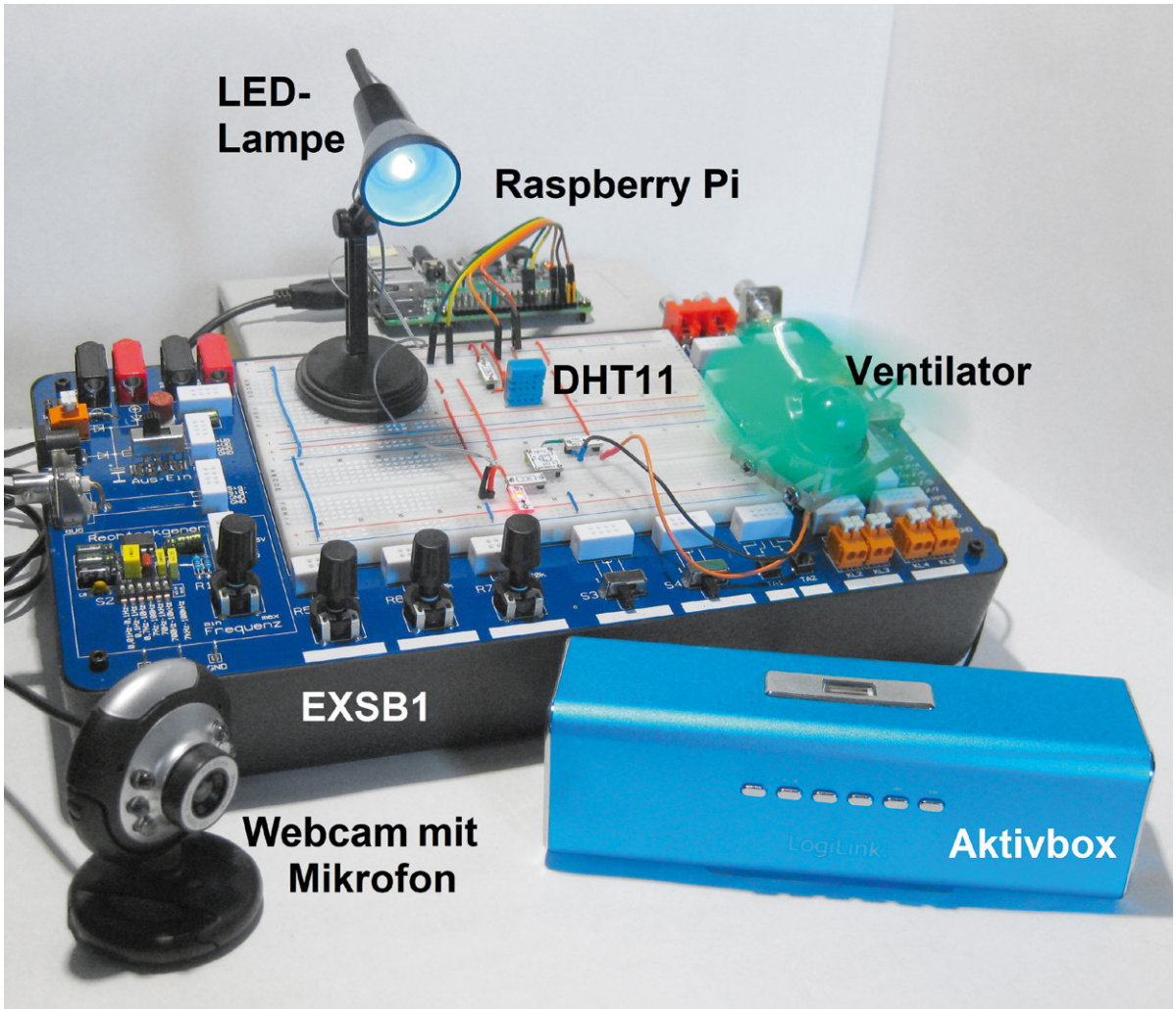
KI-Praxis IV

Teil 4

Spracherkennung und Sprachsynthese

Sprechende Computer kamen bereits vor 20 bis 30 Jahren in Mode. Filme wie „War Games“ mit Sprachausgabe über Computer wurden zu Kassenschlagern, und die TV-Serie „Knight Rider“ mit einem sehr kommunikativen Auto lief auf allen Kanälen. Unvergessen ist auch die Szene, in der Scotty aus „Raumschiff Enterprise - Star Trek V“ versucht, über die Maus mit einem Computer aus den 1980er-Jahren zu sprechen: „Computer? Hallo Computer? - Tastatur?! – wie rückständig ...“ Damals wurde das durchaus noch als erheiternd empfunden – heute sind Sprachsysteme allgegenwärtige Realität. Neben den klassischen Telefon-Sprachsystemen in Banken und Versicherungen stehen „Siri“ und „Alexa“ in vielen Haushalten zur Verfügung. Amazons „Echo“ liest Kindle-Bücher in nahezu natürlicher Sprache vor. Jeder neue Laptop erklärt bei der Inbetriebnahme die „Ersten Schritte“ mittels synthetischer Sprache. Spezielle Sprach-Chips werden dafür nicht mehr benötigt. Bereits ein Raspberry Pi genügt, um mit Sprachwiedergabe und -erkennung zu experimentieren.



Digitale Sprachverarbeitung

Die moderne Sprachsynthese ist das Ergebnis einer langen Geschichte von Versuchen, Sprache mit mechanischen Mitteln zu erzeugen. Erste Geräte zur Nachahmung der menschlichen Sprache wurden bereits vor über 200 Jahren konstruiert. Die Maschinen bestanden aus Elementen, welche verschiedene „Organe" imitierten, die vom Menschen zur Erzeugung von Sprache verwendet werden. Ein Balg für die Lunge, ein Schlauch für den Stimmapparat usw. In der zweiten Hälfte des 19. Jahrhunderts begann u. a. Hermann von Helmholtz damit, Vokale durch Überlagerung harmonischer Wellenformen zu erzeugen. Schließlich war man damit sogar in der Lage, einzelne, einigermaßen verständliche Worte zu synthetisieren.

Der erste digitale Sprachsynthesizer wurde dann in den 1960er-Jahren von den Bell Laboratories konstruiert. Anstelle eines mechanischen Apparats wurden die Wellenformen mit elektronischen Synthesizern erzeugt. Trotz erheblicher Anstrengungen konnte mit dieser Methode jedoch kaum verständliche Sprache erzeugt werden. Die moderne Text-zu-Sprache-Synthese (TTS für engl. Text to Speech) basiert auf einem wesentlich effektiveren Verfahren. Das Sprachsignal wird hierbei aus einem System von Phonemen und häufig vorkommenden Konsonanten-Clustern zusammengesetzt.

Spracherkennungssoftware ist dagegen darauf ausgerichtet, die Eingabe menschlicher Sprache zu entschlüsseln und in Text oder in Anweisungen umzuwandeln. Die Software filtert Wörter, digitalisiert sie und analysiert die einzelnen Komponenten, aus denen sie zusammengesetzt sind. Die digitale Information wird einer mathematischen Analyse unterzogen, um das Gesagte zu interpretieren. Spracherkennungsanwendungen umfassen u. A. Anrufweiterleitung, Sprachwahl, Sprachsuche und -steuerung, Dateneingabe und automatisches Diktieren und Übersetzen.

Traditionell stützten sich Spracherkennungsmodelle auf Klassifizierungsalgorithmen, die darauf ausgerichtet sind, die Verteilung von Frequenzen, Tönen und Phonemen zu erfassen. Dank fortgeschrittener Deep-Learning-Methoden werden heute neuronale Netze verwendet, um Wort- und Spracherkennung in hoher Qualität durchzuführen. Moderne Spracherkennungssoftware verwendet Natural Language Processing (NLP), um Sprache in interpretierbare Komponenten zu zerlegen. Die Software trainiert einen Datensatz bekannter gesprochener Wörter oder Sätze und liefert möglichst präzise Interpretationen. Anschließend werden die gesprochenen Wörter in Text oder Anweisungen umgewandelt.

Das Erkennen von reinen Tonsignalen ist jedoch nicht ausreichend. Um wirklich nützlich zu sein, muss eine Spracherkennungssoftware beispielsweise auch den Unterschied zwischen Eigennamen und regulären Wörtern erkennen (zum Beispiel ein „Koch" in der Küche oder das „Robert Koch Institut") und zwischen Homophonen unterscheiden können. Eine Herausforderung bei der Spracherkennung besteht somit darin, einen intelligenten Prozess zu schaffen, der nicht nur Sprache „hört", sondern auch Wissensquellen und sprachliche Informationen miteinander verbindet.

Dabei ist es wesentlich schwieriger, das gesprochene Wort zu erkennen, als Sprache elektronisch zu erzeugen. Für die Spracherkennung müssen entsprechende neuronale Modelle zunächst phonetische Einheiten erfassen. Dann ist ein „Wörterbuch" erforderlich, das alle Begriffe enthält, in welchen bestimmte Phoneme bzw. Klänge vorkommen. Schließlich muss noch ein Sprachmodell entwickelt werden, das sinnvolle Wörter und Abfolgen erkennt. Erst die Kombination dieser Verfahren liefert letztendlich brauchbare Ergebnisse. Die jeweiligen Modelle sind jedoch keineswegs statisch. Durch das Verbessern von nicht korrekt erkannten Begriffen lernt die Spracherkennung ähnlich wie ein Kleinkind, auch unklare Aussprachen oder Dialekte zu verstehen. Eine gute Spracherkennung ist also langfristig durch die Qualität des zugrunde liegenden maschinellen Lernens bestimmt.

Aus diesen Gründen ist es bislang kaum möglich, eine akzeptable Spracherkennung autonom auf kleinen Systemen wie dem Raspberry Pi zu implementieren. Im Rahmen dieses Artikels wird daher auf ein von Google zur Verfügung gestelltes Web-Interface zurückgegriffen.

Der richtige Ton macht die Musik

Zunächst soll jedoch eine Sprachausgabe realisiert werden. Das folgende Set-up geht von der Ausgabe über die 3,5-mm-Klinkenbuchse des Raspberry Pi aus. Hier können beispielsweise Kopfhörer, Aktivboxen oder ein anderes Audioausgabe-System angeschlossen werden.

Die Konfiguration erfolgt über „Raspi-config" und dessen Menüpunkt zum Steuern der Ausgabe:

Advanced Options → Audio

Hier sollte

Forced 3,5 mm ('headphone') jack

ausgewählt werden. Dadurch wird der Parameter snd-usb-audio in

/etc/modprobe.d/alsa-base.conf

von -2 auf 0 gesetzt. Die Soundausgabe kann über die Anweisung

speaker-test -t wav -c 2

überprüft werden. Nach der Eingabe des Befehls sollte aus der linken Box „front left" und aus der rechten Box „front right" zu hören sein.

Die für die spätere Spracherkennung erforderliche Soundeingabe ist etwas aufwendiger, da der Raspberry Pi nicht über einen Mikrofoneingang verfügt. Hier muss man sich mit externen Geräten behelfen. Infrage kommen entweder eine USB-Soundkarte (Bild 1) oder eine Webcam mit integriertem Mikrofon. Bei der Beschaffung einer USB-Soundkarte ist darauf zu achten, dass diese mit dem aktuellen Raspberry-Pi-Betriebssystem kompatibel ist. Die Verwendung einer Webcam hat zusätzlich den Vorteil, dass diese später auch für andere KI-Projekte wie Objekt- und Gesichtserkennung eingesetzt werden kann.

Wird das Mikrofon über eine unterstützte USB-Soundkarte oder eine audiofähige Webcam angeschlossen, sind keine weiteren Konfigurationen erforderlich. Dasselbe gilt für USB-Mikrofone, da diese lediglich eine spezielle Soundkarte mit fest integriertem Mikrofon darstellen. Für einen ersten Test und zum Einstellen der Aufnahme- und Wiedergabelautstärke kann der sogenannte „alsamixer"

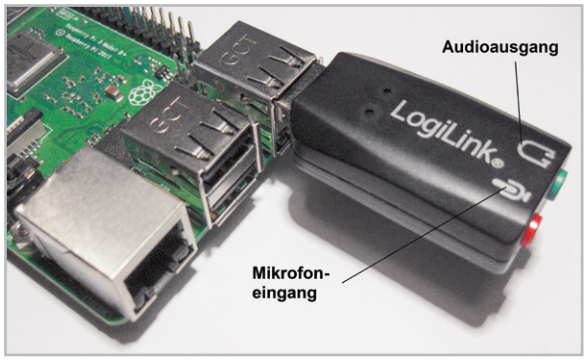


Bild 1: Externe USB-Soundkarte

verwendet werden. Nach dem Start über die Konsole wird mit der F6-Taste das gewünschte Sound-Device ausgewählt. Die Einstellungen werden über die Pfeiltasten geändert.

Erscheint statt des alsamixers nur eine Fehlermeldung, gehört der angemeldete Nutzer nicht zur Gruppe „audio“. Für den Standardnutzer „pi“ sollte dies aber kein Problem darstellen.

Nun sollte man überprüfen, ob das verwendete Mikrofon ordnungsgemäß arbeitet. Nach Anschluss der Geräte und der Eingabe der Anweisung

lsusb

erscheint die Webcam oder das Mikrofon in der ausgegebenen Liste (Bild 2).

Als Nächstes wird die Aufzeichnungslautstärke eingestellt. Über die Anweisung

alsamixer

wird eine grafischen Oberfläche gestartet (Bild 3). Hier kann über die Aufwärts-/Abwärts-Pfeiltasten die gewünschte Lautstärke eingestellt werden.

Über die Taste F6 (alle) wird die Webcam oder das Mikrofon aus der Liste ausgewählt. Der Befehl

Arecord -l

listet nochmals Mikrofon bzw. Webcam auf. Mit

Arecord -D plughw: 1,0 test.wav

kann eine Tonaufnahme gestartet werden. Die Aufzeichnung wird in der Datei „test.wav“ abgespeichert. Eine testweise Wiedergabe über Kopfhörer/Aktivboxen erfolgt über den Befehl

Aplay test.wav

Ist der aufgezeichnete Ton hörbar, arbeiten Mikrofon und Soundausgabe einwandfrei. Andernfalls sollten die Einstellungen und die Lautstärkejustierung nochmals geprüft werden.

Sprachausgabe

Das bei Linux-Nutzern beliebte TTS-System „eSpeak“ wurde seit fast einem Jahrzehnt kaum weiterentwickelt und erfordert sehr spezielle Installationen für einen einigermaßen reibungslosen Betrieb.

Für den Raspberry Pi kann jedoch eine neue eSpeak-Variante verwendet werden. Das sogenannte „eSpeak-NG“ ist weitestgehend mit dem aktuellen Raspberry Pi OS kompatibel und kann sehr einfach installiert werden:

sudo apt install espeak-ng
espeak-ng-data libespeak-ng-dev

Zudem existiert hierzu eine Python-Schnittstelle, die über

sudo pip3 install py-espeak-ng

auf den Raspberry Pi geladen werden kann. eSpeak-NG beherrscht mehrere Sprachen, neben Englisch u. a. auch Deutsch. Das folgende Programm liefert einen ersten Eindruck von den Möglichkeiten der TTS auf dem Raspberry (s. Download-Paket „eSpeak_NG_tst.py“):

```
from espeakng import ESpeakNG
from time import sleep

phrase1_en="My car is a BMW and I drive it rather fast"
phrase1_de="Guten morgen Dr Falken!"
phrase2_de="Wollen wir ein Spiel spielen?"

print(phrase1_en)
esng=ESpeakNG(voice='en')
esng.pitch = 5
esng.speed = 120
esng.say(phrase1_en, sync=True)

print(phrase1_de); print(phrase2_de);
esng=ESpeakNG(voice='de')
esng.pitch = 10
esng.speed = 150
esng.say(phrase1_de, sync=True)
esng.say(phrase2_de, sync=True)
```

Wenn alles korrekt installiert ist, sind nach dem Starten des Programms die Sätze

„My car is a BMW and I drive it rather fast“

auf Englisch und

„Guten Morgen, Dr. Falken!“
„Wollen wir ein Spiel spielen?“

auf Deutsch zu hören. Die Sprache klingt zwar noch nicht vollkommen natürlich, ist aber gut verständlich. Eine Klangprobe dazu ist im Download-Paket enthalten (eSpeak_NG_tst.mp3).

Über Online-Anwendungen könnten wesentlich bessere Sprachqualitäten erreicht werden. Text-to-Speech-Dienste (oft auch als „Text2Speech-Dienste“ bezeichnet) in der Cloud sind in der Lage, kurzzeitig hohe Rechenleistungen abzurufen, und kommen einer natürlich klingenden Sprache am nächsten. Aber auch die Offline-Alternativen wie eSpeak-NG sind inzwischen vergleichsweise gut verständlich.

Auf dem Markt sind Open-Source-Bibliotheken verfügbar, die zwar noch als synthetische Stimme erkennbar sind, aber eine sehr gut verständliche Sprachausgabe auf dem Raspberry Pi in Echtzeit bereitstellen. Hierfür ist jedoch eine vergleichsweise kostspielige Zusatz-Hardware erforderlich. Im Folgenden sollen daher die Möglichkeiten von eSpeak-NG ausgelotet werden. Neben allen vorgestellten Programmen finden sich dazu auch die Sound-Beispieldateien zu eSpeak-NG im Download-Paket.

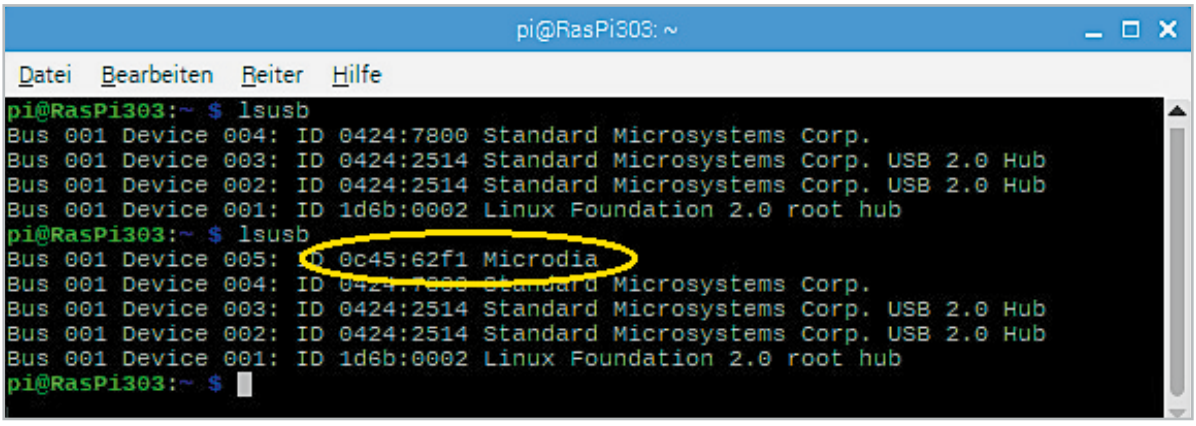


Bild 2: Die externe USB-Soundkarte/Webcam wurde korrekt als USB-Device eingebunden.

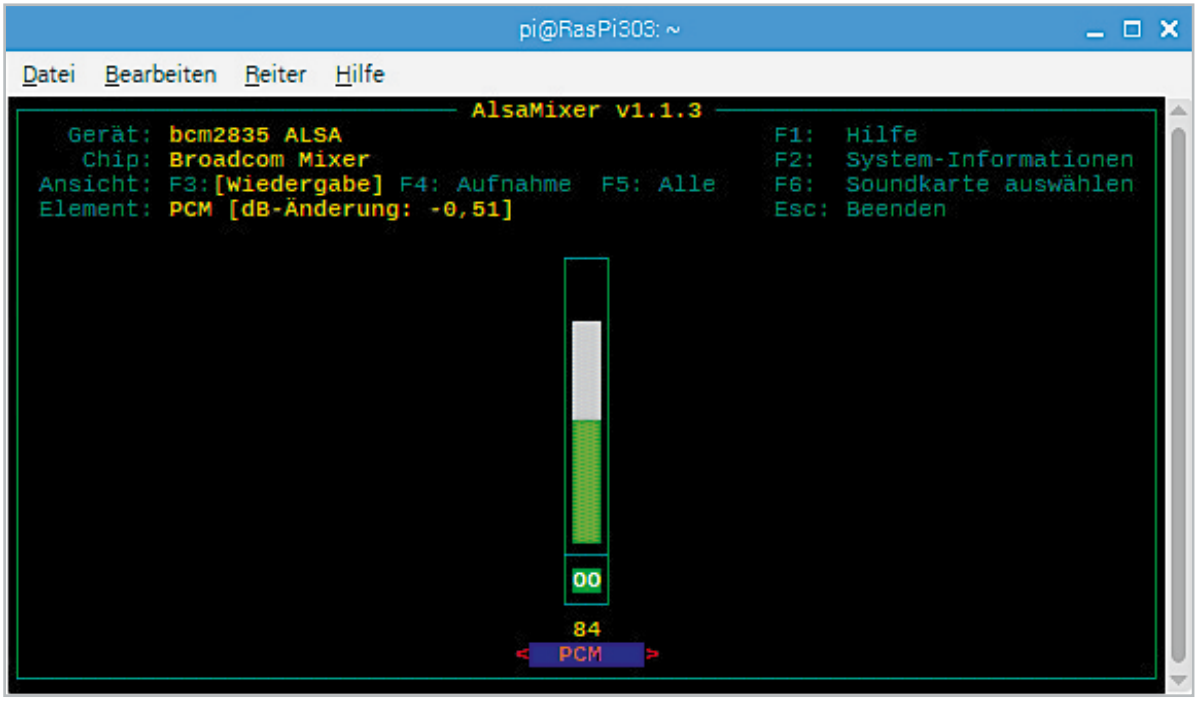


Bild 3: Mit dem AlsaMixer wird der Schallpegel des Mikrofons eingestellt.

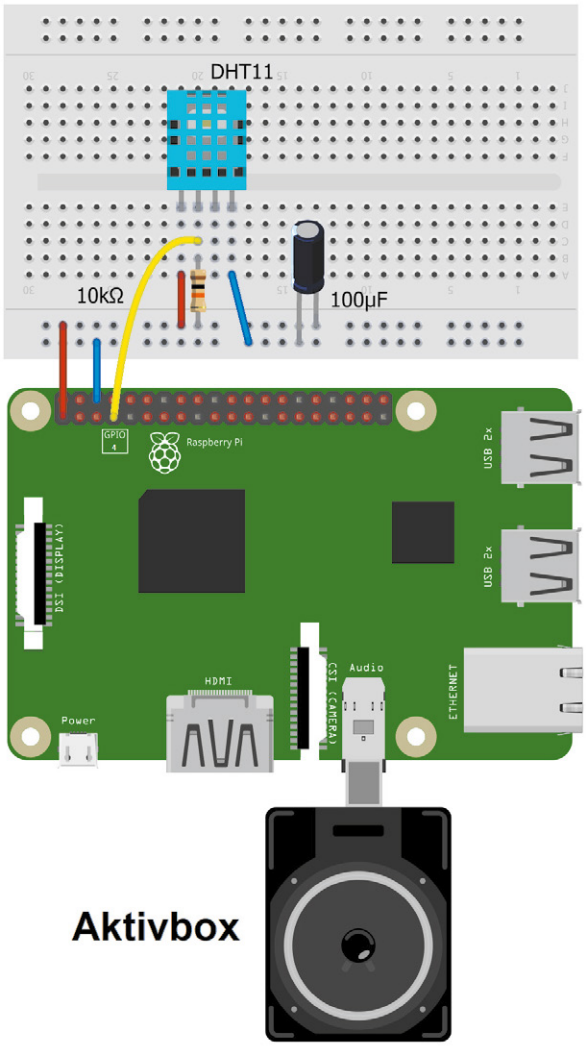


Bild 4: Schaltbild zur sprechenden Klimastation

Gesprächige Geräte

Eine interessante Anwendung für die Sprachausgabe ist der Bau von „sprechenden" Geräten. Der Vorteil beispielsweise bei einem sprachfähigen Multimeter liegt darin, dass man die Augen auf den Schaltschrank gerichtet lassen kann, während das Messgerät den aktuell anliegenden Spannungswert vorliest. Anstelle eines Multimeters soll im Folgenden eine sprechende Klimastation vorgestellt werden. Diese liefert in regelmäßigen Abständen die aktuellen Werte. Bild 4 zeigt den Anschluss des Sensors an den Raspberry Pi.

Der DHT11-Sensor benötigt für seinen Betrieb lediglich einen 10-kΩ-Widerstand als Pull-up an +3,3 V. Alternativ kann der DHT22 verwendet werden. Falls dieser als Modul eingesetzt wird (s. Materialliste), kann sogar auf den Pull-up-Widerstand verzichtet werden, da dieser dann bereits auf der Trägerplatine integriert ist. Sicherheitshalber sollte in jedem Fall noch ein Stabilisierungselko von 100 µF in die Spannungsversorgung eingeschleift werden. Für die Tonausgabe wird am Standard-Audioausgang des Raspberry Pi beispielsweise eine Aktivbox oder ein Kopfhörer angeschlossen.

Softwareseitig steht für diesen Sensortyp eine komfortable Library zur Verfügung. Sie kann über

```
git clone https://github.com/adafruit/
Adafruit_Python_DHT.git
```

von einem Github-Repository geladen werden. Nach dem Wechseln in das neu zu erstellende Verzeichnis

```
cd Adafruit_Python_DHT
```

wird die Installation über

```
sudo apt-get install build-essential python-dev
```

und schließlich

```
sudo python3 setup.py install
```

fertiggestellt. Danach kann der Sensor mit dem folgenden Code getestet werden (s. DHT11_tst.py im Download-Paket):

```
# -*- coding: utf-8 -*-

import Adafruit_DHT
from time import sleep

sensor = Adafruit_DHT.DHT11
pin = '4'
```

```
while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
    if humidity is not None and temperature is not None:
        print('Temp = {0:0.0f} °C - Humidity = {1:0.0f} %'.format(temperature, humidity))
    else:
        print('Failed to get reading. Try again!')
    sleep(1)
```

Das Programm gibt die vom Sensor erfassten Temperatur- und Luftfeuchtigkeitswerte auf die Konsole aus. Nun müssen die so aufgenommenen Werte nur noch via eSpeak ausgegeben werden. Davor ist allerdings noch eine kleine Hürde aus dem Weg zu räumen. Für die Umwandlung von Zahlen in Zahlenwörter (123 → „one hundred and twenty three"). Ist die Library „num2words" erforderlich. Diese wird über

pip3 install num2words

installiert. Dann kann das Programm „eSpeak_NG_climate_station.py" gestartet werden:

```
from espeakng import ESpeakNG
import Adafruit_DHT
from time import sleep
from num2words import num2words

sensor = Adafruit_DHT.DHT11
pin = '4'

esng=ESpeakNG(voice='en')
esng.pitch = 25
esng.speed = 180
esng.say

while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
    print('Temp = {0:0.0f} °C - Humidity = {1:0.0f} %'.format(temperature, humidity))
    esng.say("room temperature is",sync=True)
    count = num2words(temperature)
    esng.say(count,sync=True)
    esng.say(" degrees centigrade",sync=True)

    esng.say("humidity is",sync=True)
    count = num2words(humidity)
    esng.say(count,sync=True)
    esng.say(" %", sync=True)
    sleep(5)
```

Danach sollten im angeschlossenen Aktivlautsprecher im Abstand von ca. 5 Sekunden die aktuellen Werte für Temperatur und Luftfeuchte zu hören sein. Natürlich ist diese ununterbrochene Daueransage nicht sehr sinnvoll. Hier könnte man etwa über eine Tasterabfrage die Werte nur bei Bedarf ausgeben. Mithilfe von KI-Methoden geht es allerdings noch eleganter. So kann man mittels Spracherkennung dafür sorgen, dass die Klimastation nur „spricht", wenn sie „gefragt" wird.

Spracherkennung

Für eine effiziente Spracherkennung sind jedoch auch die Ressourcen eines Raspberry Pi4 mit 8 GB noch nicht wirklich ausreichend. Wie bereits in der Einleitung dargelegt, kommt man hier praktisch nicht ohne Rückgriff auf Online-Dienste aus. Ein Mittel der Wahl ist das von Google bereitgestellte Spracherkennungsmodul „SpeechRecognition". Dieses wird über die folgenden Anweisungen installiert:

```
sudo apt-get --yes install flite
sudo apt-get --yes install clang
sudo apt-get --yes install git-core
sudo apt-get --yes install swig
sudo apt-get --yes install python-pyaudio
sudo apt-get --yes install portaudio19-dev
sudo apt-get --yes install flac

pip3 install SpeechRecognition
pip3 install google-api-python-client
pip3 install pyaudio
```

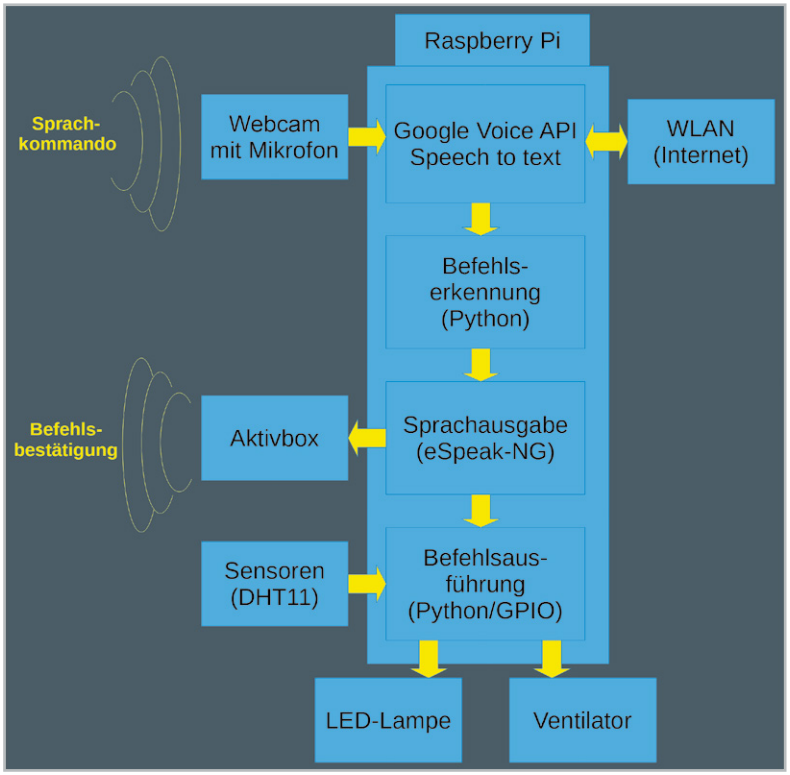
Die gesamte Installation nimmt bis zu einer Stunde in Anspruch. Die Anweisungen finden sich auch als Datei im Download-Paket, sodass die Befehle direkt via Copy-and-Paste in das Terminal übertragen werden können und das aufwendige Abtippen entfällt.

Für die Google-Speech-Recognition-Anwendung wird der Sprachbefehl des Benutzers zunächst vom Mikrofon erfasst und digitalisiert. Dann wird die Information über die Google API in maschinenlesbaren Text konvertiert. Damit ist das Programm auch als Teil eines interaktiven Sprachantwortsystems verwendbar. Über ein Python-Programm kann der Raspberry Pi über Sprachbefehle auf vorgegebene Anweisungen reagieren. Das folgende Blockdiagramm in Bild 5 zeigt die grundlegende Funktionsweise eines so aufgebauten „Chatbots":

Das vollständige Programm (ChatBot_DHT11_1V1.py) ist wieder im Download-Paket enthalten. Da dieses bereits recht umfangreich ist, werden hier nur einige wesentliche Hauptkomponenten erläutert. Die anderen Bestandteile sollten nach dem Durcharbeiten der bisherigen Artikel aber kein Verständnisproblem darstellen.

Zunächst werden einige Schlagwörter definiert:

Bild 5: Spracherkennung auf dem Raspberry Pi




```
greetings=['hello', 'hi', 'hey']
questions=['how are you', 'how are you doing']
responses=['okay', 'I am fine']
database={
 'hello Robert':'hello, dr Falken, how can i help you',
 'name':'Robert',
 'what is your name':'my name is Robert',
 'what can you do for me':'i can do many things..',
 'I love you':'i love you too'
 }
exitWord=['quit','exit','bye','goodbye']
```

Diese legen vordefinierte Antworten fest. Dabei wird jedoch eine gewisse Flexibilität erlaubt. Auf die Frage

„How are you?"

können beispielsweise die Antworten

„Okay" oder „I am fine"

ausgegeben werden. In der Hauptschleife wird über

with m as source: audio = r.listen(source)

das Tonsignal vom Mikrophon aufgenommen. Danach wird mit

value = r.recognize_google(audio)

nach verständlicher Sprache im Audiosignal gesucht. Wenn erkennbare Wörter oder sogar ganze Sätze gefunden wurden, werden diese in Text umgewandelt und stehen in der Variablen „data" zur Verfügung. Diese Variable kann nun nach Schlüsselwörtern durchsucht werden. Über einzelne Routinen wie

```
elif 'temperature' in data:
        print("You said: %s" % reply)
        humidity, temperature = DHT.read_retry(sensorType, humiturePin)
        print("The room temperature is: ", temperature, end=""); print(" °C")
        number=num2words(temperature)
        esng.say("the room temperature is", sync=True)
        esng.say(number, sync=True)
        esng.say(phrase3, sync=True)
```

oder

```
elif 'CPU temperature' in data:
        print("You said: %s" % reply)
        tempCPU=getCPUtemperature()
        print("CPU temperature: ", tempCPU, end=""); print(" °C")
        number=num2words(tempCPU)
        esng.say("the CPU temperature is", sync=True)
        esng.say(number, sync=True)
        esng.say(phrase3, sync=True)
```

wird spezifisch auf verschiedene Anweisungen reagiert. Im ersten Fall erfolgt die akustische Ausgabe der vom angeschlossenen DHT11-Sensor erfassten Werte. Im zweiten Fall liefert der Chatbot seine aktuelle CPU-Temperatur.

Ein Chatbot für die Hausautomatisierung

Einer der großen Vorteile des Raspberry Pi im Vergleich zu einem klassischen PC oder Laptop ist die Verfügbarkeit von frei programmierbaren I/O-Pins. Diese eignen sich bestens für Anwendungen beispielsweise in der Hausautomatisierung. Falls die Library zur Ansteuerung der Pins noch nicht geladen ist, kann dieses über

```
pip install Rpi.GPIO
```

nachgeholt werden. Zusammen mit den Spracherkennungs- und -ausgabemodulen steht dann eine leistungsfähige Basis für die sprachgesteuerte Hausautomatisierung zur Verfügung. Über Abfragen wie

```
elif 'light on'in data in data:
        print("turning light on")
        esng.say("turning light on", sync=True)
        GPIO.output(LEDPin,True)
```

kann unter Nutzung des entsprechenden Pins eine LED, ein Ventilator oder eine Raumbeleuchtung ein- bzw. ausgeschaltet werden. Im aktuellen Programm sind u. a. die folgenden Anweisungen implementiert:

Abfrage	Aktion
CPU temperature	liefert die CPU-Temperatur
light on/off	schaltet eine an Pin 14 angeschlossene LED
fan on/off	schaltet einen am entsprechenden Pin angeschlossenen Ventilator ein bzw. aus.
time	liefert die aktuelle Uhrzeit
room temperature	liefert die Raumtemperatur via DHT11-Sensor
humidity	liefert die Luftfeuchtigkeit via DHT11-Sensor

Darüber hinaus sind noch einige Antworten eingebaut, die besonders Kindern Freude bereiten. Versuchen sie es z. B. mit der Anfrage „What does the duck say?“. Erstaunlich ist hierbei, wie präzise die Spracherkennung zwischen „dog“ oder „duck“ unterscheidet. Natürlich kann das System problemlos auch auf andere Anweisungen erweitert werden. Unter [1] findet sich ein Video, das die Anwendung dieses sprachgesteuerten Heimautomatisierungssystems demonstriert. Bild 6 zeigt die Verschaltung der einzelnen Komponenten.

Als Transistor kann ein Universaltyp wie beispielsweise der BC548 verwendet werden, solange der Ventilatormotor nicht mehr als 100 mA zieht. Die eingezeichnete Diode (z. B. 1N4001) dient lediglich als Freilaufschutz. Als Basiswiderstand für den Transistor sollten ca. 33 kΩ gewählt werden. Alternativ können auch die entsprechenden Bauelemente aus dem Prototypenadaptersatz PAD2 (s. Materialliste) verwendet werden.

Das gesamte System kann problemlos auf einem Breadboard aufgebaut werden. Besonders geeignet für entsprechende Aufbauten ist das EXSB1-System (s. Materialliste). Dieses bietet neben einem ausreichend großen Steckfeld auch wichtige externe Komponenten wie LEDs, Stecker oder Potentiometer (s. Titelbild). Das System wird auch in späteren KI-Anwendungen immer wieder als Hardwarebasis dienen.

Expertensystem als Chatbot

Expertensysteme waren ein früher Versuch die Problemlösungsfähigkeiten von Menschen auf Maschinen zu übertragen. Im Prinzip handelt es sich dabei um Programme, die das Spezialwissen hochqualifizierter Fachleute auf eng begrenzten Aufgabengebieten nachbilden sollen. Die ersten Expertensysteme hatten beispielsweise das Ziel, Infektionskrankheiten zu diagnostizieren. Damit sollten Ärzte von Routineaufgaben entlastet werden.

Obwohl die Implementierung von Wissen in Form von Regeln prinzipiell verhältnismäßig einfach ist, wird die umfassende Wissensrepräsentation auch nur eines eng abgegrenzten Aufgabengebiets rasch extrem aufwendig. Als klassisches Beispiel kann man die Sprache selbst heranziehen. Zum Beispiel gilt im Englischen bei der Bildung der Simple-Past-Vergangenheitsform die Regel

WENN Infinitiv+ed
DANN Simple Past-Vergangenheitsform.

Diese Regel funktioniert bei vielen Verben (z. B. „walk“, „talk“). Sobald aber Verben einer Ausnahmeregel unterworfen sind (z. B. „run“, „eat“) oder im Infinitiv auf „e“ enden (z. B. „compare“, „smile“), müssen bereits wieder neue Regeln definiert werden.

Expertensysteme konnten sich daher langfristig nicht durchsetzen. So stellte sich insbesondere im medizinischen Bereich heraus, dass regelbasierte Diagnosen schnell zu nicht mehr handhabbaren Entscheidungsbäumen anwuchsen. So können etwa die einfachen „Grippe“-Symptome wie Husten, Heiserkeit und Gliederschmerzen auf Hunderte von verschiedenen Erkrankungen hindeuten. Ein erfahrener Arzt ist in der Lage, die unwahrscheinlichen Diagnosen rasch auszuschließen. Ein Expertensystem muss hierzu Dutzende von Fragen stellen.

Zudem führte die Geschwindigkeit, mit der neue Daten und Erkenntnisse erfasst werden, bei regelbasierten Expertensystemen zu einem nicht mehr kontrollierbaren Wartungsaufwand. Das Hauptproblem lag darin, dass die Systeme nicht lernfähig waren. Genau dies führte letztendlich zu neuen Denkansätzen, und u. a. auch zur Entwicklung des maschinellen Lernens.

Dennoch können Expertensysteme in klar umrissenen Bereichen erfolgreich eingesetzt werden. Wie das Beispiel der Hausautomatisierung zeigt, reichen hier vergleichsweise wenige Anweisungen aus, um die gewünschten Licht- oder Temperaturverhältnisse einzustellen. Damit hat in diesem Fall eine einfaches „Expertensystem“ durchaus seine Berechtigung.

Von ELIZA zu Alexa - Fluch oder Segen?

Die Wörter, die Menschen in ihrem täglichen Leben verwenden, können wichtige Aspekte ihrer sozialen und psychologischen Welt enthüllen. Mit den Fortschritten in der Computertechnologie ermöglicht es die Textanalyse sogar, den Sprachstil einer Person zu bewerten. Entsprechende Analyseprogramme lassen Rückschlüsse auf die soziale und psychologische Situation eines Menschen zu.

Natural Language Processing ist der Teilbereich der Informatik und künstlichen Intelligenz, der sich mit den Interaktionen zwischen Computern und der menschlichen Sprache befasst. Ein spezielles Forschungsgebiet des NLP sind Chatbots, auch bekannt als Talkbots, Chatterbots oder interaktive Agenten. Chatbots werden in Dialogsystemen für verschiedene praktische Zwecke verwendet. Häufig kommen sie in den Bereichen wie

- Kundenunterstützung und Support
- Unterhaltung, Spiele und Sport

- Finanzen und Marketing
- Gesundheit, Personalwesen und Sozialdienste
- Nachrichten und Wetterprognose

zum Einsatz. Einige Chatbots verwenden dieselben Technologien, welche auch die Grundlage von virtuellen Assistenten wie Google Assistant, Siri von Apple und Microsoft Cortana bilden.

Einfachere Systeme suchen nach Schlüsselwörtern in der Eingabe und bestimmen dann mithilfe einer Datenbank die am besten passenden Antworten. Chatbots verarbeiten den vom Benutzer präsentierten Text und antworten nach der Anwendung einer komplexen Reihe von Algorithmen. Sie identifizieren, was der Benutzer gesagt hat, und ermitteln anhand dieser Informationen geeignete Antworten. Die klassische Chatbot-Technologie verwendet meist noch keine NLP-Systeme. Subtile Sprachnuancen im gesprochenen Dialog werden daher oft nicht korrekt erkannt.

Ein erstes Vorläufersystem der Chatbots war ELIZA. Die Kommunikation erfolgte damals Mitte der 1960er-Jahre natürlich noch über Tastatur und Bildschirm. ELIZA sollte die typischen Antworten eines Psychotherapeuten imitieren. Neben einfachen Zufallselementen wurde die Konversation mithilfe von „Pattern Matching" und Substitutionsmethoden am Laufen gehalten. Eines der interessantesten Resultate dieses Experiments war, dass viele Menschen ungefiltert ihre intimsten Gedanken und Probleme preisgaben, ohne dass der Chatbot auch nur die geringste „Vorstellung" von den tatsächlichen Problemen der Personen hatte.

Eine typische „Unterhaltung" mit Eliza sah etwa so aus:

ELIZA >>> Hallo, ich bin ELIZA
Person: Hallo, ich habe eine Frage
ELIZA >>> Bist du gekommen, weil du eine Frage hast?
Person: Ja
ELIZA >>> Wie geht es Dir?
Person: Danke, mir geht es gut
ELIZA >>> Schön
Person: Wie alt bist Du?
ELIZA >>> Wir wollen von Dir reden, nicht von mir
Person: Ich habe Probleme mit meinem Vater
ELIZA >>> Warum hast du Probleme mit deinem Vater?
...

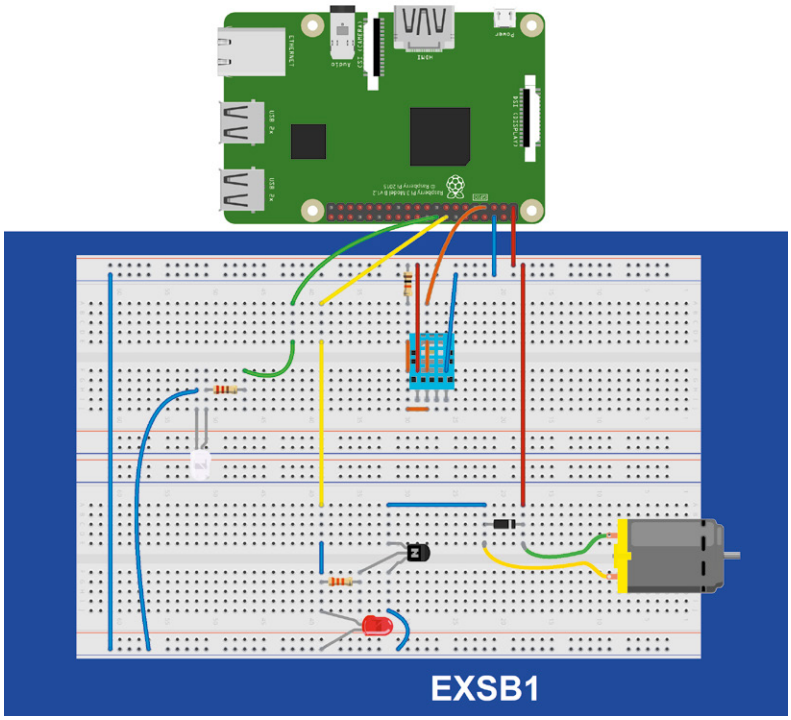


Bild 6: Schaltung zum Hausautomatisierungs-Chatbot

Obwohl das System nicht über Informationen zum menschlichen Denken oder über Emotionen verfügte, lieferte das Programm oftmals erstaunlich menschenähnliche Antworten. Chatbots werden oftmals als unheimlich oder als „Big brother"-Technologie empfunden. Dennoch kommen in vielen Haushalten zunehmend digitale Sprachassistenten wie Amazon Alexa oder Google Home zum Einsatz. Ob sich diese Sprachassistenten als Fluch oder Segen für die Menschheit entpuppen, muss die Zukunft erst noch zeigen. Allerdings sollte man immer im Auge behalten, dass ein Spracherkennungssystem prinzipiell für unlautere Zwecke missbraucht werden kann. Mag dies in weitgehend demokratischen Staaten noch eine relativ geringe Gefahr sein, kann es in totalitären Systemen schon ganz anders aussehen. Behörden wie die Staatssicherheit der ehemaligen DDR wären von digitalen Sprachassistenten sicher im höchsten Maße begeistert gewesen ...

Fazit und Ausblick

In diesem Beitrag wurde der Raspberry Pi mithilfe von eSpeak-NG und einer Google API zu einem universell einsetzbaren Chatbot ausgebaut. Die besonderen Vorteile des Systems im Vergleich zu Alexa oder Siri liegen in den beim Raspberry verfügbaren I/O-Pins. Diese erlauben es, Sensoren oder Aktoren anzuschließen, die interaktiv über Sprachbefehle abgefragt oder gesteuert werden können. Zudem hat man beim Eigenbau des Systems eine gewisse Kontrolle über die eingesetzte Technologie. Das ist bei kommerziellen Systemen meist nicht der Fall.

Im nächsten Beitrag dieser Reihe wird es um die KI-gestützte Objekterkennung gehen. Mithilfe der Pi-Cam oder einer externen Webkamera kann der Raspberry Pi über geeignete Algorithmen Objekte wie Fahrzeuge, Alltagsgegenstände (Tische, Stühle, Topfpflanzen etc.) oder auch Tiere eigenständig erkennen und klassifizieren. Auch die Sprachfähigkeiten des Systems werden nochmals zum Einsatz kommen. Sie ermöglichen die akustische Ausgabe der erkannten Objekte. Eine so entstehende "See-and-Talk-Box" kann z. B. für blinde oder sehbehinderte Menschen eine wichtige Alltagshilfe sein. ELV

Material	Artikel-Nr.
Raspberry Pi 4 Model B, 8 GB RAM	250567
DHT22-Sensor (Modul)	251190
ELV Bausatz Experimentier-/Steckboard EXSB1	153753
Prototypenadapter für Steckboards PAD2	154712

i Weitere Infos

Download-Paket zu diesem Beitrag:
Artikel-Nr. 252343

[1] https://www.youtube.com/watch?v=2ShpDrSHTEo&t=99s

Alle Links finden Sie auch online unter:
de.elv.com/elvjournal-links