FRANZIS

Die große Projekt-Box für Raspberry Pi

Über 70 Projekte für Ihren Raspberry Pi

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte Daten sind im Internet über http://dnb.ddb.de abrufbar.

Liebe Kunden!

Dieses Produkt wurde in Übereinstimmung mit den geltenden europäischen Richtlinien hergestellt und trägt daher das CE-Zeichen. Der bestimmungsgemäße Gebrauch ist in der beiliegenden Anleitung beschrieben.

Bei jeder anderen Nutzung oder Veränderung des Produktes sind allein Sie für die Einhaltung der geltenden Regeln verantwortlich. Bauen Sie die Schaltungen deshalb genau so auf, wie es in der Anleitung beschrieben wird. Das Produkt darf nur zusammen mit dieser Anleitung weitergegeben werden.

Das Symbol der durchkreuzten Mülltonne bedeutet, dass dieses Produkt getrennt vom Hausmüll als Elektroschrott dem Recycling zugeführt werden muss. Wo Sie die nächstgelegene kostenlose Annahmestelle finden, sagt Ihnen Ihre kommunale Verwaltung.

Produktmanagement: Michael Büge
Autor/Author: Burkhard Kainka, Christian Immler, Dr. G. Spanner
Art & Design: www.ideehoch2.de
Satz: DTP-Satz A. Kugge, München

INHALTSÜBERSICHT - Leseprobe

1. Vorbereitung ..... PDF-S. 4

2. Elektronik ..... PDF-S. 17

3. Sensoren und Messungen ..... PDF-S. 29

4. LED-Projekte ..... PDF-S. 37

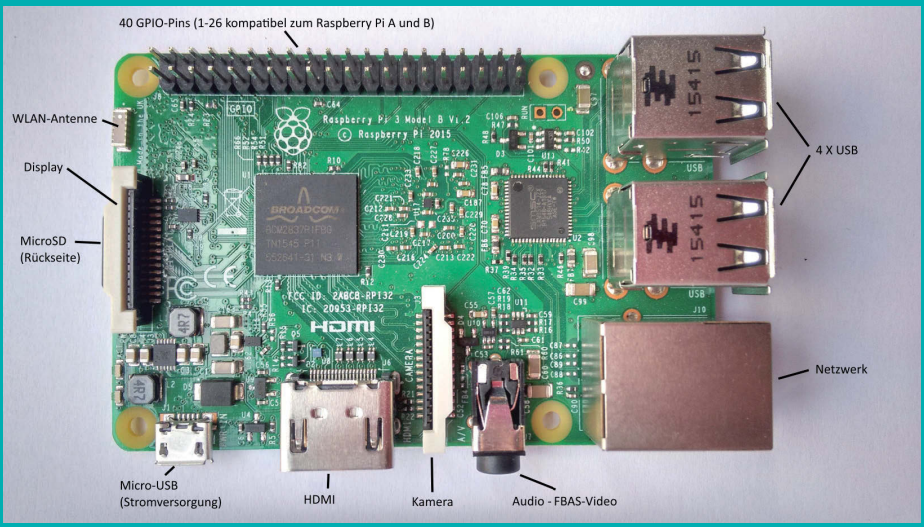
5. Display-Projekte ..... PDF-S. 53

6. Projekte mit Scratch ..... PDF-S. 70

1. Vorbereitung

BEVOR ES LOS-GEHT...

Kaum ein elektronisches Gerät in seiner Preisklasse hat in den letzten Jahren so viel von sich reden gemacht wie der Raspberry Pi. Der Raspberry Pi ist, auch wenn es auf den ersten Blick gar nicht so aussieht, ein vollwertiger Computer etwa in der Größe einer Kreditkarte – und vor allem zu einem sehr günstigen Preis. Nicht nur die Hardware ist günstig, die Software noch mehr. Das Betriebssystem und alle im Alltag notwendigen Anwendungen werden kostenlos zum Download angeboten.



Die Anschlüsse des Raspberry Pi 3

Seit Sommer 2014 haben die neuen Raspberry Pi-Modelle die abgebildete Bauform. Dies gilt für die Modelle Raspberry Pi B+, Pi2, Pi3 und den neuen Pi 3 B+. Alle diese Modelle, sowie auch der Raspberry Pi A+ mit nur einem USB-Anschluss und ohne LAN, verwenden die erweiterte GPIO-Schnittstelle mit 40 Pins. Die älteren Modelle Raspberry Pi A und B aus der Anfangszeit sind heute nur noch mit Einschränkungen kompatibel.

Mit dem speziell angepassten Linux mit grafischer Oberfläche ist der Raspberry Pi ein stromsparender, lautloser PC-Ersatz. Seine frei programmierbare GPIO-Schnittstelle macht den Raspberry Pi besonders interessant für Hardwarebastler und die neue Maker-Szene.

Der Name Raspberry Pi

Raspberry ist das englische Wort für Himbeere. Schon früher wurden Computer nach Früchten benannt, wie z. B. Apple, Apricot, Blackberry. *Pi* steht für Python Interpreter, eine wichtige Programmiersprache auf dem Raspberry Pi. Zusammen ergibt sich ein Name, der wie das englische Wort für Himbeerkuchen *raspberry pie,* klingt.

Was braucht man?

Wenn Sie diesen Text lesen, haben Sie sich sicher schon etwas mit dem Raspberry Pi beschäftigt. Deshalb fassen wir die Systemvoraussetzungen für das MakerKit nur kurz zusammen.

Raspberry Pi

Natürlich brauchen Sie einen Raspberry Pi, und zwar einen aktuellen Raspberry Pi 3 oder Pi 3 B+. Ältere Modelle wurden mit den aktuell verwendeten Softwareversionen nicht getestet.

Micro-USB-Handyladegerät

Für den Raspberry Pi reicht jedes moderne Handynetzteil. Das Netzteil muss 5 V und mindestens 2.000 mA liefern, besser 2.500 mA. Ältere Ladegeräte aus den Anfangszeiten der USB-Ladetechnik sind noch zu schwach. Schließt man leistungshungrige USB-Geräte wie externe Festplatten ohne eigene Stromversorgung an, ist ein stärkeres Netzteil erforderlich.

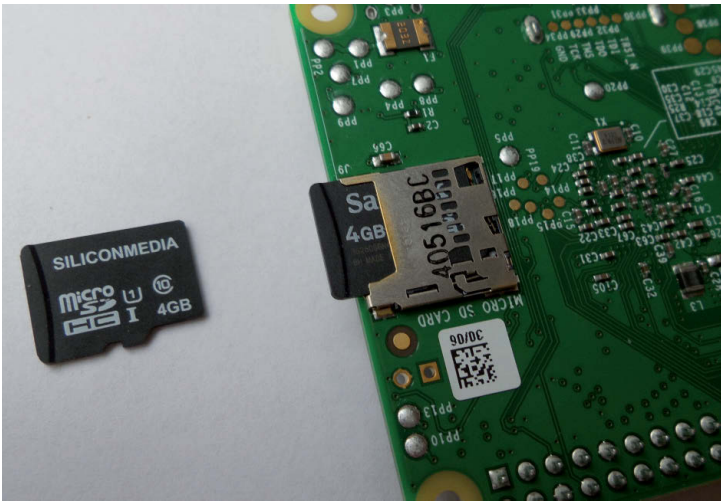
So äußert sich ein zu schwaches Netzteil

Wenn der Raspberry Pi bootet, sich dann aber die Maus nicht bewegen lässt oder das System nicht auf Tastatureingaben reagiert, deutet dies auf eine zu schwache Stromversorgung hin. Auch wenn der Zugriff auf angeschlossene USB-Sticks oder Festplatten nicht möglich ist, sollten Sie ein stärkeres Netzteil verwenden.

Ein Blitzsymbol oben rechts auf dem Bildschirm zeigt eine zu geringe Spannungsversorgung an. Fällt die Versorgungsspannung unter 4,63V, deutet dies in den meisten Fällen auf ein zu schwaches Netzteil hin, das zwar eine Nennspannung von 5V, aber nur weniger als 2000mA liefert.

Speicherkarte

Die Speicherkarte dient im Raspberry Pi als Festplatte. Sie enthält das Betriebssystem. Eigene Daten und installierte Programme werden ebenfalls darauf gespeichert. Die Speicherkarte sollte mindestens 8 GByte groß sein und nach Herstellerangaben des Raspberry Pi mindestens den Class 4-Standard unterstützen. Dieser Standard gibt die Geschwindigkeit der Speicherkarte an. Eine aktuelle Class 10-Speicherkarte macht sich in der Performance deutlich bemerkbar. Alle aktuellen Raspberry Pi-Modelle verwenden die aus Smartphones bekannten MicroSD-Speicherkarten.



Der MicroSD-Speicherkartensteckplatz auf der Rückseite des Raspberry Pi. Die Zahl im Kreis gibt die Klasse der Speicherkarte an.

Tastatur und Maus

Jede gängige Tastatur mit USB-Anschluss kann genutzt werden. Kabellose Tastaturen funktionieren manchmal nicht, da sie zu viel Strom oder spezielle Treiber benötigen.

Einige USB-Tastaturen besitzen einen weiteren USB-Anschluss für die Maus. Dadurch sparen Sie sich am Raspberry Pi einen USB-Anschluss.

Netzwerkkabel

Zur Verbindung mit dem Router im lokalen Netzwerk wird ein Netzwerkkabel benötigt. Der Raspberry Pi 3 und Pi 3 B+ haben eingebaute WLAN-Mo-

dule. Hier ist kein Netzwerkkabel zur Internetverbindung nötig. Ohne Internetzugang sind viele Funktionen des Raspberry Pi nicht sinnvoll nutzbar.

HDMI-Kabel

Der Raspberry Pi kann per HDMI-Kabel an Monitoren oder Fernsehern angeschlossen werden. Zum Anschluss an Computermonitore mit DVI-Anschluss gibt es spezielle HDMI-Adapter. HDMI-Kabel sind im Elektronikhandel zu Preisen erhältlich, die fast dem Preis des Raspberry Pi selbst entsprechen. Bei Onlineversendern (z. B. *amzn.to/VGv05j*) bekommt man sie einschließlich Versand für wenige Euro – Unterschied: für die Verwendung am Raspberry Pi außer dem Preis keiner.

Audiokabel

Über ein Audiokabel mit 3,5-mm-Klinkensteckern können Kopfhörer oder PC-Lautsprecher am Raspberry Pi genutzt werden. Das Audiosignal ist auch über das HDMI-Kabel verfügbar. Bei HDMI-Fernsehern oder Monitoren ist kein Audiokabel nötig. Wird ein PC-Monitor über ein HDMI-Kabel mit DVI-Adapter angeschlossen, geht meist an dieser Stelle das Audiosignal verloren, sodass Sie den analogen Audioausgang wieder brauchen.

Raspbian-Betriebssystem

Wir gehen bei allen Projekten davon aus, dass die aktuelle Version des Raspbian Stretch-Betriebssystems mit Pixel-Desktop auf der Speicherkarte installiert ist. Ältere Versionen sind zu aktuellen Raspberry Pi-Modellen nur eingeschränkt kompatibel und bieten auch keine GPIO-Unterstützung für die Programmiersprache Scratch. Der Raspberry Pi 3 B+ benötigt mindestens die Betriebssystemversion NOOBS 2.7.0.

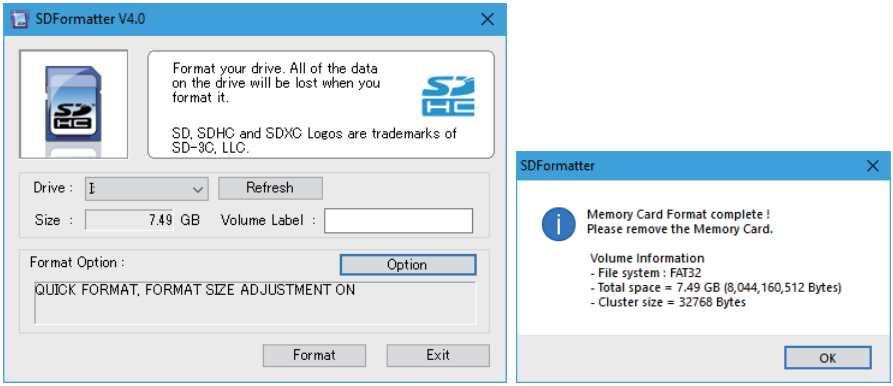
Speicherkarte im PC vorbereiten

Da der Raspberry Pi selbst noch nicht booten kann, bereiten wir die Speicherkarte auf dem PC vor. Dazu brauchen Sie einen Kartenleser am PC. Dieser kann fest eingebaut oder per USB angeschlossen werden.

Verwenden Sie am besten fabrikneue Speicherkarten, da diese vom Hersteller bereits optimal vorformatiert sind. Sie können aber auch eine Speicherkarte verwenden, die vorher bereits in einer Digitalkamera oder

einem Smartphone genutzt wurde. Diese Speicherkarten sollten vor der Verwendung für den Raspberry Pi neu formatiert werden. Theoretisch können Sie dazu die Formatierungsfunktionen von Windows verwenden. Deutlich besser ist die Software „SDFormatter" der SD Association. Damit werden die Speicherkarten für optimale Performance formatiert. Dieses Tool können Sie sich bei *www.sdcard.org/downloads/formatter_4* kostenlos herunterladen.

Sollte die Speicherkarte Partitionen aus einer früheren Betriebssysteminstallation für den Raspberry Pi enthalten, wird im SDFormatter nicht die vollständige Größe angezeigt. Verwenden Sie in diesem Fall die Formatierungsoption ***FULL (Erase)*** und schalten Sie die Option ***Format Size Adjustment*** ein. Damit wird die Partitionierung der Speicherkarte neu angelegt.



Das SDFormatter-Tool unter Windows in Aktion.

Speicherkarte wird gelöscht

Am besten verwenden Sie eine leere Speicherkarte für die Installation des Betriebssystems. Sollten sich auf der Speicherkarte Daten befinden, werden diese durch die Neuformatierung während der Betriebssysteminstallation unwiderruflich gelöscht.

Der Software-Installer NOOBS

„New Out Of Box Software" (NOOBS) ist ein besonders einfacher Installer für Raspberry Pi-Betriebssysteme. Hier braucht sich der Benutzer nicht mehr wie früher selbst mit Image-Tools und Bootblöcken auseinanderzusetzen, um eine bootfähige Speicherkarte einzurichten.

NOOBS bietet verschiedene Betriebssysteme zur Auswahl, wobei man beim ersten Start direkt auf dem Raspberry Pi das gewünschte Betriebssystem auswählen kann, das dann bootfähig auf der Speicherkarte installiert wird.

Laden Sie sich das etwa 1,5 GByte große Installationsarchiv für NOOBS von der offiziellen Downloadseite *www.raspberrypi.org/downloads* herunter und entpacken Sie es am PC auf eine Speicherkarte. Die Experimente wurden mit der NOOBS-Version 2.7.0 getestet.

Starten Sie jetzt den Raspberry Pi mit dieser Speicherkarte. Stecken Sie sie dazu in den Steckplatz des Raspberry Pi und schließen Sie Tastatur, Maus, Monitor und Netzwerkkabel an. Der USB-Stromanschluss kommt als Letztes. Damit wird der Raspberry Pi eingeschaltet. Einen separaten Einschaltknopf gibt es nicht.

Nach wenigen Sekunden erscheint ein Auswahlmenü, in dem Sie das gewünschte Betriebssystem wählen können. Wir verwenden das von der Raspberry Pi-Stiftung empfohlene Betriebssystem Raspbian.

Wählen Sie ganz unten Deutsch als Installationssprache aus und markieren Sie das vorausgewählte Raspbian-Betriebssystem. Nach Bestätigen einer Sicherheitsmeldung, dass die Speicherkarte überschrieben wird, startet die Installation, die einige Minuten dauert. Während der Installation werden kurze Informationen zu Raspbian angezeigt.

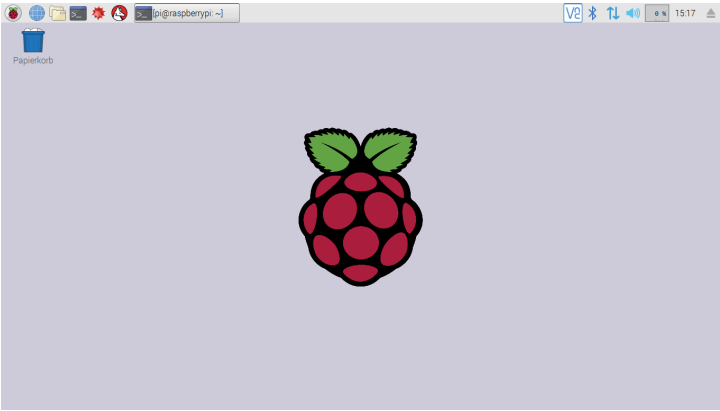
Der erste Start auf dem Raspberry Pi

Nach abgeschlossener Installation bootet der Raspberry Pi neu und startet automatisch den grafischen Desktop PIXEL. Die deutsche Sprache und die deutsche Tastaturbelegung sollten wie auch weitere wichtige Grundeinstellungen bereits automatisch gewählt sein, was leider nicht in allen Versionen korrekt funktioniert.

Fast wie Windows – die grafische Oberfläche LXDE

Viele schrecken bei dem Wort „Linux" erst einmal zurück, weil sie befürchten, kryptische Befehlsfolgen per Kommandozeile eingeben zu müssen, wie vor 30 Jahren unter DOS. Weit gefehlt! Linux bietet als offenes Betriebssystem den Entwicklern freie Möglichkeiten, eigene grafische Oberflächen zu entwickeln. So ist man als Anwender des im Kern immer

noch kommandozeilenorientierten Betriebssystems nicht auf eine Oberfläche festgelegt.



Der PIXEL-Desktop auf dem Raspberry Pi.

Das Raspbian-Linux für den Raspberry Pi verwendet die eigens entwickelte Oberfläche PIXEL auf Basis des bekannten LXDE (Lightweight X11 Desktop Environment), die einerseits sehr wenig Systemressourcen benötigt und andererseits mit ihrem Startmenü und dem Dateimanager der gewohnten Windows-Oberfläche sehr ähnelt.

Linux-Anmeldung

Selbst die bei Linux typische Benutzeranmeldung wird im Hintergrund erledigt. Falls Sie es doch einmal brauchen: Der Benutzername lautet `pi` und das Passwort `raspberry`.

Das Menü-Symbol links oben öffnet das Startmenü, die Symbole daneben den Webbrowser, Dateimanager und ein Kommandozeilenfenster. Das Startmenü ist wie unter Windows mehrstufig aufgebaut. Häufig verwendete Programme lassen sich mit einem Rechtsklick auf dem Desktop ablegen.

Raspberry Pi ausschalten

Theoretisch kann man bei dem Raspberry Pi einfach den Stecker ziehen, und er schaltet sich ab. Besser ist es jedoch, ihn wie einen PC sauber herunterzufahren. Wählen Sie dazu im Menü *Shutdown*.

2. Elektronik

DIE STROM-VERSORGUNG

Elektronische Schaltungen müssen mit Spannung versorgt werden. Hierfür soll überwiegend die Spannungsversorgung eines Raspberry Pi zum Einsatz kommen.

Die folgende Abbildung zeigt, wie dazu die beiden Busschienen des Steckboards mit dem Raspberry Pi verbunden werden müssen. Beim Aufbau von Schaltungen mit dem Raspberry Pi sollte man mit größter Sorgfalt vorgehen.

Schaltungsfehler können schnell zur Zerstörung des Controllerchips auf dem Raspberry Pi führen, und dies würde bedeuten, dass der gesamte Raspberry Pi wertlos wird, da sich Reparaturen kaum lohnen.

Konventionsgemäß verwendet man für positive Spannungen ein rotes, für negative ein schwarzes oder blaues Kabel. Es ist sinnvoll, sich gleich von Beginn an an diese Farbgebung zu halten. Am besten trennt man also zwei Leitungen an mit den entsprechenden Farben vom Jumperkabelband ab und baut damit die Stromversorgung auf.

Wichtig:
Besonders die hier verwendete Spannung von 5 V darf keinesfalls mit anderen Pins auf der Steckerleiste in Verbindung kommen. Die Eingänge des Controllers dürfen nur mit maximal 3,3 V angesteuert werden. Kommen sie mit 5 V in Berührung, werden sie überlastet. Dies kann wiederum zur Zerstörung des gesamten Raspberry Pi führen.

Der Raspberry Pi selbst wird über die Mikro-USB-Buchse (in der Abbildung mit „Power" bezeichnet) mit Strom versorgt. Als Netzteil eignet sich praktisch jedes moderne Handy-Ladegerät mit Mikro-USB-Buchse. Diese Ladegeräte können auch einzeln in vielen Variationen gekauft werden. Man sollte jedoch darauf achten, dass das Gerät in der Lage ist, mindestens einen Strom von 2 Ampere zu liefern. Zwar kommen ältere

Versionen des Raspberry Pi auch mit geringeren Leistungen aus, ab Version 3 werden allerdings tatsächlich die vollen 2 Ampere benötigt.

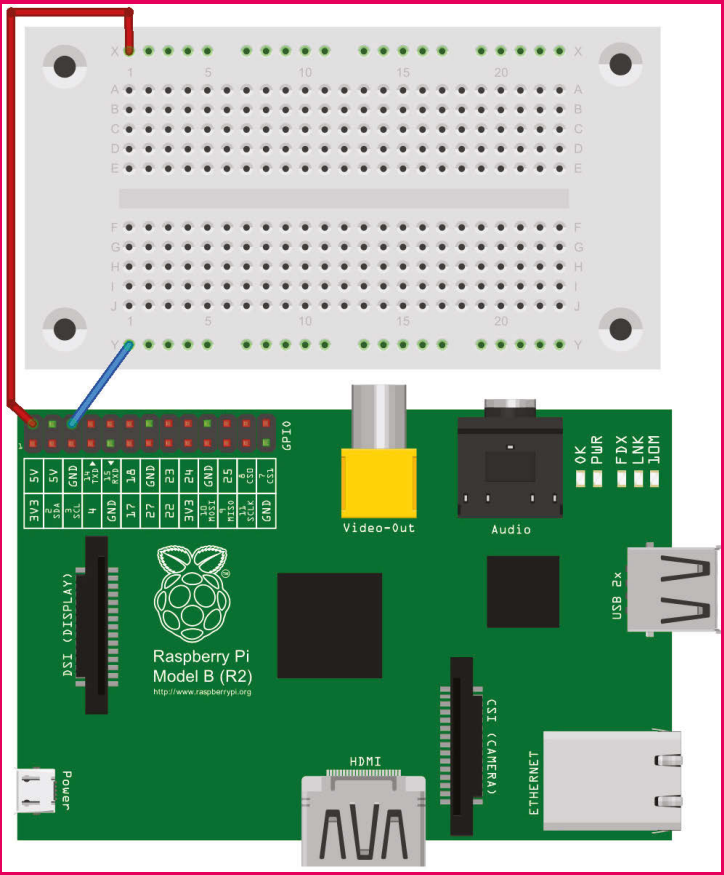


Abbildung 2.1: Versorgung des Steckboards mit 5 V

DIE ELEKTRONISCHEN BAUELEMENTE

Bevor man mit dem Experimentieren beginnt, sollte man sich mit den einzelnen Bauelementen vertraut machen und ihre korrekte Funktion überprüfen. Die für einige Komponenten gezeigten Testschaltungen können auch später verwendet werden, um einzelne Bauelemente zu testen, sollte Unklarheit über ihre Funktionsfähigkeit bestehen.

Im Folgenden werden lediglich die wichtigsten Funktionen und Eigenschaften der einzelnen Standardbauelemente kurz beschrieben. Weitere Details zu Funktionsweise und Anwendung fi nden sich in späteren Kapiteln, in denen näher auf jedes einzelne Sensorelement eingegangen wird.

Drahtbrücken

Zur Verbindung der Bauelemente benötigt man sogenannte Drahtbrücken. Hierfür werden vom beiliegenden Schaltdraht Stücke entsprechender Längen abgeschnitten. Danach sind die Enden von der Isolation zu befreien. Dies geht am einfachsten mit einem Seitenschneider. Notfalls kann man aber auch ein scharfes Messer benutzen. Dabei rollt man das Drahtstück unter der Messerklinge, bis die Isolation, nicht aber der Draht, durchtrennt ist. Dann kann man die Isolation einfach abziehen.

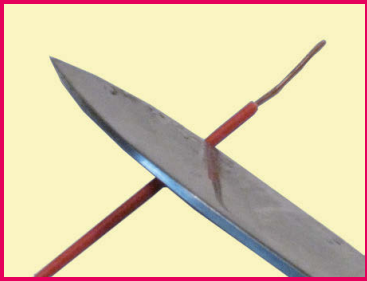


Abbildung 3.1: Abisolieren von Drähten mit einem Messer

Drahtbrücken werden im Schaltbild als einfache Linien dargestellt. Wichtig zu beachten ist, dass sich kreuzende Linien nur elektrisch verbunden sind, wenn ein zusätzlicher Punkt sie markiert.

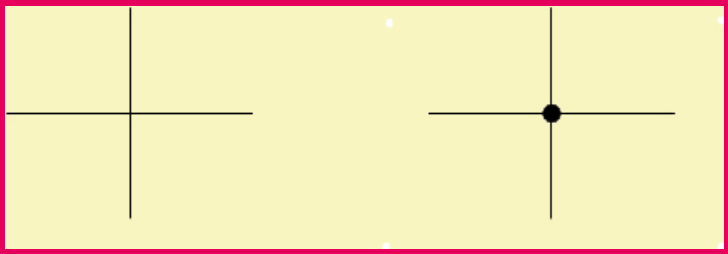


Abbildung 3.2: Leitungskreuzungen

Widerstände

Widerstände sind die einfachsten Bauelemente der Elektronik. Sie sind ungepolt und recht robust. Die Widerstandswerte werden durch Farbringe codiert. Für Widerstände mit 5 % Toleranz werden vier Farbringe verwendet. Drei geben den Widerstandswert an, der vierte für die Toleranz von 5 % ist goldfarben. Für Exemplare mit 1 % Toleranz sind fünf Farbringe erforderlich, hier gibt der braune fünfte Ring die Toleranz von 1 % an. Der fünfte Ring für die Toleranz wird in der Tabelle nicht mit aufgeführt. Den Ring für die Toleranzangabe erkennt man daran, dass er etwas brei-ter ist als die anderen. Folgende zwölf Widerstandswerte (entweder jeweils mit 5 % oder mit 1 % Toleranz) werden eingesetzt:

Anzahl	Wert	Farbringkombination für 5 % Toleranz	Farbringkombination für 1 % Toleranz
8	150 Ohm	Braun-Grün-Braun	Braun-Grün-Schwarz-Schwarz
4	1 kOhm	Braun-Schwarz-Rot	Braun-Schwarz-Schwarz-Braun
2	10 kOhm	Braun-Schwarz-Orange	Braun-Schwarz-Schwarz-Rot
2	100 kOhm	Braun-Schwarz-Gelb	Braun-Schwarz-Schwarz-Orange
2	1 MOhm	Braun-Schwarz-Grün	Braun-Schwarz-Schwarz-Gelb

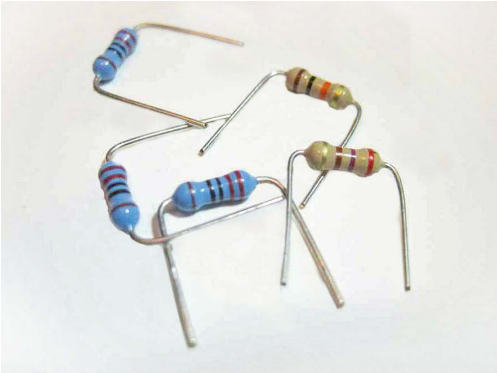


Abbildung 3.3: Widerstände

Es empfiehlt sich, die Anschlussdrähte der Widerstände mit einem Seitenschneider etwas zu kürzen. Dann sollte man sie so biegen, wie in obiger Abbildung gezeigt, d. h., die Drähte werden so umgebogen, dass beim Einstecken drei Löcher unter dem Widerstand frei bleiben. Bei Bedarf können die Drahtenden etwas zusammen- oder auseinandergebogen werden, sodass im Steckboard nur zwei oder aber vier Löcher unter dem Widerstand frei bleiben. Zu stark oder zu oft sollten die Drähte aber nicht gebogen werden, da sie sonst brechen könnten.

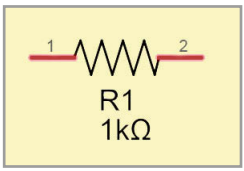


Abbildung 3.4: Schaltbild eines Widerstands

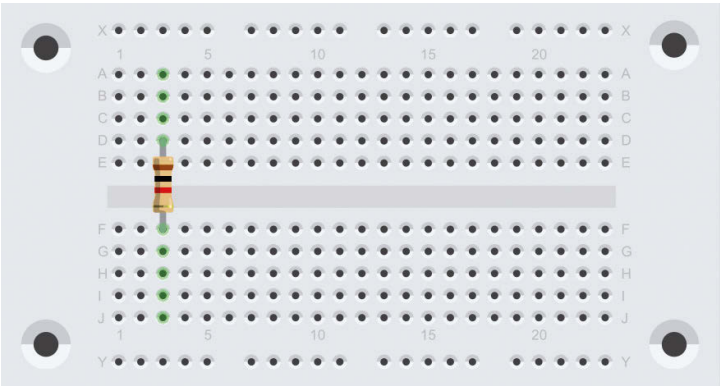


Abbildung 3.5: Widerstand in einem Steckboard

LEDs

Der allererste Hinweis ist wichtig, da seine Nichtbeachtung leider immer wieder zur unbeabsichtigten Zerstörung von LEDs führt:

Wichtig:
Licht emittierende Dioden (LEDs oder Leuchtdioden) dürfen niemals direkt an die Stromversorgung angeschlossen werden.

Bei dem direkten Anschluss an 5 V ist immer mindestens ein Widerstand von 150 Ohm vorzuschalten, damit der Diodenstrom auf einen zulässigen Wert begrenzt wird!

Die Kathode der LED ist durch eine Abflachung am Kunststoffgehäuse gekennzeichnet. Außerdem ist meist der Anschlussdraht für die Kathode etwas kürzer. Beide Merkmale sind allerdings nicht immer sehr zuverlässig oder eindeutig. Falls eine Schaltung nicht korrekt funktioniert, ist es immer eine gute Idee, die Polung der LEDs zu prüfen.

Bei den kleinen LEDs im 3-mm-Gehäuse sollte man die Anschlussdrähte nicht zu stark verbiegen, da sie mechanisch nicht so belastbar sind wie die großen 5-mm-LEDs.

Die beiden Bauteile mit den glasklaren Gehäusen werden im nächsten Kapitel als weiße LED bzw. als Fototransistor identifiziert.

Abbildung 3.6: Leuchtdioden

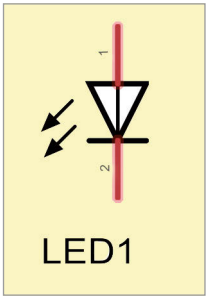


Abbildung 3.7: Schaltbild einer LED

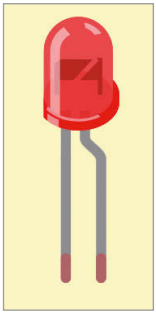


Abbildung 3.8: Aufbaubild einer LED

Der erste Stromkreis

Mit einem Widerstand und einer LED kann man die erste Schaltung aufbauen. Der positive Pol der Spannungsversorgung wird über die obere Busschiene mit einem 1-kOhm-Widerstand (Kiloohm) (Farbcode: Braun-Schwarz-Rot) verbunden. Danach folgt die LED (auf die Polung achten!), die dann über die untere Busschiene mit dem Minuspol der Spannung verbunden wird. Ist alles richtig gesteckt, kann der Raspberry Pi über ein Mikro-USB-Kabel mit Spannung versorgt werden. Die LED sollte nun leuchten.

Jetzt können nacheinander alle LEDs überprüft werden. Auch der Fototransistor kann bereits als solcher erkannt werden – im Gegensatz zur weißen LED strahlt er kein Licht aus, egal, wie herum er gepolt ist.

Die folgende Abbildung zeigt anhand eines Aufbaubilds, wie zwei LEDs auch parallel betrieben werden können.

Danach folgt das zugehörige Schaltbild. In einem Schaltbild wird der Aufbau lediglich in einer etwas abstrakteren Form wiedergegeben.

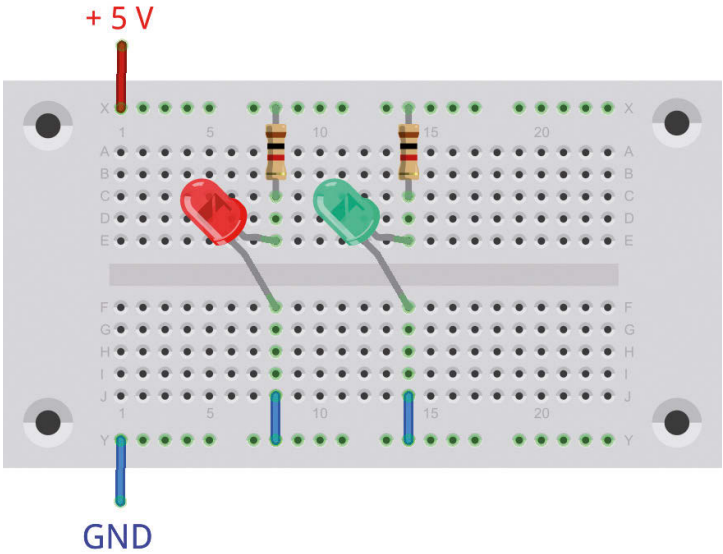


Abbildung 3.9: Paralleler Betrieb von zwei LEDs

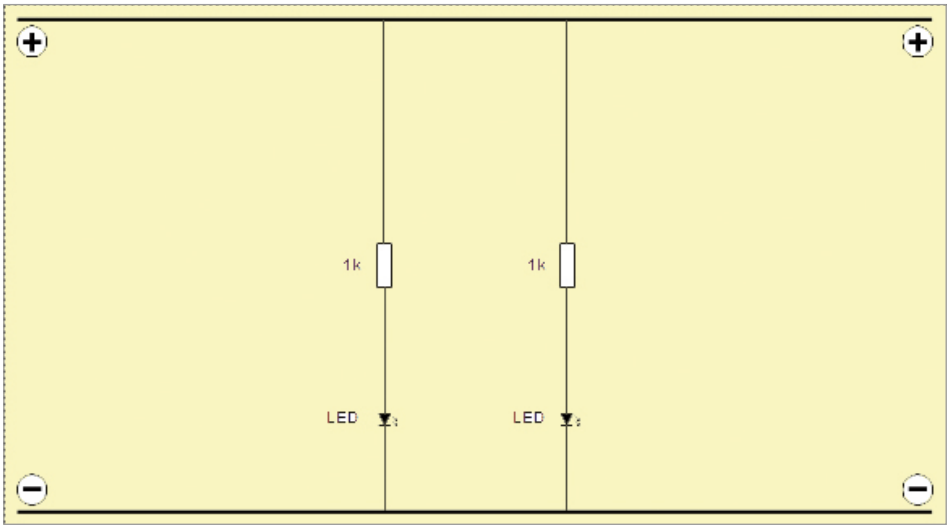


Abbildung 3.10: Schaltbild zum vorhergehenden Aufbau

Kondensatoren

Kondensatoren sind ebenfalls recht robuste Bauelemente. Die einfachen Kondensatoren sind ungepolt. Vorsicht ist aber bei den Elektrolytkonden-

satoren („Elkos") geboten. Hier ist unbedingt auf die richtige Polung zu achten, da sie bei einer Verpolung zerstört werden können. Die Elkos tragen daher immer eine entsprechende Kennzeichnung. Es sind die folgenden Werte enthalten:

Anzahl	Wert	Beschriftung
1	10 nF	103 oder 10 n
1	100 nF	104 oder 100 n
1	10 mF, 16 V	10 µ oder 10 µF
1	100 mF, 16 V	100 µ oder 100 µF

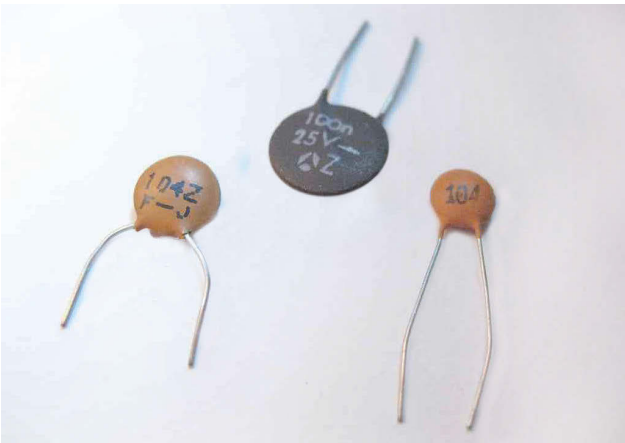


Abbildung 3.11: Kondensatoren

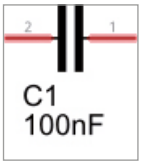


Abbildung 3.12: Schaltbild des Kondensators

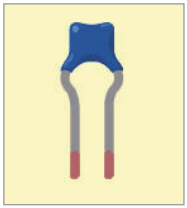


Abbildung 3.13: Kondensator im Aufbaubild

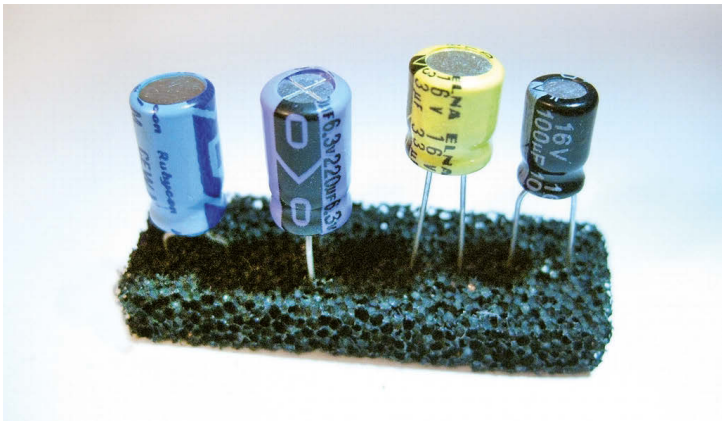


Abbildung 3.14: Elkos

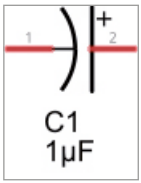


Abbildung 3.15: Schaltbild eines Elkos

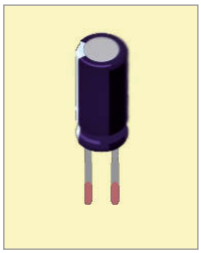


Abbildung .316: Elko im Aufbaubild

Das Potenziometer

Das Potenziometer (oder kurz „Poti") ist ein einstellbarer Widerstand. Es besteht aus einer Widerstandsbahn mit zwei Anschlüssen, auf denen mittels eines Schleifkontakts verschiedene Widerstände stufenlos eingestellt werden können. Dieser Schleifer darf nicht direkt mit einer Spannung verbunden werden, da bei Einstellung sehr kleiner Widerstandswerte hohe Ströme fließen könnten, die die dünne Widerstandsschicht des Potis zerstören würden.

Abbildung 3.17: Potenziometer

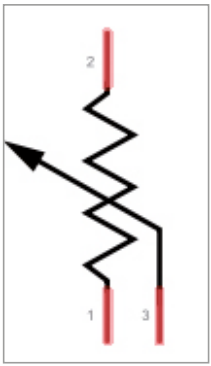


Abbildung 3.18: Schaltbild zum Potenziometer

Abbildung 3.19: Potenziometer im Aufbaubild

3. Sensoren und Messungen

GRUNDLAGEN SENSORTECHNIK

Als Sensoren (vom lateinischen Wort sensus = Gefühl) oder Messfühler werden in der Technik allgemein Bauteile bezeichnet, die bestimmte physikalische oder chemische Eigenschaften erfassen können. Nahezu alle physikalischen Größen sind mit elektronischen Sensoren messbar. Die wichtigsten Beispiele für diese Größen sind:

- Beschleunigungen und Kräfte
- Wärmestrahlung
- Temperatur
- Feuchtigkeit
- Druck
- Schallintensität
- Lichtintensität
- elektrische und magnetische Felder

Sensoren haben in der Technik eine weite Verbreitung gefunden. Von der Kraftfahrzeugtechnik über die Raumfahrt bis hin zur Medizin – elektronische Sensoren sind aus dem Alltag nicht mehr wegzudenken. Allein in einem modernen Auto befinden sich über 100 verschiedene Messfühler, anfangen bei der klassischen Motortemperaturkontrolle und Ölstandüberwachung bis hin zu den modernsten Sensorsystemen, die über Kollisionsdetektoren den Airbag auslösen, im ABS-System das Blockieren der Räder verhindern oder für elektronische Stabilitätsprogramme (ESP) die Straßenlage des Fahrzeugs erfassen. Die folgende Abbildung zeigt einige Beispiele für Sensoren, die in Kraftfahrzeugen Anwendung finden.

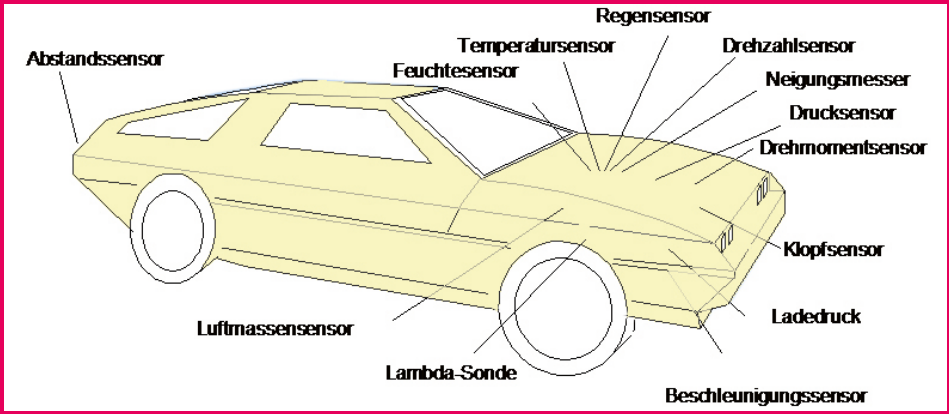


Abbildung 5.1: Kfz-Sensoren

Aber auch in anderen Bereichen sind Sensoren unentbehrlich geworden. Im Bedarfsfall können etwa in der Medizin lebenswichtige Parameter überwacht werden, als Beispiele seien hier nur das EKG und die elektronischen Blutzuckermessgeräte genannt. Ob in der chemischen Industrie, in der Umwelttechnik oder im Haushalt, auf elektronische Sensoren kann in keinem Lebensbereich mehr verzichtet werden.

Ein Lernpaket gibt eine umfassende Einführung in die wichtigsten Sensortypen und ihre Anwendungen. Die Erfassung von Temperaturen, Helligkeitswerten, Abständen oder Bewegungen wird jeweils nach Themenbereich geordnet vorgestellt und in elektronischen Beispielschaltungen praktisch erprobt.

Neben einfachen Grundschaltungen wie Thermometern und Helligkeitsmessgeräten werden auch komplexere Anwendungen wie die optische Objekterfassung sowie eine elektronische Waage mit Magnetfelddetektoren vorgestellt.

Querempfindlichkeiten

Üblicherweise dienen Sensoren der Erfassung einer einzelnen spezifischen Größe. Diese soll möglichst reproduzierbar in ein elektrisches Signal umgewandelt werden. Oftmals ist dabei ein linearer Zusammenhang zwischen Messgröße und elektrischer Größe erwünscht. Aber auch nicht lineare Übertragungsfunktionen können durch den Einsatz von Computererfassungssystemen oder Mikrocontrollern problemlos ausgewertet werden.

Ein wesentlich schwerwiegenderes Problem stellen die sogenannten Querempfindlichkeiten dar. Darunter versteht man die Tatsache, dass Sensoren oftmals nicht nur auf die eigentliche Messgröße reagieren, sondern auch auf andere physikalische Einflüsse. Die bekanntesten Beispiele sind:

- Temperaturempfindlichkeit von Lichtsensoren
- Reaktion von Schallwandlern auf mechanische Erschütterungen
- Einfluss der Luftfeuchtigkeit auf Sensoren für elektrische Felder

Auch durch eine technische Optimierung können diese Einflüsse oft nicht vollständig eliminiert werden. Hier helfen meist nur die zusätzliche Erfassung der Störgröße und eine rechnerische Korrektur des Messwerts.

Temperaturabhängige Kondensatoren

Ein Musterbeispiel für eine Querempfi ndlichkeit ist die Temperaturabhängigkeit von Keramikkondensatoren. Dieser unerwünschte Effekt ist sogar so groß, dass man ihn als praktische Anwendung nutzen könnte. So ist es durchaus möglich, mit dem Raspberry Pi die Temperaturabhängigkeit eines 100-nF-Kondensator zu vermessen.

EINFACHE SCHALTSENSOREN

Eine wichtige Gruppe von Sensoren stellen die sogenannten mechanischen Schaltsensoren dar. Bei ihnen wird durch direkte mechanisch-elektrische Kontakte ein Schaltvorgang ausgelöst. Ein weitverbreiteter Schaltsensortyp ist der sogenannte Tilt-Sensor.

Trotz ihres einfachen Aufbaus können mechanische Schaltsensoren vielfältig eingesetzt werden. Ihr Hauptvorteil liegt im geringen technischen Aufwand und dem damit verbundenen günstigen Preis.

Schaltsensoren kommen zum Einsatz, wenn beispielsweise überwacht werden soll, ob ein Gehäuse vollständig geschlossen ist. Beim Öffnen einer Klappe oder einer Tür ertönt ein Alarm, oder die Spannungszuführung zum Gerät wird unterbrochen.

Ein Hauptnachteil von mechanischen Sensoren ist ihre Störanfälligkeit. Der metallische Kontakt kann durch Korrosion stark beeinflusst werden. Bereits bei den folgenden Experimenten kann man feststellen, dass einfache Metall-auf-Metall-Kontakte nicht immer zuverlässig arbeiten. In feuchten Umgebungen ist mit häufi gen Ausfällen zu rechen. Dort ist es günstiger, Reed-Relais oder Hall-Sensoren einzusetzen.

Der Tilt-Sensor und seine Anwendungen

Das Funktionsprinzip eines Tilt- oder Neigungssensors (engl. „to tilt“ = kippen, neigen) ist einfach. Die nachfolgende Abbildung zeigt den inneren Aufbau des Bauelements. Das staubdicht abgeschlossene Gehäuse des Sensors enthält zwei oder mehr Kontaktstifte und eine leitfähige Metallkugel.

Steht der Sensor aufrecht, stellt die Metallkugel eine leitende Verbindung zwischen den Kontaktstiften her. Wird der Sensor gekippt, rollt die Kugel im Gehäuse nach unten, und der elektrische Kontakt wird unterbrochen.

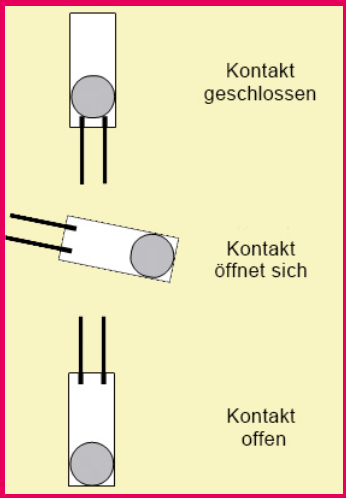


Abbildung 9.1: Funktionsprinzip des Tilt-Sensors

Einfache, aber wirkungsvolle Alarmanlage

Eine wichtige Aufgabe von Tilt-Sensoren ist die Auslösung von Alarmen. Sicherlich kennen Sie das Prinzip von Flipperautomaten. Wenn sie etwa durch Stoßen oder Anheben manipuliert werden, löst ein entsprechender Sensor einen Alarm aus. Die Warnung „TILT" wird auf der Anzeigefläche ausgegeben, und das Spiel ist beendet.

Auch zur Sicherung von Motorrädern kann ein Tilt-Sensor eingesetzt werden. Wird die Maschine in einem bestimmten Neigungswinkel abgestellt, befindet sich der Sensor in seiner Ruhelage. Bei einer Bewegung des Motorrads wird ein Alarm ausgelöst.

Die nachfolgende Abbildung zeigt den Aufbau einer Alarmanlage mit einem Tilt-Sensor und dem Raspberry Pi.

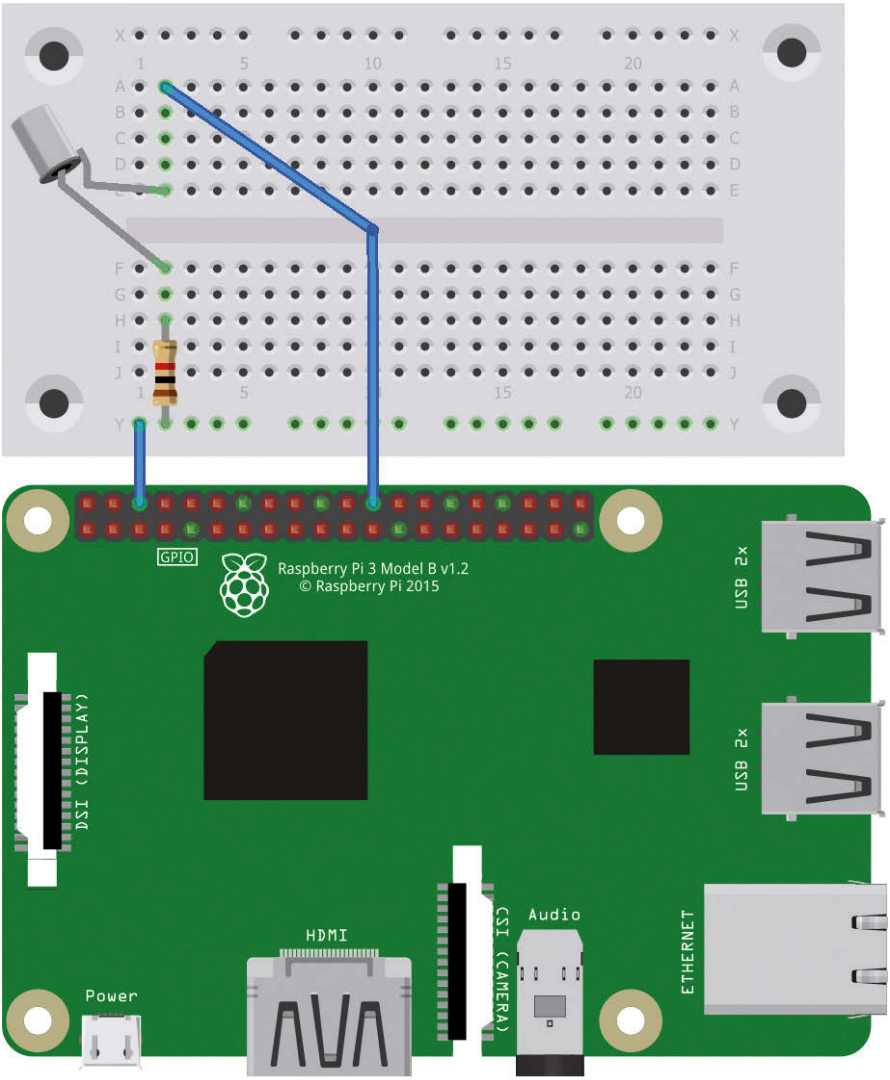


Abbildung 9.2: Tilt-Sensor am Raspberry Pi

Programmdatei
Tilt_alarm.py

Das Programm dazu sieht so aus:

```
# Tilt_alarm.py
# get GPIO-Pin status using INTERRUPT

from Tkinter import *
```

```
005 import RPi.GPIO as GPIO
006
007 GPIO.setmode(GPIO.BCM)
008 GPIO.setup(8,GPIO.IN,pull_up_down=GPIO.PUD_UP)
009
010 root=Tk()
011
012 AlarmState=StringVar()
013 AlarmState.set("Undefined")
014
015 def state_isr(pin): # subroutine for interrupt
016     if GPIO.input(pin)==0:
017         AlarmState.set("Ready")
018     else:
019         AlarmState.set("ALARM!")
020     return
021
022 root.title("Alarm Center")
023 root.geometry("280x60") # window size
024 root.font=('Arial', 30, 'normal')
025
026
027 # check statusGPIO08
028 Label(root, textvariable=AlarmState, font = root.
    font).pack()
029 GPIO.add_event_detect(8, GPIO.BOTH, callback=state_
    isr)
030 root.mainloop()
```

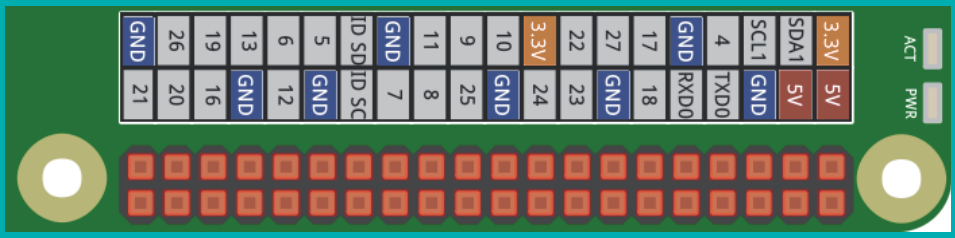
Natürlich kann man bei Bedarf die Alarmmeldung auch in auffälliger Farbe oder in einer besonderen Schriftart darstellen.

4. LED-Projekte

DIE ERSTE LED LEUCHTET AM RASPBERRY PI

Die 40-polige Stiftleiste in der Ecke des Raspberry Pi bietet die Möglichkeit, direkt Hardware anzuschließen, um z. B. über Taster Eingaben zu machen oder programmgesteuert LEDs leuchten zu lassen. Diese Stiftleiste wird als GPIO bezeichnet. Die englische Abkürzung „General Purpose Input Output" bedeutet auf Deutsch einfach „Allgemeine Ein- und Ausgabe".

Von diesen 40 Pins lassen sich die 22, die nur mit Nummern bezeichnet sind, wahlweise als Eingang oder Ausgang programmieren und so für vielfältige Hardwareerweiterungen nutzen. Die übrigen sind für die Stromversorgung und andere Zwecke fest eingerichtet.



Belegung der GPIO-Schnittstelle. Die abgerundete Ecke unten rechts entspricht der Ecke der Raspberry Pi Platine.

Vorsicht

Verbinden Sie auf keinen Fall irgendwelche GPIO-Pins miteinander und warten ab, was passiert, sondern beachten Sie unbedingt folgende Hinweise:

Einige GPIO-Pins sind direkt mit Anschlüssen des Prozessors verbunden, ein Kurzschluss kann den Raspberry Pi komplett zerstören. Verbindet man über eine LED zwei Pins miteinander, muss immer ein Vorwiderstand dazwischengeschaltet werden.

Verwenden Sie für Logiksignale immer Pin 1, der +3,3 V liefert und bis 50 mA belastet werden kann. Alle mit *GND* bezeichneten Pins, sind Masseleitungen für Logiksignale.

Jeder GPIO-Pin kann als Ausgang (z. B. für LEDs) oder als Eingang (z. B. für Taster) geschaltet werden. GPIO-Ausgänge liefern im Logikzustand *1* eine Spannung von +3,3 V, im Logikzustand *0* 0 Volt. GPIO-Eingänge liefern bei einer Spannung bis +1,7 V das Logiksignal *0*, bei einer Spannung zwischen +1,7 V und +3,3 V das Logiksignal *1*.

Pin 2 und Pin 4 liefern +5 V zur Stromversorgung externer Hardware. Hier kann so viel Strom entnommen werden, wie das USB-Netzteil des Raspberry Pi liefert. Diese Pins dürfen aber nicht mit einem GPIO-Eingang verbunden werden.

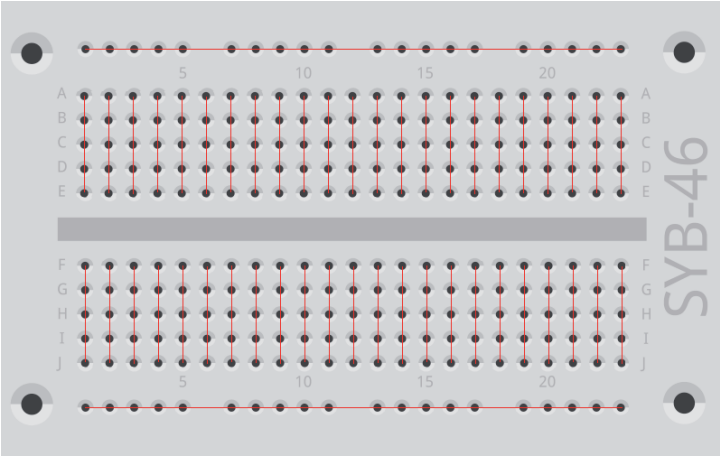
Bauteile

Mit diesen elektronischen Bauteilen, lassen sich die beschriebenen Experimente (und natürlich auch eigene) aufbauen. An dieser Stelle werden die Bauteile nur kurz vorgestellt. Die notwendige praktische Erfahrung im Umgang damit bringen dann die wirklichen Ex-perimente.

- 3x Steckplatine
- 1x LCD-Anzeige HD44780
- 1x Pfostenverbinder (16-polig)
- 1x Portexpander MCP 23017
- 10x LED
- 4x Taster
- 10x Widerstand 220 Ohm (Rot-Rot-Braun)
- 1x Widerstand 10 kOhm (Braun-Schwarz-Orange)
- 1x Widerstand 560 Ohm (Grün-Blau-Braun)
- 6x Widerstand 22 MOhm (Rot-Rot-Blau)
- 1x Potentiometer 15 kOhm
- 12x Verbindungskabel (Steckplatine – Raspberry Pi)
- Schaltdraht
- 2x Kabel mit Krokodilklemmen

Steckplatinen

Für den schnellen Aufbau elektronischer Schaltungen, ohne löten zu müssen, sind drei Steckplatinen vonnöten. Hier können elektronische Bauteile direkt in ein Lochraster mit Standardabständen ein-gesteckt werden. Bei diesen Platinen sind die äußeren Längsreihen mit Kontakten (X und Y) alle miteinander verbunden. Diese Kontaktreihen werden oft als Plus- und Minuspol zur Stromversorgung der Schaltungen genutzt. In den anderen Kontaktreihen sind jeweils fünf Kontakte (A bis E und F bis J) quer miteinander verbunden, wobei in der Mitte der Platine eine Lücke ist. So können hier in der Mitte größere Bauelemente, wie z. B. der Portexpander, eingesteckt und nach außen hin verdrahtet werden.



Verbindungskabel

Die farbigen Verbindungskabel haben alle auf einer Seite einen dünnen Drahtstecker, mit dem sie sich auf der Steckplatine einstecken lassen. Auf der anderen Seite befi ndet sich eine Steckbuchse, die auf einen GPIO-Pin des Raspberry Pi passt.

Weiterhin ist Schaltdraht notwendig. Damit stellen Sie kurze Verbindungsbrücken her, mit denen Kontaktreihen auf der Steckplati-ne verbunden werden. Schneiden Sie den Draht mit einem kleinen Sei-tenschneider je nach Experiment auf die passenden Längen ab. Um die Drähte besser in die Steckplatine stecken zu können, empfiehlt es sich, die Drähte leicht schräg abzuschneiden, sodass eine Art Keil entsteht.

Entfernen Sie an beiden Enden auf einer Länge von etwa einem halben Zentimeter die Isolierung.

Widerstände und ihre Farbcodes

Widerstände werden in der Digitalelektronik im Wesentlichen zur Strombegrenzung an den Ports eines Mikrocontrollers sowie als Vorwiderstände für LEDs verwendet. Die Maßeinheit für Widerstände ist Ohm. 1.000 Ohm entsprechen einem Kiloohm, abgekürzt kOhm, 1.000 kOhm entsprechen einem Megaohm, abgekürzt MOhm.

Die Widerstandswerte werden auf den Widerständen durch farbige Ringe angegeben. Die meisten Widerstände haben vier solcher Farbringe. Die ersten beiden Farbringe bezeichnen die Ziffern, der dritte einen Multiplikator und der vierte die Toleranz. Dieser Toleranzring ist meistens gold- oder silberfarben – das sind Farben, die auf den ersten Ringen nicht vorkommen, sodass die Leserichtung eindeutig ist. Der Toleranzwert selbst spielt in der Digitalelektronik kaum eine Rolle.

Farbe	Widerstandswert in Ohm			
	1. Ring (Zehner)	2. Ring (Einer)	3. Ring (Multiplikator)	4. Ring (Toleranz)
Silber			10^{-2} = 0,01	±10 %
Gold			10^{-1} = 0,1	±5 %
Schwarz		0	10^{0} = 1	
Braun	1	1	10^{1} = 10	±1 %
Rot	2	2	10^{2} = 100	±2 %
Orange	3	3	10^{3} = 1.000	
Gelb	4	4	10^{4} = 10.000	
Grün	5	5	10^{5} = 100.000	±0,5 %
Blau	6	6	10^{6} = 1.000.000	±0,25 %
Violett	7	7	10^{7} = 10.000.000	±0,1 %
Grau	8	8	10^{8} = 100.000.000	±0,05 %
Weiß	9	9	10^{9} = 1.000.000.000	

Die Tabelle zeigt die Bedeutung der farbigen Ringe auf Widerständen.

Es sind Widerstände in vier verschiedenen Werten:

Wert	1. Ring (Zehner)	2. Ring (Einer)	3. Ring (Multipl.)	4. Ring (Toleranz)	Verwendung
220 Ohm	Rot	Rot	Braun	Gold	Vorwiderstand für LEDs
560 Ohm	Grün	Blau	Braun	Gold	Vorwiderstand für Stromversorgung der LCD-Anzeige
10 kOhm	Braun	Schwarz	Orange	Gold	Pulldown-Widerstand für GPIO-Eingänge
20 MOhm	Rot	Rot	Blau	Gold	Widerstand für Pikey Pikey

Farbcodes der Widerstände

LEDs

An die GPIO-Ports können für Lichtsignale und Lichteffekte LEDs (LED = Light Emitting Diode, zu Deutsch Leuchtdiode) angeschlossen werden. Dabei muss zwischen dem verwendeten GPIO-Pin und der Anode der LED ein 220-Ohm-Vorwiderstand (Rot-Rot-Braun) eingebaut werden, um den Durchfl ussstrom zu begrenzen und damit ein Durchbrennen der LED zu verhindern. Zusätzlich schützt der Vorwiderstand auch den GPIO-Ausgang des Raspberry Pi, da die LED in Durchflussrichtung fast keinen Widerstand bietet und deshalb der GPIO-Port bei Verbindung mit Masse schnell überlastet werden könnte. Die Kathode der LED verbindet man mit der Masseleitung auf Pin 6 des Raspberry Pi.

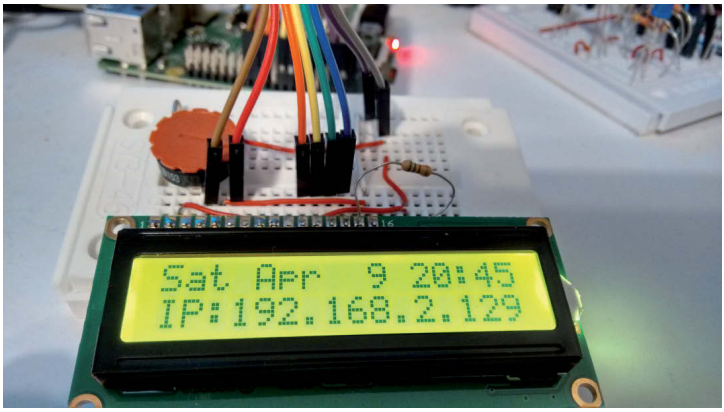
LED in welcher Richtung anschließen?

Die beiden Anschlussdrähte einer LED sind unterschiedlich lang. Der längere von beiden ist der Pluspol, die Anode, der kürzere die Kathode. Einfach zu merken: Das Pluszeichen hat einen Strich mehr als das Minuszeichen und macht damit den Draht etwas länger. Außerdem sind die meisten LEDs auf der Minusseite abgeflacht, wie eben ein Minuszeichen. Leicht zu merken: Kathode = kurz = Kante.

LC-Display (LCD)

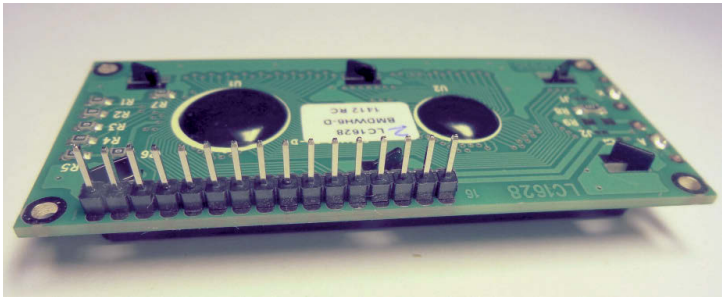
Im FRANZIS Lernpaket ist ein zweizeiliges LC-Display (LCD) enthalten, das wie die meisten derartigen Displays zum Quasi-Standard HD44780 kompatibel ist. Dabei handelt es sich um die Bezeichnung des in den Displays einge-bauten Controllers, der Zeichen in ein bis vier Zeilen darstellt, ohne dass

sich der Benutzer um die Ansteuerung der einzelnen Pixel zu kümmern braucht.



Zweizeiliges LC-Display auf einer Steckplatine am Raspberry Pi.

Das Display hat eine 16-polige Anschlussleiste. Löten Sie hier den mitgelieferten Pfostenverbinder mit den kürzeren Pins so an, dass die längeren Pins nach unten frei herausstehen. Damit wird das Display später auf die Steckplatine gesteckt.



Display mit angelötetem Pfostenverbinder.

Löten – einfach und richtig

Wer sich mit Hardwarebasteleien rund um den Raspberry Pi beschäftigt, wird ab und an auch mal etwas löten müssen. Für den Profi ist das kein Problem, für den Anfänger eigentlich auch nicht, wenn er ein paar wichtige Tricks beachtet. ***Löten ist einfach*** ist ein unterhaltsamer Comic mit Basiswissen für Hobbylöter: *bit.ly/178qobA*.

GPIO mit Python

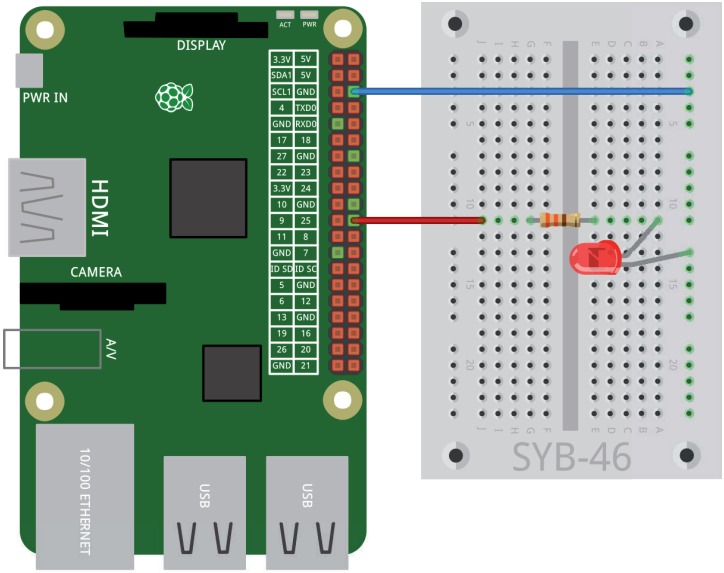
Um GPIO-Ports über Python-Programme zu nutzen, muss die Python-GPIO-Bibliothek installiert sein. In ganz alten Raspbian-Versionen musste man diese Bibliothek noch manuell installieren. Dies ist heute nicht mehr nötig, auch der früher benötigte sudo-Zugriff ist inzwischen Geschichte.

LED mit Python ein-/ausschalten

Schließen Sie wie auf dem Bild eine LED über einen 220-Ohm-Vorwiderstand (Rot-Rot-Braun) am GPIO-Port 25 (Pin 22) an und verbinden Sie den Minuspol der LED über die Masseschiene der Steckplatine mit der Masseleitung des Raspberry Pi (Pin 6).

Benötigte Bauteile

1x Steckplatine, 1x LED, 1x 220-Ohm-Widerstand (Rot-Rot-Braun), 2x Verbindungskabel



Eine LED am GPIO-Port 25.

fritzing

Das Programm `01led.py` lässt die LED 10-mal blinken:

```
001 #!/usr/bin/python
002 import RPi.GPIO as GPIO
003 import time
004
005 GPIO.setmode(GPIO.BCM)
006 GPIO.setup(25, GPIO.OUT)
007
008 for i in range(10):
009     GPIO.output(25, 1)
010     time.sleep(0.2)
011     GPIO.output(25, 0)
012     time.sleep(0.2)
013 GPIO.cleanup()
```

So funktioniert es

Das Beispiel zeigt die grundlegenden Funktionen der `RPi.GPIO`-

```
001 #!/usr/bin/python
```

Diese Zeile steht am Anfang (fast) jeden Python-Scripts, um die Datei als solches zu identifizieren. Sie wäre zwar in diesem Fall nicht unbedingt nötig, aber gewöhnen Sie sich an, sie immer als erste Zeile in ein Python-Script zu schreiben.

```
002 import RPi.GPIO as GPIO
```

Die Bibliothek `RPi.GPIO` muss in jedem Python-Programm importiert werden, in dem sie genutzt werden soll. Durch diese Schreibweise können alle Funktionen der Bibliothek über das Präfix `GPIO` angesprochen werden.

```
003 import time
```

Die Python-Bibliothek `time` hat nichts mit GPIO-Programmierung zu tun. Sie enthält Funktionen zur Zeit- und Datumsberechnung, unter anderem auch eine Funktion `time.sleep()`, mit der sich auf einfache Weise Wartezeiten realisieren lassen.

```
005 GPIO.setmode(GPIO.BCM)
```

Am Anfang jedes Programms muss definiert werden, wie die GPIO-Ports bezeichnet sind. Üblicherweise verwendet man die Standardnummerierung `BCM`.

Nummerierung der GPIO-Ports

Die Bibliothek `RPi.GPIO` unterstützt zwei verschiedene Methoden zur Bezeichnung der Ports. Im Modus `BCM` werden die bekannten GPIO-Portnummern genutzt, die auch auf Kommandozeilenebene oder in Shell-Skripten verwendet werden. Im alternativen Modus `BOARD` entsprechen die Bezeichnungen den Pin-Nummern von 1 bis 40 auf der Raspberry Pi-Platine. Dieser Modus wird nur von der Python-Bibliothek unterstützt, aber nicht von Scratch und anderen Programmen.

```
006 GPIO.setup(25, GPIO.OUT)
```

Die Funktion `GPIO.setup()` initialisiert einen GPIO-Port als Ausgang oder als Eingang. Der erste Parameter bezeichnet den Port je nach vorgegebenem Modus `BCM` oder `BOARD` mit seiner GPIO-Nummer oder Pin-Nummer. Der zweite Parameter kann entweder `GPIO.OUT` für einen Ausgang oder `GPIO.IN` für einen Eingang sein.

```
008 for i in range(10):
```

Jetzt startet eine Schleife, die zehnmal durchläuft. `for`-Schleifen wie diese sind in vielen Programmiersprachen bekannt. Der Zähler `i` nimmt nacheinander von `0` aufsteigend ganze Zahlen an. Die Schleife wird beendet, wenn der bei `range` angegebene Wert erreicht ist. Der eingerückte Programmcode wird in jedem Schleifendurchlauf wiederholt.

```
009   GPIO.output(25, 1)
```

Auf dem Port `25` wird eine `1` ausgegeben. Die angeschlossene LED leuchtet. Statt der `1` können auch die vordefinierten Werte `True` oder `GPIO.HIGH` ausgegeben werden.

Einrückungen sind in Python wichtig

In den meisten Programmiersprachen werden Programmschleifen oder Entscheidungen eingerückt, um den Programmcode übersichtlicher zu machen. In Python dienen diese Einrückungen nicht nur der Übersichtlichkeit, sondern sind auch für die Programmlogik zwingend nötig. Dafür braucht man hier keine speziellen Satzzeichen, um Schleifen oder Entscheidungen zu beenden.

```
010   time.sleep(0.2)
```

Diese Funktion aus der am Anfang des Programms importierten `time`-Bibliothek bewirkt eine Wartezeit von 0,2 Sekunden, bevor das Programm weiterläuft.

```
011    GPIO.output(25, 0)
```

Zum Ausschalten der LED gibt man den Wert `0` bzw. `False` oder `GPIO.LOW` auf dem GPIO-Port aus.

```
012    time.sleep(0.2)
```

Auch bei ausgeschalteter LED wartet das Programm wiederum 0,2 Sekunden. Danach ist die Schleife beendet, da dies die letzte eingerückte Zeile ist. Die Schleife beginnt von neuem und die LED schaltet sich wieder kurz ein.

```
013 GPIO.cleanup()
```

Am Ende eines Programms müssen alle GPIO-Ports zurückgesetzt werden. Diese Zeile erledigt das für alle vom Programm initialisierten GPIO-Ports auf einmal. Ports, die von anderen Programmen initialisiert wurden, bleiben unberührt. So wird der Ablauf dieser anderen, möglicherweise parallel laufenden Programme nicht gestört.

Python-Programme brauchen keine eigene Anweisung zum Beenden. Sie enden einfach nach dem letzten Befehl bzw. nach einer Schleife, die nicht mehr ausgeführt wird und der keine weiteren Befehle folgen.

GPIO-Warnungen abfangen

Soll ein GPIO-Port verwendet werden, der nicht zurückgesetzt, sondern möglicherweise von einem anderen oder abgebrochenen Programm noch geöffnet ist, kommt es zu Warnungen, die den Programmfluss nicht unterbrechen. Bei der Programmierung sind diese Warnungen nützlich, um Fehler zu finden. Im fertigen Programm können sie den Anwender verwirren. Die GPIO-Bibliothek bietet mit `GPIO.setwarnings(False)` die Möglichkeit, diese Warnungen zu unterdrücken.

5. Display-Projekte

GRAFISCHER SPIELWÜRFEL

Ein cooles Spiel braucht Grafik und nicht nur eine Textausgabe wie in Zeiten der allerersten DOS-Computer. Die Bibliothek PyGame liefert vordefinierte Funktionen und Objekte zur Grafikdarstellung und Spieleprogrammierung. Damit braucht man nicht mehr alles von Grund auf neu zu erfinden.

Für viele Spiele braucht man einen Würfel, aber oft ist gerade keiner griffbereit. Das nächste Programmbeispiel zeigt, wie einfach es ist, den Raspberry Pi mithilfe von Python und PyGame als Würfel zu benutzen:

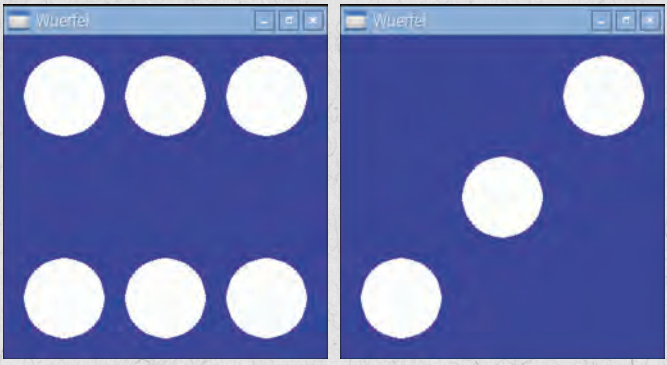


Abb. 8.1: Der Raspberry Pi als Würfel

Der Würfel soll möglichst einfach und mit nur einer Taste zu bedienen sein, und das zufällig gewürfelte Ergebnis soll grafisch wie ein »echter« Würfel angezeigt werden. Das folgende Programm `wuerfel.py` simuliert einen solchen Würfel auf dem Bildschirm.

```
# -*- coding: utf-8 -*-
import pygame, sys, random
from pygame.locals import *
pygame.init()

FELD = pygame.display.set_mode((320, 320))
pygame.display.set_caption("Wuerfel")

BLAU = (0, 0, 255); WEISS = (255, 255, 255)
P1 = ((160, 160)); P2 = ((60, 60)); P3 = ((160, 60));
P4 = ((260, 60))
P5 = ((60, 260)); P6 = ((160, 260)); P7 = ((260, 260))
mainloop = True

print "Beliebige Taste drücken, um zu würfeln, [Esc]
beendet das Spiel"

while mainloop:
    for event in pygame.event.get():
        if event.type == QUIT or (event.type == KEYUP
and event.key == K_ESCAPE):
            mainloop = False
        if event.type == KEYDOWN:
            FELD.fill(BLAU)
            ZAHL = random.randrange (1, 7); print ZAHL
            if ZAHL == 1:
                pygame.draw.circle(FELD, WEISS, P1, 40)
            if ZAHL == 2:
                pygame.draw.circle(FELD, WEISS, P2, 40)
                pygame.draw.circle(FELD, WEISS, P7, 40)
            if ZAHL == 3:
                pygame.draw.circle(FELD, WEISS, P1, 40)
                pygame.draw.circle(FELD, WEISS, P4, 40)
                pygame.draw.circle(FELD, WEISS, P5, 40)
            if ZAHL == 4:
                pygame.draw.circle(FELD, WEISS, P2, 40)
                pygame.draw.circle(FELD, WEISS, P4, 40)
                pygame.draw.circle(FELD, WEISS, P5, 40)
                pygame.draw.circle(FELD, WEISS, P7, 40)
            if ZAHL == 5:
                pygame.draw.circle(FELD, WEISS, P1, 40)
```

```
                pygame.draw.circle(FELD, WEISS, P2, 40)
                pygame.draw.circle(FELD, WEISS, P4, 40)
                pygame.draw.circle(FELD, WEISS, P5, 40)
                pygame.draw.circle(FELD, WEISS, P7, 40)
            if ZAHL == 6:
                pygame.draw.circle(FELD, WEISS, P2, 40)
                pygame.draw.circle(FELD, WEISS, P3, 40)
                pygame.draw.circle(FELD, WEISS, P4, 40)
                pygame.draw.circle(FELD, WEISS, P5, 40)
                pygame.draw.circle(FELD, WEISS, P6, 40)
                pygame.draw.circle(FELD, WEISS, P7, 40)
    pygame.display.update()
pygame.quit()
```

So funktioniert es

Dieses Programm zeigt zahlreiche neue Funktionen, besonders zur Grafi kausgabe mit der PyGame-Bibliothek, die natürlich nicht nur für Spiele, sondern auch für jegliche andere Grafi k auf dem Bildschirm verwendet werden kann.

```
import pygame, sys, random
from pygame.locals import *
pygame.init()
```

Diese drei Programmzeilen stehen am Anfang fast jedes Programms, das PyGame verwendet. Neben dem bereits erwähnten Modul `random` zur Erzeugung von Zufallszahlen werden das Modul `pygame` selbst sowie das Modul `sys` geladen, da es wichtige, von PyGame benötigte Systemfunktionen enthält, wie z. B. das Öff nen und Schließen von Fenstern. Alle Funktionen aus der PyGame-Bibliothek werden importiert, und danach wird das eigentliche PyGame-Modul initialisiert.

```
FELD = pygame.display.set_mode((320, 320))
```

Diese wichtige Funktion in jedem Programm, das eine grafi sche Ausgabe nutzt, defi n iert eine Zeichenfl äc he, ein sogenanntes Surface, die in unserem Beispiel die Größe von 320 x 320 Pixeln hat und den Namen `FELD` bekommt. Beachten Sie die Schreibweise in doppelten Klammern, die grundsätzlich für grafi sche Bildschirmkoordina-

ten verwendet wird. Ein solches Surface wird in einem neuen Fenster auf dem Bildschirm dargestellt.

```
pygame.display.set_caption("Wuerfel")
```

Diese Zeile trägt den Fensternamen ein.

```
BLAU = (0, 0, 255); WEISS = (255, 255, 255)
```

Diese Zeilen definieren die beiden verwendeten Farben Blau und Weiß. Man könnte auch jedes Mal im Programm die Farbwerte direkt angeben, was aber nicht gerade zur Übersicht beiträgt.

Darstellung von Farben auf dem Bildschirm

Farben werden in Python, wie auch in den meisten anderen Programmiersprachen, durch drei Zahlen zwischen 0 und 255 definiert, die die drei Farbanteile Rot, Grün und Blau festlegen. Bildschirme verwenden eine additive Farbmischung, bei der alle drei Farbanteile alle in voller Sättigung zusammen Weiß ergeben.

```
P1 = ((160, 160)); P2 = ((60, 60)); P3 = ((160, 60)); P4 =
((260, 60)); P5 = ((60, 260)); P6 = ((160, 260)); P7 = ((260,
260))
```

Diese Zeilen legen die Mittelpunkte der Würfelaugen fest. Auf dem 320 x 320 Pixel großen Zeichenfeld liegen die drei Achsen der Würfelaugen jeweils auf den Koordinaten `60`, `160` und `260`.

Das Koordinatensystem für Computergrafik

Jeder Punkt in einem Fenster bzw. auf einem Surface-Objekt wird durch eine x- und eine y-Koordinate bezeichnet. Der Nullpunkt des Koordinatensystems ist nicht, wie man in der Schule lernt, links unten, sondern links oben. Genau so, wie man einen Text von links oben nach rechts unten liest, erstreckt sich die x-Achse von links nach rechts, die y-Achse von oben nach unten.

Die sieben Punkte `P1` bis `P7` bezeichnen die in der Grafik angegebenen Mittelpunkte der Würfelaugen. Jedes Würfelauge hat einen Radius von 40 Pixeln. Bei 80 Pixeln Achsabstand bleiben demnach 20 Pixel zwischen den Würfelaugen und 20 Pixel zu den Fensterrändern.

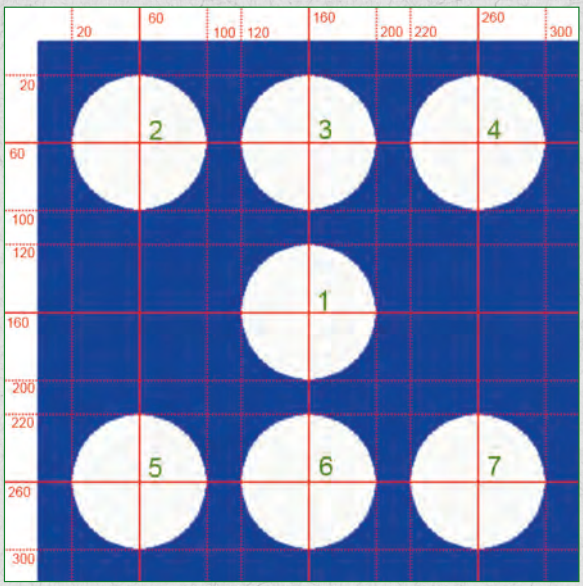


Abb. 8.2: Die Würfelaugen und ihre Koordinaten

An dieser Stelle wird mit den anderen Variablen auch noch eine Variable namens `mainloop` auf `True` gesetzt, die später für die Hauptschleife des Spiels benötigt wird.

`mainloop = True` Damit sind die Grundlagen geschaffen, und das eigentliche Spiel kann beginnen.

```
print "Beliebige Taste drücken, um zu würfeln, [Esc]
beendet das Spiel"
```

Diese Zeile erklärt dem Benutzer kurz, was zu tun ist. Bei jedem Druck auf eine beliebige Taste der Tastatur wird neu gewürfelt. `print` schreibt immer in das Python-Shell-Fenster, nicht in das neue grafische Fenster.

`while mainloop:` Jetzt beginnt die Hauptschleife des Spiels. In vielen Spielen wird eine Endlosschleife verwendet, die sich immer wiederholt und ständig irgendwelche Benutzeraktivitäten abfragt. Ir-

gendwo in der Schleife muss eine Abbruchbedingung definiert sein, die dafür sorgt, dass das Spiel beendet werden kann.

Dafür wird hier die Variable `mainloop` verwendet, die nur die beiden booleschen Werte `True` und `False` (Wahr und Falsch, Ein und Aus) annimmt. Sie steht am Anfang auf `True` und wird bei jedem Schleifendurchlauf abgefragt. Hat sie während der Schleife den Wert `False` angenommen, wird die Schleife vor dem nächsten Durchlauf beendet.

`for event in pygame.event.get():` Diese Zeile liest die letzte Benutzeraktivität und speichert sie als `event`. Im Spiel gibt es nur zwei Arten spielrelevanter Benutzeraktivitäten: Der Benutzer drückt eine Taste und würfelt damit, oder der Benutzer möchte das Spiel beenden.

```
        if event.type == QUIT or (event.type == KEYUP
and event.key == K_ESCAPE):
            mainloop = False
```

Es gibt zwei Möglichkeiten, das Spiel zu beenden: Man kann auf das x-Symbol in der oberen rechten Fensterecke klicken oder die Taste Esc drücken. Wenn man auf das x-Symbol klickt, ist der vom Betriebssystem gelieferte `event.type == QUIT`. Wenn man eine Taste drückt und wieder loslässt, ist der `event.type == KEYUP`. Zusätzlich wird in diesem Fall die gedrückte Taste in `event.key` gespeichert.

Die beschriebene `if`-Abfrage prüft, ob der Benutzer das Fenster schließen will oder (`or`) eine Taste gedrückt und losgelassen hat und (`and`) dies die Taste mit der internen Bezeichnung `K_ESCAPE` ist. Ist das der Fall, wird die Variable `mainloop` auf `False` gesetzt, was die Hauptschleife des Spiels vor dem nächsten Durchlauf beendet.

`if event.type == KEYDOWN:` Die zweite Art von Benutzeraktivität, die während des Spiels immer wieder und nicht nur einmal vorkommt, ist, dass der Benutzer eine Taste drückt. Dabei spielt es keine Rolle, welche außer der Esc-Taste das ist. Sowie eine Taste gedrückt wurde (`KEYDOWN`), wird ein wichtiger Programmteil in Gang gesetzt, der das Würfelergebnis erzeugt und auch darstellt.

```
FELD.fill(BLAU)
```

Als Erstes wird das als `FELD` bezeichnete Surface-Objekt, das eigentliche Programmfenster, mit der am Anfang als `BLAU` definierten Farbe gefüllt, um das vorherige Würfelergebnis zu übermalen.

```
ZAHL = random.randrange (1, 7)
```

Jetzt generiert die Zufallsfunktion `random` eine Zufallszahl zwischen `1` und `6` und speichert sie in der Variablen `ZAHL`.

```
print ZAHL
```

Diese Zeile schreibt nur zur Kontrolle das Würfelergebnis in das Python-Shell-Fenster. Sie können diese Zeile auch weglassen, wenn Sie auf die textbasierte Ausgabe verzichten wollen.

```
if ZAHL == 1:
    pygame.draw.circle(FELD, WEISS, P1, 40)
```

Jetzt folgen, alle nach dem gleichen Schema, sechs Abfragen. Wenn die zufällig gewürfelte Zahl einen bestimmten Wert hat, werden dementsprechend ein bis sechs Würfelaugen gezeichnet. Die dazu verwendete Funktion `pygame.draw.circle()` benötigt vier oder fünf Parameter:

- *Surface* gibt die Zeichenfläche an, auf der gezeichnet wird, im Beispiel das `FELD`.
- *Farbe* gibt die Farbe des Kreises an, im Beispiel die zuvor definierte Farbe `WEISS`.
- *Mittelpunkt* gibt den Mittelpunkt des Kreises an.
- *Radius* gibt den Radius des Kreises an.
- *Dicke* gibt die Linienstärke der Kreislinie an. Wird dieser Parameter weggelassen oder auf `0` gesetzt, wird der Kreis gefüllt.

Ist eine der `if`-Bedingungen erfüllt, sind die Würfelaugen zunächst nur auf einer virtuellen Zeichenfläche gespeichert.

`pygame.display.update()` Erst diese Zeile am Schleifenende aktualisiert die Grafik auf dem Bildschirm. Nun sind die Würfelaugen wirklich zu sehen.

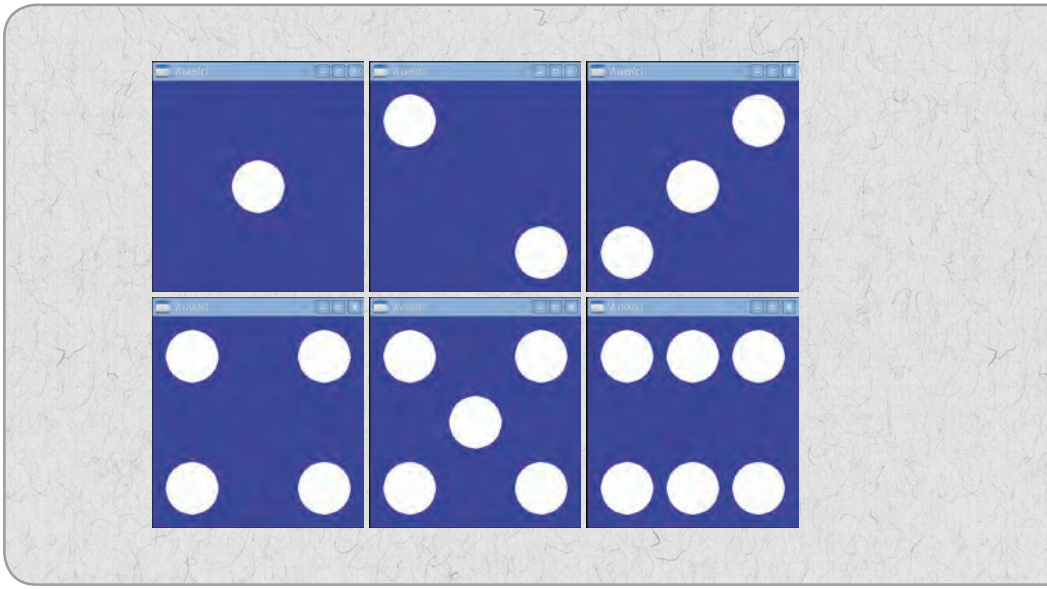


Abb. 8.3: Die sechs möglichen Würfelergebnisse

Die Schleife startet sofort von Neuem und wartet wieder auf einen Tastendruck des Benutzers. Falls während der Schleife `mainloop` auf `False` gesetzt wurde, weil der Benutzer das Spiel beenden will, wird die Schleife kein weiteres Mal durchlaufen, stattdessen wird die folgende Zeile ausgeführt:

```
pygame.quit()
```

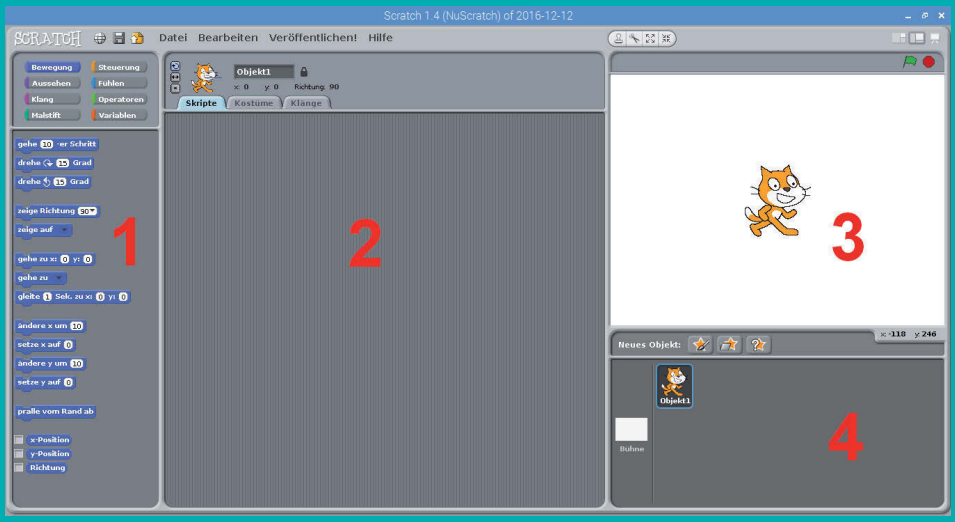
Diese Zeile beendet das PyGame-Modul, was auch das grafische Fenster schließt und danach das ganze Programm.

6. Projekte mit Scratch

DAS ERSTE PROJEKT MIT SCRATCH

Scratch ist eine intuitive Programmierumgebung, mit der Kinder und Programmiereinsteiger schnell Ideen umsetzen können, ohne sich zuerst mit Programmiertheorie auseinandersetzen zu müssen. Scratch ist ein Projekt der Lifelong-Kindergarten-Group am Media-Lab des MIT und auf dem Raspberry Pi vorinstalliert. Passend für die Zielgruppe der Kinder und Jugendlichen eignet sich Scratch besonders für interaktive Animationen und Spiele. Eine grafische Oberfläche gibt bereits alle Grundlagen vor, sodass man sich um die Programmierung der Benutzeroberfläche des eigenen Programms keine Gedanken zu machen braucht. Die Scratch-Skripte werden nicht als Text geschrieben, sondern aus vorgefertigten Elementen zusammengeklickt.

Starten Sie *Scratch* über das Menü *Entwicklung*. Scratch startet mit einem viergeteilten Programmfenster:



Scratch starten

1. Links befindet sich die Blockpalette mit allen Elementen, aus denen ein Scratch-Skript zusammengesetzt werden kann. Da ausschließlich vordefinierte Blöcke verwendet werden, kann der Benutzer keine Syntaxfehler machen, die beim Einstieg in andere Programmiersprachen sehr ärgerlich und frustrierend sind.
2. Das mittlere Feld ist am Anfang noch leer. Hier entsteht später das Skript.
3. Rechts oben ist die sogenannte Bühne, die Oberfläche, auf der das Skript abläuft. Die Katze, die dort zu sehen ist, stellt beispielhaft ein Objekt dar, das mit Scratch animiert werden kann.
4. Das Objektfenster unten rechts zeigt die im Skript verwendeten Objekte. Hier können Sie später auch selbst eigene Objekte gestalten.

Unterschied: Scratch 1 – Scratch 2

Seit der ersten Raspbian-Version war die Programmiersprache Scratch in der Version 1.x vorinstalliert. Für PCs gibt es seit einigen Jahren die neue Version Scratch 2 mit deutlich mehr Möglichkeiten. Hier lassen sich unter anderem eigene Funktionsblöcke erstellen und Erweiterungen einbinden.

Scratch 2 läuft auf dem PC online im Browser. Dafür ist deutlich mehr Rechenleistung erforderlich, als ein Raspberry Pi bieten kann, Seit NOOBS 2.4.0 ist in Raspbian eine Version von Scratch 2 vorinstalliert, die offline ohne Browser läuft und so mit der Leistung eines Raspberry Pi 3 problemlos auskommt. Mit Scratch 2 ist die Hardwaresteuerung über die GPIO-Schnittstelle wesentlich einfacher geworden. Leider fehlen aber bis jetzt Funktionen zur erweiterten GPIO-Steuerung, wie unter anderem PWM-Signale, die Möglichkeit zum Abschalten der internen Pull-down-Widerstände und die Schnittstelle zwischen Scratch und Python, weshalb wir in diesem Maker Kit weiterhin das „klassische" Scratch 1.4 verwenden.

In einem allerersten einfachen Skript soll die Katze einmal im Kreis herum laufen und dabei ihre Farbe verändern.

1. Klicken Sie im Objektfenster unten rechts auf die Katze. Diese wird hervorgehoben und erscheint als *Objekt1* oberhalb des Skriptfensters. Alle befehle im Skriptfenster beziehen sich dann auf dieses Objekt.
2. Klicken Sie in der Blockpalette oben auf das gelbe Symbol *Steuerung*. Damit werden die Blöcke zur Programmsteuerung angezeigt.

3 Ziehen Sie den abgebildeten Block aus der Blockpalette in das Skriptfenster. Auf der Scratch-Bühne ist oben rechts ein grünes Fähnchen. Dieses dient üblicherweise dazu, ein Programm zu starten. Das abgebildete Element bewirkt, dass die folgenden Skriptelemente ausgeführt werden, wenn der Benutzer auf das grüne Fähnchen klickt.

4 Die kreisförmige Bewegung wird aus einzelnen Gehen- und Drehen-Schritten zusammengesetzt. Diese werden so oft wiederholt, bis die Katze einen ganzen Kreis gegangen ist. Ziehen Sie für die Wiederholungsschleife den abgebildeten Block in das Skriptfenster und dokken Sie ihn unten an das dort bereits vorhandene Element an.

5 In jedem Bewegungsschritt soll sich die Katze um 15° drehen. Dabei dreht sie sich in 24 Schritten genau einmal. Tippen Sie in das weiße Zahlenfeld des *Wiederhole*-Blocks und ändern Sie den vorgegebenen Wert auf *24*.

6 Damit die Katze eine Kreisbewegung ausführt, muss sie zunächst einen Schritt gehen, sich danach um 15° drehen, wieder einen Schritt gehen usw. Klicken Sie oben in der Blockpalette auf das blaue Symbol *Bewegung* und ziehen Sie den abgebildeten Block in das Skriptfenster. Docken Sie ihn innerhalb der Schleife an. Ändern Sie dann noch die Schrittweise auf 20.

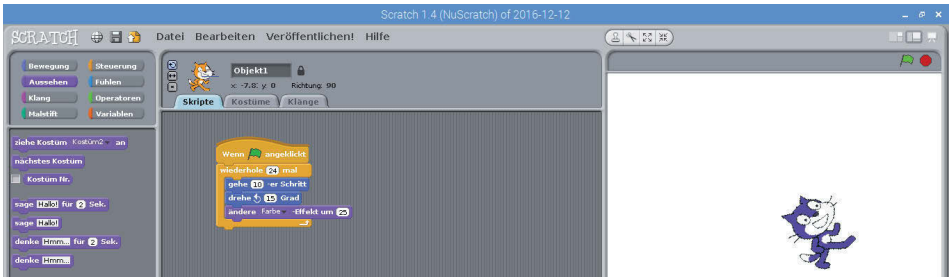
7 Ziehen Sie danach den Block für die Drehbewegung gegen den Uhrzeigersinn in das Skriptfenster. Platzieren Sie ihn so über den vorhandenen Blöcken, dass er sich innerhalb der Schleife nach der Gehbewegung einklinkt.

8 Das Skript sollte jetzt wie abgebildet aussehen. Probieren Sie aus, ob es auch wie erwartet funktioniert. Klicken Sie dazu rechts oberhalb der Bühne auf das grüne Fähnchen. Die Katze wird im Kreisgehen.

9 Jetzt fehlt nur noch die gewünschte Veränderung der Farbe. Klicken Sie dazu oben in der Blockpalette auf das violette Symbol *Aussehen*. Jetzt werden Blöcke angeboten, die das Aussehen des aktiven Objekts beeinflussen. Ziehen Sie den abgebildeten Block in das Skriptfenster in die Wiederholung hinter den *Drehe*-Block.

10 Lassen Sie das Skript jetzt wieder ablaufen, ändert die Katze im Laufe ihrer Bewegung zyklisch ihre Farbe. Am Ende der 24 Wiederholungen

steht die Katze wieder an ihrer ursprünglichen Position und hat auch wieder ihre ursprüngliche Farbe.



Gehen, drehen, Farbe ändern im Scratch-Programm 02katze1.

SCRATCH UND GPIO

Scratch bietet in der Version 1.4, die im aktuellen Raspbian Betriebssystem vorinstalliert ist, eine eigene Unterstützung für die GPIO-Schnittstelle. Früher mussten dazu externe Funktionsbibliotheken eingebunden und spezielle Einstellungen vorgenommen werden.

Um die GPIO-Unterstützung in Scratch nutzen zu können, muss der GPIO-Server über den Menüpunkt *Bearbeiten / Start GPIO server* gestartet werden. Denken Sie bei jedem neuen Scratch-Programm daran, dies zu überprüfen. Wenn die GPIO-Funktionen bereits aktiv sind, ändert sich dieser Menüpunkt automatisch in *Stop GPIO server*.

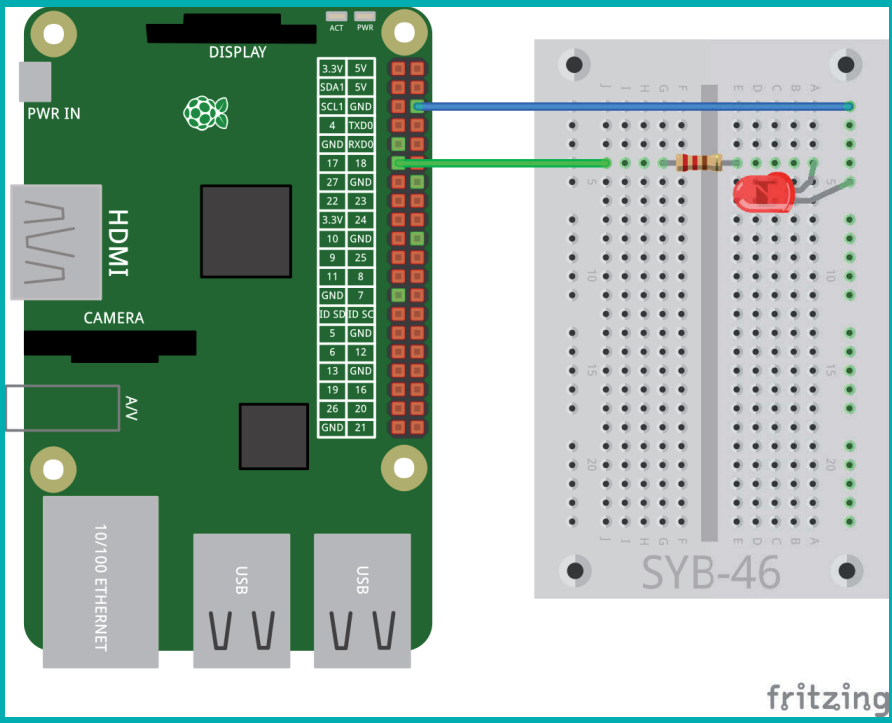
GPIO-Server in Scratch starten

Die erste LED blinkt in Scratch

Im Vergleich zu, dem Versuch, mit einem Windows PC eine extern angeschlossene LED zum Blinken zu bringen, ist Scratch wirklich ganz einfach. Bauen Sie die abgebildete Schaltung auf einer Steckplatine auf.

Benötigte Bauteile

1x Steckplatine, 1x LED rot, 1x 220-Ohm-Widerstand (Rot-Rot-Braun), 2x Verbindungskabel



Die LED mit Vorwiderstand am Raspberry Pi

Öffnen Sie über den Menüpunkt *Datei/Öffnen* das Skript `031ed2` in Scratch und Starten Sie es mit einem Klick auf das grüne Fähnchen. Die LED blinkt. Sie können das Programm natürlich auch selbst aus Scratch-Blöcken zusammensetzen.

So funktioniert das Programm

Das Programm startet wie die meisten Scratch-Programme mit dem Block *Wenn grünes Fähnchen angeklickt*. Dieser ist in Scratch auf der Blockpalette *Steuerung* zu finden. Der Block ist oben rund, passt also unter keinen anderen Block. Er muss immer als Erstes gesetzt werden.

Scratch verwendet für die GPIO-Funktionen den Steuerungsblock *sende...an alle*.

Dieser Block enthält ein Feld, in dem freier Text eingegeben werden kann. Klicken Sie darauf, erscheint eine Liste der zuletzt verwendeten Eingaben. Ein Klick auf *Neu/edit...* öffnet ein Eingabefeld für neuen Text.

Schreiben Sie in dieses Feld: `config17out`. Das definiert den GPIO-Pin 17 als Ausgang. Jeder GPIO-Pin muss, bevor er verwendet werden kann, als Ausgang (z. B. für LEDs) oder als Eingang (z. B. für Taster) definiert werden.

Jetzt soll eine Endlosschleife starten, die die LED blinken lässt, bis Sie auf das rote Stopp-Symbol klicken.

Innerhalb der Schleife wird als Erstes der GPIO-Pin 17 eingeschaltet. Ziehen Sie dazu wieder einen *sende...an alle* Block in die Schleife hinein und schreiben Sie in das Textfeld dieses Blocks: `gpio17on`. Damit wird der GPIO-Pin 17 eingeschaltet und die LED leuchtet.

Danach soll das Programm eine Sekunde warten, bis die LED wieder ausgeschaltet wird. Fügen Sie dazu einen *warte...Sek* Block ein und schreiben Sie in das Textfeld: `1`.

Um die LED auszuschalten, fügen Sie wieder einen *sende...an alle* Block hinter der Wartezeit in die Schleife ein. Schreiben Sie hier in das Textfeld: `gpio17off`. Damit wird der GPIO-Pin 17 wieder ausgeschaltet.

Zum Schluss kommt wieder ein *warte...Sek* Block mit einer Sekunde. Das bewirkt, dass die LED eine Sekunde ausgeschaltet bleibt, bevor die Schleife wieder neu beginnt und die LED einschaltet.