Messen und Steuern mit dem Raspberry Pi

Smart Blumentopf

Felicitas Fuhrmann

2020

Inhalt

[Einleitung 2](#_Toc44529507)

[Raspberry Pi 3](#_Toc44529508)

[Geschichte 3](#_Toc44529509)

[Setup 5](#_Toc44529510)

[Sensoren und Module 6](#_Toc44529511)

[Auswahl der Sensoren und Module für „Smart Blumentopf“ 6](#_Toc44529512)

[Erstes Programm für „Smart Blumentopf“ 7](#_Toc44529513)

[Einbauen des A-D-Wandlers 8](#_Toc44529514)

[Testprogramm für den MCP3008 11](#_Toc44529515)

[Verbesserung der Benutzerfreundlichkeit 12](#_Toc44529516)

[Relais 16](#_Toc44529517)

[Anschließen des Relais an den Raspberry Pi 16](#_Toc44529518)

[Testprogramm für die Pumpe 16](#_Toc44529519)

[Touchscreen 17](#_Toc44529520)

[Anschließen des Touchscreens 17](#_Toc44529521)

[Konzept der GUI (Graphical User Interface) 18](#_Toc44529522)

[Einrichten der GUI 19](#_Toc44529523)

[Einbindung des Sensors und der Pumpe 24](#_Toc44529524)

[Fertiger Code für den „Smart Blumentopf“ 26](#_Toc44529525)

[Autostarten des Programms 32](#_Toc44529526)

[Anwendung in Bezug auf die Realität 33](#_Toc44529527)

[Pro-Argumente 33](#_Toc44529528)

[Contra-Argumente 33](#_Toc44529529)

[Preisüberschlag 34](#_Toc44529530)

[Schwierigkeiten beim „Bauen“ des Smart Blumentopfs 35](#_Toc44529531)

[Fazit 35](#_Toc44529532)

[Quellen 36](#_Toc44529533)

# Einleitung

Ich habe mich entschieden, mein BLL über das Thema „Messen und Steuern mit dem Raspberry Pi“ zu schreiben, da ich in einem Schülerpraktikum bei der Firma ABB einen kleinen Einblick in das Thema bekommen habe. Nun habe ich beschlossen, mein Wissen in dem Bereich zu vertiefen und ein kleines Raspberry Pi Projekt zu starten.

Meine Idee war es, die Bodenfeuchtigkeit in einem Blumentopf zu messen und daraufhin bei entsprechenden Werten Wasser durch eine kleine Pumpe in den Topf zu befördern.

Ein Smart Blumentopf, der eigenständig den Wasserbedarf der Pflanze misst. Dieser bietet eine praktische Methode, die Versorgung der Zimmerpflanzen zu gewährleisten, beispielsweise falls der Besitzer für einen längeren Zeitraum die Pflege nicht bewältigen kann. Allerdings war mir anfangs noch unklar, ob man mit einem Raspberry Pi mehrere Module und Wasserpumpen bedienen kann. Andernfalls wäre das Projekt sehr kostenintensiv und unrentabel, wenn für jede Pflanze ein eigener Raspberry Pi benötigt würde.

Als zusätzliche Ergänzung besteht die Möglichkeit, noch ein Display anzubringen, welches folglich dann Fakten über die Bodenfeuchtigkeit ausgibt.

In dieser besonderen Lernleistung widme ich mich zunächst Fakten und Funktionen des Raspberry Pis und anschließend meiner Herangehensweise zu dem Programmieren und Einbauen der Module. Zusätzlich untersuche ich die Eignung des Programms für den alltäglichen Gebrauch.

Abschließend folgt eine Reflexion über aufgetretene Komplikationen aber auch über Erkenntnisse und Erfolge.

# Raspberry Pi

## Geschichte

Der Raspberry Pi wurde von der Raspberry Pi Foundation im Jahre 2012 veröffentlicht und ist ein Ein-Chip Rechner in Kreditkartengröße.

Der Name kommt zum einen von der Tradition, Rechner nach Früchten zu benennen ("Raspberry"). Der zweite Namensteil "Pi" ist ein Akronym von Python Interpreter, denn ursprünglich sollte der Raspberry Pi einen fest eingebauten Interpreter für Python haben.

Der Raspberry Pi wurde entwickelt, um den Menschen, die nicht über die finanziellen Möglichkeiten verfügen, einen Heimcomputer zu kaufen, einen Zugang zum Internet und Programmieren zu geben. Die Entwickler erhofften sich eine Zunahme an Informatikstudenten, da die Menge an Informatikstudenten in den letzten Jahren eher abgenommen hatte. Damit sich wirklich alle diesen Zugang leisten können, wurde der Verkaufspreis des Raspberry Pis zwischen 5 und 35 US-Dollar festgelegt.

Die meisten Menschen benutzen den Raspberry Pi, um sich alltägliche Aufgaben im Haushalt zu vereinfachen. Er wird als Wetterstation, als Smarthome Steuerung oder auch als Mediacenter verwendet.

Meistens wird der Raspberry Pi mit einer Linux Distribution verwendet, nämlich Raspbian. Mittlerweile gibt es auch schon andere Betriebssysteme für den Raspberry Pi, wie zum Beispiel Windows. Diese Betriebssysteme muss man sich allerdings selbst installieren, da der Raspberry Pi ohne eingebautes Betriebssystem gebaut wird.

Er besitzt mehrere USB-, einen HDMI-, einen LAN- und einen Kopfhörersteckplatz.

Dazu kann auch noch eine Kühlung eingebaut werden, falls man „anstrengende“ Dinge mit dem Raspberry Pi unternehmen möchte. Denn dabei kann dieser relativ schnell warm bzw. heiß werden.

Ein Bild, das Screenshot enthält.

Automatisch generierte BeschreibungAußerdem verfügt er über 40 sogenannte Pins (siehe Abbildung), die alle wichtige Aufgaben erfüllen. Benutzt werden sie, wenn man andere Geräte oder Sensoren an den Raspberry Pi anschließen will.

Sie werden von 1 nach 40 der Leserichtung nach durchnummeriert, wobei man die 1 an der quadratischen Lötstelle auf der Unterseite erkennt, alle anderen Pins haben eine kreisförmige Lötstelle.[[1]](#footnote-2), [[2]](#footnote-3)

## Setup

Da der Raspberry Pi ohne Betriebssystem geliefert wird, muss man sich dies erst selbst auf eine SD-Karte flashen. Dafür muss man lediglich auf der offiziellen Raspberry Pi Website (www.raspberrypi.org) in der Kategorie Downloads das entsprechende Betriebssystem herunterladen. Dort hat man die Wahl zwischen Raspbian und NOOBS. Raspbian ist das Standard Betriebssystem für den Raspberry Pi, muss aber selbstständig installiert werden. Die Anleitung dafür ist auch auf der Webseite zu finden. NOOBS ist die, für Anfänger geeignete, Variante und bietet verschiedene Betriebssysteme, die installiert werden können.

Nachdem man sich für eine Variante entschieden hat und diese heruntergeladen hat, kann man mit einem entsprechenden Programm die Datei auf eine SD-Karte flashen. Wenn man diese in den Raspberry Pi steckt und ihn mit dem Strom verbindet, kann er sich booten.

Danach muss man noch mehrere kleine Setup-Schritte durchgehen. Hierbei muss man die Zeitzone, die Sprache und das Passwort festlegen.

Am Ende kann man seinen Raspberry Pi noch mit dem WLAN verbinden, wenn man kein LAN-Kabel benutzt, um einige Updates herunterzuladen.

Wenn man sich für die Lite Version von Raspbian entschieden hat, fehlen einem noch ein paar Programme, die bei den anderen Betriebssystemen vorhanden gewesen wären. Diese kann man aber einfach über das Terminal herunterladen.

# Sensoren und Module

## Auswahl der Sensoren und Module für „Smart Blumentopf“

Für den Anfang des „Smart Blumentopfs“ braucht man zuerst nur den Raspberry Pi und den Feuchtigkeitssensor. Alle weiteren Teile werden erst später benötigt.

Der Feuchtigkeitssensor besteht aus zwei Teilen, nämlich dem Sensor selbst und einem kleinen zusätzlichen „Control-Panel“ über das der Raspberry Pi und der Sensor verbunden werden. Dafür braucht man insgesamt fünf female-female-jumper Kabel, zwei die die beiden Teile des Sensors verbinden und drei die den Raspberry Pi daran anschließen. Der Sensor wird an drei der 40 vorhandenen Pins des Raspberry Pis angeschlossen, daher wird er wie folgt verbunden:

* Das erste Kabel wird mit dem 1. Pin verbunden, welcher 3,3 Volt an den Sensor liefern soll.
* Das zweite wird mit einem beliebigen Masse/Ground Pin verbunden, z.B. 6. Sie sind alle miteinander verbunden, daher spielt es keine Rolle, welchen man verwendet. Es fungiert als Minuspol.
* Das letzte Kabel wird mit dem 40. Pin verbunden. Dieser übernimmt die Rolle der Datenübertragung.

Jedoch müssen diese Kabel auch mit den richtigen Stellen am Sensor verbunden werden. Das heißt der VCC-Port wird mit dem Strom Pin, der GND-Port wird mit dem Masse/Ground Pin und DO(Digital Output)-Port wird mit dem Daten Pin verbunden.

Wenn man nun den Raspberry Pi mit dem Strom verbindet, sieht man, dass eine, in meinem Fall, rote Kontrollleuchte leuchtet und anzeigt, dass der Sensor Strom bekommt. Der nächste Schritt wäre es, den Sensor zu testen. Dazu nimmt man am besten ein Glass voll Wasser und hängt den Sensor hinein. Nun sollte eine zweite, grüne, LED leuchten, die angibt, dass Feuchtigkeit gemessen wird.

## Erstes Programm für „Smart Blumentopf“

Das Programm sollte natürlich zunächst die empfangenen Daten verarbeiten und ausgeben können und dann entsprechend einen Vorgang ablaufen lassen. Da ich aber im Moment nur den Sensor ausprobieren möchte, ist dies noch nicht nötig und der Sensor wird lediglich ausgelesen.

Um ein erstes Testprogramm zu schreiben, benötigt man Python.

Wenn es nicht bereits mit dem Betriebssystem installiert wurde, kann man es einfach über das Terminal mit folgendem Befehl installieren:

sudo apt-get install python2

Nachdem Python installiert ist, kann man das erste Programm zum Testen schreiben. Hier gibt es zwei Möglichkeiten: Entweder man schreibt das Programm direkt im Terminal, oder man schreibt es in einer Entwicklungsumgebung, wie zum Beispiel Thonny.

Da ich hier nur testen möchte, ob die Verkabelung vom Raspberry Pi und dem Modul funktioniert, kommt es noch nicht auf Benutzerfreundlichkeit und Komplexität des Programms an, sondern es soll lediglich den Feuchtigkeitswert ausgeben, der während der Ausführung des Programms messbar ist.

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD) #Setzt die Art wie die Pins gezählt werden

GPIO.setup(40, GPIO.IN) #Setzt den 40ten Pin als Input fest

if GPIO.input(40) == GPIO.LOW: #Wenn der Pin als Input eine 1 bzw.…

    print(“Es ist feucht.”) #…high bekommt, wird dies ausgegeben

else:

    print(“Es ist nicht feucht.”) #Wenn nicht, wird dies ausgegeben

Hier sollten im Idealfall zwei unterschiedliche Werte herauskommen, wenn man den Sensor einmal ins Wasser hält und einmal nur in die Luft.

Dieser Code gibt nur Ja/Nein-Werte aus, die zwar auch nützlich sind, jedoch für das Ziel des „Smart Blumentopfs“ nicht benutzt werden können. Zum Ansteuern eine Pumpe sind diese Ja/Nein-Werte ungenügend, da bei der Ermittlung der Bewässerungsnotwendigkeit differenzierte Feuchtigkeitsgrade zugrunde liegen. Dies macht das Auslesen von Prozentwerten erforderlich.

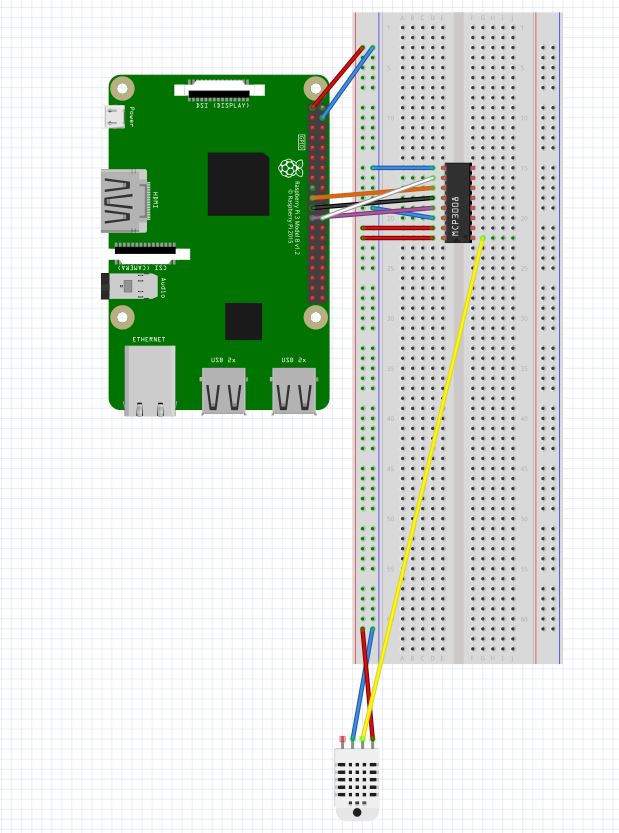
## Einbauen des A-D-Wandlers

Jetzt wird ein A-D-Wandler (analog zu digital) eingebaut, dieser kann dem Raspberry Pi die analogen Messwerte des Sensors übersetzen, sodass der Raspberry Pi sie auch versteht.

Um den A-D-Wandler einzubauen, benötigt man ein Breadboard. Dies ermöglicht es, mehrere Kabel an einem Ziel zusammenlaufen zu lassen. Natürlich kann man alles auch zusammenlöten, dies sollte man aber nur tun, wenn man sich sicher ist, dass alles funktioniert.

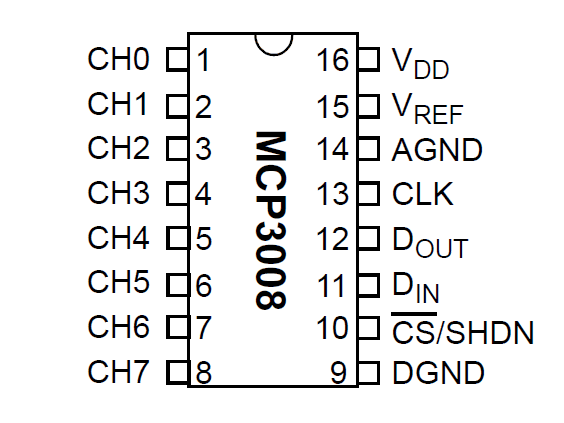
Die beiden roten bzw. blauen Spalten rechts und links sind für den Stromkreislauf. Jede Spalte ist von oben nach unten verbunden, das heißt, man kann an einer beliebigen Stelle Strom abnehmen, wenn er vorher in der Spalte schon angeschlossen wurde.

Die Mitte des Breadboards ist reihenweise aufgebaut. Hier sind die Reihen jeweils bis zur Mitte verbunden, aber dafür nicht nach oben oder unten.

 Der A-D-Wandler wird so auf das Breadboard gesteckt, dass er die Mitte überdeckt. Außerdem wird er mit seinem oberen Ende nach unten gesteckt, was die Verkabelung zwischen ihm und dem Raspberry Pi vereinfacht.

Seine Daten bekommt der MCP3008 vom Sensor. Die beiden sind über das gelbe Kabel verbunden. Diese Daten teilt der MCP3008 dem Raspberry Pi durch die SPI-Schnittstelle mit.

Die SPI-Schnittstelle oder auch Serial Peripheral Interface funktioniert durch ein Master-Slave-Prinzip, durch welches der Raspberry PI (Master) und der MCP3008 (Slave) miteinander kommunizieren. Dafür benötigen die beiden vier Kabel.

1. Das Chip-Select-Kabel: Dieses Kabel wird gebraucht, damit der Raspberry Pi weiß, mit welchem Gerät er kommunizieren soll, da es theoretisch die Möglichkeit gibt, mehrere Geräte anzuschließen. Dieses wird vom 24. Pin des Raspberry Pis zum 10. Pin des MCP3008 verbunden.
2. Das Clock-Kabel: Es wird benutzt, damit Master und Slave beide den gleichen Rhythmus bei der Kommunikation haben. Sie schieben abwechselnd Bits hin und her, und wenn sie das nicht im gleichen Takt machen, werden die Daten nicht vollständig übertragen. Das Kabel wird am 23. Pin des Raspberry Pi und am 13. Pin des MCP3008 angeschlossen.
3. Das MISO-Kabel (Master Input Slave Output):

Der Slave gibt Werte aus und der Master liest diese. Das Kabel verbindet den 21. Pin des Raspberry Pis mit dem 12. Pin des MCP3008.

1. Das MOSI-Kabel (Master Output Slave Input):

Der Master gibt Werte aus, die der Slave lesen kann. Das Kabel verläuft vom 19. Pin des Raspberry Pis zum 11. Pin des MCP3008. [[3]](#footnote-4)3

MCP3008 Pins

Zusätzlich benötigt der MCP3008 noch zwei Stromverbindungen und zwei Grundierungen. VREF wird als Referenzspannung benutzt, denn der MCP3008 gibt Prozentwerte aus, wobei VREF 100% ist, das heißt in diesem Fall entspricht 3,3V 100% und 0V 0%.

VDD liefert den Strom, den der MCP3008 braucht. Die beiden Grundierungen sind einmal für die analoge und einmal für die digitale Schaltung. Diese werden getrennt grundiert, um Schaltkreisfehler zu vermeiden. [[4]](#footnote-5)

## Testprogramm für den MCP3008

Der Raspberry Pi braucht die Administrator-Rechte, um auf die im folgenden Abschnitt erwähnten Libraries zugreifen zu können.

Deshalb kann man mit dem Befehl:

chmod 777 auslesenMCP3008.py

die Lese-, Schreib- und Ausführrechte ändern. Dieser Befehl gibt jedem Benutzer komplette Rechte auf die Datei.

Um den MCP3008 auslesen zu können müssen noch „libraries“ installiert werden, in denen der Raspberry Pi abrufen kann, wie er den MCP3008 auslesen soll und in welchem Rhythmus deren Kommunikation stattfindet. Um diese Libraries zu installieren, sind folgende Befehle erforderlich, die im Terminal eingegeben werden müssen:

sudo apt-get update

sudo apt-get install build-essentials python-dev python-smbus git

git clone https://github.com/adafruit/Adafruit\_Phython\_MCP3008.git

cd Adafruit\_Python\_MCP3008

sudo python setup.py install

Diese Befehle bringen den Raspberry Pi dazu, aus dem Internet die Adafruit Library herunterzuladen und zu installieren. Diese muss in das Programm importiert werden.

Dem Raspberry Pi muss noch durch Variablen mitgeteilt werden, an welcher Stelle der MCP3008 angeschlossen ist, da theoretisch mehrere Geräte angesprochen werden können. Dann kann man das eigentliche Programm schreiben. Dort muss man nur noch den MCP3008 auslesen und den Wert ausgeben. Das ganze Programm kann in eine Schleife geschrieben werden, sodass die Werte in einem bestimmten Rhythmus ausgegeben werden.

import time

import Adafruit\_GPIO.SPI as SPI

import Adafruit\_MCP3008

SPI\_PORT = 0  # Sagt dem Raspberry Pi an welcher Stelle der…

SPI\_DEVICE = 0  # …MCP3008 angeschlossen ist.

mcp = Adafruit\_MCP3008.MCP3008(spi=SPI.SpiDev(SPI\_PORT, SPI\_DEVICE))

while True:  # Endlos-Schleife

    value = 0

    value = mcp.read\_adc(0)  # Liest den MCP3008 aus.

    print(value)  # Gibt den Messwert aus.

    time.sleep(1)  # Jede Sekunde wird die Schleife durchlaufen.

# Dieser Wert ist frei wählbar und kann je nach Situation angepasst werden.

## 

## Verbesserung der Benutzerfreundlichkeit

Da der ausgegebene Wert zwischen 0 bei totaler Feuchtigkeit und 1023 bei totaler Trockenheit liegt, wäre es benutzerfreundlicher, wenn das Programm Werte zwischen 0 und 100 ausgeben würde. Außerdem wäre es von Vorteil, wenn totale Feuchtigkeit bei 100% und Trockenheit bei 0% liegen würde.

Um dies zu realisieren, müssen einige Änderungen an dem bereits vorliegenden Code vorgenommen werden. Zuerst müssen die ausgegeben Werte in kleinere Zahlen von eins bis 100 umgewandelt werden. Hierzu teilt man einfach durch 10,23. Um die Werte „umzudrehen“, subtrahiert man sie von 100. Dadurch entstehen teilweise Zahlen mit sehr vielen Nachkommastellen, die aus Gründen der Übersichtlichkeit auf 3 Nachkommastellen gerundet werden.

import time

import Adafruit\_GPIO.SPI as SPI

import Adafruit\_MCP3008

SPI\_PORT = 0

SPI\_DEVICE = 0

mcp = Adafruit\_MCP3008.MCP3008(spi=SPI.SpiDev(SPI\_PORT, SPI\_DEVICE))

while True:

    value = 0

    value = mcp.read\_adc(0)

    #Teilt den Wert, um auf Werte zwischen 0 und 100 zu kommen.

    #Danach wird der Wert gedreht und gerundet.

    print(format((100-value/10,23), ".3f"))

    time.sleep(1)

Als nächstes werden die Werte in einer Art Tabelle ausgegeben, sodass die Ausgabe überschaubarer wird. Zusätzlich kann der Rhythmus an den Benutzer angepasst werden, da momentan jede Sekunde ein Wert ausgegeben wird. Dies ist im realistischen Betrieb mit einer Pflanze eher überflüssig. Um die Bildschirmausgabe übersichtlicher zu gestalten werden einige leere Zeilen hinzugefügt, damit die verschiedenen Zeilen und die Tabelle nicht zu eng beieinander sind.

import time

import Adafruit\_GPIO.SPI as SPI

import Adafruit\_MCP3008

SPI\_PORT = 0

SPI\_DEVICE = 0

mcp = Adafruit\_MCP3008.Mcp3008(spi=SPI.SpiDev(SPI\_PORT, SPI\_DEVICE))

#Fragt nach, in welchem Rhythmus die Schleife durchlaufen werden soll.

rhythm = input("In welchem Rhythmus sollen die Werte ausgelesen werden?(Sekunden)")

print(" ")   #Erzeugt eine leere Zeile. -> Übersichtlichkeit

print("Lese die Feuchtigkeitswerte aus. Drücke Str-C zum Beenden.")

print(" ")

print("| Feuchtigkeit |")  # Überschrift einer Tabelle

print("-" \* 14)  # Erzeugt eine Linie.

while True:

    value = mcp.read\_adc(1)

    valueberechnet = 100-value/10.23

    output = ("| {0: < 10.3f} |")

    # Gibt die Werte in Form einer Tabelle wieder aus.

    print(output.format(valueberechnet))

    time.sleep(rhythm)

Jetzt ist es möglich, die Ausgabetabelle noch zu erweitern, um einem Benutzer zu signalisieren, in welchem Zustand die Pflanze sich befindet. Der reine Prozentwert wird um ein Kriterium wie z.B. „Feucht“ erweitert.

Zusätzlich dazu wäre es auch hilfreich, wenn man die Pflanzen kategorisieren würde, je nachdem wie groß der Wasserbedarf der Pflanze ist, sodass das Programm pflanzenspezifische Werte ausgeben kann. Hierzu habe ich 3 Kategorien festgelegt:

1. Die Pflanze braucht durchgehend nasse/feuchte Erde.
2. Die Pflanze verkraftet über kurze Zeit auch trockene Erde.
3. Die Pflanze verkraftet trockene Erde über lange Zeit.[[5]](#footnote-6)

Diese Kategorien müssen noch in das Programm eingearbeitet werden, am besten so, dass der Benutzer am Anfang wählen kann, was für eine Pflanze er hat. Hierfür wird der Code in drei Anschnitte (if-Schleifen) aufgeteilt. Diese haben zwar jetzt direkt noch keine Wirkung, aber sobald die Pumpe angeschlossen wird, kann man die Pflanzen direkt unterschiedlich bewässern und muss den Code später nicht noch umschreiben.

# -\*- coding: iso-8859-1 -\*-

import time

import Adafruit\_GPIO.SPI as SPI

import Adafruit\_MCP3008

SPI\_PORT = 0

SPI\_DEVICE = 0

mcp = Adafruit\_MCP3008.Mcp3008(spi=SPI.SpiDev(SPI\_PORT, SPI\_DEVICE))

# Lässt den Benutzer wählen, wie viel Feuchtigkeit seine Pflanze braucht.

plant = raw\_input(

    "Um welche Pflanze handelt es sich? A: Braucht viel Feuchtigkeit; B: Braucht mäßige Feuchtigkeit; C: Braucht kaum Feuchtigkeit")

if plant == "A" or plant == "a":

    rhythm = input(

        "In welchem Rhythmus sollen die Werte ausgelesen werden?(Sekunden)")

    print(" ")

    print("Lese die Feuchtigkeitswerte aus. Drücke Str-C zum Beenden.")

    print(" ")

    print("|Feuchtigkeit| |Status|")

    print("-" \* 23)

    while True:

        value = mcp.read\_adc(1)

        valueberechnet = 100-value/10.23

        output = "| {0:<12.3f} | |{1:<7}|"

        # Wenn die Werte die bestimmten Bereiche überschreiten, werden der Messwert + Hilfswort ausgegeben

        if valueberechnet > 80.00:

            print(output.format(valueberechnet, "Feucht"))

        if valueberechnet < 80.00 and valueberechnet > 60.00:

            print(output.format(valueberechnet, "Mittel"))

        if valueberechnet < 60.00:

            print(output.format(valueberechnet, "Trocken"))

        time.sleep(rhythm)

if plant == "B" or plant == "b":

    rhythm = input(

        "In welchem Rhythmus sollen die Werte ausgelesen werden?(Sekunden)")

    print(" ")

    print("Lese die Feuchtigkeitswerte aus. Drücke Str-C zum Beenden.")

    print(" ")

    print("|Feuchtigkeit| |Status|")

    print("-" \* 23)

    while True:

        value = mcp.read\_adc(1)

        valueberechnet = 100-value/10.23

        output = "| {0:<12.3f} | |{1:<7}|"

        # Wenn die Werte die bestimmten Bereiche überschreiten, werden der Messwert + Hilfswort ausgegeben

        if valueberechnet > 60.00:

            print(output.format(valueberechnet, "Feucht"))

        if valueberechnet < 60.00 and valueberechnet > 40.00:

            print(output.format(valueberechnet, "Mittel"))

        if valueberechnet < 40.00:

            print(output.format(valueberechnet, "Trocken"))

        time.sleep(rhythm)

if plant == "C" or plant == "c":

    rhythm = input(

        "In welchem Rhythmus sollen die Werte ausgelesen werden?(Sekunden)")

    print(" ")

    print("Lese die Feuchtigkeitswerte aus. Drücke Str-C zum Beenden.")

    print(" ")

    print("|Feuchtigkeit| |Status|")

    print("-" \* 23)

    while True:

        value = mcp.read\_adc(1)

        valueberechnet = 100-value/10.23

        output = "| {0:<12.3f} | |{1:<7}|"

        # Wenn die Werte die bestimmten Bereiche überschreiten, werden der Messwert + Hilfswort ausgegeben

        if valueberechnet > 35.00:

            print(output.format(valueberechnet, "Feucht"))

        if valueberechnet < 35.00 and valueberechnet > 15.00:

            print(output.format(valueberechnet, "Mittel"))

        if valueberechnet < 5.00:

            print(output.format(valueberechnet, "Trocken"))

        time.sleep(rhythm)

# Relais

## Anschließen des Relais an den Raspberry Pi

Ein Relais ist eine Art Schalter, der einen Laststromkreis öffnet und schließt, wenn er ein entsprechendes Signal bekommt, in diesem Falle vom Raspberry Pi.

Nun laufen 2 Stromkreise nebeneinander ab. Der Raspberry Pi liefert dem Relais über den Steuerstromkreis Signale, und gleichzeitig verbindet das Relais eine Wasserpumpe und eine externe Stromquelle. Das heißt, sobald der Raspberry Pi ein Signal ausgibt, schließt das Relais den Laststromkreis und die Pumpe wird aktiviert.

Um das Relais anzuschließen, werden mehrere Kabel benötigt. Zum einen drei female-female-jumper-Kabel, mit denen das Relais einmal 5 Volt bekommt (2. Pin am Raspberry Pi), einmal grundiert wird (9. Pin am Raspberry Pi) und einmal die Signale bekommt (3. Pin am Raspberry Pi). Diese Kabel werden dann mit den dementsprechenden Pins am Relais verbunden. 5 Volt mit VCC, Masse mit GND und Signal mit IN1.

Auf der hinteren Seite des Relais kann man dann den Stromkreis anbringen. Hier wird in die erste vorgegebene Öffnung das Stromkabel geschraubt und in die Zweite das Kabel, welches zur Pumpe führt.[[6]](#footnote-7)

Ich benutze hierfür eine 12V Tauchpumpe, die sich in einem großen Eimer mit Wasser befindet und dieses bei Einschaltung nach oben pumpt. Die Pumpe bekommt den Strom von einem 12V Gleichstromnetzteil.

## Testprogramm für die Pumpe

Um die Funktionalität des Relais zu testen, kann man ein kleines Testprogramm schreiben. In diesem wird einmal die Pumpe gestartet und dann wieder gestoppt.

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(3, GPIO.OUT)

def PumpeAn(pin):

    # Schaltet die Pumpe an, indem der Pin auf high gesetzt wird

    GPIO.output(pin, GPIO.HIGH)

def PumpeAus(pin):

    # Schaltet die Pumpe aus, indem der Pin auf low gesetzt wird

    GPIO.output(pin, GPIO.LOW)

PumpeAn(3)  # Schaltet die Pumpe an…

PumpeAus(3)  # …und schaltet die Pumpe wieder aus

Dieses grobe Prinzip muss noch in den Code eingebaut werden. Dafür muss man sich zwei Sachen verdeutlichen. Zum einen bei wie viel Trockenheit die Pflanze bewässert werden soll und zum anderen wie viel Wasser sie braucht.

Pflanzentyp 1, der durchgehend nasse/feuchte Erde braucht, sollte bei 60% Feuchtigkeit bewässert werden.

Pflanzentyp 2, der auch mal trockene Erde verkraftete, sollte bei 40% Feuchtigkeit bewässert werden.

Pflanzentyp 3, der über lange Zeit trockene Erde verkraftet, sollte bei 5% Feuchtigkeit bewässert werden.

# Touchscreen

Mein Ziel ist es noch einen Touchscreen einzubauen, damit der „Smart Blumentopf“ auch getrennt von Maus und Tastatur bedienbar ist.

Damit dies realisierbar ist, muss eine grafische Oberfläche programmiert werden. Auf dieser soll alles auswählbar sein, was für den Betrieb erforderlich ist. Also die Menge an Wasser sowie der Rhythmus und die Pflanzenart.

## Anschließen des Touchscreens

Der Touchscreen wird mit einem Flachbandkabel an den Raspberry Pi angeschlossen, von dem er auch den Strom (5V) bekommt.

## Konzept der GUI (Graphical User Interface)

Ungefähre Abbildung der GUI

Die GUI ist in 7 zentrale Teile unterteilt. Diese sind alle durch ein Frame definiert, welches ihren ungefähren Platz bestimmt. Jedes dieser Frames erfüllt eine andere Aufgabe (von oben links nach unten rechts):

1. Das erste Frame dient zur Erklärung des Programms.
2. Im 2. Frame kann man die Feuchtigkeit der letzten Messung ablesen und eine Messung durchführen.
3. Im 3. Frame kann man die Pflanzenart wählen.
4. Das 4. Frame lässt den Benutzer den Rhythmus, in dem die Schleife durchlaufen wird, bestimmen.
5. Im 5. Frame kann man die Menge an Wasser, die bei einer Wässerung benutzt werden soll, einstellen.
6. Im 6. Frame gibt es einen Knopf, mit dem man die Pflanze manuell bewässern kann.
7. Das 7. Frame ist eine Art „Kontrollframe“. Hier kann man den Raspberry Pi herunterfahren und/oder die Schleife des Messens und Bewässerns starten.

## Einrichten der GUI

Bevor man die GUI jedoch programmieren kann, muss man sich die Tkinter library herunterladen. Dazu öffnet man einfach das Terminal und gibt diesen Befehl ein:

sudo apt-get install python python-tk idle python-pmw python-imaging –yes

Nachdem der Raspberry Pi alles heruntergeladen hat, kann man mit:

chmod 777 gui.py

die Rechte für das Programm ändern. Dann kann man mit der Python Idle das Programm schreiben.[[7]](#footnote-8)

# -\*- coding: iso-8859-1 -\*-

import Tkinter as Tk

class GUI(object):

    def \_\_init\_\_(self):

        # Erstellt ein Fenster mit dem Namen Smart Blumentopf, welches unter der Variabel self.root bekannt ist.

        self.root = Tk.Tk()

        # Das Fenster bekommt den Titel Smart Blumentopf.

        self.root.wm\_title("Smart Blumentopf")

        # Zusätzlich bekommt es noch grün als Hintergrundfarbe…

        self.root.configure(bg="#137547")

        # …und die Bildschirm-Maße des Raspberry-Pis.

        self.root.geometry("800x440")

    def run(self):

        # Erstellt ein Frame in der 1. Reihe und über 2 Spalten

        self.ErklaerungsFrame = Tk.Frame(

            self.root, width=500, height=50, bg="#137547")

        self.ErklaerungsFrame.grid(row=0, column=0, columnspan=2, padx=25)

        # Erstellt ein Label mit Text

        self.Erklaerungstext = Tk.Label(self.ErklaerungsFrame, bg="#137547",

                                        text="Wählen Sie zuerst ihre Werte an den Schiebereglern aus. \n Drücken Sie auf Starten, wenn Sie fertig sind.",

                                        font=14, fg="white")

        self.Erklaerungstext.grid(row=0, column=0)

        # Erstellt ein Frame in der 2. Reihe und der 1. Spalte

        self.FeuchteFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.FeuchteFrame.grid(row=1, column=0, padx=35, pady=10)

        # Erstellt ein Label mit Text

        self.momFeuchte = Tk.Label(self.FeuchteFrame, bg="#137547",

                                   text="\n Feuchtigkeit bei der letzten Messung:", font=14, fg="white")

        self.momFeuchte.grid(row=0, column=0)

        # Erstellt ein weiteres Label, das durch die Messwerte ersetzt werden soll

        self.momFeuchteWert = Tk.Label(

            self.FeuchteFrame, bg="#137547", text="0 \n \n", font=14, fg="white")

        self.momFeuchteWert.grid(row=1, column=0)

        # Erstellt einen Knopf mit dem man direkt messen kann

        self.Messen = Tk.Button(self.FeuchteFrame, text="Jetzt Messen.",

                                bg="#1EBB72", width=25, command=self.messen, font=14)

        self.Messen.grid(row=2, column=0)

        # Erstellt ein Frame in der 2. Reihe  und der 2. Spalte

        self.PflanzenFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.PflanzenFrame.grid(row=1, column=1, padx=35, pady=10)

        # Erstellt ein Label mit Text

        self.Art = Tk.Label(self.PflanzenFrame, bg="#137547",

                            text="\n Wie viel Wasser braucht Ihre Pflanze: \n (1 = viel; 2 = mittel; 3 = wenig) \n \n", font=14, fg="white")

        self.Art.grid(row=0, column=0)

        # Erstellt einen Scheiberegler mit dem man seine Pflanzenart wählen kann

        self.PflanzeSlider = Tk.Scale(self.PflanzenFrame, from\_=1, to=3, resolution=1, length=320,

                                      orient="horizontal", font=14, bg="#1EBB72", highlightbackground="#1EBB72")

        self.PflanzeSlider.grid(row=1, column=0)

        # Erstellt ein Feld in der 3. Reihe und der 1. Spalte

        self.FeuchteMessenFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.FeuchteMessenFrame.grid(row=2, column=0, padx=35, pady=10)

        # Erstellt ein Label mit Text

        self.FeuchteRhythmus = Tk.Label(self.FeuchteMessenFrame, bg="#137547",

                                        text="Rhythmus der Messungen (in Minuten): \n", font=14, fg="white")

        self.FeuchteRhythmus.grid(row=0, column=0, padx=10)

        # Erstellt einen Schieberegler mit dem man den Rhythmus wählen kann, mit dem die Schleife durchlaufen wird

        self.RhythmusInput = Tk.Scale(self.FeuchteMessenFrame, from\_=1, to=300, resolution=1,

                                      length=320, orient="horizontal", font=14, bg="#1EBB72", highlightbackground="#1EBB72")

        self.RhythmusInput.grid(row=1, column=0)

        # Erstellt ein Frame in der 3.Reihe und der 2. Spalte

        self.MengeFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.MengeFrame.grid(row=2, column=1, padx=35, pady=10)

        # Erstellt ein Label mit Text

        self.Menge = Tk.Label(self.MengeFrame, bg="#137547",

                              text="Menge an Wasser (in ml): \n", font=14, fg="white")

        self.Menge.grid(row=0, column=0)

        # Erstellt ein Schieberegler, an welchem man die Menge an Wasser wählen kann

        self.MengeW = Tk.Scale(self.MengeFrame, from\_=100, to=1000, resolution=100, length=320,

                               orient="horizontal", font=14, bg="#1EBB72", highlightbackground="#1EBB72")

        self.MengeW.grid(row=1, column=0)

        # Erstellt ein Frame in der 4. Reihe und 1. Spalte

        self.WasserFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.WasserFrame.grid(row=3, column=0, padx=35, pady=10)

        # Erstellt einen Knopf, mit dem man die Pflanze wässern kann

        self.Wasser = Tk.Button(self.WasserFrame, text="Jetzt Waessern.",

                                bg="#1EBB72", width=25, command=self.waessern, font=14)

        self.Wasser.grid(row=0, column=0)

        # Erstellt ein Frame in der 4. Reihe und der 2. Spalte

        self.SteuerFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.SteuerFrame.grid(row=3, column=1, padx=35, pady=10)

        # Erstellt einen Knopf, mit dem man die Schleife starten kann

        self.StartStopp = Tk.Button(self.SteuerFrame, text="Starten",

                                    bg="#1EBB72", width=25, command=self.startstoppen, font=14)

        self.StartStopp.grid(row=0, column=0, pady=5)

        # Erstellt einen Knopf, mit dem man den Raspberry Pi herunterfahren kann

        self.Abbruch = Tk.Button(self.SteuerFrame, text="Herunterfahren",

                                 bg="#1EBB72", width=25, command=self.abbruch, font=14)

        self.Abbruch.grid(row=1, column=0, pady=5)

        # Die Schleife des Fensters

        self.root.mainloop()

    def messen(self):

        pass

    def waessern(self):

        pass

    def abbruch(self):

        pass

    def wassermenge(self):

        pass

    def messrhythmus(self):

        pass

    def blumentopf(self):

        pass

    def startstoppen(self):

        pass

    def Pflanzenwahl(self):

        pass

# Erstellt alle Frames, Labels und Knöpfe

GUI().run()

# Einbindung des Sensors und der Pumpe

Um den Code so zu erweitern, dass er alle Aufgaben eigenständig erfüllt, müssen wieder die erforderlichen Module importiert werden. Genauso wie in den vorherigen Programmen.

import Tkinter as Tk

import time

import Adafruit\_GPIO.SPI as SPI

import Adafruit\_MCP3008

import RPi.GPIO as GPIO

import os

SPI\_PORT = 0

SPI\_DEVICE = 0

mcp = Adafruit\_MCP3008.MCP3008(spi=SPI.SpiDev(SPI\_PORT, SPI\_DEVICE))

GPIO.setmode(GPIO.BOARD)

GPIO.setup(3, GPIO.OUT)

Dann müssen den Funktionen, die unten im Code stehen, noch Aktionen zugewiesen werden, da diese bis jetzt noch nichts bewirken.

 def PumpeAn(self, pin):

        GPIO.output(pin, GPIO.HIGH)

    def PumpeAus(self, pin):

        GPIO.output(pin, GPIO.LOW)

    def messen(self):

        value = 0

        value = mcp.read\_adc(1)

        valuegerundet = 100-(value/10.23)

        self.momFeuchteWert.config(text=str(valuegerundet)+"\n")

        return valuegerundet

    def waessern(self):

        self.PumpeAn(3)

        Pumpenzeit = self.wassermenge()

        time.sleep(Pumpenzeit)

        self.PumpeAus(3)

    def abbruch(self):

        os.system("sudo shutdown now")

    def wassermenge(self):

        Menge = self.MengeW.get()

        Pumpenzeit = Menge/200

        return Pumpenzeit

    def messrhythmus(self):

        rhythm = self.RhythmusInput.get()

        minutenrhythm = rhythm / 60

        return minutenrhythm

    def blumentopf(self):

        Pflanzenart = self.Pflanzenwahl()

        rhythm = self.messrhythmus()

        if Pflanzenart == 1:

            self.messen()

            valuegerundet = self.messen()

            if valuegerundet <= 60:

                self.waessern()

        if Pflanzenart == 2:

            self.messen()

            valuegerundet = self.messen()

            if valuegerundet <= 40:

                self.waessern()

        if Pflanzenart == 3:

            self.messen()

            valuegerundet = self.messen()

            if valuegerundet <= 5:

                self.waessern()

        if self.running:

            self.root.after(rhythm, self.blumentopf)

    def startstoppen(self):

        if self.running == True:

            self.StartStopp["text"] = "Starten"

            self.running = False

        else:

            self.StartStopp["text"] = "Stoppen"

            self.running = True

            self.blumentopf()

    def Pflanzenwahl(self):

        Pflanzenart = self.PflanzeSlider.get()

        return Pflanzenart

Als letztes muss noch ganz oben beim Erstellen des Fensters eine Zeile ergänzt werden, sodass diese dann so aussieht:

class GUI(object):

    def \_\_init\_\_(self):

        self.root = Tk.Tk()

        self.running = False

        self.root.wm\_title("Smart Blumentopf")

        self.root.configure(bg="#137547")

Wenn man diese einzelnen Programmabschnitte jetzt zusammensetzt, erhält man den kompletten fertigen Code für den „Smart Blumentopf“.

# Fertiger Code für den „Smart Blumentopf“

# -\*- coding: iso-8859-1 -\*-

import Tkinter as Tk

import time

import Adafruit\_GPIO.SPI as SPI

import Adafruit\_MCP3008

import RPi.GPIO as GPIO

import os

SPI\_PORT = 0

SPI\_DEVICE = 0

mcp = Adafruit\_MCP3008.MCP3008(spi=SPI.SpiDev(SPI\_PORT, SPI\_DEVICE))

GPIO.setmode(GPIO.BOARD)

GPIO.setup(3, GPIO.OUT)

class GUI(object):

    def \_\_init\_\_(self):

        self.root = Tk.Tk()

        self.running = False

        self.root.wm\_title("Smart Blumentopf")

        self.root.configure(bg="#137547")

    def run(self):

        self.ErklaerungsFrame = Tk.Frame(

            self.root, width=500, height=50, bg="#137547")

        self.ErklaerungsFrame.grid(row=0, column=0, columnspan=2)

        self.Erklaerungstext = Tk.Label(self.ErklaerungsFrame, bg="#137547",

                                        text="Wählen Sie zuerst ihre Werte an den Schiebereglern aus. \n Drücken Sie auf starten, wenn Sie fertig sind.", font=14, fg="white")

        self.Erklaerungstext.grid(row=0, column=0)

        self.FeuchteFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.FeuchteFrame.grid(row=1, column=0, padx=10, pady=10)

        self.momFeuchte = Tk.Label(self.FeuchteFrame, bg="#137547",

                                   text="\n Feuchtigkeit bei der letzten Messung:", font=14, fg="white")

        self.momFeuchte.grid(row=0, column=0)

        self.momFeuchteWert = Tk.Label(

            self.FeuchteFrame, bg="#137547", text="0 \n \n", font=14, fg="white")

        self.momFeuchteWert.grid(row=1, column=0)

        self.Messen = Tk.Button(self.FeuchteFrame, text="Jetzt Messen.",

                                bg="#1EBB72", width=25, command=self.messen, font=14)

        self.Messen.grid(row=2, column=0)

        self.MengeFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.MengeFrame.grid(row=2, column=1, padx=10, pady=10)

        self.Menge = Tk.Label(self.MengeFrame, bg="#137547",

                              text="Menge an Wasser (in ml): \n \n", font=14, fg="white")

        self.Menge.grid(row=0, column=0)

        self.MengeW = Tk.Scale(self.MengeFrame, from\_=100, to=1000, resolution=100, length=320,

                               orient="horizontal", font=14, bg="#1EBB72", highlightbackground="#1EBB72")

        self.MengeW.grid(row=1, column=0)

        self.FeuchteMessenFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.FeuchteMessenFrame.grid(row=2, column=0, padx=10, pady=10)

        self.FeuchteRhythmus = Tk.Label(self.FeuchteMessenFrame, bg="#137547",

                                        text="Rhythmus der Messungen (in Minuten): \n \n", font=14, fg="white")

        self.FeuchteRhythmus.grid(row=0, column=0, padx=10)

        self.RhythmusInput = Tk.Scale(self.FeuchteMessenFrame, from\_=1, to=300, resolution=1,

                                      length=320, orient="horizontal", font=14, bg="#1EBB72", highlightbackground="#1EBB72")

        self.RhythmusInput.grid(row=1, column=0)

        self.WasserFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.WasserFrame.grid(row=3, column=0, padx=10, pady=10)

        self.Wasser = Tk.Button(self.WasserFrame, text="Jetzt Waessern.",

                                bg="#1EBB72", width=25, command=self.waessern, font=14)

        self.Wasser.grid(row=0, column=0)

        self.PflanzenFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.PflanzenFrame.grid(row=1, column=1, padx=10, pady=10)

        self.Art = Tk.Label(self.PflanzenFrame, bg="#137547",

                            text="\n Wie viel Wasser braucht Ihre Pflanze: \n (1 = viel; 2 = mittel; 3 = wenig) \n \n", font=14, fg="white")

        self.Art.grid(row=0, column=0)

        self.PflanzeSlider = Tk.Scale(self.PflanzenFrame, from\_=1, to=3, resolution=1, length=320,

                                      orient="horizontal", font=14, bg="#1EBB72", highlightbackground="#1EBB72")

        self.PflanzeSlider.grid(row=1, column=0)

        self.SteuerFrame = Tk.Frame(

            self.root, width=250, height=120, bg="#137547")

        self.SteuerFrame.grid(row=3, column=1, padx=10, pady=10)

        self.StartStopp = Tk.Button(self.SteuerFrame, text="Starten",

                                    bg="#1EBB72", width=25, command=self.startstoppen, font=14)

        self.StartStopp.grid(row=0, column=0, pady=5)

        self.Abbruch = Tk.Button(self.SteuerFrame, text="Herunterfahren",

                                 bg="#1EBB72", width=25, command=self.abbruch, font=14)

        self.Abbruch.grid(row=1, column=0, pady=5)

        self.root.mainloop()

    def PumpeAn(self, pin):

        # Schaltet Pumpe an

        GPIO.output(pin, GPIO.HIGH)

    def PumpeAus(self, pin):

        # Schaltet Pumpe aus

        GPIO.output(pin, GPIO.LOW)

    def messen(self):

        # Liest den Sensor aus und gibt den Wert an das Label weiter

        value = 0

        value = mcp.read\_adc(1)

        valuegerundet = 100-(value/10.23)

        self.momFeuchteWert.config(text=str(valuegerundet)+"\n")

        return valuegerundet

    def waessern(self):

        # Schaltet Pumpe ein und wartet solange, bis die gewünschte Wassermenge gepumpt wurde und schaltet die Pumpe dann aus

        self.PumpeAn(3)

        Pumpenzeit = self.wassermenge()

        time.sleep(Pumpenzeit)

        self.PumpeAus(3)

    def abbruch(self):

        # Fährt den Raspberry Pi herunter

        os.system("sudo shutdown now")

    def wassermenge(self):

        # Berechnet die Dauer, die die Pumpe laufen muss, um die gewünschte Wassermenge zu erreichen

        Menge = self.MengeW.get()

        Pumpenzeit = Menge/200

        return Pumpenzeit

    def messrhythmus(self):

        # Berechnet die Zeitspanne, die der Raspberry Pi warten soll, bevor er die Schleife erneut durchläuft

        rhythm = self.RhythmusInput.get()

        minutenrhythm = rhythm / 60

        return minutenrhythm

    def blumentopf(self):

        # Wählt die eingegebene Pflanzenart aus und durchläuft je nachdem eine andere if-Schleife

        Pflanzenart = self.Pflanzenwahl()

        rhythm = self.messrhythmus()

        if Pflanzenart == 1:

            # Misst die Feuchte

            self.messen()

            # Liest Feuchte aus

            valuegerundet = self.messen()

            if valuegerundet <= 60:

                # Wässert bei gegebenen Bedingungen die Pflanze

                self.waessern()

        if Pflanzenart == 2:

            self.messen()

            valuegerundet = self.messen()

            if valuegerundet <= 40:

                self.waessern()

        if Pflanzenart == 3:

            self.messen()

            valuegerundet = self.messen()

            if valuegerundet <= 5:

                self.waessern()

        if self.running:

            # Ruft sich selbst nach Ablauf des Rhythmus wieder auf

            self.root.after(rhythm, self.blumentopf)

    def startstoppen(self):

        # Je nachdem wann man auf den Knopf drückt, startet oder stoppt die Schleife

        if self.running == True:

            self.StartStopp["text"] = "Starten"

            self.running = False

        else:

            self.StartStopp["text"] = "Stoppen"

            self.running = True

            self.blumentopf()

    def Pflanzenwahl(self):

        # Liest aus, welche Pflanzenart gewählt wurde

        Pflanzenart = self.PflanzeSlider.get()

        return Pflanzenart

GUI().run()

# Autostarten des Programms

Um die Benutzung direkt neben einem Blumentopf noch zu vereinfachen, würde es helfen, wenn sich beim Start das Programm öffnen würde oder wenn man mit ein paar einfachen Klicken, dies öffnen könnte.

Zuerst habe ich probiert, den Code beim Star des Raspberry Pis über crontab zu öffnen. Hierfür hat man einfach ein Bash-Programm geschrieben, welches die GUI im Idealfall aufrufen sollte.[[8]](#footnote-9)

#!bin/sh

cd /

cd /home/pi

sudo python guifertig.py

cd /

Dies hat jedoch nicht funktioniert. Vermutlich liegt das an der Kommunikation zwischen dem Raspberry Pi und dem Display, denn bei anderen Anwendungen hat es funktioniert. Daher habe ich mich für eine andere Lösung entschieden.

Das bereits geschriebene Bash-Programm kann man benutzen, um jetzt bei Öffnung des Terminals die GUI zu starten. Dafür muss man einfach das Kommando:

nano .bashrc

ins Terminalfenster eingeben. Dann kann man in der untersten Zeile des geöffneten Programms einfach folgende Zeile ergänzen[[9]](#footnote-10):

sh launcher.sh

# Anwendung in Bezug auf die Realität

## Pro-Argumente

Ein Vorteil des Smart Blumentopfs ist die einfache Bedienbarkeit. Denn die einzige Sache, um die man sich kümmern muss, ist ein genügender Wasservorrat, auf den die Pumpe zur Bewässerung zugreift.

Außerdem hilft der Smart Blumentopf auch alten und/oder vergesslichen Menschen, indem er ihnen die Arbeit des Pflanzen-Gießens abnimmt.

Die Konstruktion ist außerdem ein wertvoller Helfer bei Abwesenheit und Urlaub.

Der Smart Blumentopf ließe sich auch ideal im Außenbereich einsetzen. Das Relais würde in diesem Fall die elektrische Brunnenpumpe (wahlweise über 433MHz Funksteuerung) ansprechen, an die ein Rasensprenger angeschlossen ist.

## Contra-Argumente

Dagegen sprechen leider auch einige Nachteile. Man kann zwar mit einem Raspberry Pi, bzw. zwei AD-Wandlern 16 Blumentöpfe bewässern, jedoch bräuchte man dafür auch 16 Pumpen, es sei denn man versorgt über einen Sensor und über eine Pumpe mehrere gleichwertige Pflanzen (Verzweigung des Gießschlauchs).

Ein weiteres Problem stellen die Feuchtigkeitssensoren dar, die es auf dem Markt nur in minderwertiger Qualität gibt. Diese korrodieren nach kurzer Zeit beim Kontakt mit Wasser und liefern ungenaue Ergebnisse.

Außerdem ist der Preis (s. nächste Seite) des Smart Blumentopfs nicht ganz verhältnismäßig. Dieser liegt bei ca. 190€, und wenn man das System auf mehrere Pflanzen erweitern möchte, steigern sich die Anschaffungskosten. Hinzu kommen auch noch Betriebskosten (Strom).

## Preisüberschlag

|  |  |
| --- | --- |
| **Bauteil** | **Preis des Bauteils** |
| Raspberry Pi 3[[10]](#footnote-11) | ca. 38€ |
| Netzteil f. Raspberry Pi[[11]](#footnote-12) | ca. 10€ |
| Micro-SD-Karte 16GB[[12]](#footnote-13) | ca. 7€ |
| Breadboard + Kabel als Set[[13]](#footnote-14) | ca. 23€ |
| MCP3008[[14]](#footnote-15) | ca. 2,50€ |
| Bodenfeuchtigkeitssensoren[[15]](#footnote-16) | ca. 11€ |
| Relais[[16]](#footnote-17) | ca. 6€ |
| Netzteil f. Pumpe | ca. 10€ |
| Pumpe[[17]](#footnote-18) | ca. 10€ |
| Schlauch | ca. 5€ |
| Display[[18]](#footnote-19) | ca. 63€ |
| (Maus + Tastatur als Set) | ab ca. 15€ |

Das Netzteil, den Schlauch und Maus + Tastatur hatte ich schon zu Hause und habe daher hier nur einen ungefähren Preis aufgeführt aber keinen Link.

# Schwierigkeiten beim „Bauen“ des Smart Blumentopfs

Die Fehlersuche beim Smart Blumentopf gestaltete sich teilweise schwierig, da ich mich bei den meisten Themen erst einlesen musste und die entsprechenden Anleitungen und Erklärungen in einigen Fällen sehr ungenau oder falsch waren.

Zum einen hatte ich zuerst eine falsche library ausgewählt, mit der das Auslesen des MCP3008 nicht funktioniert hat. Des Weiteren brauchte ich auch Admin-Rechte, um diese in Python zu benutzen.

Der Autostart hat mir auch einige Probleme bereitet, da dies mit der GUI nicht wie gedacht funktioniert hat und ich mir eine Ersatzlösung ausdenken musste.

Außerdem hatte ich kleinere Probleme, wie fehlende Zeichen oder Schreibfehler. Diese waren jedoch in dem langen Code auch teilweise sehr schwer zu finden und haben so auch viel Zeit beansprucht.

Der minderwertigen Qualität der Zusatzmodule fielen mehrere Sensoren zum Opfer.

# Fazit

Die Arbeit mit dem Raspberry Pi hat mir trotz aller Komplikationen viel Spaß bereitet. Durch Tüfteln und Ausprobieren konnte ich meine Fehler beseitigen und immer wieder kleinere Erfolgserlebnisse spüren. Der Raspberry Pi ist ein sehr vielseitiger Ein-Chip-Computer, der nahezu unendliches Potential in sich birgt. Durch meine Recherche habe ich viele andere interessante Projekte entdeckt, von denen ich sicher das ein oder andere mal ausprobieren werde. Ebenso habe ich viel beim Programmieren in Python und im Terminal (Bash) dazugelernt.

# Quellen

<http://www.netzmafia.de/skripten/hardware/RasPi/RasPi_SPI.html>

<https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008>

<https://www.raspiprojekt.de/images/anleitungen/mcp3008/mcp3008_pins.gif>

<https://indibit.de/wp-content/uploads/2015/08/Raspberry-Pi-Model-B-GPIO-Belegung.png>

<https://www.123zimmerpflanzen.de/pflege/giessen>

<https://www.youtube.com/watch?v=51f3ZazNW-w>

<https://www.amazon.de/Raspberry-Model-ARM-Cortex-A53-1-2GHz-Bluetooth/dp/B01CD5VC92/ref=sr_1_3?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=raspberry+pi+3&qid=1591464838&quartzVehicle=72-820&refinements=p_89%3ARaspberry+Pi&replacementKeywords=raspberry+pi&rnid=669059031&sr=8-3>

<https://www.amazon.de/Raspberry-Pi-offizielles-Netzteil-Model/dp/B07TMPC9FG/ref=sr_1_3?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=raspberry+pi+netzteil&qid=1591464884&sr=8-3>

<https://www.amazon.de/SanDisk-Ultra-microSDHC-Speicherkarte-Adapter/dp/B073S9SFK2/ref=sr_1_8?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=micro+sd+karte+16g&qid=1591464864&sr=8-8>

<https://www.amazon.de/Elegoo-%C3%9CBERARBEITETES-Stromversorgungsmodul-Jumperkabel-Potentiometer/dp/B01M7N4WB6/ref=sr_1_4?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=breadboard+elegoo&qid=1591464116&sr=8-4>

<https://www.conrad.de/de/p/microchip-technology-mcp3008-i-p-datenerfassungs-ic-analog-digital-wandler-adc-extern-pdip-651456.html>

<https://www.amazon.de/Kuman-Feuchtigkeitssensor-Kompatibel-Automatisches-Bew%C3%A4sserungssystem/dp/B071LDS4T6/ref=sr_1_3?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=bodenfeuchtigkeitssensor+raSPBERRY+PI+KUMAN&qid=1591465808&sr=8-3>

<https://www.amazon.de/dp/B078Q8S9S9/ref=sr_1_3?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=relais+5v+4+channel&qid=1591464934&quartzVehicle=812-409&replacementKeywords=relais+5v+channel&sr=8-3>

<https://www.conrad.de/de/p/barwig-typ-03-0333-niedervolt-tauchpumpe-720-l-h-6-539082.html?gclid=Cj0KCQjw_ez2BRCyARIsAJfg-ktsNgSM7xaTVfUd_M3n0oBebF9q7i76Sz2Hr1tI0uo71Anx2jaGu_gaAhNJEALw_wcB&hk=SEM&WT.srch=1&WT.mc_id=google_pla&s_kwcid=AL%21222%213%21430836017425%21%21%21g%21%21&ef_id=Cj0KCQjw_ez2BRCyARIsAJfg-ktsNgSM7xaTVfUd_M3n0oBebF9q7i76Sz2Hr1tI0uo71Anx2jaGu_gaAhNJEALw_wcB%3AG%3As>

<https://www.berrybase.de/raspberry-pi-co/raspberry-pi/displays/offizielles-raspberry-pi-7-display-mit-kapazitiven-touchscreen?c=320&px=m>

<https://www.tutorialspoint.com/python/python_gui_programming.htm>

<https://www.youtube.com/watch?v=i7H70wf4iXY>

<https://developer-blog.net/raspberry-pi-autostart-von-programmen/>

1. https://de.wikipedia.org/wiki/Raspberry\_Pi [↑](#footnote-ref-2)
2. <https://indibit.de/wp-content/uploads/2015/08/Raspberry-Pi-Model-B-GPIO-Belegung.png> [↑](#footnote-ref-3)
3. 3 https://www.raspiprojekt.de/images/anleitungen/mcp3008/mcp3008\_pins.gif [↑](#footnote-ref-4)
4. http://www.netzmafia.de/skripten/hardware/RasPi/RasPi\_SPI.html [↑](#footnote-ref-5)
5. https://www.123zimmerpflanzen.de/pflege/giessen [↑](#footnote-ref-6)
6. <https://www.youtube.com/watch?v=51f3ZazNW-w> [↑](#footnote-ref-7)
7. https://www.tutorialspoint.com/python/python\_gui\_programming.htm [↑](#footnote-ref-8)
8. https://www.youtube.com/watch?v=i7H70wf4iXY [↑](#footnote-ref-9)
9. https://developer-blog.net/raspberry-pi-autostart-von-programmen/ [↑](#footnote-ref-10)
10. <https://www.amazon.de/Raspberry-Model-ARM-Cortex-A53-1-2GHz-Bluetooth/dp/B01CD5VC92/ref=sr_1_3?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=raspberry+pi+3&qid=1591464838&quartzVehicle=72-820&refinements=p_89%3ARaspberry+Pi&replacementKeywords=raspberry+pi&rnid=669059031&sr=8-3> [↑](#footnote-ref-11)
11. <https://www.amazon.de/Raspberry-Pi-offizielles-Netzteil-Model/dp/B07TMPC9FG/ref=sr_1_3?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=raspberry+pi+netzteil&qid=1591464884&sr=8-3> [↑](#footnote-ref-12)
12. <https://www.amazon.de/SanDisk-Ultra-microSDHC-Speicherkarte-Adapter/dp/B073S9SFK2/ref=sr_1_8?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=micro+sd+karte+16g&qid=1591464864&sr=8-8> [↑](#footnote-ref-13)
13. <https://www.amazon.de/Elegoo-%C3%9CBERARBEITETES-Stromversorgungsmodul-Jumperkabel-Potentiometer/dp/B01M7N4WB6/ref=sr_1_4?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=breadboard+elegoo&qid=1591464116&sr=8-4> [↑](#footnote-ref-14)
14. <https://www.conrad.de/de/p/microchip-technology-mcp3008-i-p-datenerfassungs-ic-analog-digital-wandler-adc-extern-pdip-651456.html> [↑](#footnote-ref-15)
15. <https://www.amazon.de/Kuman-Feuchtigkeitssensor-Kompatibel-Automatisches-Bew%C3%A4sserungssystem/dp/B071LDS4T6/ref=sr_1_3?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=bodenfeuchtigkeitssensor+raSPBERRY+PI+KUMAN&qid=1591465808&sr=8-3> [↑](#footnote-ref-16)
16. <https://www.amazon.de/dp/B078Q8S9S9/ref=sr_1_3?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=relais+5v+4+channel&qid=1591464934&quartzVehicle=812-409&replacementKeywords=relais+5v+channel&sr=8-3> [↑](#footnote-ref-17)
17. <https://www.conrad.de/de/p/barwig-typ-03-0333-niedervolt-tauchpumpe-720-l-h-6-539082.html?gclid=Cj0KCQjw_ez2BRCyARIsAJfg-ktsNgSM7xaTVfUd_M3n0oBebF9q7i76Sz2Hr1tI0uo71Anx2jaGu_gaAhNJEALw_wcB&hk=SEM&WT.srch=1&WT.mc_id=google_pla&s_kwcid=AL%21222%213%21430836017425%21%21%21g%21%21&ef_id=Cj0KCQjw_ez2BRCyARIsAJfg-ktsNgSM7xaTVfUd_M3n0oBebF9q7i76Sz2Hr1tI0uo71Anx2jaGu_gaAhNJEALw_wcB%3AG%3As> [↑](#footnote-ref-18)
18. <https://www.berrybase.de/raspberry-pi-co/raspberry-pi/displays/offizielles-raspberry-pi-7-display-mit-kapazitiven-touchscreen?c=320&px=m> [↑](#footnote-ref-19)