

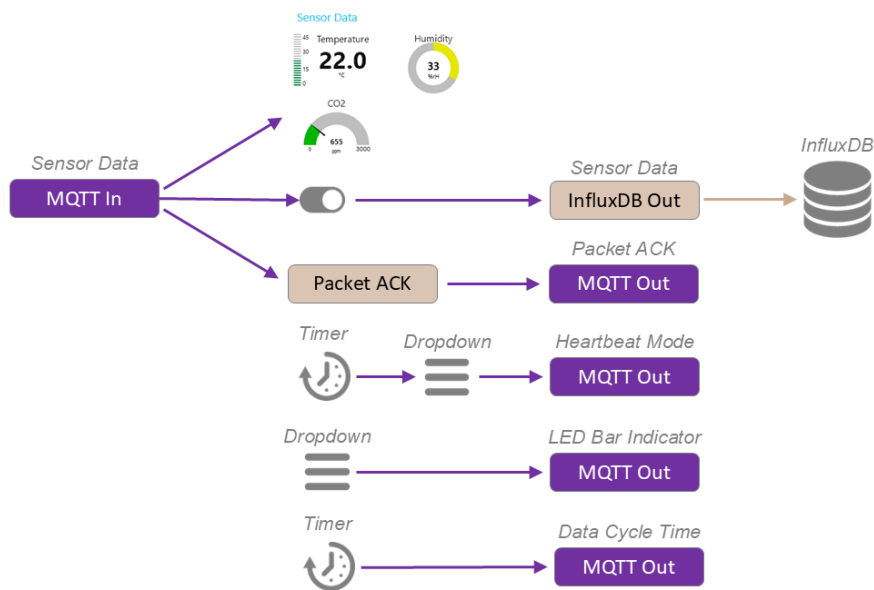
Projekt: [https://github.com/ronaldsieber/ESP32SmartBoard\\_NodeRED](https://github.com/ronaldsieber/ESP32SmartBoard_NodeRED)  
Lizenz: MIT  
Autor: Ronald Sieber

# ESP32SmartBoard\_NodeRED

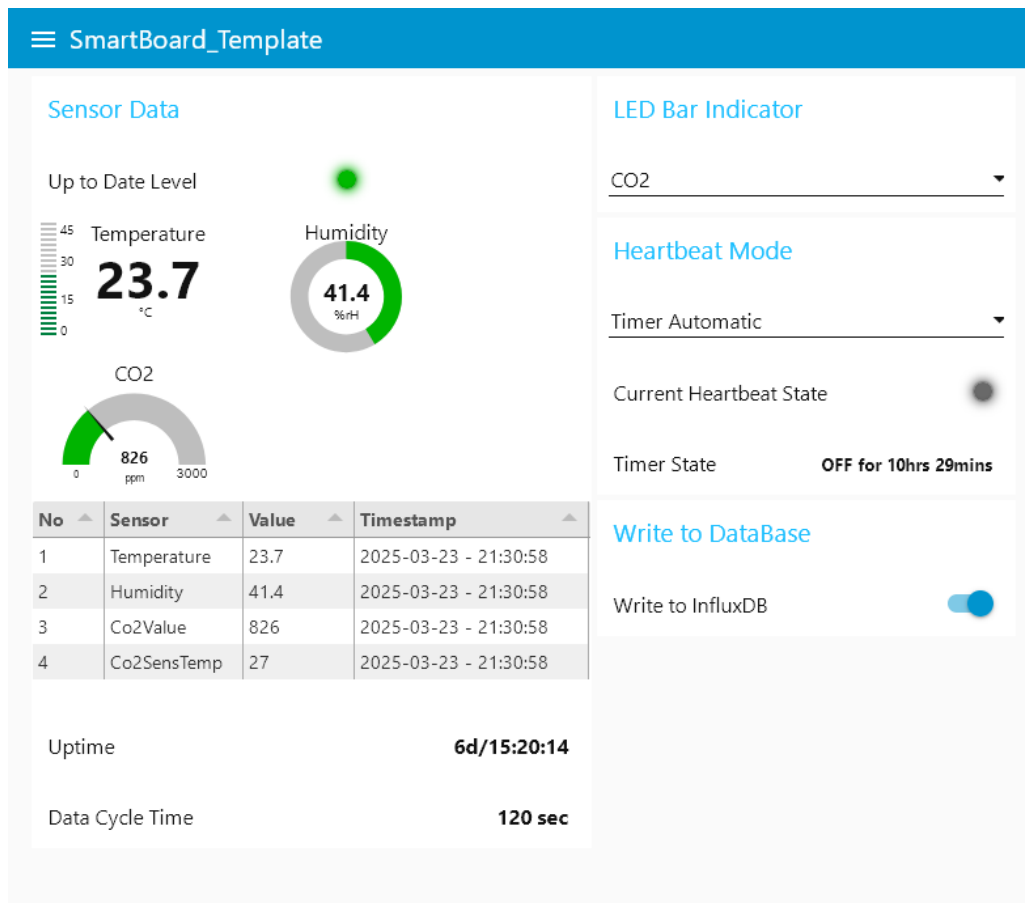
Dieser Node-RED Flow vervollständigt das Arduino Projekt <ESP32SmartBoard\_MqttSensors\_BleCfg> um ein Dashboard und ermöglicht zudem das Speichern der Sensordaten in einer Datenbank. Das Dashboard zeigt die aktuellen Sensordaten an und stellt dem Anwender eine GUI zum Konfigurieren des Laufzeitverhaltens des ESP32SmartBoard bereit.

## Projekt-Überblick

Der Node-RED Flow kommuniziert mit dem *ESP32SmartBoard* über MQTT Messages. Er empfängt Sensordaten vom Board, zeigt diese auf dem Dashboard grafisch an und schreibt diese parallel dazu in eine InfluxDB Datenbank. Außerdem beinhaltet das Dashboard Steuerelemente, die es dem Anwender ermöglichen, das Verhalten des Boards zu konfigurieren. Dazu sendet der Flow entsprechende Konfigurations-Nachrichten via MQTT an das Board. So lässt sich beispielsweise über ein Dropdown Menü wählen, ob der LED Bar Indikator den Messwert des Temperatur-, Luftfeuchte- oder des CO<sub>2</sub>-Sensors anzeigt. Der Heartbeat-Modus (Blinken der blauen LED auf dem ESP32DevKit) wird primär über einen konfigurierbaren Timer gesteuert. Um das Board auch in Schlafräumen nutzen zu können, wird Heartbeat darüber nachts deaktiviert und nur tagsüber eingeschaltet. Alternativ kann Heartbeat auch manuell über ein Dropdown Menü dauerhaft ein- bzw. ausgeschaltet werden.



Node-RED selbst ist eine grafische, Event-basierte Low-code Programmierumgebung. Sowohl die Erstellung des Programmes ("Flow" genannt) als auch die Anzeige des Dashboards erfolgen ausschließlich im Browser. Damit eignen sich für die Programmierung alle gängigen Endgeräte wie PC und Laptop, für die Anzeige des Dashboards darüber hinaus auch Tablets oder Smartphones.

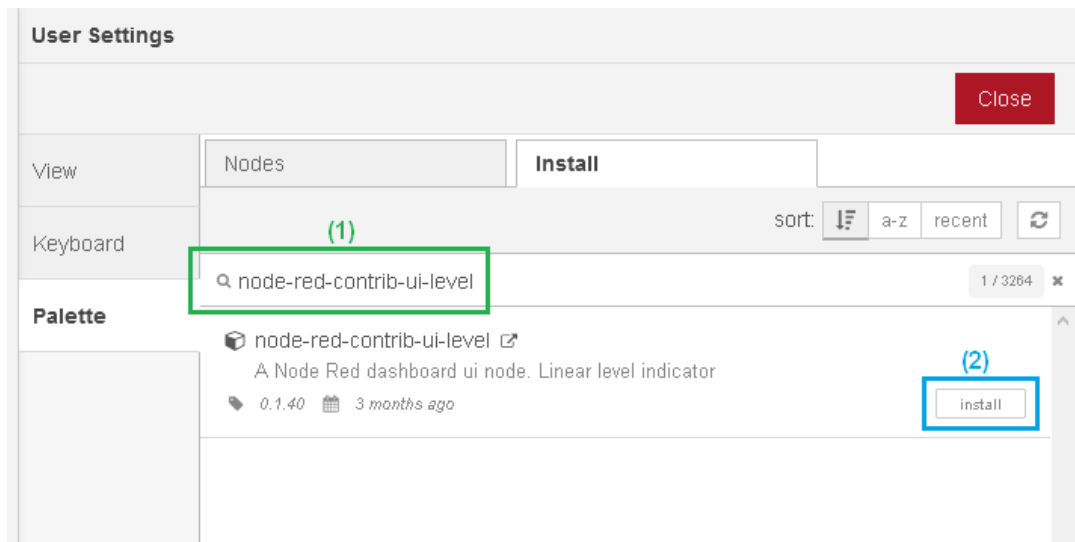


## Setup des Flows in Node-RED

### Installation von Drittanbieter-Nodes

Der Node-RED Flow für das *ESP32SmartBoard* nutzt verschiedene Nodes von Drittanbietern, die zunächst manuell zur Node-RED Palette hinzugefügt werden müssen. Dazu sind folgende Schritte notwendig:

- Node-RED Main Menü ("Hamburger Menü") -> Manage Palette
- Im Paletten Manager zum Tabsheet "*Install*" wechseln und mit Hilfe des Suchfeldes die unten aufgeführten Drittanbieter-Nodes suchen und installieren:



### Liste der zu installierenden Drittanbieter-Nodes:

- `node-red-dashboard`
- `node-red-contrib-influxdb`
- `node-red-contrib-bigtimer`
- `node-red-contrib-countdown`
- `node-red-contrib-simple-gate`
- `node-red-contrib-ui-led-fork`
- `node-red-contrib-ui-level`
- `node-red-node-ui-table`

### Import des Flows in Node-RED


Um den <Flow> in Node-RED einzufügen, sind folgende Schritte notwendig:

- Node-RED Main Menü ("Hamburger Menü") -> Import
- Für "import to" als Ziel "new flow" auswählen
- Via "select a file to import" das File <ESP32SmartBoard\_NodeRED.json> importieren

## Import nodes

**Clipboard**

Paste flow json or 

 select a file to import (2)

Library

Examples

Import to 

current flow

new flow

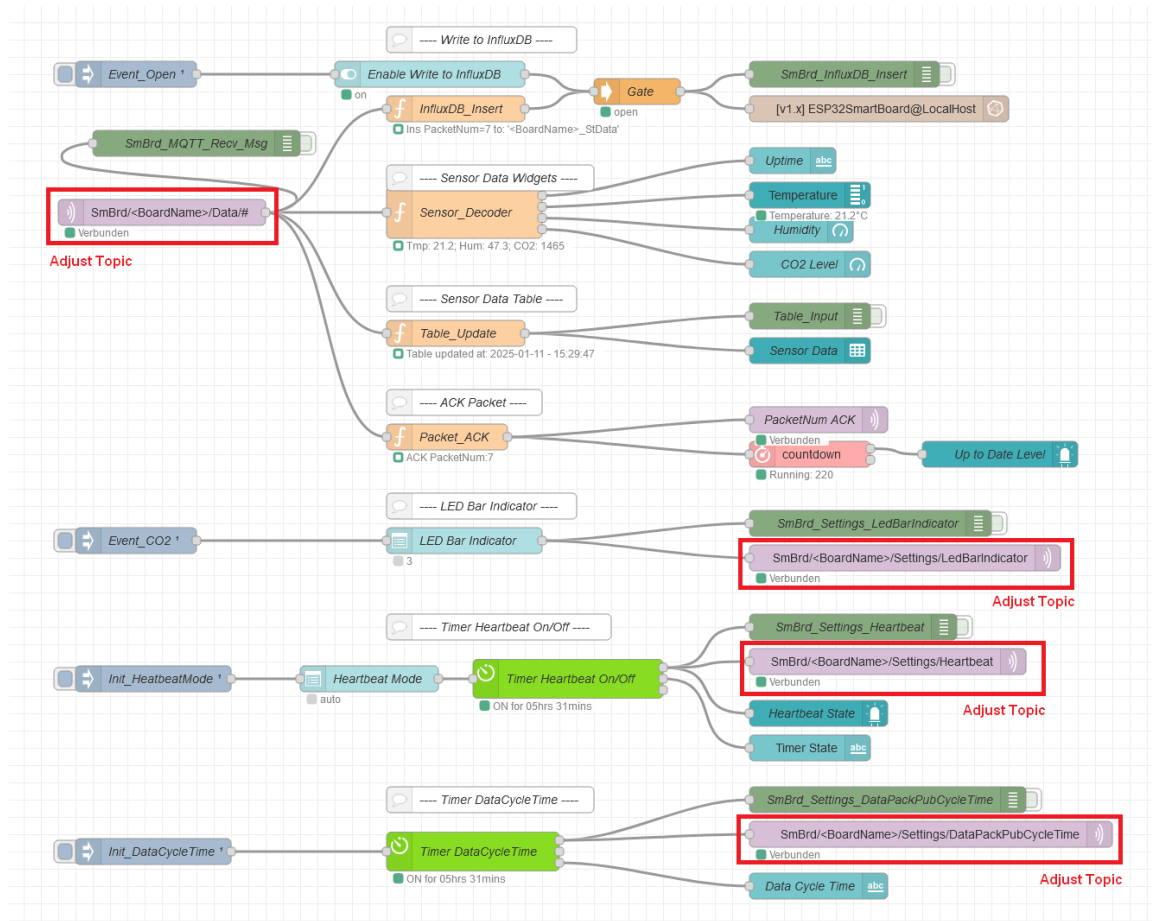
 (1)

Cancel

Import

## Verknüpfung des Node-RED Flows mit dem ESP32SmartBoard

Das *ESP32SmartBoard* und Node-RED Flow kommunizieren untereinander mit einem Set von MQTT Messages. Dazu nutzt der Flow insgesamt fünf MQTT-Nodes, davon einen für den Empfang von Sensordaten (1x MQTT In) und insgesamt vier weitere Nodes zum Versenden von Konfigurations-Messages (4x MQTT Out):



Um untereinander Daten austauschen zu können, müssen sowohl *ESP32SmartBoard* als auch Node-RED Flow das gleiche Set von MQTT Messages verwenden. Daher ist es zum Verlinken von *ESP32SmartBoard* und Node-RED Flow notwendig, für vier der MQTT-Nodes (siehe Markierung im Bild oben) die vom Board genutzten Topics zu übernehmen. Dies erfolgt im Property Dialog der Nodes, indem dort das Feld "Topic" entsprechend angepasst wird:

The image shows a 'Edit mqtt in node' window. At the top are buttons for 'Delete', 'Cancel', and 'Done'. Below is a 'Properties' section with several fields: 'Server' with the value '127.0.0.1:1883', 'Topic' with the value 'SmBrd/240AC4617D20/#' (highlighted with a red box), 'QoS' with the value '0', 'Output' with the value 'auto-detect (string or buffer)', and 'Name' with the value 'Name'.

Details zum Aufbau der Topics beschreibt der Abschnitt "*Individualisierung der MQTT-Topics zur Laufzeit*" im Arduino Projekt [<ESP32SmartBoard\\_MqttSensors\\_BleCfg>](#). In der Standardkonfiguration zeigt das *ESP32SmartBoard* während des Boot-Vorgangs eine Liste aller verwendeten Topics im seriellen Terminal-Fenster an.

## Implementierungs-Details des Node-RED Flows

*ESP32SmartBoard* und Node-RED Flow tauschen ihre Daten über MQTT Messages aus. So werden die Sensordaten des Boards über den *MQTT Input Node* in den Flow eingelesen. Der Function Node [<Sensor\\_Decoder>](#) arbeitet als Multiplexer und gibt die empfangenen Sensordaten in Abhängigkeit vom Topic (= Sensortyp) an jeweils einem seiner 3 Ausgänge aus. Dadurch erhält jedes der 3 nachgelagerten Instrumenten-Widgets (Temperatur, Luftfeuchtigkeit, CO2-Level) nur genau die für das jeweilige Element relevanten Daten.

Zusätzlich zu den Instrumenten-Widgets werden die Messwerte kompakt in einer Tabelle angezeigt. Im Function Node [<Table\\_Update>](#) werden die über den *MQTT Input Node* empfangenen Sensordaten formatiert und um einen Timestamp erweitert an das Tabellen-Widget übergeben.

Parallel zur Anzeige in den Dashboard-Widgets werden die Sensordaten in eine InfluxDB Datenbank geschrieben. Das erlaubt es, die Messwerte später als Zeitreihe auszuwerten (z.B. in Grafana). Für das Schreiben der Daten in die Datenbank ist der *Influxdb Out* Node zuständig. Der vorgelagerte Function Node [<InfluxDB\\_Insert>](#) extrahiert den Sensornamen aus dem empfangenen MQTT Topic und übergibt diesen als Attribut *msg.measurement* im Return-Objekt. Der *Influxdb Out* Node übernimmt ihn anschließend von dort als Measurement Name für den Eintrag der Daten in die InfluxDB.

Der Function Node [<Packet\\_ACK>](#) extrahiert die Paketnummer des empfangenen Sensordaten-Paketes, der nachfolgenden *MQTT Output Node* sendet diese als Quittung zurück an das *ESP32SmartBoard*.

Mit der Flusskette aus einem *BigTimer* Node und nachfolgendem *MQTT Output Node* wird die Zykluszeit der Datenübertragung vom *ESP32SmartBoard* gesteuert. So wird tagsüber eine Zykluszeit von typisch 1 Minute benutzt, nachts wird das Sende-Intervall dagegen auf 2 Minuten vergrößert, so dass weniger Daten in die Datenbank geschrieben werden.

Die beiden Dropdown Menüs erlauben es dem Anwender, das Verhalten des Boards zu konfigurieren. Dazu sendet der Flow entsprechende MQTT Nachrichten an das Board. Die den Dropdown Nodes vorgelagerten *Inject* Nodes selektieren einmalig beim Deploy die jeweiligen Default-Werte der Dropdown Menüs. Der Heartbeat-Modus (Blinken der blauen LED auf dem ESP32DevKit) wird im

Zustand *"Timer Automatic"* durch den *BigTimer* Node gesteuert. Über den Property Dialog des Nodes lassen sich die Zeitpunkte zum Ein- und Ausschalten des Heartbeat festlegen:

The screenshot shows the 'Edit bigtimer node' dialog box. The 'Properties' tab is selected. The 'Name' field contains 'Big Timer'. The 'Comment' field is empty. The 'On Time' field is set to '08:00' and the 'Off Time' field is set to '21:00'. These two fields are highlighted with a red rectangle. Below these, there are two rows of 'On Time2' and 'Off Time2' fields, both set to '00:00'. At the bottom, there are two rows of 'On Offset' and 'Off Offset' fields, both set to '0'. The dialog has 'Delete', 'Cancel', and 'Done' buttons at the top.

Durch die Zeitsteuerung kann das Board auch in Schlafräumen genutzt werden, wobei nachts Heartbeat automatisch deaktiviert wird, um so das ggf. störende Blinken zu unterbinden. Der Timer lässt sich durch das vorgelagerte Dropdown Menü deaktivieren und stattdessen der Heartbeat-Modus manuell festlegen. Dies erfolgt durch schreiben der Steuerbefehle *"auto"*, *"manual on/off"* bzw. *"quiet"* an den Timer. An seinem ersten Ausgang emittiert der *BigTimer* bei einer Status-Änderung jeweils das erforderliche Kommando, um das Heartbeat-Verhalten des *ESP32SmartBoards* zu setzen (ein/aus). Der nachfolgende MQTT Output Node generiert dann die entsprechende MQTT Nachricht für das Board. An seinem zweiten Ausgang gibt der *BigTimer* einmal je Minute Informationen zum aktuellen Timer-Status aus. Das Attribut *msg.extState* des Return-Objektes liefert die Status-Information als Text, der im nachfolgenden Text-Widget im Dashboard angezeigt und minütlich aktualisiert wird (z.B. *"On for 12hrs 34mins"*).

Detaillierte Informationen zum *BigTimer* Node sind auf der Webseite <https://tech.scargill.net/big-timer/> zu finden.