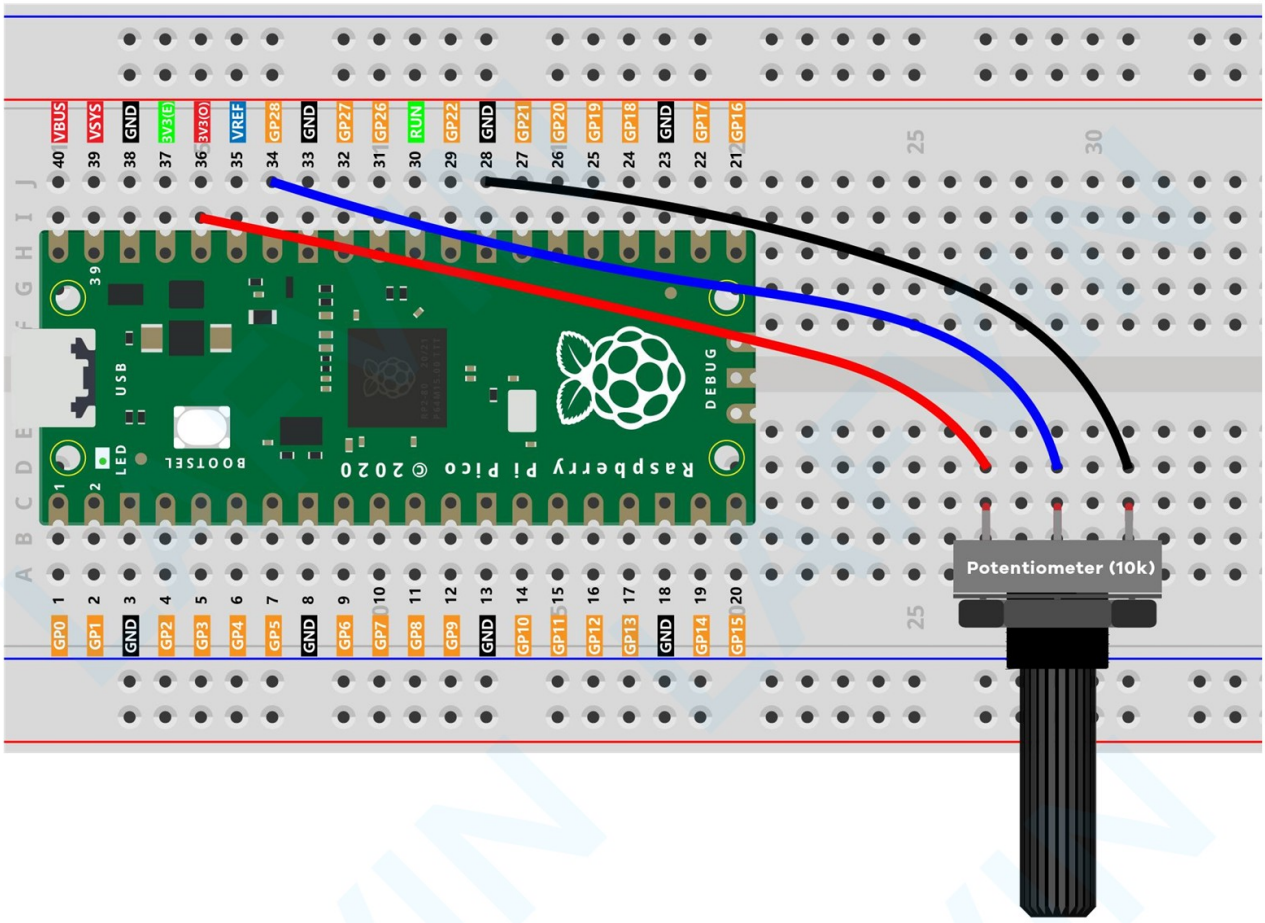
Potentiometer

Ein Potentiometer (‚Poti') ist ein Drehregler, der intern einen Drehwinkelabhängigen Widerstand hat. Ist er „voll aufgedreht", so fließt die volle Spannung, ist er „zugedreht", so ist die Spannung (fast) 0. Ausgelesen wird der Poti über ein ADC (Analog Digital Converter), so dass das Auslesen analog zum Thermistor erfolgt.

Der am ADC gemessene Wert liegt wie immer zwischen 0 und 65535, wobei in der Realität der Minimalwert eher um die 200 liegt. Wer es super präzise braucht, der sollte das bedenken.

Anschluß des Potentiometers



Beispielcode Potentiomenter auslesen

```
# Prüft das Auslesen der Potentiometerwerte
# -------------------------------------------------
import machine
import time

# Betriebsanzeige
led_intern = machine.Pin(25, machine.Pin.OUT)

# Potentiometer (analogerInput) definieren
potentiometer = machine.ADC(26)

# Hauptprogramm
while True:
    # Wert auslesen und in Konsole ausgeben
    aktuellerWert = potentiometer.read_u16()
    print("Neuer Wert = {}".format(aktuellerWert))
    # interne LED toggeln und halbe Sekunde warten
    led_intern.toggle()
    time.sleep(0.5) # 500 Millisekunden
```

Servos ansteuern

Es gibt, simpel zusammengefasst, zwei Arten von Elektromotoren:

- Antriebsmotoren, z.B. im Elektroauto oder in vielen Maschinen, wo eine dauuerhafte Drehbewegung benötigt wird.
- Stellmotoren (Servomotoren oder kurz ‚Servos' genannt). Diese können oft nur innerhalb eines bestimmten Drehbereich (z.B. innerhalb von 180 Grad) rotieren, sind dafür aber sehr präzise. Sie eignen sich vor allem um etwas präzise auszurichten (z.B. einen Zeiger) oder zu Bewegen (Hebel, etc.)

Wir schauen uns einen Stellmotor vom Typ SG90, der innerhalb von 180° rotieren kann.

Ansteuerung von Servos über PWM

Die Ansteuerung erfolgt über Pulsweitenmodulation (Pulse Width Modulation, kurz PWM). In einem PWM-Signal kann man eine Information im Rechtecksignal mit unterschiedlicher Pulsbreite übertragen. Die Information befindet sich im Verhältnis der Impulsbreite bzw. Impulsweite zur Periodendauer des Signals.[1]

Der Raspberry Pi Pico kann an jedem GPIO ein PWM-Signal erzeugen und damit Geräten über PWM ansteuern. Einmal programmiert, wiederholt der PWM-Generator das Signal so lange, bis es geändert oder abgeschaltet wird. Die Initialisierung eines GPIO als PWM-Generator erfolgt über die **PWM**-Klasse aus der machine-Bibliothek:

```
pwm = machine.PWM(machine.Pin(16))
pwm.freq(50)
```

Die über PWM angeschlossenen Geräte werden über DutyCycle und Frequenz gesteuert. Die Frequenz bestimmt wie oft das Signal pro Sekunde wiederholt wird, während der DutyCycle bestimmt, wie lange während einer Signalperiode der Pegel des Signals auf High gesetzt ist.

Programmatisch kann man den SG90 dabei über das Timing oder diskrete Werte steuern.

Steuerung über das Timing

Beim Timing gelten folgende Werte in Nanosekunden:

- 0° Ausrichtung entspricht dem Wert `500000`
- 90° Ausrichtung entspricht dem Wert `1500000`
- 180° Ausrichtung entspricht dem Wert 2`500000`

Diese werden mit

```
pwm.duty_ns(1500000)
```

gesetzt. Mit `pwm.deinit()` wird am Ende der PWM-Generator wieder deaktiviert.

1 https://www.elektronik-kompendium.de/sites/raspberry-pi/2802081.htm

Steuerung über diskrete Werte

Bei der Steuerung über diskrete Werte gelten für den beigefügten SG90 die folgenden Werte:

- 0° Ausrichtung entspricht dem Wert `1638`
- 90° Ausrichtung entspricht dem Wert `4915`
- 180° Ausrichtung entspricht dem Wert `8192`

Diese werden mit

`pwm.duty_u16(4915)`

gesetzt. Mit `pwm.deinit()` wird am Ende der PWM-Generator wieder deaktiviert.

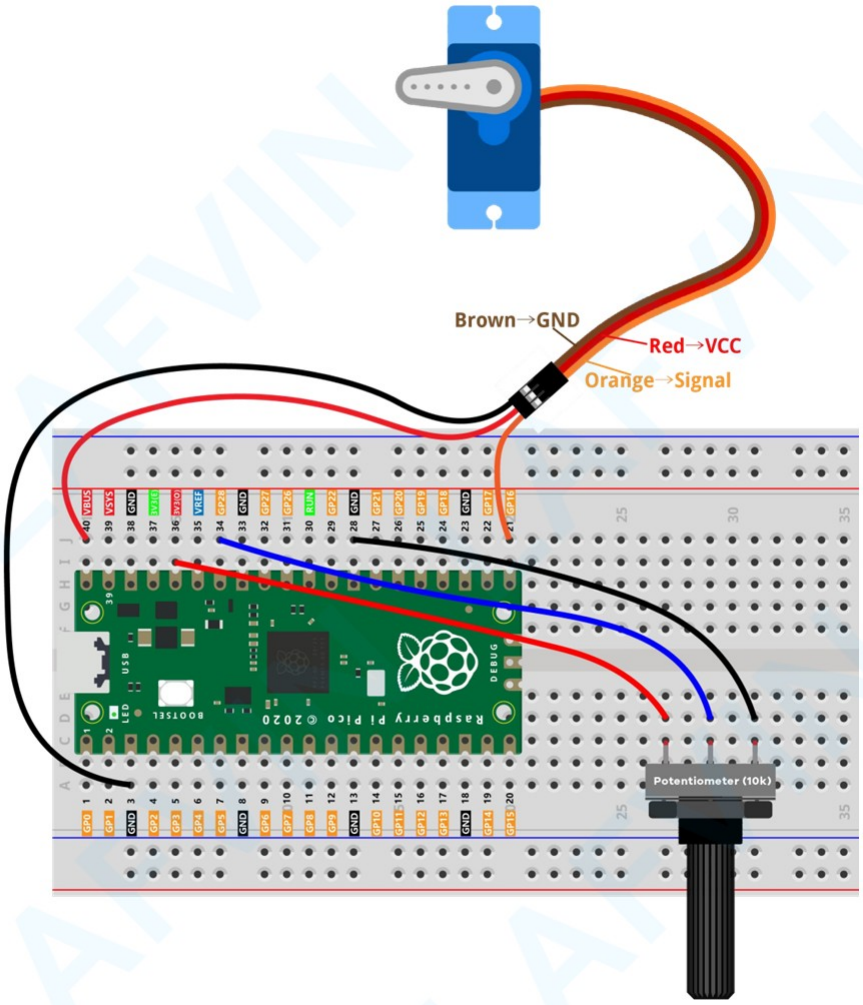
Beispielcode Motor steuern

Entnommen aus https://www.elektronik-kompendium.de[2]:

```
from machine import Pin, PWM
from time import sleep

# GPIO für Steuersignal
servo_pin = 28

# 0 Grad
grad000 = 1638
# 90 Grad
grad090 = 4915
# 180 Grad
grad180 = 8192

pwm = PWM(Pin(servo_pin))
pwm.freq(50)
print('Position: Ganz Links (0 Grad)')
pwm.duty_u16(grad000)
sleep(2)

print('Position: Mitte (90 Grad)')
pwm.duty_u16(grad090)
sleep(2)

print('Position: Ganz Rechts (180 Grad)')
pwm.duty_u16(grad180)
sleep(2)

print('Position: Mitte (90 Grad)')
pwm.duty_u16(grad090)
sleep(2)

pwm.deinit()
```

2https://www.elektronik-kompendium.de/sites/raspberry-pi/2706231.htm

Beispiel Servo mit Potentiometer steuern



Ziel: In Abhängigkeit von der Position des Poti bewegt sich der Servo auf die entsprechende Stellung.

⚠ Wichtig: der Servo bekommt seinen Strom mit 5V, d.h. vom VSYS-Pin (Pin 40), das Potentiometer dagegen verwendet 3.3 Volt (Pin 35 3V3).

Beispielcode

```
# Bibliotheken laden
import machine
import time

# DutyWert fuer 0 Grad
dutyWert000Grad = 1638
# DutyWert fuer 180 Grad
dutyWert180Grad = 8192

# Nutzbarer Wertebereich von 0 - 180 Grad
dutyWerteBereich = dutyWert180Grad - dutyWert000Grad

# Pins initialisieren
ledOnBoard = machine.Pin(25, machine.Pin.OUT)
adcPoti = machine.ADC(26)

# Motor initialisieren
pwm = machine.PWM(machine.Pin(16))
pwm.freq(50)

# Bestimmt Winkel für Zeiger aus Temperatur
def bestimmeZeigerWinkelAusADC(adcWert):
    winkel = (adcWert / 65535) * 180.0
    return winkel

# Bestimmt den an den Motor zu übergebenden Wert anhand des Winkels
def bestimmeDutyWertFuerWinkel(winkel):
    # nur gültige Winkel akzeptieren
    if (winkel > 180):
        winkel = 180
    elif (winkel < 0):
        winkel = 0
    # Steuerwert bestimmen
    dutyBereichsWert = (winkel / 180.0) * dutyWerteBereich
    dutyWert = int(dutyBereichsWert + dutyWert000Grad)
    print("PWM DutyWert: {} (entspricht {:.2f}°)".format(dutyWert,
winkel))
    return dutyWert

while True:
    # ADC auslesen und Steuerwert für Zeiger bestimmen
    adcWert = adcPoti.read_u16()
    winkel = bestimmeZeigerWinkelAusADC(adcWert)
    dutyWert = bestimmeDutyWertFuerWinkel(winkel)
    # Motor mit berechnetem DutyWert ansteuern
    pwm.duty_u16(dutyWert)
    # onboard LED toggeln und schlafen
    ledOnBoard.toggle()
    time.sleep(0.5)
```