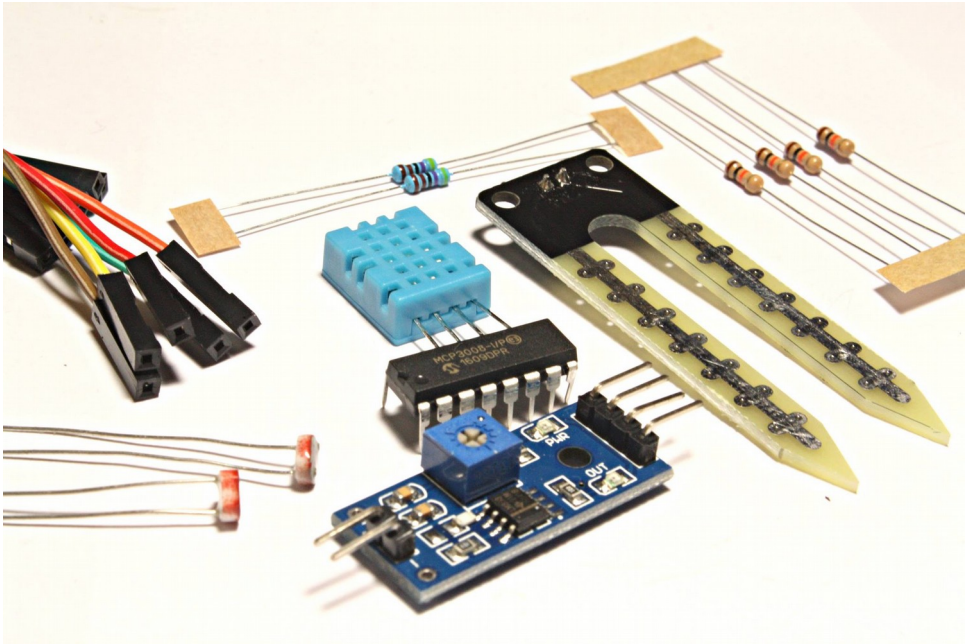
Jugend Programmiert

Die

Schlaue Pflanze



www.codingworld.io

Vorwort

Alle Schaltungen, Programme und Bauteile wurden vorher von uns Geprüft. Trotzdem können wir leider Fehler im Buch, den Schaltungen oder Programmen nicht ausschließen. Bei Fragen oder Problemen gibt es unten die Kontaktmöglichkeiten. Mehr Informationen findest du auch auf der Internetadresse der Seitenzahl.

LEDs

sollten nicht aus naher Entfernung direkt angesehen werden, ein direkter Blick kann zu Schäden an den Netzhäuten führen. Diese kann gefährlich sein, auch wenn sie nicht sofort klar erkennbar sind. Die LEDs dürfen nur wie in den Anleitungen beschrieben verwendet werden, höhere Ströme oder Stromspannungen sind zu vermeiden.

Dieses Produkt entspricht den geltenden Europäischen Richtlinien und trägt ein CE-Kennzeichen. Der richtige Gebrauch ist in dem Beiliegenden Buch erklärt.

CE

Bauen sie Schaltungen immer wie beschrieben auch, achten sie auf die Verschiedenen GPIO Pins vom Raspberry Pi.

Herausgeber

Jugend Programmiert
Coding World UG (haftungsbeschränkt) | Homwer.com
Obdrupstraße 23 A, 24986 Mittelangeln
www.codingworld.io
Support: support@cw42.de
Feedback: feedback@cw42.de

Moin Moin und Willkommen zur Smart Plant - Schlaue Pflanze Anleitung!

In diesem Kit sind alle wichtigen Komponenten enthalten, um Daten über deine Pflanze zu messen und auszuwerten. Das Ziel ist es, dass du am Ende nicht nur eine schlaue Pflanze hast, sondern dich auch mehr mit der Technik dahinter auseinandersetzen konntest. Alles was du dafür brauchst, ist ein laufender Raspberry Pi sowie das Grundwissen aus dem Jugend Programmiert Starterkit. Zusätzlich kannst du natürlich mit der Hardware des Starterkits deine Messstation noch ergänzen.

Mit folgenden Inhalten werden wir uns auseinandersetzen:

Die Lufttemperatur und Feuchtigkeit mit dem DHT11 messen

Da nicht jeder Mensch mit seinem Raspberry Pi die Temperatur messen möchte, ist die Bibliothek und die benötigte Software leider nicht vorinstalliert. Du musst diese Bibliothek also selber aus dem Internet laden. Wenn dein Pi schon eine Internetverbindung hat, dann mach hier einfach weiter. Wenn dein Pi noch keine Internetverbindung hat, kannst du den Pi einfach über ein Lan-Kabel an den Router ins Netz bringen. Mehr wie immer online!

```bash
$ sudo apt-get update
```

Mit den apt-get-Befehlen verwalten wir die Programme, welche auf unserem Computer oder Raspberry Pi installiert sind. Speziell bei diesen Quellen werden die Softwarequellen aktualisiert, damit wir auch immer nur die neusten Programme installieren.

Wenn du den Sensor mit Python2 benutzen möchtest musst du die folgenden Befehle einfach nur durch python ersetzen (also die 3 weglassen) ;)

```bash
$ sudo apt-get install build-essential python3-dev
```

Hiermit installieren wir jetzt das Programm, das wir brauchen. Was es wirklich kann, zeigt sich gleich.

Und mit folgendem Befehl lädst du dir die Bibliothek herunter:

```bash
$ git clone https://github.com/coding-world/Python_DHT.git
```

Das besondere ist dabei, dass die Bibliothek in den Ordner Python_DHT runtergeladen wird und deinem Computer noch nicht bekannt ist. Deswegen müssen wir erst einmal in den Ordner der Bibliothek wechseln.

```bash
$ cd Python_DHT
```

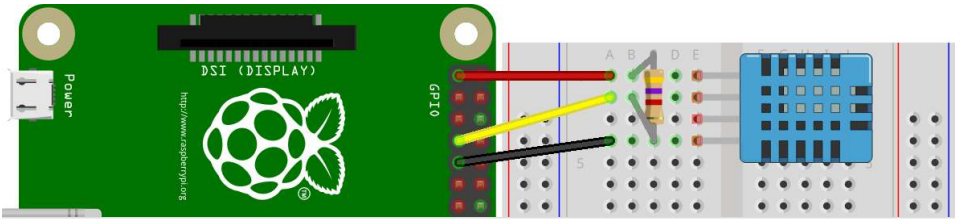
Mit

```bash
$ sudo python3 setup.py install
```

installieren wir die Bibliothek auf dem Raspberry Pi und können sie jetzt auch mit `import` in unserem Programm benutzen. In der nächsten Einheit werden wir dann den Sensor anschließen.

Schließe zunächst, wie immer, alles richtig an den Pi an. Die Vorderseite des DHT11 erkennst du an dem Gitter.

Anschließen



Anschlüsse am Pi	Anschlüsse am DHT
3,3V	1. Pin DHT
GPIO4	2. Pin DHT
Widerstand 4,7kOhm	Zwischen 1. Pin DHt und 2. Pin DHT
GND (Ground)	4. Pin DHT

- Zwischen dem 1. und 2. Pin am DHT musst du noch einen Widerstand anschließen. Das dient nicht nur zur Stromversorgung des Sensors, sondern auch dazu, dass dieser Messwerte liefern kann, die akkurater sind. Am besten eignet sich dafür der Widerstand mit

4,7kOhm. Widerstände erkennst du an ihrem aufgemalten Farbcode. Beim 4,7kOhm lautet dieser: Gelb, Lila, Schwarz, Braun und Gold. *

Bitte achte genau darauf, den Sensor in dieser Reinfolge anzuschließen! Wenn du den 3,3V und den GND Pin verwechselst, kommt es zu einem Kurzschluss und der Sensor kann dadurch kaputt gehen und sehr sehr warm werden. Achte darauf! Solltest du bemerken, dass der Sensor unnormal warm wird oder sich sogar das Plastik verbiegt, schalte sofort deinen Raspberry Pi aus oder entferne die Verbindung zwischen dem Breadboard und dem Sensor!

Wie dir hoffentlich aufgefallen ist, benutzen wir nicht den 3. Pin am DHT Sensor. Diesen kannst du sonst benutzen, um die Daten analog auszulesen.

Jetzt, da alles richtig angeschlossen und installiert ist, können wir uns endlich in der nächsten Einheit daran machen, die Daten auszulesen.

Jetzt ist alles fertig eingerichtet und du kannst mit dem Programmieren anfangen. Wie immer einfach eine neue Datei anlegen und los:

Bash
```
$ sudo nano dht11_einfach.py
```

Das unbeschriebene Blatt mit Code füllen…

Python
```
1 import Python_DHT
2
3 sensor = Python_DHT.DHT11
4 pin = 4
5 feuchtigkeit, temperatur = Python_DHT.read_retry(sens
6 print("Temperatur = "+str(temperatur)+ "C Feuchtigkei
```

Am Anfang binden wir wie immer die benötigte Bibliothek ein. In Zeile 3 erstellen wir eine Variable namens sensor und speichern in ihr einfach nur die Art des Sensors, den wir auslesen möchten. In unserem Fall ist das der der DHT11.

In Zeile 4 legen wir den Pin am Pi fest, den wir nutzen möchten.

In Zeile 5 lernst du wieder etwas Neues. Man kann nämlich auch mehrere

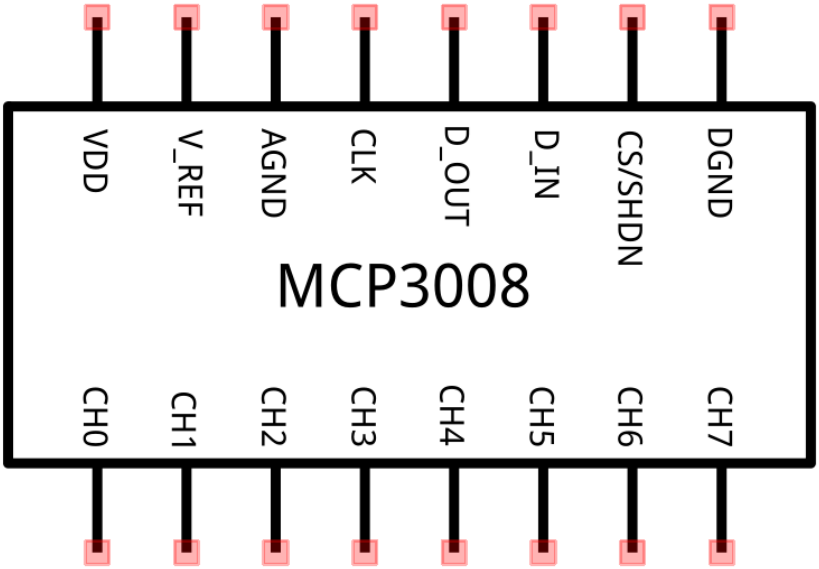
Variablen durch ein Komma getrennt hintereinander erzeugen und ihnen hinter dem Gleichzeichen, wieder durch Kommata getrennt, verschiedene Werte zuweisen. In diesem Fall geben wir mit der Funktion `.read_retry()` an, welchen Sensor und welchen Pin wir nutzen wollen.

Ab Zeile 6 geben wir wie gewohnt alle Informationen in der Konsole aus.

Jetzt bist du dran: Kriegst du es hin, dass ab einer bestimmten Temperatur, z.B. 25°C, eine LED anfängt zu leuchten? Tipp: Mit einem Atemhauch lässt sich die Temperatur erhöhen und logische Operatoren sind bestimmt auch nicht verkehrt.

MCP3025208 Analoge Werte am Raspberry Pi Messen

Mit dem Raspberry Pi haben wir ein mächtigen Computer und können mit den GPIO Pins einfach auf die Umwelt zugreifen und dort Daten lesen. Wir haben aber ein Problem, nämlich dass wir nur digitale Daten lesen können. Digitale Daten sind dabei nur eine 0 oder 1. Heißt, wenn wir mit Sensoren arbeiten, einen hohen oder einen niedrigen Strom haben. Ein Beispiel ist dabei ein Taster. Wenn Strom fließt, messen wir eine 1 und wenn kein Strom fließt, messen wir eine 0. Andere Sensoren übermitteln aber analoge Werte. Diese befinden sich zwischen 0 und 1. Um diese auszulesen, wird ein ADC gebraucht. Ein Analog zu Digital Konverter. Mit dem MCP3008 haben wir genau so ein Gerät. Über SPI können wir es an den Raspberry Pi anschließen und haben dann die Möglichkeit bis zu acht analoge Anschlüsse auszulesen.



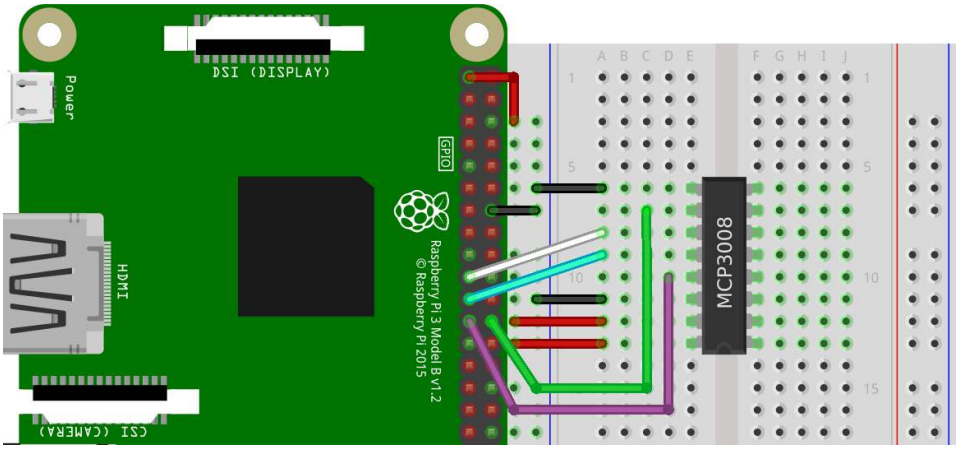
Die untere Seite erkennst du an der halbrunden Anordnung. CH0 - CH7 sind die analogen Eingänge, die wir benutzen werden.

- VDD: Spannung, die der MCP3008 für den Betrieb braucht
- V_REF: Ist die höchste Spannung, welche die analogen Anschlüsse bekommen. Diese werden gebraucht, um richtige Werte zu messen
- AGND: Masse für die analogen Anschlüsse
- CLK: Abkürzung für Clock (Uhr). Koordiniert die Geschwindigkeitszeit
- D_OUT: Ausgang zur SPI Schnittstelle am Raspberry Pi
- D_IN: Eingang zur SPI Schnittstelle am Raspberry Pi
- CS/SHDN: Abkürzung für Cable Select. Mit dem SPI lassen sich verschiedene Geräte gleichzeitig ansteuern
- DGND: Masse für den MCP3008

Wie gesagt brauchen wir SPI, wenn du das noch nicht aktiviert hast, gibt es hier eine Anleitung dafür:

SPI Aktivieren

Nachdem wir die Software Grundlagen hinter uns haben, geht es darum, die Hardware richtig anzuschließen.



Die halbrunde Nase zeigt in diesem Fall nach unten. Stecke den MCP3008 am besten über den mittleren Kanal des Breadboards, also zwischen e und f.

Anschlüsse Raspberry Pi	Anschlüsse MCP3008
GND	DGND
GPIO 8N	CS/SHD
GPIO 10	D_IN
GPIO 9	D_OUT
GPIO 11	CLK
GND	AGND
V_REF	3,3V
VDD	3,3V

Bash
```bash
$ nano mcp.py
```

Python
```python
#!/usr/bin/python
import spidev
import time
import os
import RPi.GPIO as gpio

# SPI Verbindung herstellen
spi = spidev.SpiDev()
spi.open(0,0)

# Liest Daten vom MCP3008
def analogEingang(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data

while True:
  print("0: "+str(analogEingang(0)))
  print("1: "+str(analogEingang(1)))
  print("2: "+str(analogEingang(2)))
  print("3: "+str(analogEingang(3)))
  print("4: "+str(analogEingang(4)))
  print("5: "+str(analogEingang(5)))
  print("6: "+str(analogEingang(6)))
  time.sleep(0.2)
```

Jetzt musst du nur noch das Programm ausführen

Bash
```bash
$ sudo python mcp.py
```

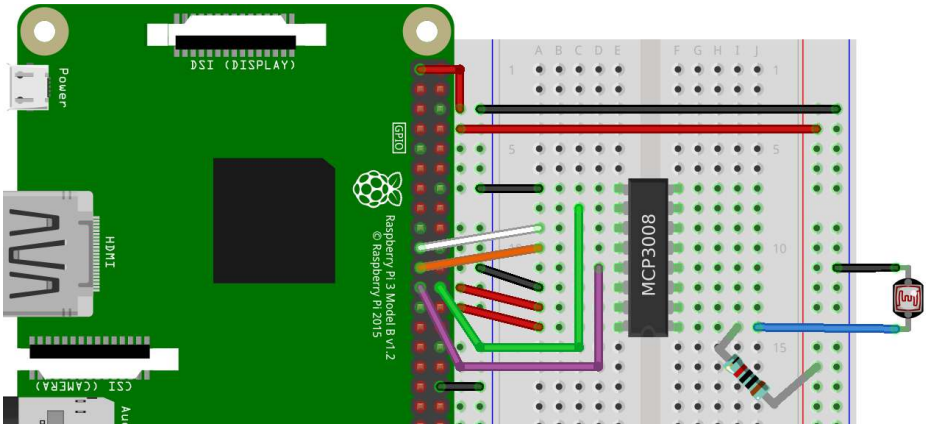
Wichtig ist, dass du das Programm mit Rootrechten ausführst und daran denkst, dass es sich dabei um ein Python2 Programm handelt. Wenn jetzt

keine Werte ausgegeben werden, sollte dich das nicht wundern, denn du hast ja auch noch nichts angeschlossen, was sinnvolle Werte ausgeben könnte ;)

Also schnell weiter zum ersten Sensor!

Der Photowiderstand am Pi

Wie der MCP3008 richtig angeschlossen wird, haben wir schon gelernt. Diesen brauchen wir auch, wenn wir den Photowiderstand am Raspberry Pi anschließen wollen. Der Photowiderstand ist, wie der Name es schon sagt, ein Widerstand und kann damit die Strommenge begrenzen. Doch was für einen Widerstandswert dieser hat, hängt von der Stärke des Lichts ab. Bei normalen Zimmerlicht liegt der Widerstandswert bei ca. 10KΩ während in der Dunkelheit der Wert bei über 2MegaΩ liegen kann. Mit dem MCP3008 können wir diesen Widerstandswert messen und damit Rückschlüsse auf die Helligkeit ziehen.

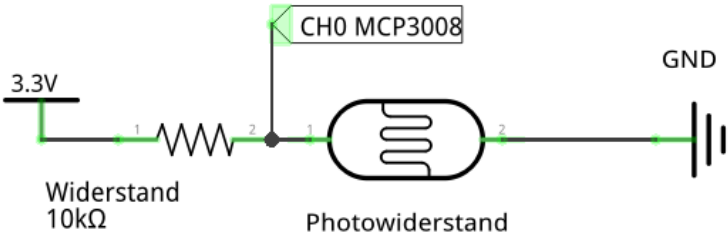


Anschlüsse Pi	Anschlüsse Sensor
3,3V + 10KΩ Widerstand	Seite 1 Photowiderstand
	CH0 + Seite 1 Photowiderstand
GND	Seite 2 Photowiderstand

```python
import spidev
import time
import os
import RPi.GPIO as gpio

# SPI Verbindung herstellen
spi = spidev.SpiDev()
spi.open(0,0)

# Liest Daten vom MCP3008
def analogEingang(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data

while True:
  print("0: "+str(analogEingang(0)))
  time.sleep(0.1)
```

Schritt für Schritt

Der Programmcode sollte dir nicht sehr unbekannt vorkommen, den es wird alles schon im Kapitel über den MCP3008 erklärt. Wir lesen einfach nur den ersten analogen Eingang aus, dieser wird auch CH0 genannt. Doch wir sollten noch einmal auf die elektronische Seite eingehen. Dafür haben wir extra diesen Schaltplan:



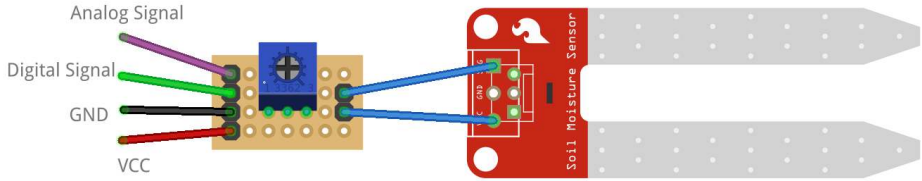
→ cw42.de/pflanze/11

Alles fängt mit der 3,3V Stromquelle an. Interessant wird es aber erst nach dem 10KΩ Widerstand, denn dort haben wir einmal den Anschluss zum analogen Eingang und über den Photowiderstand zu Erdung (GND). Wichtig ist zu wissen, dass Strom immer den weg des geringsten Widerstandes gehen will, bildlich gesprochen. Umso höher der Widerstandswert des Photowiderstandes ist, desto mehr Strom geht zum analogen Eingang und kann gemessen werden. Heißt, wenn du den Photowidertand z.B. mit dem Finger verdunkelst, steigt der gemessene Wert, während der Wert beim direkten Anleuchten, z.B. mit dem Smartphone, sinkt. So können wir einfache Rückschlüsse auf die Helligkeit ziehen.

Wir haben einen Wert von ca. 60 gemessen beim direkten Beleuchten des Photowiderstands, bei normaler Zimmerbelichtung ein Wert von ca. 300 und bei kompletter Dunkelheit einen Wert von 800. Diese Werte können bei dir natürlich variieren, sollten sich aber in einen ähnlichen Bereich befinden.

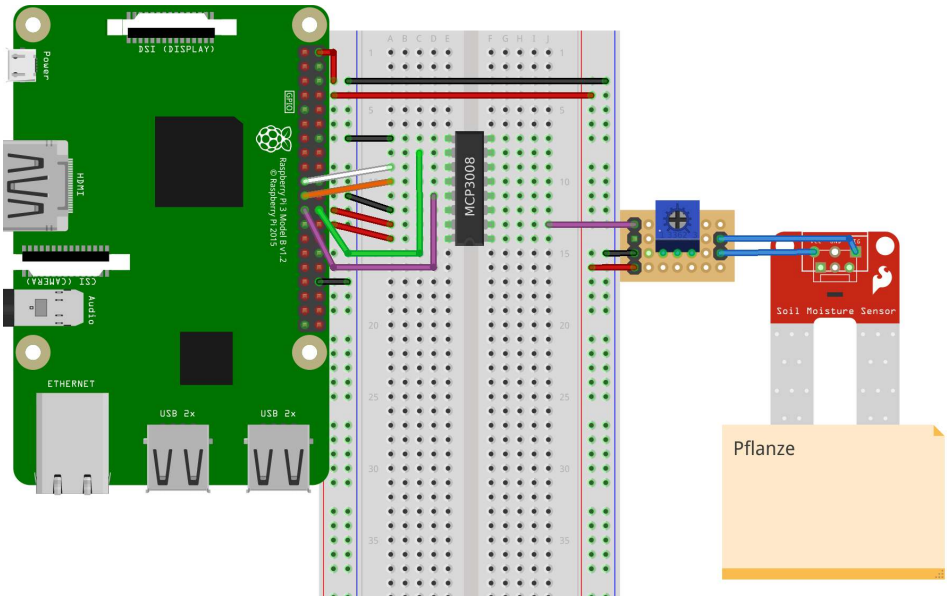
Boden Feuchtigkeitssensor

Gerade wenn wir mit Pflanzen hantieren, ist es wichtig zu wissen, ob sie noch gegossen werden müssen. Das gilt vor allem für Topfpflanzen, die abhängig vom Gießverhalten der Menschen sind. Doch das kann unter Umständen schwierig sein, da man schnell die Übersicht verlieren kann, wann das nächste Mal gewässert werden sollte. Dafür gibt es aber einen Bodenfeuchtigkeitssensor. Während der Luftfeuchtigkeitssensor genau das getan hat - die Feuchtigkeit in der Luft gemessen - macht der Bodenfeutchigkeitssensor genau das Selbe, nur halt für den Boden.



Um die ganze Theorie auch in die Tat umzusetzen, gibt es zwei Beine, die in den Boden gesteckt werden. Da Feuchtigkeit Strom leitet, können wir den

Boden in diesem Fall als Widerstand verstehen. Den entstehenden Widerstandswert können wir messen und wissen dann genau, wann unsere Topfpflanze endlich mal wieder etwas Wasser brauchen könnte.



Anschlüsse Pi	Anschlüsse Sensor
5V	VCC
	Analoges Singal -> CH1
GND	GND

Python

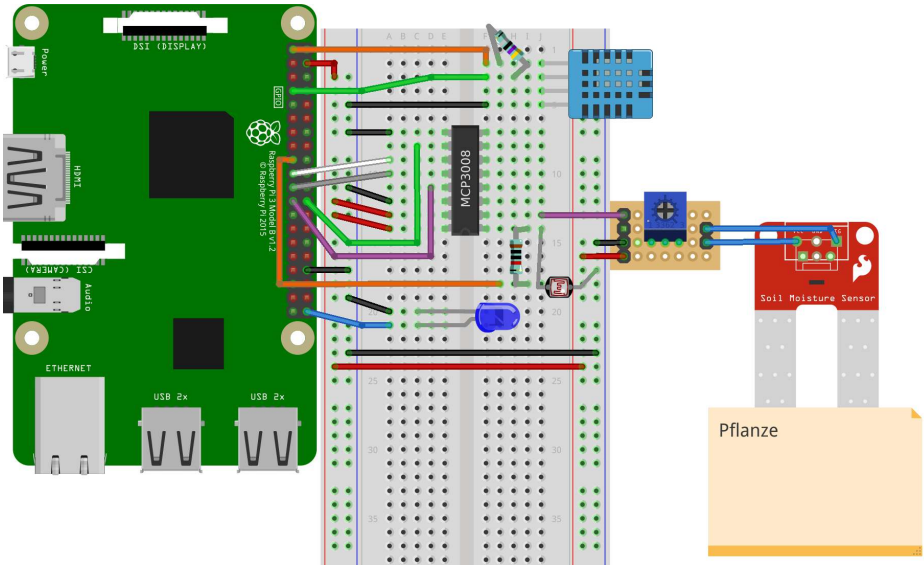
```python
import spidev
import time
import os
import RPi.GPIO as gpio

# SPI Verbindung herstellen
spi = spidev.SpiDev()
spi.open(0,0)
```

```
9
10  # Liest Daten vom MCP3008
11  def analogEingang(channel):
12    adc = spi.xfer2([1,(8+channel)<<4,0])
13    data = ((adc[1]&3) << 8) + adc[2]
14    return data
15
16  while True:
17    print("Feuchtigkeit: "+str(analogEingang(1)))
18    time.sleep(0.2)
```

Das Programm funktioniert ähnlich wie der Photowiderstand. Ein analoger Wert wird ausgelesen und wiedergegeben. Auch hier können die Messwerte sich unterscheiden. Bei einer vertrockneten/sehr trockenen Pflanze kommen wir auf einen Wert von über 900, während wir bei unserer gut gewässerten Topfpflanze einen Wert von ca. 400 messen können. Diese Wert sind natürlich von Pflanze zu Pflanze unterschiedlich.

Die Schlaue Pflanze

Wenn du hier angekommen bist, hast du den MCP3008, den DHT11, den Photowiderstand und den Feuchtigkeitssensor schon behandelt. Alle einzelnen Komponenten führen wir jetzt zusammen. Das Ziel ist es, dass wir nachher mit allen Sensoren die Daten von unserer Pflanze lesen können. Zur Ausgabe haben wir hier noch eine blaue LED angeschlossen. Diese leuchtet immer dann, wenn die Pflanze gegossen werden muss. Damit das Programm regelmäßig die Daten liest, richten wir am Ende dafür noch ein Cronjob an. Als aller erstes müssen wir alle Sensoren anschließen. Am besten machst du das Schritt für Schritt, damit am Ende alles funktioniert.



Anschlüsse Pi	Anschlüsse Sensoren
3,3V	Rote Leisten auf dem Breadboard
GND	Blaue Leisten auf dem Breadboard
3,3V	1. Pin DHT11
GPIO4	2. Pin DHT11
Widerstand 4,7kOhm	Zwischen 1. Pin DHt und 2. Pin DHT
GND	3. Pin DHT11
GND	DGND
GPIO 8N	CS/SHD
GPIO 10	D_IN

→ cw42.de/pflanze/15

GPIO 9	D_OUT
GPIO 11	CLK
GND	AGND
3,3V	V_REF
3,3V	VDD
	3,3V + 10KΩ Widerstand + Seite 1 Photowiderstand
	CH0 + Seite 1 Photowiderstand
GND	Seite 2 Photowiderstand
3,3V	VCC Feuchtigkeitssensor
	Analoges Singal -> CH1
GND	GND Feuchtigkeitssensor
GPIO21	langes Bein Blaue LED
GND	kurzes Bein Blaue LED

schlaue_pflanze.py

Python

```python
import RPi.GPIO as gpio
import Python_DHT
import spidev
import time
import os

# LED
led = 21
gpio.setmode(gpio.BCM)
gpio.setup(led, gpio.OUT)

# DHT11
sensor = Python_DHT.DHT11
pin = 4

# SPI Verbindung herstellen
spi = spidev.SpiDev()
spi.open(0,0)

# Liest Daten vom MCP3008
def analogEingang(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data

def pflanze_messen():
  helligkeit = analogEingang(0)
  bodenFeuchtigkeit = analogEingang(1)

  if bodenFeuchtigkeit > 650:
    print("Du solltest deine Pflanze Waessern")
    gpio.output(led, gpio.HIGH)
  else:
```

```
34        gpio.output(led, gpio.LOW)
35
36      feuchtigkeit, temperatur = Python_DHT.read_retry(se
37      print("==========================")
38      print("Feuchtigkeit: "+str(bodenFeuchtigkeit))
39      print("Helligkeit: "+str(helligkeit))
40      print("Temperatur: "+str(temperatur))
41      print("Feuchtigkeit: "+str( feuchtigkeit)+"%")
42      time.sleep(1.2)
43
44  pflanze_messen()
```

Wenn du das Programm jetzt ausführst, sollte bei einer Pflanze, die noch gegossen werden muss, die Blaue LED aufleuchten. Wenn nichts aufleuchtet, ist alles gut. In Zeile 30 kannst du diesen Grenzwert auch anpassen. Denn dieser ist von Pflanze zu Pflanze unterschiedlich.

Damit wir das Programm nicht immer mit der Hand ausführen müssen, richten wir jetzt noch einen Cronjob ein. Vereinfacht gesagt können wir mit einem Cronjob bestimmte Programme zu bestimmten Zeiten automatisch ausführen lassen.

Bash

```
$ crontab -e
```

```
@hourly \home\pi\schlaue_pflanze.py
```

Weitere Möglichkeiten

Unser Hauptprojekt ist zu Ende. Zusammen mit der LED wissen wir jetzt immer, wann unsere geliebte Pflanze gegossen werden muss. Doch damit sind noch nicht alle Möglichkeiten erschöpft. Was ist, wenn wir die LED mal übersehen? Können wir nicht noch anders mit der Person, die die Pflanze gießen soll, in Kontakt treten?

E-Mails

Eine Möglichkeit ist das Versenden von E-Mails. Auf http://cw42.de/emailVersenden findest du heraus, wie du das machen kannst. Dort könntest du dann auch noch die anderen Messwerte übermitteln.

Display

Wir haben schon öfters mit dem 84 x 48 Zeichen Display gearbeitet und hier gibt es ein perfektes Beispiel. Auf dem Display können die gesamten Messwerte dargestellt werden. Dazu gibt es gerade mit der Zimmertemperatur auch die Möglichkeit, eine Anzeige für die Uhrzeit zu bauen. Sollte dann die Pflanze Wasser brauchen, leuchtet nicht nur die LED, sondern auch die Anzeige auf dem Display kann sich verändern. http://cw42.de/pflanzeDisplay

Twitter Bot

Mit Twitter kannst du schnell kleine Nachrichten von maximal 140 Zeichen in die Welt senden. Wir haben so schon unseren eigenen Ficus twitter.com/jp_norbert online gebracht. Mit einem Cronjob kannst du so die Messwerte von deiner Pflanze posten. http://cw42.de/twitterBot

Auf Wiedersehen

Wir hoffen, dir hat dieses kleine Paket gefallen! Wenn du noch mehr erleben und arbeiten willst, findest du das auf codingworld.io Dort gibt es weitere Projekte und Hintergrundinfos für dein Projekt!