



Heimautomation

mit Arduino, ESP8266 und Raspberry Pi

Das eigene Heim als Smart Home für
Heimwerker, Bastler und Maker

Inhaltsverzeichnis

	Einleitung	9
	Aufbau des Buches	9
	Mehr Informationen	10
	Danksagung	11
1	Smarthome-Hardware	13
1.1	Arduino	13
1.1.1	Arduino als Sensor- und Aktormodul	14
1.1.2	Arduino-Boards	14
1.1.3	Entwicklungsumgebung IDE	20
1.1.4	Programmierung, Programmstruktur	23
1.1.5	Praxisbeispiel: Temperaturmesser mit NTC und LED	23
1.1.6	Bibliotheken	28
1.1.7	Shields	32
1.1.8	Arduino im Miniaturformat	36
1.1.9	Arduino im Batteriebetrieb	42
1.2	Raspberry Pi	43
1.2.1	Minimal-Anforderungen	43
1.2.2	Raspberry-Pi-Boards	44
1.2.3	Installation	45
1.2.4	Remote-Zugriff	53
1.2.5	Schnittstellen zur Außenwelt	59
1.3	IoT- und Smarthome-Infrastruktur	63
2	Internet-Connectivity	65
2.1	Ethernet-Shield	65
2.2	WiFi-Verbindung	67
2.3	Arduino als Webclient	67
2.4	Arduino als Webserver	70
3	ESP8266	75
3.1	ESP-Module	75
3.1.1	ESP-01	75
3.1.2	ESP-12	76
3.2	Integration in Arduino-IDE	79

3.3	ESP8266-Boards	82
3.3.1	Wemos D1	82
3.3.2	Wemos D1 Mini	83
3.3.3	NodeMCU	87
3.4	Praxisbeispiel: Blink	88
3.5	WiFi mit ESP8266	90
3.5.1	WiFi-Bibliothek für ESP8266	90
3.6	Praxisbeispiel: Wemos-Webclient	92
3.7	Praxisbeispiel: Webclient mit Sensordaten	95
3.8	Praxisbeispiel: Webclient mit HTTPS	96
3.9	Firmware Tasmota	100
3.9.1	Funktionen	102
3.9.2	Installation Tasmota	102
3.10	Praxisbeispiel: Tasmota mit Tasmotizer	104
3.11	Praxistest: Tasmota schaltet Ausgang	111
3.12	Praxisbeispiel: Sonoff-Schaltmodule	112
4	Protokolle	119
4.1	HTTP	119
4.2	MQTT	123
5	Arduino als MQTT-Client	129
5.1	PubSubClient-Bibliothek	129
5.2	MQTT Publish mit Arduino	130
5.3	MQTT Subscribe mit Arduino	137
5.4	MQTT Publish und Subscribe mit ESP8266	140
5.5	MQTT-Topics organisieren	145
5.6	Praxisbeispiel: Sensordaten senden	146
6	MQTT und Node-Red mit Raspberry Pi	151
6.1	Raspberry Pi als Schaltzentrale	151
6.2	Mosquitto als MQTT-Broker	152
6.3	Node-Red	153
6.4	Flows mit Node-Red	159
6.5	MQTT mit Node-Red	166
6.6	Node-Red-Dashboard	169
6.7	Praxisbeispiel: Anzeige des Node-Red-Dashboards auf mobilen Geräten	177
6.8	Praxisbeispiel: Serielle Daten von Arduino Uno empfangen	178
6.9	Praxistipp: Kompakter Arduino für Datenerfassung	189

7	Arduino als Sensor-Node	193
7.1	Praxisbeispiel: Aufbau Sensor-Node	193
7.2	Praxisbeispiel: Temperatursensor (NTC).....	196
7.3	Praxisbeispiel: Helligkeitssensor BH1750	199
7.4	Praxisbeispiel: Umweltsensor SHT31	202
7.5	Praxisbeispiel: Barometer (BME680).....	206
7.6	Praxisbeispiel: Datenübertragung mit 433-MHz-Funkmodul	215
7.7	Praxisbeispiel: RFLink-433-MHz-Gateway	226
7.8	Praxisbeispiel: ESP8266 als RF-Gateway.....	233
7.9	Praxisbeispiel: RF-Gateway mit Sonoff RF Bridge	236
8	MQTT-Anwendungen	245
8.1	Praxisbeispiel: Ausgänge von Arduino und Raspberry Pi schalten.	245
8.2	Praxisbeispiel: Fernbedienung für Fernseher	253
8.3	Praxisbeispiel: Drahtlose Klingel	261
8.4	Praxisbeispiel: 8-Kanal-Analog/Digital-Wandler über MQTT	264
8.5	Praxisbeispiel: Briefkastenwächter	275
9	Smarthome-Plattformen	287
9.1	Home Assistant	287
9.2	openHAB	301
10	IoT- und Smarthome-Projekte	303
10.1	Aquarium-Timer	303
10.2	Stromwächter	308
	10.2.1 Stromwächter mit Sonoff Pow	309
	10.2.2 Stromwächter mit Stromsensor	316
10.3	Waschmaschinenwächter	323
10.4	Gefrierschrankwächter	328
10.5	RGB-Streifen (Neopixel) steuern	340
	Stücklisten	353
	Stichwortverzeichnis	361



Einleitung

Die Automatisierung der eigenen Wohnung oder des eigenen Hauses ist ein spannendes Thema für jeden praktisch veranlagten Bastler. Dank der vielen Module und Lösungen kann jeder Anwender sein »Smart Home« individuell und nach eigenen Wünschen aufbauen.

Mit Arduino, ESP8266 und Raspberry Pi können Sie kostengünstig einzelne Lösungen realisieren: Sei es die Raumüberwachung mittels eines Sensor-Netzwerks oder die Lichtsteuerung und deren Visualisierung mit Raspberry Pi und Node-Red.

Eine Schaltzentrale mit Raspberry Pi und Standardschnittstellen wie MQTT erlauben die einfache und offene Integration von Selbstbau-Modulen wie auch fertigen, kommerziellen Modulen und Anwendungen.

Dieses Buch richtet sich an Bastler und Maker, die bereits etwas Erfahrung mit Arduino und Raspberry Pi gesammelt haben und nun praktische Anwendungen in ihrem Heim aufbauen möchten.

Aufbau des Buches

Dieses Buch ist so aufgebaut, dass Sie zuerst Grundlagen über das Arduino-Board, den ESP8266 und den Raspberry Pi lernen. Anschließend werden in verschiedenen Themenkapiteln praktische Projekte aufgebaut und in den nachfolgenden Anwendungskapiteln 8 bis 10 realisiert.

In **Kapitel 1** wird die im Buch verwendete Hardware mit Arduino und Raspberry Pi vorgestellt und in Betrieb genommen.

Die Verbindung dieser Microcontroller-Boards mit dem heimischen Netzwerk oder WLAN wird in **Kapitel 2** erklärt.

Die sehr verbreiteten Module der ESP8266-Reihe werden in **Kapitel 3** in Betrieb genommen. Mit der Installation der bekannten Firmware Tasmota haben Sie eine optimale Basis für die Projekte in den weiteren Kapiteln.

In **Kapitel 4** werden die verbreiteten Protokolle HTTP und MQTT vorgestellt.

Ein Arduino, der über MQTT ins Netzwerk integriert ist, bietet eine einfache Hardware als Sensor- und Aktormodul in der Heimautomation. In **Kapitel 5** nutzt das Arduino-Board dabei die weitverbreitete `PubSubClient`-Bibliothek.

In **Kapitel 6** wird der Minicomputer Raspberry Pi als Schaltzentrale aufgebaut. Die zentrale Anwendung dabei ist die webbasierte Entwicklungsumgebung Node-Red.

In jeder Heimautomation fühlen Sensoren die Umwelt. In **Kapitel 7** werden verschiedene Sensoren eingesetzt, um Zustände im Heim zu erfassen. Mittels verschiedener Gateway-Lösungen können Sensordaten empfangen und verarbeitet werden.

Das **Kapitel 8** beschreibt praktische MQTT-Anwendungen, die man über Node-Red schalten und verwalten kann. Eine drahtlose Fernbedienung erlaubt die Ansteuerung des Fernsehers über Tablet oder Smartphone. Weiter werden analoge Daten eingelesen, die Klingel an der Haustür vernetzt und die Überwachung des Briefkastens ins heimische Netz integriert.

In **Kapitel 9** werden Heimautomations-Lösungen vorgestellt. In einfachen Schritten kann die Anwendung Home Assistant als Basis für eine Heimautomation eingerichtet und konfiguriert werden.

Weitere praktische Projekte wie ein webbasierter Aquarium-Timer, ein Stromwächter zur Energie-Überwachung oder die Temperatur-Überwachung des Gefrierschranks werden in **Kapitel 10** beschrieben.

Mehr Informationen

Weitere Informationen zu den Heimautomations-Projekten im Buch sind auf meiner Website erhältlich:

<https://555circuitslab.com>

Im Downloadbereich finden Sie alle Beispielskripte, 3D-Vorlagen, Ergänzungen und Erweiterungen.

Bei Anmerkungen und Anregungen können Sie mich gerne per E-Mail oder über Twitter kontaktieren.

E-Mail: maker@555circuitslab.com

Twitter: <https://twitter.com/arduinopraxis>

Weiterführende Informationen zum Thema Arduino und Sensoren und laufend neue Projekte beschreibe ich in meinem Arduino-Blog.

<http://arduino-praxis.ch>

Auf der Verlags-Website finden Sie Details zu meinen bisherigen Buchprojekten:

Arduino Praxiseinstieg

<https://mitp.de/0054>

Sensoren im Einsatz mit Arduino

<https://mitp.de/150>

Danksagung

Ich möchte mich ganz herzlich bei meiner Familie, meiner Frau Aga und meinen Jungs Tim und Nik bedanken, dass sie mir die Zeit und den Freiraum für dieses Buch-Projekt gewährt haben.

Ein großer Dank geht auch wieder an meine Lektorin Sabine Schulz vom mitp-Verlag. Es war wieder eine sehr angenehme und erfolgreiche Zusammenarbeit.

Dieses Buch widme ich meinem Vater Bernhard.

Im Februar 2021

Thomas Brühlmann

MQTT und Node-Red mit Raspberry Pi

In diesem Kapitel wird der Raspberry Pi als Schaltzentrale für das IoT-System oder Smarthome aufgebaut.

Die Schaltzentrale wird Daten von verschiedenen Quellen empfangen und verarbeiten. Die zentralen Komponenten dieser Schaltzentrale sind der MQTT-Broker und die grafische Oberfläche Node-Red.

6.1 Raspberry Pi als Schaltzentrale

Der Raspberry Pi als Schaltzentrale für unser System wurde bereits in Kapitel 1 kurz vorgestellt.

In Abbildung 6.1 ist nochmals das Prinzip der Infrastruktur abgebildet.

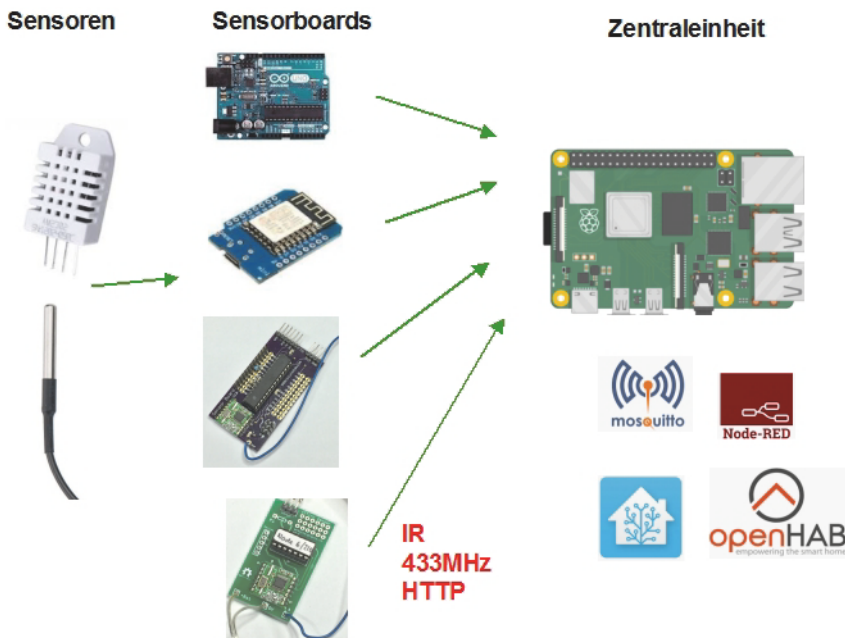


Abb. 6.1: Raspberry Pi als Schaltzentrale

Der Raspberry Pi ist die Zentraleinheit und verarbeitet über seine Schnittstellen die empfangenen Daten und löst Aktionen aus.

Der in Abschnitt 1.2 installierte Raspberry Pi mit dem grafischen Betriebssystem und dem in Kapitel 4 installierten MQTT-Broker ist eine gute Basis für das zukünftige System.

Was bisher noch nicht erwähnt wurde, ist die grafische Entwicklungsumgebung Node-Red. Diese erkläre ich später noch im Detail. Node-Red ist auch bereits installiert und kann über das Startmenü des Raspberry Pi OS aufgerufen werden.

Über die USB-Schnittstellen können externe Arduino-Module direkt serielle Daten liefern. In einem Praxisbeispiel in diesem Kapitel zeige ich diese Lösung.

Mittels MQTT-Schnittstelle können Sensor-Boards, basierend auf dem ESP8266, drahtlos Daten und Informationen an die Zentraleinheit liefern.

Das Raspberry-Board ist ein richtiger Computer und kann somit direkt mit einem Bildschirm zur Anzeige von Daten und zur Dateneingabe eingesetzt werden.

6.2 Mosquitto als MQTT-Broker

Der Mosquitto-MQTT-Broker ist eine kompakte Serveranwendung und wurde bereits in Kapitel 4 installiert und überprüft.

Ebenfalls in Kapitel 4 wurden die ersten Topics über die Terminal-Konsole abonniert und mit Daten befüllt.

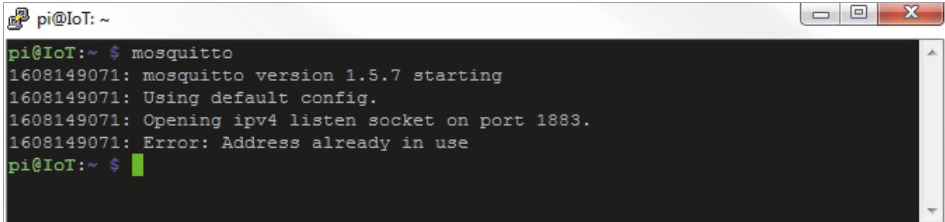
Für den Betrieb des MQTT-Brokers ist grundsätzlich kein User-Interface nötig.

Informationen zum Broker können Sie über das Terminal abfragen.

Mit dem Befehl

```
mosquitto
```

erhalten Sie die Version des installierten MQTT-Brokers (Abbildung 6.2).



```
pi@IoT: ~
pi@IoT:~ $ mosquitto
1608149071: mosquitto version 1.5.7 starting
1608149071: Using default config.
1608149071: Opening ipv4 listen socket on port 1883.
1608149071: Error: Address already in use
pi@IoT:~ $
```

Abb. 6.2: Installierter Mosquitto-Broker

Da der MQTT-Broker auf dem Raspberry Pi läuft, ist entsprechend auch die zugehörige IP-Adresse bekannt. Sein Standard-Port ist 1883. Beide Werte, IP und Port, werden später für die Konfiguration des MQTT-Brokers in Node-Red noch benötigt.

6.3 Node-Red

Node-Red ist ein grafisches Werkzeug, um sogenannte Flows zu erstellen. Durch das Zusammensetzen von einzelnen Elementen, als Nodes bezeichnet, kann man grafische Informationsflüsse realisieren.

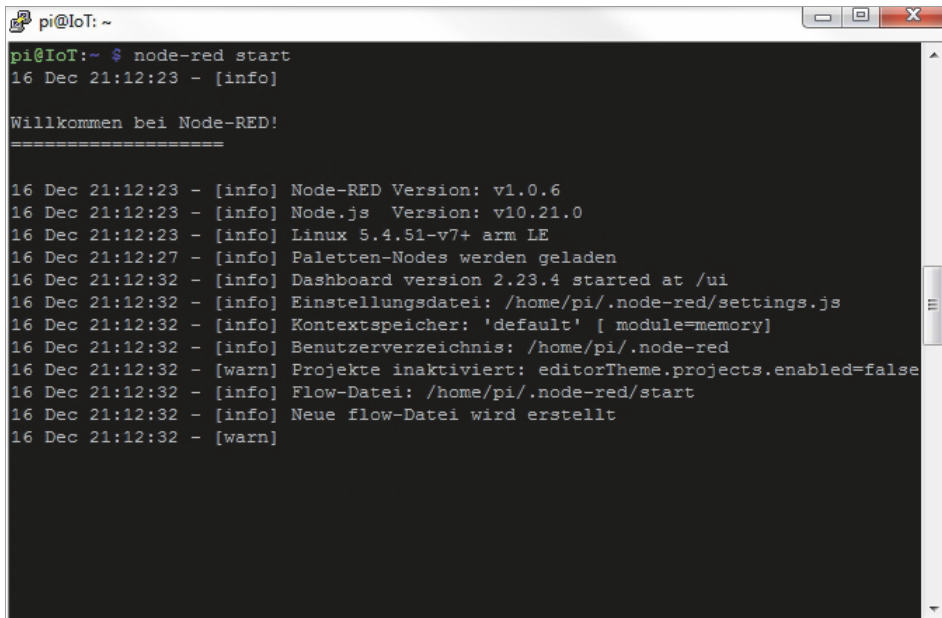
In der aktuellen Version des Raspberry Pi OS ist Node-Red bereits installiert.

Start von Node-Red

Das Frontend von Node-Red wird über den Browser aufgerufen. Vor dem Start im Browser muss der Node-Red-Server über ein Terminalfenster auf dem Raspberry Pi mit folgendem Befehl gestartet werden.

```
node-red start
```

Nach Start im Terminal meldet sich das System (Abbildung 6.3).



```
pi@IoT: ~  
pi@IoT:~$ node-red start  
16 Dec 21:12:23 - [info]  
  
Willkommen bei Node-RED!  
=====
```

```
16 Dec 21:12:23 - [info] Node-RED Version: v1.0.6  
16 Dec 21:12:23 - [info] Node.js Version: v10.21.0  
16 Dec 21:12:23 - [info] Linux 5.4.51-v7+ arm LE  
16 Dec 21:12:27 - [info] Paletten-Nodes werden geladen  
16 Dec 21:12:32 - [info] Dashboard version 2.23.4 started at /ui  
16 Dec 21:12:32 - [info] Einstellungsdatei: /home/pi/.node-red/settings.js  
16 Dec 21:12:32 - [info] Kontextspeicher: 'default' [ module=memory]  
16 Dec 21:12:32 - [info] Benutzerverzeichnis: /home/pi/.node-red  
16 Dec 21:12:32 - [warn] Projekte inaktiviert: editorTheme.projects.enabled=false  
16 Dec 21:12:32 - [info] Flow-Datei: /home/pi/.node-red/start  
16 Dec 21:12:32 - [info] Neue flow-Datei wird erstellt  
16 Dec 21:12:32 - [warn]
```

Abb. 6.3: Node-Red-Start

Nach dem Start-Befehl läuft der Node-Red-Server, bis das System gestoppt oder der Raspberry Pi neu gestartet wird.

Folgende Betriebsarten stehen für den Betrieb von Node-Red zur Verfügung.

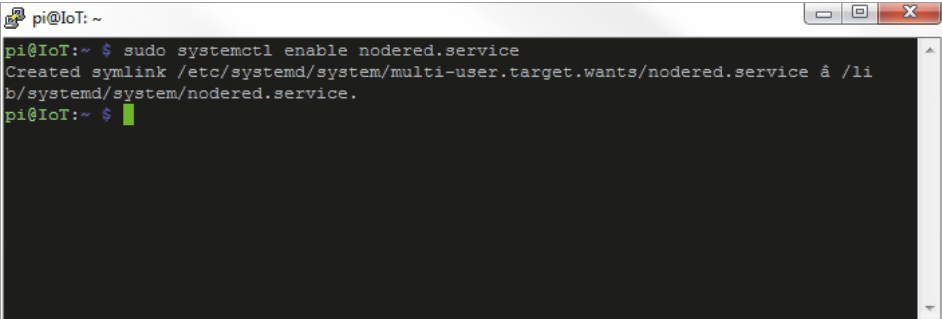
Befehl	Beschreibung
node-red-start	Starten von Node-Red
node-red-stop	Beenden von Node-Red
node-red-log	Anzeigen der Log-Dateien
sudo systemctl enable nodered.service	Aktivieren des automatischen Starts bei jedem Neustart
sudo systemctl disable nodered.service	Deaktivieren des automatischen Starts bei jedem Neustart

Tabelle 6.1: Node-Red-Betriebsarten

Der automatische Start von Node-Red erfolgt also gemäß Tabelle 6.1 mit dem Befehl.

```
sudo systemctl enable nodered.service
```

Damit ist der Start von Node-Red bei einem Neustart des Systems aktiviert (Abbildung 6.4).



```
pi@IoT: ~  
pi@IoT:~ $ sudo systemctl enable nodered.service  
Created symlink /etc/systemd/system/multi-user.target.wants/nodered.service â /lib/systemd/system/nodered.service.  
pi@IoT:~ $
```

Abb. 6.4: Node-Red für Autostart einrichten

Nach Start des Node-Red-Servers kann das Node-Red-Frontend über den Browser aufgerufen werden. Dieser Service läuft standardmäßig über den Port 1880.

`http://IP-des-Raspberry:1880`

Im Browser erscheint die Oberfläche der Anwendung (Abbildung 6.5).

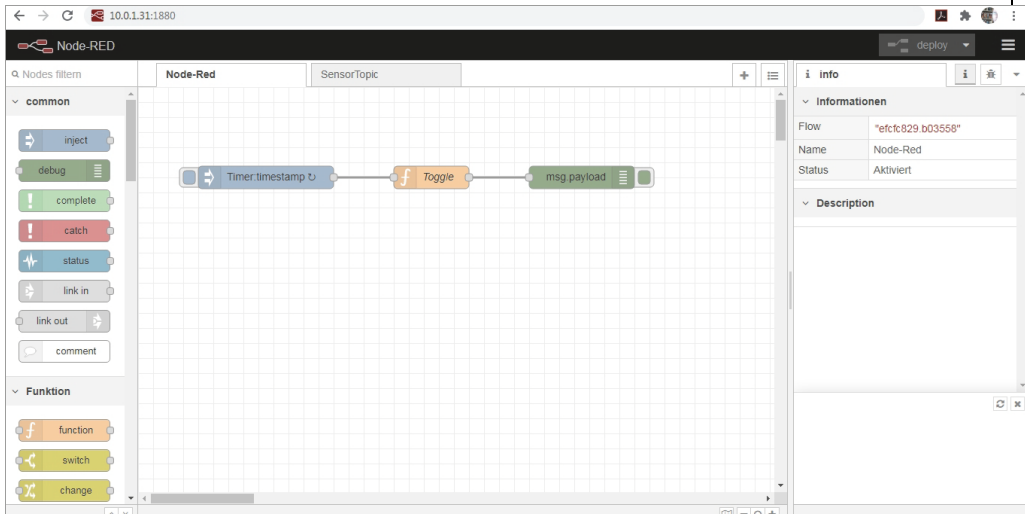


Abb. 6.5: Node-Red-Frontend

Oberfläche

In Abbildung 6.6 ist die Oberfläche von Node-Red dargestellt.

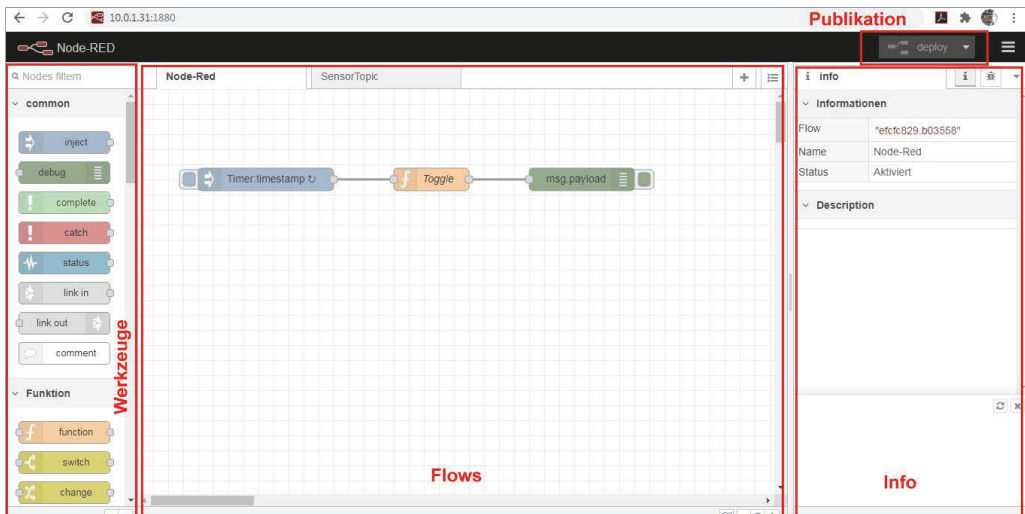


Abb. 6.6: Node-Red – Oberfläche

Das zentrale Element ist der mittlere Bereich mit den unterschiedlichen Flows. Über einzelne Tabs kann man die einzelnen Flows strukturieren.

In der linken Spalte ist die Werkzeugpalette angeordnet.

Die rechte Spalte ist für Informationen und das Debugging vorbereitet.

Oberhalb der rechten Spalte ist ein Knopf platziert, der zum Speichern der Änderungen verwendet wird. Sobald eine Änderung im Flow erfolgt, wird dieser Knopf aktiviert und mit roter Farbe markiert. Mit Klick auf diesen Knopf mit der Beschriftung `DEPLOY` werden die Änderungen gespeichert und aktiviert (Abbildung 6.7).

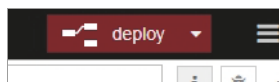


Abb. 6.7: Node-Red: Änderungen speichern mit »deploy«

In Abbildung 6.6 sehen Sie den ersten Flow-Tab, der mit `NODE-RED` betitelt ist.

Im Flow selbst sind drei Nodes sichtbar, die miteinander verknüpft sind.

Werkzengleiste

Die Werkzeugliste oder Node-Palette listet die einzelnen verfügbaren Nodes in Kategorien sortiert auf.



Abb. 6.8: Node-Red: Werkzeugpalette

Mit der Standard-Installation von Node-Red wird ein Standard-Set von Nodes mitgeliefert. Weitere Nodes können aus dem Internet geladen werden.

Über das Menü oben rechts kann die Menüstruktur der Node-Red-Verwaltung aufgerufen werden (Abbildung 6.9).

Unter dem Menüpunkt `PALETTE VERWALTEN` gelangen Sie in die Node-Verwaltung. Im Register `INSTALLIEREN` können Sie nach Nodes suchen. Im Beispiel suchen wir nach dem `Dashboard-Node` (Abbildung 6.10). Das Dashboard werden wir uns später noch im Detail anschauen.



Abb. 6.9: Node-Red: Node-Palette verwalten

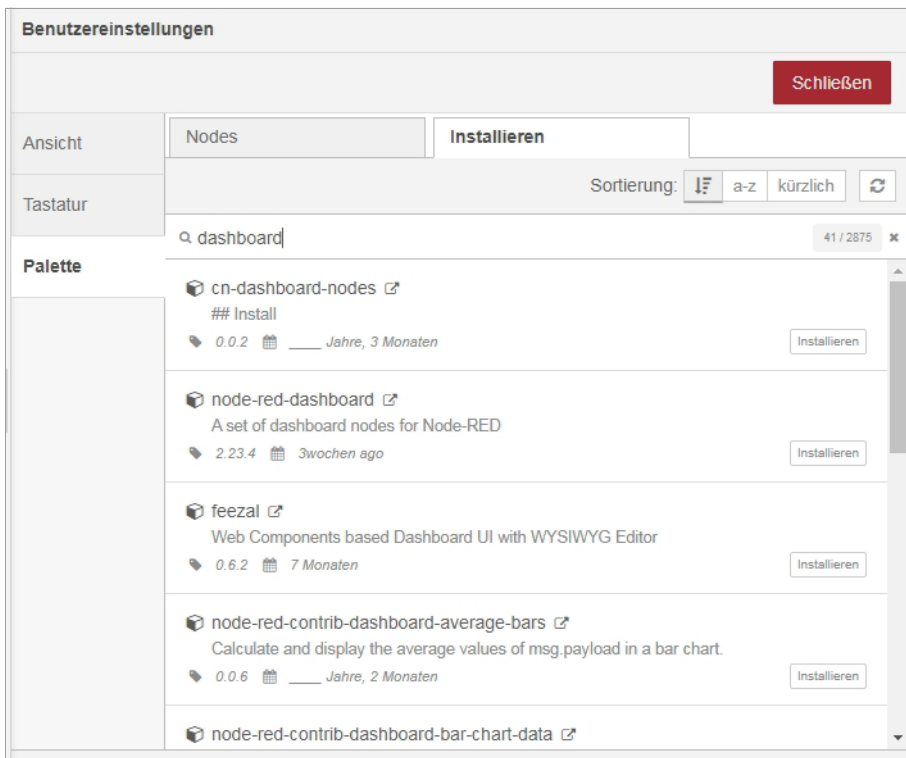


Abb. 6.10: Node-Red: verfügbare Nodes zum Installieren

Den gesuchten Node-Eintrag können Sie durch Klick auf **INSTALLIEREN** zur Installation wählen.

Nach der Bestätigung der Installation wird der gewählte Node installiert und in der Werkzeugpalette verfügbar gemacht (Abbildung 6.11).

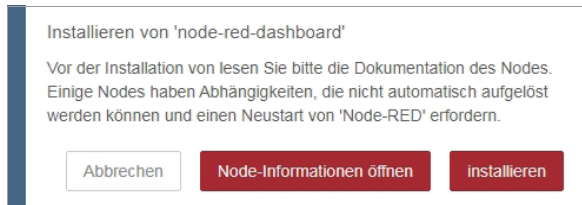


Abb. 6.11: Node-Red: Installation von Node bestätigen

Debug-Fenster

Das Debug-Fenster in der rechten Spalte beinhaltet sowohl ein Informationsregister als auch ein Debug-Register.

Im Debugging können Ausgaben, die via Debug-Node ausgegeben werden, dargestellt werden.

In Abbildung 6.12 ist die Debugging-Ausgabe für den ersten Flow dargestellt. Hier wird im Sekundentakt eine Ausgabe gemacht.



Abb. 6.12: Node-Red: Debugging-Ausgabe

Diese Debugging-Ausgabe ist sehr nützlich bei der Flow-Erstellung und zum Überwachen des Datenflusses.

6.4 Flows mit Node-Red

Ein Flow in Node-Red besteht aus mehreren Nodes, die miteinander verbunden sind (Abbildung 6.13).



Abb. 6.13: Node-Red: Flow mit Nodes

Hinter jedem Node versteckt sich eine Funktion beziehungsweise Aktion. Durch das Zusammenschalten der einzelnen Nodes kann man einfache oder komplexe Informationsflüsse realisieren.

Einen einzelnen Node können Sie mittels Drag&Drop aus der Werkzeugpalette auf den mittleren Flow-Bereich ziehen.

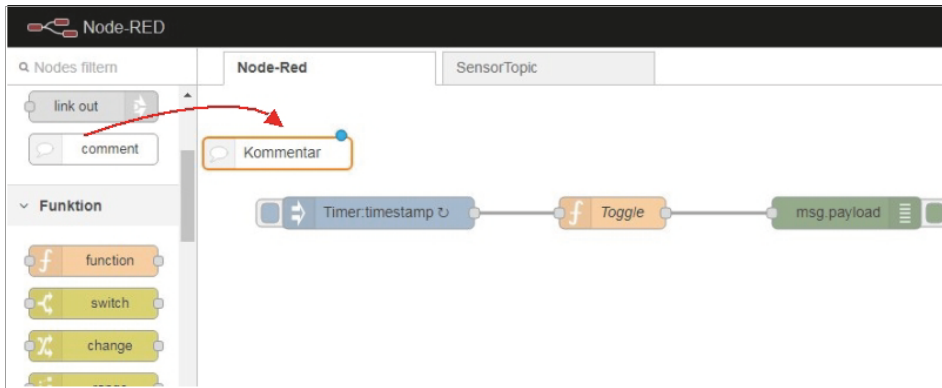


Abb. 6.14: Node-Red: Node wählen mit Drag&Drop

Im Flow-Bereich können Sie dann den Node nach Wunsch platzieren.

Der Beispiel-Node in Abbildung 6.14 ist ein Kommentar. Ein Kommentar ist nie mit einem anderen Node verbunden.

Plaziert man nun zwei einzelne Nodes, die dann im Flow zusammengehören, muss anschließend eine Verbindung geschaffen werden.



Abb. 6.15: Node-Red: Nodes ohne Verbindung

Dazu müssen die in Abbildung 6.16 rot markierten Verbindungsstellen miteinander verbunden werden.



Abb. 6.16: Node-Red: Nodes verbinden

Mit gedrückter Maustaste (meist die Taste mit dem Zeigefinger) klicken Sie auf den ersten Verbindungspunkt. Wenn Sie nun die Taste gedrückt lassen und die Maus verschieben, sehen Sie den orange Verbindungsfaden. Führen Sie den Mauszeiger nun auf den zweiten Verbindungspunkt, bis dieser orange markiert ist.



Abb. 6.17: Node-Red: Nodes verbinden

Nun können Sie die Maustaste loslassen und die Verbindung zwischen den beiden Nodes ist erstellt (Abbildung 6.18).



Abb. 6.18: Node-Red: Nodes sind verbunden.

Hinweis

Beim Erstellen von Flows sind auf den einzelnen Nodes teilweise blaue Punkte sichtbar. Die blauen Punkte geben an, dass diese Nodes noch nicht gespeichert wurden. Sobald man mittels **deploy** den Flow speichert, verschwinden die blauen Punkte.

Eigenschaften der Nodes

Mit dem Platzieren einzelner Nodes und dem Verbinden mit anderen Objekten ist es meist nicht getan. Hinter den einzelnen Nodes aus der Werkzeugpalette verstecken sich oft umfangreiche Funktionen, die über die Eigenschaften des jeweiligen Nodes konfiguriert werden können.

Mit Klick auf einen Node, im Beispiel der `timestamp`-Node, öffnet sich das Eigenschaften-Fenster und verschiedene Konfigurationsmöglichkeiten werden aufgelistet (Abbildung 6.19).

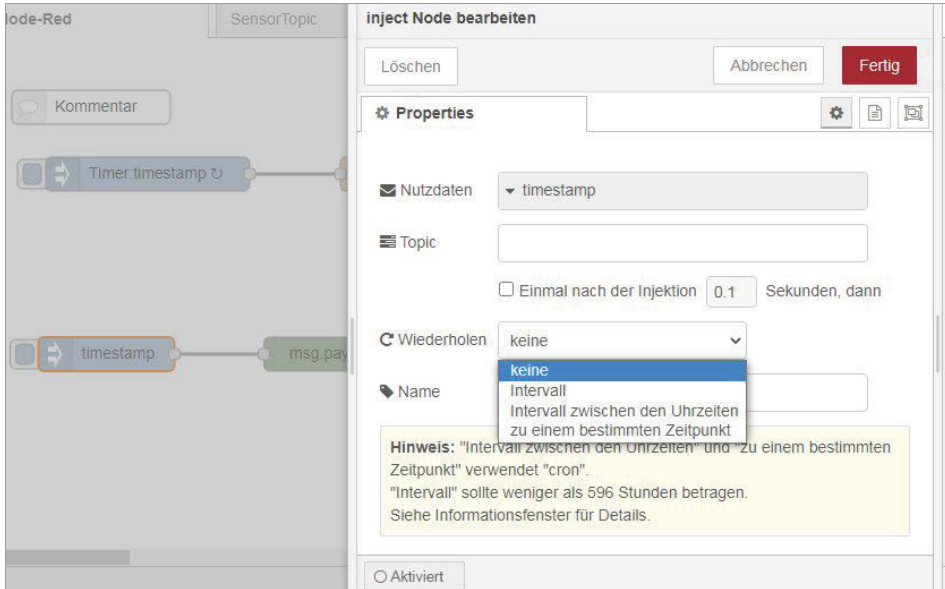


Abb. 6.19: Node-Red: Eigenschaften des Nodes

Der Node **timestamp** kann beispielsweise als Taktgeber (Intervall) mit einstellbarer Zeit verwendet werden (Abbildung 6.20).

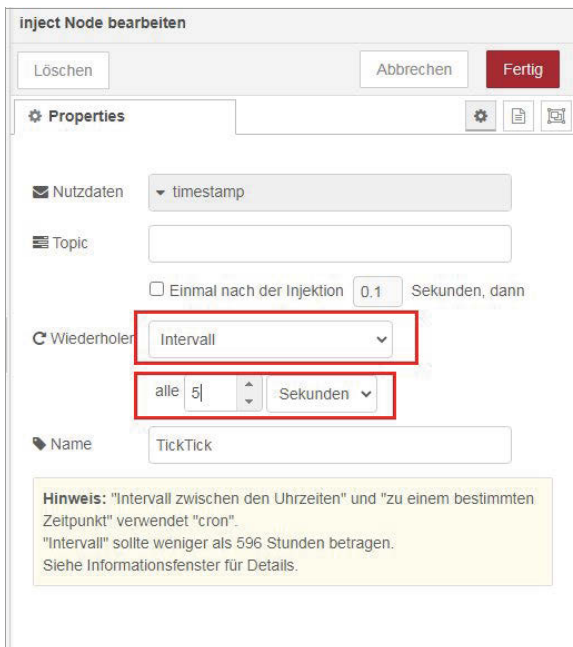


Abb. 6.20: Node-Red: Node-Eigenschaften für Intervall-Timer

Jeder Node besitzt unterschiedliche Funktionen und Eigenschaften. In der rechten Info-Spalte werden beim Klicken auf einen einzelnen Node zusätzliche Informationen dargestellt.

Mehrere Flows

Ein Flow einer Anwendung kann schnell aus vielen Nodes und entsprechend vielen Verbindungen zwischen ihnen bestehen.

Hat man verschiedene Datenflüsse, lohnt es sich, diese etwas zu strukturieren.

Ein hilfreiches Mittel ist dabei, die einzelnen Flows in einzelnen Tabs zu speichern.

Mit dem Pluszeichen am rechten Rand des Flow-Bereichs kann ein neuer Tab oder Reiter erstellt werden (Abbildung 6.21). Idealerweise gibt man dem Reiter einen sinnvollen Namen.



Abb. 6.21: Node-Red: neuen Tab oder Reiter erstellen

Flows mit JavaScript erweitern

Viele Funktionen können in Node-Red durch Zusammenstellen realisiert werden. Dabei kann der Anwender die vielen Konfigurationsmöglichkeiten der einzelnen Nodes nutzen. Informationen fließen dabei von Node zu Node. Viele Nodes besitzen dazu Ein- und Ausgänge.

Node-Red ist bekanntlich eine webbasierte Anwendung. Die Darstellung und die Funktion der einzelnen Nodes erfolgen mittels der bekannten Webtechnologien HTML und JavaScript. JavaScript als Skript-Sprache wird im Browser des Anwenders ausgeführt und ein Entwickler kann eigenen JavaScript-Code in seine Webseite einbauen.

Mit eigenem JavaScript-Code können Sie somit die Funktionalität Ihrer Flows erweitern oder anpassen. Der für solche Funktionserweiterungen vorgesehene Node ist in der Werkzeugleiste mit FUNCTION bezeichnet (Abbildung 6.22).



Abb. 6.22: Node-Red: Funktions-Node

Dieser Funktions-Node hat standardmäßig einen Eingang und einen Ausgang.

In Abbildung 6.23 ist der ganze Flow zum Toggeln dargestellt.



Abb. 6.23: Node-Red: Toggle-Flow

Nachdem Sie den Funktions-Node auf die Arbeitsfläche gezogen haben, können Sie ihn mit einem Doppelklick bearbeiten. Es öffnet sich ein Eigenschaften-Fenster, über das Sie einen Titel und Funktionscode eingeben können (Abbildung 6.24). Am unteren Ende des Fensters kann zusätzlich die Anzahl der Ausgänge angegeben werden.

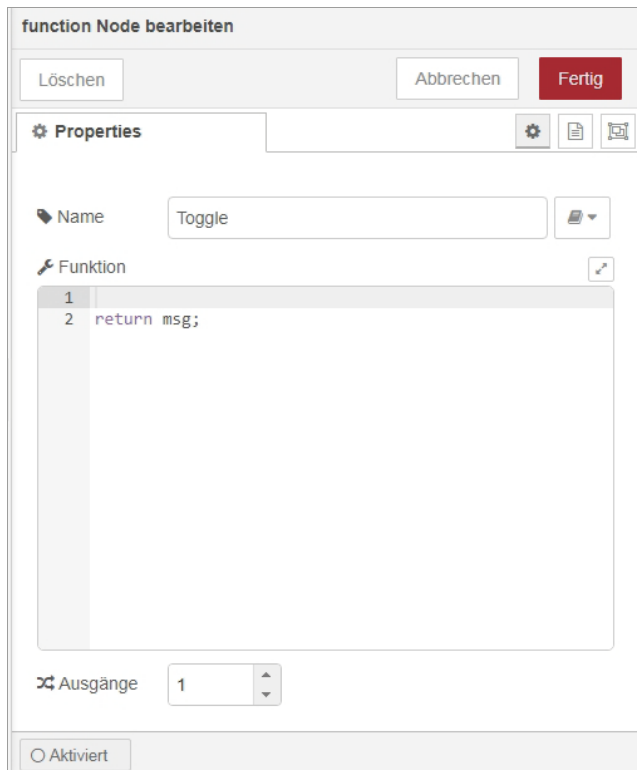


Abb. 6.24: Node-Red: Funktions-Node bearbeiten

Der Funktionsblock in diesem Beispiel wird mit TOGGLE betitelt und soll einen Input vom Inject-Node, der als Intervall läuft, auswerten und jeweils den Ausgang von 1 auf 0 und zurück schalten (Vorgang wird auch als Toggeln bezeichnet).

Der lauffähige Code ist im Codebereich eingetragen und gespeichert. (Abbildung 6.25).



Abb. 6.25: Node-Red: JavaScript-Code für Toggle

Über den Debug-Node kann nun die Ausgabe im Debug-Fenster überprüft werden. Die Ausgabe toggelt zwischen den Werten ON und OFF (Abbildung 6.26).

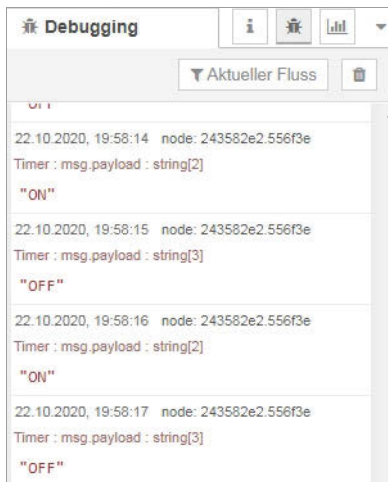


Abb. 6.26: Node-Red: Debug-Ausgabe

In diesem JavaScript-Code wird zuerst eine Variable `count` deklariert, falls diese noch nicht existiert, und ein Wert 0 zugewiesen (`smarthome_kap6_funktion_toogle.txt`):

```
var count=context.get('count') || 0;
```

Nun wird der Wert von `count` um 1 erhöht:

```
count +=1;
```

In der Variablen wird ein leerer String gespeichert:

```
var schalter ="";
```

Jetzt wird mit der Modulo-Funktion geprüft, ob die Zahl gerade ist. Dies ist der Fall beim Resultat 0.

Wenn das Resultat 0 ist, wird in der Variablen `schalter` ein String mit ON gespeichert, andernfalls ist der String-Wert OFF:

```
if (count % 2 !== 0)
{
    schalter="ON";
}
else
{
    schalter = "OFF";
}
```

Die Payload, die sogenannte Nutzlast, beinhaltet den String mit der zu übertragenden Nachricht. In diese Payload wird nun der Zählerstand gespeichert:

```
msg.payload=count;
```

Der Zählerstand wird im Context-Store der Variablen `count` zugewiesen:

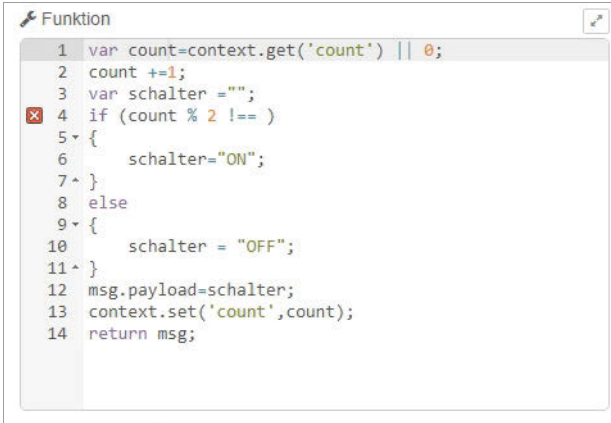
```
context.set('count',count);
```

Zum Schluss wird die Nachricht `msg` zurückgegeben und gelangt zum nächsten Node. In diesem Fall zum Ausgabe-Node:

```
return msg;
```

Das Eingabefeld für den Code ist nicht nur eine Texteingabe, sondern beinhaltet eine Code-Validierung.

Falls ein Fehler in der Syntax vorhanden ist, wird dies mit verschiedenen Zeichen ausgegeben. In Abbildung 6.27 ist die Modulo-Funktion fehlerhaft. Es fehlt die Zahl 0 für den korrekten Vergleich.



```

1 var count=context.get('count') || 0;
2 count +=1;
3 var schalter ="";
4 if (count % 2 !== )
5 {
6     schalter="ON";
7 }
8 else
9 {
10     schalter = "OFF";
11 }
12 msg.payload=schalter;
13 context.set('count',count);
14 return msg;
    
```

Abb. 6.27: Node-Red: JavaScript-Code mit Fehler

6.5 MQTT mit Node-Red

Das Einlesen und Ausgeben von MQTT-Topics ist recht simpel und kann mit den beiden Nodes `mqtt in` und `mqtt out` realisiert werden (Abbildung 6.28).

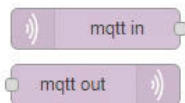


Abb. 6.28: Node-Red: MQTT-Nodes

Das Abonnieren und Ausgeben eines Topics, in unserem Fall des bereits getesteten Topics `SensorTopic`, ist in Abbildung 6.29 abgebildet.



Abb. 6.29: Node-Red: MQTT-Flow

Mit Klick auf den MQTT-Node können die Eigenschaften eingetragen werden. In diesem Fall sind das der MQTT-Server und der Name des Topics (Abbildung 6.30).

Die weiteren Einstellungen in diesem Fenster können mit den Standard-Einstellungen übernommen werden.

Das Feld QoS bezeichnet das MQTT-Feature *Quality of Service*. Damit wird die Zuverlässigkeit der Nachrichtenzustellung definiert. MQTT kennt drei Stufen, die mit 0, 1 oder 2 definiert werden. Genau diese drei Werte können im Eigenschaften-Fenster des MQTT-Servers eingetragen werden.

Die drei Stufen des QoS sind in Tabelle 6.2 beschrieben:

Quality-of-Service-Stufe	Beschreibung
0	Nachrichtenversand einmalig Keine Prüfung, ob erhalten
1	Nachricht wird versendet Empfänger bestätigt dem Sender Nachrichtenempfang
2	Nachricht wird versendet Nachricht kommt garantiert an und garantiert nur einmal

Tabelle 6.2: MQTT – Quality of Service (QoS)

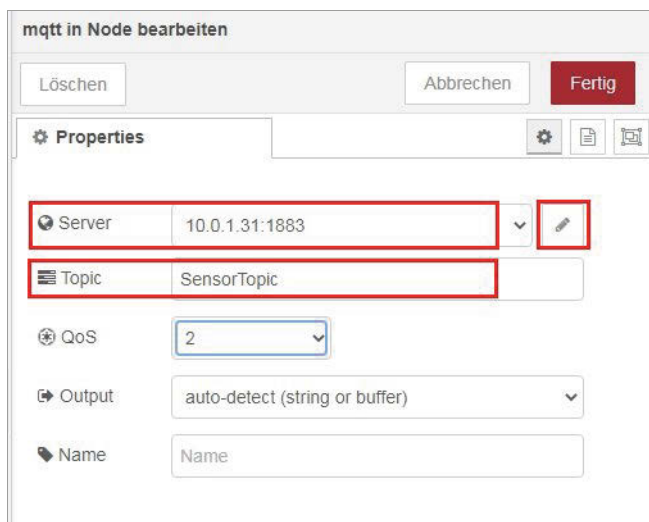


Abb. 6.30: Node-Red: MQTT-Topic

Über das Auswahlménú kann ein bestehender Server ausgewählt oder ein neuer Server erfasst werden (Abbildung 6.31).

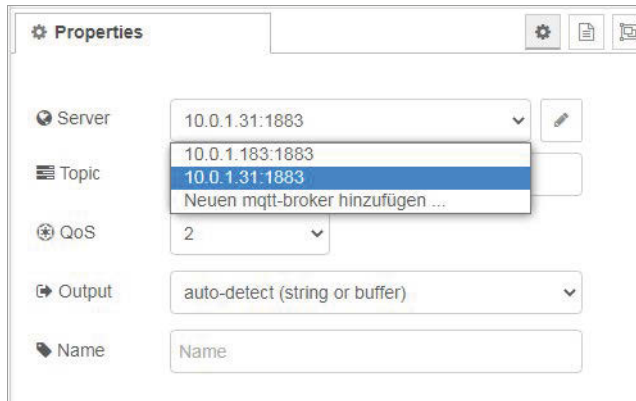


Abb. 6.31: Node-Red: MQTT-Server

Ein bestehender Server kann mittels der Editier-Funktion mit dem Stift bearbeitet werden (Abbildung 6.32).

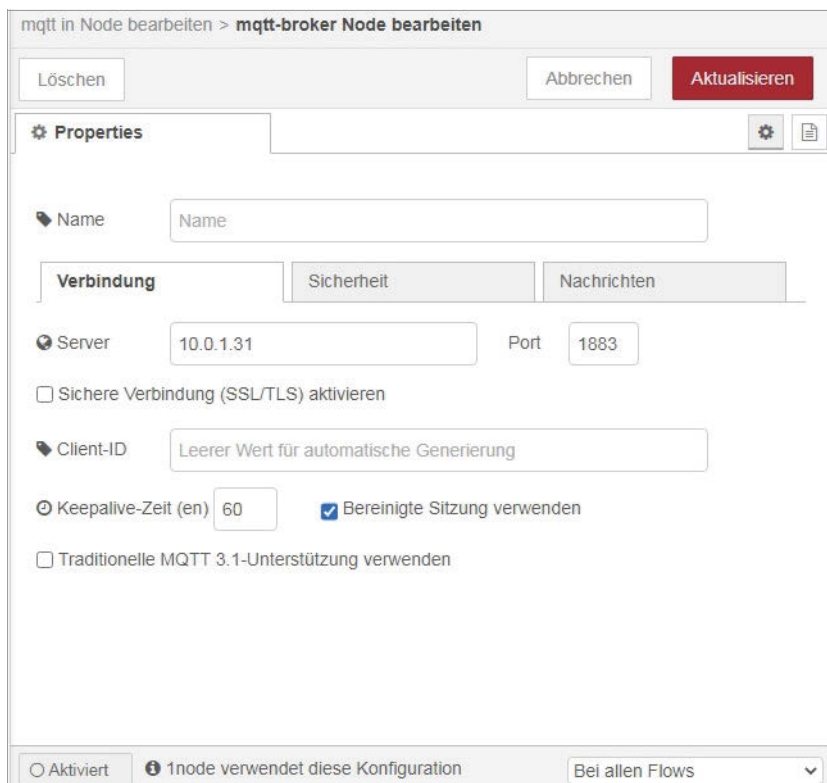


Abb. 6.32: Node-Red: MQTT-Server bearbeiten

Wird nun über das Terminal ein Wert auf den Topic `SensorTopic` publiziert (Abbildung 6.33), wird dieser dann umgehend im Debug-Fenster von Node-Red ausgegeben (Abbildung 6.34).

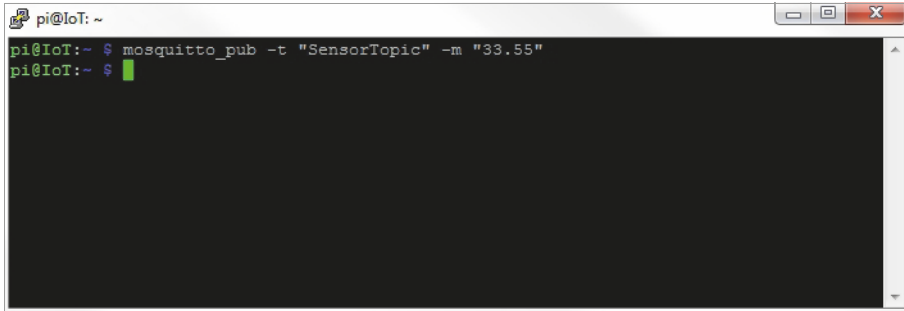


Abb. 6.33: Node-Red: Publizieren von Topic-Daten

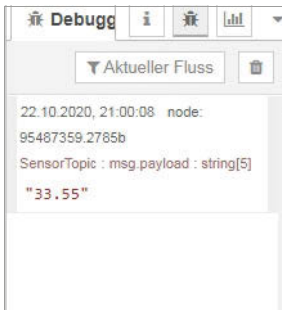


Abb. 6.34: Node-Red: Abonniertes Topic im Debug-Fenster

In der Praxis wird eine von einem Topic empfangene Nachricht im Flow weiterverarbeitet oder an eine Ausgabefunktion weitergegeben.

Im nachfolgenden Abschnitt wird das Node-Red-Dashboard in Betrieb genommen. Über das Dashboard können Sensordaten übersichtlich dargestellt werden.

6.6 Node-Red-Dashboard

Das Node-Red-Dashboard ist eine nützliche Erweiterung, die in keiner Node-Red-Installation fehlen darf.

Nach der Installation der Dashboard-Palette, wir haben diese bereits bei der Einführung von Node-Red (Abschnitt 6.3) installiert, stehen Ihnen in der Werkzeugpalette Nodes zur Erstellung von Bedienelementen und Anzeigen zur Verfügung (Abbildung 6.35).

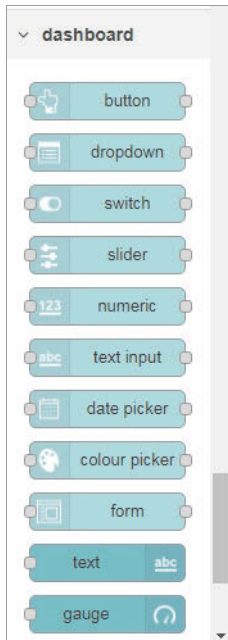


Abb. 6.35: Node-Red: Werkzeugpalette des Dashboards

Diese einzelnen Nodes für Eingabe oder Ausgabe auf Ihrem Informationsbildschirm können wie alle Nodes in den eigenen Flow integriert werden.

In Abbildung 6.36 ist ein Flow dargestellt, der einen Schiebeschalter (Sonoff) als Eingabeelement und zwei Anzeigeelemente in Form eines Zeigerinstruments (Gauge) und eines Grafik-Diagramms (On/Off) enthält.

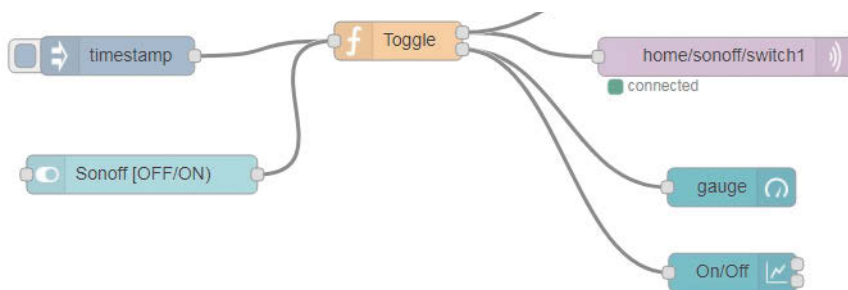


Abb. 6.36: Node-Red: Flow mit Dashboard-Nodes

Im Dashboard schlussendlich sehen die Ein- und Ausgabe-Nodes gemäß Abbildung 6.37 aus. Mit dem Schiebeschalter kann ein Sonoff-Schaltelement ein- und ausgeschaltet werden.

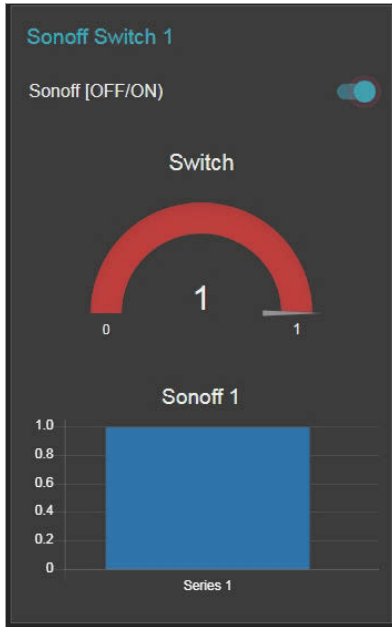


Abb. 6.37: Node-Red: Dashboard

Aufruf Dashboard

Das Dashboard wird über die folgende Adresse aufgerufen:

<http://IP-des-Raspberry:1880/ui/>

Beim Erstaufruf ohne konfiguriertes Dashboard erscheint ein Informationsbildschirm (Abbildung 6.38).

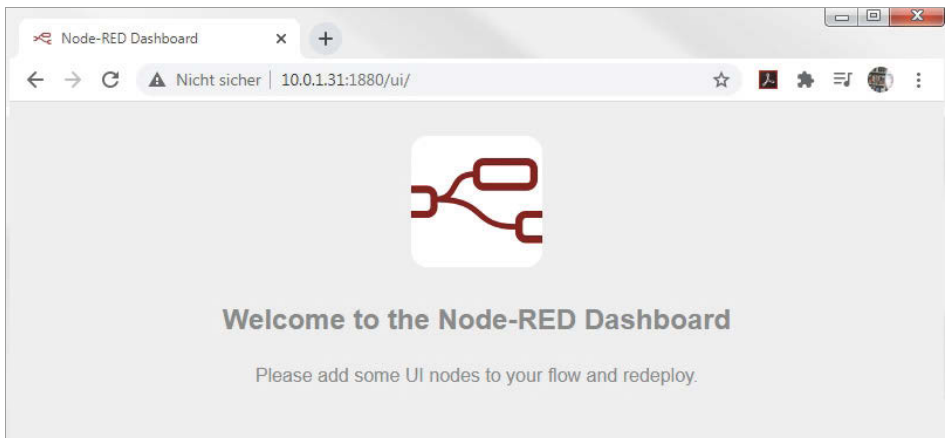


Abb. 6.38: Node-Red: Dashboard-Startseite

Dashboard erstellen

Bei der Erstellung eines Dashboards wird im ersten Schritt immer ein Flow mit den nötigen Eingabe- und Ausgabe-Nodes erstellt.

Im ersten Beispiel wird ein Schalter auf dem Dashboard platziert. Der Status (Ein/Aus) wird auf einem Zeigerinstrument angezeigt. Für den Flow benötigt man einen Switch-Node und einen Gauge-Node (Abbildung 6.39).



Abb. 6.39: Node-Red: Dashboard-Flow

Mit Doppelklick auf den Schalter-Node können Sie die Eigenschaften bearbeiten.

Der Node bekommt die Bezeichnung Ein/Aus, die über das Label erfasst werden. Für die Anzeige auf dem Dashboard muss das Element einer `ui_group` (Group) zugewiesen werden. Dies ist die Zuordnung zum jeweiligen Dashboard (Abbildung 6.40).

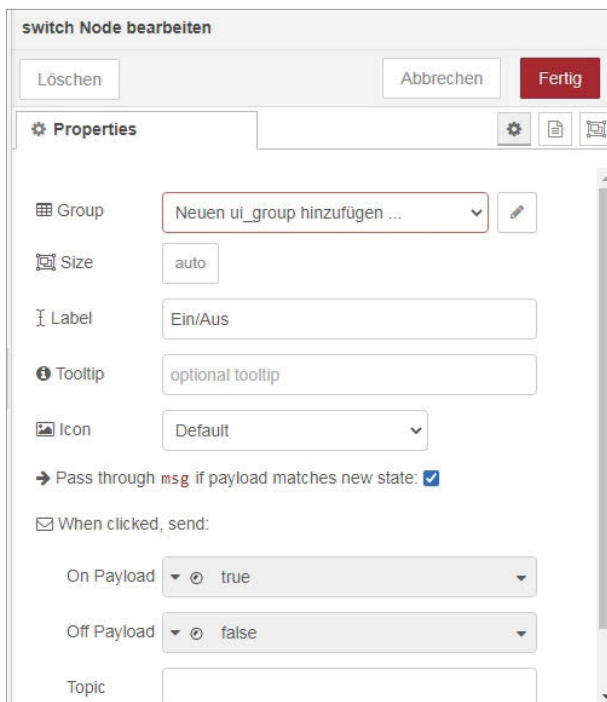


Abb. 6.40: Node-Red: Eigenschaften-Switch

Darum muss zuerst mit dem Stiftzeichen eine neue Gruppe definiert werden. Wir nennen sie **Smarthome**.

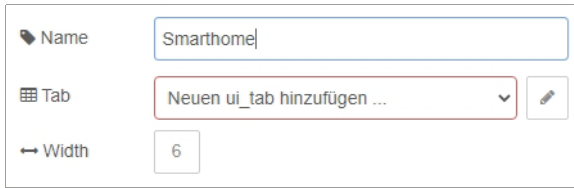


Abb. 6.41: Node-Red: Gruppe

Der nächste Schritt ist die Erstellung eines Tabs, also eines Menüpunkts in der Navigation, über den das Dashboard mit dem Schalter erreicht werden kann.

Mit dem Stiftzeichen kann ein neues `ui_tab` erstellt werden.

Wir nennen es **Home**.

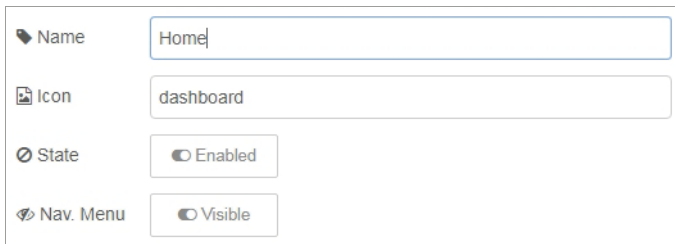


Abb. 6.42: Node-Red: UI-Tab erstellen

Nach Hinzufügen und FERTIG ist die Gruppe und der Navigationspunkt erstellt.

Nun werden noch die Werte in der Payload definiert. Dazu verwenden wir Zahlenwerte (Number). Bei On Payload wird eine 1, bei Off Payload eine 0 geliefert (Abbildung 6.44).

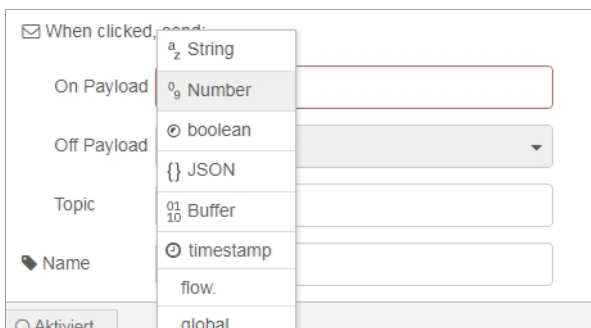
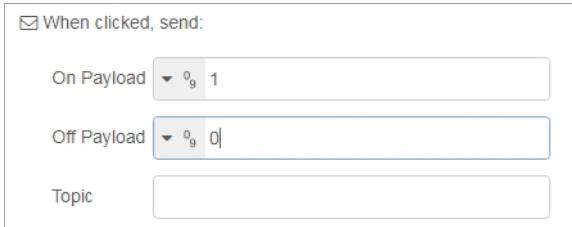


Abb. 6.43: Node-Red: Payload-Werte I



When clicked, send:

On Payload

Off Payload

Topic

Abb. 6.44: Node-Red: Payload-Werte II

Nun kann der Flow mit DEPLOY gespeichert werden.

In der DEBUG-Spalte unter DASHBOARD findet man den Tab HOME mit der Node-Gruppe SMARTHOME (Abbildung 6.45).

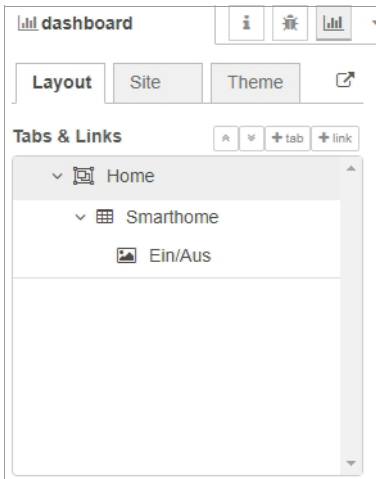


Abb. 6.45: Node-Red: Dashboard-Struktur

Nun wird noch der Gauge-Node mit einem Doppelklick bearbeitet.

Der Node bekommt das Label SCHALTER und der Wertebereich (RANGE) wird von 0 bis 1 gesetzt.

Die Gruppenzuordnung zu SMARTHOME belassen wir. So erscheint das Zeigerinstrument im gleichen Dashboard und in der gleichen Gruppe wie der Schalter.

The image shows the configuration panel for a Gauge node in Node-Red. The settings are as follows:

- Group:** [Home] Smarthome
- Size:** auto
- Type:** Gauge
- Label:** Schalter
- Value format:** {{value}}
- Units:** units
- Range:** min 0, max 1
- Colour gradient:** A horizontal bar with three segments: green, yellow, and red.
- Sectors:** 0, ..., optional, ..., optional, ..., 1

Abb. 6.46: Node-Red: Gauge-Node, Eigenschaften setzen

Nach dem Speichern kann das Dashboard aufgerufen werden. Dazu geben Sie die Adresse im Browser ein oder verwenden aus der Dashboard-Struktur-Anzeige den Link mit dem Pfeil (Abbildung 6.47).

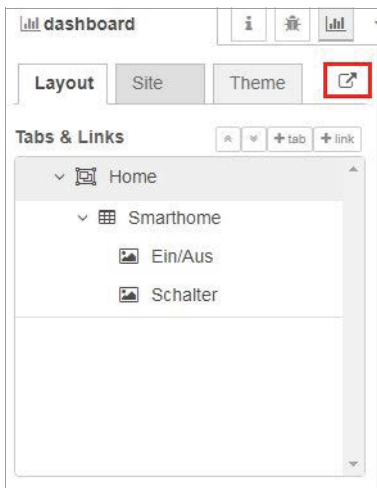


Abb. 6.47: Node-Red: Aufruf Dashboard

Jetzt öffnet sich ein neuer Browser-Reiter und das Dashboard wird angezeigt (Abbildung 6.48).

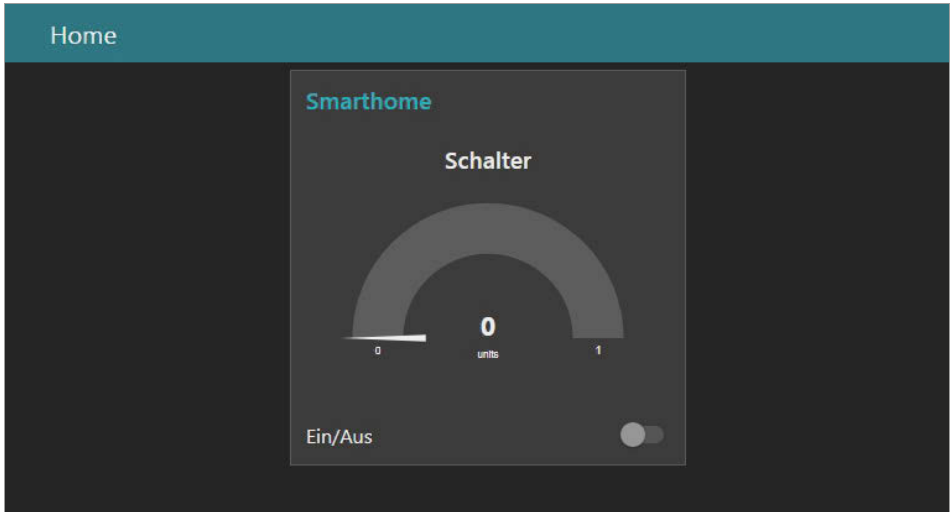


Abb. 6.48: Node-Red: Dashboard mit Schalter (AUS)

Mit der Betätigung des Schiebeschalters kann der Schalter eingeschaltet werden (Abbildung 6.49).

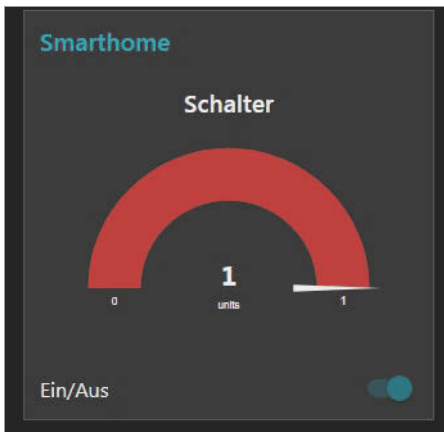


Abb. 6.49: Dashboard mit Schalter (EIN)

Damit ist das erste Dashboard erstellt.

Die Schalterbetätigung hat in diesem Flow noch keine Funktion. Hierzu muss der Fluss mit einem Ausgang eines Boards oder einem Topic (`mqtt out`) verbunden werden.

Stichwortverzeichnis

38 kHz 257
3D-Drucker 190
433 MHz 64, 215
433-MHz-Empfänger 222
433-MHz-Funkmodul 215, 216
433-MHz-Funkübertragung 195
433-MHz-Gateway 226
433-MHz-Sender 218
/dev/ttyACM0 184

A

Abonnent 123
Adafruit 207
Analog/Digital-Wandler 264
Analoger Eingang 17
Antenne 217
Aquarium-Timer 303
Arduino 13
 Anschlussmöglichkeiten 16
 Installation 20
 Minimalschaltung 36
 Programmierung 23
 Stecker 16
 Stromversorgung 17
Arduino Mega 17
Arduino Pro Mini 19
Arduino Uno 15
Arduino-Board *siehe* Arduino
Atmega328 39
ATtiny 39
AVRDude 227

B

Barometer 206
Batteriebetrieb 42
Beispielskript 10
Benutzeroberfläche 297
Betriebsart 154
Bezugsquelle
 Einzelkomponente 359
BH1750 178
 Busadresse 199

Bibliothek 28
 verwalten 29
Bibliotheksverwalter 30
Bigtimer 303
Bigtimer-Node 351
Blink 22
BME680 206
Board-Paket 40
Boardverwalter 80
Boardverwalter-URL 79
Body 119
Bootloader 36
Breakout-Board 78, 207
Briefkasten 276
Briefkastenwächter 275
Busadresse 179

C

callback() 133
Chart 189
Clone *siehe* Arduino
Color-Picker 349
COM-Port 23
configuration.yaml 292

D

Dashboard 172
Dashboard-Palette 169
Datenauswertung 236
Datenerfassung 189
Datenstring 213
Datenübertragung 215
Debug-Fenster 158
DHCP 68
DHT11 203
DHT22 203
Downloadbereich 10
Drahtlose Klingel 261
Drahtlose Verbindung 14, 65
DS18B20 329

E

Eingang
 analoger 17
 Einzelkomponente
 Bezugsquelle 359
 Energy-Monitor-Board 318
 Entwicklungsumgebung 13, 20
 Erweiterungsplatine *siehe* Shield
 ESP-01 *siehe* ESP8266
 ESP-12 *siehe* ESP8266
 ESP8266 65, 75
 Arduino-IDE 75
 MQTT Client 140
 RF-Gateway 234
 WiFi 67
 ESP8266-12 67
 ESP-Modul *siehe* ESP8266
 Espressif 75
 Etcher 47
 Ethernet-Bibliothek 66
 Ethernet-Shield 34, 65

F

Fernbedienung 253
 Filesystem 122
 Firmata 249
 Flash 15
 Flow 156, 159
 Fotowiderstand *siehe* LDR
 Friendly Name 109
 FS1000a 217
 FTDI 37
 FTDI-Anschluss *siehe* FTDI
 Funkmodul 217
 Funktions-Node 162, 185

G

Gefrierschrankwächter 328
 General-Purpose Input Output *siehe* GPIO
 GPIO 60
 Grundinstallation 46

H

Hauptprogramm 23
 HDMI 45
 Header 119
 Helligkeitssensor 179, 199
 Home Assistant 63, 287
 Installation 288
 Raspberry Pi 287
 HTTP 119
 HTTPS *siehe* ESP8266
 Hypertext-Transfer-Protokoll *siehe* HTTP

I

ICSP 17
 IDE *siehe* Entwicklungsumgebung
 Image-Datei 48
 index.html 122
 Infrarot-Signal 254
 Intranet-Anwendung 121
 IP-Adresse 121
 IR-Code 260
 IR-Empfänger 257
 IR-Sender 254

J

JavaScript-Code 162
 JeeLib-Bibliothek 43

K

Klingel
 drahtlose 261

L

Laststrom 317
 LDR 147
 LED-Streifen
 Ansteuerung 342
 Leistungsaufnahme 312
 Leuchtbalken 341
 Lichtschrankenschaltung 276
 Linux-Betriebssystem 43
 loop() 23
 Low-Power Library 43
 Lux 179
 Luxmeter 202

M

Maschine-Maschine-Kommunikation 119
 math.h 197
 MCB3008 264
 Message Queuing Telemetry Transport *siehe* MQTT
 Messkanäle 274
 Messwandler 317
 Messwerte 271
 Microcontroller 13, 14
 Microcontroller-Boards *siehe* Microcontroller
 Mosquitto 125, 152
 MQTT 119, 123
 Prinzip 124
 MQTT Publish 130
 MQTT Subscribe 137
 MQTT-Anwendung 245
 MQTT-Broker 64, 124, 125, 152

MQTT-Client 129, 130
 MQTT-Node 167
 MQTT-Topic 145
 MQTT-Verbindung 132
 MQTT-Wildcard 146

N

Node 159
 Node-ID 216
 NodeMCU 87
 Node-Red 63, 124, 153
 mit Raspberry Pi 151
 MQTT 166
 Oberfläche 155
 Palette verwalten 156
 Start 153
 Werkzeugpalette 159
 Node-Red-Dashboard 169
 NTC 23, 24, 196

O

Onboard-LED 253
 openHAB 63, 301

P

Payload 173
 Nutzlast 186
 Port
 serieller 183
 Port 1880 154
 Programm 15
 Protokolle 119
 Protonly 34, 194
 Protoshield 33, 194
 Publish 126
 Publisher 123
 PubSubClient 129
 PubSubClient-Bibliothek *siehe* PubSubClient
 Putty 56
 PWM 253

Q

QoS *siehe* Quality of Service
 Quality of Service 167

R

Radiohead-Library 218
 Raspberry Pi 13, 43, 65, 151
 Installation 45
 Raspberry Pi OS 46, 152
 Raspberry-Pi-Board 44
 Rauchmelder 229

reconnected() 136
 Regeln 242
 Remote-Zugriff 53
 RF Bridge 236
 RF433-MHz-Shield 329
 RF-Gateway 233
 RFLink 226, 227
 RF-Shield 35
 RGB-Streifen 340
 RJ45 65
 RPi 3 Modell B+ *siehe* Raspberry Pi
 RPi 4 Modell B *siehe* Raspberry Pi
 rpi gpio out 246
 RPi *siehe* Raspberry Pi
 RPi Zero *siehe* Raspberry Pi
 Rules 242

S

SCT-013 317
 SD-Karte 47
 Sensirion 203
 Sensor-Board 64
 Sensormodul 193
 Sensor-Node 193
 serial in 183
 Serieller Port 183
 setup() 23
 Setup-Funktion *siehe* setup()
 Shield 32, 65
 SHT31 202
 Sketch 15
 Sleep-Funktion 42
 Smarthome 63
 Plattformen 287
 Smarthome-Hardware 13
 Smarthome-Infrastruktur *siehe* Smarthome
 Smartphones 177
 Sonoff 112
 FTDI-Adapter 117
 serielle Schnittstelle 116
 Sonoff POW 112
 Sonoff RF Bridge 237
 SPI-Bus 264
 SSH 53
 Standard-Bibliothek 28
 Standarduser 51
 Steinhart-Hart-Gleichung 26
 Stromwächter 308
 Sonoff Pow 309
 Stromsensor 316
 Stückliste 353
 Subscribe 126

T

Tasmota 100, 235
 Installation 102
Tasmotizer 104
Temperaturmesser 23
Temperatursensor 24, 196
Thermistor 196
Timer-Node *siehe* Bigtimer
Topic 123, 236
Topic-Level 145
TSOP4838 257
ttyAMA0 184

U

Umweltsensor 64, 204
URL 67
USB-Anschluss 16
USB-Serial-Adapter 239

V

Verbindung
 drahtlose 14, 65

Verbindungsaufbau 133
VNC 53
VNC Server 58

W

Waschmaschinenwächter 323
Webclient 67
 mit HTTPS 96
Webkommunikation 121
Webserver 70, 121
Wemos D1 82
Wemos D1 Mini 83
 Anschlussbelegung 84
 Shields 86
 Technische Daten 84
Wemos-Webclient 92
WiFi mit ESP8266 *siehe* ESP8266
WiFi-Verbindung 67
Wiz5100 66
WS28xx 341
WWW 119