```
1   const http = require('http');
2
3   const hostname = '127.0.0.1';
4   const port = 3000;
5
6   const server = http.createServer((req, res) => {
7     res.statusCode = 200;
8     res.setHeader('Content-Type', 'text/plain');
9     res.end('Hello World\n');
10  });
11
12  server.listen(port, hostname, () => {
13    console.log('Server running at http://${hostname}:${port}/');
14  });
```

Dabei lauscht ein Webserver auf Port 3000, der uns mit „Hallo Welt“ begrüßt. Wem das zu kompliziert ist, der verwendet ein einfaches `console.log('Hello World\n');`.

Auf einen Blick

Pro und contra:

(+) Sprache für alle Webentwickler

(+) große Standardbibliothek

(–) Laufzeitumgebung zur Ausführung benötigt (Browser oder Node.js)

Anwendungsgebiete:

Node.js ist die Verbindung zwischen klassischen Programmen und dem Web. JavaScript ist die Sprache, die hinter MakeCode steckt (siehe Abschnitt 2.4.4).

Sonstiges:

Mit http://johnny-five.io und Firmata lassen sich auch viele Boards mit Node.js steuern.

2.4.4 Grafische Programmiersprachen

Etwas aus der Reihe der klassischen Programmiersprachen fallen die sogenannten grafischen Programmiersprachen, oft auch Blocksprachen genannt. Die bekanntesten Vertreter sind Scratch (*https://scratch.mit.edu*), Microsoft MakeCode (*https://www.makecode.com*) und OpenRoberta Lab (*https://lab.open-roberta.org*). Hierbei muss man nicht die Syntax einer gewissen Programmiersprache lernen, sondern klickt die Bestandteile seines Programms einfach zusammen. Die einzelnen Bausteine, wie z. B. Schleifen, If-Abfragen oder Variablen, sind so gestaltet, dass sie nur in einer syntaktisch korrekten Reihenfolge miteinander verknüpft werden können, ähnlich einem Puzzle. Der Sprachumfang steht normalen Programmiersprachen in der Regel in nichts nach, auch wenn manche Dinge textuell

einfacher bzw. kürzer zu realisieren sind. Somit lassen sich prinzipiell auch komplexere Programme grafisch programmieren, jedoch werden diese schnell unübersichtlich.

Anwendung finden die grafischen Programmiersprachen vor allem im Lernbereich, speziell wenn es darum geht, Kinder an das Programmieren heranzuführen. Das Konzept von Variablen, Abfragen und Schleifen lässt sich auf diese Weise einfacher lernen als in einem einfachen Texteditor. Dass diese grafischen Sprachen aber nicht nur für Kinder geeignet sind, merkt man recht schnell, wenn man sie ein wenig ausprobiert. Innerhalb von wenigen Minuten hat man seine Lösung mit den entsprechenden Blöcken einfach, schnell und komfortabel zusammengepuzzelt.

Wichtig ist jedoch, dass die entsprechende Entwicklungsumgebung viele Bibliotheken als Bausteine zur Verfügung stellt. Beim Calliope mini ist dies mit MakeCode sehr gut gelöst, in dieser Umgebung werden sehr viele Features des Calliope-mini-Boards als Bausteine zur Verfügung gestellt.

Bild 2.6 zeigt das obligatorische „Hello World“ in Scratch.

Bild 2.6 „Hello World“ in Scratch

Auf einen Blick

Pro und contra:

(+) kein Vorwissen notwendig

(+) einfach, schnell und angenehm

(+) für Kinder und Jugendliche geeignet

(–) Die Möglichkeiten sind teilweise eingeschränkt.

(–) Programme werden schnell unübersichtlich.

Anwendungsgebiete:

Grafische Programmiersprachen eignen sich für Einsteiger, die Programmiergrundlagen erlernen und direkt praktisch anwenden möchten (z. B. Ansteuerung eines Servomotors).

Sonstiges:

https://code.org bietet verschiedene Kurse mithilfe von grafischer Programmierung an. Wer lieber seine eigene Android-App schreiben möchte, sollte mal bei *http://appinventor.mit.edu/explore* vorbeischauen.

3.1 Sicherheit: Security vs. Safety

Wenn man in den Medien von Sicherheitsproblemen im IoT-Umfeld hört, ist damit meistens der Schutz (von Daten) vor unbefugtem Zugriff von Angreifern gemeint. Das Englische bietet hier den Begriff *Security* an, im Deutschen wird er manchmal auch als Angriffssicherheit umschrieben.

Daneben gibt es noch einen weiteren Aspekt der Sicherheit: die Betriebssicherheit (im Englischen *Safety* genannt). Hierbei geht es darum, dass vom Betrieb eines Geräts keine Gefahr ausgeht, d. h., der Benutzer sollte bei ordnungsgemäßem Gebrauch nicht verletzt werden. Als einfaches Beispiel dient hier eine Akku-Stichsäge: Das versehentliche Einschalten beim Transport könnte ernsthafte Verletzungen verursachen und ist damit ein hohes Safety-Risiko. Um dies zu verhindern, muss man erst einen zweiten Sicherheitsschalter zur Seite drücken, bevor man den „Abzug" betätigen kann.

Neben den normalen Betriebsrisiken wie der 220-V-Netzspannung ist Safety für deine IoT Gadgets insofern relevant, als du dir immer überlegen musst: Was könnte passieren, wenn sich das Gadget nicht so verhält, wie es soll, oder eine Aktion versehentlich oder willentlich durch einen Angreifer ausgelöst wird? Brennt die Küche ab, wenn die Kaffeemaschine unbeabsichtigt eingeschaltet wird? Steht die Haustür plötzlich offen, wenn der Strom ausfällt? Kann ich die Türe überhaupt noch öffnen, wenn der Strom ausfällt? Diese Fragen solltest du dir bei jedem Gerät stellen, dass du in Betrieb nimmst – und als Maker gleich zweimal.

3.2 Security-Analyse am Beispiel des Raspberry Pi

Mit mehr als 23 Millionen verkauften Stück ist der Raspberry Pi das perfekte Beispiel, um dir die Security eines Systems im Detail anzusehen. Dazu setzt du einfach mal die Brille eines Angreifers auf und suchst nach verwundbaren Systemen. Es ist davon auszugehen, dass die meisten Anwender die Standarddistribution Raspbian verwenden.

Um nicht in rechtliche Schwierigkeiten zu kommen, bitten wir dich, folgende Schritte nur theoretisch zu betrachten und sämtliche Versuche nur mit deinen eigenen Systemen durchzuführen! Ohne Erlaubnis des jeweiligen Eigentümers erfüllt man sehr schnell verschiedenste Tatbestände der Computerkriminalität.

Sucht man mit der „Hacker-Suchmaschine“ Shodan.io nach „Raspbian“, findet man aktuell ca. 159 000 öffentlich erreichbare Raspberry-Pi-Systeme[1] (Bild 3.1). Dies sagt zwar noch nichts über die Sicherheit der Systeme aus, aber zumindest entkräftet dies den Mythos „die IP meines Raspberry Pi errät doch sowieso keiner“.

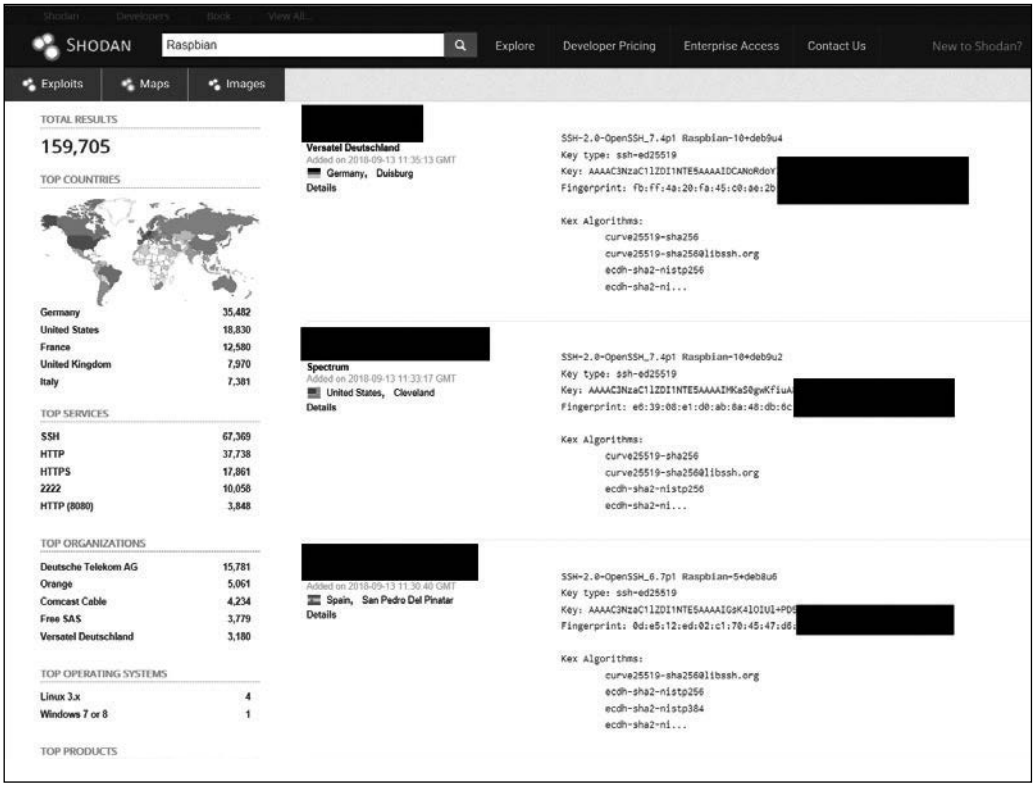

Bild 3.1 Screenshot der „Hacker-Suchmaschine“ Shodan.io

Security-Tipp: Kein direkter Zugriff über das Internet

Jedes System, das im Internet direkt erreichbar ist, wird über entsprechende Suchmaschinen auch gefunden. Es ist in der Regel deutlich sicherer, das System hinter einem Router zu betreiben. Die Kommunikation des Systems mit dem Internet wird dadurch nicht beeinträchtigt. Sollte dennoch ein Zugriff von außen notwendig sein, sollte man entweder nur die Dienste, die wirklich benötigt werden, per Port Forwarding freischalten oder besser gleich ein VPN benutzen, wie in Abschnitt 3.4.2 beschrieben.

[1] Es ist davon auszugehen, dass ein Teil dieser Systeme bewusst offen und anfällig ist (sogenannte Honey Pots). Diese Honey Pots werden aufgestellt, um mehr über aktuelle Angriffe und Methoden zu erfahren und auch um Hacker zu überführen. Das ist ein weiterer Grund, warum man nicht an beliebigen Systemen herumprobieren sollte.

Schränkt man die Suche etwas weiter auf „Raspbian SSH“ ein, finden sich nur noch ca. 91 000 Systeme. Damit fallen ca. 40 % der Systeme aus dem Angriffsszenario heraus, weil der SSH-Dienst gar nicht erst aktiviert oder nicht von außen zugänglich ist.[2]

Security-Tipp: Unnötige Dienste ausschalten

Was nicht läuft und was nicht installiert ist, kann auch nicht angegriffen werden. Solltest du einen öffentlichen Dienst auf deinem Raspberry Pi bereitstellen wollen, solltest du wirklich nur diesen per *Port Forwarding* freischalten.

Der SSH Dienst kann generell als sehr sicher angesehen werden, sofern er aktuell ist, richtig konfiguriert wurde und die User Accounts über sichere Passwörter verfügen.

Für jeden der Einträge könnte man nun in einer CVE-Datenbank (Common Vulnerabilities and Exposures)[3] nachschauen, ob es passende Sicherheitslücken und Exploits[4] gibt. Hierfür gibt es mit Nessus und Metasploit zwei populäre Tools, die einem die Arbeit erleichtern. Mit ein bisschen Glück hat man den passenden Exploit angewendet, und schon hat man ein System erfolgreich übernommen. Da dies jedoch nicht unsere Systeme sind und wir nicht in rechtliche Schwierigkeiten kommen wollen, lassen wir diesen Schritt hier aus. Bitte probiere das – wenn überhaupt – ausschließlich an eigenen Systemen aus!

Security-Tipp: Systeme aktuell halten

In jeder Software finden sich früher oder später Bugs und Fehler. Diese werden in neueren Versionen behoben. Deshalb solltest du immer schauen, ob das System auf dem neuesten Stand ist, und Updates installieren. Auf einem Raspbian-System ist dies einfach: `sudo apt update && sudo apt upgrade`. Auf den sonstigen Embedded-Systemen ist dies leider etwas schwieriger.

Doch bevor ein Angreifer zu den schweren Geschützen der Exploits greift, wird er erst einmal einen deutlich leichteren Weg versuchen: Standard-Logins und -passwörter sowie Brute-Force-Angriffe auf das Passwort. Für unsere Raspberry-Pi-Systeme ist dies relativ einfach, denn Raspbian besitzt einen Standard-User namens `pi` mit dem Passwort `raspberry`. In fast jeder Raspberry-Pi-Anleitung wird das Ändern des Passwortes als einer der ersten Schritte beschrieben. Dennoch sieht man das Standardpasswort in der Praxis immer und immer wieder. Aus diesem Grund zeigen neuere Versionen von Raspbian auch eine Warnmeldung an, wenn man noch immer das Standardpasswort verwendet und SSH aktiv ist. Diese Standardkombination bei jedem der Systeme einmal auszuprobieren, ist

[2] Oder auf einem anderen Port lauscht, der nicht gescannt wurde. Das Ändern des Standard-Ports eines Dienstes hilft zwar gegen diverse Skript-Kiddie-Attacken, aber auch nur bedingt, denn Shodan.io listet für Raspbian ssh port:„2222“ immer noch ca. 10 000 Ergebnisse!

[3] Zum Beispiel: *http://cve.mitre.org*

[4] Tools und Software, um Sicherheitslücken auszunutzen

4.1.6 Calliope mini

Der Calliope mini[23] tanzt bei unseren Hardwareplattformen ein wenig aus der Reihe. Schon wenn man die Packung öffnet, wird man von einem sternförmigen Board, einer farbenfrohen Broschüre und einer Menge Sticker überrascht (Bild 4.19). Versorgt man das Board mit Strom, entweder über das mitgelieferte Batterie-Pack oder über das ebenfalls mitgelieferte Micro-USB-Kabel[24], wird man vom Board mit einem Smiley auf dem LED-„Display“ begrüßt und gleich aufgefordert, ein paar Features des Boards interaktiv auszuprobieren.

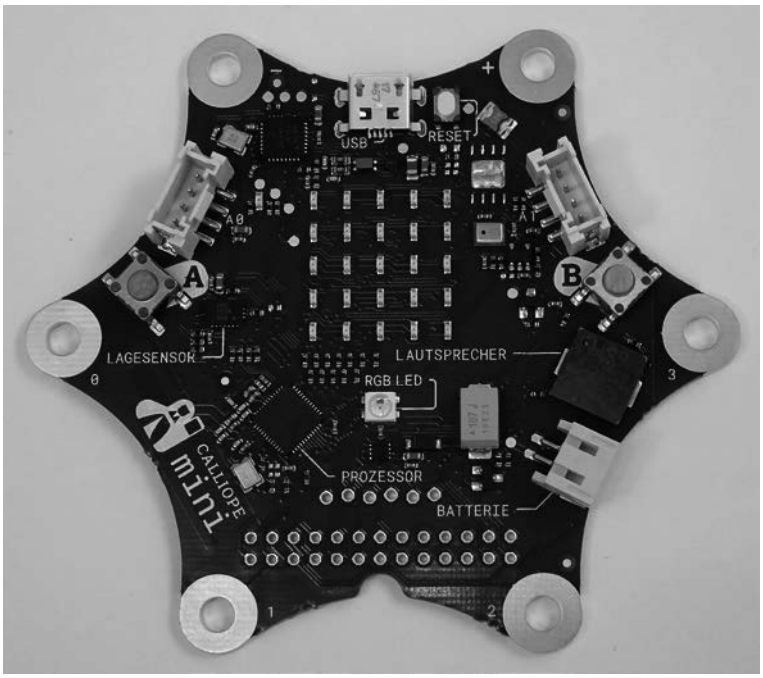

Bild 4.19 Calliope mini: Alle Komponenten auf dem Board sind beschriftet (© Sonja Hemmersbach).

Du merkst schon, hier ist alles ganz anders als bei den anderen Plattformen, denn die Zielgruppe ist eine andere: Der Calliope mini wurde speziell dafür entwickelt, Kindern und Jugendlichen das Programmieren näherzubringen – mit Spiel, Spaß und Spannung. Die Programmierung gestaltet sich vollkommen anders als bei den anderen Boards: online im Browser, und zwar grafisch.

Ein vollständiges Thermometer ist in einem der vielen Online-Editoren[25] schnell zusammengeklickt (Bild 4.20). Anschließend klickst du auf Herunterladen und speicherst die

[23] So lautet der offizielle Name, dennoch wird das mini meistens weggelassen – schreibfaule Programmierer eben.
[24] Natürlich auch im Calliope-Design.
[25] In diesem Fall mit Open Roberta: *http://lab.open-roberta.org*

.hex-Datei auf den Calliope mini, der sich wie ein ganz normaler USB-Stick als Laufwerk beim Betriebssystem meldet. Nach wenigen Momenten startet der Calliope mini neu und zeigt dir nun immer die aktuelle Temperatur an. Ziemlich einfach, oder?

Bild 4.20 Thermometer mit dem Calliope mini

Dazu gibt es noch eine ganze Menge sehr gut aufbereiteter Projekte und Schulungsunterlagen auf Deutsch, denn der Calliope mini ist „Made in Germany“ und speziell für die deutsche Schullandschaft entwickelt worden. Es gibt sogar ein Schulbuch vom Cornelsen Verlag dazu: *Coden mit dem Calliope mini* (ISBN 978-3-06-600012-2), einsetzbar für die dritte Klasse.

Dennoch ist die Idee des Calliope mini nicht ganz neu, sondern basiert auf einer Initiative der guten alten BBC aus Großbritannien: dem BBC micro:bit. Entwickelt wurde das kleine Board dort 2015 und wird seit 2016 kostenlos, mit Unterstützung von Sponsoren und Partnern, an Schüler der siebten Klassen verteilt. Im Gegensatz zum Calliope bietet der micro:bit etwas weniger Peripherie, ansonsten sind die Boards weitestgehend miteinander kompatibel. Du solltest auf der Suche nach Informationen zum Calliope auch immer mal mit „microbit“ als Suchwort suchen.

Auch wenn du vermutlich nicht mehr ganz der eigentlichen Zielgruppe angehörst, solltest du die Calliope-Plattform nicht als Spielzeug abtun. Neben einer kleinen ARM-Cortex-M0-CPU bringt die Plattform allerhand Peripherie mit: Angefangen bei Temperatur- und Beschleunigungssensor über Gyroskop und Magnetometer, Lautsprecher und Mikrofon bis hin zur RGB-LED und einer 5 × 5 LED-Matrix ist alles an Bord. Bluetooth Low Energy (BLE) wird natürlich auch direkt unterstützt.

Nach kurzer Eingewöhnung macht das „Hacken“ mit dem Calliope unglaublich viel Spaß, speziell wenn man nur schnell etwas ausprobieren möchte: den Online-Editor aufgerufen, ein Programm zusammengeklickt und per Speichern auf den Calliope geflasht, fertig. Einzig der Preis von knapp 35 € erscheint auf den ersten Blick relativ hoch, speziell im Vergleich zum ESP32 oder Raspberry Pi Zero W. Dafür erhält man jedoch eine sehr hochwertige Dokumentation auf Deutsch, und die Peripherie kostet allein auch schon einiges. Außerdem ist das Design toll. Ob sich der Calliope mini langfristig durchsetzen wird und sich jedes Kind damit für das Programmieren begeistern lässt, bleibt abzuwarten. Doch nur zur Erinnerung: Der Raspberry Pi war ursprünglich auch als Lernplattform für Kids gedacht. Vielleicht steht die nächste Revolution tatsächlich vor der Tür.

5.1.6 Fertiges Programm der Spardose

Das fertige Programm ist nachfolgend dargestellt. Einzelne, wichtige Codestellen wurden innerhalb des Codes dokumentiert. Den Code findest du auch auf GitHub unter dem Namen „IOTASavingsBox“.

```
#include <TimeLib.h>
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Wire.h>  // Only needed for Arduino 1.6.5 and earlier
#include "SSD1306.h" // alias for `#include "SSD1306Wire.h"`
#include "OLEDDisplayUi.h"
#include "images.h"
#include <ArduinoJson.h>

// Wifi-Settings, please change
const char* ssid = "MyWLAN";
const char* password = "secret";

// IOTA Settings
char server[] = "my.iotaserver.de"; // Please use some other IOTA server

int status = WL_IDLE_STATUS;
WiFiClient client;

String balance = "";
float addressbalance = 0;

//Adresse zum Testen:
//OHKQZAWRCGOHPHGEJFWAXL9JAWVUUMBCNMALLSTWLTKHOLFALWXYCCJRRHFRLGEZIWDTJXRUMNNQABEVX

String ip = "";
String address = "";
String addresscache = "";

bool thereWasAClientCall = false;

ESP8266WebServer webserver(80);    // Set Webserver Port (e.g. 80)

// Initialize the OLED display using Wire library
SSD1306  display(0x3c, D1, D2);
OLEDDisplayUi ui ( &display );

int screenW = 128;
int screenH = 64;
int screenCenterX = screenW/2;
int screenCenterY = ((screenH-16)/2)+16;
int clockRadius = 23;

// utility function for digital clock display: prints leading 0
String twoDigits(int digits){
  if(digits < 10) {
    String i = '0'+String(digits);
    return i;
  }
  else {
```

```
271
272     // if the server's disconnected, stop the client
273     if (thereWasAClientCall)
274     {
275       if (!client.connected()) {
276         Serial.println();
277         Serial.println("Disconnecting from server...");
278         client.stop();
279       }
280     }
281 }
```

5.1.7 Fertige Umsetzung der Spardose

Die fertige Umsetzung besteht aus dem Hardware-Gadget und der Weboberfläche, die über den Browser erreicht werden kann. Bild 5.5 und Bild 5.6 zeigen den Prototypen und das Web-Frontend.

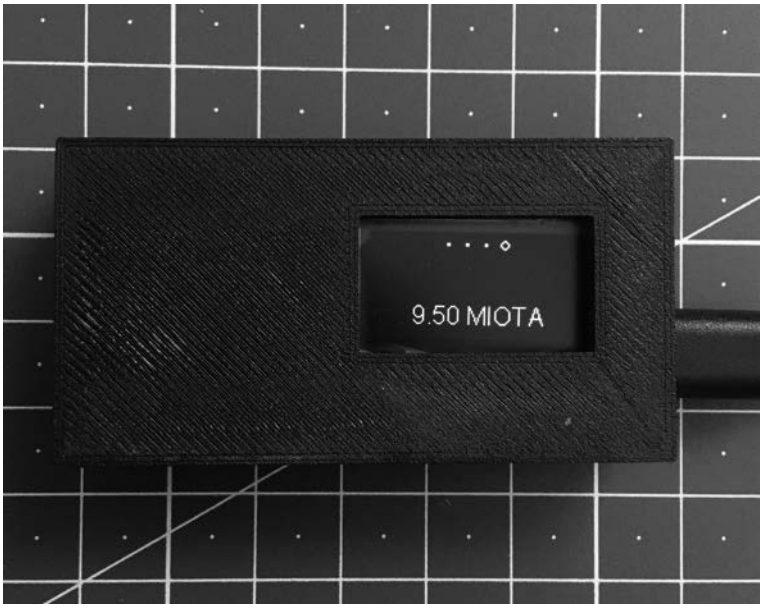

Bild 5.5 Die IOTA-Spardose zeigt den Kontostand an.

Der Aufbau des Spiels ist relativ identisch zum Spiel „Don't move“: Zuerst musst du ein paar Werte in der Funktion `beim Start` initialisieren (Bild 5.24). Neben den bekannten Variablen `Punkte`, `go` und `SpielStartZeit` erstellst du in der Variablen `Spieler` einen sogenannten Sprite – unsere Spielfigur auf dem virtuellen Spielfeld. Diesen setzt du an Position 2/2 auf die LED-Matrix.

> Die LED-Matrix zählt von links unten bei 0/0 los. Links oben liegt 5/0 an, rechts oben 5/5, rechts unten 0/5. 2/2 ist somit die Mitte des Spielfeldes.

Um den Neigungssensor zu kalibrieren bzw. warmlaufen zulassen, liest du schon einmal die beiden Gradwerte aus.

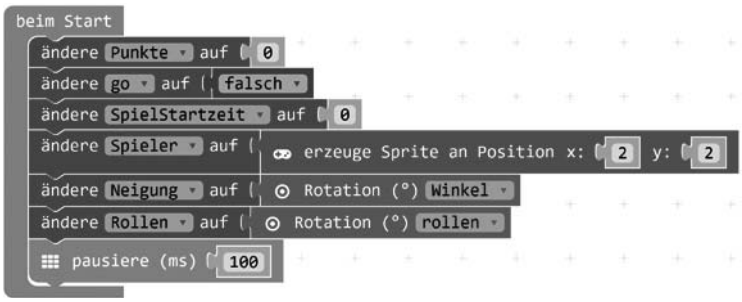

Bild 5.24 „Keep your balance“: Auch hier müssen die passenden Variablen gesetzt werden. Mit einem Sprite erzeugst du eine Spielfigur.

Den Countdown und Spielstart durch Drücken auf A+B kannst du direkt aus dem Spiel „Don't move“ übernehmen (Bild 5.17) und kommst damit schon zum Herzstück von „Keep your balance“: der `dauerhaft`-Schleife (Bild 5.27). Hier liest du zuerst den `Neigungs`winkel und den Rollwinkel (`Rollen`) aus und überprüfst dann zuerst den aktuellen Neigungswert. Die passende Zuordnung von Winkel zu Feld lässt sich am besten in Bild 5.25 nachvollziehen. Ist der Wert größer als 10, setzt du Feldposition 4, ist er zwischen 5 und 10, setzt du Position 3. Ist er kleiner als −10, ergibt sich Feldposition 0, liegt er zwischen −10 und −5, setzt du Position 1. In allen anderen Fällen setzt du ihn in die Mitte auf Position 2.

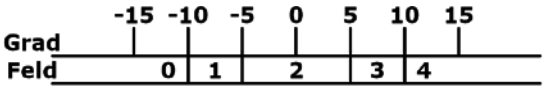

Bild 5.25 Neigungswinkel und Feldposition

Für den Rollwinkel ist die Situation ähnlich (Bild 5.26). Hier musst du jedoch unterscheiden, ob der Wert negativ (links) oder positiv (rechts) ist. Ist er negativ und größer als −170, ergibt sich die Position 0, liegt er zwischen −170 und −175, die Position 1. Ist er positiv und der Wert kleiner als 170, ergibt sich Position 4, liegt er zwischen 175 und 170, die Position 3, ansonsten die Position 2.

Sobald der Bot erfolgreich erstellt wurde, erhältst du einen sogenannten API Access Token, eine Art zufälliges Passwort, den du später für die Implementierung deines Bots benötigst.

Wenn du möchtest, kannst du über weitere Befehle wie `/setdescription` oder `/setuserpic` deinen Bot noch etwas für die Öffentlichkeit individualisieren. Um den Umgang mit dem Bot zu vereinfachen, kannst du mit dem Kommando `/setcommands` die unterstützten Befehle setzen und deinem Bot mit `gotmilk - Wie viel Milch ist noch im Kühlschrank?` den `/gotmilk`-Befehl hinzufügen. Dieser Schritt ist zwar nicht unbedingt notwendig, ist aber einfach gute Praxis und macht den Umgang mit dem Bot etwas einfacher.

Der komplette Registrierungsprozess ist in Bild 5.61 dargestellt. Damit ist die Registrierung deines Bots erst einmal abgeschlossen. Jetzt musst du ihn in den nächsten Schritten noch mit Leben füllen, indem du ihn auf deinem ESP implementierst.

Mehr Details zu Bots findest du unter *https://core.telegram.org/bots*.

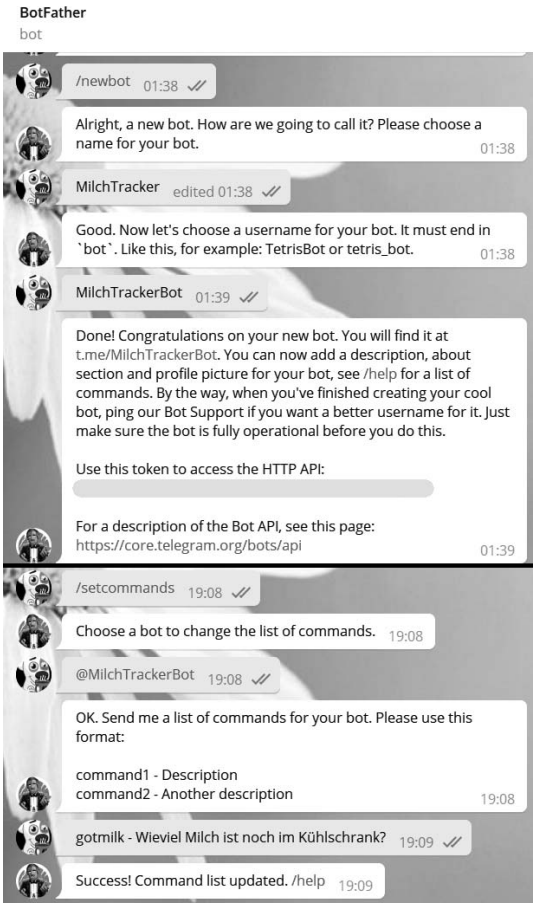

Bild 5.61
BotFather macht uns ein Angebot, das wir nicht ablehnen können: Er registriert unseren Bot kostenlos für uns.

6.2.2 Einrichtung des MQTT-Sensors

Um Home Assistant etwas Leben einzuhauchen, fügst du nun den MQTT-Sensor der Konfiguration hinzu. Hierzu öffnest du wieder die Konfigurationsdatei `/home/homeassistant/.homeassistant/configuration.yaml` und fügst dort eine neue „Section“ mit dem Namen `mqtt` ein (Listing 6.6).

Listing 6.6 `mqtt`-Section in `configuration.yaml`

```
mqtt:
  # broker: IP oder Hostname des Brokers
  broker: iotathome
  port: 1883
  username: iotathome
  # password in secrets.yaml als mqtt_passwort
  password: !secret mqtt_password
```

Unter `broker` gibst du die IP bzw. den Hostnamen des MQTT Brokers ein. Port, Username und Passwort sind optional. Eine kleine Besonderheit gibt es beim `password`. Um es nicht direkt in die für jedermann lesbare `configuration.yaml` zu speichern, gibst du mit `!secret` den Namen einer Variablen (hier `mqtt_password`) an, die in der `secrets.yaml` definiert ist.

Diese sieht dann aus, wie in Listing 6.7 dargestellt.

Listing 6.7 Geheimes Passwort in `/home/homeassistant/.homeassistant/secrets.yaml`

```
mqtt_password: geheimesPasswort
```

In der Standardinstallation von Home Assistant ist die `secrets.yaml` leider für jedermann lesbar. Damit kann man das Passwort auch direkt in die `configuration.yaml` speichern. Besser ist es jedoch, die Leserechte für alle außer dem User `homeassistant` zu entfernen:

```
sudo chmod go-r /home/homeassistant/.homeassistant/secrets.yaml
```

Nachdem der Broker konfiguriert ist, trägst du nun deine beiden Sensoren in die `configuration.yaml` ein. Dort suchst du die Section `sensors` und fügst beide nach dem bestehenden Sensor `yr`[12] ein. Komplett sieht dieser Abschnitt dann wie in Listing 6.8 aus.

[12] Der Name *yr* kommt von der verwendeten Webseite *http://yr.no,* ein Dienst vom norwegischen Wetterdienst. Auch diesen „Sensor“ kann man konfigurieren: *https://www.home-assistant.io/components/sensor.yr*

Listing 6.8 MQTT-Sensor-Konfiguration in `configuration.yaml`

```
sensor:
  - platform: yr
  # mqtt als Plattform für unseren Sensor
  - platform: mqtt
    # Frei wählbarer Name, der in der Weboberfläche angezeigt wird
    name: "Temperatur"
    # MQTT Topic, so wie wir es in unserem ESP konfiguriert haben.
    state_topic: "sensors/zimmer_1/temp"
    # Einheit – hier Grad Celsius
    unit_of_measurement: '°C'
    # Damit keine Lücken bei der Visualisierung entstehen, setzen wir hier true
    force_update: true
    # Es handelt sich um einen Temperatursensor.
    # Damit wird ein Thermometer als Icon angezeigt
    # Alle Devices mit derselben Klasse werden in einem Graphen zusammengefasst
    device_class: temperature

  - platform: mqtt
    name: "Feuchtigkeit"
    state_topic: "sensors/zimmer_1/hum"
    unit_of_measurement: '%'
    force_update: true
    device_class: humidity
```

Wie man sieht, ist die Konfiguration sehr einfach. Wer mehr über MQTT-Sensoren erfahren will, findet in der Dokumentation (*https://www.home-assistant.io/components/sensor.mqtt*) alle Details.

Rufst du nun die Weboberfläche auf, sollten eigentlich deine beiden neuen Sensoren auftauchen, da Home Assistent in der Regel bemerkt, dass die Konfigurationsdatei verändert wurde. In der Praxis funktioniert dies leider nicht immer, auch ein Neuladen der Konfiguration über die Weboberfläche (Einstellungen > Allgemein > Hauptsystem Neu Laden) löst das Problem nicht zuverlässig. Im Zweifelsfall startest du Home Assistant einfach neu, in diesem Fall wird die neue Konfiguration definitiv übernommen.

Über Einstellungen > Allgemein > Konfiguration Prüfen kannst du vor einem Neustart überprüfen, ob deine Konfiguration Fehler enthält, die einen Neustart verhindern würden.

Sobald alles geklappt hat, erscheinen sowohl auf der Seite *Übersicht* der Weboberfläche (Dashboard, Bild 6.4) als auch im *Verlauf* deine beiden neuen Sensoren. In unserem Beispiel wurde der YR-Sensor noch so konfiguriert, dass auch die Außentemperatur und Außenfeuchtigkeit angezeigt werden (Bild 6.5).

Bild 6.4 Home-Assistant-Übersichtsseite (Dashboard): Oben sind die aktuellen Sensorwerte dargestellt.

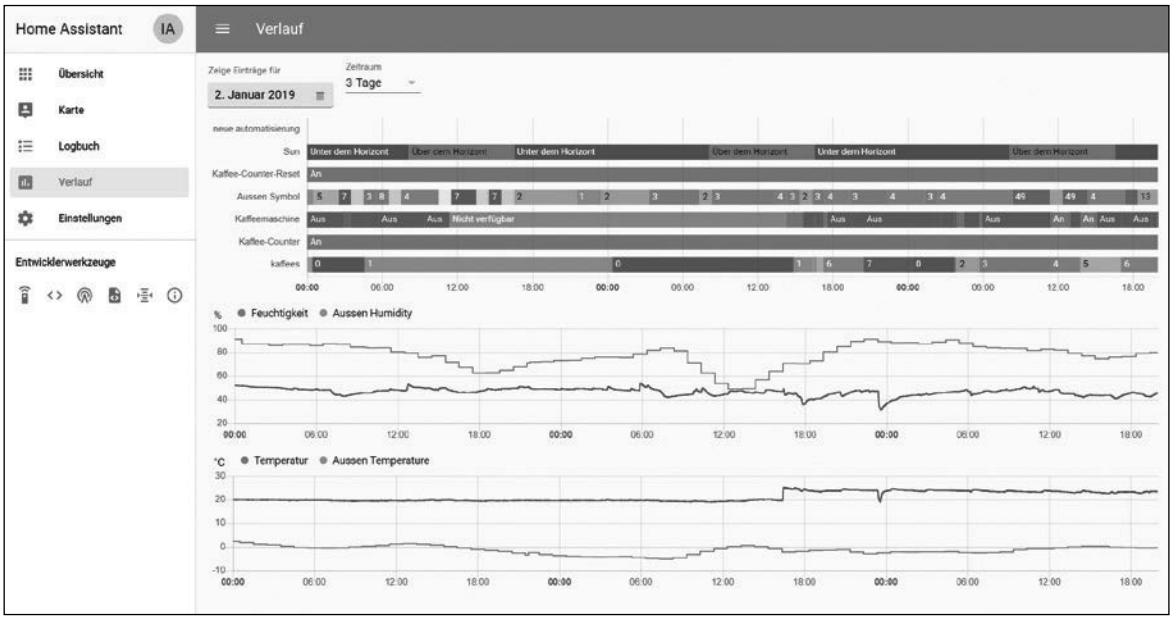

Bild 6.5 Unter *Verlauf* findet sich die grafisch aufbereitete Historie der Werte und Zustände.

Es ist auch möglich, die Broker-Komponente so zu konfigurieren, dass neue MQTT-Devices wie Sensoren automatisch entdeckt und hinzugefügt werden. Hierzu müssen die MQTT-Topics jedoch bestimmten Regeln entsprechen. Die Details hierzu findest du unter *https://www.home-assistant.io/docs/mqtt/discovery*. Dieses Vorgehen lohnt sich vor allem, wenn man sehr viele (gleichartige) Sensoren einsetzen möchte.