E. F. Engelhardt

SMART HOME MANIFEST

Hausautomation und Heimvernetzung für Maker

800 SEITEN

ARDUINO™, ESP8266 UND RASPBERRY PI

50+ PROJEKTE

FRANZIS

E. F. Engelhardt

SMART HOME MANIFEST

Hausautomation und Heimvernetzung für Maker

Über den Autor:
E. F. Engelhardt, Jahrgang 1975, hat bereits über 40 Computerbücher veröffentlicht – und keines dieser Bücher ist wie andere Computerbücher: Der Autor beginnt direkt mit Praxis, ohne langatmige Technikerläuterungen. Engelhardt zeichnet sich dadurch aus, dass er alle seine Projekte selbst entwickelt.

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.

Alle Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, dass sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung etwaiger Fehler sind Verlag und Autor jederzeit dankbar. Internetadressen oder Versionsnummern stellen den bei Redaktionsschluss verfügbaren Informationsstand dar. Verlag und Autor übernehmen keinerlei Verantwortung oder Haftung für Veränderungen, die sich aus nicht von ihnen zu vertretenden Umständen ergeben. Evtl. beigefügte oder zum Download angebotene Dateien und Informationen dienen ausschließlich der nicht gewerblichen Nutzung. Eine gewerbliche Nutzung ist nur mit Zustimmung des Lizenzinhabers möglich.

© 2018 Franzis Verlag GmbH, 85540 Haar bei München

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Das Erstellen und Verbreiten von Kopien auf Papier, auf Datenträgern oder im Internet, insbesondere als PDF, ist nur mit ausdrücklicher Genehmigung des Verlags gestattet und wird widrigenfalls strafrechtlich verfolgt.

Die meisten Produktbezeichnungen von Hard- und Software sowie Firmennamen und Firmenlogos, die in diesem Werk genannt werden, sind in der Regel gleichzeitig auch eingetragene Warenzeichen und sollten als solche betrachtet werden. Der Verlag folgt bei den Produktbezeichnungen im Wesentlichen den Schreibweisen der Hersteller.

Autor: E. F. Engelhardt
Programmleitung und Lektorat: Benjamin Hartlmaier
Satz: DTP-Satz A. Kugge, München
art & design: www.ideehoch2.de
Druck: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn
Printed in Germany

ISBN 978-3-645-60588-5

Vorwort

Vom Internet der Dinge oder neudeutsch Internet of Things (IoT) liest und hört man überall. Ist das wirklich etwas Neues oder wieder einmal »alter Wein in neuen Schläuchen«?

Bei IoT haben physische Dinge, z. B. ein Kühlschrank oder eine Waschmaschine, einen LAN- oder WLAN-Anschluss und damit die Möglichkeit, eine Verbindung ins Internet herzustellen. Mit IPv6 werden die IP-Adressen so schnell nicht ausgehen, und die prognostizierten 20 Milliarden Geräte im Jahr 2020 können auch alle ins Netz. Die Geräte können selbstständig miteinander interagieren. Das selbstständige Interagieren muss man natürlich programmieren oder ist vom jeweiligen Hersteller in einem Mikrocontroller vorprogrammiert. Möglich ist inzwischen sehr viel, das Einschalten der Lampen bei Dämmerung wird dabei zur Fingerübung. Meist wird das Smartphone als Fernbedienung und Informationszentrale für die vielen heimischen Helferlein genutzt – wenn Sie es verlieren, verlieren Sie nicht nur Ihre Daten, sondern auch den Schlüssel zu Ihrem Heim. Wie Sie sehen, sind Smart Home und IoT untrennbar miteinander verbunden. Aus diesem Grund wurde in diesem Buch das Smart Home als großes Anwendungsbeispiel für IoT gewählt.

Bei der Einrichtung eines Smart Home geht es darum, die Geräte der unterschiedlichen Hersteller unter einen Hut zu bekommen. Genau hier setzt dieses Buch an. Ob Alarm- oder Bewegungsmelder, Waschmaschine oder Fingerabdruckscanner, der Autor hat sehr viele Geräte in seinem Haus verbaut, vernetzt, programmiert und nebenbei die aufkommenden Schwierigkeiten gelöst und für Sie anschaulich dokumentiert. Wer keine fertigen Bausteine nutzen will, kann auf Arduino™, ESP8266 oder den Raspberry Pi setzen. Mit viel Quellcode lernen Sie, wie Sie selbst ein IoT-Smart-Home aufbauen und damit meist mehr Möglichkeiten haben als mit fertigen Geräten.

Technik-Liebhabern kann es nie genug vernetzte Technik im Haus sein. Aber wo viel Technik eingesetzt wird, entstehen auch Sicherheitslücken. Wollen Sie etwa die Webcam-Bilder aus dem Spielzimmer Ihrer Kinder im Internet sehen? Eher nicht. Deshalb ist Sicherheit auch ein großes Thema in diesem Buch. Je mehr Geräte ins Internet »telefonieren« können, umso mehr Informationen fließen nach draußen. Hier sollten Sie vorsorgen und Herr Ihrer Daten bleiben und nicht direkt alle Daten in die Cloud eines Herstellers legen. Nehmen Sie sich die Zeit für eine ordentliche Installation, dokumentieren Sie Ihre Planung, dann haben Sie auch länger Freude am Internet der Dinge.

Zurück zur Anfangsfrage: Alles neu ist es nicht, aber inzwischen funktioniert die Integration der Geräte in die heimische Gerätelandschaft sehr gut, lässt sich gut

bedienen und ist außerdem bezahlbar. Ob Sie lieber basteln oder ein fertiges Gerät kaufen, den Anschluss ans Internet schaffen Sie ohne große Hürden. In diesem Buch sind viele Projekte realisiert, die alle im Heim des Autors laufen. Auch wenn Sie keine Waschmaschine mit Smartphone-Anschluss wollen, die vorgestellten Lösungen bringen Sie sicher auf viele neue Ideen. Schreiben Sie uns doch, wie Ihr Internet der Dinge aussieht und wie smart Ihr Heim inzwischen ist.

Wir wünschen Ihnen viel Spaß mit dem Buch!

Autor und Verlag

Sie haben Anregungen, Fragen, Lob oder Kritik zu diesem Buch? Sie erreichen den

Autor per E-Mail unter *ef.engelhardt@gmx.de*.

Inhaltsverzeichnis

1 Smart-Home-Bausteine 17
1.1 LAN/WLAN-Router: der Datenverteiler 18
TCP/IP-Protokoll als gemeinsamer Nenner 18
Über die Vergabe der IP-Adressen 20
IP-Adressen im Internet übermitteln 20
Aus dem Internet ist nur der Router sichtbar 21
Dynamische DNS-Lösung für Internetzugriffe konfigurieren 22
Portfreigabe für den direkten Zugriff 27
1.2 MQTT und IoT 29
Installation von Mosquitto 30
Betrieb von Mosquitto 32
Mosquitto-Schnellcheck ohne MQTT-Client 33
Arduino als MQTT-Client 34
Pretzelboard als Temperatursensor 34
Arduino UNO als MQTT-Client 38
Raspberry Pi als MQTT-Client 43
Raspberry Pi als MQTT-Publisher 45

2 Mehr Sicherheit für Hausautomation und IoT 49
2.1 IoT-Gefahren durch versehentliche oder mutwillige Manipulation 49
Ruhiger schlafen – mechanische Redundanz ohne Schlüssel 50
2.2 Mehr Sicherheit für das Smart Home und IoT-Devices 51
Physical Web mit Bluetooth-Low-Energy- oder WLAN-IoT-Geräten 52
Sicher surfen – Internetgeräte und Webbrowser absichern 52
Webbrowserlücken finden 53
E-Mail – Phishing, Spam und ominöse Dateianhänge verhindern 55
2.3 IoT-Geräte im Smartphone-Umfeld 56
Eigene E-Mail-Adresse/Wegwerfadresse für IoT-Geräte 57
Facebook, Twitter & Co. – Vorsicht bei Social-Media-Archivdateien 57
2.4 Intranet der Dinge – Einsatz von Funknetzen in der Hausautomation 59
IoT-Geräte im Netz finden und schützen 60
MAC-Adresse finden mit arp 61
IP-Pfadfinder tracert 62
Mehr Sicherheit und Übersicht: eigenes WLAN-Netzwerk für IoT-Geräte 63

Schnellübersicht zur sicheren WLAN-Routerkonfiguration 64
2.5 Licht, Steckdosen oder Heizung steuern 68
Benutzerkonten vor unbefugten Zugriffen absichern 69
Basisschutz für den Raspberry Pi 69

3 Smart-Home-Zentrale im Eigenbau 71
3.1 Raspberry Pi: Standards und Anschlüsse 72
Durchblick im FS20- und HomeMatic-Protokoll 74
Angepasstes Funkmodul für den GPIO-Einsatz 75
USB-Adapter als Alternative für den Raspberry Pi 76
3.2 Kameramodul für Raspberry Pi 1, 2, 3 79
Kameramodul anschließen 79
Betriebssystem und Firmware auffrischen 80
Camera Module in Betrieb nehmen 81
Fotografieren mit Kommandozeilenbefehl 82
LED abschalten und heimlich fotografieren 83
Programmierung der Raspberry-Pi-Kamera 83
Infrarotfotografie mit dem Pi-NoIR-Modul 85
3.3 GPIO-Schnittstelle: Pinbelegung und Zugriff 86
Aufklärung über die GPIO-Pinbelegung 87
Direkter GPIO-Zugriff mit WiringPi 88
WiringPi-Bibliothek und Pinzuordnung 90
3.4 FHEM: die zentrale Anlaufstelle 92
FHEM-Startdatei für die COC-Erweiterung anpassen 94
Laufenden Apache-Prozess restarten 95
Anpassen der FHEM-Konfigurationsdatei 95
Initialer Start von FHEM 98
Mehr Sicherheit für FHEM: HTTPS aktivieren 100
FHEM mit Zugriffskennwort absichern 102
Funkkomponenten in Betrieb nehmen 103
3.5 ownCloud: Datenwolke ohne Limit 108
Raspberry Pi für ownCloud vorbereiten 109
ownCloud installieren und konfigurieren 115
Mehr Sicherheit: Benutzerkonto absichern 117

4 Alarm und Bewegungsmelder 119
4.1 Raspberry-Pi-SMS meldet Netzwerkausfall 120
Bluetooth und Gnokii in Betrieb nehmen 120
SMS über die Kommandozeile senden 125
Raspberry Pi mit SMS-Nachrichten steuern 128

4.2 Bewegungsmelder mit dem PIR-Modul ... 129
Shell-Skript für den Bewegungsmelder ... 132
PIR-Skript als Daemon im Dauereinsatz ... 133
WiringPi-API mit Python bekannt machen ... 135
4.3 Briefkastenalarm mit Benachrichtigung ... 136
Reed-Schalter und Sensoren im Einsatz ... 137
Shell-Skript für den Schaltereinsatz ... 138
4.4 Paparazzi Pi zeigt Neues aus dem Vogelhaus ... 140
Funktionsweise der USB-Webcam prüfen ... 141
Piri-Skript als Vorlage nutzen und aufbohren ... 142
Ohne Strom nix los: Akkupack auswählen ... 144
Vogelhausmontage: kleben und knipsen ... 145
4.5 Smarte Rauchmelder – Smartphone als Feuerwehr ... 146
Nest-Rauchmelder einrichten per App ... 147
Rauchmelderüberwachung mit dem Smartphone ... 147
4.6 Rauch- und CO-Alarmbenachrichtigung mit PHP ... 150
Zugriff auf den Rauchmelder mit der Google-Nest-API ... 151
PHP-Curl nachrüsten ... 151
Google-Cloud-Rauchmelderdaten mit PHP auslesen ... 152
Rauchmelder – Alarm mit E-Mail-Benachrichtigung ... 154
Regelmäßige Überprüfung per Cronjob ... 158
4.7 Low-Cost-Bewegungsmelder im Eigenbau ... 161
Arduino Nano mit ESP8266-WLAN-Modul ... 163
Pretzelboard: Arduino Nano und ESP8266 ... 164
Bewegungsmelder über WLAN – Arduino-Sketch im Detail ... 168
Raspberry Pi als Webserverschnittstelle ... 174
Alarmfunktionen und E-Mail-Benachrichtigung ... 180

5 Arduino und Raspberry Pi als Türwächter ... 185
5.1 An- und Abwesenheitserkennung mit dem Smartphone ... 186
Bluetooth und WLAN – Erkennung mit Tücken ... 186
5.2 Wo bin ich? – Geofencing in der Hausautomation ... 188
Geofency/Geofancy im Einsatz ... 188
5.3 Türklingelbenachrichtigung mit Foto ... 193
FS20-KSE-Funkmodul in die Türklingel einbauen ... 193
Die Funkmodulkonfiguration ist schnell erledigt ... 194
Neuer E-Mail-Account nur für die Klingel ... 196
fswebcam: Shell-Fotografie mit der Klingel ... 199
Skript für den E-Mail-Versand über FHEM ... 200
FHEM und Raspberry Pi verheiraten ... 201

6 DoorBird in der Hausautomation 203
6.1 Klingelpreller ade! – Einbau, Montage und Anschluss 205
DoorBird-App besorgen und installieren 207
Grundkonfiguration per DoorBird-App 209
App und Klingel über WLAN-Verbindung koppeln 212
LAN/WLAN & LTE – DoorBird im Praxiseinsatz 217
6.2 HTTP-Zugriff auf die DoorBird-Funktionen 218
Livestream der DoorBird-Türstation 221
Bildarchiv des DoorBird anzapfen 222
Türöffner- und Lichtschalterfunktion mit Relais 223
Automatische Benachrichtigung einschalten 224
6.3 DoorBird-Modding mit dem Raspberry Pi 224
Netzwerk für den Dauerbetrieb einrichten 225
Raspberry Pi: Webserver einrichten 227
Klingelbenachrichtigung per E-Mail-Nachricht 232

7 Bild-/Videoüberwachung zu Hause 243
7.1 Netatmo – Welcome to the Cam-Jungle 244
7.2 Willkommen – Inbetriebnahme mit dem Smartphone 245
7.3 Wenig Speicherplatz? – Speicherkarte tauschen 250
7.4 Willkommen zu Hause – Gesichtserkennung mit der Welcome-Kamera 252
7.5 Welcome Python – Indoor-Kamera mit dem Raspberry Pi auswerten 253
Python-Klasse für Welcome-Kamera 253
Methodenaufruf über welcome.py 257

8 Haustür öffnen ohne Schlüssel 259
8.1 Elektronisches Schloss: Lösungen für die Haus- oder Wohnungstür 260
Motorschloss im Türblatt 260
Elektronischer Schließzylinder 261
Alte und neue Welt verbinden 265
8.2 Smart Schreiner: Mehrfachverriegelung montieren 265
Mehrfachverriegelung und Motorschloss einbauen 266
8.3 Das Hirn der Haustür: Steuereinheit anschließen 271
Schlüssel immer dabei: Fingerscanner anschließen 276
Motorschloss mit Steuereinheit verbinden 279
Inbetriebnahme der ekey-Steuereinheit 280
8.4 Inbetriebnahme der Steuereinheit 284
Ersteinrichtung der ekey-Steuereinheit 284
Fingerscanner in Betrieb nehmen 285

Eingebautes Logging der Vorgänge nutzen ... 288
8.5 Gut informiert: Logging und Fingerscannerüberwachung mit dem UDP-Konverter ... 289
home-Protokoll: Aufbau des UDP-Datenpakets ... 292
8.6 Fingerscannerüberwachung mit dem Raspberry Pi ... 294
Portscanner mit nmap ... 294
Datentransport mit netcat ... 295
UDP-Datenpaket mit Python lesen ... 296
UDP-Datenpaket mit Python auswerten und speichern ... 297
8.7 Fingerscannerüberwachung mit dem Arduino ... 301
Ethernet-Shield in Betrieb nehmen ... 302
UDP-Pakete mit dem Arduino parsen ... 302
Logging und Auswertungen – Logdatei im Eigenbau ... 307
8.8 Raspberry Pi im Verteilerkasten ... 313
Grundinstallation und Konfiguration ... 314
Image auswählen und auf Micro-SD-Karte installieren ... 315
Spätere Inbetriebnahme: root oder pi? ... 317
Drahtlos kommunizieren via WLAN-Bluetooth-Dongle ... 318
Kein DHCP-Server: statischer Zugriff nötig ... 319
Gegen das Vergessen – WLAN-Adapter-Schlafmodus abschalten ... 321
Windows-Zugriff auf den Raspberry Pi mit Samba ... 322
8.9 Tür-Motorschlosssteuerung mit dem Raspberry Pi ... 327
Apache und PHP installieren und aktualisieren ... 328
PHP-Installation prüfen und Apache in Betrieb nehmen ... 328
Sicherer Zugriff: .htaccess und .htpasswd erzeugen ... 330
Türrelais über GPIO-Pin schalten ... 332
GPIO-Steuerung über PHP und JavaScript ... 335
Nicht lange suchen – Türschalter im Telefonbuch speichern ... 337
8.10 Mobilfunkanschluss für die Haustür ... 339
Neue Firmware: gefahrlos und schnell ... 339
8.11 GPRS/GSM-Modul am Raspberry Pi ... 346
Steuerung des Mobilfunkmodems mit Python ... 347
Mobilfunkmodem mit minicom-Konsole steuern ... 351
SMS-Empfang und Versand mit minicom-Konsole ... 354
AT-Kommandos mit Python einsetzen ... 362
SMS-Versand mit Python ... 363
8.12 Türöffnung per SMS am Raspberry Pi ... 368
Haustür öffnen per SMS ... 368
SMS-Empfang und Textanalyse ... 369
Automatische Türöffnung per SMS-Nachricht ... 373

8.13 Haus-/Wohnungstür mit dem Arduino steuern ... 379
Bauteile für die Arduino-Mobilfunksteuerung der Haustür ... 381
Arduino mit LCD-Bildschirm an I²C-Schnittstelle ... 382
SIM900-Mobilfunkmodem mit dem Arduino verbinden ... 384
Mobilfunklogik im Arduino-Sketch ... 386
SMS-Versand mit dem Arduino ... 389
SMS-Empfang mit dem Arduino ... 394
8.14 Netzwerk-Shield für UDP, Mail, NTP und Logging ... 398
Grundkonfiguration des Netzwerk-Shields ... 399
Datum und Uhrzeit auf LCD ausgeben ... 401
Micro-SD-Kartenanschluss des Ethernet-Shields nutzen ... 402
E-Mail-Benachrichtigung mit dem Arduino ... 405
Arduino als Gateway: SMS-Nachricht als E-Mail weiterleiten ... 410
Logdatei per E-Mail-Anhang versenden ... 412

9 Energiekosten fest im Griff ... 415
9.1 Unter Strom: Smart Home im Eigenbau ... 415
Drehstromzähler einbauen und anschließen ... 418
1-Wire-Geräte am Raspberry Pi anschließen ... 419
1-Wire-Bus und 1-Wire-USB-Connector prüfen ... 421
OWFS kompilieren und installieren ... 424
Zählermodul am Raspberry Pi in Betrieb nehmen ... 431
FHEM-Konfiguration für den Stromzähler ... 433
9.2 Kampf der Stand-by-Verschwendung ... 436
Vorteile von steuerbaren Steckdosen ... 436
Markenprodukt oder China-Ware? ... 437
9.3 IP-Steckdosen made in Germany ... 438
Für Profis: Rutenbeck TCR IP 4 ... 439
TCR IP 4 in Betrieb nehmen ... 439
Mit Trick 17 durch die HTTP-Hintertür ... 442
Rutenbeck-Steckdose per Shell-Skript steuern ... 443
Hacking Rutenbeck: Schalten via HTTP-Adresse ... 444
9.4 Billigsteckdosen mit dem Pi koppeln ... 446
Taugliche Funksteckdosen mit Fernbedienung ... 447
Funksteckdosen via GPIO mit Raspberry Pi koppeln ... 448
China-Chip: Schaltung entschlüsselt ... 448
Mit dem Lötkolben ran an den IC-Baustein ... 451
Steckdosen schalten mit der Shell ... 455
Steckdosen schalten mit Python ... 457
Billigsteckdosen und FHEM koppeln ... 458

DIP-Schalter-Codierung entschlüsselt ... 458
DIP-Schalter und FHEM verknüpfen ... 459
9.5 Praktische Gimmicks der TC-IP-1-Dosen ... 461
Waschmaschine und Trockner überwachen ... 463
9.6 Energiemonitor mit JeeLink-Arduino ... 464
Raspberry Pi für Arduino-IDE vorbereiten ... 466
JeeLink-Adapter über Arduino-IDE flashen ... 466
Arduino-JeeLink-Adapter und FHEM updaten ... 473

10 Temperatur, Klima und Heizung ... 479
10.1 Heizkosten senken mit dem Raspberry Pi ... 479
Temperaturmessung Marke Eigenbau ... 480
Temperatursensor in Betrieb nehmen ... 483
Funktionsprüfung des Temperatursensors ... 484
Kernelmodule automatisch laden ... 486
Heizungsverbrauch messen und dokumentieren ... 487
10.2 Heizkörperthermostate kontra Schimmelbefall ... 490
Neue Funkheizkörpermodule montieren ... 492
Steuereinheit mit den Thermostaten verheiraten ... 494
Kopplung mit Fenstern und Türen ... 495
Heizungsreglereinheit mit Raspberry Pi koppeln ... 495
Temperatursteuerung in Haus und Wohnung ... 497
10.3 Körperwaage als Thermometer für die Heizung ... 499
App für Smart Body Analyzer installieren ... 502
Withings Smart Body Analyzer steuert die Heizung ... 508
10.4 Pi als elektronischer Wetterfrosch ... 510
Wetterstationen für den Pi-Teameinsatz ... 511
Inbetriebnahme einer USB-Wetterstation ... 512
Logging-Intervall einrichten ... 515
Datentransport zum Raspberry Pi ... 515
Rohdaten ins pywws-Format konvertieren ... 517
Template für die aktuelle Wettervorhersage ... 520
10.5 Sonne, Regen, Wind – Netatmo-Wetterstation ... 526
Netatmo-Wetterstation am Computer vorbereiten ... 527
iPhone- und Android-App im Einsatz ... 534
Heizungssteuerung, Lüftung & Co.: Python-Schnittstelle im Eigenbau ... 536
Python-API besorgen und einrichten ... 540
Netatmo-Wetterstation im Zusammenspiel mit FHEM ... 543
10.6 Grüner Daumen mit dem Smartphone ... 544
Einstecken, installieren, gießen: Flower-Power im Smart Home ... 544

Pflanzenüberwachung mit dem Raspberry Pi ... 548
JSON- und Requests-Bibliothek nachrüsten ... 549
Mehr Sicherheit mit SSL ... 550

11 LED-Lampen und Lichteffekte ... 555
11.1 LED-Lichtspielhaus: Hue-Bridge und -Lampen ... 555
Philips Hue: Lampensteuerung mit Apple ... 556
Unterschiedliche Hue-Lampen zusammenschalten ... 558
Hue-Lampen und iPhone: Zwangshochzeit per App ... 561
11.2 Made in Dresden: lange Leitung für Hue-Stripes ... 566
Abisolieren und stecken: LED-Streifen anschließen ... 567
Schalteinheit FLS PP mit Hue-System koppeln ... 568
11.3 Hue-Lampen ohne App steuern ... 570
Schalter mit Hue-System verbinden ... 570
Hue-Steuerung mit Python selbst gebaut ... 572
11.4 FHEM mit Hue-Lampen nachrüsten ... 582
Perl CPAN auf dem Raspberry Pi installieren ... 584
Hue-Lampen in FHEM einrichten ... 587
11.5 Hue-Alternative: WLAN-Lampen aus China ... 590
WLAN-Lampen und Wi-Fi-Bridge einrichten ... 592
Wi-Fi-Lampen mit dem Smartphone steuern ... 594
Wi-Fi-Lampen mit dem Raspberry Pi steuern ... 597
11.6 Playbulb – Licht und Gesang aus der Lampenfassung ... 604
Farbe oder nicht – eine Frage des Geldbeutels ... 605
Bluetooth-Kopplung auf Umwegen ... 606
11.7 Lampensteuerung und Lichteffekte auf Knopfdruck ... 608
Bauteileliste für das WLAN-Schalterprojekt ... 608
Schaltung und GPIO-Pins verheiraten ... 614
Hürden und Stolperfallen bei der Inbetriebnahme ... 628

12 Unterhaltungs- und Haushaltselektronik steuern ... 631
12.1 iOS und HomeKit – Siri macht Strom ... 632
Was bin ich? – Siri gibt Antwort ... 632
Steckdosen mit Siri schalten ... 634
iPhone: Kontakt für Gerät erstellen und konfigurieren ... 635
HomeKit – Strippenzieher im Hintergrund ... 637
12.2 Steckdosen mit Python über UDP steuern ... 649
UDP-Steuerung mit Python ... 656
Energiemessung und mehr: TC IP 1 WLAN und FHEM ... 657
12.3 Weinkühlschrank mit dem Raspberry Pi ... 659

I²C-Protokoll – neue Spielregeln ... 661
SHT21: nötige Vorbereitungen treffen ... 666
I²C-Bus: Schnittstelle wecken und checken ... 669
Feuchte- und Temperaturmessung für optimale Lagerung ... 670
Temperatur- und Feuchtigkeitsalarm per SMS ... 672
12.4 Garage und Türen mit dem Smartphone öffnen ... 673
Handy, Tablet & Co.: Bluetooth als Aktor ... 674
To be or not to be Admin: root-Werkzeuge für Benutzer ... 677
Shell-Skript für die Bluetooth-Erkennung erstellen ... 679
12.5 Computer und NAS-Festplatten steuern ... 681
Sicheres Log-in ohne Passwort: SSH-Keys im Einsatz ... 681
NAS-Server: Netzwerkfestplatten konfigurieren ... 682
Raspberry Pi per Windows-Desktopverknüpfung schalten ... 684
Manchmal knifflig: SSH-Parameter finden ... 685
Windows-Computer per Shell-Kommando schalten ... 686
Shutdown-Skript erstellen ... 688
Shell-Skript und FHEM verbinden ... 690
12.6 Scanner und Drucker ganz ausschalten ... 692
Drucker vorbereiten: CUPS installieren ... 692
CUPS-Backend anpassen ... 693
Skript zum Schalten der Steckdose ... 695
FHEM-Konfiguration der FS20-Druckersteckdose ... 697
12.7 Funken und steuern – Smartphone als Fernbedienung ... 700
VU+ DUO² – die TV-Box für Tüftler ... 702
FRITZ!Box-Festnetz mit Kabel-/SAT-TV-Box koppeln ... 712
Eine für alle: Logitech Harmony im Wohnzimmer ... 718
12.8 Hausgeräte im Smart Home mit Miele@mobile ... 748
Miele-Gerät mit Kommunikationsmodul nachrüsten ... 749
Miele-Geräte mit dem Heimnetzwerk verbinden ... 749
Grundinstallation des Miele-Gateways ... 751
Waschmaschine mit Miele-Gateway koppeln ... 754
Android oder Apple iOS – Miele@mobile-App im Einsatz ... 756
Waschmaschinen-TV im Wohnzimmer ... 761
12.9 Einfache Haushaltsgeräteüberwachung mit dem Arduino ... 770
Gyrosensoren – Begriffe und Unterschiede ... 770
Inbetriebnahme des MPU-6050 ... 772
Experimente mit dem MPU-6050-Gyrosensor ... 773
ESP8266 und Gyroskop im Zusammenspiel ... 775
PHP-Schnittstelle für Waschmaschinen-Logging und mehr ... 780

A Anhang 783
A.1 Konsolen-Basics: Wichtige Befehle im Überblick 783
A.1.1 Zugriff auf Dateien und Verzeichnisse regeln 784
A.2 Basisausstattung für den Python-Einsatz 785
A.2.1 Python-Bibliothek RPi.GPIO für GPIO-Zugriff installieren 786
C Abkürzungen 788

Stichwortverzeichnis 789

1

Smart-Home-Bausteine

Die Schnittstelle zwischen dem Internet und dem Heimnetz ist das heimische Internetzugangsgerät, in der Regel der LAN/WLAN-Router – eine wichtige Tatsache, vor allem wenn Sie im Bereich Messen, Steuern, Regeln bei der Hausautomation von unterwegs mit dem Smartphone Ihre Gerätschaften zu Hause kontrollieren möchten. Hier stehen natürlich verschiedene Lösungsansätze zur Auswahl: entweder direkt das betreffende Gerät mit seiner eigenen App steuern und kontrollieren oder eine Mastermind-Lösung anstreben – eine Hausautomation-Kommandozentrale, die sich um die Vorortgerätschaften kümmert und über eine gemeinsame App die jeweiligen Schnittstellen zur Verfügung stellt. Dafür ist dann ein weiteres Gerät nötig, das in der Regel neben dem LAN/WLAN-Router dauerhaft online und damit an sieben Tagen 24 Stunden im Einsatz ist. Heute ist das Thema Netzwerkeinrichtung zu Hause eigentlich keine große Sache mehr – knifflig wird es erst, wenn unterschiedliche Computer vernetzt und mit gewöhnlichen Haushaltsgeräten und verschiedenen Technologien gekoppelt werden sollen. In dem Fall muss man selbst ein wenig Hand anlegen, damit es klappt. Anschließend können Sie mit dem Smartphone über eine App die gekoppelten Geräte – etwa Heizung, Lichtschalter, Waschmaschine, Rauchmelder, Klingelanlage und was noch alles in einem Haushalt an Gerätschaften benötigt wird – bequem steuern und kontrollieren.

1.1 LAN/WLAN-Router: der Datenverteiler

Um die Verteilung der Daten in Ihrem Heimnetzwerk kümmert sich in der Regel ein Router, der den Datenverkehr gezielt steuert und die Netzbelastung in Grenzen hält. Der Router wickelt sozusagen alle Aufträge ab, die von den Clients an ein anderes Netz geschickt werden. Ob es sich beim adressierten Netz um ein weiteres Unternehmensnetz handelt oder um das Internet, spielt keine Rolle.

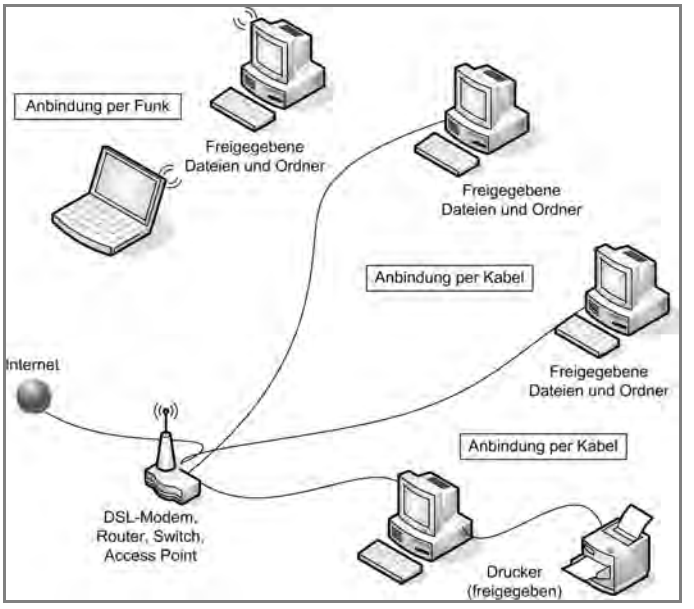

Bild 1.1: Beispiel eines Netzwerks, bestehend aus Kabel- und WLAN-Verbindungen mit Datei- und Druckerfreigaben.

Wie auch immer in Ihrem Netzwerk Daten übertragen werden und welches Betriebssystem Sie auch einsetzen – an TCP/IP, der Internetprotokollfamilie, kommen Sie nicht vorbei. Jetzt brauchen Sie sich aber nicht mit so diffizilen Dingen wie Protokollschichten, Headern oder dergleichen herumzuschlagen, für Sie genügen die Basics der Adressierung. Außerdem müssen Sie wissen, dass TCP/IP festlegt, wie Daten im Internet und im Netzwerk übermittelt werden. Bei einer Netzwerkverbindung oder einer Internetverbindung wird keine direkte Verbindung zwischen zwei Punkten hergestellt, wie das beispielsweise beim Telefonieren der Fall ist.

TCP/IP-Protokoll als gemeinsamer Nenner

Die Daten werden vielmehr in kleine Pakete zerlegt und auf den Weg zum Ziel geschickt. Wo sie hinmüssen, steht in der Adresse. Am Ziel werden die Pakete wieder in der richtigen Reihenfolge zusammengesetzt. Auch das wird über TCP/IP gesteuert,

denn Reihenfolge und Anzahl der Pakete werden ebenfalls übermittelt. Dazu kommen noch ein paar Prüfgeschichten und sonstige Informationen – das muss Sie aber nicht interessieren. Damit ein Rechner über TCP/IP angesprochen werden kann, muss seine Adresse, die sogenannte IP-Adresse, bekannt sein. Die Adressierung ist bei TCP/IP in ihrer Struktur festgelegt. Auf der Basis von Version IPv4 können bis zu 4.294.967.296 Geräte in ein Netzwerk integriert werden. IPv4 nutzt 32-Bit-Adressen, die Weiterentwicklung IPv6 hingegen setzt auf 128-Bit-Adressen.

Eine TCP/IP-Adresse ist immer gleich aufgebaut: Sie setzt sich zusammen aus einem Netzwerkteil und einem Hostteil (Adressenteil). In der Regel ist die 32-Bit-Adresse in einen 24-Bit-Netzwerkteil und einen 8-Bit-Hostteil aufgeteilt. Der Hostteil wird im LAN (im lokalen Netzwerk) zugeteilt, während der Netzwerkteil von der IANA (*Internet Assigned Numbers Authority*) vergeben wird, die über die offiziellen IP-Adressen wacht.

Für die Konfiguration des Hostteils sind in einem sogenannten Class-C-Netzwerk – das ist ein typisches privates Netz – 254 Geräteadressen für angeschlossene Clients verfügbar. Die Endadresse `255` ist für den Broadcast (zu Deutsch: Rundruf, also Übertragung an alle) reserviert, während die Adresse `0` für das Netzwerk selbst reserviert ist.

Für die Aufteilung des Netzwerk- und Hostteils ist die Netzmaske zuständig: Im Fall eines Class-C-Netzwerks gibt die Adresse `255.255.255.0` eine sogenannte Trennlinie zwischen beiden Teilen an. Die binäre `1` steht für den Netzwerkteil, und die `0` steht für den Adressteil.

So entspricht die Netzwerkmaske

```
255.255.255.0
```

binär:

```
11111111.11111111.11111111.0000000
```

Die ersten 24 Bit (die Einsen) sind der Netzwerkanteil.

Sie müssen sich aber gar nicht mit der Adressvergabe herumschlagen, denn der heimische Rechner ist immer mit den nachfolgenden Daten ansprechbar. Einige Klassen von Netzwerkadressen sind für spezielle Zwecke reserviert. Man kann an ihnen ablesen, mit welchem Netzwerk man es zu tun hat. Beispielsweise ist eine IP-Adresse beginnend mit `192.X.X.X` oder `10.X.X.X` ein internes Netzwerk, in Ihrem Fall ein Heimnetzwerk.

Adressbereich	Netzwerk
192.168.0.0	Heimnetz, bis zu 254 Clients
172.16.0.0	Unternehmensnetz, bis zu 65.000 Clients
10.0.0.0	Unternehmensnetz, bis zu 16 Mio. Clients

Sobald aus einem heimischen Rechner ein Netz aus mehreren Computern wird, beginnt die IP-Adresse mit `192.168.0`. Auf dieser Basis können in das Netz bis zu 254 Geräte eingebunden werden, indem die letzte Zahl von 0 bis 254 hochgezählt wird. Allerdings hat kaum jemand zu Hause so viele Geräte im Einsatz, es wird bei überschaubaren Adressbereichen bleiben.

Über die Vergabe der IP-Adressen

Gewöhnen Sie sich für die Vergabe der IP-Adressen entweder die automatische Zuweisung via DHCP oder eine statische Zuweisung mit festen Adressen an. Wenn Sie mit festen Adressen arbeiten, sollten Sie gegebenenfalls nur ausgewählte, leicht merkbare IP-Adressen verwenden, also `192.168.0.1` für den Router, `192.168.10` für den zentralen Rechner und für weitere die Endnummern `20`, `30` etc. Wer generell Schwierigkeiten hat, sich die Nummern zu merken, kann die Computer beispielsweise nach Alter nummerieren – in der Regel weiß man genau, welchen Computer man zuerst gekauft hat.

Der Vollständigkeit halber sei hier auch das sogenannte Gateway erwähnt. Innerhalb des Heimnetzwerks können sämtliche Geräte direkt miteinander kommunizieren und Daten austauschen. Soll hingegen eine Verbindung zu einem Gerät aufgebaut werden, das sich nicht innerhalb des adressierbaren Adressbereichs befindet, müssen die Heimnetze miteinander verbunden werden. Diese Aufgabe übernimmt das Gateway bzw. der Router, der quasi sämtliche verfügbaren Netzwerke kennt und die Pakete bzw. Anforderungen entsprechend weiterleitet und empfängt. Im Internet sind daher einige Router in Betrieb, da es technisch nahezu unmöglich ist, dass ein einzelner Router alle verfügbaren Netze kennt und direkt adressiert.

In der Regel hat der LAN/WLAN-Router auch einen DHCP-Server eingebaut, der für die Vergabe der IP-Adressen im Heimnetz zuständig ist. Sind Daten für eine IP-Adresse außerhalb des Heimnetzes bestimmt, werden sie automatisch an das konfigurierte Standard-Gateway, also den Router, weitergeleitet. Verbindet sich der heimische LAN/WLAN-Router mit dem Internet, versteckt er das private Netz hinter der öffentlichen IP-Adresse, die der LAN/WLAN-Router beim Verbindungsaufbau vom Internetprovider erhalten hat. Dieser Mechanismus der Adressumsetzung, NAT (*Network Address Translation*) genannt, sorgt dafür, dass die Datenpakete vom Heimnetz in das Internet (und wieder zurück) gelangen.

IP-Adressen im Internet übermitteln

Alle Server im Internet sind ebenfalls über eine IP-Adresse ansprechbar, aber das könnte sich keiner merken. Wer weiß schon, dass sich hinter `217.64.171.171` `www.franzis.de` verbirgt? Deshalb gibt es im Internet zentrale Server, deren einzige Aufgabe darin besteht, für die von Ihnen eingegebene Internetadresse (URL) den richtigen Zahlencode bereitzustellen.

Nichts anderes passiert nämlich bei der Eingabe der URL: Der Rechner übermittelt seine Anfrage im Klartext an den sogenannten *Domain Name Server* (DNS). Ein DNS-Server führt eine Liste mit Domainnamen und den IP-Adressen, die jedem Namen zugeordnet sind.

Wenn ein Computer die IP-Adresse zu einem bestimmten Namen benötigt, sendet er eine Nachricht an den DNS-Server. Dieser sucht die IP-Adresse heraus und sendet sie an den PC zurück. Kann der DNS-Server die IP-Adresse lokal nicht ausfindig machen, fragt er einfach andere DNS-Server im Internet, bis die IP-Adresse gefunden ist. Damit die Daten, die Sie angefordert haben – und im Internet wird jede Seite aus übermittelten Daten aufgebaut –, auch wieder zu Ihnen bzw. zu Ihrem Rechner zurückgelangen, braucht der Server Ihre IP-Adresse. Nun wird nicht jedem Internetteilnehmer kurzerhand eine IP-Adresse verliehen – dafür gibt es einfach nicht genug Adressen. Stattdessen hat jeder Provider einen Pool mit IP-Adressen, die jeweils nach Bedarf vergeben werden.

Diese Technik ist quasi nichts anderes als die eines DHCP-Servers (*Dynamic Host Configuration Protocol*). Damit bekommen alle an ein Netzwerk angeschlossenen Computer, egal ob WLAN oder nicht, automatisch die TCP/IP-Konfiguration zugewiesen. Zusammen mit Ihrer Anfrage bei einer URL wird also Ihre eigene dynamische Adresse übermittelt, damit Sie auch eine Antwort bekommen.

Aus dem Internet ist nur der Router sichtbar

Wenn Sie Ihr Netzwerk mit einem Router für den Internetzugang ausstatten, übernimmt Ihr Router künftig einen Teil der Aufgaben rund um die Adressierung. Das macht Ihnen das Leben nicht nur etwas leichter, sondern vor allem viel sicherer, denn nach außen tritt lediglich der Router in Erscheinung, Ihren PC bekommt das Internet nicht so leicht zu sehen. Das beginnt schon damit, dass von außen nicht mehr die zugewiesene Adresse des Rechners zu sehen und zu verwenden ist, sondern die des Routers. Alle Anfragen stellt der Router, alle Antworten nimmt er entgegen und leitet sie netzwerktechnisch betrachtet als Switch innerhalb des heimischen Netzes an den passenden Rechner weiter.

Für den Router gibt es also intern den Nummernkreis `192.168.X.X` und nach außen alle anderen. Der einzelne Rechner ist nicht mehr direkt ansprechbar, sondern die Adresse ist immer die des Routers. Das ist ein erster Schritt in Richtung mehr Sicherheit im Internet, denn nun kann nicht mehr direkt auf möglicherweise offene Ports Ihres Rechners oder eines anderen im Netz zugegriffen werden. Noch mehr Sicherheit bietet eine im Router aktivierte Firewall, deren Ziel es ist, nur zulässige und ungefährliche Pakete durchzulassen und bestimmte Pakete kurzerhand abzulehnen. Sie nehmen ja an der Haustür auch nicht jede Nachnahme an.

Dynamische DNS-Lösung für Internetzugriffe konfigurieren

Möchten Sie Ihre Heimnetzsteuerzentrale auch über das Internet erreichen, etwa weil Sie vom Büro aus die Heiztemperatur im heimischen Wohnzimmer regeln möchten, benötigen Sie für Ihren LAN/WLAN-Router zu Hause eine dynamische DNS-Lösung. Mithilfe einer dynamischen IP-Adresse machen Sie den LAN/WLAN-Router im Internet bekannt. Jedes Mal, wenn Sie sich in das Internet einloggen, bekommt Ihr LAN/WLAN-Router automatisch vom Provider eine IP-Adresse zugeteilt. TCP und IP sind die wichtigsten Protokolle, die für die Kommunikation zwischen Rechnern möglich sind. Es gibt jedoch weitere Protokolle und Techniken wie beispielsweise SSH, mit denen Sie beim Lesen dieses Buchs in Berührung kommen. TCP/IP kommt in einem Netzwerk zum Einsatz, und jeder Computer, der in einem Netzwerk TCP/IP nutzen möchte, braucht eine IP-Adresse. Diese IP-Adresse lautet bei jeder Einwahl anders – sie stammt aus einem IP-Adressenpool, den der Provider reserviert hat.

DNS: Namen statt Zahlen

Der Vorteil von DNS ist, dass Sie den Computer auch über seinen Namen ansprechen können. Es ist einfacher, statt einer IP-Adresse wie `http://192.168.123.1` die Adresse `http://IHRDOMAINNAME.DNSSERVICEANBIETER.NET` einzutippen, denn Namen lassen sich leichter merken als Zahlen bzw. IP-Adressen.

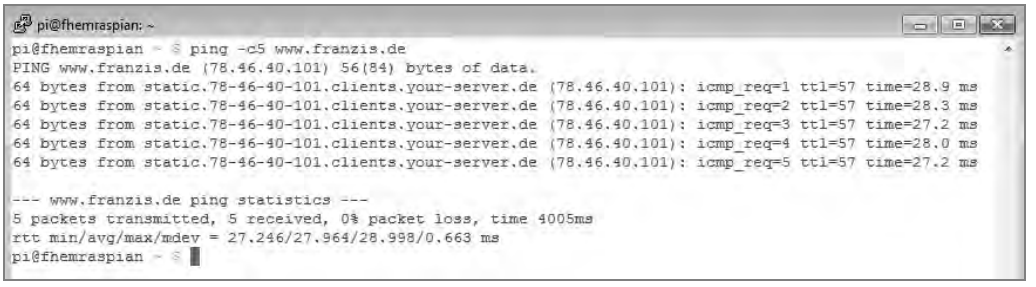

Bild 1.2: Mit dem Befehl `ping DNS-Name` finden Sie die IP-Adresse eines DNS-Namens heraus. In diesem Beispiel lautet die IP-Adresse für `www.franzis.de`: `78.46.40.101`.

Geben Sie beispielsweise *http://IHRDOMAINNAME.DNSSERVICEANBIETER.NET* in die Adressleiste des Webbrowsers ein, erkennt dieser am *http*-Kürzel, dass er das HTTP-Protokoll verwenden muss. Der doppelte Schrägstrich *//* bedeutet, dass es sich um eine absolute URL handelt. Mit der URL *IHRDOMAINNAME.DNSSERVICEANBIETER.NET* wird ein Kontakt zu dem DNS-Server Ihres ISP (**I**nternet **S**ervice **P**rovider) hergestellt. Damit wird dieser DNS-Name in eine IP-Adresse umgewandelt.

Für dynamisches DNS gibt es verschiedene Anbieter, die eine solche Funktionalität kostenlos zur Verfügung stellen. Die bekanntesten Anbieter sind in der folgenden

Tabelle aufgeführt. Die Vorgehensweise bei der Anmeldung ist im Prinzip immer die gleiche, für welche Sie sich entscheiden, bleibt Ihnen überlassen.

Anbieter (kostenlos)	
no-ip.com	www.no-ip.com
DNSExit	www.dnsexit.com/
AVM (nur für Fritzboxen mit FRITZ!OS ab Version 5.20)	https://www.myfritz.net/login.xhtml
FreeDNS	freedns.afraid.org
Securepoint spdyn	www.spdyn.de

Egal für welchen Anbieter Sie sich entscheiden, die Prozedur des Registrierens und Einrichtens sowie die Konfiguration des Clients bleiben Ihnen nicht erspart. Im nächsten Schritt richten Sie den DSL-WLAN-Router so ein, dass Sie aus dem Internet Zugriff auf den Raspberry Pi bekommen – am besten über einen Port, der nur Ihnen bekannt ist.

DynDNS Domain bei spdyn.de einrichten

Made in Germany – wer in Sachen dynamisches DNS zum Nulltarif einen deutschen Anbieter auswählt, macht nicht viel verkehrt. Generell ist das Vorgehen bei sämtlichen Anbietern dasselbe: Erst nach dem Registriervorgang und der Bestätigung des E-Mail-Kontos haben Sie die Möglichkeit, einen kostenlosen dynamischen DNS-Anschluss zu reservieren.

Nach der Registrierung bei dem Lüneburger Anbieter Securepoint (spdyn.de) fügen Sie dem angelegten Profil bei *Hosts* von Benutzer einen IPv4-Host hinzu und richten auf dem Raspberry Pi den spdns-Dynamic-DNS-Update-Client ein. Alternativ konfigurieren Sie den heimischen Router so um, dass er den Securepoint-DNS-Service kennt.

Bild 1.3: Zunächst tragen Sie den persönlichen Hostnamen ein und wählen im zweiten Schritt im Dropdown-Menü die gewünschte Domäne aus.

spdns Dynamic DNS mit der FRITZ!BOX

In diesem Abschnitt wird exemplarisch die weit verbreitete FRITZ!BOX verwendet, doch auch die Router-Konkurrenz im DSL/Kabel/Glasfasermarkt bietet die Einrichtung eines Anbieters für die dynamische DNS-Adresse ein. Bei der FRITZ!BOX finden Sie die DynDNS-Einstellungen im Menü *Erweiterte Einstellungen -> Internet -> Freigaben -> Dynamic DNS*. Da nur die bekanntesten Anbieter von AVM in dem Einrichtungsassistenten gelistet werden, wählen Sie für die Verwendung des Securepoint Dynamic DNS Service als Dynamic-DNS-Anbieter den Eintrag *Benutzerdefiniert* aus und tragen dort nachstehenden URL als Update-URL ein:

```
update.spdyn.de/nic/update?hostname=<domain>&myip=<ipaddr>
```

Beachten Sie, dass in diesem Beispiel die spitzen Klammern keine Platzhalter darstellen – geben Sie die Zeile einfach eins zu eins in den FRITZ!BOX-Dialog ein. Unterstützt die FRITZ!BOX das DynDNS-Update auch über SSL, dann verwenden Sie dort stattdessen die nachstehende Update-URL:

```
https://update.spdyn.de/nic/update?hostname=<domain>&myip=<ipaddr>
```

Viele Internetanbieter bieten zusätzlich zu einer IPv6-Hostadresse eine IPv4-Host-Adresse an und früher war es üblich, ausschließlich Adressen aus dem IPv4-Hostadressenbereich zu verwenden. Kabelanbieter wie Kabel Deutschland / Vodafone stellen aus Kompatibilitätsgründen den Internetanschluss sowohl mit einer IPv6-Adresse als auch mit einer IPv4-Hostadresse im Parallelbetrieb zur Verfügung. In Sachen dynamisches DNS bedeutet dies, dass die FRITZ!BOX neben dem IPv4-Host auch den IPv6-Host aktualisieren kann. Soll dies über spdyn.de erfolgen, dann verwenden Sie für beide Adressen idealerweise denselben Hostnamen und tragen im Feld *Update-URL* die nachfolgenden Zeilen ein:

```
https://update.spdyn.de/nic/update?hostname=<domain>&myip=<ipaddr>
https://update.spdyn.de/nic/update?hostname=<domain>&myip=<ip6addr>
```

Dieses Feld *Update-URL* ist im Menü *Erweiterte Einstellungen -> Internet -> Freigaben -> Dynamic DNS* zu finden.

Bild 1.4: Füllen Sie das Feld *Domainname* mit dem persönlichen spdyn-Hostnamen, das Feld *Benutzername* mit der Bezeichnung des spdyn-Accounts samt dazugehörigem Kennwort. Nach Klick auf die Schaltfläche *Übernehmen* kümmert sich die FRITZ!BOX um die automatische Aktualisierung der WAN-IP-Adresse für das dynamische DNS.

Soll stattdessen der Raspberry Pi die Aktualisierung der WAN-IP-Adresse für die dynamische DNS-Adresse übernehmen – beispielsweise, weil der eingesetzte Router keine diesbezüglichen Einstellungen für dynamisches DNS bietet – dann gehen Sie wie im nächsten Abschnitt beschrieben vor.

DNS-Update-Client für den Raspberry Pi

Das nachfolgende Verfahren lässt sich auch mit alternativen DynDNS-Anbietern anwenden – in diesem Beispiel wird der spdns-Dynamic-DNS-Update-Client für den oben angelegten spdyn.de-Account installiert und konfiguriert. Er steht kostenfrei auf spdns.de zur Verfügung und lässt sich bequem per `wget`-Kommando in das Home-Verzeichnis des Benutzers `pi` laden, vorausgesetzt der Raspberry Pi ist mit dem Internet verbunden.

```
cd ~
wget http://my5cent.spdns.de/wp-
content/uploads/2014/12/spdnsUpdater_bin.tar.gz
```

Nach dem Download entpacken Sie die Archivdatei per `tar`-Kommando:

```
tar -zxvf spdnsUpdater_bin.tar.gz
```

Die Archivdatei besteht aus zwei Dateien. Die Konfiguartionsdatei wird mit `sudo`-Berechtigungen in das `/etc`-Verzeichnis verschoben, die Binärdatei für das eigentliche Update wird in ein eigenes Verzeichnis – hier `updater` – verschoben, die nötigen Berechtigungen werden angepasst und die Eigentümerschaft gesetzt. Zu guter Letzt bearbeiten Sie mizt dem nano-Editor die Konfigurationsdatei `spdnsu.conf` im `/etc`-Verzeichnis und tragen dort den dynamischen Host und den Benutzernamen samt dazugehörigem Kennwort ein.

```
sudo mv spdnsu.conf /etc/
sudo mkdir -p updater
sudo mv spdnsu updater/
sudo chmod u+x updater/spdnsu
sudo chown -R pi:pi /home/pi/updater/
sudo nano /etc/spdnsu.conf
```

Nach dem Speichern der Konfigurationsdatei können Sie den Updater manuell ausführen und testen ob er auch die WAN-IP-Adresse des genutzten Internetanschlusses zurückliefert:

```
./updater/spdnsu
```

Die WAN-IP-Adresse des genutzten Internetanschlusses wird in diesem Beispiel in die Datei */tmp/spdnsuIP.cnf* geschrieben, deren Inhalt sich per `cat`-Kommando auf der Kommandozeile anzeigen lässt.

```
cat /tmp/spdnsuIP.cnf
```

Bild 1.5: Wird der Updater nach der Konfiguration manuell gestartet, wird die aktuelle IP-Adresse der WAN-Verbindung in die Datei */tmp/spdnsuIP.cnf* geschrieben.

Wer diese Aktualisierung automatisch vom Raspberry Pi erledigen lassen möchte, der nutzt dort die *crontab*-Datei. Dafür starten Sie diese im Bearbeitungsmodus

```
sudo crontab -e
```

und tragen dort die nachstehende Zeile ein:

```
*/10 * * * * /home/pi/updater/spdnsu
```

Speichern Sie die Datei und beenden Sie den Nano-Editor – nach einem Neustart des Raspberry Pi sollten Sie testen, ob alles wie gewünscht funktioniert, damit der Raspberry Pi die aktuelle WAN-Adresse automatisch an den DynDNS-Anbieter übermittelt.

Portfreigabe für den direkten Zugriff

Die TCP- und UDP-Ports (*User Datagram Protocol*) sorgen für die Kommunikation auf Netzwerk- bzw. Anwendungsebene. Grundsätzlich gilt auch hier: Weniger ist mehr. Je weniger Ports geöffnet und Dienste verfügbar sind, desto weniger Angriffsfläche bietet der LAN/WLAN-Router nach außen. So können Sie die Nutzung bestimmter Internetdienste wie das Surfen im WWW (HTTP), das *File Transfer Protocol* (FTP) und viele weitere für alle oder einige Benutzer in Ihrem Netzwerk blockieren.

Doch Vorsicht: Wird der Router zu sicher eingestellt, leidet die Funktionalität, weil bestimmte Programme nicht mehr richtig funktionieren. Wer beispielsweise einen Webserver (HTTP-Protokoll mit Port 80) hinter einem Router betreiben möchte, der muss den LAN/WLAN-Router so einstellen, dass die Anfragen aus dem Internet auch bis zum heimischen Webserver kommen können. Erst dann kann dieser reagieren und die Anfragen beantworten. Welchen Port Sie öffnen, hängt von dem eingesetzten Serverprogramm und vor allem von Ihren persönlichen Ansprüchen und Sicherheitsbedürfnissen ab.

Der LAN/WLAN-Router kann so eingestellt werden, dass bestimmte Ports am Router zwar offen sind, die Daten, die dort ankommen, aber nur an einen bestimmten Rechner bzw. eine bestimmte IP-Adresse weitergeleitet werden. Diese Technik läuft unter Portweiterleitung bzw. Port-Triggering. Die Porteinstellungen des WLAN-Routers nehmen Sie über die Weboberfläche vor, im Fall einer FRITZ!Box ist das der Dialog *Internet/Portfreigabe*.

Achten Sie darauf, dass bei der Konfiguration einer Portfreigabe die Zieladresse immer gleich bleibt. Es ist möglicherweise besser, für den Zielrechner im heimischen Netz eine feste IP-Adresse einzurichten. Verwenden Sie im Zweifelsfall statt einer DHCP-Adresse für den Computer eine statische IP-Adresse. Mithilfe der Portfreigabe lassen sich Dienste und verwendete Ports explizit bestimmten Rechnern im Heimnetz, in diesem Fall dem Raspberry Pi, zuordnen.

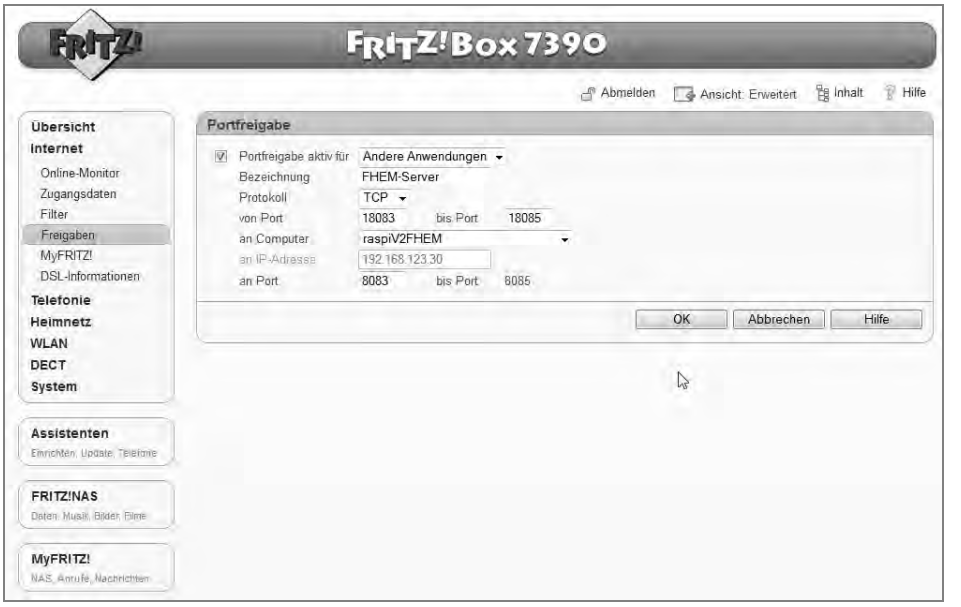

Bild 1.6: Per Klick auf die Schaltfläche *Freigaben* richten Sie eine neue Verbindung von außen mit dem Raspberry Pi im Netzwerk ein. In diesem Fall ist die Weboberfläche von außen über die Ports `18083` bis `18085` erreichbar, während im Heimnetz die Standardports `8083` bis `8085` genutzt werden.

Bild 1.7: Die *Dienstetabelle* listet bei Routern aus dem Hause Netgear alle Dienste auf, die aktuell gesperrt sind. Sie können dieser Tabelle Dienste hinzufügen oder welche daraus löschen.

Abhängig vom LAN/WLAN-Routermodell, ist auch der umgekehrte Fall möglich, und es lassen sich bestimmte Dienste und Ports für bestimmte Rechner blockieren. Bei Netgear-Modellen ist dafür der Schalter *Dienste sperren/Block Services* zuständig, mit

dem Sie den Internetzugang bestimmter Benutzer in Ihrem lokalen Netzwerk basierend auf deren IP-Adressen sperren können.

1.2 MQTT und IoT

Das Protokoll *Message Queue Telemetry Transport* (MQTT) ist beim Einsatz von IoT-Geräten ein oft genutztes Benachrichtigungsprotokoll, das in eigenen Heimautomations- und IoT-Projekten zum Einsatz kommen kann. Dafür wird ein sogenannter MQTT-Broker als Server zum Zweck des Datensammelns benötigt, der dafür auch rund um die Uhr als Service zum Sammeln und Verteilen der Informationen im Heimnetz zur Verfügung stehen muss. Damit sind generell mindestens drei Geräte für den Aufbau einer MQTT-Architektur nötig:

- MQTT-Broker (Datenzentrale)
- IoT-Sensor
- MQTT-Client

Die Daten, wie beispielsweise die Messwerte eines Sensors, werden zuvor in einer im MQTT-Broker definierten Datenstruktur regelmäßig automatisiert abgelegt. Über einen Client, egal ob Smartphone, Computer oder ein anderes IoT-/Hausautomationsgerät, der die Datenstruktur des eigenen MQTT-Brokers kennt, können diese Werte und Einträge ebenfalls regelmäßig und automatisiert ausgelesen werden. Auf diesen – oft lästigen – Umweg wird im Rahmen der verschiedenen Projekte in diesem Buch verzichtet. Stattdessen wird der direkte Weg zum Kommunikationspartner – wie beispielsweise zum Raspberry Pi, der die Auslieferung der Daten und die Kommunikation mit dem »Endkunden« erledigt – gewählt. Wer hingegen mehrere gleichartige Sensoren – beispielsweise in jedem Raum jeweils einen Luftdrucksensor, einen Temperatursensor und einen Bewegungsmelder – im Einsatz hat, der kann über den zentralen MQTT-Datenverteiler Aufwand sparen.

In diesem Projekt ist nach der Grundinstallation des MQTT-Brokers wie beispielsweise des kostenlosen Mosquitto (`http://mosquitto.org/download/`) und der Konfiguration der Umgebung auch jeweils die Einrichtung eines MQTT-Clients für die Datenablage sowie für das Auslesen der Daten nötig. Für Arduino-Boards steht dafür ein vielversprechender Ansatz auf GitHub (`https://github.com/256dpi/arduino-mqtt`) zur Verfügung, mit dem sich ein Arduino-Mikrocontroller als MQTT-Client und als Datenquelle für den MQTT-Server nutzen lässt. Die spätere Verwendung und Anbindung an einen Raspberry Pi stellt mit Python keine größere Hürde dar – hier reicht nach der Installation des `paho.mqtt.client`-Pakets die Definition `import paho.mqtt.client as paho` im Code aus, um ein entsprechendes Objekt wie `client = paho.Client()` zu erzeugen. Anschließend lassen sich die Basismethoden der API per Python verwenden.

Installation von Mosquitto

Bevor es an die Installation von Mosquitto geht, können Sie prüfen, welche MQTT-Protokollversion im Repository zur Verfügung steht. Bei Fertig-Sensoren kann es nämlich vorkommen, dass die verwendete MQTT-Protokollversion nicht mit jener auf dem MQTT-Broker zusammenpasst – dies lässt sich vor der Installation per `apt-get` überprüfen. Mit dem nachstehenden Kommando durchsuchen Sie den `apt-cache` nach Paketen, die mit der Bezeichnung Mosquitto verknüpft sind:

```
sudo apt-cache search mosquitto
```

In diesem Fall werden sowohl der Subscriber- als auch der Publisher-MQTT-Client mit einer Eigenbaulösung implementiert, sodass die Standardversion per `apt-get install` über die Kommandozeile installiert werden kann. Eine ältere bzw. angepasste Mosquitto-Version bzw. ihr Quellcode zum Selbstübersetzen ist auf den Projektseiten (`http://mosquitto.org/download/`) erhältlich.

```
:~ $ sudo apt-cache search mosquitto
libmosquittopp-dev - MQTT version 3.1 client C++ library, development files
mosquitto - MQTT version 3.1/3.1.1 compatible message broker
mosquitto-dbg - debugging symbols for mosquitto binaries
python-mosquitto - MQTT version 3.1 Python client library
python3-mosquitto - MQTT version 3.1 Python 3 client library
libmosquitto-dev - MQTT version 3.1/3.1.1 client library, development files
libmosquitto1 - MQTT version 3.1/3.1.1 client library
libmosquitto1-dbg - debugging symbols for libmosquitto binaries
libmosquittopp1 - MQTT version 3.1/3.1.1 client C++ library
libmosquittopp1-dbg - debugging symbols for libmosquittopp binaries
mosquitto-clients - Mosquitto command line MQTT clients
:~ $
```

Bild 1.8: Neben dem eigentlichen Mosquitto-Server sind über `apt-get` auch verschiedene MQTT-Clients in unterschiedlichen Versionen verfügbar.

Generell sind die Unterschiede bei der Installation zwischen Ubuntu-Linux, Raspbian Wheezy und Jessie sehr gering. Zunächst holen Sie sich die Schlüsselinformationen auf den Computer und fügen sie dem lokalen System hinzu.

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
```

Dann folgt die Update-Information – dies erledigen Sie ebenfalls auf der Kommandozeile:

```
sudo apt-get update
```

Beim Raspberry-Pi-Einsatz verwenden Sie für das aktuelle Raspbian Jessie die passende `list`-Datei für das Repository.

```
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
```

Analog gehen Sie vor, wenn noch die alte Wheezy-Welt auf dem Raspberry Pi zum Einsatz kommt – etwa bei Raspberry-Pi-Modellen der ersten Generation:

```
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-wheezy.list
```

Ist die Repository-Liste ergänzt, laden Sie sie und bringen die Update-Information abermals auf den neuesten Stand:

```
sudo apt-get update
```

Nach dem Update können Sie nun endlich den Mosquitto-Server auf dem Computer installieren. Es empfiehlt sich, zusätzlich die Clientwerkzeuge zu installieren, damit bei Bedarf auch lokal auf dem Server des MQTT-Brokers individuell Daten direkt auf der Konsole abgefragt werden können.

Bild 1.9: Bevor Sie Mosquitto installieren, bringen Sie mit `apt-get update` das System auf den aktuellen Stand.

Mosquitto steht unter anderem für Betriebssysteme aus dem Debian-Umfeld zur Verfügung. Neben dem Raspberry Pi gehört auch die Ubuntu-Familie dazu – dort verwenden Sie das nachstehende Kommando zur Mosquitto-Installation:

```
sudo apt-get install mosquitto mosquitto-clients
```

Soll ein Raspberry Pi als Mosquitto-Server im Heimnetz fungieren, ist das folgende Installationskommando zu verwenden:

```
sudo apt-get install mosquitto mosquitto-clients python-mosquitto
```

Ist im Heimnetz bereits ein Mosquitto-Server im Einsatz, reicht auf den Computern im Heimnetz ein einfacher MQTT-Client – das Serverpaket braucht nicht installiert zu werden. Um auf dem Raspberry Pi einen einfachen Mosquitto-Client zu installieren, verwenden Sie dieses Kommando:

```
sudo apt-get install mosquitto-clients python-mosquitto
```

Zu guter Letzt benötigen die Programmiersprachen und Tools auf dem Raspberry Pi die passenden Schnittstellen in Form der MQTT-Bibliotheken:

```
sudo apt-get upgrade
sudo apt-get install python-pip
```

bzw. bei Python 3:

```
sudo apt-get install python3-pip
```

Nach der Installation wird der MQTT-Server automatisch geladen und gestartet. Über die IP-Adresse des Raspberry Pi ist der MQTT-Server nun im Heimnetz verfügbar, nach der Basisinstallation wird automatisch auch der Standard-TCP-Port `1883` für das MQTT-Protokoll verwendet.

Betrieb von Mosquitto

Sind neben dem Server auch der Mosquitto-Client sowie die Python-Unterstützung installiert, prüfen Sie im ersten Schritt nach der Installation, ob Mosquitto gestartet wurde.

```
ps -A | grep mosquitto
```

Generell startet Mosquitto nach jedem Einschalten des Computers automatisch. Wie andere Dienste lässt sich auch Mosquitto mit dem `service`-Kommando von Hand beenden, (neu) starten, und der Status kann abgefragt werden.

```
sudo service mosquitto restart
sudo service mosquitto stop
sudo service mosquitto start
```

Der komplette Systemstatus kann lokal auf dem MQTT-Server mit dem Kommandozeilenbefehl

```
mosquitto_sub -v -t '$SYS/#'
```

ausgegeben werden. Die Mosquitto-Konfiguration lässt sich auch bequem ohne MQTT-Client lokal testen – dafür benötigen Sie zwei Kommandozeilenfenster.

Mosquitto-Schnellcheck ohne MQTT-Client

Sind Sie mit dem Raspberry Pi, der als MQTT-Server fungiert, per SSH-Client verbunden, erstellen Sie einfach eine zweite, neue Verbindung zum MQTT-Server und melden sich dort jeweils mit dem Benutzer `pi` an. Arbeiten Sie lokal auf dem Raspberry-Pi-System – etwa auf der GUI –, öffnen Sie das LXTerminal zweimal, damit in einem Fenster die MQTT-Hinweise angezeigt werden, die vom zweiten Fenster aus per Befehl initiiert werden.

Arbeiten Sie lokal, reicht die Angabe des `localhost` oder der IP-Adresse `127.0.0.1` aus – im Heimnetz verwenden Sie stattdessen die IP-Adresse des Raspberry Pi, der als MQTT-Broker/-Server fungiert.

In diesem Beispiel wird per `mosquitto_sub` das Topic `ubuSensor` erzeugt, und anschließend lauschen der Mosquitto-Server unter der Localhost-Adresse `127.0.0.1` auf die dazugehörigen Meldungen:

```
mosquitto_sub -h 127.0.0.1 -t ubuSensor/#
```

Das zweite SSH-Fenster zum Raspberry-Pi-Client wird benötigt, um über die Kommandozeile den Raspberry-Pi-Mosquitto-Server mit der IP-Adresse – in diesem Beispiel der IP `192.168.123.45` – mit Nachrichten/Kommandos zu versorgen. Sendet der Raspberry-Pi-Mosquitto-Client nun die Kommandos

```
mosquitto_pub -h 192.168.123.45 -t ubuSensor/DoorOpen -m "true"
mosquitto_pub -h 192.168.123.45 -t ubuSensor/DoorOpen -m "false"
```

sollte im zweiten Konsolenfenster des Raspberry-Pi-Mosquitto-Servers mit der IP `192.168.123.45` zunächst `true`, dann `false` ausgegeben werden. Das wäre beispielsweise der Status der Türöffnung, falls dort ein entsprechend eingerichteter Sensor angebracht ist, der nur die beiden Zustände mitteilen kann.

Bild 1.10: Neben den einfachen Zuständen wie `true`/`false` kann über MQTT auch komplexeres Datenmaterial einfach und bequem zum heimischen MQTT-Server übertragen werden.

Damit funktioniert die Mosquitto-Server-Lösung und kann als Datensammler für die MQTT-Pakete im Heimnetz für Raspberry-Pi-Clients, Arduino- und Pretzelboard-IoT-Lösungen sowie eigene Sensoren verwendet werden.

Arduino als MQTT-Client

Aufgrund der Technologie und der Bauweise eignen sich die kleinen Arduino-Mikrocontroller aller Art nahezu perfekt für die IoT-Welt, denn sie bringen zum einen diverse Schnittstellen und Anschlussmöglichkeiten für Aktoren und Sensoren mit und benötigen zum anderen vergleichsweise wenig Strom. Sogar mit Akkus bzw. Batterien lassen sie sich betreiben. Generell ist für den MQTT-Einsatz natürlich eine Verbindung zum MQTT-Broker nötig, damit Messwerte, Status, Informationen und Ähnliches turnusmäßig dort abgeliefert werden können. Demzufolge ist ein Netzwerkanschluss wie beispielsweise über Wi-Fi oder eine kabelgebundene Ethernet-Verbindung für den Einsatz eines Arduino und kompatible MQTT-Clients zwingend notwendig.

Pretzelboard als Temperatursensor

Zunächst ist es dem MQTT-Broker egal, welche Daten von wem in welchem Format zu Verfügung gestellt werden. Wichtig ist, dass der bzw. die möglichen Empfänger in Form von MQTT-Abonnenten diese Daten nicht nur lesen, sondern auch entsprechend interpretieren können. Demzufolge sollten die Attribute entsprechende Bezeichnungen haben, damit Sie auch den Überblick über die Aktoren, Sensoren und MTTQ-Clients im Heimnetz behalten. Setzen Sie beispielsweise einen Arduino oder kompatible Boards zur Temperaturmessung ein, ist die eigentliche Schaltung in wenigen Augenblicken auf dem Steckboard umgesetzt.

DS18B20 - Pin	Arduino -Pin	Bemerkung
GND / 1	GND	Masse
DQ / 2	D2	Datenpin
VDD / 3	VCC / 5V	Spannung

In diesem Beispiel werden der günstige Dallas DS18B20 zum Stückpreis von ca. 2,20 Euro, ein 4,7-kΩ-Widerstand sowie die nötigen Pin-Kabel eingesetzt, die Strom, Masse und Datensignal übertragen. Der Temperatursensor DS18B20 von Dallas eignet sich für Messungen im Bereich von –55 °C bis +125 °C und reicht somit für den normalen Hausgebrauch im Rahmen dieses Projekts aus.

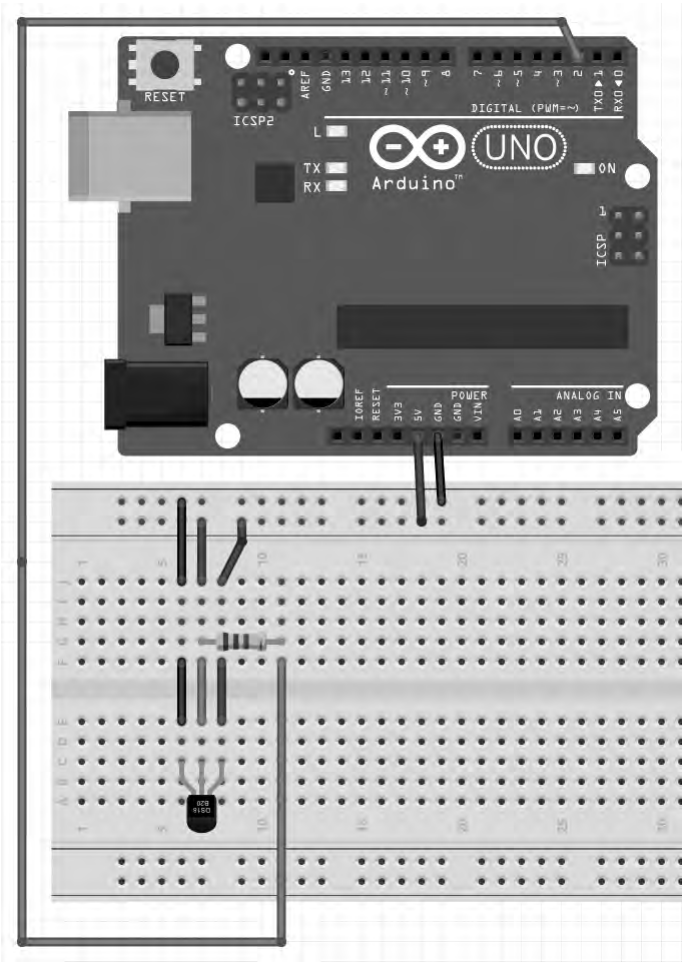

Bild 1.11: Egal ob Pretzelboard oder Arduino-Original: Der DS18B20 kommt mit einer Spannung von 3,0 bis 5,5 V aus. Damit nutzen Sie den VDD-Anschluss für die 5 V des Arduino und die Masse zum GND-Pin des Arduino.

Der Datenpin wird mit dem 5-V-Anschluss sowie einer Seite des 4,7-kΩ-Widerstands verbunden, das Gegenstück des Widerstands wird über das Steckboard mit dem Datenpin (D2) des Arduino verbunden.

Bild 1.12: Testaufbau: Auf dem Steckboard ist der DS18B20 schnell mit dem Arduino verbunden.

»Steht« die Schaltung, prüfen Sie, ob die für den 1-Wire-Betrieb nötigen Bibliotheken in der Arduino-IDE zur Verfügung stehen.

Bezeichnung	Bezugsquelle
OneWire Library	www.pjrc.com/teensy/arduino_libraries/OneWire.zip
DallasTemperature Library	http://download.milesburton.com/Arduino/MaximTemperature/DallasTemperature_372Beta.zip

Zum Einsatz kommen im nachfolgenden Sketch die OneWire Library sowie die DallasTemperature Library, die beide nach dem Download über den in der Arduino-IDE integrierten ZIP-Installer hinzugefügt werden können.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2
//
// arduino_pretzelboard_temperatur.ino
//
OneWire oneTempWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneTempWire);

void setup(){
delay(1000);
Serial.begin(19200);
Serial.print("[Pretzelboard] :");
Serial.println(" Pin D2 – Temperature Sensor DS18B20");
```

```
delay(1000);
sensors.begin();
}

void loop() {
Serial.println();
Serial.print("[Pretzelboard] Hole Temperatur...");
sensors.requestTemperatures();
Serial.println("...OK");
Serial.print("Sensor = ");
Serial.print(sensors.getTempCByIndex(0));
Serial.println(" Grad C");
delay(2000);
}
```

Nach der Installation werden sie wie gewohnt per `include`-Direktive am Anfang des Sketches eingebunden. Der Pin-Anschluss D2 wird im Sketch der Konstanten `ONE_WIRE_BUS` zugewiesen, die ihrerseits für den Konstruktor der `OneWire`-Klasse nötig ist. Die `DallasTemperature`-Klasse wiederum zeigt auf das definierte `OneWire`-Objekt und stellt über die Funktion `requestTemperatures` schließlich die gemessene Temperatur bereit.

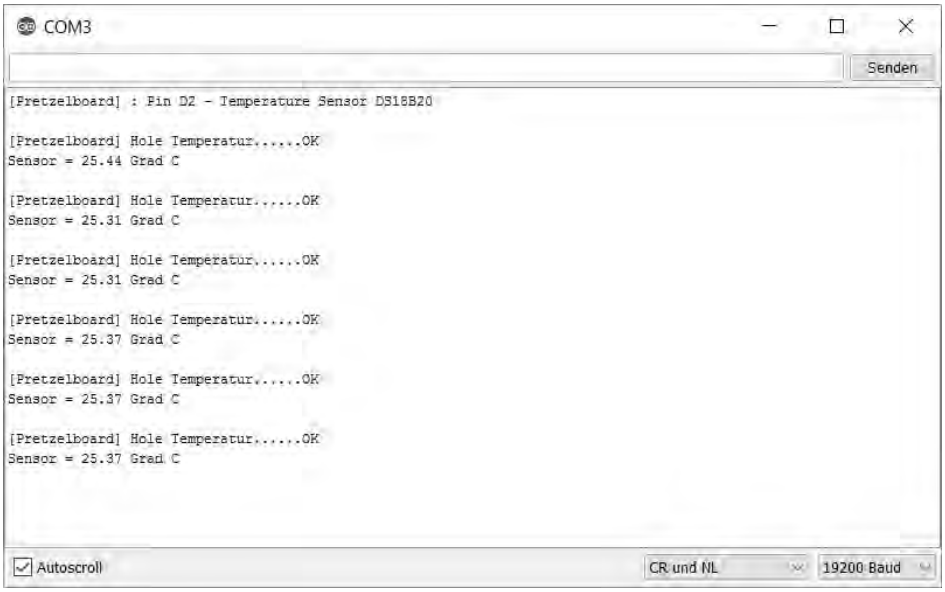

Bild 1.13: Nach dem Kompilieren des Sketches und dem Hochladen des Kompilats in den Speicher liefert der Sensor die Temperatur wie gewünscht auf der seriellen Konsole ab.

Im nächsten Schritt nehmen Sie Verbindung zum heimischen Netzwerk auf. Dafür kommt ein Arduino Uno zum Einsatz, der sich mit wenigen Handgriffen durch ein Ethernet-Shield erweitern lässt. Die Schaltung für den Temperatursensor sowie die verwendeten Pins auf dem Arduino bleiben unverändert.

Arduino UNO als MQTT-Client

Der obige Sketch für das Auslesen der Temperatur dient in diesem Projekt als Basis und wird zusätzlich mit einer MQTT-Client-Library bestückt. Dieser MQTT-Client ist als ZIP-Paket über `http://knolleary.net/arduino-client-for-mqtt` erhältlich – er kann aber auch über den in der Arduino-IDE eingebauten Library Installer per Suchbegriff `mqtt` gesucht und anschließend installiert werden. Wie auch immer, auch in diesem Fall muss die neue Library über die `include`-Direktive im Sketch eingebunden werden, damit die `mqtt`-Funktionen genutzt werden können.

```
#include <SPI.h>
#include <Ethernet.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <PubSubClient.h> // http://knolleary.net/arduino-client-for-mqtt/
```

Wie im obigen Pretzelboard-Beispiel ist der Temperatursensor an Pin D2 angeschlossen. Für den Netzwerkzugriff legen Sie schließlich noch die MAC-Adresse des Ethernet-Shields im Sketch fest. Für optionale »Beschriftungen« und Variablenbezeichnungen wurde die MAC-Adresse auch gleich in einer lesbare Übersetzung in der Variablen `macstr` angelegt.

```
#define ONE_WIRE_BUS 2
byte MAC_ADDRESS[] = { 0xDA, 0xEA, 0xBA, 0xFA, 0xFE, 0xED };
char macstr[] = "DAEABAFAFEED";
EthernetClient ethClient;
```

Zwingend nötig ist die Angabe der IP-Adresse des MQTT-Servers im Heimnetz in der `localserver`-Variablen des Typs `byte`, deren Adressanteile mit Kommata getrennt werden:

```
byte localserver[] = { 192, 168, 123, 45 };
```

Der vorhandene Arduino-Sketch dient als MQTT-Client zur Kommunikation mit dem Server, die über den TCP Port `1883` stattfindet. Die Funktion `callback` im Konstruktor ist nötig für die Ausgabe der Nachrichten, falls der MQTT-Server Nachrichten und Bestätigungen zurückmeldet.

```
PubSubClient mqttClient(localserver, 1883, callback, ethClient);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature TempSensors(&oneWire);
```

Nach der Bindung des Temperatursensors an den OneWire-Bus werden der MQTT-Name des Arduino-Clients sowie der Topic-Name – also der Ast im MQTT-Baum, in den die Informationen eingehängt werden sollen – definiert.

```
String clientName = String("d:dahoam:arduino:") + macstr;
String topicName = String("iot/event/status/fmt/json");
```

Anschließend werden die Temperaturvariable und die Variable für die String-Konvertierung definiert.

```
float tempC = 0.0;
char tempBuffer[100];
```

Im `setup()`-Block werden sämtliche Geräte einmalig initialisiert und in Betrieb genommen.

```
void setup(){
 // Initialisierung der seriellen Schnittstelle
 // zur Ausgabe der empfangenen Nachrichten
 Serial.begin(19200);
 // Initialisierung des Ethernets
 if (Ethernet.begin(MAC_ADDRESS) == 0) {
    Serial.println("[MQTT] Failed with DHCP");
    return;
  }
TempSensors.begin();
}
```

Im wiederkehrenden `loop`-Block wird die Temperatur ausgelesen und bei einer Abweichung über die `if`-Abfrage neu gesetzt.

```
void loop(){
 char clientStr[34];
 clientName.toCharArray(clientStr,34);
 char topicStr[26];
 topicName.toCharArray(topicStr,26);
TempSensors.requestTemperatures();
 if (tempC != TempSensors.getTempCByIndex(0) ) {
 tempC = TempSensors.getTempCByIndex(0);}

 if (!mqttClient.connected()) {
    Serial.print("Trying to connect to: ");
    Serial.println(clientStr);
    mqttClient.connect(clientStr);
  }
```

Falls die Daten vom `mqttClient` als JSON-String übermittelt werden, ist eine alternative Möglichkeit das Auslagern der dafür nötigen Arbeiten in die Funktion `buildJson`.

```
  if (mqttClient.connected() ) {
    String json = buildJson();
    char jsonStr[200];
    json.toCharArray(jsonStr,200);
    boolean pubresult = mqttClient.publish(topicStr,jsonStr);
    Serial.print("attempt to send ");
    Serial.println(jsonStr);
    Serial.print("to ");
    Serial.println(topicStr);
    if (pubresult){
      Serial.print("[");
      Serial.print (clientName);
      Serial.println ("] successfully sent");
    }
    else {
      Serial.print("[");
      Serial.print(clientName);
      Serial.println("] unsuccessfully sent");
    }
  }
```

Über die Rückmeldung an das Flag `pubresult` (erfolgreich/nicht erfolgreich) wird eine Bildschirmmeldung ausgegeben, die mitteilt, ob die MQTT-Nachricht übermittelt worden ist oder nicht.

```
delay(10000);
```

Schließlich ist die bereits angesprochene Callback-Funktion von MQTT zu erstellen. Die Funktion wird automatisch aufgerufen, falls der MQTT-Server eine Rückmeldung zum Client schickt – beispielsweise ein `CONNACK`. Dafür werden zunächst die nötigen Variablen und Hilfsvariablen für die String-Konvertierung definiert, die Ende-Markierung `'\0'` wird gesetzt, und die Nachricht wird in ein String-Objekt für die Ausgabe auf der seriellen Konsole konvertiert.

```
void callback(char* topic, byte* payload, unsigned int length) {
  int i = 0;
  char message_buff[100];
 Serial.println("Message arrived: topic: " + String(topic));
 Serial.println("Length: " + String(length,DEC));
  for(i=0; i<length; i++) {
   message_buff[i] = payload[i];
   }
  message_buff[i] = '\0';
// Konvertierung der Nachricht in einen String
  String msgString = String(message_buff);
  Serial.println("Payload: " + msgString);
}
```

Das Zusammenbauen des JSON-Strings erfolgt zeichenweise. Nach der Initialisierung wird Schritt für Schritt der Inhalt zusammengebaut – für den Wert der Temperatur ist eine Typumwandlung über die `dtostrf`-Funktion nötig.

```
String buildJson() {
  String data = "{";
  data+="\n";
  data+= "\"d\": {";
  data+="\n";
  data+="\"iotName\": \"Arduino DS18B20\",";
  data+="\n";
  data+="\"temperature (C)\": ";
  data+=dtostrf(tempC, 1, 2, tempBuffer);
  data+="\n";
  data+="}";
  return data;
}
```

Nach dem Speichern des Sketches in der Arduino-IDE wird es Zeit, diesen in Betrieb zu nehmen. Passen Sie den Sketch an, speichern Sie ihn und laden Sie ihn auf den Arduino hoch. Ist der Mosquitto-Server im Heimnetz aktiv, können Sie auf der Konsole des Raspberry Pi prüfen, ob auch Temperaturmeldungen vom Arduino eintreffen. Die einfachste Möglichkeit ist die Überwachung des entsprechenden Topics – in diesem Beispiel wurde über die Variable `topicName` im Sketch der Pfad `iot/event/status/fmt/json` festgelegt.

Mit dem Kommando:

```
mosquitto_sub -h 127.0.0.1 -t iot/event/status/#
```

horchen Sie den Mosquitto-Server nach Nachrichten ab. Schlägt der Arduino-Sensor an, sollte er einen JSON-String publizieren, der anschließend auf der Konsole des MQTT-Servers im definierten Format erscheinen sollte.

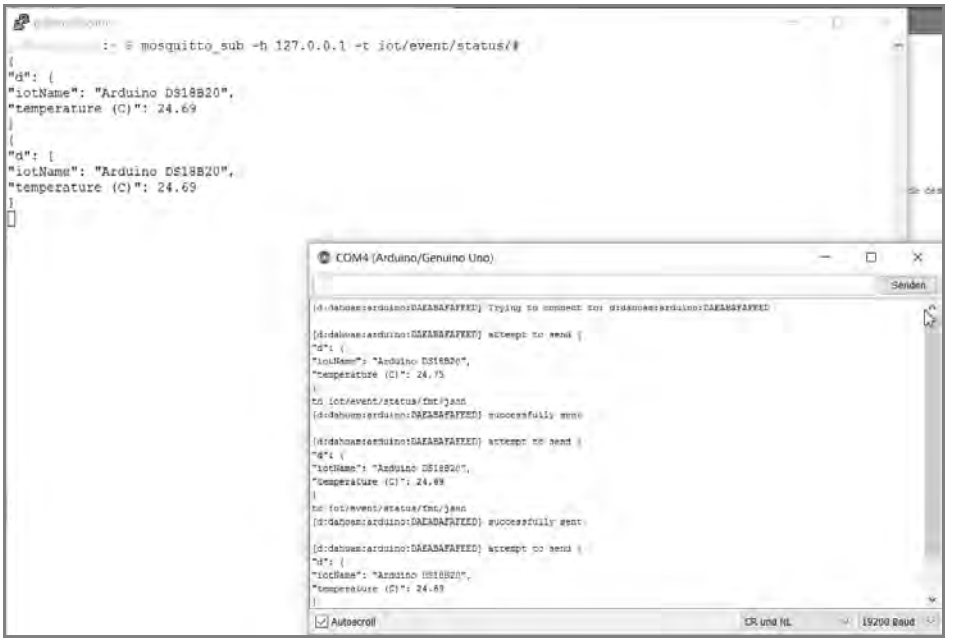

Bild 1.14: Übertragung erfolgreich: Der ebenfalls auf der seriellen Konsole der Arduino-IDE angezeigte Temperaturwert wird über das MQTT-Protokoll zum verknüpften MQTT-Broker im Heimnetz geschickt, dort in Empfang genommen und in der Konsole ausgegeben.

Wer den Beispielsketch in der Praxis einsetzt, wird merken, dass der Arduino sozusagen bei jeder kleinsten Temperaturänderung bereits ein Update und somit auch eine MQTT-Ausgabe produziert. Das ist in der Praxis jedoch kaum notwendig. Aus diesem Grund ist es sinnvoll nur dann, eine Benachrichtigung anzutriggern, wenn die Temperaturabweichung einen vorher festgelegten Grenzwert überschreitet. In diesem Beispiel wurde dieser auf eine Abweichung von 2 °C festgelegt. Die Anpassung im Code ist minimal, die Zeile

```
  if (tempC != TempSensors.getTempCByIndex(0) ) {
```

wurde um eine Differenzwertberechnung ergänzt, die den absoluten Wert der Temperaturabweichung bestimmt.

```
  if (tempC != TempSensors.getTempCByIndex(0) && ( abs(tempC –
TempSensors.getTempCByIndex(0)) > thresholdC) ) {
```

Die Abweichung von 2 °C ist über die Konstante `thresholdC` am Anfang des Sketches festgelegt. Wie auch immer Sie den Wert für die gewünschte Abweichung wählen, der Arduino-Publisher ist nun aktiv und übermittelt die Daten zum MQTT-Broker. Abgesehen vom Arduino lässt sich der MQTT-Broker noch mit weiteren Geräten als

Publisher und Subscriber koppeln, und im nächsten Projekt kommt dafür ein weiterer Raspberry Pi samt angeschlossenem Sensor zum Einsatz.

Raspberry Pi als MQTT-Client

Um die Verwendung und die Funktionsweise des MQTT-Benachrichtigungsprotokolls im IoT-Umfeld zu Hause zu demonstrieren, kommt in diesem Projekt nun neben dem Raspberry-Pi-MQTT-Server und dem Arduino-Client ein weiterer Raspberry Pi als MQTT-Client zum Einsatz. Für die Python-Anbindung nutzt dieser die Paho-Library, die sich per `pip`-Routine auf den Raspberry Pi nachinstallieren lässt:

```
sudo pip install paho-mqtt
```

Es wird davon ausgegangen, dass bereits ein MQTT-Broker installiert wurde – hier ein Raspberry Pi mit der IP-Adresse `192.168.123.45` – und somit im Heimnetz zur Verfügung steht. In diesem Projekt übermittelt ein Arduino als »Publisher« seine Temperaturdaten an den Server, der als MQTT-Broker fungiert. Diese Daten des Arduino werden in dem nachstehenden Projekt über den MQTT-Broker von einem weiteren Raspberry Pi, der als sogenannter »Subscriber« im Heimnetz fungiert, per Python abgefragt und ausgelesen. Ist der Quellcode der Datei `pi_readmqtt.py` aus dem Verzeichnis `\mqtt` über die Kommandozeile des Raspberry Pi gestartet, können Sie die Daten vom lokalen MQTT-Broker lesen und abonnieren – in diesem Fall stellt ein Arduino per MQTT die Temperaturdaten zur Verfügung.

```
# -*- coding: utf-8 -*-
#!/usr/bin/python
# ------------------------------------------------------------------------
# E.F.Engelhardt, Franzis Verlag, 2016
# ------------------------------------------------------------------------
# pi_readmqtt.py
# ------------------------------------------------------------------------
import paho.mqtt.client as mqtt
# ------------------------------------------------------
# mqtt-Konfig
localserver = "192.168.123.45"
topic = "iot/#"
```

Generell wird die `callback`-Funktion benötigt, damit der MQTT-Client die `CONNACK`-Rückmeldungen des MQTT-Servers empfangen kann – damit bestätigt der Server den Eingang der übermittelten Daten.

```
def on_connect(client, userdata, rc):
   print("Connected with "+localserver +"\n\nresult code : "+str(rc))
```

In dieser `on_connect`-Funktion wird über das `subscribe`-Kommando das gewünschte Topic abonniert. Bei einem etwaigen Verbindungsabbruch wegen eines Netzwerk-

fehlers etc. wird die Verbindung bei einer erfolgreichen Netzwerkverbindung automatisch wieder aufgenommen – die MQTT-Technik ist also fehlertolerant.

```
   print("subscribe : " +topic)
   client.subscribe(topic)
# ------------------------------------------------
```

Über die `on_message`-Funktion wird die Rückmeldung des MQTT-Servers auf der Konsole ausgegeben.

```
def on_message(client, userdata, msg):
   print "Topic: ", msg.topic+'\nMessage: '+str(msg.payload)
# ------------------------------------------------
def start_mqtt():
   client = mqtt.Client()
   client.on_connect = on_connect
   client.on_message = on_message
   client.connect(localserver, 1883, 60)
   client.loop_forever()
# ------------------------------------------------
start_mqtt()
# ----------------------- EOF --------------------------
```

Mit dem Kommando `sudo python pi_readmqtt.py` starten Sie auf der Konsole das Programm, um das gewünschten Topic aus dem MQTT-Broker zu fischen.

```
:~ $ sudo python pi_readmqtt.py
Connected with 192.168.123.45

result code : 0
subscribe : iot/#
Topic:  iot/event/status/fmt/json
Message:
"d": {
"iotName": "Raspberry Pi DHT11",
"temperature (C)": 25.0,
"humidity": 44.0
}
Topic:  iot/event/status/fmt/json
Message:
"d": {
"iotName": "Raspberry Pi DHT11",
"temperature (C)": 25.0,
"humidity": 44.0
}
Topic:  iot/event/status/fmt/json
Message:
"d": {
"iotName": "Raspberry Pi DHT11",
"temperature (C)": 25.0,
"humidity": 44.0
}
```

Bild 1.15: Verbindung per Raspberry Pi erfolgreich: Der MQTT-Server übermittelt die Daten des gewünschten Topics.

Im nächsten Schritt verwenden Sie den Client-Raspberry als IoT-Datenlieferant für den MQTT-Broker.

Raspberry Pi als MQTT-Publisher

In diesem Projekt wurde der Billigsensor DH11 aus Fernost eingesetzt, der unter der Bezeichnung `DHT11 (KY-015)` als Bestandteil des 37-teiligen Sensorpakets `37 in 1 Sensor Modul Kit` auf den gängigen Internetplattformen erhältlich ist. Um diesen Temperatur- und Feuchtigkeitssensor mit dem Raspberry Pi zu koppeln, benötigen Sie anders als bei den Einzelbauteilen keine passende Schaltung, da dieses Modell bereits einen 10-kΩ-Pull-up-Widerstand auf der Miniplatine mitbringt. Im Fall eines Einzelbauteils verwenden Sie zusätzlich ein Steckboard und einen 10-kΩ-Widerstand, der die Datenleitung mit dem Spannungspin verbindet. Der Temperatur-/Luftfeuchtigkeitssensor DHT11 deckt einen Temperaturbereich von 0 bis 50 °C mit einer Abweichung von +/–2 °C sowie eine relative Feuchte von 20 bis 95 % (+/–5 %) ab. Die nötige Spannungsversorgung von 3 bis 5,5 V ist für den Arduino- und Raspberry-Pi-Einsatz prädestiniert. Beim Anschluss an der GPIO-Pinleiste des Raspberry Pi werden drei Pins gemäß der nachstehenden Tabelle benötigt.

DHT11 (KY-015) – Pin	RPi-Pin	Bemerkung
Se / 1	7 / GPIO4	Datenpin
Mitte / 2	2 / 5V	Spannung
– / 3	9 / GND	Masse

Ist der Raspberry Pi mit dem DHT11-Sensormodul verbunden, benötigen Sie für den einfachen Python-Zugriff eine passende Python-Library, um die Basisfunktionen wie das Auslesen der Messwerte aus den Registern nicht mühsam zusammenfrickeln zu müssen. Idealerweise existiert für den DHT11 und den DHT22 (baugleich) bereits ein Zwillingsprodukt von Adafruit, für das eine passende Python-Library kostenfrei zum Download zur Verfügung steht. Um diese auf dem Raspberry Pi zu installieren, nehmen Sie nachfolgende Anweisungen in der Konsole vor – am Ende des Tages lässt sich die Adafruit-Python-Library dann für den DHT11-Sensor verwenden:

```
sudo apt-get update
sudo apt-get install build-essential python-dev git
mkdir -p dht11
cd  dht11
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
sudo python setup.py install
```

Sind die Voraussetzungen für den Python-Einsatz geschaffen, lesen Sie die Daten des DHT11 regelmäßig aus und publizieren sie per JSON-Datensatz auf dem MQTT-Broker im Heimnetz.

Daten des Temperatur- und Feuchtigkeitssensors mit MQTT publizieren

Das Datenkabel des Sensors wird mit dem GPIO4-Pin der GPIO-Leiste des Raspberry Pi verbunden, und dieser Pin wird auch für die Verwendung der Adafruit_DHT-Bibliothek im Python-Code benötigt. Generell ist der Aufbau des Python-Codes zum Publizieren der MQTT-Daten nahezu identisch mit dem Code des Subscribers, also dem Auslesen von Daten von MQTT-Brokern, was dem einfachen MQTT-Protokoll geschuldet ist. Das muss kein Nachteil sein, denn es hält auch den Entwicklungsaufwand in Grenzen. In diesem Fall wird eine weitere Funktion mit der Bezeichnung `buildJson` benötigt, die ihrerseits den JSON-Datensatz zusammenbaut und die beiden Variablen für die Temperatur sowie die Luftfeuchtigkeit über die `Adafruit_DHT.read_retry`-Funktion befüllt.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import re, os, time
import paho.mqtt.client as mqtt
import Adafruit_DHT
#
GPIOpin = 4
#
localserver = "192.168.123.45"
topic = "iot/event/status/fmt/json"
#
def on_connect(client, userdata, flags, rc):
   print("Connected with "+localserver +"\n\nresult code : "+str(rc))
# ----------------------------------------------------
def start_mqtt():
   # client = mqtt.Client()
   client.on_connect = on_connect
   client.connect(localserver, 1883, 60)
   print "Verbindung zu "+localserver+" erfolgt..."
# ----------------------------------------------------
def buildJson():
  humidity, temperature = Adafruit_DHT.read_retry( Adafruit_DHT.DHT11,
GPIOpin )
  if (humidity is not None and temperature is not None):
#    print "Temp={0:f}*C  Humidity={1:f}%".format(temperature, humidity)
    tempDHT = str(temperature)
    humiDHT = str(humidity)
  else:
    tempDHT = "0"
```

```
    humiDHT = "0"
  data= ""
  "{".join(data)
  data+="\n"
  data+= "\"d\": {"
  data+="\n"
  data+="\"iotName\": \"Raspberry Pi DHT11\","
  data+="\n"
  data+="\"temperature (C)\": "
  data+=tempDHT
  data+=","
  data+="\n"
  data+="\"humidity\": "
  data+=humiDHT
  data+="\n"
  data+="}"
  return data
# ----------------------------------------------------
# schreibe daten in mqtt-json
client = mqtt.Client()
start_mqtt()
while True:
  # read sensor data
  print buildJson()
  client.publish(topic, buildJson())
  time.sleep(10)
```

Passen Sie in dem Codebeispiel `pi_publishmqtt_dht11.py` die Variablen `localserver` und `topic` an Ihre Heimnetz- und MQTT-Serverumgebung an und starten Sie den Raspberry-Pi-Publisher auf der Kommandozeile. Nach wenigen Augenblicken sollte der MQTT-Broker im Heimnetz anschlagen und den übermittelten JSON-Datensatz auf der Konsole anzeigen.

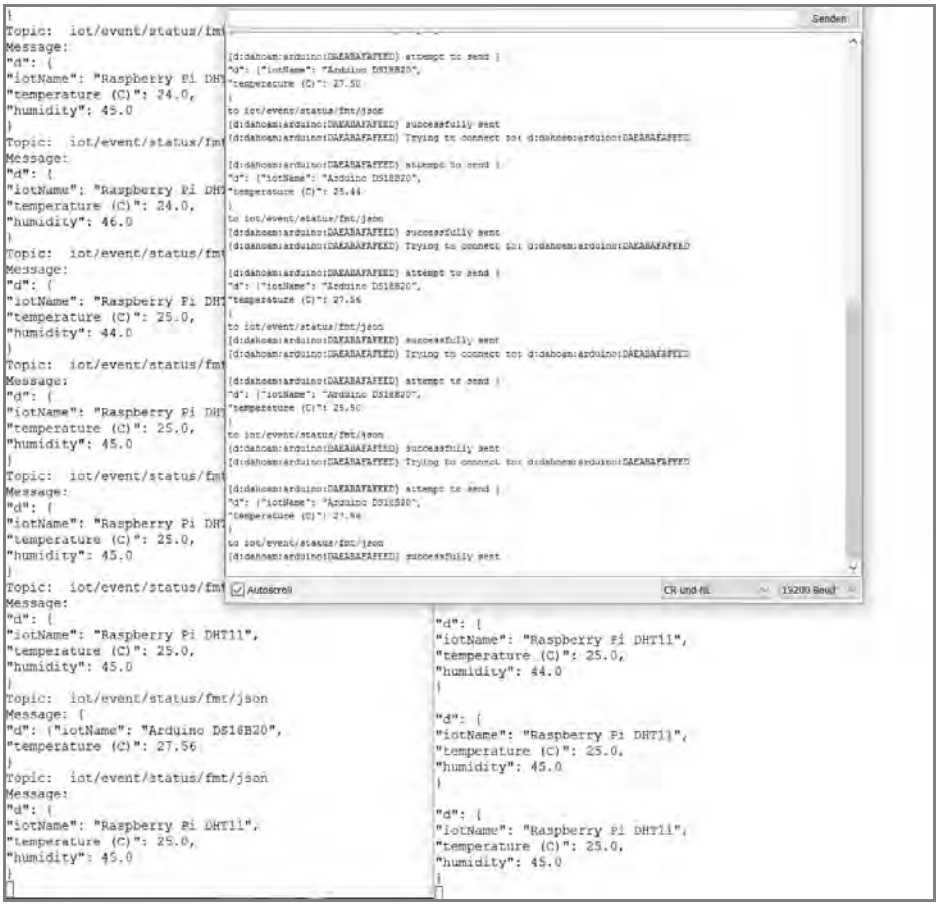

Bild 1.16: Links sehen Sie das Konsolenfenster des MQTT-Brokers, das die Messwerte des Arduino (mittig) und des Raspberry Pi mit dem DHT11-Sensor übersichtlich per JSON-Ausgabe darstellt.

Das eigentliche Publizieren erfolgt schließlich über die definierte `while True`-»Dauerschleife«, die den JSON-Datensatz zum definierten Topic zum MQTT-Broker überträgt. Für den schnellen Austausch von »kleinen« Informationen und Zuständen ist das MQTT-Protokoll eine praktische Ergänzung und eine tolle Sache – bei vielen Aktoren und Sensoren stellt es jedoch auch einen großen Aufwand dar, der sich im Heimnetz am Ende des Tages lohnen sollte.

Stichwortverzeichnis

Symbole

.htaccess erzeugen 330
.htaccess-Datei, Zugriffsschutz 174
.htpasswd erzeugen 330
/dev/ttyACM0 352
/dev/ttyUSBsu0 352
/dev/video 140
/etc/crontab-Datei 159

1

1-Wire 423
  OWFS 424
1-Wire-Sensoren 480
1-Wire-Technik 660

8

868 MHz 72, 74

A

Aktive Inhalte 54
Alarmfunktionen 180
Anwesenheitserkennung 186
Apache 95, 114
  installieren 328
  mod_webdav 113
Apple 556
apt-get 112, 650
Arduino 466
  DallasTemperature Library 36
  ESP8266 770
  Fingerscannerüberwachung 301
  Haus-/Wohnungstür 379
  Haushaltsgeräteüberwachung 770
  LCD-Bildschirm 382
  Mobilfunksteuerung der Haustür 381
  MQTT 34
  OneWire Library 36
  SIM900-Mobilfunkmodem 384
  UDP 301
  UDP-Pakete parsen 302
  Waschmaschinenüberwachung 770
Arduino Nano, mit WLAN-Modul 163
Arduino-IDE 466
Arduino-Sketch
  Authentifizierung 169
  Raspberry Pi benachrichtigen 171
  Verbindung ins Heimnetz 168
  WLAN 168
ARP-Cache 62
ARP-Rundsendung 62
AT+CMGS 359
AT-Kommandos 167, 362
ATmega 466
authorized_keys 681
autocreate 104

B

BACnet 72
Baumarktsteckdosen 458
Benachrichtigung mit PHP 150
Beschleunigungssensor 770
Betriebssystem auffrischen 80
Betriebssystem-Images 315
Bewegungsmelder 132, 142
  im Eigenbau 161
  low-cost 161

Shell-Skript 132
über WLAN 168
Bild- und Videoüberwachung 243
Kamera 244
Bildarchivierung, chronologisch 235
BLE 52
Bluetooth 186, 318, 675
Erkennung über Shell-Skript 679
Raspberry Pi 3 632
Bluetooth als Aktor 674
Bluetooth Low Energy Siehe BLE
Bluetooth-Authentifizierung 675
Bluetooth-Daemon 675
Bluetooth-Mini-USB-Adapter 120
bluez-utils 675
Brandgefahr 447

C

CC1101 72
ChallengeResponseAuthentication 686
Class-C-Netzwerk 19
COC 75
Computer steuern 681
config.txt 83
Couldn't open RAW socket 651
Cronjob, regelmäßige Überprüfung 158
crontab 518
CSI-Anschluss 79
CUL 72, 76
CUL-Modul 75
CUPS 95, 692
curl 113
cURL 686

D

Daemon 133
Datentransport, netcat 295
Dauerbetrieb, Netzwerk 225
define 697
Degrees of Freedom Siehe DOF
DHCP 20, 788
DHCP-Server 20
dmesg 77
DNS 22
DNS-Adresse 117
DNS-Server 21
DOF 770
DoorBird 203
API 218
App besorgen 207
App und Klingel koppeln 212
automatische Benachrichtigung 224
Bildarchiv 222
Eigenschaften 204
HD-Kamera 205
per App konfigurieren 209
Praxis 217
Relais 223
WLAN-Gerät 205
DoorBird-Modding 224
DoorBird-Türstation
Livestream 221
Drahtlos kommunizieren 318
Drehstromzähler 417, 418
Dropbox 108
Drucker 692
DS1920 424
DS9097U 419
DS9490R 419
DSL-WLAN-Router 27
Dummy 690
DynDNS 22

E

E27-Lampenfassung 562, 592
E27-Sockel 591
eHZ 417, 788
ekey-Steuereinheit 280
Ersteinrichtung 284
Fingerscanner 285

Inbetriebnahme 284
ekey-System, Logging 288
Elektronischer Schließzylinder 261
Elektronisches Schloss 260
Elgato Eve 644
E-Mail
Datenschutz 55
Spam verhindern 55
E-Mail-Benachrichtigung 180
Python-Skript 180
E-Mail-Versand optimieren 181
Enable Camera 81
Energiemessung 464
Energy Manager 462
EnOcean 72
EOF 788
EQ3 74
ESP8266 52, 770
exec 694
ext3 111

F

Farben umrechnen 578
FAT32 111
fdisk 111
Fernbedienungen der Harmony-Serie 718
Feuchtigkeitsmessung 670
FHEM 78, 201, 444, 459, 464, 473, 657
aktualisieren 100
DIP-Schalter 459
FS20-Druckersteckdose 697
Hue-Lampen 582
Logfile 78
Plot 435
Start 98
Startskript 94
Zugriffskennwort 102
fhem2mail 445
FHT 74, 103
FHZ1000 72
Fingerscannerüberwachung, Arduino 301
Firmware
auffrischen 80
SIM900-Modem 339
FQDN 113
FS20 72, 74, 495
FS20-KSE 193
FS20-Notation 104
fstab 112
fswebcam 141, 199
Funkheizkörpermodule 492
Funksmog 437
Funksteckdose 105, 464
Funksteckdosenset 447
Funkthermostat 494
Funktion
checkblue.sh 680
tempsensor.pl 487
tempsensor.py 488
fuse 424
fusermount 427

G

Gaszähler 432
Gateway 20
General Purpose Input/Output 86
Geofencing 188
Geofancy 188
Geofency 188
Geofencing-App 186
Gerätecode 104
Gerätenamen 20
Gertboard 86
getcap 679
GitHub 84
GND 788
Gnokii 120
SMS-Versand 125
Gnokii-Wiki 122
Google Drive 108

Google-Nest-API 151
GPIO 75, 86, 447, 448
  Pinbelegung 87
GPIO-API 91
GPIO-Bibliothek 786
GPIO-Steuerung, über PHP und JavaScript 335
gplot 436
GPU 82
groupadd 679
Gyrosensor 770
Gyroskopsensor
  MPU-6050 Siehe MPU-6050

H

H.264 82
Harmony-Hub
  Inbetriebnahme 728
Haus- und Wohnungstür 185
Hausautomation 71
  Sicherheit 49
Hausautomation, Fertigprodukte 51
Haushaltselektronik steuern 681
Haushaltsgeräteüberwachung, mit dem Arduino 770
Haustür 259
  Arduino 185, 379
  Arduino-Mobilfunksteuerung 381
  elektronischer Schließzylinder 261
  elektronisches Schloss 260
  Fingerscanner anschließen 276
  Mehrfachverriegelung 265
  Mobilfunkanschluss 339
  Motorschloss 260
  öffnen per SMS 368
  ohne Schlüssel 259
  Raspberry Pi 185
  Steuereinheit anschließen 271
  und GSM kombinieren 368
hcitool 120
Heimnetzwerk 17
Heizkörperthermostate 490
Heizungsreglereinheit
  FHEM 497
  mit Raspberry Pi 495
Heizungsverbrauch messen 487
HomeKit
  Elgato Eve 644
HomeMatic 72, 495
  Komponenten 97
  Protokoll 74
home-Protokoll 292
HT 788
HTTP 788
HTTP-Protokoll 100
https 101
HTTPS-Protokoll 100
HTTP-Zugriff, DoorBird 218
Hue 556
Hue-Alternative 590
Hue-App 562
Hue-Bridge 561
Hue-Lampen, ohne App 570
Hue-Starterpack 558
Hue-Steuerung 572
Hutschiene 439

I

I2C 788
I2C-Fertiglösung 660
i2c-tools 669
IANA 19
id_rsa.pub 681
IFTTT 632
IMU 770
  DOF 770
  Freiheitsgrad 770
Inbetriebnahme
  root oder pi 317
  Welcome-Kamera 245

Inertial Measurement Unit Siehe IMU
Inertialeinheit Siehe IMU
Infrarot 128
init.d 698
inittab 76
Internet
  Sicherheit 52
Internet der Dinge Siehe IoT
Internet of Things Siehe IoT
IoT 29, 49, 788
  Benachrichtigungsprotokoll 29
  Gefahren 49
  Sensor 29
  Sicherheit 49
IoT-Geräte
  Sicherheit bei der Hausautomation 59
IoT-Geräte
  Benutzerkonten absichern 69
  Datenschutz im Smartphone-Umfeld 56
  eigenes WLAN-Netzwerk 63
  E-Mail-Wegwerfadresse 57
  im Netz schützen 60
  IP-Adresse/MAC-Adresse finden 64
  sichere WLAN-Routerkonfiguration 64
IP-Adresse 20, 22
iPhone 632
IrDA 128
IR-Infrarotfilter 85

J
Java 558
JavaScript Object Notation 584
JeeLink-Adapter 466
JeeLink-Arduino 464
JSON 584
JSON- und Requests-Bibliothek 549

K
Kameramodul 79
Kamerasensor 79
Klingelbenachrichtigung, per E-Mail-Nachricht 232
KNX 72
Kommandozeile 82
Kondensator 193
Konsole 132
Konsolen-Basics 783
Körperwaage 499
  als Thermometer 499
  Heizungssteuerung 508
  mit Withings-App 502

L
L 788
l2ping 187, 676
Lampstealer 558
LAN 788
LAN/WLAN-Router 18, 117
LAN-Adapter 72
LAN-Router 18
LED abschalten 83
LED-Klebestreifen 557
LED-Lampen 592
LED-Lampen, Playbulb 604
LED-Leuchtmittel 555
  Hue-Bridge und -Lampen 555
LED-Stripes 566
libusb 421
Lichteffekte 608
Lichtsteuerung 608
localhost 113
Logdatei, im Eigenbau 307
Logitech-Fernbedienungen 718
LON 72
Lötkolben 454
Lötzinn 454
lsusb 77
Luftdruck 511
Luftfeuchtigkeit 511, 660

M

MAC-Adresse 61
  ermitteln 61
Mailversand, über Python-Funktion 237
Maxim Integrated 421
Mediacenter 108
Mehrfachverriegelung, mit Motorschloss 266
Message Queue Telemetry Transport
  Siehe MQTT
Messen 119
Microsoft SkyDrive 108
Miele@mobile 748, 756, 757, 760
Miele@mobile-App 756
Miele-Gateway, Grundinstallation 751
Miele-Geräte
  mit dem Heimnetzwerk 749
  mit Kommunikationsmodul 749
minicom 351
Mobilfunkanschluss 339
Mobilfunkschnittstelle 350
modprobe 427
Mosquitto 29
  Inbetriebnahme 32
  Installation 30
Motorschloss, mit Steuereinheit verbinden 279
Mountpoint 110
MPU-6050 772
  Inbetriebnahme 772
MPU-6050-Gyrosensor, Experimente 773
MQTT 29, 788
  Arduino 29, 34
  Arduino-Bibliothek 38
  Broker 29
  Client 29
  Publisher 30
  Raspberry Pi 29, 43
  Subscriber 30

N

N 788
NAS 681, 682
NAS-Server 682, 690
  Shutdown-Skript 688
NAT 20, 788
nestprotect2log.php 160
netcat 649
netstat 434
Netzwerkdrucker 692
Netzwerkspeicher Siehe NAS
newgrp 679
NFS 144
Notify 445
NTP 381, 788
Nummernkreis 21

O

OneWire 119
OpenJDK 558
openssl 101
owfs 429
OWFS 424
OWFS-Dateisystem 424
owhttpd 429
ownCloud 108, 109
  Administratorkonto 116
  Android 116
  Browser 116
  Installation 115
  iOS 116
  Konfiguration 115
owserver 429

P

PasswordAuthentication 686
Patch-Kabel 420
PCA 301 465
PE 788

PEN 788
Perl CPAN 584
Pflanzenüberwachung 544
App Parrot Flower Power 544
JSON- und Requests-Bibliothek 549
Raspberry Pi 548
SSL 550
Phasenprüfer 274
Philips 556
Philips Hue 557
Philips-Hue-App 562
PHP installieren 328
php.ini 114
PHP5 113
PHP-Curl 151
Physical Web 52
Pi 788
Pi NoIR 85
PiFace 86
PIR 129
PIR-Modul 130
Portfreigabe
FRITZ!Box 27
Konfiguration 27
Portscanner 294
post_max_size 114
poweroff 685
Prellen 183
Pretzelboard 164, 770
Inbetriebnahme 165
printers.conf 696
Public Key 681
Python, GPIO-Bibliothek 786
pywss 518

R

Raspberry Pi
Akku 144
ARM-Prozessor 119
Briefkastenalarm 136
Fingerscannerüberwachung 294
Funksteckdosen 448
Garage öffnen 673
Heimnetz 684
Heizungssteuerung 479
IC ULN2803A 451
im Verteilerkasten 313
im Vogelhaus 140
Modell B 88
MQTT 43
ownCloud 108
per SMS steuern 128
Portscanner mit nmap 294
root-Werkzeuge 677
Shell-Fotografie 199
SMS-Gateway 120
Stromrechnung senken 436
Stromverbrauch 415
Temperatursteuerung 479
Tür-Motorschlosssteuerung 327
Webserver einrichten 227
Webserverschnittstelle 174
Wetterstation 510
Wettervorhersage 520
Raspberry Pi 3
Bluetooth 632
WLAN 632
Raspberry-Pi-Basisschutz 69
Raspberry-Pi-Kamera 79
Pi NoIR 85
Programmierung 83
Raspbian 314
raspi-config 81
raspistill 82, 84
raspivid 82, 84
Rauchmelder 146
Ablaufdatum 149
mit E-Mail-Benachrichtigung 154
Nest Protect 146
per App 147

Rauchmelderdaten, mit PHP auslesen 152
Rauchmelderüberwachung per Smartphone 147
Rauchmelderzugriff, per Google-Nest-API 151
Redundanz, Sicherheit 50
Reed-Schalter 137
Regeln 119
Repository-Verwaltung 84
RJ11 420
RJ11-Standard 420
RJ12 420
RJ12-Standard 420
RJ45 420
Router 21
rpc 687
RPI 788
RS485 277
Runlevel 134
Rutenbeck 438
  Energy Manager 462
  Schaltlösungen 438
  Steckdosen 438
  TCR IP 4 438
Rx-Verbindungen 356

S

S0 419
S0-Zählermodul 432
S1 788
Samba 144, 687
Samba-Konfiguration 324
Samba-Testprogramm 325
SAT-TV-Box, FRITZ!Box 712
Schaltschrank
  Verkabelung 274
Schließmethoden, mehrere 50
Schlüssel ersetzen 259
Schutz vor manipulierten Webseiten 53
sendEmail 197
sendip 650
Sensor
  Accelerometer 770
  Barometer 770
  Beschleunigungssensor 770
  Bewegungssensor 770
  Gyrosensor 770
  IMU 770
  Kompass 770
  Magnetometer 770
  PIR 770
  Temperatursensor 770
serial-Bibliothek 363
setcap 679
SHT21 666
Sicherheitseinstellungen 55
SIM900
  Firmware-Update 341
  GPRS/GSM-Modul 346
SIM900-Mobilfunkmodem, mit dem Arduino verbinden 384
SIM900-Modem 347
  richtig verkabeln 340
sim900-sende-sms.py 364
SIM-Karte 120
Siri 632, 633
  Heizung 632
  Hue 632
  Hue 2 632
  Licht 632
  Steckdosen 632
Skript
  curute.sh 443
  HPLJ2100 695
  piri.sh 132
  piriblink.sh 139
  piriblinkcam.sh 142
  sim900-sende-sms.py 364
  smscheck.sh 129
  udprute.py 656

udprute.sh 652
udprute2.sh 658
udpswitch.sh 653
Smart Home 71, 415
Smartphone
als Fernbedienung 700
Hausgeräte im Smart Home 748
smbpasswd-Datei 324
SML 417
SML-Frames 417
SMS 373
PDU 358
Textversand 358
SMS-Gateway 120
SMS-Nachrichtenversand 378, 389
SMS-Versand 354, 358
mit Python 363
socat 649
Social-Media-Plattformen, Datenschutz 57
socket 696
SoftwareSerial 389
Buffer-Overflow 389
Empfangspuffer anpassen 389
SoftwareSerial.h 389
Solaranlage 432
SourceForge 84
SPI 788
sqlite 113
SSH 682
SSID 318, 592
SSL 788
Standortbestimmung 188
Steckdosen 437, 458
mit Python schalten 457
Steckdosen-Modding 446
Steuern 119
StrictHostKeyChecking 686
Stromrechnung senken 436
Stromverbrauch 416
Stromzähler 416, 419, 432
FHEM-Konfiguration 433
sudo 70, 118
SUID-Bit 652
Surfen, sicheres 52

T

TAN 128
TC IP 1 462
TC IP 1 WLAN 657
TC IP 4 LAN 444
TCP/IP 18
Netzwerk 61
TCP/IP-Protokoll 18
TCR IP 4 439
Temperatur 511, 660
Temperaturfühler 424
Temperaturmessung 670
Temperatursensor 483
testparm 325
TLS 788
tracert 62
trigger 699
Trockner 463
ttyAMA0 76
Tür-/Fensterkontakt 495
Türklingel 193
Türklingel DoorBird 203
Türklingelbenachrichtigung 193
Türrelais, über GPIO-Pin 332
Türspion 83
TV-Boxen 700
Hausautomation 710
Linux-kompatibel 703
per Computer und App 706
VU+ DUO2 702
Tx-Verbindungen 356

U

UDP 649
UDP-Datenpaket, mit Python 296

UDP-Konverter 289
Überwachung 289
UDP-Datenpaket 292
UDP-Steuerung, mit Python 656
uname 81
upload_max_filesize 114
USB-Festplatte 111
USB-Webcam 141
USB-Wetterstation 512
usermod 679
UUID 112

V

VCC 788
VDD 788
Verbrauchsanzeige 415
Versionsstand 81
visudo 677
Vogelhaus-Projekt
Akku 144
Einbaukomponenten 145

W

Waschmaschine 463
Waschmaschinenüberwachung 770
Watchdog 128
Webbrowserlücken finden 53
Weinkühlschrank 659
Welcome-Kamera
Gesichtserkennung 252
Python 253
Speicherkarte tauschen 250
Wetterstation 510
mit FHEM 543
Netatmo 526
Netatmo-Wetter-App 534
Python-API einrichten 540
Python-Schnittstelle im Eigenbau 536
Vorbereitung 527
Wettervorhersage, Template 520
wget 112
Wi-Fi-Bridge 592
Wi-Fi-Lampen
mit Raspberry Pi 597
mit Smartphone 594
Python-Modul 600
Windstärke 511
WiringPi 88
WiringPi, Pinbelegung 90
WiringPi-API 456
Python 135
Withings-Waage, mit FHEM koppeln 508
WLAN 128, 318, 788
Schnelleinrichtung 318
SSID 318
WLAN-Lampen 592
WLAN-Lampen, Inbetriebnahme 628
WLAN-LED-Lampen 590
WLAN-Schalter 608
Schaltung und GPIO-Pins 614
Wohnungstür
Arduino 185
Raspberry Pi 185
WZ 788

Z

Zählermodul 431
ZigBee Light Link 556

E. F. Engelhardt

SMART HOME MANIFEST

Hausautomation und Heimvernetzung für Maker

Kaffeemaschine, Waschmaschine und Fernseher mit dem Router verbinden, und schon hat man mehr Komfort? Ganz so einfach ist es nicht, denn eine Internetverbindung alleine bringt erst mal nicht viel mehr als Updates und ungebetene Datenvoyeure. Aber mit etwas Planung und Zeit sowie einem der vielen Minicomputer können Sie Ihr eigenes Smart Home aufbauen und dabei Geld und Energiekosten sparen.

Bei der Einrichtung eines Smart Home geht es darum, die Geräte der unterschiedlichen Hersteller unter einen Hut zu bekommen. Genau hier setzt dieses Buch an. Ob Alarm- oder Bewegungsmelder, Waschmaschine oder Fingerabdruckscanner: Der Autor hat sehr viele Geräte in seinem Haus verbaut, vernetzt, programmiert und nebenbei die aufkommenden Schwierigkeiten gelöst und für Sie anschaulich dokumentiert. Wer keine fertigen Bausteine nutzen will, der kann auf Arduino™, ESP8266 oder den Raspberry Pi setzen.

Mit viel Quellcode erfahren Sie, wie Sie selbst ein Smart-Home aufbauen, mit dem Smartphone steuern und damit meist mehr Möglichkeiten haben als mit fertigen Geräten. Denn fertige Geräte bringen häufig Inkompatibilitäten mit, die man in seinem eigenen Smart Home ganz sicher nicht braucht.

QUELLCODES DER PROJEKTE STEHEN ZUM DOWNLOAD BEREIT.

Aus dem Inhalt:

- Smart-Home-Bausteine
- MQTT und IoT
- Mehr Sicherheit für Hausautomation
- Smart-Home-Zentrale im Eigenbau
- FHEM: die zentrale Anlaufstelle
- Alarm und Bewegungsmelder
- Raspberry Pi mit SMS-Nachrichten steuern
- Briefkastenalarm mit Benachrichtigung
- Arduino und Raspberry Pi als Türwächter
- Türklingelbenachrichtigung mit Foto
- DoorBird-Modding mit dem Raspberry Pi
- Bild-/Videoüberwachung zu Hause
- Elektronisches Schloss: Lösungen für die Haus- oder Wohnungstür
- Fingerscannerüberwachung mit dem Raspberry Pi
- Mobilfunkanschluss für die Haustür
- Netzwerk-Shield für UDP, Mail, NTP und Logging
- Energiekosten fest im Griff
- Temperatur, Klima und Heizung
- ESP8266 mit Gyroskop
- und vieles mehr …

SCHRITT FÜR SCHRITT ZUM SICHEREN VERNETZTEN HEIM: STEUERN SIE MIT DEM SMARTPHONE UND EINEM MINICOMPUTER IHR HEIM, GEWINNEN SIE DAMIT ZEIT UND SPAREN SIE GELD.

9 783645 605885

39,95 EUR [D]
ISBN 978-3-645-60588-5

Besuchen Sie unsere Website
www.franzis.de

FRANZIS