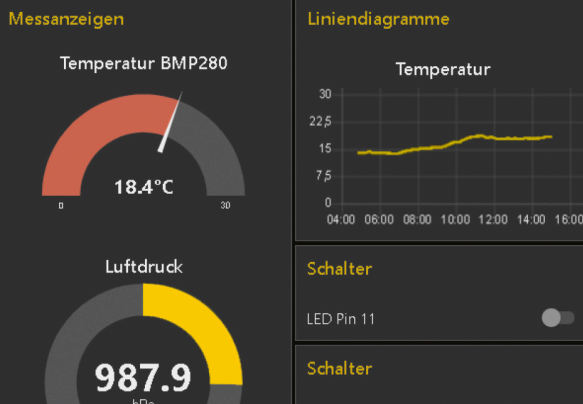
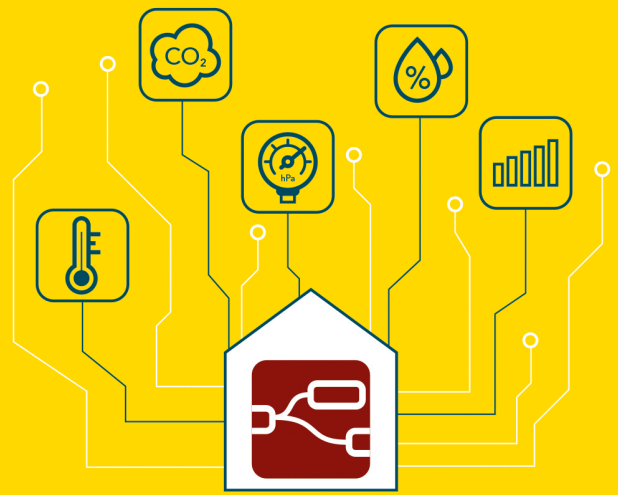


Udo Brandes

Flow, JSONata
ngrok, funct
Context, MSQ
Raspberry Pi
inject, Node



Node-RED

Das umfassende Handbuch

- ▶ Von den Grundlagen bis zum fertigen Dashboard
- ▶ Steuerungen und Logik visuell programmieren
- ▶ Daten verarbeiten in Maker- und IoT-Projekten und in der Hausautomation

4., aktualisierte Auflage



Beispiel-Flows und Projektdateien zum Download



Rheinwerk
Computing

Kapitel 4

Das Node-RED-Dashboard

Eine Datenverarbeitung ohne visuell ansprechende Darstellung der Ergebnisse ist heutzutage kaum mehr vorstellbar. Node-RED unterstützt Sie hier in gewohnter Weise mit einfachen Nodes, die aber auch komplexe Anforderungen abdecken können. Möglich ist (fast) alles.

Das Dashboard von Node-RED ist eine mächtige grafische Benutzeroberfläche (*User Interface, UI*). Sie fügt sich lückenlos in das Node-RED-Konzept ein, da die einzelnen Elemente durch Nodes repräsentiert werden.

Dieses Kapitel führt auf Basis der Grundlagen aus Kapitel 3 und kleiner Anwendungen mit dem Raspberry Pi in die Anwendung einfacher Dashboard-Darstellungen ein. An geeigneter Stelle, insbesondere in Kapitel 11, erfolgt dann noch einmal eine detailliertere Behandlung anhand ausgewählter weiterreichender Beispiele.

Die Sache hat aber einen nicht unerheblichen Pferdefuß: Das originale Dashboard von Node-RED basiert auf Angular v1.0, das offiziell veraltet ist – es funktioniert, aber keiner gibt eine Garantie, dass es noch lange seinen Dienst tun wird. Bislang hat das Node-RED-Entwicklerteam noch keinen Nachfolger vorgestellt.

Es gibt allerdings mit Node-RED Dashboard 2.0 eine Alternative: Es ist als Open-Source-Projekt angelegt. Dahinter steht jedoch die Firma *FlowFuse* (<https://flowfuse.com>). FlowFuse wurde 2021 von Nick O’Leary, dem Co-Gründer von Node-RED ins Leben gerufen. Damit bestehen recht gute Aussichten, dass der kostenlose Zugang zu den neuen Dashboard-Nodes erhalten bleibt, auch wenn Sie von einer Firma bereitgestellt werden, deren Geschäftszweck natürlich darin besteht, solche Dienste gegen ein Entgelt anzubieten.

Für dieses Buch möchte ich Ihnen zunächst das originale Node-RED Dashboard vorstellen, das noch immer weit verbreitet ist. Abschnitt 4.8 widmet sich dann dem Dashboard 2.0. Sie können beide parallel betreiben.

4.1 Installation

Die Installation des Dashboards erledigen Sie mit dem Palettenmanager (**Hauptmenü • Palette verwalten • installieren**). Suchen Sie nach »dashboard«. Die Liste enthält mehr als 40 Treffer. Als Grundlage für die Beispiele in diesem Buch verwende ich aus den oben genannten Gründen das Paket *node-red-dashboard*.

Nach der Installation verfügt die Palette über zwei neue Gruppen (*dashboard* und *Dashboard*) mit insgesamt 16 Nodes sowie drei weiteren Konfigurations-Nodes. Tabelle 4.1 liefert Ihnen einen Überblick.


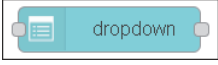
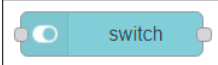



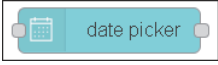





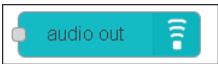
Node	Funktion
 button	Eine Schaltfläche. msg.payload enthält den Inhalt des Felds <i>Payload</i> .
 dropdown	Ein Drop-down-Auswahlfeld. Mehrere Wert-Label-Paare können nach Bedarf hinzugefügt werden.
 switch	Ein Schalter. Statusänderungen erzeugen eine msg.payload mit den angegebenen On- und Off-Werten.
 slider	Ein Schieberegler. Die Werte ändern sich zwischen den Grenzen <i>min</i> und <i>max</i> .
 numeric	Ein Feld für die Eingabe eines Zahlenwerts.
 text input	Ein Feld für die Eingabe einer Zeichenkette.
 date picker	Ein Dialog für die Datumsauswahl.
 colour picker	Ein Dialog für die Farbauswahl. Er verbindet Eingaben mit gleichen Farbwerten zu Gruppen.
 form	Ein Feld für die Formulareingabe.
 text	Ein Feld für die Textausgabe.
 gauge	Eine analoge Messgeräteausgabe. Der Node stellt eine Ausgabe wie bei einem Messinstrument dar.
 chart	Eine Diagrammausgabe.
 audio out	Eine Audioausgabe. Der Node spielt Audiodateien oder Text-to-Speech (TTS) ab.

Tabelle 4.1: Die Dashboard-Nodes




Node	Funktion
	Ein allgemeine Benachrichtigung. Der Node zeigt Dialogmeldungen an.
	Der Node konfiguriert eine dynamische, d. h. zur Laufzeit veränderbare Dashboard-Steuerung.
	Eine Vorlage für dynamische Benutzeroberflächenelemente.

Tabelle 4.1: Die Dashboard-Nodes (Forts.)

4.2 Browseraufruf und Einstellungen

Das Dashboard erreichen Sie im Browser über die IP-Adresse des Node-RED-Editors. Ergänzen Sie zusätzlich noch die Pfadbezeichnung *ui*.

Der vollständige Aufruf lautet daher `http://<IP-Adresse-Node-RED>/ui`, also z. B.:

```
http://192.168.1.210:1880/ui
```

Die Pfadbezeichnung können Sie in der Datei `settings.js` ändern:

```
ui: { path: "ihr_neuer_Pfad" },
```

Das Dashboard können Sie entsprechend dem `http-in`-Node in der Konfigurationsdatei `settings.js` mit `httpNodeAuth` absichern, damit der Zugriff nur verschlüsselt möglich ist. Wie das geht, erkläre ich in Abschnitt 1.6, »Node-RED absichern«.

4.3 Der Schnelleinstieg: So erstellen Sie Ihre erste Dashboard-Ausgabe

Das Dashboard weist eine bestimmte Hierarchie auf:

- **Basis:** Definiert die Basis-URL (z. B. `/ui`) für Ihr Dashboard.
- **Seite:** Eine bestimmte Seite, zu der ein Besucher navigieren kann.
- **Gruppe:** Eine Sammlung von Widgets, die auf eine Seite gerendert ist.
- **Widget:** Ein einzelnes Widget (z. B. Diagramm, Schaltfläche, Formular), das im Dashboard dargestellt wird.

Die Konfiguration eines Dashboards kann recht umfassend und komplex sein. Beginnen Sie deshalb mit einem kleinen Beispiel, um ein Gefühl für die Darstellung zu bekommen.

Unser Ziel ist, über ein Dashboard per Knopfdruck einen Gruß abzusetzen. Die Anzeige des Grußes soll nach zwei Sekunden wieder erlöschen.

Erstellen Sie dazu einen Flow mit drei Nodes gemäß Abbildung 4.1.

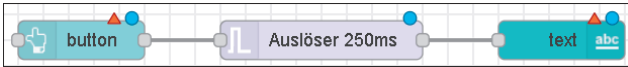


Abbildung 4.1: Der Flow, der einen Gruß absenden soll

Rote Dreiecke oberhalb der Nodes zeigen an, dass Sie noch weitere Einstellungen vornehmen müssen.

Schritt 1: Den button-Node konfigurieren und eine Dashboard-Gruppe erstellen

Öffnen Sie die Maske mit den Node-Eigenschaften (siehe Abbildung 4.2).

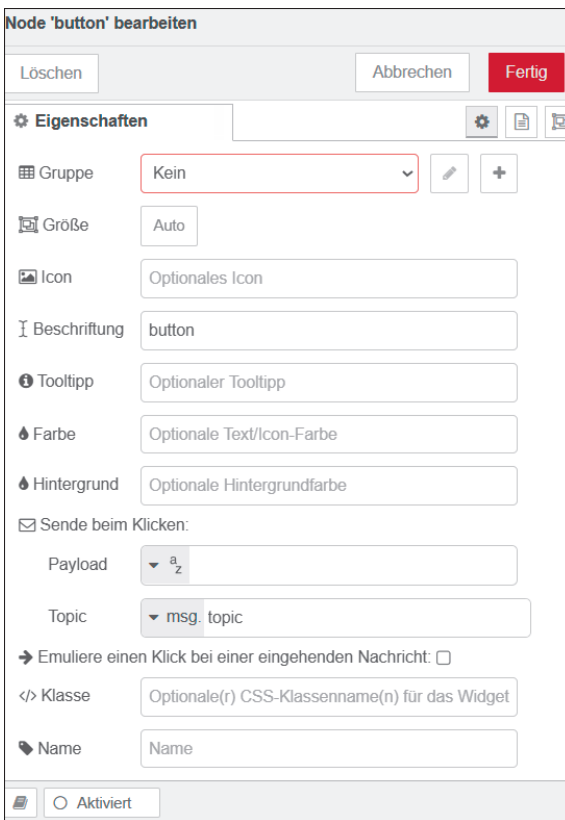


Abbildung 4.2: Die Eigenschaften des button-Nodes

Sofort sticht das rot umrandete Feld **Gruppe** ins Auge. Hier legen Sie die Zugehörigkeit des Nodes zu einer Dashboard-Gruppe fest. Eine Dashboard-Seite enthält eine oder mehrere Gruppen (weitere Informationen dazu finden Sie in Abschnitt 4.4).

Üblicherweise enthält die Drop-down-Liste alle bereits definierten Gruppen zur Auswahl. Die Liste ist zu Beginn selbstverständlich leer. Erstellen Sie daher eine neue Gruppe durch Klick auf das Stift-Icon. Sie gelangen zu einem weiteren Fenster, in dem Sie die Dashboard-Gruppe konfigurieren können (siehe Abbildung 4.18).

Schritt 2: Den Dashboard-Tab festlegen

Eine wesentliche Eigenschaft einer Dashboard-Gruppe ist die Zugehörigkeit zu einem Dashboard-Tab (siehe Abschnitt 4.4). Auch hier haben Sie die Möglichkeit, einen vorhandenen Dashboard-Tab auszuwählen oder zu einem Fenster für die Neuerstellung zu wechseln. Die Liste ist zu Beginn ebenfalls leer, sodass Sie durch Klick auf das Stift-Icon ein weiteres Fenster (siehe Abbildung 4.17) zur Konfiguration eines neuen Tabs aufrufen müssen.

Lassen Sie die Einstellungen unverändert und gehen Sie mit **Hinzufügen** zurück. Der Dashboard-Tab steht nun fest (siehe Abbildung 4.3, **Tab: Home**).

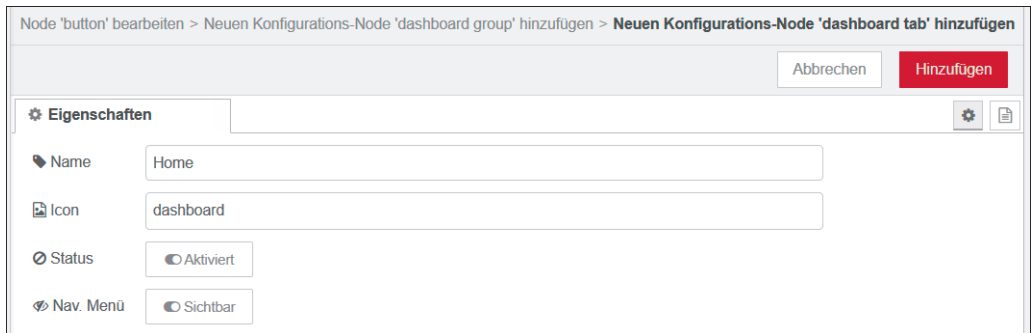


Abbildung 4.3: Konfigurieren Sie den groupconfig-Node.

Lassen Sie erneut die Einstellungen unverändert und gehen Sie mit **Hinzufügen** zurück. Nun haben Sie auch die Dashboard-Gruppe definiert. In Abbildung 4.4 sehen Sie das an **Gruppe: [Home] Standard**.

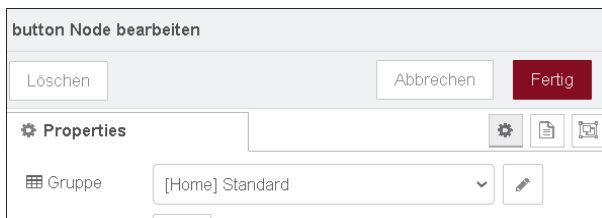


Abbildung 4.4: Der button-Node ist konfiguriert.

Fügen Sie vor dem Klick auf **Fertig** bei **Payload** noch »Hallo Welt« ein.

Die roten Dreiecke sind jetzt verschwunden, weil Node-RED die getroffenen Einstellungen zur Dashboard-Ausgabe auch auf den *text*-Node übertragen hat.

Schritt 3: Den trigger-Node einstellen

Der *trigger*-Node hat in diesem Beispiel die Aufgabe, die eingehende Nachricht weiterzuleiten und nach zwei Sekunden eine Löschnachricht hinterherzuschicken. Abbildung 4.5 zeigt die Eigenschaften des *trigger*-Nodes, die Sie für unser Beispiel brauchen.

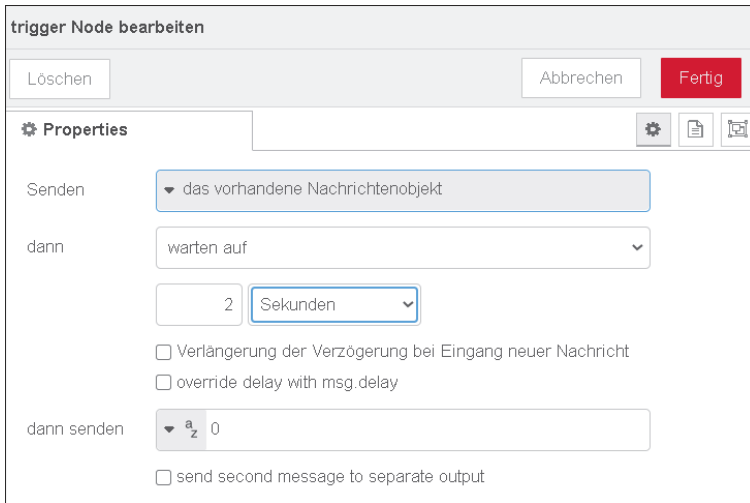


Abbildung 4.5: Die Einstellungen des *trigger*-Nodes

Schritt 4: Die Dashboard-Ausgabe starten

Klicken Sie nun auf **Deploy** und öffnen Sie das Dashboard in einem Browser, um den Flow zu testen. Wie Abbildung 4.6 zeigt, sehen Sie einen Button, mit dem Sie einen kurzen Text produzieren können.

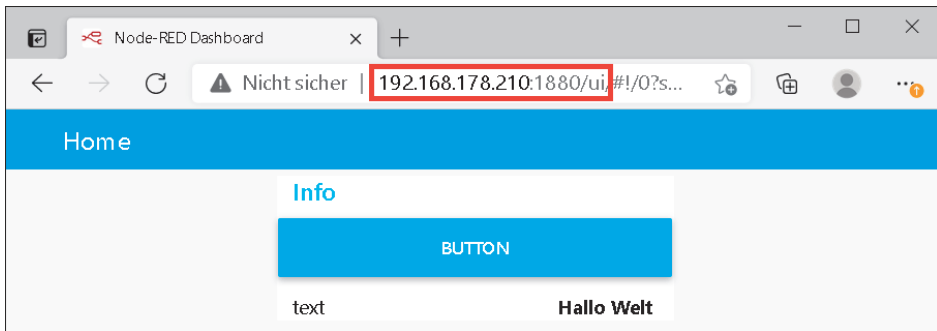


Abbildung 4.6: Flow-Gruß, Dashboard-Ausgabe

4.4 Das Dashboard-Design bestimmen

Die erste Dashboard-Ausgabe ist nur ein erster Blick auf die weite Spielwiese des Dashboard-Designs. Die folgenden Abschnitte sollen wesentliche Aspekte aufgreifen und Ih-

ren Werkzeugkasten für eigene Entwicklungen anreichern. Aber was ist das Dashboard eigentlich?

Mit dem Browseraufruf (siehe Abschnitt 4.2) gelangen Sie gewissermaßen zur Startseite Ihrer Webanwendung. Ihre Website kann aber mehrere Seiten haben. In Node-RED heißen diese Seiten *Tabs* (siehe auch Abschnitt 4.4.2). Das Layout einer Seite orientiert sich an einem Gitter bzw. Zeichenraster.

Eine Seite gliedert sich in Spalten (sogenannte *Gruppen*, siehe auch Abschnitt 4.4.2). Die Spaltenbreite beträgt per Voreinstellung sechs Rasterkästchen (siehe Abbildung 4.7), und ein Rasterkästchen hat per Voreinstellung eine Größe von 48×48 Pixeln.

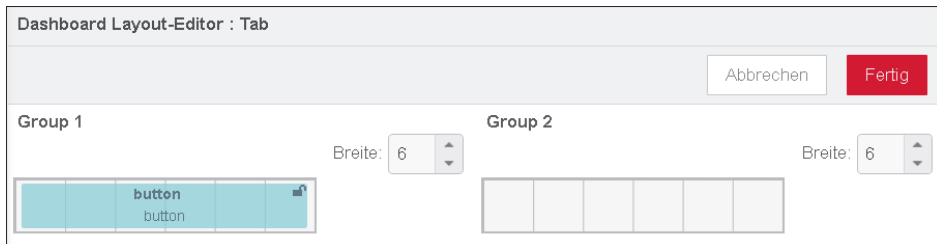


Abbildung 4.7: Die Aufteilung des Node-RED-Dashboards

Die grafischen Erscheinungsbilder der Nodes werden *Widgets* genannt. In einer solchen Gruppe befinden sich die Widgets dann wie in einer Liste.

Gruppen und Widgets orientieren sich in Größe und Platzierung an diesem Raster.

Darüber hinaus kennt das Node-RED-Dashboard noch ein weiteres Element, das *Link* heißt. Schließlich können auch Icons Verwendung finden.

Demnach versteht man in Node-RED unter dem Begriff *Dashboard* also alle einzelnen Tabs und Links Ihrer Webanwendung.

Das Design Ihres Dashboards legen Sie auf zwei Wegen fest:

- über die rechte Seitenleiste (siehe Abschnitt 4.4.2)
- über Konfigurations-Nodes für Tabs und Gruppen (siehe Abschnitt 4.4.4)

4.4.1 Icons

Einige Widgets (z. B. das Button-Widget, siehe Abbildung 4.2) oder Tabs (siehe Abbildung 4.17) lassen sich mit Icons aufwerten. Sie können vier Icon-Sets einsetzen:

- **Font Awesome 4.7** (<https://fontawesome.com/v4.7.0/icons>)

Ergänzen Sie beispielsweise die Eigenschaft **Icon** des *button*-Nodes aus dem vorherigen Beispiel um das Icon *User-Circle*, um eine ansprechendere Gestaltung der Schaltfläche zu erreichen (siehe Abbildung 4.8). Der passende Eintrag lautet `fa-user-circle`.



Abbildung 4.8: Button-Widget mit Icon

■ **Angular Material Icons** (<https://klarsys.github.io/angular-material-icons>)

Setzen Sie für eine ansprechendere Gestaltung des Dashboard-Menüeintrags aus Abbildung 4.9 beispielsweise die Eigenschaft **Icon** des *tabconfig*-Nodes auf das Icon *Home*. Der Eintrag lautetet *home*.

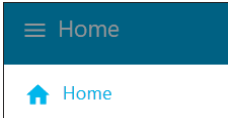


Abbildung 4.9: Dashboard-Menü-Icon

■ **Weather Icons Lite** (https://github.com/Paul-Reed/weather-icons-lite/blob/master/css_mappings.md)

Hier finden Sie spezielle Wetter-Icons, z. B. *wi-darksky-cloudy*.

■ **Material Design Iconfont** (<https://jossef.github.io/material-design-icons-iconfont>)

Für dieses Icon-Pack benötigen Sie eine Internetverbindung, damit das Icon auch angezeigt wird. Wenn Sie beispielsweise ein Wecker-Icon suchen, nutzen Sie den Eintrag *mi-alarm_on*.


4.4.2 Die rechte Seitenleiste

Die rechte Seitenleiste enthält ein weiteres Icon namens **Dashboard**. Der Eintrag ist in drei Registerkarten unterteilt (siehe Abbildung 4.10):

- **Layout**
- **Site**
- **Theme**



Abbildung 4.10: Das Icon »Dashboard« und seine Registerkarten

Das -Icon auf der rechten Seite blättert unmittelbar das Dashboard auf.

Layout

Die Registerkarte **Layout** beginnt mit der Zeile **Tabs & Links** (siehe Abbildung 4.11). Die Icons in dieser Zeile expandieren die Listeneinträge bzw. klappen sie wieder zusammen oder erstellen einen neuen Tab (+ **Tab**) bzw. einen neuen Link (+ **Link**).

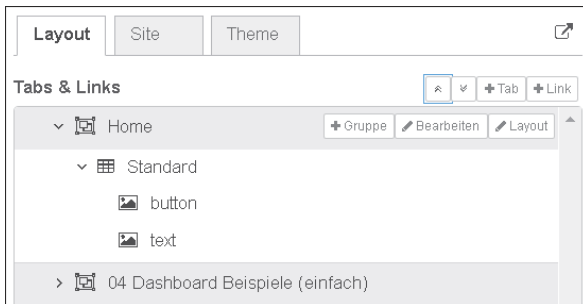


Abbildung 4.11: Dashboard-Einstellungen mit dem Tab »Layout«

Die Reihenfolge der Listeneinträge bestimmt die Darstellung im Dashboard. Der erste Tab der Liste ist der erste Menüeintrag des Dashboard-Menüs und gleichzeitig der Startbildschirm Ihrer Website. Die Gruppen unterhalb des Tabs sind von links nach rechts geordnet, die Widgets von oben nach unten. Die Reihenfolge ändern Sie per Drag-and-drop.

Abweichende Reihenfolge zwischen Editor und Dashboard

Es kann passieren, dass die Reihenfolge im Dashboard eine andere ist als die im Node-RED-Editor vorgegebene. Dies ist ein kleiner Bug in Node-RED. Verschieben Sie die Tabs im Node-RED-Editor erneut und deployen Sie die Änderungen. Der Fehler sollte dann behoben sein.



Sobald ein Listeneintrag mit dem Mauszeiger den Fokus erhält, erscheinen weitere Schaltflächen:

- bei Tabs – + **Gruppe**, **Bearbeiten**, **Layout**
- bei Gruppen – + **Abstand**, **Bearbeiten**
- bei Widgets – **Bearbeiten**

Bearbeiten führt zur Maske für die Einstellungen des jeweiligen Nodes.

Mit + **Gruppe** (ebenfalls mit + **Tab**) erstellen Sie eine neue Gruppe und gelangen in die Konfiguration der Gruppe bzw. des Tabs.

Ein Klick auf **Layout** öffnet den Dashboard-Layout-Editor (siehe Abbildung 4.12). Anpassungen sind hier übersichtlicher und einfacher zu bewerkstelligen als über die einzelnen Konfigurations-Nodes.

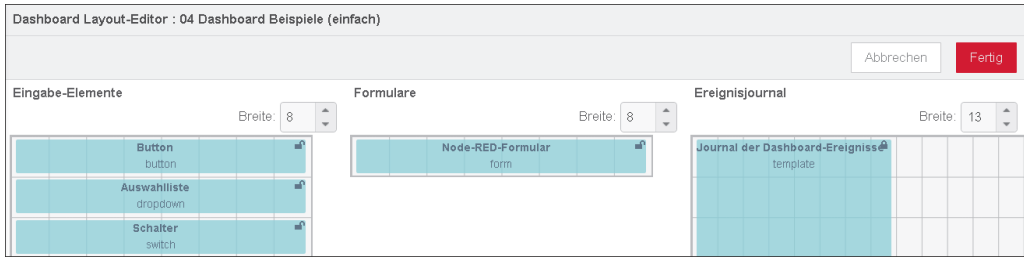


Abbildung 4.12: Der Dashboard-Layout-Editor



Das Layout des Dashboards

Der Layoutalgorithmus des Dashboards versucht immer, Elemente so hoch und so weit links wie möglich in ihrem Container zu verorten. Dies gilt sowohl für die Positionierung von Gruppen auf der Seite als auch für die Positionierung von Widgets in einer Gruppe.

Das führt bei einer Gruppe mit der Breite 6 mit sechs Widgets (jedes hat eine Breite von 2) zu einer Anordnung in zwei Zeilen mit jeweils drei Widgets.

Dagegen stehen alle sechs Widgets nebeneinander, wenn zwei Gruppen (Breite 6) zu je drei Widgets angelegt sind – zumindest solange Ihr Browserfenster breit genug ist. Wenn Sie den Browser verkleinern, verschiebt sich die zweite Gruppe irgendwann in eine Spalte unter die erste.

Im Hinblick auf eine dynamische Änderung der Seitengröße auf kleineren Bildschirmen sind mehrere Gruppen anstelle einer größeren sinnvoller.

+ **Abstand** fügt ein Abstands-Widget (*Spacer*) ein. Spacer helfen bei der Gliederung des Layouts. Die Größe ist per Default ein Rasterkästchen, lässt sich aber hinsichtlich der Anzahl horizontaler und vertikaler Elemente einstellen bzw. auf **AUTO** setzen.

+ **Link** fügt einen Link ein, den Sie über das Dashboard-Menü auswählen können. Abbildung 4.13 zeigt als Beispiel eine Verlinkung zur Node-RED-Homepage.

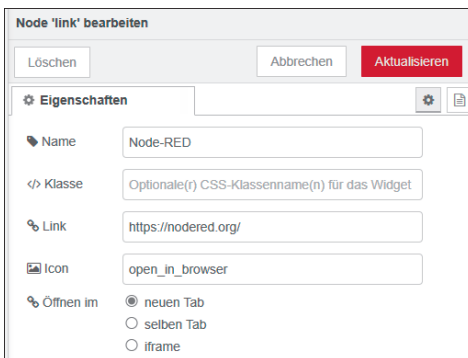


Abbildung 4.13: Einen link-Node definieren

Site

Die Registerkarte **Site** beinhaltet globale Layouteinstellungen Ihrer »Node-RED-Website«.

Größen	Horizontal	Vertikal
1x1 Widget-Größe	48	48
Widget-Abstände	6	6
Gruppenabstände innen	0	0
Gruppenabstände außen	6	6

Abbildung 4.14: Dashboard-Einstellungen mit dem Tab »Site«

- **Titel** – Voreingestellt ist »Node-RED Dashboard«.
- Die **Optionen** betreffen die Anzeige der Titelleiste und die Anzeige der Seitenleiste, sofern Ihr Dashboard mehrere Tabs hat. Eine weitere Option gilt dem Wischen zwischen Tabs sowie möglichen Erscheinungsbildern (hierzu folgen weitere Informationen in Kapitel 11, »Dashboards für Fortgeschrittene«).
- **Datumsformat** – Voreingestellt ist das Format **DD.MM.JJJJ** (also z. B. 12.12.2026). Klicken Sie auf das **i**-Icon, um sich weitere Varianten anzusehen.
- **Größen** – Mit den Vorgaben beeinflussen Sie die Rastergröße und die Platzierung von Widgets und Gruppen (siehe Abbildung 4.15).

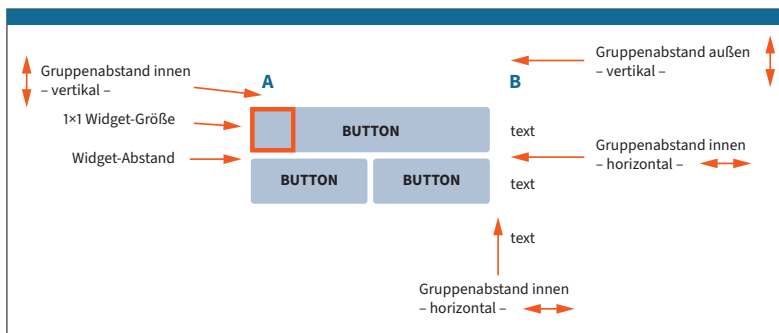


Abbildung 4.15: So definieren Sie die Größen im Dashboard.

- **1×1 Widget-Größe** – Bestimmt die Größe eines Rasterkästchens in Pixeln. Der kleinste Wert ist 24.
- **Widget-Abstände** – Legt den horizontalen und vertikalen Abstand zwischen zwei Widgets fest. Abstand 0 bedeutet, dass alle Widgets unmittelbar aneinanderstoßen.
- **Gruppenabstände** – Gruppenabstände betreffen die Platzierung der Gruppen.

Theme

Das Register **Theme** bietet Ihnen die Möglichkeiten aus Abbildung 4.16, um das Erscheinungsbild Ihres Dashboards einzustellen.

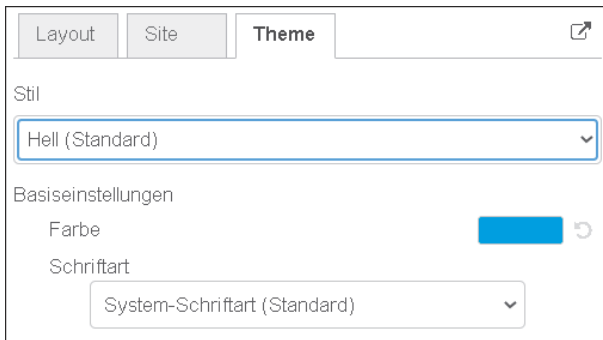


Abbildung 4.16: Gestaltungsoptionen mit dem Tab »Theme«

Wählen Sie bei **Stil** die Option **Dunkel**, wenn Sie eine augenfreundliche Erscheinung bevorzugen, oder definieren Sie sich Ihren eigenen Stil (**Benutzerdefiniert**).

Die Option **Farbe** bezieht sich auf die Farbe der Widgets. Sie können den Farbeditor nutzen und im RGB-Farbspektrum eine Farbe wählen. Mit Hilfe des Color-Picker können Sie direkt aus dem Bildschirm eine Farbe übernehmen.

4.4.3 Custom CSS

Bei einigen Nodes können Sie das Erscheinungsbild gezielt festlegen, z. B. bei dem *template*-Node, dem *chart*-Node, dem *notification_node*, dem *gauge*-Node, dem *dashboard-link*-Node und dem *groupconfig*-Node. Beispiele zum *template*-Node finden Sie in Kapitel 7 und in Kapitel 12.

4.4.4 Die Konfiguration von Tabs und Gruppen

Das Node-RED-Dashboard kennt für die Konfiguration von Tabs und Gruppen zwei Konfigurations-Nodes (Beiden sind Sie schon begegnet):

- den *tabconfig*-Node für Dashboard-Tabs
- den *groupconfig*-Node für Dashboard-Gruppen

Der tabconfig-Node

Abbildung 4.17 zeigt den Eingabebildschirm des *tabconfig*-Nodes.

button Node bearbeiten > Neuen dashboard groupconfig-Node hinzufügen >
Neuen dashboard tabconfig-Node hinzufügen

Abbrechen **Hinzufügen**

Properties

Name: Home

Icon: dashboard

Status: Aktiviert

Nav. Menü: Sichtbar

Aktiviert 0 -Nodes verwenden diese Konfiguration Bei allen Flows

Abbildung 4.17: Die Eigenschaften von »dashboard-tabconfig«

Mit dem *tabconfig*-Node können Sie folgende Parameter für Ihr Dashboard setzen:

- **Name** ist die Bezeichnung des Tabs; sie findet sich in der Tab-Überschrift des Dashboards und in der Menüleiste.
- Das **Icon** (siehe Abschnitt 4.4.1) erscheint in der Menüleiste des Dashboards.
- Die Option **Status** bezieht sich auf die Anzeige des Tabs im Dashboard. Ist der Status **Deaktiviert**, erscheint der Tab in der Menüliste des Dashboards, lässt sich aber nicht mehr auswählen.
- **Nav. Menü** – Üblicherweise führt die Menüleiste des Dashboards jeden Tab auf (**Sichtbar**). Die Option **Versteckt** blendet die Anzeige des Tabs im Dashboard aus.

Die Fußzeile erlaubt ein Aktivieren bzw. Deaktivieren des Nodes. Wenn Sie ihn deaktivieren, führt das dazu, dass kein Menüeintrag auf dem Dashboard vorhanden ist. Außerdem gibt es Informationen darüber, wie häufig Widgets diesen Konfigurations-Node nutzen.

Der groupconfig-Node

Die Konfiguration des *groupconfig*-Nodes sehen Sie in Abbildung 4.18.

Der *groupconfig*-Node erlaubt folgende Einstellungen:

- **Name** gibt die Bezeichnung der Gruppe an; sie ist die Spaltenüberschrift der Dashboard-Seite.
- **Tab** ist der Dashboard-Tab, dem die Gruppe zugeordnet ist.

- **Klasse** erlaubt mit Hilfe von CSS die Darstellung anzupassen
- **Breite** bezieht sich auf die zugebilligte Rasterbreite
- **Gruppenname anzeigen** sorgt dafür, dass die Spaltenüberschrift angezeigt wird.
- **Gruppenreduzierung zulassen** blendet die Widgets der Gruppe aus, die Spalte als solche bleibt jedoch in voller Breite sichtbar.

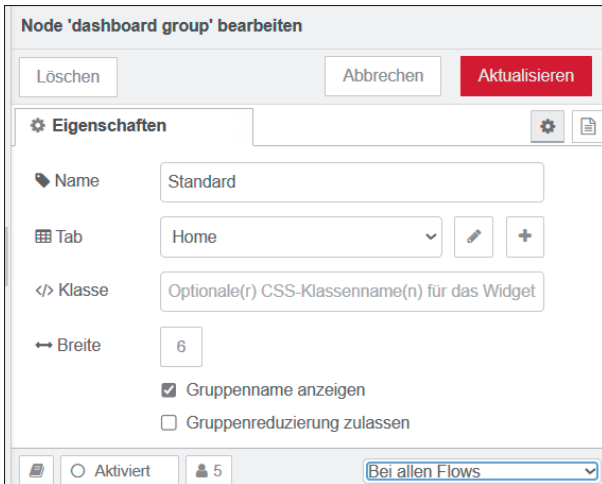


Abbildung 4.18: Die Eigenschaften von »dashboard-groupconfig«

Die Fußzeile erlaubt ein Aktivieren bzw. Deaktivieren des Nodes. Ein Deaktivieren hat zur Folge, dass die Gruppe nicht mehr im Dashboard angezeigt wird. Außerdem gibt es dort Informationen darüber, wie häufig Widgets diesen Konfigurations-Node nutzen. In einer Dropdown-Liste können Sie den Anwendungsbereich des Nodes auf einen speziellen Flow einschränken.

4.5 Die Dashboard-Widgets in Aktion

In diesem Abschnitt möchte ich Ihnen anhand eines kleinen Beispiels Dashboard-Eingabe-Nodes bzw. -Widgets vorstellen. Den Beispiel-Flow sehen Sie in Abbildung 4.19. Er verwendet bereits den *function*-Node (siehe Abschnitt 5.1) und den *template*-Node (siehe Abschnitt 11.2), sodass an dieser Stelle lediglich die notwendigen Codezeilen stehen.

In diesem Flow sind alle Dashboard-Eingabe-Nodes zumindest einmal vertreten. Verschaffen Sie sich also eine Übersicht.

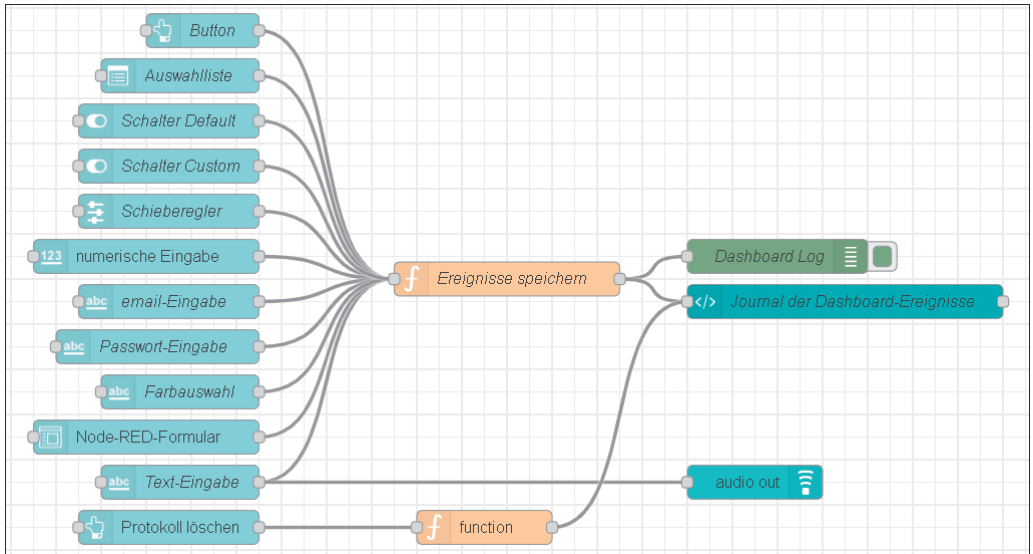


Abbildung 4.19: Ein Beispiel-Flow mit Dashboard-Eingabe-Nodes

4.5.1 Der button-Node

Der *button*-Node repräsentiert einen Taster. Bemerkenswert ist, dass er neben dem Ausgangsport auch über einen Eingangsport verfügt. Abbildung 4.20 zeigt die Konfigurationsmöglichkeiten.

Gruppe	[04 Dashboard Beispiele (einfach)] Eingabe-Elemente
Größe	Auto
Icon	fa-star
Beschriftung	Taster
Tooltipp	ein Klick für den Flow-Start
Farbe	#ffffff
Hintergrund	Optionale Hintergrundfarbe
Sende beim Klicken:	
Payload	true
Topic	Test Taster
Emuliere einen Klick bei einer eingehenden Nachricht: <input type="checkbox"/>	
Name	Button

Abbildung 4.20: Die Eigenschaften des *button*-Nodes

Die Parametrisierungsoptionen sind folgende:

- **Gruppe** – bestimmt den Tab und die Gruppe, in der das Widget erscheinen soll.
- **Größe** – legt die Größe des Widgets in Rasterkästchen fest. **Auto** bedeutet eine Größe von Gruppenbreite $\times n$ Rasterkästchen. Innerhalb der Grenzen der Gruppengröße sind andere Maße möglich (z. B. 2×2).
- **Icon** – Einem Button lässt sich ein Icon zuordnen (siehe Abschnitt 4.4.1). In diesem Beispiel ist es das Icon *fa-star* aus der Sammlung *Font Awesome*.
- **Beschriftung** – Hiermit können Sie das Widget beschriften. Die Beschriftung erfolgt in Großbuchstaben.
- **Tooltipp** – Legen Sie hier fest, welche Meldung im Dashboard erscheint, wenn das Widget mit dem Mauszeiger den Fokus erhält.
- **Farbe** – Das ist die hexadezimal codierte RGB-Schriftfarbe des Widgets (z. B. entspricht #ffffff der Farbe Weiß).
- **Hintergrund** – Hier definieren Sie die hexadezimal codierte RGB-Hintergrundfarbe des Widgets.
- **Sende beim Klicken** – Hier legen Sie die Payload und das Topic fest, die gesendet werden, wenn der Button im Dashboard angeklickt wird.
 - **Payload** – Der Nachrichteninhalt kann mit Ausnahme von Umgebungsvariablen alle sonstigen Formate haben (Zeichenkette, Zeitstempel etc.).
 - **Topic** – das Nachrichtenthema.
- **Emuliere einen Klick ...** – Üblicherweise löst ein Klick auf das Widget den Flow aus. Den gleichen Effekt erzielt der Eingang einer Nachricht am Eingangsport des *button*-Nodes, wenn dieses Kontrollkästchen aktiviert ist. Es handelt sich um sogenannte *programmgesteuerte Klicks*.
- **Name** – die Bezeichnung des Nodes im Node-RED-Editor.

4.5.2 Der dropdown-Node

Der *dropdown*-Node erzeugt ein Auswahllisten-Widget. Abbildung 4.21 zeigt die Einstellungsoptionen.

Folgende Optionen stehen Ihnen zur Verfügung:

- **Group** – Hier legen Sie den Tab und die Gruppe fest, in der das Widget erscheinen soll.
- **Size** – die Größe des Widgets in Rasterkästchen. **Auto** bedeutet eine Größe von Gruppenbreite $\times 1$ Rasterkästchen. Innerhalb der Grenzen der Gruppengröße sind andere Maße möglich (auch z. B. ein Schrumpfen auf 2×1).
- **Label** – Das ist die Beschriftung des Widgets.
- **Tooltipp** – Diese Meldung erscheint im Dashboard, wenn das Widget mit dem Mauszeiger den Fokus erhält.

- **Placeholder** – Der Platzhalter steht im Zusammenhang mit dem Kontrollkästchen **If msg arrives on input ...**, mit dem Sie festlegen, ob die Message weitergeleitet werden soll. Bei aktiviertem Kontrollkästchen erscheint der Platzhalter im Dashboard. Eine Ausgabe am Ausgangsport unterbleibt jedoch.
- **Options** – enthält die Auswahlliste. Einzelne Einträge fügen Sie mit **+ option** hinzu. Jeder Listeneintrag besteht aus der zu sendenden Nachricht – dies kann eine Zeichenkette, eine Zahl oder ein boolescher Wert sein (z. B. »Hallo«) – und der Bezeichnung in der Auswahlliste (z. B. »Option 1: Gruß«).
- **Allow multiple selection** – erlaubt eine Mehrfachauswahl. msg-Payload ist dann eine Objekt-Tabelle.
- **If msg arrives on input ...** – Diese Option müssen Sie aktivieren, wenn Sie einen Platzhalter nutzen wollen.
- **Name** – Bezeichnung des Nodes im Node-RED-Editor.

The screenshot shows the configuration panel for a dropdown widget in Node-RED. The fields are as follows:

- Group:** [04 Dashboard Beispiele (einfach)] Eingabe-Elementer
- Size:** Auto
- Label:** Auswahlliste
- Tooltip:** optional tooltip
- Placeholder:** optional placeholder
- Options:** A list of three options:
 - Option 1: Gruß (with selected value 'Hallo')
 - Option 2: Glückszahl (with selected value '77')
 - Option 3: boolesche Var (with selected value 'true')
- Allow multiple selections from list:**
- If msg arrives on input, pass through to output:**
- Topic:** Auswahlliste
- Name:** Auswahlliste

Abbildung 4.21: Die Eigenschaften des dropdown-Nodes

Im Flow-Design-Bereich des Node-RED-Editors erscheint die Nachricht der jeweils getätigten Auswahl (z. B. »Hallo«), sofern Sie die Ausgabe einer Statusmitteilung über **Hauptmenü • Einstellungen • Ansicht** aktiviert haben (siehe Kapitel 2, »Das zentrale Tool: der Node-RED-Editor«).

4.5.3 Der switch-Node

Der *switch*-Node erfüllt die Funktionalität eines Schalters. Abbildung 4.22 zeigt die Konfigurationsmöglichkeiten.

The screenshot shows the configuration interface for a switch node in Node-RED. The settings are as follows:

- Group:** [04 Dashboard Beispiele (einfach)] Eing
- Size:** 4 x 1
- Label:** Schalter 2
- Tooltip:** optional tooltip
- Icon:** Custom (with Animate checked)
- On Icon:** fa-power-off (Colour: #f00000)
- Off Icon:** fa-power-off (Colour: #00ff00)
- Pass through msg if payload matches new state:**
- When clicked, send:**
 - On Payload:** a₂ An
 - Off Payload:** a₂ Aus
 - Topic:** a₂ Test Schalter
- Name:** Schalter Custom

Abbildung 4.22: Die Eigenschaften des switch-Nodes

Die Parametrisierungsoptionen entsprechen in weiten Teilen denen beim *button*-Node:

- **Group** – der Tab und die Gruppe, in der das Widget erscheinen soll.
- **Size** – die Größe des Widgets in Rasterkästchen. **Auto** bedeutet eine Größe von Gruppenbreite × 1 Rasterkästchen. Innerhalb der Grenzen der Gruppengröße sind andere Maße möglich (z. B. 2 × 2).
- **Label** – Das ist der Text auf dem Widget. Er erscheint in Großbuchstaben.
- **Tooltip** – Diese Meldung erscheint im Dashboard, wenn das Widget mit dem Mauszeiger den Fokus erhält.
- **Icon** – Sie haben die Auswahl zwischen **Default** oder **Custom**, um das Icon (siehe Abschnitt 4.4.1) und die Farbe weiter anzupassen.
- **Pass through msg if payload ...** – leitet die Nachricht vom Eingangs-Node weiter.
- **When clicked, send** – Bei dieser Option bestehen folgende Einstellungsmöglichkeiten:
 - **On Payload** – Der Nachrichteninhalt kann mit Ausnahme von Umgebungsvariablen alle sonstigen Formate haben (Zeichenkette, Zeitstempel etc.)

- **Off Payload** – wie **On Payload**.
- **Topic** – das Nachrichtenthema.
- **Name** – die Bezeichnung des Nodes im Node-RED-Editor.

4.5.4 Der slider-Node

Der *slider*-Node ist mit einem Schieberegler vergleichbar, der numerische Werte innerhalb eines definierten Bereichs ausgibt. Abbildung 4.23 zeigt die entsprechenden Node-Eigenschaften.

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

- Group:** [04 Dashboard Beispiele (einfach)] Eing
- Size:** Auto
- Label:** slider
- Tooltip:** optional tooltip
- Range:** min 0, max 10, step 1
- Output:** continuously while sliding
- If msg arrives on input, pass through to output:**
- When changed, send:**
- Payload:** Current value
- Topic:** Test Schieberegler
- Name:** Schieberegler

Abbildung 4.23: Die Eigenschaften des slider-Nodes

Als Node-Eigenschaften können Sie festlegen:

- **Group** – der Tab und die Gruppe für das Dashboard.
- **Size** – die Größe des Widgets in Rasterkästchen. **Auto** bedeutet eine Größe von Gruppenbreite \times 1 Rasterkästchen.
- **Label** – der Text für das Widget. Er erscheint in Großbuchstaben.
- **Tooltip** – Diese Meldung erscheint im Dashboard, wenn das Widget mit dem Mauszeiger den Fokus erhält.
- **Range** – Das ist der Nummernbereich in einer Spanne von **min** bis **max** und einer Schrittweite von **step**.
- **Output** – bestimmt die Art der Ausgabe:
 - **continuously while sliding** – gibt die Nachricht bei jedem Schritt aus.
 - **only on release** – wartet auf das Ende des Schiebевorgangs und gibt erst dann die Nachricht aus.

- **If msg arrives on input ...** – leitet die Nachricht vom Eingangs-Node weiter.
- **When changed, send:**
 - **Payload** – Der Nachrichteninhalt ergibt sich als numerischer Wert aus der Schiebereglerposition und der Output-Einstellung.
 - **Topic** – das Nachrichtenthema.
- **Name** – die Bezeichnung des Nodes im Node-RED-Editor.

4.5.5 Der numeric-Node

Der *numeric*-Node weist viele Übereinstimmungen mit dem *slider*-Node auf, allerdings erfolgt die Auswahl der Zahl ähnlich einer Drop-down-Liste. Insofern sind auch die Parameter, die Sie in Abbildung 4.24 sehen, weitgehend identisch.

The image shows the configuration interface for a numeric node in Node-RED. The settings are as follows:

- Group:** [04 Dashboard Beispiele (einfach)] Eing
- Size:** Auto
- Label:** numerische Eingabe
- Tooltip:** optional tooltip
- Value Format:** {{value}}
- Range:** min: 0, max: 10, step: (empty)
- Wrap value from max to min and min to max:**
- If msg arrives on input, pass through to output:**
- When changed, send:**
 - Payload:** Current value
 - Topic:** a_2 Test numeric-Node
- Name:** (empty)

Abbildung 4.24: Die Eigenschaften des numeric-Nodes

Der wesentliche Unterschied zum *slider*-Node ist das zusätzliche Kontrollkästchen **Wrap value ...**. Wenn Sie es aktivieren, stoppt die Auswahl nicht an den Bereichsendpunkten, sondern folgt einem nicht enden wollenden Nummernkreis (z. B. folgt nach 10 wieder 0).

4.5.6 Der text-input-Node

Im Dashboard sind Texteingaben über den *text-input*-Node möglich. Abbildung 4.25 zeigt die Konfigurationsoptionen.

Der *text-input*-Node weist gegenüber den anderen Dashboard-Eingabe-Nodes eine spezielle Besonderheit auf, nämlich die Option **Mode** (siehe Abbildung 4.25). Der Beispiel-

Flow setzt den *text-input*-Node an vier Stellen ein, und zwar bei der E-Mail-Eingabe, der Passwordeingabe, der Farbauswahl und der Texteingabe. Testen Sie die Wirkung der Modi im Dashboard.

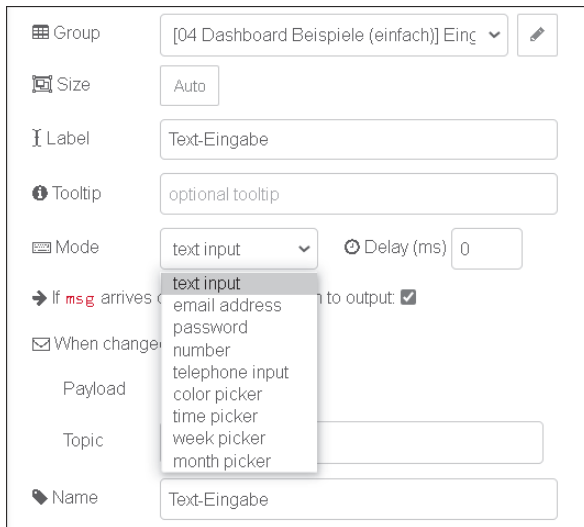


Abbildung 4.25: Die Eigenschaften des *text-input*-Nodes

Die Angabe einer Verzögerung bei **Delay** bewirkt, dass der eingegebene Text nicht mehr in seiner Gesamtheit ausgegeben wird, sondern in Zeichenfolgen, die jeweils alle bislang erfassten plus das aktuelle Zeichen umfassen (z. B. h, he, heu, heut, heute).

Im Beispiel-Flow habe ich einen *text-input*-Node mit einem *audio-out*-Node verbunden. Die Eigenschaften sehen Sie in Abbildung 4.26.

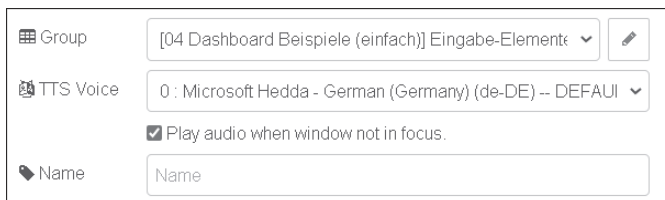


Abbildung 4.26: Die Eigenschaften des *audio-out*-Nodes

4.5.7 Der *form*-Node

Eingabeformulare gestalten Sie mit dem *form*-Node, dessen Optionen Sie in Abbildung 4.27 sehen.

Die Wirkungsweise der Optionen **Gruppe**, **Größe** und **Beschriftung** sowie **Topic** und **Name** ist identisch mit denen der bisher erörterten Nodes.

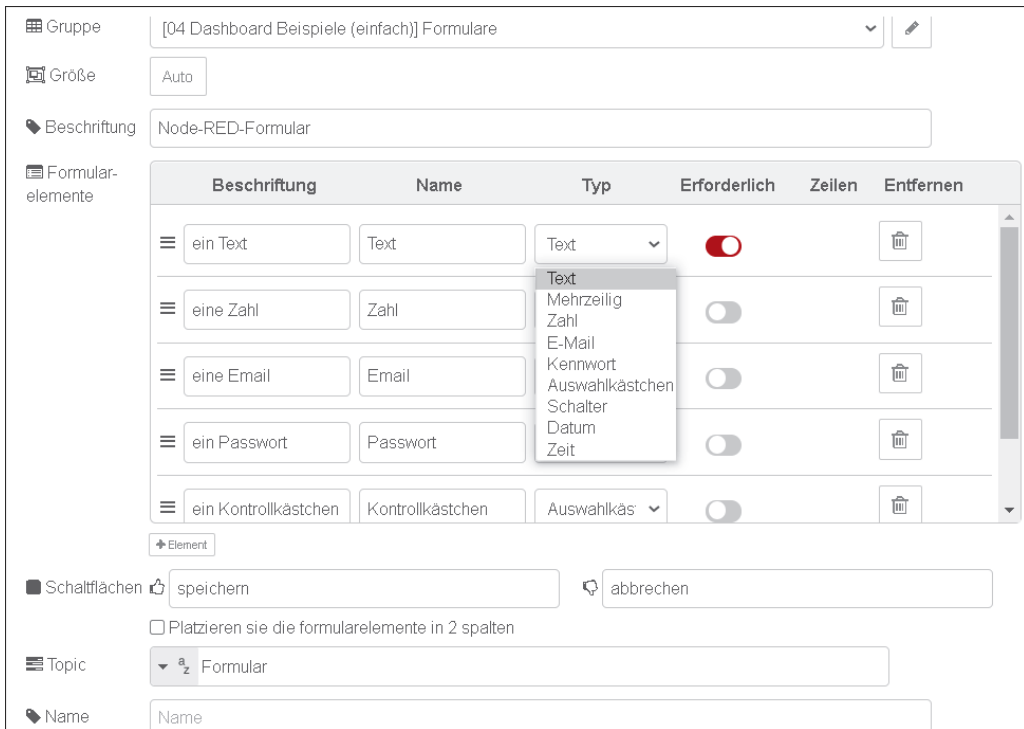


Abbildung 4.27: Die Eigenschaften des form-Nodes

Die Formularelemente sind in einer Liste organisiert. + **Element** fügt ein Element hinzu; ein Klick auf das Papierkorb-Icon löscht den Eintrag.

Die Spaltenüberschriften sind weitgehend selbsterklärend. Das Aktivieren von **Erforderlich** hat zur Folge, dass ein Speichern nur bei gefülltem Dashboard-Listenelement möglich ist. Ein Stern neben der Dashboard-Beschriftung weist auf die Eigenschaft als Pflichtfeld hin (siehe Abbildung 4.28).

Ist der Elementtyp **Mehrzeilig**, können Sie unter **Zeilen** die Anzahl festlegen.

`msg.payload` ist hier ein Objekt, das die Listeneinträge enthält.

4.5.8 Die beiden function-Nodes und der template-Node

Der *function*-Node wird ausführlich in Kapitel 5, »Funktionen programmieren«, beschrieben, daher soll an dieser Stelle ein knapper Hinweis auf die Aufgaben der Nodes und die Angabe des Codes genügen.

Der erste *function*-Node (siehe Listing 4.1) fügt die `msg.payload` der eingegangenen Nachricht in einer Tabelle an, speichert die Tabelle und gibt sie zur Ausgabe weiter:

```

var dashboardLog = flow.get('dashboardLog')|| [];
dashboardLog.push(msg);
if (dashboardLog.length > 20){
    // Löschen der ältesten Nachricht bei mehr als 20 Einträgen
    dashboardLog.shift();
    dashboardLog.length = 20;
}
// Liste speichern
flow.set('dashboardLog', dashboardLog);
// Liste weitergeben
msg = {};
msg.payload = dashboardLog;
return msg;

```

Listing 4.1: Der erste function-Node

Der zweite *function*-Node löscht die gespeicherten Nachrichten und die Dashboard-Ausgabe:

```

flow.set('dashboardLog', []);
msg = {};
msg.payload = [];
return msg;

```

Listing 4.2: Der zweite function-Node

Der *template*-Node sorgt für die aufbereitete Darstellung der erhaltenen Nachricht:

```

<ul>
  <li ng-repeat="x in msg.payload">
    <font color="red">{{x.topic}}</font>
    <ul>
      <li>{{x.payload}}</li>
    </ul>
  </li>
</ul>

```

Listing 4.3: Der template-Node

Abbildung 4.28 zeigt den Flow in Aktion und führt die gezeigten Nodes zusammen.

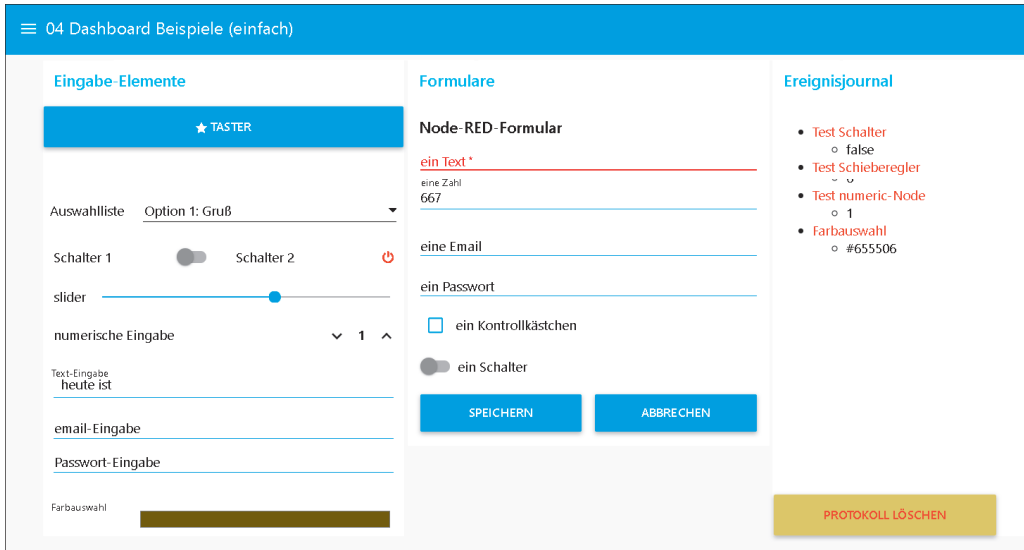


Abbildung 4.28: Der Beispiel-Flow im Dashboard

4.6 Charts und Messanzeigen mit dem Raspberry Pi

Häufig läuft Node-RED auf einem Raspberry Pi. Was liegt also näher, als mithilfe des Raspberry Pi eine kleine Indoor-Wetterstation zu realisieren? Das Beispiel umfasst auch, wie Sie die GPIOs des Raspberry Pi mit Node-RED steuern können.

Sie brauchen:

- einen Sensor BMP280 für die Temperatur- und Luftdruckmessung
- eine LED
- einen Widerstand (z. B. 470 Ω)
- ein Steckbrett und Jumper-Kabel

Abbildung 4.29 zeigt das Schaltbild, nach dem Sie den Sensor, die LED und den Raspberry Pi verkabeln müssen.

Da das Pin-out des Sensors abweichen kann, zeigt Tabelle 4.2 die Verdrahtung in einer Übersicht.

	RPi	BMP280	Farbe
GND	Pin 9 – GND	GND	Schwarz
VCC	Pin 1 – VCC	VCC	Rot

Tabelle 4.2: Die Verdrahtung des Pi mit dem BMP280

	RPi	BMP280	Farbe
SDA	Pin 3 – SDA1	SDA	Grün
SCL	Pin 5 – SCL1	SCL	Gelb

Tabelle 4.2: Die Verdrahtung des Pi mit dem BMP280 (Forts.)

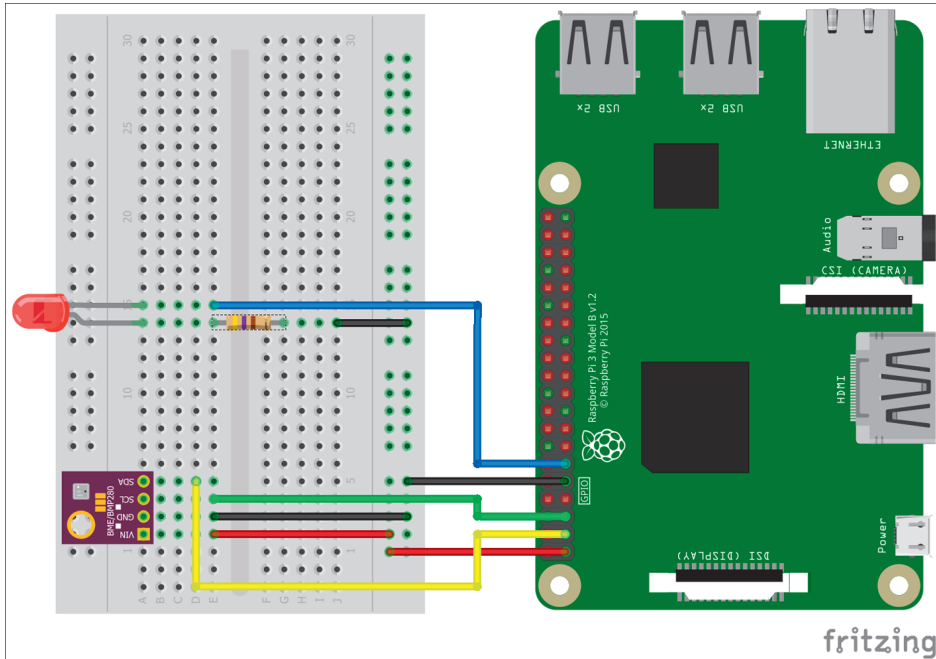


Abbildung 4.29: Schaltung für den Sensor BMP280

4.6.1 Sensoren schalten und das benötigte Paket installieren

Sie müssen zunächst den Sensor an den Raspberry Pi anschließen, dann das entsprechende Paket installieren und schließlich den Flow bauen. Gehen Sie das Projekt schrittweise an.

Schritt 1: Schaltung erstellen

Verbinden Sie die einzelnen Bauteile. Aber Achtung: Die Pin-Belegung des BMP280 kann vom Schaltplanbeispiel abweichen: SCL (BMP280) gehört an RPi-GPIO3 (SCL1), SDA (BMP280) an RPi-GPIO2 (SDA1).

Inhalt

Materialien zum Buch	13
Einleitung	15
1 Node-RED – das Setup: So starten Sie	23
1.1 Node-RED – das zentrale Element	23
1.2 Node-RED aufsetzen	25
1.2.1 Node-RED auf einem Android-Gerät	25
1.2.2 Node-RED auf einem Windows-PC	25
1.2.3 Node-RED auf einem Raspberry Pi	29
1.2.4 Node-RED auf einem Linux-System	32
1.3 Node-RED überall: Docker-Container starten	34
1.3.1 Docker auf einem Ubuntu-System	34
1.3.2 Docker auf dem Raspberry Pi	35
1.3.3 Node-RED in Docker ausführen	36
1.4 Die Ausgaben von Node-RED beim Start	38
1.5 Node-RED administrieren	39
1.5.1 Dateien und Ordner	39
1.5.2 Dateien unter Docker	41
1.5.3 Einstellungen von Node-RED ändern	41
1.6 Node-RED absichern	43
1.6.1 Passwortschutz für den Node-RED-Editor	44
1.6.2 HTTPS aktivieren	45
1.6.3 Sicherheit für den http-in-Node	50
1.6.4 Die Konfigurationsdatei absichern	50
1.7 Node-RED Projekte	51
1.8 Fazit	60
2 Das zentrale Tool: der Node-RED-Editor	61
2.1 Den Node-RED-Editor in einem Browser öffnen	61
2.2 Die Kopfleiste	64
2.2.1 Die Schaltfläche deploy	64
2.2.2 Die Benutzerauthentifikation	66
2.2.3 Das Hauptmenü	66

2.3	Die linke Seitenleiste	74
2.3.1	Der Flow-Explorer	75
2.3.2	Die Node-Palette	75
2.4	Der Arbeitsbereich	83
2.4.1	Die Verwaltungsleiste für die Flows	84
2.4.2	Der Flow-Design-Bereich	85
2.4.3	Die Fußleiste des Arbeitsbereichs	91
2.4.4	Das Kontextmenü	92
2.5	Die rechte Seitenleiste	93
2.5.1	Das Register info	93
2.5.2	Das Register Hilfe	95
2.5.3	Das Register Debugging	95
2.5.4	Das Register Konfiguration	96
2.5.5	Das Register Kontext	97
2.5.6	Das Register Dashboard	97
2.6	Der erste Flow	97
2.7	Flows mit AI entwickeln	99
2.8	Gute Programmierung	101
2.8.1	Die Flow-Struktur	101
2.8.2	Das Message-Design	108
2.8.3	Die Dokumentation	108
2.9	Flows mit dem Flow-Debugger debuggen	109
2.9.1	Schrittweises Durchlaufen eines Flows	111
2.9.2	Messages löschen	111
2.9.3	Breakpoints nutzen	112
2.10	Probleme mit nrlinter aufspüren	112
2.10.1	Installation	113
2.10.2	Mit nrlinter arbeiten	113
2.11	Fazit	115
3	Das Fundament: die Basics von Node-RED	117
3.1	Das Message-Konzept von Node-RED	117
3.1.1	JSON – das Datenformat für den Datenaustausch	117
3.1.2	Messages in Node-RED	121
3.2	Die Geschwister inject-Node und debug-Node	125
3.2.1	Der inject-Node	125
3.2.2	Der debug-Node	127

3.3	Messages manipulieren: Der change-Node und seine Begleiter	129
3.3.1	Der switch-Node	130
3.3.2	Der change-Node	131
3.4	Der delay-Node	133
3.5	Dateiformate konvertieren	133
3.6	Auf Prozessereignisse reagieren	135
3.6.1	Der status-Node	135
3.6.2	Der complete-Node	136
3.6.3	Der catch-Node	137
3.7	Sequenzen (Folgen)	139
3.8	Fazit	144
4	Das Node-RED-Dashboard	145
4.1	Installation	145
4.2	Browsersaufruf und Einstellungen	147
4.3	Der Schnelleinstieg: So erstellen Sie Ihre erste Dashboard-Ausgabe	147
4.4	Das Dashboard-Design bestimmen	150
4.4.1	Icons	151
4.4.2	Die rechte Seitenleiste	152
4.4.3	Custom CSS	156
4.4.4	Die Konfiguration von Tabs und Gruppen	156
4.5	Die Dashboard-Widgets in Aktion	158
4.5.1	Der button-Node	159
4.5.2	Der dropdown-Node	160
4.5.3	Der switch-Node	162
4.5.4	Der slider-Node	163
4.5.5	Der numeric-Node	164
4.5.6	Der text-input-Node	164
4.5.7	Der form-Node	165
4.5.8	Die beiden function-Nodes und der template-Node	166
4.6	Charts und Messanzeigen mit dem Raspberry Pi	168
4.6.1	Sensoren schalten und das benötigte Paket installieren	169
4.6.2	Die LEDs schalten	171
4.6.3	Wetterdaten erheben und ausgeben	171
4.7	Das Diagramm-Kaleidoskop	177

4.8	Das Dashboard 2.0	178
4.8.1	Installation	179
4.8.2	Der Schnelleinstieg	179
4.9	Fazit	183
5	Funktionen programmieren	185
5.1	Einsatz und Funktionsweise des function-Nodes	185
5.1.1	Eingangsnachrichten lesen	186
5.1.2	Nachrichten erstellen	188
5.1.3	Code zur Ausführung bei setup und close	191
5.1.4	Ereignisse loggen	192
5.1.5	Das Erscheinungsbild ändern	193
5.2	Programmierung mit JavaScript	193
5.2.1	Codeeditoren	193
5.2.2	Zeichen, Kommentare und Begriffe	194
5.2.3	Daten und Datentypen	196
5.2.4	Variablen und Konstanten	197
5.2.5	Ausdrücke und Operatoren	202
5.2.6	Das Array-Objekt (Tabellen)	209
5.2.7	Das Date-Objekt	211
5.2.8	Funktionen	216
5.2.9	Kontrollstrukturen	218
5.3	Programmbeispiele für den function-Node	227
5.3.1	Eine Zeichenfolge aufteilen	228
5.3.2	Eine Nachricht verzögern	228
5.3.3	Auf eine Umgebungsvariable zugreifen	229
5.3.4	Zusätzliche Module nachladen	229
5.3.5	Mit Kontextvariablen arbeiten	230
5.3.6	Nachrichten zusammenführen	231
5.3.7	Mit Puffern arbeiten	232
5.4	Externe Module laden	233
5.4.1	Den Hostnamen ausgeben	234
5.4.2	Den RGB-Farbwert prüfen	234
5.5	Der Monaco-Codeeditor	236
5.6	Fazit	236

6	Daten über Netzwerkprotokolle abrufen	237
6.1	Daten von einem Server im Netz abrufen	237
6.1.1	Grundlagen von HTTP-Verbindungen	238
6.1.2	Die Nodes nutzen	242
6.2	MQTT: das IoT-Protokoll	263
6.2.1	Installation und ein Flow für den Einstieg	265
6.2.2	Einen eigenen MQTT-Broker aufsetzen	269
6.2.3	Node-RED an den Mosquitto-Broker anbinden	272
6.2.4	Der Shelly 1-Schalter	272
6.3	Arduino & Co. über USB anbinden	275
6.4	Fazit	282
7	Daten mit Node-RED teilen	283
7.1	E-Mails versenden	283
7.1.1	Das E-Mail-Konto konfigurieren	284
7.1.2	Der E-Mail-Versand	285
7.1.3	Der E-Mail-Empfang	290
7.2	Instant-Messaging und Bots mit Telegram	293
7.2.1	Instant Messaging	294
7.2.2	Bots	294
7.2.3	Telegram	295
7.3	Geräte mit Pushbullet vernetzen	310
7.3.1	Pushbullet einrichten	311
7.3.2	Erste Schritte mit der Pushbullet-API	313
7.3.3	Pushbullet für Node-RED	316
7.4	Sprachsteuerung mit Alexa	320
7.5	Google-Dienste integrieren	326
7.6	Mit künstlicher Intelligenz experimentieren	333
7.6.1	OpenAI-Account und API-Key erstellen	334
7.6.2	Ein Bild erstellen	334
7.6.3	Eine Textnachricht generieren	336
7.7	Fazit	338
8	Daten speichern und archivieren	339
8.1	Kontextvariablen	339
8.1.1	Kontextvariablen vom Typ node	340

8.1.2	Kontextvariablen vom Typ flow	341
8.1.3	Kontextvariablen vom Typ global	343
8.1.4	Kontextvariablen im Dateisystem speichern	344
8.2	Daten in Dateien speichern	345
8.2.1	Messdaten speichern und wieder auslesen	346
8.2.2	Dateimanager	352
8.3	Node-RED und InfluxDB	356
8.3.1	InfluxDB, eine Time Series Database	357
8.3.2	InfluxDB installieren	359
8.3.3	Die ersten Schritte mit InfluxDB	361
8.3.4	Mit Node-RED Daten in InfluxDB speichern	366
8.3.5	Mit Node-RED Daten aus der InfluxDB auslesen	369
8.3.6	Die InfluxDB sauber halten	373
8.4	Node-RED und SQLite	376
8.4.1	Aufbau einer SQLite-Datenbank	376
8.4.2	SQLite installieren	379
8.4.3	Mit Node-RED Daten in der SQLite-Datenbank speichern	382
8.4.4	Mit Node-RED Daten aus der SQLite-Datenbank löschen	384
8.4.5	Mit Node-RED Daten aus der SQLite-Datenbank auslesen	385
8.5	Fazit	385
9	Node-RED-Hacks	387
9.1	Python-Skripte einbinden	387
9.2	Timer	389
9.2.1	Ausgaben	391
9.2.2	Timersteuerung	392
9.2.3	Erweiterte Möglichkeiten	392
9.3	»Himmelserscheinungen« auswerten	393
9.4	Wetterdaten mit OpenWeatherMap	395
9.4.1	Das openweathermap-Konto	396
9.4.2	Eine Wetteransage	397
9.4.3	Ein Frostwächter	399
9.5	Zeitangaben formatieren	399
9.5.1	Der Node Date/Time Formatter	400
9.5.2	Der humanizer-Node	402
9.6	Mit Bilddateien arbeiten	402
9.7	Einen QR-Code generieren	404

9.8	Geräte mit Ping orten	406
9.8.1	Anwesenheitsmitteilung senden	406
9.8.2	Alarmanlage aktivieren	408
9.8.3	Erreichbarkeit eines Servers überprüfen	408
9.9	Auf eine FRITZ!Box zugreifen	410
9.9.1	Anwesenheitsbenachrichtigung	411
9.9.2	Benachrichtigung bei Anrufen	413
9.9.3	Gastzugang schalten	414
9.10	FTP – Daten zwischen Rechnern übertragen	416
9.10.1	Einen FTP-Server aufsetzen	417
9.10.2	Das Verzeichnis mit Node-RED lesen	419
9.10.3	Node-RED-Konfigurationsdateien sichern	421
9.11	Benachrichtigungen mit Ntfy versenden	423
9.11.1	Nutzerkonto erstellen	423
9.12	Fazit	426
10	Apps und externe Anbindung	427
10.1	Apps aus den App-Stores	427
10.2	Blynk 2.0	428
10.2.1	Mit Blynk auf den Raspberry Pi zugreifen	428
10.2.2	Blynk und Node-RED kommunizieren miteinander	438
10.3	Die Termux-App	444
10.3.1	Node-RED auf dem Android-Gerät	444
10.3.2	Die Termux:API	446
10.3.3	Die Termux:API-App mit Node-RED nutzen	449
10.4	Der »Überall-Zugriff« mit ngrok	451
10.5	Fazit	457
11	Dashboards für Fortgeschrittene	459
11.1	Dynamische Dashboard-Steuerung	459
11.2	Der template-Node (Widget)	461
11.2.1	Einfache (statische) HTML-Ausgaben mit dem ui-template-Node ...	462
11.2.2	Den Eingabeport nutzen	467
11.2.3	ui-template-Node – den Ausgabeport nutzen	475
11.2.4	Statusinformationen im Dashboard-Header ausgeben	478
11.3	Ein aufwendigeres Beispiel	479

11.4	Ein alternatives Dashboard mit uibuilder	482
11.5	Fazit	488
12	Node-RED in andere Dienste integrieren	489
12.1	ioBroker	489
12.1.1	Installation und Inbetriebnahme	490
12.1.2	ioBroker-Objekte: So schalten Sie eine LED	493
12.1.3	ioBroker und Node-RED	495
12.2	Node-RED versus externe Dienste	500
12.3	Fazit	502
13	Eigene Nodes erstellen	503
13.1	Anforderungen definieren	503
13.2	Arbeitsverzeichnis erstellen und ausgestalten	504
13.3	Die Datei package.json generieren	505
13.4	Die Datei <node>.js programmieren	506
13.4.1	Der Rahmen	506
13.4.2	Den Rahmen ausfüllen	507
13.5	Ein Icon erstellen	511
13.6	Die Datei basic-math.html generieren	511
13.7	Den Node basic-math in Node-RED testen	514
13.8	Ausblick	518
13.9	Fazit	518
14	Mit Node-RED Mikrocontroller programmieren ...	519
14.1	Der Soft- und Hardwarerahmen	520
14.2	Installation von Moddable	520
14.3	Den Mikrocontroller an Moddable anbinden	523
14.4	Einrichtung in Node-RED und Test	525
14.5	Fazit	528
	Index	529

Index

.npm (Ordner) 40
 ~/.node-red 39

A

Action list 67
 Administration 39
 Agent 100
 AJAX 260
 Alarmanlage (Beispiel) 408
 Alexa 320
 Alexa Home Skill Bridge 322
 alexa-home-Node 325
 Altitude 395
 Amazon AWS 269
 Amazon Web Services 25
 American Standard Code for Information
 Interchange 195
 Android → Termux
 Angular 461, 472
 Angular-Direktive 476
 md-toolbar 479
 Angular-Filter 474
 AngularJS 461
 Angular-Material 461
 API 23, 253
 Application Programming Interface 23
 apt 445
 Arbeitsbereich 83
 Arduino 275
 Arduino Nano 276
 Array 209
 ASCII 195, 233
 Ausdruck 202
 AVM 410
 Azimuth 395

B

Backup 39, 421
 base64-Node 256
 BeagleBone Black 25
 Begrenzer 196
 Benutzerauthentifikation 64
 Benutzername (Node-RED) 42
 Bezeichner 196
 BH1750 276

Bibliothek 69
 Bigtimer 389
 Bilddateien 402
 Bitcoin-Kurs (Beispiel) 253
 Bitoperationen
 AND-Verknüpfung 206
 Linksverschiebung 207
 Negation 207
 OR-Verknüpfung 207
 Rechtsverschiebung 207
 XOR-Verknüpfung 207
 Bixby 320
 Blynk
 App 430, 433, 434, 436
 Raspberry Pi 428
 Bme280-Node 170, 171
 BMP280 169, 327
 Boolescher Ausdruck 219
 BootstrapVue 487
 BotFather 297, 304
 Bots 293
 break 222
 Breakpoint 109
 Buffer 123, 232
 button-Node 159, 460

C

callmebot 283
 Callmonitor 411
 Carriage Return 240
 Cascading Style Sheets 260
 Casten 365
 catch-Node 137
 Cayenne 269
 CCU.IO 489
 Certificate Signing Request 47
 Cgroups 34
 change-Node 129, 131
 chart-Node 252
 Chat-ID (Telegram) 302
 Chromium 61
 Client-Server-Prinzip 238
 Comet 260
 command-Node 306
 comment-Node 102, 109
 complete-Node 136

- continue 223
 - Continuous Querys 372
 - Cortana 320
 - credentialSecret 42
 - CSV-Format 140
 - csv-Node 140
 - Custom authentication token 50
- D**
-
- Dashboard 68, 97, 145, 146
 - audio-out-Node 165
 - Bild anzeigen 463
 - button-Node 159
 - Charts und Messanzeigen 168
 - Datumsformat 155
 - dropdown-Node 160
 - dynamische Steuerung 459
 - form-Node 165
 - Google Maps 465
 - Header 478
 - HTML-Ausgabe 463
 - Icons 151
 - Layout 153
 - Links einbinden 466
 - numeric-Node 164
 - Raspberry Pi 168
 - Site 155
 - slider-Node 163
 - Statusinformationen 478
 - switch-Node 162
 - Tabs und Gruppen 156
 - text-input-Node 164
 - Theme 156
 - Widgets 158
 - dashboard-switch-Node 414, 460
 - Dashboard-Tab 149
 - DataURI 405
 - Date/Time-Formatter-Node 400
 - Dateiformate (CSV, JSON, XML) 133
 - Dateimanager 352
 - Datenbanken
 - InfluxDB 356
 - SQLite 376
 - Zeitreihen 339, 356
 - Datentypen 196
 - Debian 32
 - Debug
 - Einstellungen 128
 - Fenster 128
 - Debugger 95, 109
 - Debugnachrichten 68
 - debug-Node 41, 97, 262, 309
 - Ausgabe 127
 - über 128
 - Decoration 87, 193
 - Deployment 64
 - Dienstprogramm 25
 - Docker 34
 - Dateisystem 41
 - Node-RED 36
 - Raspberry Pi 35
 - Ubuntu 34
 - Document Object Model 260
 - Dokumentation 108
 - dropdown-Node 160, 355
- E**
-
- editorTheme 236
 - Eigene Nodes
 - Anforderungen 503
 - Arbeitsverzeichnis 504
 - Icon 511
 - JavaScript 506
 - Main-Node-Definition 511
 - Nodes testen 514
 - package.json 503, 505
 - E-Mail 283
 - Anlage 288, 292
 - Empfang 290
 - HTML-Anweisungen 286
 - Versand mit Node-RED 285
 - email-in-Node 291, 292
 - email-out-Node 285, 287, 289, 291
 - Errors in Subflows 139
 - ESP32 269
 - ESP8266 269
 - every() 226
 - exec-Node 387, 422
 - Exportieren 71
 - Externe Module 233
- F**
-
- Facebook 294
 - false 219
 - Ferne (remote) 58
 - FHEM 311
 - Field-Keys 359
 - Field-Set 359
 - File Transfer Protocol 416

- file-in-Node 259, 289, 348
 - file-Node 347, 351
 - find() 227
 - findIndex() 227
 - Firmata 281
 - Flow-basierte Programmierung 24
 - Flow-Design-Bereich 85
 - Flow-Explorer 75
 - FlowFuse 145
 - FlowFuse Dashboard 178
 - Flows 24
 - Anordnung der Nodes* 102
 - Best Practices* 101
 - Debugger* 109
 - dokumentieren* 108
 - erstellen* 85
 - Gruppen* 106
 - Hallo Welt* 97
 - Linten* 112
 - Message-Design* 108
 - quick-add-Dialog* 86
 - schrittweise durchlaufen* 111
 - Struktur* 101
 - Subflows* 104
 - Verbindungen ändern* 97
 - Verwaltungsleiste* 84
 - Zuordnung* 102
 - flows (Ordner) 40
 - flows_raspberrypi_cred.js 40
 - flows.json 40
 - Flux 364
 - Folgen 139
 - forEach() 226
 - form-Node 165
 - for-Schleife 224
 - Forum 16
 - Fremdschlüssel 378
 - FRITZ!Box 62, 455
 - Anwesenheitsbenachrichtigung* 411
 - Callmonitor* 411
 - Gastzugang* 414
 - Nodes* 410
 - fritzbox.callmonitor-Node 413
 - fritzbox-lookup-Node 414
 - fritzbox-Node 411, 414
 - Frostwächter (Beispiel) 399
 - fs-file-lister-Node 352
 - FTP 416
 - ftp-in-Node 420
 - Function Expression 217
 - functionExternalModules 235
 - functionGlobalContext 233
 - function-Node 158, 185, 289, 290
 - asynchrone Ausgabenachrichten* 191
 - Beispiele* 227
 - Eigenschaften* 185
 - Funktionen 216
 - Funktionsausdruck 217
 - Funktionsreferenz 124
 - Fußleiste 91
- ## G
-
- gauge-Node 173, 254
 - GitHub 55
 - Git-Repository 51
 - Globale-Konfigurations-Nodes 75
 - Google 326
 - Google Assistant 320
 - Google Cloud Platform 326
 - Google Drive 327
 - Google Maps 465
 - Go-Stil 365
 - Grafana 177, 477
 - graph-Node 174, 347
 - Groups 73
 - Gruppen 106, 151
- ## H
-
- Hauptmenü 64, 66
 - Header 478
 - Hoisting 200
 - Home Assistant 15
 - Hostname 234, 243
 - HTML 461
 - html-Node 251
 - HTTP 45, 423
 - Anfrage* 240
 - Authentifizierung* 254
 - Protokoll* 239
 - http-in-Node 50, 257, 258
 - httpNodeAuth 50
 - http-request-Node 247, 250, 253, 254, 257, 320, 332
 - http-response-Node 257–259, 261
 - HTTPS 45
 - humanizer-Node 402
 - Hyper-V 25

I

I2C-Bus	237, 276
IBM bluemix	269
Icons	
<i>Angular Material Icons</i>	152
<i>Font Awesome</i>	151
<i>Material Design Iconfont</i>	152
<i>Weather Icons Lite</i>	152
ICQ	294
ifconfig	62
image-Node	403
IMAP	285, 291
Import	68
influx	361
Influxd	361
InfluxDB	351, 356
<i>Datenbank anlegen</i>	361
<i>Datenbank auswerten</i>	364
<i>Installation</i>	359
influxdb-in-Node	369
influxdb-out-Node	367
InfluxQL	364
Information Packets	24
inject-Node	97, 125, 256, 262, 354, 422
<i>Message</i>	126
<i>Wiederholungen</i>	126
<i>Zeitpunkt</i>	126
Instant Messaging	294
Internet of Things	263
ioBroker	
<i>Adapter</i>	490
<i>Bedienung</i>	491
<i>Instanzen</i>	493
<i>Node-RED</i>	495
<i>Nodes</i>	497
iobroker-get-Node	498
IP-Adresse	62, 451
ipconfig	62
IP-Symcon	489
isnan()	220

J

JavaScript	15, 185, 193, 260
JavaScript Object Notation	117
JavaScript-Objekt	117
join-Node	142
Joins	364
JQuery	461
JSfiddle	193

JSON	117
<i>Ausdruck</i>	123
<i>boolescher Wert</i>	118
<i>Liste</i>	118
<i>Name-Wert-Paare</i>	118
<i>Nullwert</i>	118
<i>Objekte</i>	119
<i>Tabellen</i>	119
<i>Zeichenketten</i>	119
JSONata	123, 185
JSONata-Ausdruck	131, 176, 246, 343, 474
JSONata-Ausdruckseditor	124
JSON-Format	301
JSON-Formatter	69
JSON-Nodes	134
JSON-Objekt	403, 460

K

Knoten	24
Kommentare	195
Konfiguration	96
Konfigurationsdatei	41
Konfigurations-Node	72, 76, 95, 146
Konstanten	202
Kontext	97
Kontextvariablen	192, 230, 339
<i>flow</i>	341, 348
<i>global</i>	343
<i>node</i>	340, 354
Kontrollstrukturen	
<i>do-while-Schleife</i>	223
<i>for-Schleife</i>	224
<i>if-Bedingung</i>	219
<i>if-else</i>	220
<i>switch ... case</i>	221
<i>while-Schleife</i>	222
Kopfbereich	64
Künstliche Intelligenz	333

L

Line Feed	240
Link	151
link-in-Node	102, 103
link-out-Node	102
Linksverschiebung	208
Linters	113
Linux	32, 269
localhost	243
Log-Level	42

- Long Polling 260
 - lower-case-Node 525
 - Luftqualität (Beispiel) 249
- ## M
-
- MAC-Adresse 411
 - macOS 269
 - Main-Node-Definition 511
 - Measurement 359
 - Message Queue Telemetry Transport 263
 - Message-Objekte 126
 - Messages 108, 117
 - Payload* 121
 - Messenger-Dienste 283
 - Messungen 357
 - Microsoft Azure 25
 - Mikrocontroller-Simulator 522
 - Moddable 520
 - SDK* 520
 - Model Context Protocol (MCP) 100
 - Module, externe 233
 - Modulo 203
 - Monaco 236
 - Mosquitto 269
 - MQTT
 - Clients* 265
 - Last Will* 265
 - Protokoll* 263
 - Publish* 264
 - Quality of Service* 265
 - Subscribe* 264
 - Testament* 265
 - Topic* 265
 - MQTT-Broker 264
 - mqtt-out-Nodes 268
 - msgid 121
 - msg-Konzept 117
 - Mustache 467
 - Hash* 468
 - Sektoren* 468
 - MyFRITZ! 451, 455
- ## N
-
- Nachrichten-Objekt 127
 - Namespaces 34
 - Netzwerk
 - Client-Server-Prinzip* 238
 - Protokolle* 237
 - Topologien* 237
 - ngrok 451
 - Authtoken* 453
 - ngrok-Account* 452
 - Node RED Client Editor 427
 - node_modules 39
 - Node.js 15, 26
 - Nodemailer-Format 288
 - Node-RED
 - absichern* 43
 - administrieren* 39
 - Blog* 74
 - Dashboard* 145
 - deinstallieren* 40
 - Dokumentation* 518
 - Einstellungen* 41
 - Installation* 25
 - Messages* 117
 - Projekte* 51
 - Raspberry Pi* 29
 - Was ist Node-RED* 23
 - Webseite* 74
 - Windows* 27
 - node-red-admin 44
 - node-red-contrib-advanced-ping 409
 - node-red-contrib-alexa-home-skill 321
 - node-red-contrib-alexa-local 321
 - node-red-contrib-bigtimer 389
 - node-red-contrib-blynk-iot 439
 - node-red-contrib-bme280 170
 - node-red-contrib-config 344
 - node-red-contrib-contextbrowser 344
 - node-red-contrib-fritz 410
 - node-red-contrib-fs 352
 - node-red-contrib-ftp 416
 - node-red-contrib-google 326
 - node-red-contrib-image-tools 402
 - node-red-contrib-influxdb 366
 - node-red-contrib-moments 82, 399
 - node-red-contrib-mqtt-broker 265
 - node-red-contrib-qr-code-generator 404
 - node-red-contrib-telegrambot 295
 - node-red-contrib-termux-api 444, 449
 - node-red-dashboard 145
 - Node-RED-Editor 61
 - Passwortschutz* 44
 - node-red-node-arduino 281
 - node-red-node-base64 256
 - node-red-node-pushbullet 316
 - node-red-node-suncalc 393
 - node-red-node-test-helper 518

- Nodes 24, 77
 Alexa 320
 Arduino 275
 Aussehen ändern 90
 bigtimer 389
 Blynk 428, 439
 Dashboard 146
 Dateien speichern 345
 Dateiformate 133
 Eigenschaften ändern 90
 E-Mail 283
 FRITZ!Box 410
 FTP 416
 Google 326
 HTTP-Verbindung 242
 InfluxDB 366
 ioBroker 497
 MQTT 265
 OpenWeatherMap 395
 Ping 406
 Prozessereignisse 135
 Pushbullet 310
 Python 387
 Raspberry Pi 168
 SQL 382
 Telegram 293
 Termux:API 449
 Timer 389
 UDP-Verbindung 242
 uibuilder 483
 ui-control 459
 Updates 77
 verbinden 102, 103
 Wetterdaten 395
 Zeitangaben 399
 Notepad++ 193
 notification-Node 175, 355
 npm 27, 32, 233
 eigene Nodes veröffentlichen 518
 init 505
 nrlinter 112
 Ntfy 423
 numeric-Node 164
- O**
-
- OAuth 44
 OpenAI 333
 openHAB 15, 311, 489
 Open-Router 337
 OpenWeatherMap 395, 481
 openweathermap-in-Node 397
 openweathermap-Nodes 396
 Operanden 202
 Operator
 = 203
 arithmetischer 202
 bedingter 204
 Bitoperator 206
 Concat 204
 Dekrement 203
 Inkrement 203
 sizeof 209
 Zuweisung 203
 Oracle VirtualBox 25
 OSI-Schichtenmodell 239
- P**
-
- package.json 503
 Pakete
 installieren 78
 npm 82
 Vulnerabilities 83
 Palette 75
 Palettenmanager 67, 73, 145
 Parser 133
 Passwort (Node-RED) 42
 Passwort-Hash 44
 payload 121
 Ping 406
 pkg 445
 play-audio-Node 398
 playcode.io 193
 POP3 291
 pop3 285
 Postman Echo 255
 PowerShell 27
 Präzedenzregel 202
 Programmierung 101, 193, 503
 Array 209
 Aussagenlogik 205
 Begriffe 196
 Codeeditoren 193
 Datum und Zeit 211
 Funktionen 216
 Kommentare 194
 Kontrollstrukturen 218
 objektorientierte Programmierung 118
 Variablen 197
 Zeichen 194
 Proxy 248

Prozessereignisse	135
Public-Key-Verfahren	46
Puffer	232
Pulsweitenmodulation	171
Push	260
Pushbullet	310
pushbullet-in-Node	317
pushbullet-Node	408
pushbullet-out-Node	316, 318
Python	387

Q

QR-Code	296, 404
qrcode-gen-Node	404, 415
Query-Anfrage	258

R

Raspberry Pi	29
<i>LEDs schalten</i>	171
<i>Sensoren</i>	169
Raspberry Pi OS	32
rbe-Node	408
Rechtsverschiebung	208
RedMobile	427
Relation	378
Remote-RED	427
reply-Node	309
Representational State Transfer	255
Request-Header	256
Resource Path	240
REST-API	423
REST-Clients	255
Retention Policys	358, 374
RGB-Farbe	176
RGB-Farbwert	234
rpi-gpio-out-Node	171
RSA-Schlüssel	47
Rückgabewert	216

S

Schlüsselwörter	196
Scope	198
Seitenleiste	340
<i>Dashboard</i>	97
<i>Debugging</i>	95
<i>Hilfe</i>	95
<i>Info</i>	93
<i>Konfiguration</i>	96

<i>Kontext</i>	97
sender-Node	309
Sequenzen	139
Sequenzielle Speichermedien	345
Sequenzregeln	131
Service	25
settings.js	40, 41, 147, 233
Shards	374
Shelly	272
Shelly 1	269
Sicherheit	43
Sicherung	69
Siri	320
Skills (Alexa)	321
Skriptsprache	193
Skype	294
slider-Node	163
SMTP	284
smtp	285
Social Media	17
some()	227
Sonnenauf-/untergang	393
Sonoff	269
Spacer	154
split-Node	141
SQL-Datenbanken	339
SQLite	376
sqlitebrowser	376
sqlite-Node	383
SSL-Zertifikate	
<i>Linux/Raspberry Pi</i>	48
<i>Windows</i>	49
Startmeldungen	38
status-Node	135
Steuerzeichen	194
Storage Engine	358
Subflow	73, 104
Subfunktion	216
Subroutine	216
suncalc-Node	393
SVG-Grafik	476
switch-Node	130, 259
Systemkonsole	128

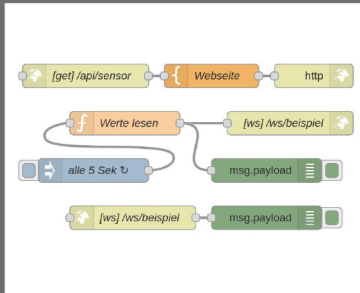
T

tabconfig-Node	157
Tabs	151
Tag-Keys	359
Tag-Set	359
Tastenkürzel	73, 87

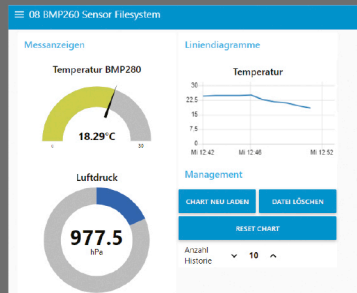
- TCP 239
 - tcp-in-Node 242, 243
 - tcp-out-Node 242, 243
 - TCP-Port 243, 246
 - tcp-request-Node 245
 - Telegram 294
 - telegram-receiver-Node 304
 - template-dashboard-Node 415
 - template-Node 158, 257, 261, 415
 - Termux 25
 - Dashboard 446
 - Node-RED 444
 - Node-RED-Editor 445
 - pkg 444
 - Sensoren 447
 - Termux:API 446
 - Text To Speech 398
 - text-input-Node 164
 - text-Node 149
 - Theme (Dashboard) 156
 - Threema 294
 - TICK 364
 - Timeout 175
 - Timer 389
 - Timestamp 123
 - TLS/SSL 45
 - TLS-Protokoll 265
 - Transmission Control Protocol 239
 - trigger-Node 150, 407
 - true 219
 - Tupel 378
 - TypeScript 461
 - Typumwandlung 208
- ## U
-
- Ubuntu 32
 - UDP 239
 - udp-in-Node 245
 - udp-out-Node 244
 - uibuilder 482
 - ui-control-Node 459, 460
 - ui-template-Node 461, 463, 476
 - Umgebungsvariable 124, 131
 - Uncatchable errors 139
 - uncaughtException errors 139
 - Unix-Timestamp 362
 - Unterprogramm 216
 - URI 241
 - URL 241
 - User Datagram Protocol 239
 - User-Interface 41
 - UTF-8-Zeichen 320
- ## V
-
- Variablen
 - Definition 198
 - Deklaration 197
 - Gültigkeitsbereiche 198
 - vCard 404
 - Verschlüsselungszertifikat 46
 - viewer-Node 402
 - Virtuelle Maschine 25
 - Visual Editor 123
 - Visual Studio Code 193
 - VPN 451
 - vsFTPd 417
 - vue.js 183
 - VueJS 487
- ## W
-
- Websocket 259
 - websocket-out-Node 262
 - Wertregeln 131
 - Wetteransage (Beispiel) 397
 - Wetterdaten 171, 395
 - WhatsApp 283, 294
 - Widgets 151, 158, 461
 - Windows 25, 269
 - Windows Subsystem for Linux 34
 - WLAN-Gastzugang 414
 - WLAN-Verbindung 62
- ## X
-
- XHTML 260
 - XML 260
 - XSLT 260
- ## Z
-
- Zeitangaben 399
 - Zeitreihen 356, 357

Einfach visuell programmieren

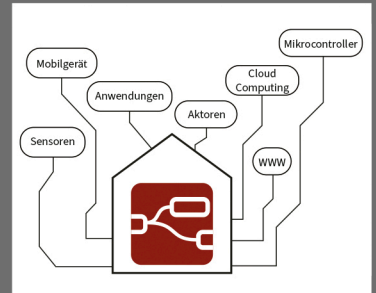
Mit dem grafischen Programmierkonzept von Node-RED bauen Sie Steuerungen einfach und ohne viel Code auf. Udo Brandes zeigt Ihnen, wie Sie Ihre Anforderungen übersichtlich modellieren und in strukturierte Abläufe überführen. So entstehen im Handumdrehen passende Dashboards und Automatisierungen für Ihre IoT- und Smart-Home-Projekte.



Flows aufbauen



Dashboards & Visualisierungen



Steuerung für Maker-Projekte

Unkompliziert loslegen

Nichts geht über KISS: Keep it Simple, Stupid. Und was könnte einfacher sein, als Programmierlogik direkt im Browser visuell zu erstellen? Vermeiden Sie Code-Chaos und bauen Sie schnell erste Tests und Prototypen.

Für Maker-Projekte und das Smart Home

Node-RED bietet Werkzeuge für die Heimautomation, für IoT-Prototyping oder Maker-Projekte. Speichern Sie Messwerte performant in InfluxDB ab, integrieren Sie Ihre FRITZ!Box und verarbeiten Sie Sensordaten mit dem Raspberry Pi.

Node-RED für Ihre Projekte

Dieser umfassende Einstieg begleitet Sie auch bei anspruchsvollen Szenarien: Programmieren Sie eigene Nodes, steuern Sie Ihr Setup mobil per App, gestalten Sie professionelle Dashboards, greifen Sie auf die ChatGPT-API zu oder programmieren Sie Mikrocontroller mit Node-RED.



Beispiel-Flows und alle Projektdateien zum Download



Udo Brandes beschäftigt sich mit der Heimautomation, Mikrocontrollern und dem großen Maker-Bereich. In diesem Handbuch zeigt er Ihnen, wie Sie mit Node-RED eigene Projekte umsetzen.

Aus dem Inhalt

- Installation und Grundlagen
- Administration und sichere Anbindung
- Der Editor von Node-RED
- Die Basics: Nodes und Flows
- Daten auf Dashboards anzeigen
- Eigene Dashboards erstellen
- Funktionen programmieren
- Grundlagenwissen zu JavaScript, Node.js und gutem Programmieren
- Daten abrufen und speichern
- Datenaustausch über MQTT und TCP/IP
- Hacks: fortgeschrittene Nodes nutzen
- App-Steuerung und Zugriff von unterwegs
- Eigene Nodes entwickeln
- Mikrocontroller steuern

