothek Arduino:** <https://www.arduino.cc/en/Reference/Servo>

### **Programmieroberflächen:**

- » Arduino IDE: <https://www.arduino.cc/en/Main/Software>
- » Web-Editor: <https://create.arduino.cc/editor>
- » Arduino-Erweiterung "Visual Micro" für Atmel Studio oder Microsoft Visual Studio:  
<http://www.visualmicro.com/page/Arduino-for-Atmel-Studio.aspx>
- » PlatformIO: <https://platformio.org/>

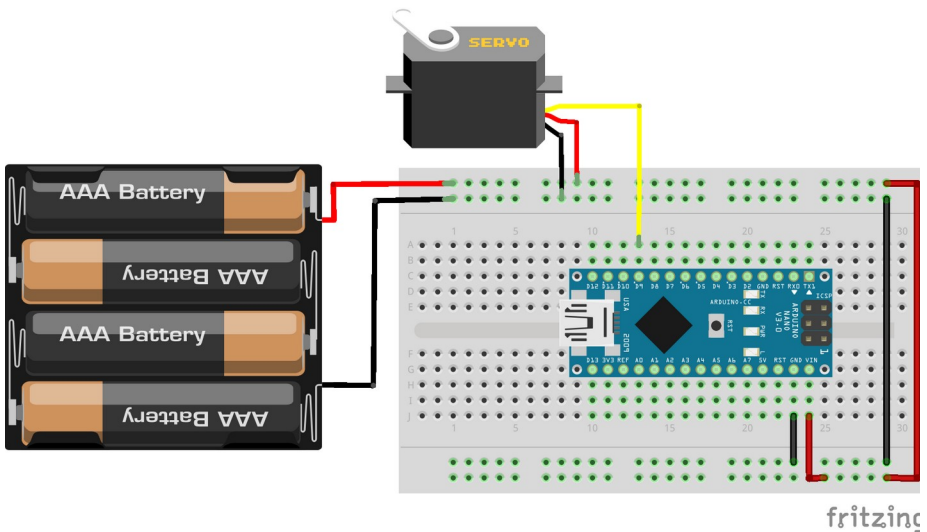
### **Arduino Tutorials, Beispiele, Referenz, Community:**

- » <https://www.arduino.cc/en/Tutorial/HomePage>
- » <https://www.arduino.cc/en/Reference/HomePage>

# Einrichtung der Moduls am Arduino

Wir beginnen mit der Einrichtung der Hardware:

Bitte beachten Sie, dass Modellbuantriebe meist mehr Strom benötigen als ein Arduino o.ä. Mikrocontroller leisten können. Deshalb ist eine separate Stromversorgung unabdingbar. Haben Sie bei Ihrem Projekt eine Spannungsquelle mit 5V können Sie diese verwenden. Für einen mobilen Betrieb ist ein Li-Po 2S empfehlenswert, dieser liefert ca. 7,4V.

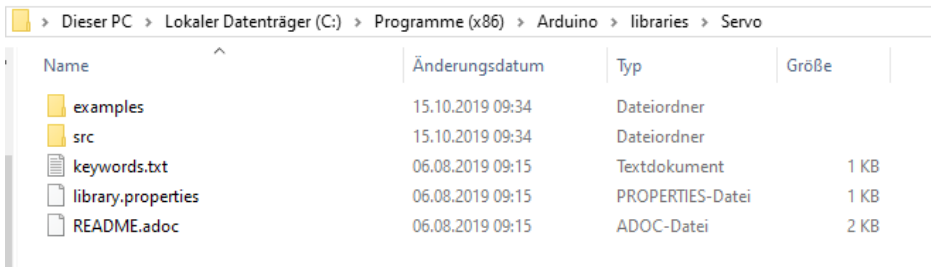


Bei der Wahl des richtigen Pins für die Datenleitung ist es wichtig, dass dieser PWM-Signale unterstützt. Bei den meisten Arduinos u.ä. Controller ist der entsprechende Port mit „~“ markiert. Genauere Angaben finden Sie in der Anleitung Ihres Controllers.

## Die Installation der Library:

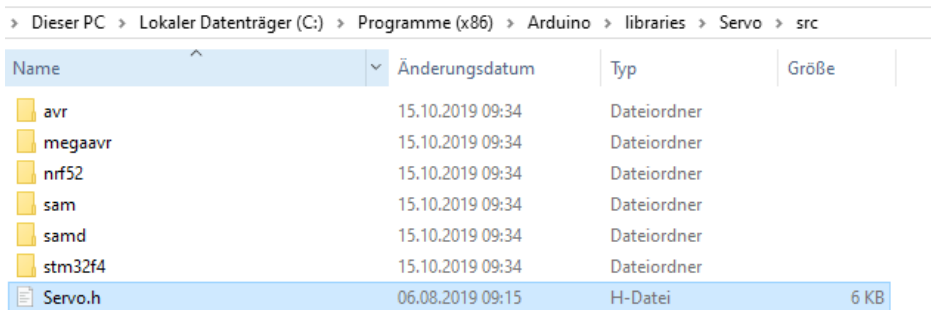
Die bei der Installation der Arduino IDE bereits enthaltene Standardbibliothek kann mit einer kleinen Anpassung problemlos genutzt werden. Eine eigene Bibliothek ist nicht erforderlich.

Haben Sie die Arduino IDE wie in unserem E-Book angegeben installiert, sollte die Standard-Library unter folgendem Pfad zu finden sein:



Name	Änderungsdatum	Typ	Größe
examples	15.10.2019 09:34	Dateiordner	
src	15.10.2019 09:34	Dateiordner	
keywords.txt	06.08.2019 09:15	Textdokument	1 KB
library.properties	06.08.2019 09:15	PROPERTIES-Datei	1 KB
README.adoc	06.08.2019 09:15	ADOC-Datei	2 KB

Die benötigte Servo.h finden Sie im Ordner „src“.



Name	Änderungsdatum	Typ	Größe
avr	15.10.2019 09:34	Dateiordner	
megaavr	15.10.2019 09:34	Dateiordner	
nrf52	15.10.2019 09:34	Dateiordner	
sam	15.10.2019 09:34	Dateiordner	
samd	15.10.2019 09:34	Dateiordner	
stm32f4	15.10.2019 09:34	Dateiordner	
Servo.h	06.08.2019 09:15	H-Datei	6 KB

In der Konfigurationsdatei Servo.h müssen die Pulsweiten für den Servoantrieb angepasst werden um den vollen Stellbereich nutzen zu können.

Öffnen wir die Datei mit einem beliebigen Editor finden wir die entsprechenden Einträge zwischen Zeile 78 und 88.

```
78 #define Servo_VERSION      2      // software version of this library
79
80 #define MIN_PULSE_WIDTH    544    // the shortest pulse sent to a servo
81 #define MAX_PULSE_WIDTH    2400   // the longest pulse sent to a servo
82 #define DEFAULT_PULSE_WIDTH 1500  // default pulse width when servo is attached
83 #define REFRESH_INTERVAL   20000  // minimum time to refresh servos in microseconds
84
85 #define SERVOS_PER_TIMER    12     // the maximum number of servos controlled by one timer
86 #define MAX_SERVOS         (_Nbr_16timers * SERVOS_PER_TIMER)
87
88 #define INVALID_SERVO      255    // flag indicating an invalid servo index
```

In Zeile 80 setzten wir die minimale Pulsbreite auf **500** und in Zeile 81 die maximale Pulsbreite auf **2500**, wie in nachstehender Abbildung zu sehen:

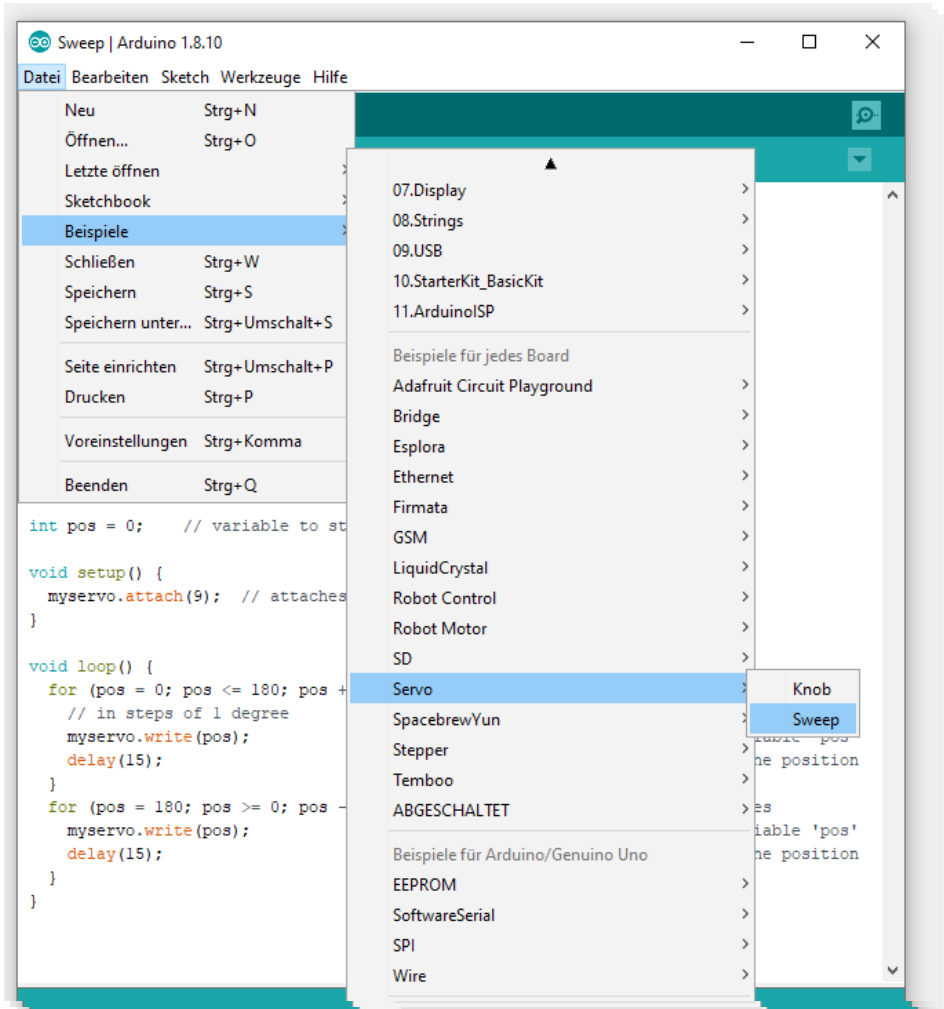
```
78 #define Servo_VERSION      2      // software version of this library
79
80 #define MIN_PULSE_WIDTH    500    // the shortest pulse sent to a servo
81 #define MAX_PULSE_WIDTH    2500   // the longest pulse sent to a servo
82 #define DEFAULT_PULSE_WIDTH 1500  // default pulse width when servo is attached
83 #define REFRESH_INTERVAL   20000  // minimum time to refresh servos in microseconds
84
85 #define SERVOS_PER_TIMER    12     // the maximum number of servos controlled by one timer
86 #define MAX_SERVOS         (_Nbr_16timers * SERVOS_PER_TIMER)
87
88 #define INVALID_SERVO      255    // flag indicating an invalid servo index
```

Im Anschluss öffnen wir die Arduino IDE und den, der Standardbibliothek beiliegendem Beispielsketch „Sweep“



## Die erste Bewegung unseres Antriebs:

Der mitgelieferte Sketch „Sweep“ fährt den Servoantrieb Winkel



für Winkel, über einen Zähler, den gesamten Stellbereich ab.  
Sie finden diesen unter Datei → Beispiele → Servo → Sweep.

## **Du hast es geschafft! Herzlichen Glückwunsch!**

Ab jetzt heißt es lernen und ausprobieren. Du weißt nun wie ein Mikrocontroller Modellbau Servoantriebe ansteuern kann. Jetzt kannst du versuchen die Werte praktisch einzusetzen.

Diesen Aktor und noch mehr Hardware findest du natürlich in deinem Online-Shop auf:

<https://az-delivery.de>

Viel Spaß!

## **Impressum**

<https://az-delivery.de/pages/about-us>