Analog/Digital-Wandler (A/D-Wandler)

Von Hause aus kann der Raspberry Pi nur digitale Signale verarbeiten. Er kann also nur überprüfen, ob an einem Pin eine Spannung anliegt oder nicht. Dieses wird bei dem Gebrauch von Schaltern eingesetzt. Um analoge Messungen auswerten zu können, bei denen es auch auch Zwischenwerte geht, wird ein Analog/Digital-Wandler, oder auch kurz AD, benötigt. Dieser misst eine Spannung und wandelt sie in eine Zahl in einem bestimmten Zahlenbereich um. Der MCP3008 ist ein solcher 10-Bit A/D-Wandler. Er kann Zahlenwerte zwischen 0 und $2^{10} - 1 = 1023$ erfassen. Der MCP3208 hat sogar eine Auflösung von 12 Bit und liefert somit Werte von 0 bis $2^{12} - 1 = 4095$.

1 SPI einrichten

Damit der A/D-Wandler nicht mit 10 Leitungen angeschlossen werden muss, kommuniziert er mit dem Raspberry Pi über das SPI-Protokoll. Dazu hat der Pi vorbelegte Pins die im Device-Tree aktiviert werden müssen. Dieses kann in raspi-config unter »Advanced Options« eingestellt werden oder durch hinzufügen folgender Zeile in die /boot/config.txt:

```
dtparam=spi=on
```

Nach einem Neustart des Raspberry Pi steht dann das SPI zur Verfügung.

2 Anschluss des A/D-Wandlers

Im ersten Schritt wird nur der A/D-Wandler an den Raspberry Pi angeschlossen. Die Anschlüsse des MCP3008 sind identisch zum MCP3208 und in der linken unteren Zeichnung abgebildet. Wie dieser A/D-Wandler mit dem Raspberry Pi verbunden werden muss, ist der der Schaltzeichnung und der rechten Tabelle zu entnehmen. In dieser Aufstellung ist bereits die Referenzspannung mit angeschlossen, weshalb in dieser Anordunung die zu messende Spannung niemals 3,3 V überschreiten darf.

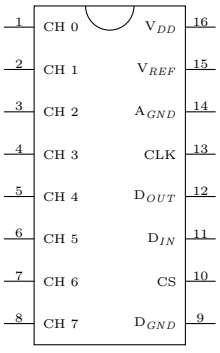


Abbildung 1
MCP3208

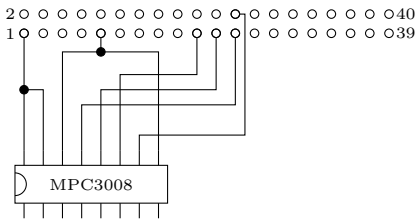


Abbildung 2
MCP3208 am Raspberry Pi

A/D	GPIO
9 (D_{GND})	9 (GND)
10 (CS)	24 (CE0)
11 (D_{IN})	19 (MOSI)
12 (D_{OUT})	21 (MISO)
13 (CLK)	23 (SCLK)
14 (A_{GND})	9 (GND)
15 (V_{REF})	1 (3,3 V)
16 (V_{DD})	1 (3,3 V)

Tabelle 1
Pinbelegung des ADs am Raspberry Pi

3 Einbindung in Python

Die Einbindung des MCP3008 in Python ist durch die Python Bibliothek »gpiozero«[1] sehr einfach. Es muss nur bei der Erzeugung des A/D-Wandler Objekts der Kanal angegeben werden, von dem der Wert ausgelesen werden soll.

```
1 import gpiozero
2 adw = gpiozero.MCP3008(channel=3)
3 value = adw.value # liefert Wert von 0 bis 1
4 value2 = adw.raw_value # liefert Wert von 0 bis 1023
```

Bei dem oberen Beispiel ist der Unterschied zwischen value und raw_value zu beachten. Beim ersten wird der des A/D-Wandlers so umgerechnet, dass er immer zwischen 0 und 1 liegt. Beim zweiten liegt der Wert direkte Wert vor, der beim MCP3008 bis 1023 geht, beim MCP3208 aber bis 4095.

4 Anschluss eines Potentiometers

Das Potentiometer ist ein regelbarer Widerstand. Schließt man ihn zwischen der Masse und 3,3 V an, so erhält man zwischen Masse und dem Anschluss für den regelbaren Widerstand eine Spannung zwischen 0 V und 3,3 V, je nachdem wie weit das Potentiometer eingestellt ist. Mit Hilfe des A/D-Wandlers kann man so aus einem Potentiometer einen Regler für Programme machen.

Der Anschluss eines Potentiometers ist der Schaltzeichnung zu entnehmen. In diesem Fall ist er an Kanal 0 angeschlossen, es wäre aber auch jeder andere der 8 Kanäle möglich. Entsprechend sieht auch das dazugehörige Python-Programm aus.

```
1 import time, gpiozero
2
3 adc = gpiozero.MCP3008(channel=0)
4
5 while True:
6   wert = int(adc.value * 100)
7   print('Der␣Poti␣steht␣auf', wert, '%')
8   time.sleep(1)
```

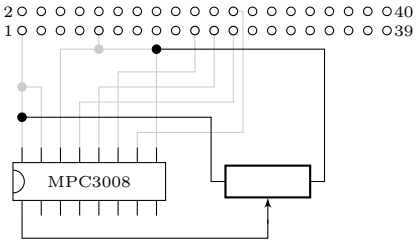
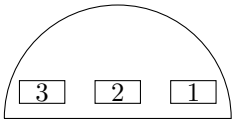


Abb. 3: Regelbarer Widerstand am AD-Wandler

[1] Diese Bibliothek kann ggf. mit sudo apt-get install python3-gpiozero nachinstalliert werden

5 Anschluss eines analogen Thermometers

Ein analoger Thermometerchip hat am Datenpin je nach gemessener Temperatur eine unterschiedliche Spannung. Für den Einsatz am Raspberry Pi eigenen sich dabei Thermometer aus der TMP36-Reihe. Diese haben folgende Belegung: Pin 1 ist für die Spannung, Pin 2 liefert die Daten und an Pin 3 muss die Masse angeschlossen werden.



blick von unten

Abb. 4: Analoges Thermometer TMP36

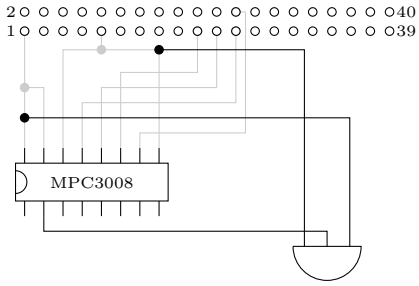


Abb. 5: Analoges Thermometer am AD-Wandler

Dem Schaltplan ist zu entnehmen, wie das Thermometer an den Kanal 1 angeschlossen wurde. Um aus dem gemessenen Wert die Temperatur bestimmen zu können, muss dieser passend umgerechnet werden. Beim TMP36 kann man folgende Formel dafür nutzen:

$$t = (v \cdot 3{,}3 - 0.5) \cdot 100$$

Dabei ist v der gemessene Wert zwischen 0 und 1 und die $3{,}3$ kommt durch die Referenzspannung am A/D-Wandler.

Beim Auslesen ist zu beachten, dass die gemessenen Werte des TMP36 bereits bei der Messung um 0.5° vom echten Wert abweichen können. Daher sollte man die errechneten Werte im Programm auf maximal eine Nachkommastelle runden. Das komplette Python-Programm zum auslesen des Thermometers sieht dann so aus:

```
1 import time, gpiozero
2
3 adc = gpiozero.MCP3008(channel=1)
4
5 while True:
6   temp = (adc.value * 3.3 -0.5) * 100
7   temp = round(temp, 1)
8   print('Die Temperatur ist', temp, 'C')
9   time.sleep(1)
```