

MICHAEL WEIGEND

RASPBERRY PI

FÜR **KIDS**

5. AUFLAGE

PROGRAMMIEREN LERNEN
UND EXPERIMENTIEREN MIT
ELEKTRONIK, SCRATCH UND PYTHON



INHALT

X	SCRATCH IM DIENST DER WISSENSCHAFT	3
	Die digitale Kellerassel	3
	Wie funktioniert ein Gaschromatograph?	9
	Fragen	13
	Antworten zu den Fragen	14
Y	STETS ZU DIENSTEN - - DER RASPBERRY PI ALS WEBSERVER	15
	Raspberry Pi als Webserver	15
	Projekt: Wie spät ist es? Dynamische Webseiten	21
	Projekt: Spion im Garten	25
	Tethering – Das Handy zum Hotspot machen	29
	Projekt: Streng geheim! Eine Website mit Zugangsschutz	30
	Projekt: Über eine Webseite eine LED steuern	34
	Fragen	37
	Aufgabe: Temperaturmessung über das Netz	37
	Antworten zu den Fragen	38
	Lösung der Aufgabe	38



Computer spielen in den Naturwissenschaften eine wichtige Rolle. In diesem Kapitel werden einige typische Beispiele vorgestellt. Wir simulieren eine Kellerassel, die Schutz in der Dunkelheit sucht und einen Gaschromatographen.

DIE DIGITALE KELLERASSEL

Kennst du Asseln? Du findest diese kleinen krebsartigen Tiere draußen im Garten unter altem Holz oder Laub. Sie haben eine ovale Form, sind ziemlich platt, besitzen sieben Beinpaare und vorne lange Fühler. Asseln lieben die Dunkelheit. Wenn eine Assel dem Tageslicht ausgesetzt ist, krabbelt sie wild umher und sucht eine Stelle, an der es dunkel und feucht ist. Genau dieses Verhalten versuchen wir zu simulieren. Wir entwickeln mit Scratch in zwei Schritten eine digitale Assel, die laufen kann (Schritt 1) und die sich vorzugsweise unter einem Blatt aufhält (Schritt 2).

EIN KRABELNDE ASSEL

Das Ziel des ersten Schritts ist eine Assel, die auf einer unregelmäßigen, zufälligen Zickzackbahn über den Bildschirm krabbelt. Wenn sie an den Rand stößt, prallt sie ab.

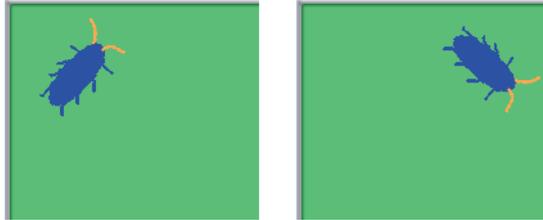


Abb. X.1: Wenn die Assel an den Rand stößt, prallt sie ab.

DIE KOSTÜME

Starte Scratch und lösche die Katze. Berühre die Schaltfläche FIGUR WÄHLEN und klicke dann auf MALEN. Es öffnet sich das Fenster des Malprogramms. Male mit dem Ellipse-Werkzeug und dem Pinsel eine Assel. Auf dem Kopf sind zwei Fühler in einer besonderen Farbe (z.B. orange), die sonst nicht in deiner Assel vorkommt. Die Farbe hat eine besondere Bedeutung. Dazu später mehr. Wichtig ist, dass der Kopf mit den Fühlern nach rechts zeigt; denn das ist die Bewegungsrichtung.

In Wirklichkeit sind Asseln grau und haben sieben Beinpaare. Aber ich denke, ein bisschen künstlerische Freiheit ist erlaubt. Klicke auf OK, wenn die Assel fertig ist. Dann machst du eine Kopie des ersten Kostüms. Bearbeite die Kopie und verändere die Position der Beine und Fühler ein wenig. Wenn man diese Kostüme abwechselnd zeigt, sieht es aus als würden sich die Beine bewegen und die Fühler wackeln.

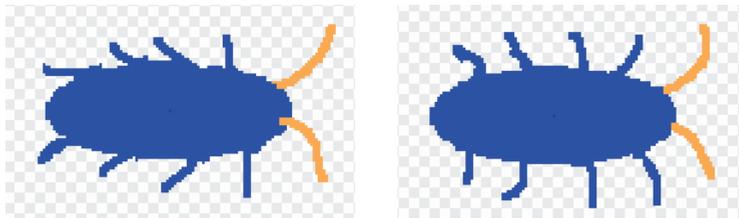


Abb. X.2: Die beiden Kostüme der digitalen Assel

Gibt der Figur den Namen Assel.

DAS SKRIPT

Die Asse1-Figur hat nur ein einziges Skript, das die gesamte Bewegung bewerkstelligt.



Abb. X.3: Das Skript der krabbelnden Asse1.

SO FUNKTIONIERT'S

- 1 Die Asse1 startet immer an derselben Stelle.
- 2 Die Asse1 dreht sich ein kleines bisschen nach rechts (wenn die Zufallszahl negativ ist) oder links. Hier wird für die unregelmäßige Zickzackbewegung gesorgt.
- 3 Die Asse1 soll vom Bildrand abprallen.
- 4 Hier wird für die Vorwärtsbewegung gesorgt. Durch den Kostümwechsel bewegen sich die Beine.

SUCHE NACH EINEM VERSTECK

Im zweiten Schritt legen wir einige Blätter auf die Bühne. Die Asse1 soll sich nun so verhalten:

Wenn sie sich auf der freien grünen Fläche im vollen Sonnenlicht befindet, ist sie ganz aufgeregt und läuft wild herum. Sobald sie aber ein Blatt gefunden hat und sich darunter verstecken kann, wird sie langsam und ruhig. Sie bewegt sich die ganze Zeit, aber die meiste Zeit bleibt sie unter einem Blatt.

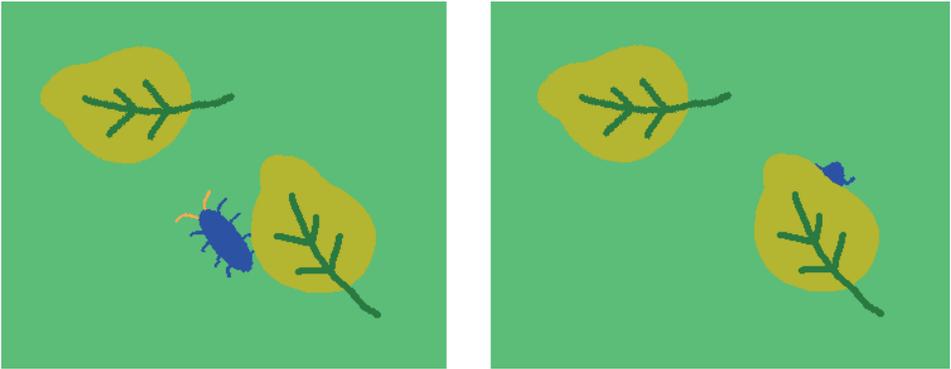


Abb. X.4: Die Assel versteckt sich unter einem Blatt.

DAS BLATT

Zeichne eine neue Figur, die ein Blatt darstellt. Mache von dieser Figur eine oder mehrere Kopien und verteile sie auf der Bühne. Die Szenerie sieht interessanter aus, wenn du die Blätter etwas verdrehst. Das geht so:

- ◇ Wähle die Figur, die du drehen möchtest.
- ◇ Klicke im Informationsfeld unter der Bühne auf die Richtung.
- ◇ Stelle einen neuen Winkel ein.

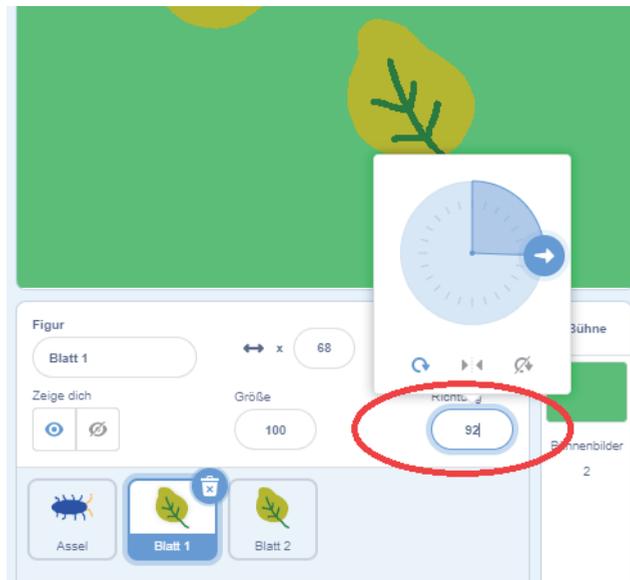


Abb. X.5: Eine Figur drehen

Wichtig ist, dass die Blätter auf dem Bildschirm *vor* der Assel liegen, damit sich die Assel darunter verstecken kann. Die Figur, die zuletzt auf der Bühne mit der Maus bewegt worden ist, ist immer ganz vorne. Um also ein Blatt nach vorne zu bringen, brauchst du es nur ein wenig zu verschieben. Oder aber du klickst auf den Baustein komme nach vorn in der Sammlung Aussehen.

DIE SKRIPTE FÜR DIE ASSEL

Wie erreicht man, dass sich die Assel vorzugsweise unter einem Blatt aufhält? Überlegen wir zunächst einmal, wie sich die Assel in den beiden Situationen verhalten müsste, die in Abbildung X.6 dargestellt sind.

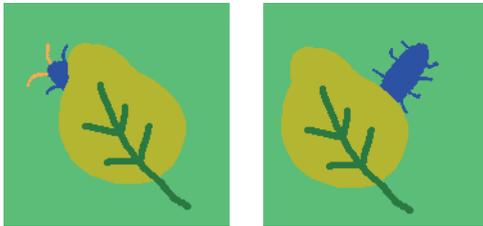


Abb. X.6: Zwei Fälle: 1) Die Assel kommt unter einem Blatt hervor (links).
2) Die Assel hat ein Blatt gefunden (rechts).

Erste Situation. Die Assel verlässt aus Versehen die schützende Deckung eines Blattes. Da sie sich zumindest teilweise nach dem Zufallsprinzip bewegt, kann das schon einmal vorkommen. Jetzt müsste sie sich eigentlich umdrehen, um wieder zum Blatt zurückzukehren.

Zweite Situation. Nach langer Suche mit raschen Bewegungen und großen Richtungswechseln hat die Assel endlich ein Blatt gefunden. Sie hat es gerade mit ihrem Fühler berührt. Jetzt muss sie ihr Bewegungsmuster ändern. Bloß kein großer Richtungswechsel! Die Richtung stimmt, denn vor ihr liegt ein Blatt. Auch sollte sie jetzt langsamer werden. Denn über ihr ist ein schützendes Blatt, das sie nicht so schnell verlassen will. Hier will sie sich möglichst lange aufhalten.

Die Assel braucht also zwei Bewegungsmuster:

- ◇ Schnelle Bewegung (große Schritte) und große zufällige Richtungsänderungen.
- ◇ Langsame Bewegung (kleine Schritte) und geringe zufällige Richtungsänderung.

Die Assel bewegt sich nach dem ersten Muster, wenn sie gerade im Freien ist. Spürt sie mit ihren Fühlern ein Blatt, bewegt sie sich nach dem zweiten Muster.

Für jedes Bewegungsmuster schreibst du ein eigenes Skript. Je nachdem, ob die Fühler gerade das typische Grün eines Blattes erspüren oder nicht, wird das zweite bzw. das erste Muster angewendet.

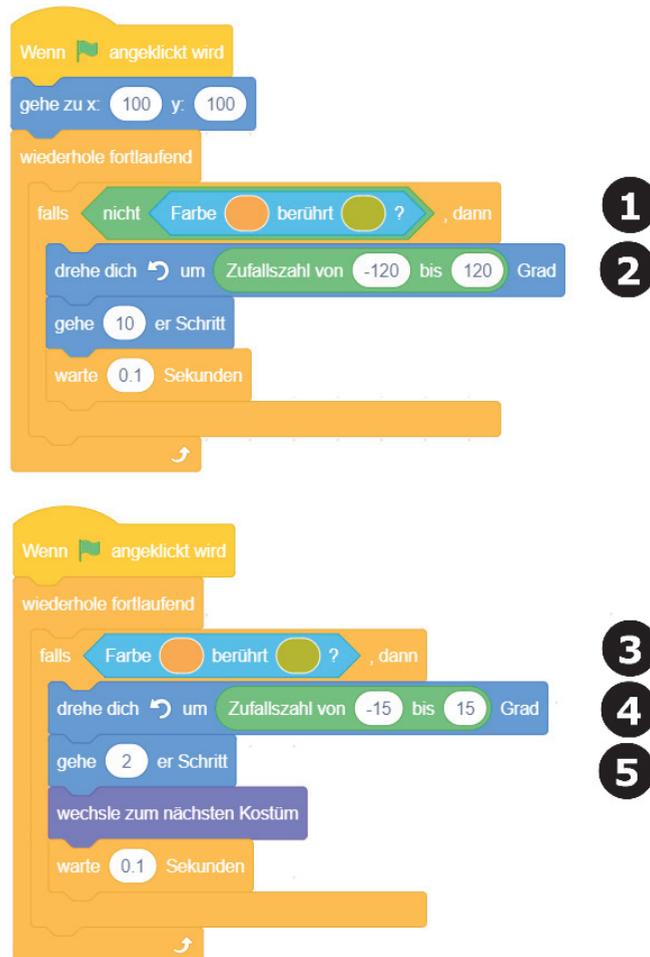


Abb. X.7: Zwei Skripte für zwei unterschiedliche Bewegungsmuster

SO FUNKTIONIERT'S

- ❶ Mit dem Baustein  programmierst du den Farbsensor. Die erste Farbe ist die Farbe der Fühler, die zweite Farbe ist die grüne Farbe des Blattes. Um eine Farbe auszuwählen, klickst du zuerst auf die Farbe auf dem Programmbaustein. Dann sieht der Mauszeiger aus wie eine Lupe. Wenn du nun die gewünschte Farbe auf der Bühne anklickst (das Orange der Fühler oder das Grün eines Blattes), übernimmt sie der Programmbaustein. Die restlichen Anweisungen des Skriptes werden nur ausgeführt, wenn die Assel *nicht* ein Blatt berührt.
- ❷ Die Assel dreht sich hektisch um einen großen zufälligen Winkel nach rechts oder links (maximal 120 Grad).
- ❸ Die folgenden Anweisungen werden ausgeführt, wenn die Assel unter einem Blatt ist.

- 4 Nur eine geringe zufällige Richtungsänderung (um maximal 15 Grad) findet statt.
- 5 Die Assel ist langsam. Sie bewegt sich nur um Zweierschritte nach vorne.

WIE FUNKTIONIERT EIN GASCHROMATOGRAPH?

Mit einem Gaschromatographen finden Chemiker die Zusammensetzung von Stoffgemischen heraus. Der Gaschromatograph besteht aus einer langen, dünnen Röhre (auch Säule genannt), die mit einem porösen Material gefüllt ist. Durch die Röhre wird ein Gas gepumpt. Am Eingang spritzt man die Probe in den Gasstrom – z.B. Feuerzeuggas, das meist ein Gemisch aus Butan und Propan ist. Die Butanmoleküle kommen etwas langsamer durch die Säule als die kleineren Propanmoleküle. Am Ausgang der Säule ist ein Detektor. Er liefert ein Signal an einen Computer, der eine Messkurve erstellt.

WAS ZEIGT DIE ANIMATION?

Nun kann man in die Säule des Gaschromatographen nicht hineinsehen. Aber das, was da drinnen passiert, kannst du in einem animierten Modell darstellen. Genau darum geht es in diesem Projekt. Abbildung X.8 zeigt Screenshots aus der Animation.

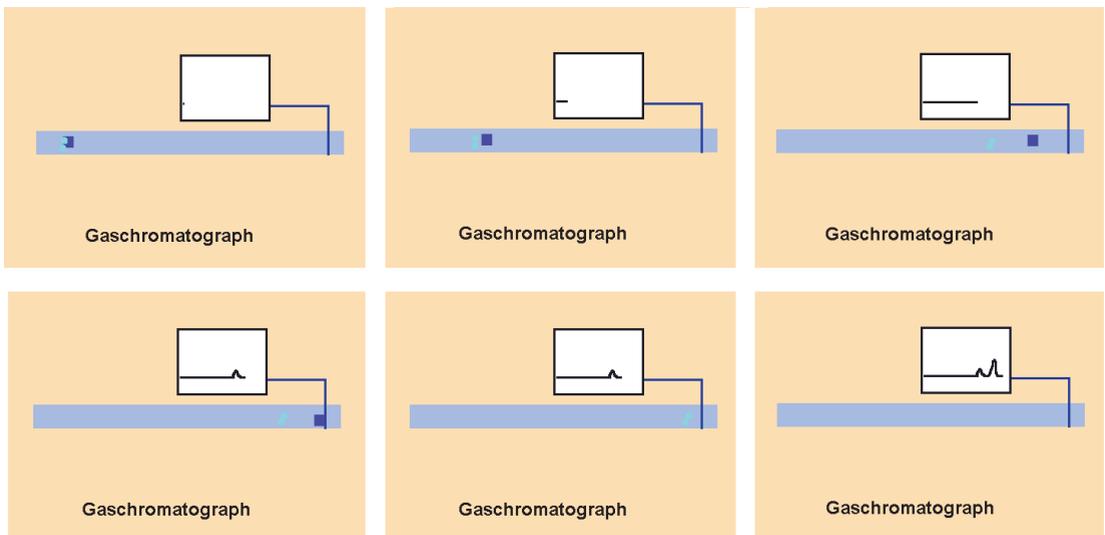


Abb. X.8: Screenshots aus einem Scratch-Video, das erklärt wie ein Gaschromatograph funktioniert.

Man sieht zwei unterschiedliche Arten von Molekülen: zwei langsame Butanmoleküle (modellhaft als Kreisflächen dargestellt) und ein schnelleres Propanmolekül (Quadrat). Sie wandern durch die Säule und trennen sich dabei allmählich in zwei

Gruppen auf. Das Display des Detektors zeigt zunächst eine waagrechte Linie. Wenn Moleküle den Detektor treffen, entsteht ein »Hügel« (Peak). Dieses Diagramm nennt man Gaschromatogramm. An den Peaks kann man erkennen, welche Stoffe in der untersuchten Probe enthalten sind.

DIE BÜHNE

Das Projekt besteht aus dem Hintergrundbild der Bühne und vier Figuren. Die Figur Kurve zeigt die Entwicklung des Chromatogramms. Die drei anderen Figuren stellen wandernde Moleküle dar (Propan, Butan1, Butan2)



Abb. X.9: Überblick: Die Figuren des Projekts

Die Bühne besitzt kein Skript. Das Bühnenbild der Bühne zeigt alles, was sich *nicht* verändert.

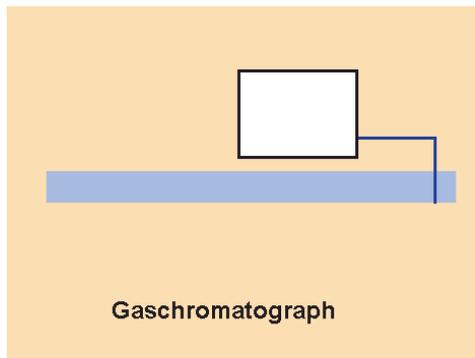


Abb. X.10: Das Bühnenbild

DAS DIAGRAMM

Das Diagramm wird durch die Figur Kurve mit vielen Kostümen erzeugt. Jedes Kostüm zeigt das Diagramm zu einem bestimmten Zeitpunkt. Das erste Kostüm erhält den Namen Start. Die Namen der anderen Kostüme sind unwichtig.



Abb. X.11: Die Kostüme der Figur Kurve zeigen die Entwicklung des Gaschromatogramms.

Die gesamte Animation dauert sechs Sekunden. Nach dem Start des Programms wird das erste Kostüm gezeigt. Danach wird jede Sekunde das Bild gewechselt.

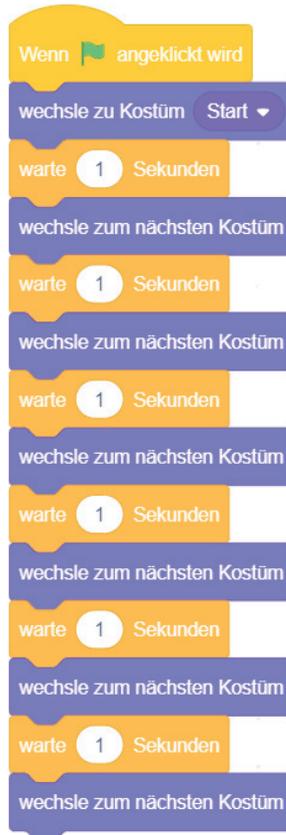


Abb. X.12: Das Skript der Figur Kurve

DIE MOLEKÜLE

In diesem Beispiel gibt es drei Moleküle, die durch die Säule des Gaschromatographen wandern, zwei runde (etwas langsamer) und ein quadratisches (etwas schneller). Für jedes Molekül gibt es eine eigene Figur mit einem eigenen Kostüm und einem eigenen Skript. Die Skripte sind freilich sehr ähnlich.

Da die Moleküle am Ende versteckt werden, müssen sie sich zuerst sichtbar machen. Dann werden sie auf ihre Anfangsposition gesetzt. Die Anfangspositionen dürfen

nicht genau gleich sein, damit man sieht, dass es mehrere Moleküle sind. Dann gleiten sie zu ihrer Endposition am Detektor des Gaschromatographen. Die Zeit ist beim schnelleren Propan etwas kürzer (5 Sekunden, siehe eingekreiste Stelle in Abbildung X.13) als beim langsameren Propan. Zum Schluss verschwinden die Moleküle.

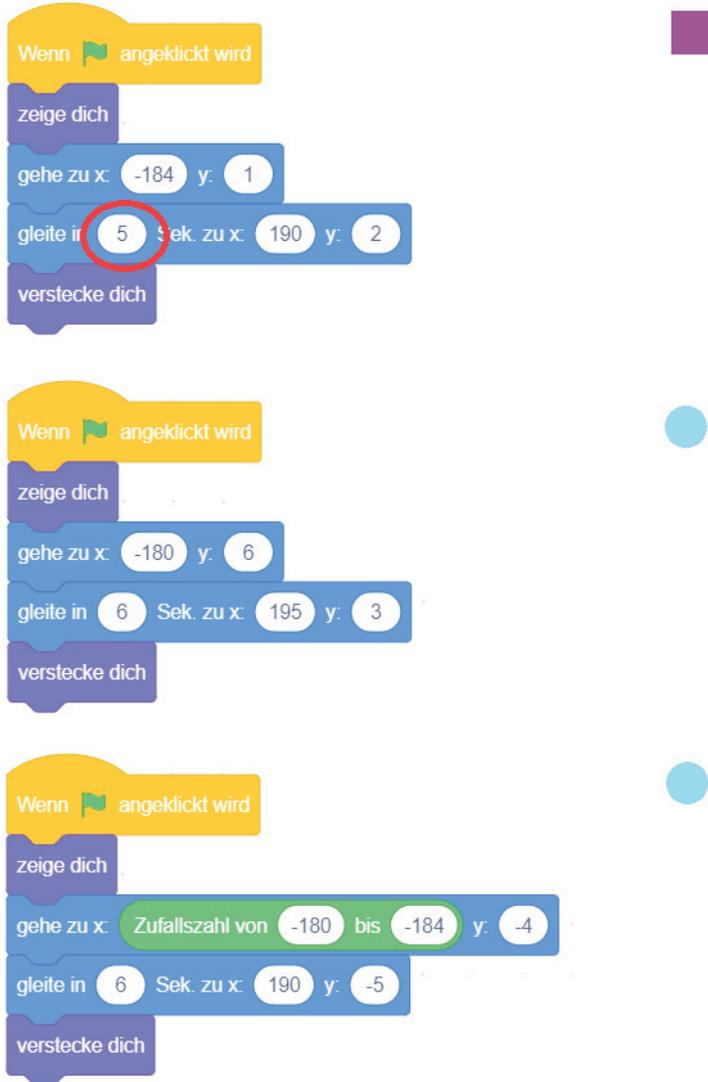


Abb. X.13: Kostüme und Skripte der Moleküle.

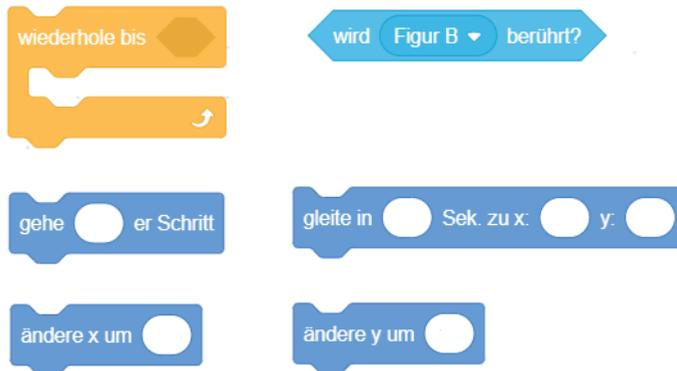
WEITERE IDEEN

Animierte Modelle zur Erklärung von sichtbaren und unsichtbaren Vorgängen spielen in der Wissenschaft eine große Rolle. Vielleicht möchtest du dein nächstes Referat mit einer selbstgemachten Animation aufpeppen. Frage deine Lehrerin oder deinen Lehrer. Themen gibt es genug.

- ◇ Chemische Reaktionen (z.B. Wasserstoffmoleküle stoßen mit Sauerstoffmolekülen zusammen. Dabei entstehen Wassermoleküle.)
- ◇ Die Kettenreaktion im Atomreaktor
- ◇ Wie funktioniert ein Enzym?
- ◇ Wie funktioniert ein Transistor?

FRAGEN

1. Auf der Bühne liegen zwei Figuren. Wie kann man dafür sorgen, dass beim Programmlauf (grüne Flagge ist angeklickt worden) Figur A immer im Vordergrund ist?
2. Eine Figur soll sich bewegen, bis sie eine zweite Figur berührt, die sich auch bewegt. Welchen der folgenden Blöcke kann man für diese Aufgabe *nicht* verwenden?



3. Ein Auto, das sich von links nach rechts bewegt, soll immer schneller werden. In dem Skript in der Abbildung fehlt eine Anweisung. Welche der vier Anweisungen führt nicht zu einer Erhöhung der Geschwindigkeit?

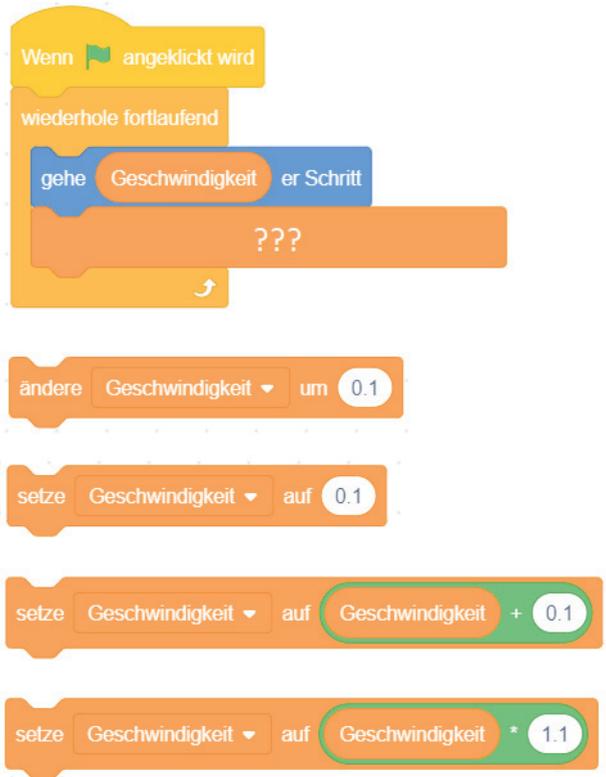
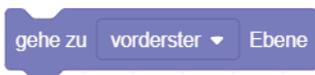


Abb. X.14: Skript des Roboters mit vier Lücken.

ANTWORTEN ZU DEN FRAGEN

1. Die Anweisung



setzt das Objekt in den Vordergrund.

2. Die Anweisung



ist ungeeignet, weil sie zu einem *festen* Zielpunkt führt. Aber hier bewegt sich das Ziel.

3. Die Anweisung setze Geschwindigkeit auf 0.1 führt nicht zu einer Veränderung der Geschwindigkeit. Sie bleibt auf 0.1.



Ein Server ist ein Computersystem, das ständig läuft und Aufträge erledigt. Ein Webserver ermöglicht den Zugriff auf Webseiten über das lokale Netz oder das Internet. Der Raspberry Pi verbraucht wenig Strom und ist deshalb die ideale Hardware für einen Webserver. In diesem Kapitel erfährst du zunächst, wie du einen Apache-Server installierst und als Webserver nutzt. Dann geht es um dynamische Webseiten, die z. B. die Uhrzeit, das Livebild der Kamera oder die Temperatur anzeigen. Außerdem entwickeln wir interaktive Webseiten mit Schaltflächen und Eingabefeldern, über die man Geräte fernsteuern kann.

RASPBERRY PI ALS WEBSERVER

Das WWW (World Wide Web) besteht aus vielen Milliarden Webseiten, die durch Links (engl. *link* = Verbindung) miteinander verknüpft sind. Du kennst das: Ein Link (oder Hyperlink) ist ein Bild oder ein Textstück, das man anklicken kann. Wenn man mit der Maus auf den Link zeigt, sieht der Mauszeiger plötzlich aus wie eine Hand. Drückt man die linke Maustaste, erscheint eine neue Webseite. Dieses Netzwerk aus verbundenen Seiten nennt man Hypertext.

Was passiert nun, wenn du im Internet eine Webseite aufrufst? Dein Browser sendet eine Anfrage an einen Webserver, der auf einem entfernten Computer im Internet läuft. Der Server bearbeitet deine Anfrage und sendet dir als Antwort eine Webseite. Die wird dann im Browser angezeigt.

Die Übertragung der Webseiten und die Arbeitsweise der Webserver werden durch das *Hypertext Transfer Protocol* (HTTP) geregelt. Das bedeutet auf Deutsch Hypertext-Übertragungsprotokoll.

Für den Aufbau der einzelnen Webseiten gibt es ein spezielles Format, eine Art Sprache, die man HTML nennt (engl. *Hypertext Markup Language*, Hypertext-Auszeichnungssprache).

Wenn du deinen Raspberry Pi zu einem Webserver machen möchtest, musst du zwei Dinge tun, die anschließend Schritt für Schritt erklärt werden:

- ◇ Du installierst einen HTTP-Server (z.B. Apache) und sorgst dafür, dass er ständig läuft.
- ◇ Du speicherst in einem speziellen Verzeichnis HTML-Dateien, die das Informationsangebot deines Webserver beinhalten.

In diesem Abschnitt erfährst du, wie das geht.

DEN APACHE-SERVER INSTALLIEREN

Wahrscheinlich der bekannteste HTTP-Server ist der Apache-Server. Du installierst ihn im LXTerminal-Fenster mit folgendem Kommando:

```
$ sudo apt-get install apache2
```

Nach der Installation wird das Serverprogramm sofort gestartet. Auch nach jedem Neustart des RPi läuft ab jetzt immer automatisch der Apache-Server im Hintergrund, ohne dass du etwas merkst. Dein RPi ist jetzt ein Webserver.

Probiere ihn direkt einmal aus. Klicke auf dem Desktop oben links auf  und starte so den Webbrowser *Chromium*. Gib in das Adressfeld `http://localhost/index.html` ein. Es erscheint eine Webseite wie in Abbildung Y.1. Sie besagt: »Es funktioniert! Das ist die voreingestellte Webseite für den Server. Die Server-Software läuft, aber bis jetzt wurde noch kein Inhalt hinzugefügt.«

Das, was du in das Adressfeld eines Browsers einträgst, nennt man *Uniform Resource Identifier* (einheitlicher Bezeichner für Ressourcen) oder kurz URI. Der URI einer Webseite setzt sich aus mehreren Teilen zusammen:

- ◇ `http://` Die Buchstaben stehen für *Hypertext Transportprotokoll*. Dieser Teil des URI bringt zum Ausdruck, dass es sich um eine Webseite handelt.

- ◇ Name des Webservers oder IP-Nummer. Der Name `localhost` steht immer für den Webserver, der auf demselben Rechner läuft wie das Browser-Programm.
- ◇ Verzeichnis und Dateiname (Pfad) der Datei mit der Webseite.

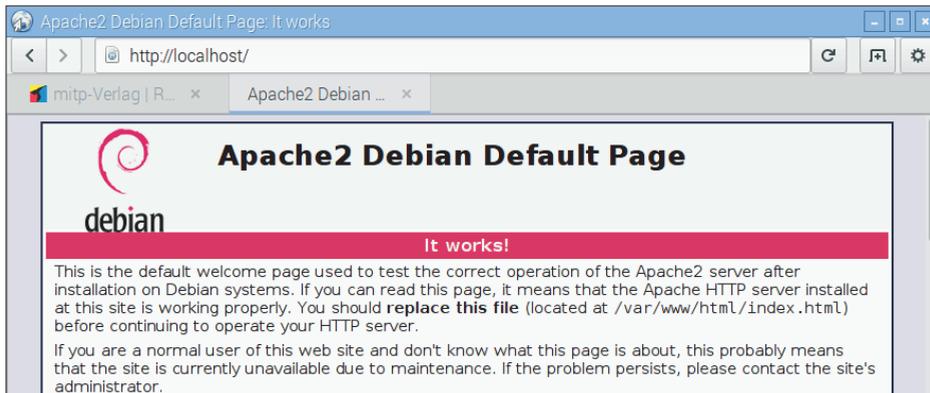


Abb. Y.1: Voreingestellte Startseite des Apache-Servers

Der Dateiname `index.html` hat eine Besonderheit. Man kann ihn auch weglassen. Das heißt, du kannst in das Adressfeld des Browsers auch einfach nur `http://localhost/` eingeben (wie in Abbildung Y.1). Die Seite `index.html` ist die Startseite einer Website, die aus mehreren einzelnen Seiten besteht, die durch Links verbunden sind.

Aber wo ist diese Datei `index.html` gespeichert? Das Installationsprogramm des Apache-Servers hat einige neue Verzeichnisse angelegt. Besonders wichtig ist das Verzeichnis `/var/www/html`. Dort sind alle HTML-Seiten gespeichert, die vom Apache-Server angezeigt werden. Und dort findest du auch (bisher als einzige Datei) `index.html`.

DIE EIGENE WEBSEITE IM LOKALEN NETZ

Ersetze die vorgegebene Startseite des Apache-Servers durch eine selbst gemachte Seite! Webserver dienen dazu, Informationen zu verbreiten. Wenn dein Raspberry Pi in das lokale Funknetz eingebunden ist, kann jeder andere Nutzer des *lokalen Netzes* (aber sonst niemand) deine Webseite lesen. (Dazu später mehr.) Die Reichweite ist auf das lokale Netz beschränkt. Deine Seite ist *nicht* im Internet sichtbar.

Auf deiner Webseite kannst du Dinge veröffentlichen, die für die Menschen in deiner Umgebung interessant sind: Termine, Fotos oder aktuelle Neuigkeiten. Abbildung Y.2 zeigt ein ganz einfaches Beispiel. Mit einer Überschrift, etwas Text und einem Hyperlink, den man anklicken kann. Du denkst dir natürlich etwas anderes aus.

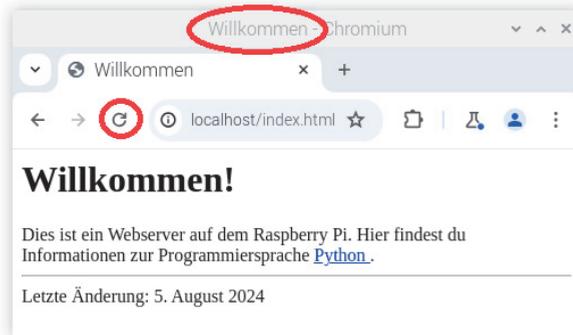


Abb. Y.2: Eine neue Startseite; im Rahmen steht der Titel der Seite (eingekreist)

EIN NEUER EIGENTÜMER

In dem Verzeichnis `/var/www/html` stehen die Webseiten, die der Apache-Server zeigt. Zur Zeit kann nur der Administrator (`root`) das Verzeichnis `/var/www/html` und die Datei `/var/www/html/index.html` verändern. Mit dem `ls`-Befehl kannst du das nachprüfen:

```
$ ls -l /var/www/html
```

Hier steht hinter dem Befehl `ls` noch `-l` (kleiner Buchstabe l). Das `-l` nennt man ein *Argument*. Damit wird der Befehl `ls` ein wenig abgewandelt. Er liefert jetzt zusätzliche Informationen über das Verzeichnis, das (als zweites Argument) dahinter steht. Wenn du `↵` gedrückt hast, erscheint in der nächsten Zeile eine Auflistung aller Dateien im Verzeichnis `/var/www/html`. Da es zurzeit nur eine Datei in diesem Verzeichnis gibt, besteht diese Auflistung nur aus einer einzigen Zeile:

```
-rw-r--r-- 1 root root 177 Nov 8 11:47 index.html
```

Abbildung Y.3 versucht zu erklären, was die Buchstaben und Zeichen bedeuten.

Im ersten Teil werden die Rechte an der Datei `/var/www/html/index.html` beschrieben. Grundsätzlich gibt es drei Zugriffsrechte für Dateien:

- ❖ Das Leserecht `r` (*read* = lesen) bedeutet, dass der Inhalt der Datei (z.B. ein Bild oder ein Text) gelesen oder irgendwie dargestellt werden kann. Der Inhalt darf aber nicht geändert werden.
- ❖ Das Schreibrecht `w` (*write* = schreiben) bedeutet, dass diese Datei geändert oder gelöscht werden darf.
- ❖ Das Ausführungsrecht `x` (*execute* = ausführen) bedeutet, dass die Datei ausgeführt werden darf. Das ergibt natürlich nur bei Programmdateien einen Sinn.

Bei Verzeichnissen bedeuten die Zugriffsrechte *r*, *w*, *x* etwas anderes:

- ◇ Das Leserecht *r* (*read* = lesen) bedeutet, dass der Inhalt des Verzeichnisses aufgelistet werden kann.
- ◇ Das Schreibrecht *w* (*write* = schreiben) bedeutet, dass in diesem Verzeichnis Dateien erzeugt, verändert und gelöscht werden können.
- ◇ Das Ausführungsrecht *x* (*execute* = ausführen) bedeutet, dass man in das Verzeichnis wechseln darf, was aber noch nicht bedeutet, dass man sich auf den Inhalt auflisten darf.

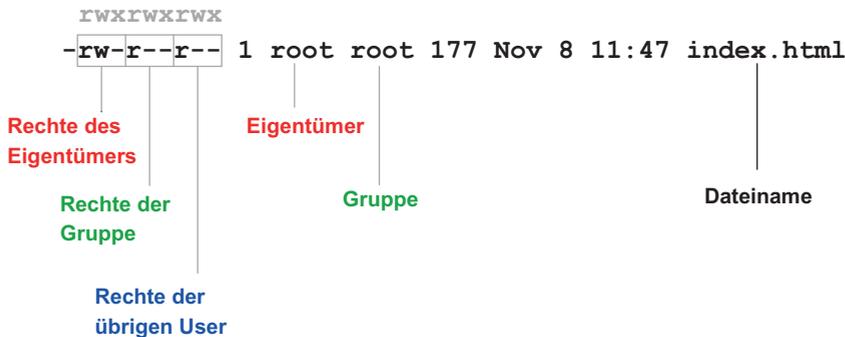


Abb. Y.3: Berechtigungen

Nun gibt es auch drei Personengruppen, denen Zugriffsrechte zugewiesen werden können: der Eigentümer (ein einzelner Benutzer), die Gruppe (mehrere Benutzer des Computers) und die übrigen Benutzer.

Jeder dieser Personengruppen sind nun durch die Buchstaben *r*, *w* und *x* Zugriffsrechte zugewiesen. Wenn das Recht erteilt ist, steht da der betreffende Buchstabe, ansonsten ist das ein Strich.

Bei der Datei `/var/www/html/index.html` hat der Eigentümer das Lese- und Schreibrecht. Gruppenmitglieder und die übrigen Benutzer dürfen nur lesen.

Die Zahl nach der Beschreibung der Zugriffsrechte (hier eine 1) ist die Anzahl der Verweise im Dateisystem auf diesen Eintrag.

Das erste `root` bezeichnet den Eigentümer der Datei. Das ist der Administrator. Das zweite `root` ist die Gruppe, der die Datei gehört. Hier heißt die Gruppe zufälligerweise auch `root`.

Wenn du als User `pi` Webseiten einrichtest, brauchst du das Recht, die Datei zu verändern (Schreibrecht *write*). Die anderen sollen aber kein Schreibrecht haben. Deshalb ist es besser, wenn der Benutzer `pi` neuer Eigentümer sowohl vom Verzeichnis `/var/www/html` als auch von der Datei `/var/www/html/index.html` wird. Mit dem Unix-Befehl `chown` (*change owner* = ändere den Eigentümer) kannst du die

Besitzverhältnisse ändern: Du musst diesen Befehl in der Rolle des Administrators starten (sudo).

```
$ sudo chown pi /var/www/html /var/www/html/index.html
```

Prüfe nach, wer jetzt der Besitzer der Datei ist:

```
$ ls -l /var/www/html
-rw-r--r-- 1 pi root ...
```

DIE HTML-SEITE

Eine HTML-Seite ist nichts weiter als eine Textseite. Starte IDLE oder den Texteditor MOUSEPAD (START|ZUBEHÖR|TEXT EDITOR). Gib folgenden Text ein:

```
<html>
  <head>
    <title> Willkommen</title>
  </head>
  <body>
    <h1>Willkommen!</h1>
    Dies ist ein Webserver auf dem Raspberry Pi.
    Hier findest du Informationen zur Programmiersprache
    <a href="http://www.python.org"> Python </a>.
    <hr/>
    Letzte Änderung: 5. August 2024
  </body>
</html>
```

Speichere die Seite unter dem Pfad `/var/www/html/index.html`. Damit wird die alte Startseite überschrieben. Das Abspeichern geht so:

➤ Klicke auf DATEI|SPEICHERN UNTER

➤ Gib in das leere Feld neben NAME: den Pfad ein: `/var/www/html/index.html`

In dem Browserfenster lädst du die Seite erneut, indem du den gebogenen Pfeil rechts neben dem Adressfeld anklickst (in Abbildung Y.2 eingekreist). Jetzt siehst du die selbst gestaltete Webseite.

Wie funktioniert das? Der HTML-Text enthält sogenannte Tags (sprich: »Tägs«) in spitzen Klammern. Es sind Markierungen, die das Format eines Textbereiches festlegen. Die Tags treten in der Regel paarweise auf. Zu einem öffnenden Tag `<...>` gehört ein schließendes Tag, das mit einem Schrägstrich beginnt `</...>`. Der ganze HTML-Text beginnt mit `<html>` und endet mit `</html>`. Er besteht aus zwei Teilen:

◆ Der Kopf wird durch `<head> ... </head>` markiert. Er enthält unsichtbare Informationen über das Dokument. Hier steht im Kopf der Titel. Er erscheint nicht in

der Ansicht des Dokumentes, aber der Titelleiste des Webbrowsers (im Bild eingekreist).

- ◇ Der Körper wird durch `<body> ... </body>` markiert. Hier steht das, was man sehen kann.

Hier noch einige Hinweise zu den Tags im Body des HTML-Dokumentes:

- ◇ Mit `<h1> ... </h1>` wird eine große Überschrift erzeugt. Alles, was zwischen den beiden Tags steht, wird in großen Buchstaben geschrieben.
- ◇ Mit ` ... ` wird ein Hyperlink festgelegt. Das, was zwischen den Tags steht, erscheint im Browser farbig und unterstrichen. Wenn du mit der Maus auf einen solchen Hyperlink klickst, wechselt der Browser zu einer neuen Webseite. Deren Adresse ist in dem Attribut `href="..."` angegeben. In diesem Beispiel führt der Hyperlink zur Python-Homepage.
- ◇ Umlaute (ä, ö, ü) werden in HTML durch Zeichenfolgen codiert: (ä, ö und ü). Mit Großbuchstaben funktioniert das genau so (Ä, Ö und Ü). Das ß übrigens wird durch ß dargestellt.
- ◇ Das Tag `<hr/>` stellt eine waagerechte Linie dar.

DEN WEBSERVER IM LOKALEN NETZ NUTZEN

Der Raspberry Pi ist mit dem lokalen Netz (einem LAN oder einem WLAN) verbunden. Du kannst auch von anderen Rechnern in dem gleichen lokalen Netz auf den Webserver zugreifen und Informationen anfordern.

Auf dem Raspberry Pi heißt der Webserver `localhost`. Auf anderen Computern verwendest du die IP-Nummer des Raspberry Pi. Wie findest du die IP-Nummer deines RPi heraus? Öffne ein LXTerminal und gib das folgende Kommando ein:

```
$ ifconfig
```

Die IP-Nummer deines Raspberry Pi steht hinter `inet adresse`. Mehr dazu ist in Kapitel 2.

PROJEKT: WIE SPÄT IST ES? DYNAMISCHE WEBSEITEN

Die ersten Webseiten waren völlig statisch, d.h. HTML-Dokumente, die als Textdateien im Verzeichnis `/var/www/html` gespeichert sind und vom Apache-Server

gesendet werden, wenn jemand danach fragt. Diese Webseiten sind statisch, weil sie immer bleiben, wie sie sind, und sich nicht ändern.

Manchmal ändert sich aber die Information, die auf der Webseite gezeigt werden soll – zum Beispiel die Uhrzeit. Dann muss die HTML-Seite *dynamisch* erzeugt werden, sobald eine Anfrage an den Webserver kommt.

WAS IST EIN CGI-SKRIPT?

Ein CGI-Skript ist ein Programm, das eine HTML-Seite dynamisch erzeugt. CGI heißt *Common Gateway Interface* und ist ein Standard, der den Aufbau von CGI-Skripten und den Umgang des Webserver mit ihnen regelt. Das CGI-Skript wird aufgerufen wie eine gewöhnliche Webseite. Der HTTP-Server sendet aber keine gespeicherte Textdatei, sondern er lässt das CGI-Skript ausführen und sendet die Ausgabe dieses Skriptes. In diesem Abschnitt entwickelst du eine Webseite, die das Datum und die Uhrzeit anzeigt, wenn sie aufgerufen wird.



Abb. Y.4: Eine dynamisch erzeugte Webseite mit Datum und Uhrzeit

Die HTML-Seite wird von einem Python-Skript erzeugt, das unter dem Namen `zeit.cgi` gespeichert ist. Ja, es ist ein Python-Skript, aber der Dateiname endet nicht auf `.py`, sondern auf `.cgi`. Damit wird ein Python-Programm zum CGI-Skript.

DAS VERZEICHNIS CGI-BIN

Alle CGI-Skripte sind in einem speziellen Verzeichnis gespeichert. Bei deinem Apache-Server ist das `/usr/lib/cgi-bin`. Zunächst gehört es aber dem Administrator `root`. Der Benutzer `pi` hat kein Schreibrecht und kann in diesem Verzeichnis nichts speichern. Was tun? Am besten ist es, die Besitzverhältnisse zu ändern. Mit dem `chown`-Kommando wird `pi` zum Eigentümer gemacht:

```
$ sudo chown pi /usr/lib/cgi-bin
```

Den Apache-Server auf CGI-Skripte vorbereiten

Damit dein Apache-Server CGI-Skripte verarbeiten kann, musst du in einem LXTerminal folgendes Kommando eingeben:

```
$ sudo a2enmod cgi
```

Danach musst du den Server neu starten:

```
$ sudo service apache2 restart
```



DIE PROGRAMMIERUNG

Öffne nun mit IDLE 3 ein neues Editorfenster und schreibe das nachfolgende Skript. Speichere es unter dem Pfad

```
/usr/lib/cgi-bin/zeit.cgi
```

Achte darauf, dass du wirklich `.cgi` anstelle des sonst üblichen `.py` schreibst.

PROGRAMMTEXT

```
#!/usr/bin/python3

from time import asctime
HTML = '''Content-type: text-html; char-set=utf-8

<html>
<head>
    <title>Die Zeit</title>
</head>
<body>
    <h1>Datum und Uhrzeit</h1>
    {}
</body>
</html>''' #2

print(HTML.format(asctime())) #3
```

SO FUNKTIONIERT'S

- #1 Die erste Zeile ist besonders wichtig. Hier wird der Python-Interpreter angegeben, der das Skript ausführen soll. Achtung! In dieser Zeile darf kein Python-Kommentar wie z.B. `#1` stehen.
- #2 Hier wird ein langer String definiert, der über mehrere Zeilen geht (dreifache Anführungszeichen). Es ist das Muster für ein HTTP-Paket, das der Apache-Server versenden soll. In der ersten Zeile wird bestimmt, dass es ein HTML-Text ist. Außerdem wird die Zeichencodierung (`utf-8`) angegeben. Achtung! Die zweite

Zeile *muss* leer bleiben. Danach folgt ein normaler HTML-Text mit einem Platzhalter {}.

- #3 Mit einer `print()`-Anweisung wird das HTTP-Paket ausgegeben und kann dann vom Apache-Server weitergesendet werden. Der Platzhalter {} wird durch einen String mit Datum und Uhrzeit ersetzt.

DAS SKRIPT MIT IDLE 3 TESTEN

Wenn du das Skript eingegeben und gespeichert hast, solltest du es mit RUN|RUN MODULE (oder `F5`) starten, um zu prüfen, ob es auch den richtigen Text produziert.

```
>>> ===== RESTART =====
>>>
Content-type: text/html; char-set=utf-8

<html>
  <head>
    <title>Die Zeit</title>
  </head>
  <body>
    <h1> Datum und Uhrzeit</h1>
    Mon Aug  3 05:55:35 2020
  </body>
</html>
>>>
```

DAS SKRIPT AUSFÜHRBAR MACHEN

Damit das Skript tatsächlich funktioniert, musst du die Programmdatei ausführbar machen. Mit dem Kommando `chmod +x` erhält jeder Benutzer des Systems das Ausführungsrecht (`x`, *execute*).

```
$ sudo chmod +x /usr/lib/cgi-bin/zeit.cgi
```

Mit `ls -l` kannst du prüfen, ob die Rechte richtig eingestellt sind.

```
ls -l /usr/lib/cgi-bin/zeit.cgi
```

Jetzt kannst du auf deinem RPi den Browser *Chromium* öffnen und die dynamische Webseite testen. Gib in das Adressfeld folgenden URI ein:

```
http://localhost/cgi-bin/zeit.cgi
```

Wenn du die Seite mehrmals hintereinander aufrufst, erscheint jedes Mal eine andere Uhrzeit. Dir ist sicher aufgefallen, dass im URI nicht `/usr/lib` vorkommt. Der interne Pfad unterscheidet sich von der Adresse, die ein Besucher der Website benutzt.

Wenn du die Seite von deinem Handy oder von einem anderen Computer aus besuchen möchtest, gibst du anstelle von localhost die IP-Adresse deines Raspberry Pi an. Der URI ist dann z.B.:

```
http://192.168.178.40/cgi-bin/zeit.cgi
```

Du findest die IP-Adresse mit dem Kommando

```
$ ifconfig
```

Mehr dazu in Kapitel 1.

PROJEKT: SPION IM GARTEN

In eine HTML-Seite kann man auch Bilder einbinden. Wenn das Bild ein Live-Kamerabild ist, das regelmäßig aktualisiert wird, hast du eine Webcam.

Für dieses Projekt brauchst du außer dem Raspberry Pi

- ◇ eine mobile Stromversorgung und
- ◇ ein Kameramodul.

Das Ganze kann irgendwo innerhalb der Reichweite des WLANs aufgestellt werden, zum Beispiel in der Nähe der Maulwurfshügel im Garten, um zu beobachten, ob der kleine Graber vielleicht mal an die Erdoberfläche kommt.

Wenn alles funktioniert, kannst du von irgendeinem Computer des lokalen Netzes oder von deinem Handy aus eine Webseite mit dem Live-Bild der Kamera abrufen.

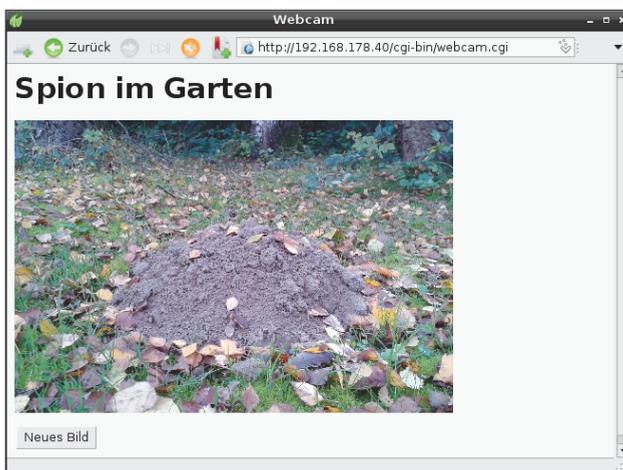


Abb. Y.5: Eine Webcam im Garten

Die Software ist ein CGI-Skript, das eine Webseite mit Schaltfläche erzeugt. Drückt man auf die Schaltfläche, wird mit `raspistill` ein neues Bild aufgenommen.

MEHR RECHTE FÜR CGI-SKRIPTE!

Um mit `raspistill` ein Foto machen zu können, braucht der Apache-Server, der das cgi-Skript ausführt, bestimmte Zugriffsrechte. Um richtig erklären zu können, wie wir gleich diese Zugriffsrechte einstellen, muss ich etwas ausholen.

Auf deinem Raspberry Pi bist du wahrscheinlich als User `pi` eingeloggt. Das Wort `pi` ist dein Username. Du bist aber auch in einer Gruppe namens `pi`. Immer wenn ein neuer User angelegt wird, erzeugt das System auch eine Gruppe mit dem gleichen Namen. Eine Gruppe auf einem Linux-System fasst User zusammen, die bestimmte Rechte haben. Sie haben die Erlaubnis (engl.: *permission*), bestimmte Programme auszuführen und bestimmte Dateien zu lesen oder zu verändern. Als User `pi` bist du in mehreren Gruppen. Mit dem Kommando `groups` kannst du deine Gruppen auflisten lassen:

```
$ groups
pi adm dialout cdrom sudo audio video plugdev games users input netdev
gpio i2c spi
```

Als Mitglied der Gruppe `video` hast du zum Beispiel die Erlaubnis, die Programme zum Zugriff auf die Kamera zu verwenden.

Auch der Apache-Server läuft unter einem Usernamen. Ganz so wie ein menschlicher User. Sein Username ist `www-data`. Und er gehört auch zu einer Gruppe namens `www-data`. Damit der Apache-Server auf die Kamera zugreifen kann, muss er der Gruppe `video` zugefügt werden. Das machst du mit folgendem Kommando:

```
$ sudo adduser www-data video
```

Wenn alles gut gegangen ist, liefert dir Linux eine Erfolgsmeldung:

```
Füge Benutzer »www-data« der Gruppe »video« hinzu ...
Benutzer www-data wird zur Gruppe video hinzugefügt.
Fertig.
```

Jetzt kann der Apache-Server mit `raspistill` ein Bild aufnehmen, aber er kann es noch nicht speichern. Er soll die Bilddatei im Ordner `/var/www/html` speichern. Dieser Ordner ist für die html-Seiten vorgesehen. Sieh dir die Zugriffsrechte in diesem Ordner an:

```
pi@raspberrypi:~ $ ls -l /var/www/html
-rw-r--r-- 1 pi pi 324 Aug 2 21:12 index.html
```

Der User pi ist Besitzer des Ordners. Er kann schreiben (w) und lesen (r). Die Mitglieder der Gruppe pi dürfen nur lesen und alle anderen dürfen auch nur lesen.

Als Nächstes sorgst du dafür, dass die Gruppe www-data das Schreibrecht bekommt. Das machst du in zwei Schritten.

Im ersten Schritt wird der Ordner /var/www/html der Gruppe www-data zugeordnet:

```
$ sudo chgrp -R www-data /var/www/html
```

Der Befehl chgrp bedeutet »change group« (ändere die Gruppe). Das erste Argument -R (von *recursive*, das bedeutet hier in etwa »wiederholend«) bewirkt, dass alle Dateien, die sich jetzt schon in dem Ordner befinden, ebenfalls der Gruppe www-data zugeordnet werden.

Damit die Änderung wirksam wird, startest du den Raspberry Pi neu:

```
$ sudo reboot
```

Im zweiten Schritt bekommt die Gruppe www-data Schreibrecht auf alle Dateien in diesem Ordner:

```
pi@raspberrypi:~ $ sudo chmod -R g+w /var/www/html
```

Hier wird wieder das Argument -R verwendet. Schau dir wieder die Rechte-Zuordnung an:

```
pi@raspberrypi:~ $ ls -l /var/www/html
-rw-rw-r-- 1 pi www-data 324 Aug 2 21:12 index.html
```

An zwei Stellen hat sich etwas geändert: Die Gruppe ist jetzt nicht mehr pi sondern www-data. (Der User pi ist nach wie vor Besitzer.) Und Gruppenmitglieder dürfen auf die Datei schreibend zugreifen. Die gleichen Rechte gelten für alle Dateien, die in diesem Ordner gespeichert werden. Mit diesen Einstellungen funktioniert das folgende cgi-Skript.

PROGRAMM

```
#!/usr/bin/python3
#webcam.cgi
import os

KOMMANDO = 'raspistill -t 100 -w 400 -h 300' \
           + ' -o /var/www/html/bild.jpg -n'           #1

HTML = '''Content-type: text/html; char-set=utf-8
```

```

<html>
  <head>
    <title>Webcam</title>
  </head>
  <body>
    <h1> Spion im Garten </h1>
    <p>
      
    </p>
    <form action="http://192.168.178.40/cgi-bin/webcam.cgi"
      method="POST">
      <input type="Submit" value="Neues Bild" />
    </form>
  </body>
</html>' ' #2

os.system(KOMMANDO) #3
print(HTML) #4

```

SO FUNKTIONIERT'S

#1 Dieses Kommando bewirkt, dass die Kamera ein neues Bild aufnimmt. Es wird immer unter demselben Dateinamen gespeichert. Das Verzeichnis ist das Wurzelverzeichnis des Apache-Servers. Der String wird hier auf zwei Zeilen verteilt, weil es in diesem Buch nicht in eine Zeile passt.

#2 Diese Konstante ist die HTML-Seite, das von dem CGI-Skript immer wieder ausgegeben wird.

Der Tag `` setzt das Bild mit dem angegebenen Pfad an diese Stelle. Das Attribut `alt="Gartenansicht"` bewirkt, dass als Alternative zum Bild der Text `Gartenansicht` erscheint. Der Alternativtext soll den Inhalt des Bildes beschreiben und wird von Leseprogrammen für Blinde genutzt. Der Text wird aber auch dann gezeigt, wenn aus irgendwelchen Gründen die Bilddatei nicht verfügbar ist.

Mit `<form ...> ... </form>` wird ein CGI-Formular definiert. Alle interaktiven Webseiten, bei denen man etwas eingeben oder anklicken kann, besitzen Formulare. Im ersten Tag wird mit dem Attribut `action` festgelegt, was passieren soll, wenn die Schaltfläche angeklickt wird. In diesem Fall wird einfach dasselbe CGI-Skript wieder aufgerufen. (Achte darauf, dass die *richtige IP-Nummer* eingetragen wird.) Außerdem wird noch die Methode der Datenübertragung (`get` oder `post`) festgelegt. Dazu später mehr.

Im Tag `<input type="Submit" .../>` wird die Schaltfläche – der Submit-Button – definiert. Das Attribut `value` legt die Beschriftung fest.

#3 Hier wird ein neues Foto aufgenommen.

#4 Ausgabe der Webseite. Der Text ist immer exakt der gleiche. Aber das gezeigte Bild ist jedes Mal ein anderes, weil der Inhalt der Bilddatei sich ändert.

Speichere das Skript unter dem Dateinamen `webcam.cgi` im Ordner `/usr/lib/cgi-bin` Mit dem folgenden Kommando machst du dein neues cgi-Skript ausführbar:

```
$ sudo chmod +x /usr/lib/cgi-bin/webcam.cgi
```

Du brauchst den mobilen Raspberry Pi nur mit der Stromquelle zu verbinden. Dann startet das Betriebssystem. Der Apache-Server beginnt automatisch mit seiner Arbeit. Die Webcam ist einsatzbereit.

Du kannst die Webseite von einem beliebigen Computer ansehen, der sich im gleichen WLAN wie dein Raspberry Pi befindet.

Was tun bei einem Server-Fehler?

Manchmal arbeitet dein cgi-Skript nicht so, wie du es erwartest. Beim Aufruf des Skripts mit dem Webbrowser erscheint die Meldung »Internal Server Error«. Was tun? Einen Hinweis, wo der Fehler stecken könnte, findest du im Logfile des Apache-Servers. Das ist eine Textdatei, in der der Apache-Server Fehlermeldungen speichert. Öffne diese Datei z.B. im Mousepad-Texteditor mit folgendem Kommando:

```
$ sudo mousepad /var/log/apache2/error.log
```



TETHERING - DAS HANDY ZUM HOTSPOT MACHEN

Nicht an jedem Ort hat man ein WLAN für Experimente mit dem Raspberry Pi zur Verfügung. Falls du ein iPhone oder Android-Smartphone besitzt, kannst du es zum Hotspot machen und über das Handy auf die Webseiten deines RPi zugreifen. Hier als Beispiel eine Anleitung für ein Android-Handy von Samsung. Auf den anderen Smartphones läuft es ähnlich.

- Wähle die Funktion EINSTELLUNGEN. Das Icon sieht aus wie ein Zahnrad.
- Wähle VERBINDUNGEN und dann MOBILE HOTSPOT UND TETHERING.
- Klicke auf MOBILE HOTSPOT.
- Sorge dafür, dass alle Geräte zugreifen können.

Vielleicht hast du schon ein Passwort eingestellt, um z.B. eine Verbindung zu deinem Laptop zu erstellen. Gib für dieses Projekt besser ein neues Passwort ein, das du anderen Leuten weitergeben kannst.

Nun hast du alles vorbereitet und musst nur noch den Hotspot aktivieren. Bei Samsung-Handys gibt es dafür einen Schiebeschalter, den du nach rechts schieben musst.

Jetzt gibt es ein WLAN, das von deinem Handy gesteuert wird. Du kannst aber *auf dem gleichen Handy* auch den Webbrowser benutzen, um auf deine Website zu kommen.

PROJEKT: STRENG GEHEIM! EINE WEBSITE MIT ZUGANGSSCHUTZ

Vielleicht möchtest du, dass deine Webcam oder irgendein anderer Webservice nicht für alle Menschen erreichbar ist, die einen Zugang zum lokalen Netz haben. Für diesen Fall entwickeln wir in diesem Abschnitt eine Website mit einer Login-Seite. Im Browser gibt man – wie in Abbildung Y.6 – eine Adresse ein. Die Webseite enthält ein Eingabefeld, in das der Besucher einen Namen einträgt. Danach klickt er auf die Schaltfläche LOGIN.



Abb. Y.6: Login-Webseite (statisch)

Die Bilder der Webcam werden nur gezeigt, wenn der Name in einer gespeicherten Liste von Namen vorkommt und somit akzeptiert wird (Abbildung Y.7).

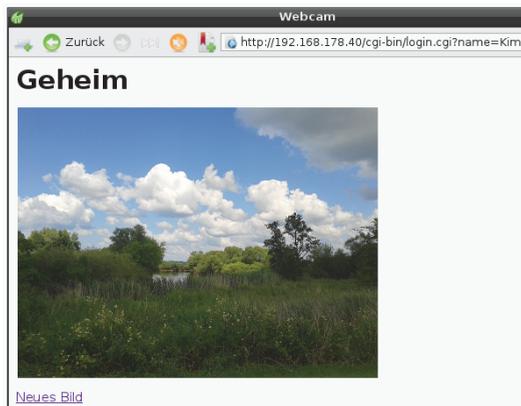


Abb. Y.7: Dynamische Webseite mit Livebild; der Link unten führt wieder zur Login-Seite.

Wenn der Name dem System unbekannt ist, gibt es eine entsprechende Meldung (Abbildung Y.8).

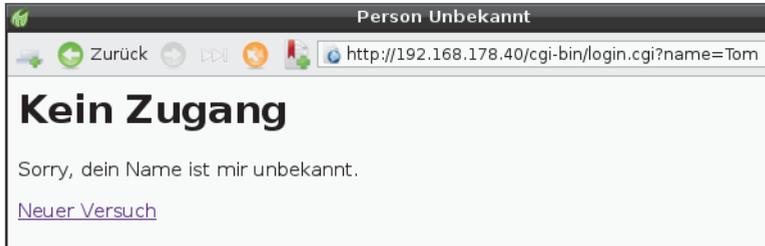


Abb. Y.8: Dynamisch erzeugte Antwortseite; der Link führt wieder zur Login-Seite.

Das Projekt besteht aus zwei Teilen:

- ◇ Eine statische HTML-Seite enthält ein Formular mit einem Eingabefeld für den Namen.
- ◇ Ein CGI-Skript wird von der statischen Login-Seite aus aufgerufen. Es prüft den Namen und liefert eine dynamisch erzeugte Webseite als Antwort zurück.

DIE LOGIN-SEITE

Die Login-Seite ist eine statische HTML-Seite. Speichere sie unter dem Pfad

```
/var/www/html/login.html
```

Sie ist dann im Wurzelverzeichnis des Apache-Servers. Der richtige Speicherort ist sehr wichtig.

DER STATISCHE HTML-TEXT

```
<html>
<head>
  <title>Login</title>
</head>
<body>
  <h1>Nur für Eingeweihte</h1>
  <form method="get"
        action="/cgi-bin/login.cgi">
    Name: <input type="text" name="name"> <br/>
    <input type="Submit" value="Login"/>
  </form>
</body>
</html>
```

SO FUNKTIONIERT'S

Das Dokument enthält wieder ein Formular `<form ...> ... </form>`. Die Übertragungsmethode ist `get`. (Die Erklärung dazu kommt gleich.) Wenn der `SUBMIT`-Button angeklickt wird, wird das CGI-Skript `/cgi-bin/login.cgi` aufgerufen. In dem Formularbereich wird ein einzeliges Eingabefeld definiert:

```
<input type="text" name="name">
```

Wenn der `SUBMIT`-Button angeklickt wird, ruft der Webserver den URI auf, der im `<form ...>`-Tag im Attribut `action="..."` angegeben ist. Da die Methode `get` gewählt worden ist, wird an den URI noch ein Querystring gehängt. Ein Querystring beginnt mit einem Fragezeichen und gibt die Inhalte der Variablen des Formulars an. Hier ist die einzige Variable der Inhalt des Eingabefelds. Wenn in das Eingabefeld `Kim` eingetragen worden ist, lautet der aufgerufene URI plus Querystring so:

```
/cgi-bin/login.cgi?name=Kim
```

DAS CGI-SKRIPT

Speichere das folgende CGI-Skript unter dem Pfad

```
/usr/lib/cgi-bin/login.cgi
```

Vergiss nicht, die Datei ausführbar zu machen:

```
$ sudo chmod +x /usr/lib/cgi-bin/login.cgi
```

PROGRAMM

```
#!/usr/bin/python3
# login.cgi

import cgi, cgitb, os
cgitb.enable() #1

KOMMANDO = 'raspistill -t 100 -w 400 -h 300' + \
           '-o /var/www/html/bild.jpg -n' #2
FREUNDE = ['Tina', 'Kim', 'Dennis'] #3
UNBEKANNT = '''Content-type: text/html

<html>
<head>
  <title> Person Unbekannt </title>
</head>
<body>
  <h1> Kein Zugang</h1>
```

```
    <p>Sorry, dein Name ist mir unbekannt. </p>
    <a href="/login.html"> Neuer Versuch </a>
  </body>
</html>''' #4

WEBCAM = '''Content-type: text/html; char-set=utf-8

<html>
<head>
  <title>Webcam</title>
</head>
<body>
  <h1> Geheim</h1>
  <p>
    
  </p>
  <a href="/login.html"> Neues Bild </a>

</body>
</html>''' #5

form = cgi.FieldStorage() #6
person = form.getvalue('name') #7

if person in FREUNDE: #8
    os.system(KOMMANDO)
    print(WEBCAM)
else:
    print(UNBEKANNT)
```

SO FUNKTIONIERT'S

- #1 Das Modul `cgitb` bietet eine Hilfe bei der Fehlersuche. Bei CGI-Skripten bekommst du ja normalerweise keine Fehlermeldung des Python-Interpreters. Das wird anders, wenn dein Programm diese Anweisung enthält. Wenn jetzt etwas nicht stimmt, wird dir im Browser eine Webseite mit Fehlermeldungen angezeigt.
- #2 Der String enthält das Kommando zum Aufnehmen eines Bildes. Es wird im Wurzelverzeichnis des Apache-Servers gespeichert.
- #3 Eine Liste mit den Namen der Leute, die auf die geheime Seite zugreifen dürfen.
- #4 Das ist die Webseite, die gesendet wird, falls das Login nicht erfolgreich war. Mit dem Tag ` ... ` wird ein Link zur Login-Seite (statische HTML-Seite) hergestellt.
- #5 Dies ist die Webseite, die zurückgesendet wird, falls das Login erfolgreich war. Sie zeigt das aktuelle Kamerabild. Unten ist wieder ein Link zur Login-Seite.
- #6 Das Modul `cgi` braucht man, wenn man die Eingabedaten einer Seite mit Formular verarbeiten will. Hier wird ein `cgi.FieldStorage`-Objekt erzeugt. Es enthält

alle Daten des Formulars, die im Querystring übergeben worden sind. In diesem Fall ist das nicht allzu viel – nur ein eingegebener Name.

- #7 Hier wird der Wert der Variablen `name` aus dem Querystring abgefragt und der Variablen `person` zugewiesen.
- #8 Wenn der Name in der Liste `FREUNDE` vorkommt, wird die Webseite mit dem Kamerabild geliefert, ansonsten gibt es eine Fehlermeldung.

PROJEKT: ÜBER EINE WEBSEITE EINE LED STEuern

Über eine interaktive Webseite kannst du von einem anderen Computer aus Geräte steuern, die an den GPIO des Raspberry Pi angeschlossen sind:

- ◇ einen Motor, der z. B. eine Kamera bewegt
- ◇ einen elektrischen Türöffner
- ◇ eine Klingel oder
- ◇ Leuchtdioden

Als Beispiel machen wir etwas ganz Einfaches. Wir steuern eine Leuchtdiode. Die interaktive Webseite wird im Browser mit einem URI aufgerufen, der ungefähr so aussieht:

`http://192.168.178.40/cgi-bin/led.cgi`

An dem Dateinamen sieht man schon, dass es eine Webseite ist, die von einem CGI-Skript erzeugt wird. Mit Radiobuttons wählst du den gewünschten Zustand und klickst auf die Schaltfläche `SCHALTEN` (Abbildung Y.9). Nach kurzer Zeit zeigt die Leuchtdiode das gewünschte Verhalten.



Abb. Y.9: Interaktive Webseite zum Steuern einer Leuchtdiode mit Radiobuttons

HARDWARE

Die Anode der LED (langer Draht) ist über einen 100-Ohm-Widerstand mit Pin 1 (3,3 Volt) des GPIO verbunden. Die Kathode (kurzer Draht) führt zu Pin 10. Wir steuern Pin 10 mit dem CGI-Skript. Wenn er niedrigen Spannungspegel hat (False), fließt Strom, und die LED leuchtet.

Speichere das folgende CGI-Skript unter dem Pfad

```
/usr/lib/cgi-bin/led.cgi
```

Mache die Programmdatei für alle Benutzer ausführbar:

```
$ sudo chmod +x /usr/lib/cgi-bin/led.cgi
```

Damit das CGI-Skript auf den GPIO zugreifen kann, musst du den User `www-data` (unter dem der Apache-Server läuft) der Gruppe `gpio` hinzufügen:

```
sudo adduser www-data gpio
```

Vergiss nicht, den RPi neu zu starten, damit diese Änderung wirksam wird.

PROGRAMM

```
#!/usr/bin/python3
# led.cgi
import cgi
from RPi import GPIO
from time import sleep

HTML='''Content-type: text/html; charset=utf-8

<html >
<body>
<h1> Steuerung der Leuchtdiode</h1>
<form action="/cgi-bin/led.cgi"
      method="GET">
  <input type="radio" name="modus" value="blinken" {}>
  Blinken <br/>
  <input type="radio" name="modus" value="dauer" {}>
  Dauerleuchten <br/>
  <input type="radio" name="modus" value="aus" {}>
  Aus <br/>
  <input type="submit" value="Schalten"/>
</form>
</body>
</html>''' #1
```

```

def blinken():
    for i in range(10):
        GPIO.output(10, False)
        sleep(0.2)
        GPIO.output(10, True)
        sleep(0.2)

GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.OUT)
form = cgi.FieldStorage()
modus = form.getvalue('modus', 'aus') #2
if modus == 'blinken':
    blinken()
    print(HTML.format('checked', '', '')) #3
else:
    if modus == 'dauer':
        GPIO.output(10, False)
        print(HTML.format('', 'checked', ''))
    else:
        GPIO.output(10, True)
        print(HTML.format('', '', 'checked'))

```

SO FUNKTIONIERT'S

- #1 Das ist das Muster für die Webseite, die das Skript an den Apache-Server zurückgibt. Es enthält drei Platzhalter {}, in die später (#4) jeweils eine Zeichenkette eingetragen wird.

Das Formular `<form ...> ... </form>` enthält drei `input`-Tags vom Typ `radio`. Alle drei Tags haben den gleichen Namen (Attribut `name="modus"`), besitzen aber jeweils andere Werte (Attribut `value`). Mit Radiobuttons wird eine Einfachauswahl bewerkstelligt. Das heißt, es kann immer nur einer der Radiobuttons ausgewählt sein. Er hat dann den Zustand `checked`. Nach dem Anklicken eines `SUBMIT`-Buttons wird in einem Querystring der Wert des ausgewählten Radiobuttons übertragen (siehe Abbildung Y.9).

- #2 Aus dem `FieldStorage`-Objekt wird der Wert der Variablen `modus` ausgelesen. Der Name der Variablen ist das erste Argument des Methodenaufrufs. Das zweite Argument `'aus'` ist ein Wert, der verwendet wird, falls es keinen Querystring gibt. Bedenke: Beim *ersten Aufruf* der Webseite wird *kein Querystring* an den URI angehängt. Das passiert frühestens, wenn man die Seite schon mal aufgerufen hat und auf den `SUBMIT`-Button klickt.
- #3 Wenn im Formular die Option `BLINKEN` gewählt worden ist und demzufolge nun die Variable `auswahl` den Wert `'blinken'` besitzt, passieren zwei Dinge: Erstens

wird die Funktion `blinken()` aufgerufen. Sie sorgt dafür, dass die LED zehn Mal blinkt. Zweitens wird hier ein Tupel aus dem String `'checked'` und zwei leeren Strings gebildet. Dieses Tupel wird in Zeile #4 mit dem Muster für die Webseite verknüpft. Das bewirkt, dass im Tag für den ersten Radiobutton `'checked'` steht (und in den anderen beiden nichts dergleichen). Das wiederum hat zur Folge, dass in der Webseite der erste Radiobutton ausgewählt ist. Kurz und knapp: Der Zustand der Radiobuttons geht nicht verloren, sondern wird beibehalten.

FRAGEN

1. Nenne drei Typen von `<input>`-Tags, die man für die Dateneingabe in einem Formular verwenden kann.
2. Was bewirkt ein Klick auf den SUBMIT-Button eines Formulars?
3. Was ist ein Querystring?
4. In welchem Verzeichnis des Linux-Verzeichnisbaums werden standardmäßig die statischen HTML-Seiten für den Apache-Server gespeichert?
5. In welchem Verzeichnis des Linux-Verzeichnisbaums werden CGI-Skripte für den Apache-Server gespeichert?
6. Welchen Port verwendet der Apache-Server üblicherweise?

AUFGABE: TEMPERATURMESSUNG ÜBER DAS NETZ

Es gibt viele Situationen, in denen man wissen möchte, welche Temperatur gerade an einem bestimmten Ort herrscht. Zum Beispiel prüfen Winzer ständig die Temperatur im Weinfass (manchmal Tag und Nacht), weil die Temperatur verrät, wie schnell der Gärprozess läuft. Je wärmer es ist, desto aktiver sind die Weinhefen, die den Alkohol produzieren. Im Chemielabor steht eine Gärflasche mit Zuckerwasser und Hefe. Darin ist ein digitaler Temperatursensor vom Typ DS1820. Er ist vorchriftsmäßig an den 1-Wire-Bus des Raspberry Pi angeschlossen (siehe Kapitel 14). Du möchtest auf deinem Handy die Temperatur lesen können.

Entwickle ein CGI-Skript, das eine dynamische Webseite wie in Abbildung Y.10 liefert. Klickt man unten auf den Link, wird erneut gemessen und der angezeigte Temperaturwert aktualisiert.



Abb. Y.10: Interaktive Webseite auf einem Smartphone

ANTWORTEN ZU DEN FRAGEN

1. Checkbox `<input type="checkbox" ...>`, Radiobutton `<input type="radio" ...>`, einzeiliges Eingabefeld `<input type="text" ...>`.
2. Beim Klick auf den SUBMIT-Button wird eine neue Web-Resource aufgerufen, üblicherweise ist das ein CGI-Skript. Der URI steht im Anfangstag des Formulars: `<form action="..." ...>`. Außerdem werden die Daten gesendet, die der Benutzer in das Formular eingetragen hat.
3. Wenn die Übertragungsmethode `get` gewählt worden ist, werden die Daten des Formulars als Querystring gesendet. Er beginnt mit einem Fragezeichen. Die Variableninhalte werden in der Form `name=wert` dargestellt. Zwischen mehreren Variablenwerten steht das Zeichen `&`. Beispiel:
`?lampe=an&ventilator=an.`
4. Statische HTML-Seiten: `/var/www/html/`.
5. CGI-Skripte: `/usr/lib/cgi-bin/`.
6. HTTP-Server verwenden Port 80.

LÖSUNG DER AUFGABE

VORBEREITUNG

Schließe den Temperatursensor an und finde zunächst die ID-Nummer heraus. Das geht so:


```
</font>
</body>
</html>''' #2

def readTemp(): #3
    ''' Miss Temperatur in Grad Celsius '''
    f = open(PFAD, 'r')
    daten = f.read()
    f.close()
    return int(daten)/1000

print(HTML.format(readTemp())) #4
```

SO FUNKTIONIERT'S

- #1 Hier steht der exakte Pfad zur Datei mit den Temperaturdaten. Die `xxx` ersetzt du durch die ID-Nummer des Temperatursensors.
- #2 Das ist das Muster der HTML-Seite, die zurückgeschickt wird. Sie enthält einen Platzhalter für den Temperaturwert. Eine Besonderheit ist das Tag ` ...`. Durch die Attribute werden einige Merkmale der Schrift festgelegt. Wichtig ist die Schriftgröße. Mit `size=8` wird die Schrift acht Mal größer als normal. So ist die Webseite auch auf dem kleinen Display eines Handys gut zu lesen.
- #3 Die Funktion liest die Datei mit den Temperaturdaten, liest die Temperatur und gibt sie als Kommazahl zurück.
- #4 In dieser Anweisung passiert eine ganze Menge: Zuerst wird die Funktion `readTemp()` aufgerufen. Sie liefert die Temperatur. Diese Zahl wird in das Muster der HTML-Seite eingetragen, und zwar anstelle des Platzhalters `{}`. Dabei wird übrigens die Kommazahl mit der Temperatur automatisch in einen String umgewandelt. Schließlich wird der komplette Text (die Webseite) mit `print()` ausgegeben.