



Leitfaden Technisches Monitoring

Im Projekt MonSEC - Monitoring Secure



Stand: 22.03.2023

Gefördert durch:



aufgrund eines Beschlusses
des Deutschen Bundestages

Inhaltsverzeichnis

1 Einleitung	7
2 Arten des Monitorings	9
2.1 Energiemonitoring (EMon)	9
2.2 Anlagenmonitoring (AMon)	9
2.3 Gebäude- und Behaglichkeitsmonitoring (GBMon)	10
2.4 Einregulierungsmonitoring (ERMon)	11
2.5 Langzeitmonitoring (LZMon)	11
3 Sensorik	12
3.1 Übersicht	12
3.2 Temperatur	12
3.2.1 Vergleich und Bewertung der Sensortypen	16
3.2.2 Praktische Anwendung und Auswahlkriterien	18
3.3 Luftfeuchte	18
3.4 CO ₂	19
3.5 Volumenstrom	19
3.5.1 Volumenstrommessung bei Flüssigkeiten	20
3.5.2 Volumenstrommessung bei Gasen	20
3.6 Beleuchtungsstärke	21
3.7 Solarstrahlung	21
3.8 Messgeräte	22
3.8.1 Wärme- und Kälteenergie	22
3.8.2 Stromzähler	23
3.8.3 Kombinierte Sensoren	24
3.8.4 Wetterstation	24
3.9 Auswahl von Sensoren und Messgeräten	24
3.10 Fehler beim Einbau von Sensoren und Messgeräten	25
3.11 Fehler beim Betrieb von Sensoren	27
4 Erfassung – Messtechnik Topologie	37
4.1 Messkette	38
4.2 Übertragungsstrecken	38
4.3 BUS-Systeme	40
4.3.1 Physikalische Busebene	40
4.3.2 Protokollebene	43

4.4	Funkübertragungsstrecken	45
4.4.1	Wireless LAN (WLAN)	45
4.4.2	LoRaWAN	46
4.4.3	XBee	46
4.4.4	Mobilfunk (5G)	46
4.5	Speicherprogrammierbare Steuerung	46
5	Datenspeicherung	48
5.1	CSV-Datei	48
5.2	Datenbanken	49
5.3	Datenpunktbezeichnungen	50
5.4	Metadatensystem	52
6	Redaktion	54

Abbildungsverzeichnis

1.1	Schematische Darstellung eines Monitorings [TH Rosenheim]	7
2.1	Gliederung des Technischen Monitoring [2]	9
2.2	Gliederung des Energiemonitoring [2]	10
2.3	Gliederung des Anlagenmonitoring [2]	10
2.4	Gliederung des Gebäude- und Behaglichkeitsmonitoring [2]	11
3.1	Genauigkeitsklassen für Pt Widerstandsthermometer nach DIN EN 60751	13
3.2	Anschlussarten von Pt Widerstandsthermometern nach DIN EN 60751 . .	14
3.3	Prinzip der Temperaturmessung mit Thermopaaren	14
3.4	Grenzabweichungen nach DIN EN 60584	15
3.5	Digitaler Temperatursensor DS18B20-PAR (Quelle: Maximintegrated) . .	16
3.6	Aufbau kapazitiver Feuchtefühler	28
3.7	BME280 (Quelle: Adafruit)	29
3.8	Genauigkeit von Feuchtesensoren (schematisch)	30
3.9	Prinzip der NDIR Messung (Quelle: Laser Components)	30
3.10	Prinzip des Flügelradzählers (Quelle: Wikipedia)	31
3.11	Prinzip des Ultraschall-Laufzeitverfahrens	31
3.12	Flügelradsensor [Ahlborn]	32
3.13	Thermisches Anemometer [Ahlborn]	32
3.14	Luxmeter [Ahlborn]	33
3.15	Pyranometer [Ahlborn]	33
3.16	Prinzip und Einbau von WMZ	34
3.17	Wandler- sowie Direktstromzähler (Quelle: ABB)	34
3.18	Aufbau einer Wetterstation (Quelle: Firma Lufft)	35
3.19	Strahlungsschutzgehäuse für Temperatursensoren im Außenbereich . . .	36
3.20	Anordnung der Messpunkte im Heizkreis (Quelle: AMEV)	36
4.1	Topologie eines Monitoringsystems	37
4.2	Messkette innerhalb des Sensors	38
4.3	Schematische Darstellung von Übertragungsstrecken	39
4.4	Darstellung der Signalübertragung in Bussystemen	41
4.5	Master/Slave Schema	42
4.6	Schematische Darstellung eines Pegelwandlers	43
4.7	Beispielhafte Abbildung einer SPS [Beckhoff]	47
5.1	Struktur des BUDO-Schemas [?]	51

5.2 Anwendungsbeispiel der mondias Metadatenstruktur	53
--	----

Tabellenverzeichnis

3.1	Typische Messgrößen und Sensoren	12
3.2	Wärmeleistung und relevante Größen	23
3.3	Empfehlungen für die Sensorauswahl	25
5.1	Datenpunkte und ihre Werte	49

1 Einleitung

Technisches Monitoring (TMon) ist das Erfassen, Speichern und Auswerten von Zustands- und Prozessgrößen von Gebäuden, Anlagen und Räumen. Eines der wichtigsten Merkmale eines TMon ist, dass die Messwerte in einer Datenbank gespeichert werden und die Auswertung der Messgrößen unter Berücksichtigung des zeitlichen Verlaufs erfolgt. Dadurch werden zusätzliche Erkenntnisse gewonnen. Das Ziel eines TMon ist in der Regel eine Optimierung der Gebäude bzw. Anlagen.

Technisches Monitoring

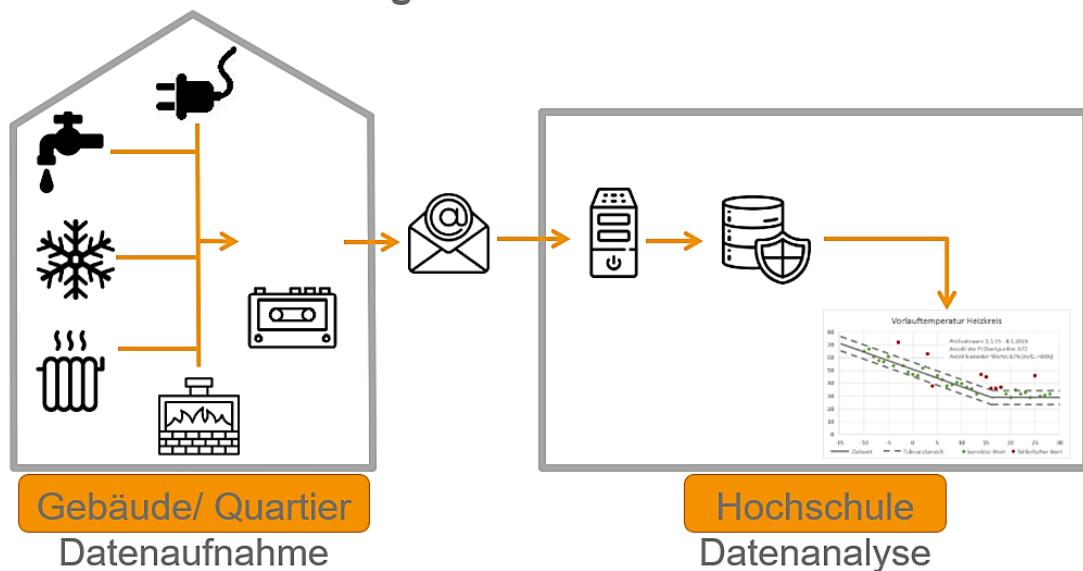


Abbildung 1.1: Schematische Darstellung eines Monitorings [TH Rosenheim]

Ein TMon wird aufgrund der steigenden Technisierungsgrade sowie der wachsenden Anzahl an gebäudetechnischen Anlagen zunehmend wichtiger, um einen fehlerfreien und effizienten Betrieb der Anlagen bzw. der Gebäude sicherzustellen. Der Aufwand für ein TMon lässt sich aufgrund des vielfältigen Nutzens der Daten rechtfertigen.

Wie in Abbildung 1.1 schematisch dargestellt, ist die Aufgabe eines Monitorings die Erhebung, Speicherung und Analyse von Messdaten. Anhand dieser Einteilung lässt sich das Monitoring auch in die folgenden Teilespekte einteilen.

- **Sensorik und Erfassung**

Dieser Bereich beschäftigt sich mit der Messtechnik, die für ein Monitoring benötigt wird. Durch Sensoren werden die physikalischen Größen gemessen und durch

elektrische Signale an einen Computer weitergeleitet, der diese dann in einer Datenbank speichert. In dem Hauptkapitel wird sowohl auf die verschiedenen Sensoren als auch auf verschiedene Wege für die Signalweiterleitung (die sogenannte Messkette) eingegangen. Die Messkette beinhaltet neben der physikalischen Weiterleitung auch verschiedene Übertragungsprotokolle.

- **Datenspeicherung**

Dieser Punkt beschäftigt sich mit der Speicherung der Daten. Dabei wird ausgedehnt von allgemein bekannten Datenformaten (Excel und CSV) hauptsächlich auf Datenbankformate eingegangen.

- **Visualisierung und Auswertung**

Sind die Daten in der Datenbank (oder in anderen Datenformaten) gespeichert, ist der anschließende Schritt die Auswertung der Daten. Dabei nimmt auch die Visualisierung, also das Sichtbarmachen der Daten in Form von Graphen und Diagrammen, einen hohen Stellenwert ein.

2 Arten des Monitorings

Technisches Monitoring (TMon) lässt sich grundsätzlich in drei Arten sowie zwei Betriebsweisen einteilen. Diese sind in der VDI 6041 „Technisches Monitoring von Gebäuden und gebäudetechnischen Anlagen“ definiert.

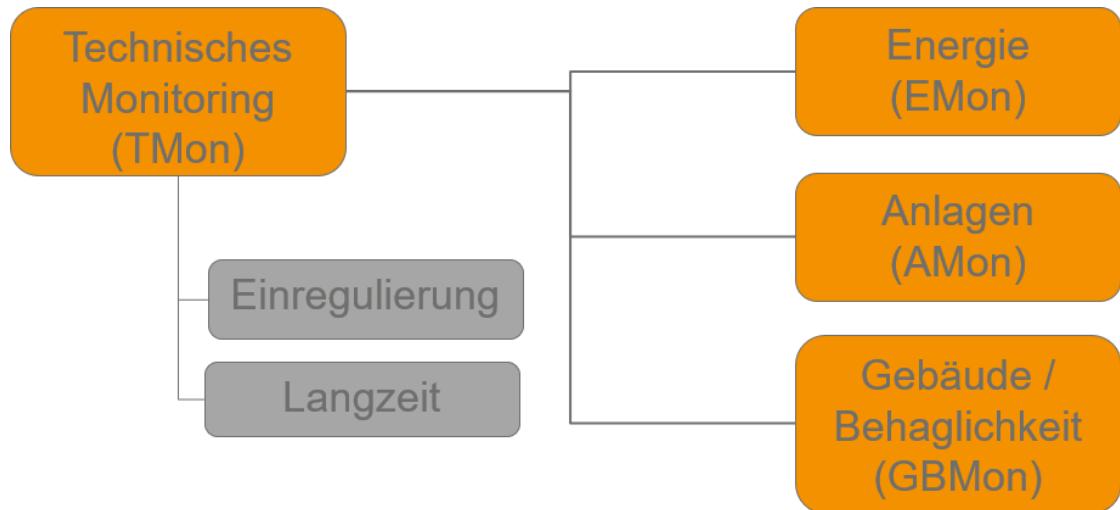


Abbildung 2.1: Gliederung des Technischen Monitoring [2]

Die drei Bereiche haben verschiedene Anforderungen an das Monitoringequipment und erfassen unterschiedliche Messgrößen. Dabei können sich die Bereiche auch überlappen.

2.1 Energiemonitoring (EMon)

EMon (siehe Abbildung 2.2) betrachtet die Energie- und Medienverbräuche sowie die Leistung und Volumenströme. Das EMon liefert ein Bild über den Energieverbrauch der Gebäude und bietet Möglichkeiten der Optimierung der Energieeffizienz. Ein Einsatz für Abrechnungszwecke, zum Beispiel im Mehrfamilienhaus, ist ebenfalls möglich. Hier werden allerdings geeichte Zähler benötigt.

2.2 Anlagenmonitoring (AMon)

AMon (siehe Abbildung 2.3) betrachtet hauptsächlich die Betriebszustände von technischen Anlagen. Die Hauptaufgaben dieser Art des Monitorings sind die Überwachung der

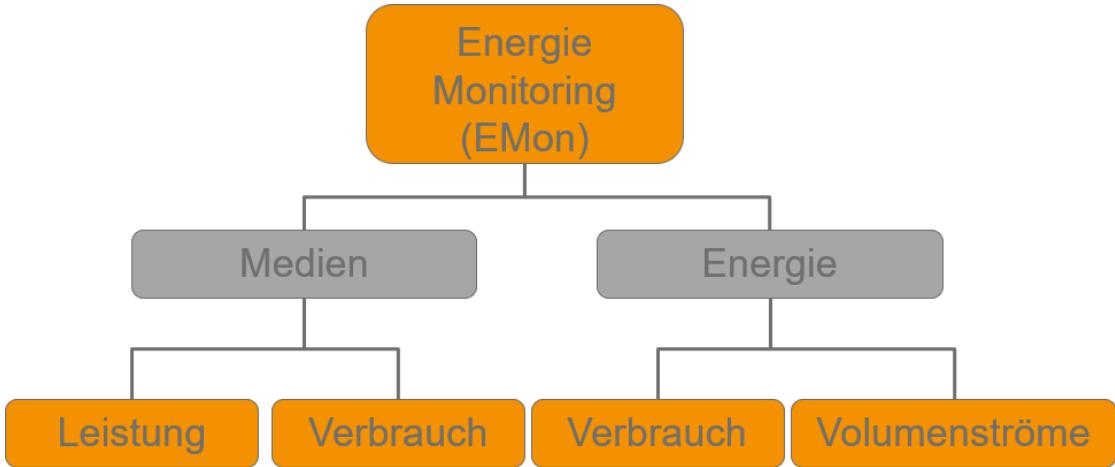


Abbildung 2.2: Gliederung des Energiemonitoring [2]

korrekten Funktion und die Betriebsoptimierung. Darüber hinaus kann damit auch eine bedarfsorientierte Wartung gesteuert werden.



Abbildung 2.3: Gliederung des Anlagenmonitoring [2]

2.3 Gebäude- und Behaglichkeitsmonitoring (GBMon)

GBMon (siehe Abbildung 2.4) erfasst die Raumkonditionierung und liefert daher Aussagen über das Nutzerverhalten sowie Funktion und Regelungsqualität der Raumkonditionierungsanlage, aber auch der Zustand der Gebäudehülle spielt eine Rolle. Neben der

messtechnischen Erfassung kann auch die Befragung der Nutzer eine Aussage über das Verhalten sowie Empfinden liefern. Die erfassten Daten liefern Aussagen über die Betriebsdaten unter Einfluss der Regelung, des Gebäudes und des Nutzerverhaltens. Dabei kann man vor allem eine Abweichung zum Sollzustand erkennen.

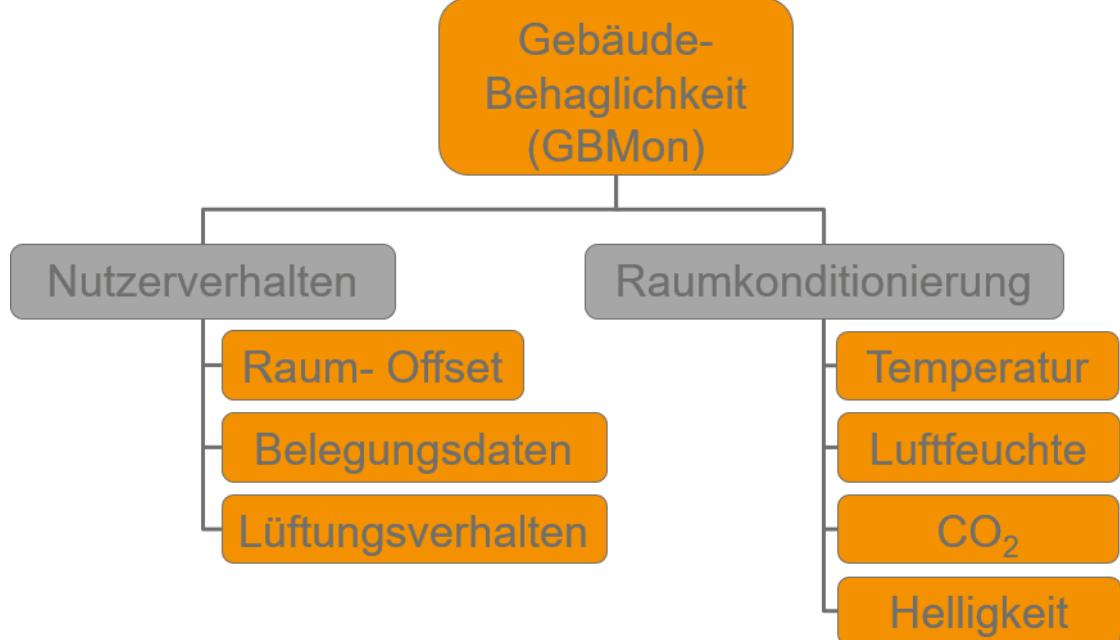


Abbildung 2.4: Gliederung des Gebäude- und Behaglichkeitsmonitoring [2]

2.4 Einregulierungsmonitoring (ERMon)

Das ERMon beginnt mit dem Start des Regelbetriebs des Gebäudes und hat das vorrangige Ziel, die gebäudetechnischen Anlagen korrekt einzustellen und diese zu optimieren. Diese Betriebsweise erstreckt sich in der Regel über die ersten beiden Jahre der Nutzung. Hierbei werden detaillierte Analysen der Energieverbräuche und Betriebszustände ermittelt und ausgewertet. Während eine Standardinbetriebnahme üblicherweise nur die Funktion der Anlagen überprüft, geht es beim ERMon vor allem um die Regelfunktionen der Anlagen und das Zusammenspiel der verschiedenen Anlagen.

2.5 Langzeitmonitoring (LZMon)

Das LZMon beginnt im Anschluss an das ERMon und dient der Erhaltung des optimalen Betriebs und dem frühzeitigen Erkennen von Fehlerzuständen über die gesamte Laufzeit des Gebäudes. Für ein LZMon können eventuell die Zahl der Datenpunkte und die Messfrequenz reduziert werden.

3 Sensorik

AM Anfang der Messkette stehen die Sensoren. Im Folgenden werden typische Sensoren zur Erfassung von Messgrößen beschrieben und es wird hingewiesen, was bei einem Monitoring zu beachten ist.

3.1 Übersicht

Sensoren erfassen einzelne (oder bei kombinierten Sensoren mehrere) physikalische Größen und stellen diese meistens als elektronisches Signal zur Verfügung. Dabei kann man zwischen analogen und digitalen Sensoren unterscheiden. Analoge Sensoren stellen die gemessene Größe meistens durch eine Spannungsdifferenz am Sensorausgang dar. Für die weitere Verwendung werden diese durch die entsprechende Messtechnik (Analog-Digital-Wandler - ADW) in digitale Signale zur weiteren Verarbeitung umgewandelt.

Sensoren, die heutzutage erhältlich sind, werden häufig bereits mit einer digitalen Schnittstelle zur Datenabfrage versehen. Dies bedeutet, dass der Sensor bereits einen ADW enthält, womit der Ausgang des Sensors als digitale Schnittstelle zur Verfügung steht. Je nach Art (und auch je nach Preis) des Sensors stehen dabei unterschiedliche Schnittstellen zur Verfügung.

In einem TMon (auf Gebäude bezogen) werden typischerweise folgende Messgrößen aufgenommen. Diese sind in Tabelle 3.1 aufgelistet.

Name	Formelzeichen	Einheit	Typische Sensoren
Temperatur		°C	Widerstand, Thermocouple
Luftfeuchte rel.		%	
Druck		Pa	Membran, Piezo
CO ₂ Gehalt		ppm	NDIR
Volumenstrom		m ³ /h	Flügelrad, Ultraschall
El. Spannung		V	
El. Strom		A	

Tabelle 3.1: Typische Messgrößen und Sensoren

3.2 Temperatur

Im TMon spielt die Temperurmessung eine sehr große Rolle, da die Temperaturdifferenz in Wärmestromberechnungen einfließt. Aber auch in der Komfortbeurteilung spielt sie

Formelzeichen
raus wenn
wir nicht
ausfüllen

eine große Rolle. Wichtig zu wissen ist, dass bei einer Messung einer Temperaturdifferenz, die Messgenauigkeit doppelt eingeht. Dies spielt eine außerordentliche Rolle bei kleinen Temperaturdifferenzen. Ein Beispiel soll dies verdeutlichen: Wenn bei einem Heizkreis der Unterschied zwischen Vor- und Rücklauf 20 K beträgt und die Temperaturfühler eine Messgenauigkeit von 0,1 K haben, dann beträgt die maximale Messunsicherheit 0,2 K, macht also bei 20 K eine Abweichung bis zu 1%. Bei einer Temperaturdifferenz von 2 K beträgt die Messgenauigkeit für die Temperaturdifferenz nun schon 10%. Dies gilt damit natürlich auch für die Ungenauigkeit des Wärmestroms.

Um Temperaturen mit nur 0,1 K Ungenauigkeit messen zu können, müssen neben der reinen Messkette Senor bis zum digatalen Messwert viele praktische Details beachtet werden. Der Messwert ist immer die Temperatur des Sensors selbst, aber wir wollen ja

Temperaturen können durch viele Arten des Erfassens von physikalischen Eigenschaften ermittelt werden. Im TMon werden zu diesem Zweck gemeinhin widerstandsisierte Sensoren und Thermoelemente herangezogen. Bei der widerstandsisierten Temperaturmessung bieten sich, aufgrund des weitgehend linearen Verhältnisses von Temperatur und Widerstand, vor allem Sensoren aus Metallen an. Hier sind vor allem Platinwiderstände wie Pt100 und Pt1000. Auch bieten diese den Vorteil in der DIN EN 60751 normiert zu sein, wo beispielsweise Gültigkeitsbereiche und Grenzabweichungen festgelegt sind (siehe Abbildung 3.1).

Tabelle 3 – Genauigkeitsklassen für Thermometer			
Klasse	Gültigkeitsbereich °C		Grenzabweichung ^a °C
	Drahtgewickelte Widerstände	Schichtwiderstände	
AA	-50 bis +250	0 bis +150	$\pm(0,1 + 0,0017 t)$
A	-100 bis +450	-30 bis +300	$\pm(0,15 + 0,002 t)$
B	-196 bis +600	-50 bis +500	$\pm(0,3 + 0,005 t)$
C	-196 bis +600	-50 bis +600	$\pm(0,6 + 0,01 t)$

^a $|t|$ = Betrag der Temperatur in °C ohne Berücksichtigung des Vorzeichens.

Abbildung 3.1: Genauigkeitsklassen für Pt Widerstandsthermometer nach DIN EN 60751

Neben den Platinsensoren gibt es noch Nickelsensoren. Diese haben zwar den Vorteil eines größeren Änderungskoeffizienten, sind aber hinsichtlich Langzeitstabilität, Linearität und Genauigkeit den Pt-Sensoren unterlegen. Auch der frühere deutliche Preisunterschied ist kaum noch vorhanden und spielt nur noch bei Massenware eine Rolle.

Bei der Messung mit widerstandsisierten Systemen, welche mit Zweileiteranschluss ausgeführt sind, muss beachtet werden, dass der Widerstand der Sensorleitung die Messung verfälschen kann. Dies muss vor allem bei langen Sensorleitungen berücksichtigt werden. Folglich muss beim Einsatz von Zweileitersystemen der Leitungswiderstand ermittelt und vom Messwert subtrahiert werden. Hinzu kann im Lauf der Zeit noch eine Zunah-

me des Leitungswiderstands durch Korrosion vor allem an Steckverbindungen entstehen. Je höher der Widerstandswert des Sensors, um so weniger verfälscht der Leitungswiderstand das Messergebnis. Eine technische Lösung dieses Umstandes bieten Anschlüsse mit mehreren Leitern (siehe Abbildung 3.2).

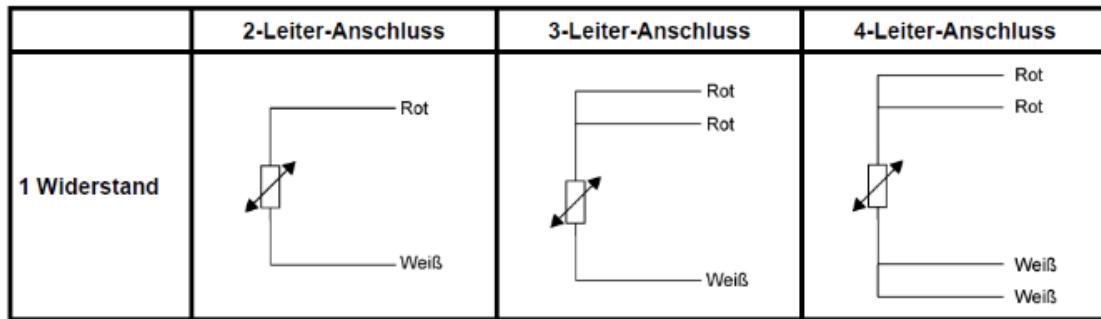


Abbildung 3.2: Anschlussarten von Pt Widerstandsthermometern nach DIN EN 60751

Bei Dreileiter- und Vierleitersystemen bilden sich weitere Messkreise, in denen der Leitungswiderstand ermittelt und als Referenz genommen werden kann. Sensoren der Klassen AA und A müssen mindestens einen Dreileiteranschluss aufweisen. Die größte Genauigkeit liefert das Vierleitersystem. Bei dieser Verschaltung, auch H-Brücke genannt, werden in einem (Strom-) Kreis die Spannung und in dem anderen Kreis die Stromstärke bestimmt. Hierdurch lässt sich der Einfluss des Kabelwiderstandes weitestgehend eliminieren.

Thermoelemente bestehen aus zwei unterschiedlichen Metallen, welche an der Messstelle verbunden sind (siehe Abbildung 3.3). Durch die temperaturabhängige Potentialdifferenz beider Metalle liegt an der Messstelle bei Temperaturänderungen eine Spannung an. Das Verhältnis von Temperatur und Potentialdifferenz ist nichtlinear.

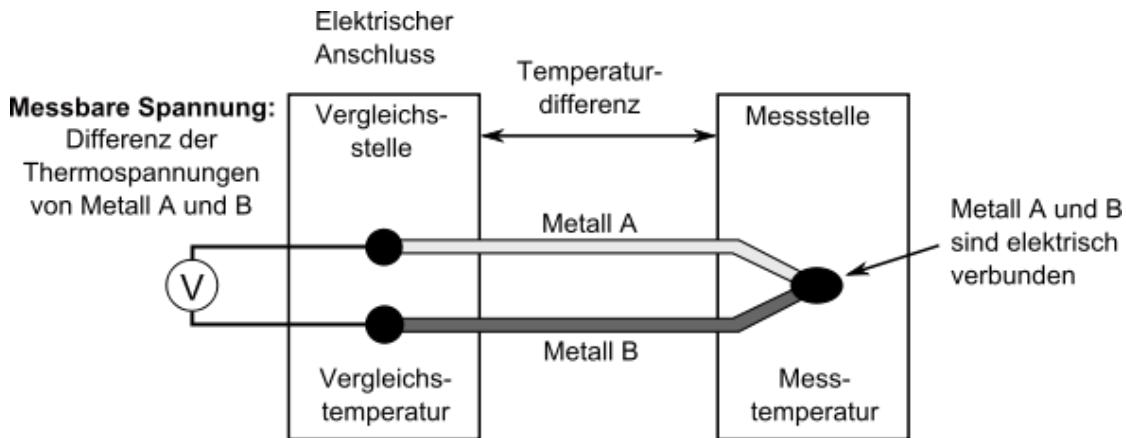


Abbildung 3.3: Prinzip der Temperaturmessung mit Thermopaaren

Thermoelemente werden nach DIN EN 60584 nach ihren Metall-/Legierungspaaren in

Typen klassifiziert. Beispielsweise: Typ K (Ni-CrNi), Typ T (CuCu-Ni), Typ E (NiCr-CuNi). Ersichtlich ist der jeweilige Typ des Thermoelements auch durch die farbliche Kodierung nach DIN EN 60584.

Die Vorteile von Thermoelementen sind eine sehr geringe Trägheit, die Unempfindlichkeit gegenüber der Leitungslänge und der sehr große Messbereich, vor allem bei hohen Temperaturen bis 1700°C. Die Nachteile sind die geringere Genauigkeit, Nichtlinearität und Langzeitstabilität (siehe Abbildung 3.4). Außerdem muss bei Steckverbindern darauf geachtet werden, dass die passenden Materialien verwendet werden, damit keine zusätzlichen Thermospannungen das Messergebnis verfälschen.

Tabelle 1. **Klassen der Grenzabweichungen für Thermopaare** (Vergleichsstellen-Temperatur 0°C *)

Typ	Klasse 1	Klasse 2	Klasse 3 ¹⁾
Typ T			
Temperaturbereich	- 40 °C bis + 125 °C	- 40 °C bis + 133 °C	- 67 °C bis + 40 °C
Grenzabweichung	± 0,5 °C	± 1 °C	± 1 °C
Temperaturbereich	125 °C bis 350 °C	133 °C bis 350 °C	- 200 °C bis - 67 °C
Grenzabweichung	± 0,004 · t	± 0,0075 · t	± 0,015 · t
Typ E			
Temperaturbereich	- 40 °C bis + 375 °C	- 40 °C bis + 333 °C	- 167 °C bis + 40 °C
Grenzabweichung	± 1,5 °C	± 2,5 °C	± 2,5 °C
Temperaturbereich	375 °C bis 800 °C	333 °C bis 900 °C	- 200 °C bis - 167 °C
Grenzabweichung	± 0,004 · t	± 0,0075 · t	± 0,015 · t
Typ J			
Temperaturbereich	- 40 °C bis + 375 °C	- 40 °C bis + 333 °C	-
Grenzabweichung	± 1,5 °C	± 2,5 °C	-
Temperaturbereich	375 °C bis 750 °C	333 °C bis 750 °C	-
Grenzabweichung	± 0,004 · t	± 0,0075 · t	-

Abbildung 3.4: Grenzabweichungen nach DIN EN 60584

Einige Hersteller bieten auch digitale Temperatursensoren an (siehe beispielsweise Abbildung 3.5). Diese sind zusätzlich zum Messfühler mit einem Analog-Digital-Wandler ausgestattet. Das generierte elektronische Signal kann je nach Bauweise durch diverse BUS-Systeme übertragen werden. Im Vergleich zur analogen elektrischen Messung werden direkt verarbeitbare digitale Signale geliefert, dadurch kann die Topologie der Messtechnik simpler gestaltet werden. Auch bieten digitale Signale einen Vorteil bei großen Leitungslängen. Aufgrund der eingebauten Elektronik ist der zum Einsatz geeignete Temperaturbereich jedoch eingeschränkt.

Neben den Platinwiderständen und Thermoelementen werden auch NTC- und PTC-Thermistoren zur Temperaturmessung verwendet. Diese Thermistoren nutzen den temperaturabhängigen Widerstand von Halbleitermaterialien, um Temperaturen zu messen.

NTC-Thermistoren (Negative Temperature Coefficient): NTC-Thermistoren haben einen negativen Temperaturkoeffizienten, was bedeutet, dass ihr Widerstand mit steigender Temperatur abnimmt. Sie sind besonders empfindlich und eignen sich gut für Anwendungen, bei denen eine hohe Genauigkeit und Empfindlichkeit erforderlich

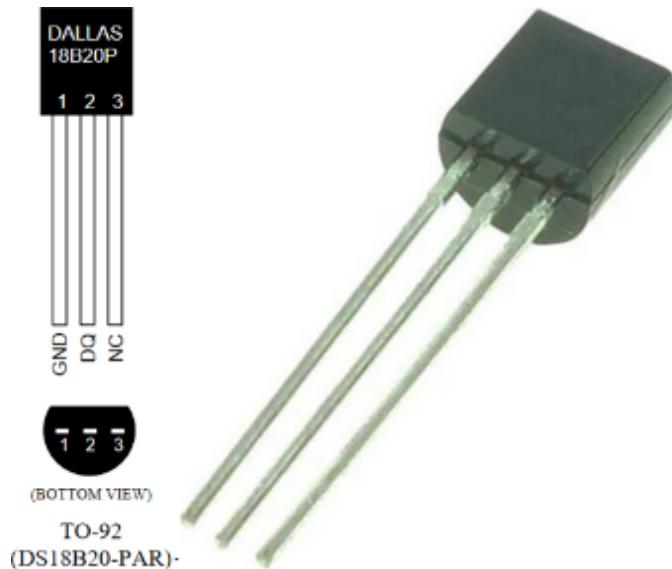


Abbildung 3.5: Digitaler Temperatursensor DS18B20-PAR (Quelle: Maximintegrated)

sind. Typische Einsatzbereiche sind die Messung von Flüssigkeits- und Gas-Temperaturen sowie die Überwachung von elektronischen Bauteilen.

PTC-Thermistoren (Positive Temperature Coefficient): PTC-Thermistoren haben einen positiven Temperaturkoeffizienten, d.h., ihr Widerstand steigt mit zunehmender Temperatur an. Sie werden oft als Temperatursensoren verwendet, aber auch als Überstromschutzelemente, da sie bei Erreichen einer bestimmten Temperatur einen hohen Widerstand erzeugen und somit den Stromfluss begrenzen. PTC-Thermistoren sind weniger empfindlich als NTC-Thermistoren und werden häufig in Haushaltsgeräten und Automobilanwendungen eingesetzt.

3.2.1 Vergleich und Bewertung der Sensortypen

Platinwiderstände (Pt100, Pt1000):

- **Vorteile:**

- Hohe Genauigkeit und Langzeitstabilität.
- Weitgehend lineares Verhalten zwischen Temperatur und Widerstand.
- Normierte Sensoren nach DIN EN 60751.

- **Nachteile:**

- Höherer Kostenaufwand im Vergleich zu Thermoelementen.
- Messfehler durch Leitungswiderstand, besonders bei langen Leitungen.

Thermoelemente:

- **Vorteile:**

- Breiter Temperaturmessbereich.
- Schnelle Ansprechzeit.
- Robuste Bauweise und kostengünstig.

- **Nachteile:**

- Nichtlineares Verhältnis von Temperatur und Spannung.
- Empfindlich gegenüber elektromagnetischen Störungen und Thermospannungseffekten.

NTC-Thermistoren:

- **Vorteile:**

- Hohe Empfindlichkeit und Genauigkeit bei kleinen Temperaturänderungen.
- Kostengünstig und in kleinen Bauformen erhältlich.

- **Nachteile:**

- Stark nichtlineares Verhalten.
- Begrenzt einsetzbar bei hohen Temperaturen.

PTC-Thermistoren:

- **Vorteile:**

- Selbstregulierend bei Übertemperaturschutzanwendungen.
- Robuste und einfache Konstruktion.

- **Nachteile:**

- Weniger empfindlich als NTC-Thermistoren.
- Begrenzte Genauigkeit bei Temperaturmessungen.

Digitale Temperatursensoren:

- **Vorteile:**

- Direkte digitale Ausgabe, die einfach in elektronische Systeme integriert werden kann.
- Weniger anfällig für Signalverluste über lange Kabelwege.

- **Nachteile:**

- Eingeschränkter Temperaturbereich aufgrund der eingebauten Elektronik.
- Potenziell höherer Stromverbrauch und Kosten im Vergleich zu analogen Sensoren.

3.2.2 Praktische Anwendung und Auswahlkriterien

Die Auswahl des geeigneten Temperatursensors hängt stark von der spezifischen Anwendung ab. Für hochpräzise Messungen in Laborumgebungen oder bei kritischen Prozessen sind Platinwiderstände ideal. Thermoelemente eignen sich besser für industrielle Anwendungen mit extremen Temperaturen. NTC- und PTC-Thermistoren sind gut für Anwendungen, bei denen Kosten und Platz eine Rolle spielen, wie in der Elektronik und Haushaltsgeräten. Digitale Sensoren bieten Vorteile in komplexen Systemen, wo eine einfache Integration und zuverlässige Datenübertragung über lange Strecken erforderlich sind.

Bei Temperaturmessungen von Gasgemischen wie der Außen- und Innenraumluft müssen Faktoren, welche eine Verfälschung des Messergebnisses zufolge haben könnten, durch den Messaufbau weitgehend ausgeschlossen werden. Zu diesem Zweck müssen die Sensoren gegen etwaige Strahlungsanteile geschützt werden. Auch die Durchmischung/Umwälzung der Luft und eine freie Umströmung des Sensors, sowie die Vermeidung von Kondensatbildung an der Messstelle muss gewährleistet sein. Es ist wichtig, bei der Auswahl auch die Umgebungsbedingungen, die erforderliche Genauigkeit und die möglichen Störquellen zu berücksichtigen, um eine optimale Messung zu gewährleisten.

3.3 Luftfeuchte

Zur Bestimmung der relativen Luftfeuchte haben sich in der Praxis vor allem Hygrometer etabliert, da deren Anwendung im Vergleich zu anderen Methoden am praktikabelsten ist. Das Hygrometer bedient sich der hygrokopischen Eigenschaften der eingesetzten Materialien, also die Fähigkeit nach einer gewissen Zeit Wasser aus der Umgebungsluft aufzunehmen.

Modernere Bauformen des Hygrometers machen sich meist elektrischen Eigenschaften des hygrokopischen Stoffs zunutze. Kapazitive Sensoren nutzen eine hygrokopische Polymer- oder Keramikschicht als Dielektrikum, so dass mit steigender Luftfeuchte auch die Kapazität des Kondensators steigt (siehe Abbildung 3.6). Dieses Messverfahren zählt zu den wirtschaftlichsten Arten der Luftfeuchtemessung, bietet allerdings eine geringe Langzeitstabilität und muss nach VDI 3789 alle zwei Jahre neu kalibriert werden (vgl. [1]).

Eine weitere Möglichkeit ist die Messung mittels Impedanzsensoren. Dabei wird der ohmsche Widerstand eines hygrokopischen Materials gemessen. Dieser ändert sich mit der Einlagerung von Feuchte.

Psychrometer erheben Messwerte mittels einem nassen und einem trockenen Temperatursensor, Rechnungen oder Tabellen lassen anhand der gemessenen Temperaturen und deren Differenz die Bestimmung der relativen Luftfeuchte zu.

Das Taupunktspiegelhygrometer bedient sich der optischen Bestimmung. Ein Spiegel wird so weit abgekühlt, bis Tauwasseranfall zu beobachten ist. Anhand von Spiegeltemperatur zum Zeitpunkt des Tauwasseranfalls und Umgebungsluftdruck lässt sich die absolute Luftfeuchte, mit Messwerten zur Lufttemperatur auch die relative Luftfeuchte berechnen. Der Taupunktspiegel ist in der Regel am genauesten. Allerdings ist er auf-

grund der langen Messzeiten von mehreren Sekunden und der hohen Anschaffungskosten nur für spezielle Anwendungsbereiche geeignet.

Für die Messung von Luftfeuchte in TMon eignen sich am besten kombinierte Sensoren, die zusätzlich zur Feuchte meistens auch noch die Temperatur messen. Die digitalen Versionen dieser Sensoren sind zum Beispiel der DHT22 als günstigste Variante, sowie der BME280 der Firma Bosch (siehe Abbildung 3.7), der zusätzlich noch den Luftdruck misst.

Bei der Genauigkeit von Feuchtesensoren gilt es zu beachten, dass diese im Bereich von ca. 5-95% relative Feuchte eine höhere Stabilität aufweisen als in den Grenzbereichen. Dies ist in Abbildung 3.8 verdeutlicht.

Ein weiterer Punkt ist die Tatsache, dass Feuchtesensoren einen altersbedingten Drift aufweisen, der entweder eine Neukalibrierung oder einen Austausch erfordert. Die Zeit dafür liegt je nach Sensor zwischen 2 bis 5 Jahren.

3.4 CO₂

Die Messung der Kohlenstoffdioxidkonzentration von Gasen erfolgt in den meisten Fällen mittels nichtdispersiver Infrarotsensoren (NDIR). Dieses Prinzip basiert auf den dämpfenden bzw. absorbierenden Eigenschaften von Gasen bezüglich spezifischer Wellenlängenbereiche. Dafür wird im Messgerät in einer mit dem zu prüfenden Gasgemisch gefüllten Kammer die spezifische Wellenlänge von CO₂ (4,26µm) emittiert und nach dem Durchqueren der Kammer vom Sensor erfasst. Diese wird der gemessenen Lichtintensität in einer mit einem Referenzgas gefüllten Kammer gegenübergestellt. Die Lichtintensität lässt so Rückschlüsse auf die CO₂-Konzentration des Gasgemisches zu (siehe Abbildung 3.9). Nichtdispersive Infrarotsensoren bieten den Vorteil einer hohen Messgenauigkeit und einer langen Lebensdauer, sind aber bei einer Reaktionszeit von bis zu fünf Minuten relativ träge und mit hohen Anschaffungskosten verbunden.

Ebenfalls kann der CO₂-Gehalt der Luft durch elektrochemische Methoden bestimmt werden. Entsprechende Sensoren bestehen aus einer mit einer porösen Membran abgetrennten Reaktionskammer, welche mit einem Elektrolyt gefüllt ist und mindestens zwei katalytische Elektroden aufweist. Bei Kontakt des zu messenden Gasmoleküls mit Elektrolyt und Elektroden wird ein Stromfluss erzeugt. Elektrochemische Gassensoren bieten den Vorteil, relativ genaue Absolutwerte von Gaskonzentrationen bestimmen zu können. Der Einfluss von Temperaturänderungen auf elektrochemische Methoden muss jedoch bei Messungen dringend beachtet werden. Außerdem müssen die Sensoren in kurzen Intervallen (etwa 6-12 Monate) neu kalibriert werden. Aus diesem Grund kommen diese Geräte meistens nur in Laborgeräten zum Einsatz.

3.5 Volumenstrom

Im Zuge des TMon werden Volumenströme von Flüssigkeiten und Gasen bestimmt. Der Volumenstrom [m³/h, l/s] wird dabei aus Messwerten der Strömungsgeschwindigkeit v [m/s] und dem durchströmten Querschnitt A [m²] ermittelt:

3.5.1 Volumenstrommessung bei Flüssigkeiten

Bei der Flügelradmessung wird das Medium im Durchflussgeber an einem Rotor (axial oder radial) vorbeigeführt und dieser in eine Drehbewegung versetzt (siehe Abbildung 3.10). Diese wird über einen magnetischen Impulsgeber proportional zur Strömungsgeschwindigkeit in ein elektrisches Frequenzsignal umgewandelt. Eventuelle Abweichungen der Linearität der Umwandlung können zu Messfehlern führen. Das Signal wird je nach Ausführung des Messgeräts in einer integrierten Auswerteeinheit verarbeitet oder zur Datenverarbeitung weitergeleitet. Die Konstruktionsvarianten unterscheiden sich in Ein- und Mehrstrahlflügelradzähler. Mehrstrahlflügelradzähler bieten infolge einer symmetrischen Anströmung des Flügelrads eine höhere Messstabilität und Messgenauigkeit. Flügelrad-Durchflussmesser sind aufgrund ihrer breiten Einsatzmöglichkeiten in vielen Bereichen etabliert und werden auch bei der Wärmemengenmessung zur Volumenermittlung eingesetzt. Zu beachten sind die jeweiligen Messbereiche der einzelnen Ausführungen. Im dauerhaften Einsatz ist zu beachten, dass das Anlaufverhalten der beweglichen Teile durch Ablagerungen oder auch Abnutzung beeinflusst werden kann. Eine regelmäßige Wartung und Neujustierung ist daher anzuraten.

Eine weitere Möglichkeit, Volumenströme von Flüssigkeiten zu messen, ist mittels Ultraschallmessung (siehe Abbildung 3.11). Das Funktionsprinzip der meisten Ultraschall-Durchflussmessgeräte bedient sich dabei der Frequenzverschiebung akustischer Reflexionen einer bewegten Flüssigkeit. Es wird ein Ultraschallsignal in das Medium emittiert, vom Medium reflektiert und von einem Ultraschallsensor detektiert. Die Differenz der reflektierten zur emittierten Frequenz ist proportional zur Durchflussgeschwindigkeit. Wichtige Kenngrößen bei der Auswahl des Messgerätes sind: die Art des strömenden Mediums, dessen Fließgeschwindigkeit, die Dicke der Rohrwandung sowie der Abstand zwischen Sender und Empfänger. Die Vorteile dieser Methode sind die berührungsreie Messung ohne mechanisch bewegliche Teile, kurzen Messintervallen und auch die Möglichkeit, bereits relativ geringe Volumenströme messen zu können.

Für eine magnetisch induktive Messung des Durchflusses von Flüssigkeiten muss das Medium eine elektrische Mindestleitfähigkeit aufweisen. Messinstrumente dieses Typs machen sich die Eigenschaft zunutze, dass an einem elektrischen Leiter, in diesem Fall das Medium, wenn er durch ein mittels Spulen erzeugtes magnetisches Feld bewegt wird, eine Induktionsspannung anlegt. Diese Spannung ist proportional zur magnetischen Feldstärke, Länge des Leiters und Geschwindigkeit des Mediums und wird an quer zum Magnetfeld angeordneten Elektroden erfasst und an ein Auswertegerät weitergeleitet. Die Länge des Leiters entspricht dem Abstand der Messelektroden. Neben der hohen Messgenauigkeit ist auch berührungsreie Datenerhebung ohne bewegliche Teile aufgrund geringen Wartungsaufwands von Vorteil.

3.5.2 Volumenstrommessung bei Gasen

Der Volumenstrom von Gasen lässt sich mittels Flügelrad-Anemometer (siehe Abbildung 3.12), ebenfalls anhand der Strömungsgeschwindigkeit ermitteln. Analog zur Flügelradmessung bei Flüssigkeiten kommt ein leicht laufendes Windrad und ein Impulsgeber

zum Einsatz. Dabei fallen geringe Strömungsgeschwindigkeiten bei den meisten Modellen aus dem Messbereich. Auch die Wartungsintensität mechanisch bewegter Teile ist zu beachten.

Bei Messungen mit thermischen Anemometern (siehe Abbildung 3.13) wird ein umströmter Sensor mit temperaturabhängigem elektrischem Widerstand elektrisch beheizt. Die vorbeiströmende Luft kühlt diesen ab. Anhand elektrischer Messgrößen wie benötigte Heizleistung und Widerstand kann auf die Luftgeschwindigkeit geschlossen werden. Neben der Erfassung von relativ geringen Strömungsgeschwindigkeiten können bei dieser Messmethode auch richtungsunabhängige Messungen vorgenommen werden.

Das Prandtlsche Staurohr ist eine häufig verwendete Methode zur fest installierten Volumenstrommessung von Gasen. Dieses Prinzip macht sich den quasistatischen Staudruck des Mediums beim Umströmen eines Festkörpers zunutze. Als Teil des Messgerätes ist ein Rohr in Strömungskanal angebracht. Entgegen der Strömungsrichtung ist an diesem ein Drucksensor für den Staudruck angebracht. Je nach Bauart finden sich an weiteren Stellen weitere Sensoren zur Bestimmung des atmosphärischen Drucks. Anhand der ermittelten Druckdifferenz kann über folgenden Zusammenhang die Geschwindigkeit der umströmenden Luft errechnet werden.

Je nach Bauart finden sich unterschiedliche Ausprägungen von Stauelementen und Sensorsanordnungen in gängigen Produkten. Gemeinhin zeichnen sich Prandtlsche Staurohre durch eine geringe Wartungsintensität und eine gute Eignung für hohe Strömungsgeschwindigkeiten aus. Jedoch sind die Messaufbauten richtungsabhängig, empfindlich gegenüber turbulenten Strömungen und geringe Strömungsgeschwindigkeiten fallen meist aus dem Messbereich heraus.

3.6 Beleuchtungsstärke

Die Messung der Beleuchtungsstärke erfolgt in einem Luxmeter (siehe Abbildung 3.14) über Photodioden. Diese Halbleiterdiode wandeln Licht mittels Photoeffekt in elektrischen Strom um oder generieren einen beleuchtungsabhängigen Widerstand. Die gemessene Beleuchtungsstärke wird in Lux [lx] angegeben. Normative Anforderungen zur Lichtmessung finden sich in DIN 5032. Zur Beleuchtung von Arbeitsstätten finden sich in der DIN EN 12464 Anforderungen, die als Richtwerte auch auf andere Innenräume übertragen werden können.

3.7 Solarstrahlung

Zur Messung der Solarstrahlung werden Pyranometer (siehe Abbildung 3.15) herangezogen. In diesen sind geschwärzte elektrische Bauelemente integriert, welche die einfal lende Strahlung absorbieren und sich erwärmen. Die Temperaturdifferenzen zwischen der bestrahlten Oberfläche und dem vor Strahlung geschützten Gehäuseinneren erzeugen Thermospannungen. Diese sind nach Hinzuziehen eines Kalibrierfaktors proportional zur Bestrahlungsstärke. Um den Sensor optimal der einfallenden Strahlung auszusetzen, sind diese horizontal unter einer transparenten Abdeckung angeordnet.

Die gesamt auf der Erdoberfläche auftreffende Solarstrahlung wird als Globalstrahlung G [W/m^2] bezeichnet. Sie ergibt sich aus der auftreffenden direkt von der Sonne emittierten Strahlung und der durch Streuung, Reflexion und Emission durch andere Körper auftretenden Diffusstrahlung. Pyranometer ohne Beschattung messen den Momentanwert der Globalstrahlung. Sollen differenziertere Werte erhoben werden, so ist ein Messgerät mit Verschattungsring oder automatischer Verschattung zur Messung der Diffusstrahlung vonnöten. Der Messort sollte möglichst frei von verschattenden Objekten, wie z.B. Bäumen, etc. gewählt werden. Außerdem sind eventuell reflektierende Oberflächen in der Nähe des Pyranometers zu beachten, da diese die Messgenauigkeit ebenfalls beeinträchtigen können.

Die maximale Strahlung stellt die Solarkonstante dar. Sie entspricht der Globalstrahlung am oberen Rand der Atmosphäre und beträgt 1367 W/m^2 . Anhaltswerte für die lokale durchschnittliche Globalstrahlung bieten die Solarstrahlungskarten der Europäischen Kommission (<http://re.jrc.ec.europa.eu/pvgis/>).

3.8 Messgeräte

Messgeräte sind Instrumente, welche einzelne physikalische Größen messen und aus diesen weitere Messgrößen zusammenfassen. Das wichtigste Messgerät in der Gebäudetechnik ist der sogenannte Energiezähler. Dieser ermittelt unter Berücksichtigung von gemessenen Grundgrößen in einem internen Rechenweg zusätzlich Werte wie zum Beispiel die über den Zähler laufende Leistung. Durch Aufsummieren der Leistung über der Zeit wird schließlich meistens noch die Energie bestimmt. Energiezähler werden zur Messung der elektrischen Energie sowie der thermischen Energie verwendet. Dies schließt auch die Messung des Primärenergieverbrauchs (Gas, Öl, Festbrennstoff) ein.

3.8.1 Wärme- und Kälteenergie

Zur Ermittlung der Wärmeenergie kommen Wärmemengenzähler (WMZ) zum Einsatz. Diese bestehen aus einem Durchflusssensor und Thermosensoren in Vorlauf und Rücklauf des Systems (vgl. Abbildung 3.16).

Anhand der ermittelten Fließgeschwindigkeit wird der Volumenstrom des Mediums ermittelt und mittels der gemessenen Temperaturdifferenz lässt sich die Wärmeleistung bestimmen. Tabelle 3.2 zeigt die für die Berechnung der Wärmeleistung relevanten Größen.

Durch Integration der Leistung wird zudem noch die Energie aufsummiert und als Messwert zur Verfügung gestellt.

Je nach Bauart kommen dabei die zuvor beschriebenen verschiedenen Sensorarten zur Volumenstrom- und Temperaturmessung zum Einsatz. Aufgrund der Anforderung der meist dauerhaften Messung und der erhöhten Temperatur empfiehlt sich die Verwendung von Ultraschall-Durchflusssensoren. Die Durchflusssensoren werden aufgrund der geringeren Temperatur des Mediums meist am Rücklauf des Systems angebracht.

Dasselbe Messprinzip wird auch bei Kältemengenzählern angewandt. Die Temperaturdifferenz von Vorlauf zu Rücklauf ist hierbei jedoch negativ. Die restlichen Anforderungen

Größe	Formelzeichen	Einheit
Wärmeleistung	\dot{Q}	W
Volumenstrom	\dot{V}	m^3/h
Temperaturdifferenz	$\Delta\vartheta$	$^\circ C$
Dichte (Medium)	ρ	kg/m^3
Spez. Wärmekapazität (Medium)	c	$J/(kg \cdot K)$

Tabelle 3.2: Wärmeleistung und relevante Größen

bleiben weitgehend analog zur Wärmemengenmessung. Bei der Wahl des Messgerätes ist gesondert zu beachten, dass Tauwasserbildung an kalten Oberflächen und daraus resultierende Anforderungen an die Ausführung der Technik berücksichtigt werden müssen.

Auch sind Hybridvarianten erhältlich, sogenannte Klimazähler identifizieren selbstständig die Temperaturdifferenz zwischen Vor- und Rücklauf und erfassen diese in getrennten Zählregistern. Sie ermöglichen die simultane Messdatenerfassung und Ausgabe an einem Gerät.

Die Genauigkeit und Anforderungen an diese Messgeräte sind in der DIN EN 1434 angegeben.

3.8.2 Stromzähler

Ein weiterer Zählertyp wird für die Erfassung von Messgrößen des elektrischen Stroms eingesetzt. Neben der Erfassung von thermischen Größen stellt elektrischer Strom eine weitere wichtige Form der verbrauchten Energie dar. Dabei sind für den Zweck des TMon Kombigeräte von Vorteil. Diese messen alle Größen (Spannung, Strom, Leistung, Energie) eines Stromkreises.

Messtechnisch existieren verschiedene Wirkweisen. Dazu gehören Systeme mit induktiven Stromwandlern, Rogowskispulen, Nebenschlusswiderständen oder Hallsensoren. Der wichtigste Unterschied ist dabei die Wandler- sowie die Direktmessung. Bei der Wandlermessung wird eine Spule (Rogowskispule) um das zu messende Kabel gelegt. Der Vorteil dieser Technik ist die Robustheit sowie die Möglichkeit, auch hohe Ströme ohne Schaden der Messtechnik zu messen. Nachteil dabei ist der erhöhte Aufwand der Technik. Es werden unter anderem Verstärker und Wandler benötigt. Ein weiterer, wenn auch kleiner Nachteil ist, dass mit diesen kein Gleichstrom gemessen werden kann.

Eine weitere Möglichkeit sind die Direktmessgeräte. Hier wird das Messgerät in den Stromkreis integriert. Diese Messtechnik ist in der Regel genauer als die Wandlermessung, wobei diese nur bis zu 63 Ampere zulässig sind.

In Abbildung 3.17 sind die elektrischen Stromzähler zu erkennen. Auf der linken Seite ist ein Wandler-, auf der rechten Seite ein Direktzähler.

3.8.3 Kombinierte Sensoren

Oftmals sollen verschiedene Messgrößen am gleichen Ort gemessen werden. Vor allem für das Behaglichkeitsmonitoring werden meistens Temperatur, Luftfeuchte und der CO₂-Gehalt in Räumen gemessen. Hier sollte man auf kommerziell erhältliche kombinierte Sensoren zurückgreifen. Diese werden meistens auch mit einer Hochlevel-Busanbindung wie zum Beispiel Modbus RTU ausgestattet geliefert.

3.8.4 Wetterstation

Wetterstationen dienen der vollständigen Erfassung der Umgebungsbedingungen und werden in der Regel auf dem Dach angebracht. Je nach Ausführung werden neben der Lufttemperatur, -feuchte und -druck auch Windrichtung und -geschwindigkeit sowie die Solarstrahlung gemessen. Auch die Regenmenge kann bestimmt werden.

Um Messfehler zu vermeiden, sind für die Messung im Freien besondere Vorkehrungen zu treffen, um zum Beispiel den Einfluss der Solarstrahlung auf die Temperaturmessung zu vermeiden. In Abbildung 3.18 ist ein typischer Aufbau einer Wetterstation abgebildet.

3.9 Auswahl von Sensoren und Messgeräten

Bei der Auswahl von Sensoren und Messgeräten gilt es zu beachten, dass diese auch für die Messaufgabe geeignet sind. Eine Abweichung zwischen Sensor und Anforderungen führt in der Regel zu falschen Ergebnissen. Es gilt auf die folgenden Punkte zu achten.

- **Messbereich:** Der Messbereich gibt den Arbeitsbereich des Sensors an. Dieser sollte zu den erwartenden Messdaten passen. Dies gilt vor allem für Temperatursensoren und Durchflusssensoren. Liegen die Messwerte außerhalb des Arbeitsbereiches, werden in der Regel falsche Werte geliefert.
- **Genauigkeit:** Unter der Genauigkeit von Sensoren versteht man sowohl die Auflösung der Messwerte als auch den relativen Fehler beim Wiederholen der Messung. Die Auflösung gibt das kleinste mögliche Intervall zwischen zwei aufeinanderfolgenden Messwerten (z.B. 21,0 °C – 21,1 °C 1 Nachkommastelle) an. Da jede Messung mit einer Messungenauigkeit behaftet ist, gibt der relative Fehler in Prozent den Vertrauensbereich des Sensors an. Dieser Wert wird für viele Sensoren und Messgeräte in Genauigkeitsklassen angegeben. Es sollte darauf geachtet werden, dass der relative Fehler bei ca. ±1% liegt. Als maximaler Fehler sollten ±3% nicht überschritten werden. Die benötigte Auflösung gibt dabei die Messaufgabe vor. Ist die Auflösung zu gering, kann es zu widersprüchlichen Auswertungen kommen.
- **Abtastrate:** Die Abtastrate gibt die kleinste mögliche Wiederholungsfrequenz für die Messung an. Dies kann je nach Art der Messung zwischen wenigen Sekunden und einigen Minuten liegen und sollte je nach Anforderungen der Messaufgabe gewählt werden.

- **Trägheit / Zeitkonstante:** Unter der Trägheit bzw. der Zeitkonstante eines Sensors versteht man die Zeit, bei der der Sensor auf eine Änderung der Messgröße reagiert und den richtigen Wert anzeigt.

In Tabelle 3.3 sind Empfehlungen für verschiedene Messgrößen aufgelistet. Es handelt sich dabei um Richtwerte, die in einem Gebäudemonitoring erfasst werden. Bei anderen Anforderungen müssen diese entsprechend angepasst werden.

Messgröße	Messbereich	Genaugigkeit	Abtastrate
Arbeit (el./therm.)	Kontinuierlich (Zähler)		15 min ... 24 h
Temperatur			1 s ... 1 min
Leistung (el./therm.)			1 s ... 10 s
Volumenstrom (Wasser hydraulisch)			1 s ... 10 s
Feuchtigkeit (Luft)			1 s ... 1 min
CO ₂ -Gehalt (Luft)			1 s ... 1 min

Tabelle 3.3: Empfehlungen für die Sensorauswahl

Grundsätzlich sind die Sensoren und Abtastraten an die jeweilige Messaufgabe anzupassen. So reicht für die Ermittlung der Jahresenergiebilanz grundsätzlich eine niedrige Abtastrate. Bei dynamischen Prozessen wie z.B. das Anfahren einer Wärmepumpe sollten höhere Raten angestrebt werden, um Lastspitzen auch mit aufzuzeichnen.

Die gängigsten Sensortypen sind normalerweise genormt bzw. über Eichverfahren festgelegt. Die wichtigsten Normen sind im Folgenden aufgelistet.

- MID Richtlinie MI-004: EU-Eichvorschrift über „Wärmezähler“
- Dt. Eichgesetz: „Gesetz über das Maß- und Eichwesen“
- DIN EN 60751: „Widerstandsthermometer“
- DIN EN 60584: „Thermopaare“
- VDI 3786 Blatt 4: „Meteorologische Messungen Luftfeuchte“

3.10 Fehler beim Einbau von Sensoren und Messgeräten

Nach der Auswahl der Sensoren werden diese an der Messstelle installiert. Dabei kann es zu Fehlern kommen, wenn der Sensor oder das Messgerät an der falschen Stelle platziert wird. Diese Fehler führen meistens zu Unstimmigkeiten in den Messwerten und lassen sich in der Regel nur durch hohen Aufwand feststellen. Die folgenden Regeln und Beispiele sollen einen kleinen Überblick über mögliche Fehlerquellen verschaffen.

- **Keine Verschattung von Temperatursensoren im Außenbereich:** Temperatursensoren reagieren auch auf die Strahlungswärme der Sonne. Diese Strahlungsanteile beeinflussen die Messung. Um dies zu verhindern, sollte der Sensor vor der Sonneneinstrahlung geschützt werden. Dies kann zum Beispiel durch Anbringung des Sensors im verschatteten Bereich geschehen. Ist dies nicht möglich, sollte ein Strahlungsschutzgehäuse verwendet werden. In Abbildung 3.19 ist ein typisches Strahlungsschutzgehäuse, wie es bei Wetterstationen verwendet wird, dargestellt.
- Soll die Temperatur in Hydraulikrohren durch Anlegesensoren bestimmt werden, müssen diese gedämmt werden. Ansonsten wird eine mittlere Temperatur aus Rohr und Umgebung gemessen.
- Dasselbe gilt, wenn Temperatursensoren nicht vollständig in Tauchhülsen eingebracht werden.
- Bei dem Einbringen von Sensoren (z.B. Temperatur) in Unterputzinstallationen gilt zu beachten, dass es hier aufgrund von sonstiger verbauter Elektronik die Abwärme einen falschen Messwert liefert.
- Bei der Betrachtung von Temperaturen in Rohrleitungen sollte stets der Anlagenzustand mit aufgezeichnet werden. Bei Stillstand der Anlagen misst der Sensor die Umgebungstemperatur (z.B. Kellertemperatur anstatt die Soletemperatur).
- Keine Luftdurchmischung bei der Messung von Lufttemperatur und Luftfeuchtigkeit: Bei der Anbringung der Sensoren (vor allem im Innenbereich) sollte auf eine ausreichende Luftdurchmischung geachtet werden. Dies gilt auch für CO₂-Sensoren. Es gilt zu beachten, dass die Messstelle sich nicht in einer Ecke befindet und nicht durch Gegenstände beeinträchtigt wird.
- Falsche Auswahl des Wasserkreises bei Wärmemengenzählern: Dieser Fehler kommt ausschließlich bei Hydraulikkreisläufen mit Mischventilen vor. Dabei sollte beachtet werden, dass die Volumenstrommessung sowie die beiden Temperaturmessungen sich im gleichen Teilabschnitt befinden. Entsprechend alles vor bzw. nach dem Mischer. In Abbildung 3.20 sind die richtigen Positionen markiert.

Es handelt sich dabei um die beiden Temperaturmessstellen (1 bzw. 2), die Heizkreispumpe (3) sowie das 3-Wege-Mischventil (4). Die nach der AMEV optionale Messung der Wärmemenge (5) ist oberhalb der Abzweigung des Mischventils zu entnehmen.

Weitere Fehler, die unabhängig von der Platzierung der Sensoren sind, können bei der Inbetriebnahme der Sensoren passieren.

- **Falsche Auswahl der Sensorkennlinie, Wandlungsfaktoren:** Bei Temperatursensoren mit Widerstandsmessung muss die Sensorkennlinie in der Aufnahmeeinheit hinterlegt werden. Eine falsche Wahl der Kennlinie führt zu Fehlern in den Messwerten. Dabei handelt es sich nur um kleine Abweichungen, wodurch dieser Fehler meistens nicht erkannt wird. Bei Volumenstromsensoren muss

ein Wandlungsfaktor eingegeben werden, der aus der Anzahl der Impulse den korrekten Volumenstrom berechnet. Auch hier führt eine falsche Eingabe zu nicht klar erkennbaren Fehlern bei den Messwerten.

3.11 Fehler beim Betrieb von Sensoren

Neben den Fehlern beim Einbau von Sensoren gibt es auch Fehler, welche sich über die Betriebszeit des Sensors einstellen können. Die Kenntnis über die Möglichkeiten dieser Fehler sind ausschlaggebend für einen sicheren Betrieb über die gesamte Laufzeit.

- Viele Sensoren verändern mit der Zeit die Kennlinie ihres Messsignals, z.B. Alterung von kapazitiven Feuchtesensoren.
- CO₂-Sensoren haben einen sogenannten Alterungsdrift (Verschiebung des „Nullpunktes“). Dieser muss von Zeit zu Zeit nachkalibriert werden.
- Äußere Einflüsse, z.B. Schmutz, Kalk im Wasser bei beweglichen Sensoren, beeinflussen mit der Zeit das Messergebnis (Flügelräder etc.).
- Messleitungen besitzen einen eigenen, kleinen Widerstand. Dieser muss bei allen Widerstandsmessungen herausgerechnet werden (z.B. Pt-Widerstandsthermometer).
- Kabelbrüche oder fehlerhafte Kontakte verursachen einen unendlich großen Leitungswiderstand.

Bei alterungsbedingten Abweichungen hilft nur der Austausch bzw. eine Neukalibrierung.

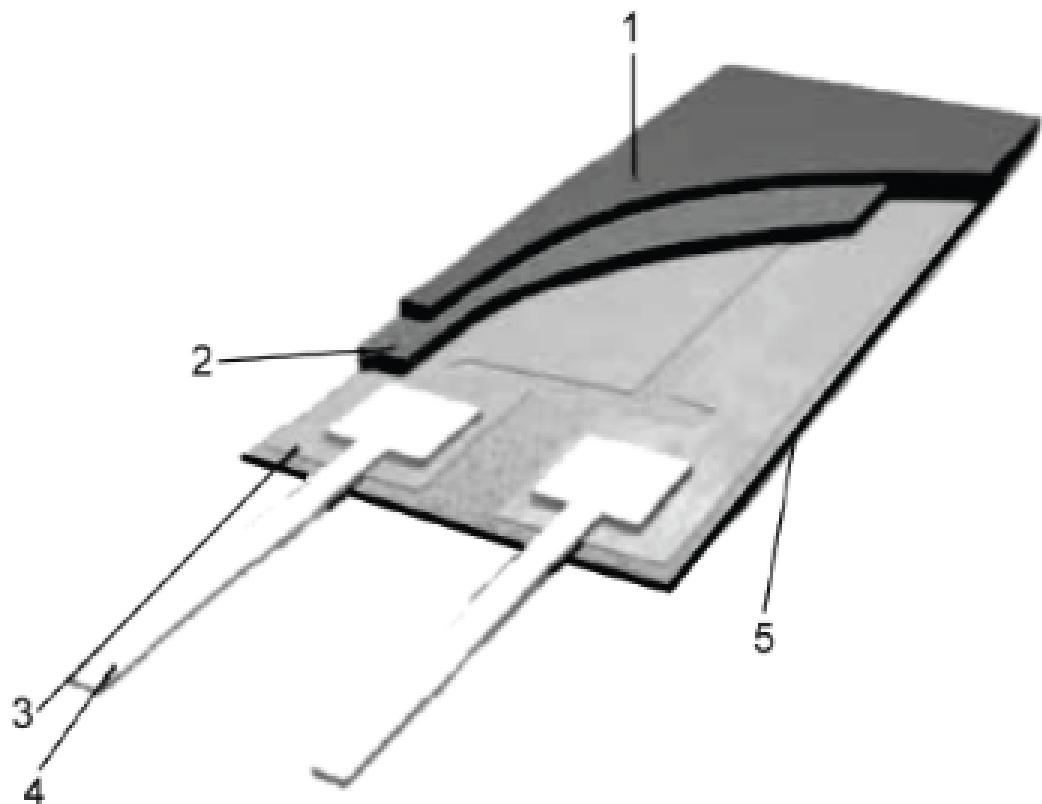


Bild 6. Kapazitiver Feuchtesensor (Quelle: testo industrial services GmbH, geändert)

- | | |
|------------------------------------|---|
| 1 obere, gasdurchlässige Elektrode | 2 hygrokopische Polymer- schicht als Dielektrikum |
| 3 untere Elektrode | 4 Anschlüsse |
| 5 Träger | |

Abbildung 3.6: Aufbau kapazitiver Feuchtefühler

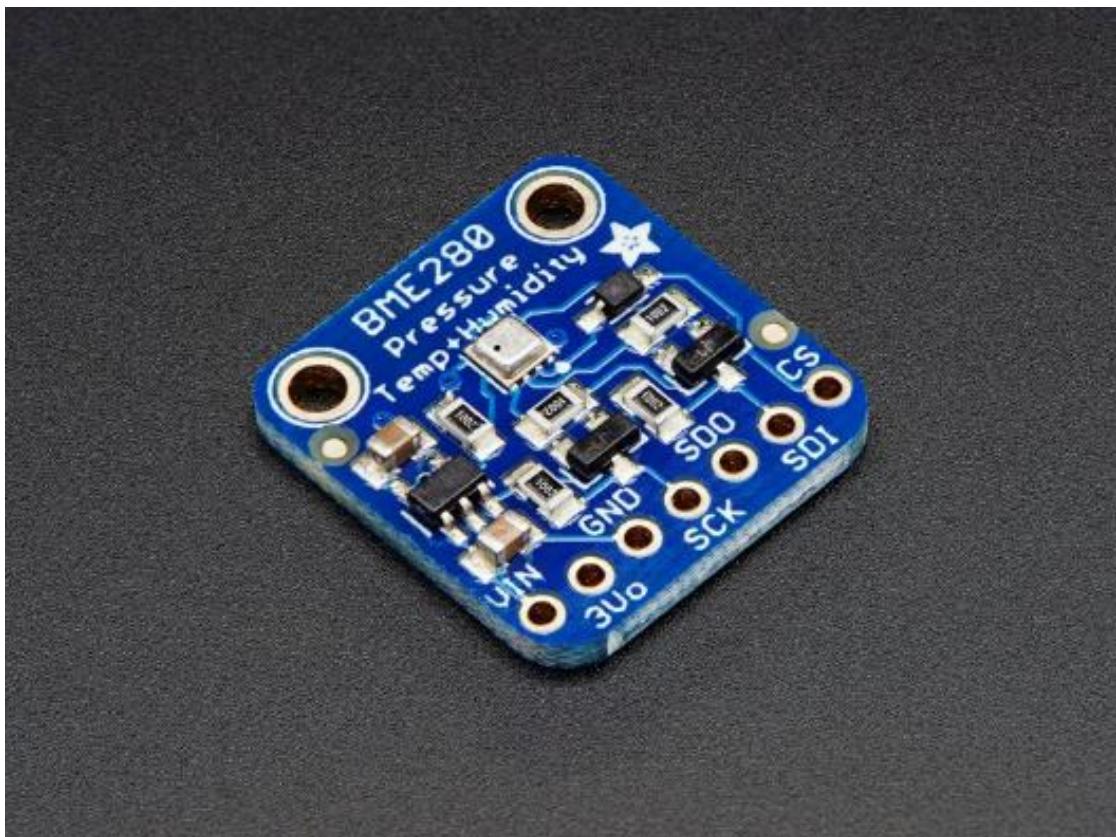


Abbildung 3.7: BME280 (Quelle: Adafruit)

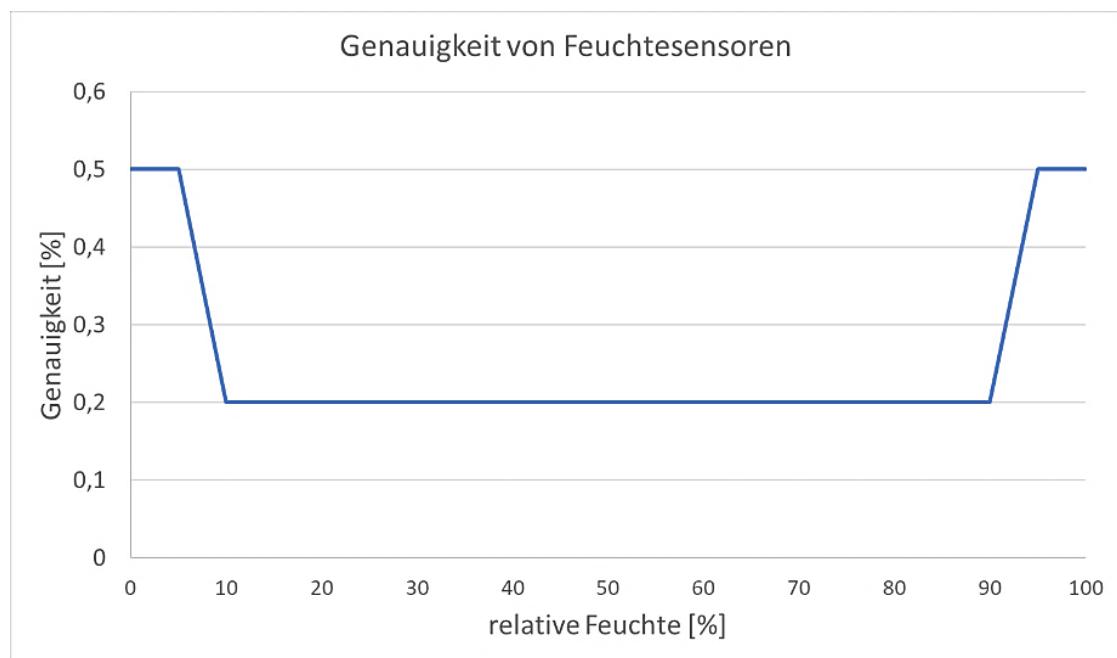


Abbildung 3.8: Genauigkeit von Feuchtesensoren (schematisch)

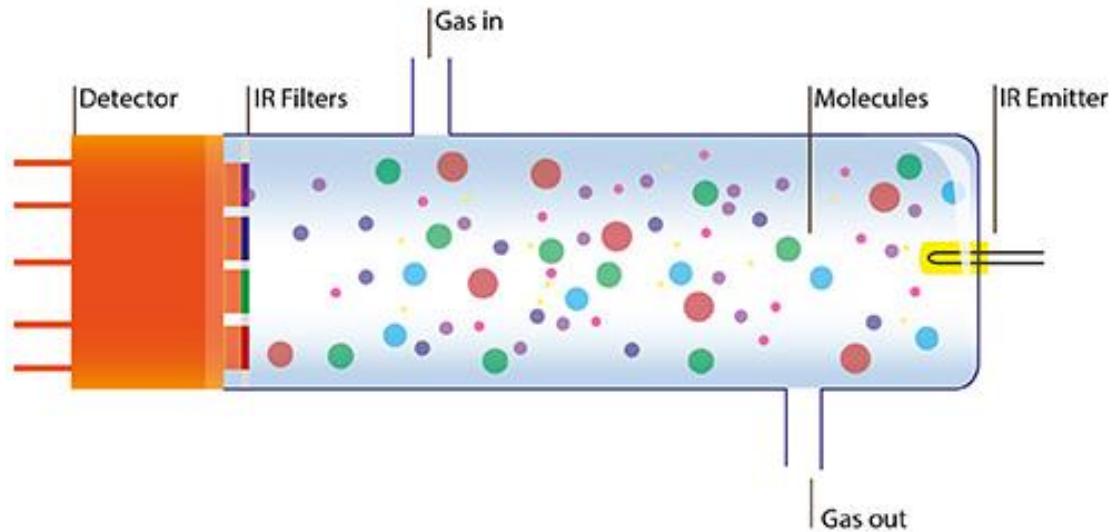
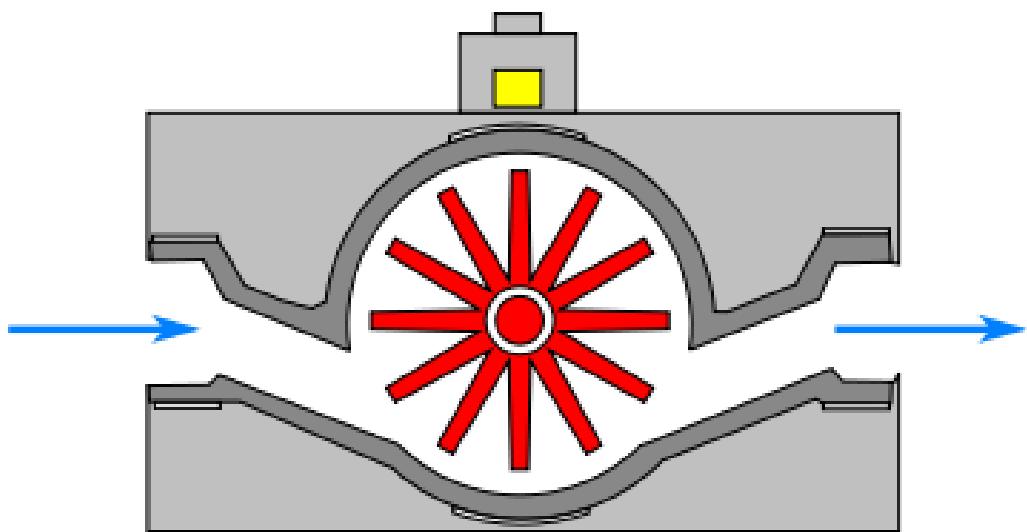


Abbildung 3.9: Prinzip der NDIR Messung (Quelle: Laser Components)



Schnittzeichnung Flügelrad-Durchflussmesser

Abbildung 3.10: Prinzip des Flügelradzählers (Quelle: Wikipedia)

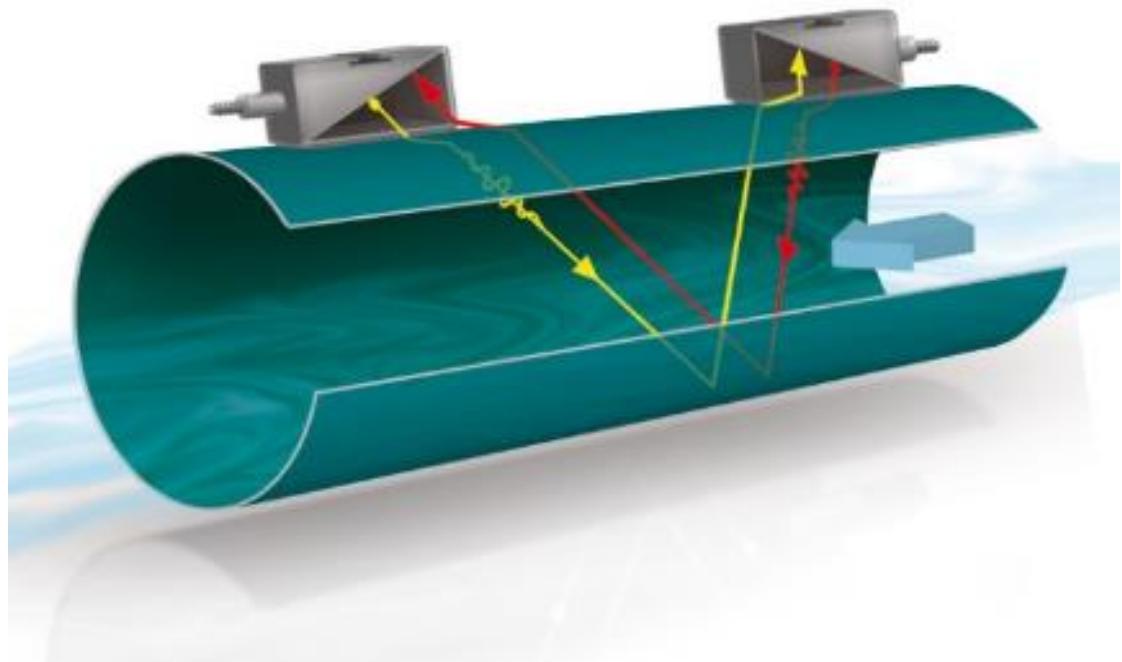


Abbildung 3.11: Prinzip des Ultraschall-Laufzeitverfahrens



Abbildung 3.12: Flügelradsensor [Ahlborn]



Abbildung 3.13: Thermisches Anemometer [Ahlborn]



Abbildung 3.14: Luxmeter [Ahlborn]



Abbildung 3.15: Pyranometer [Ahlborn]

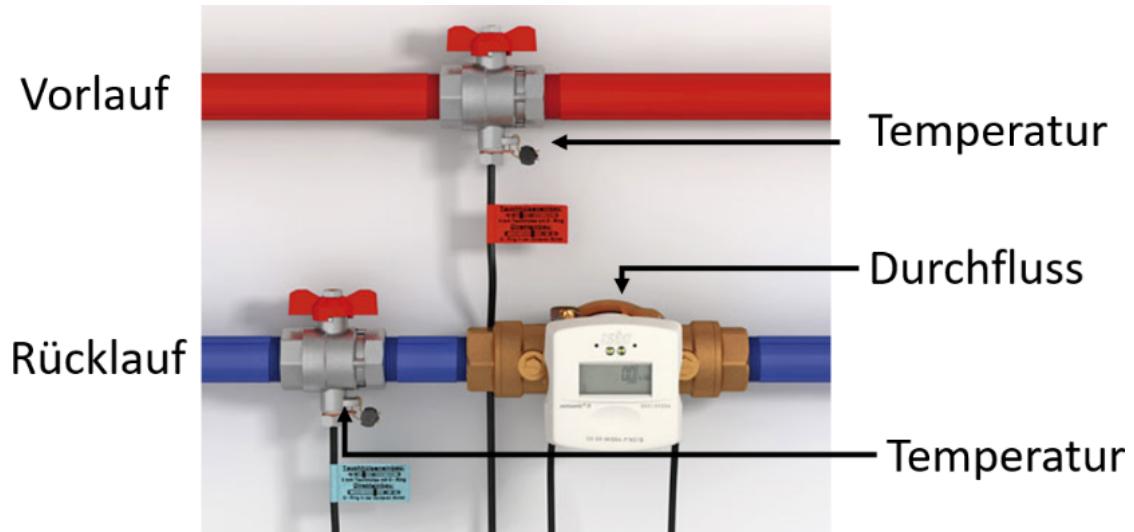


Abbildung 3.16: Prinzip und Einbau von WMZ



Abbildung 3.17: Wandler- sowie Direktstromzähler (Quelle: ABB)

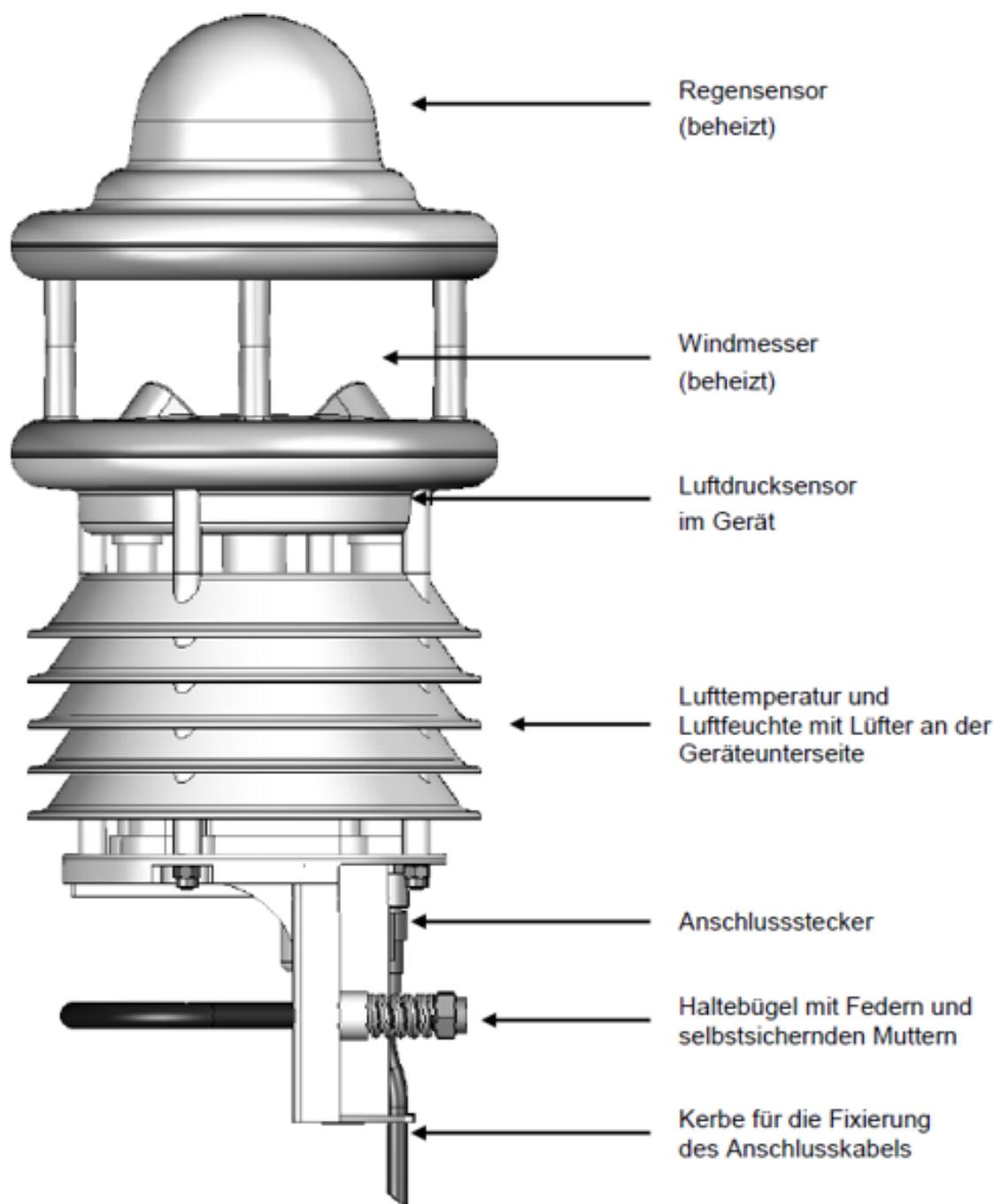


Abbildung 3.18: Aufbau einer Wetterstation (Quelle: Firma Lufft)



Abbildung 3.19: Strahlungsschutzgehäuse für Temperatursensoren im Außenbereich

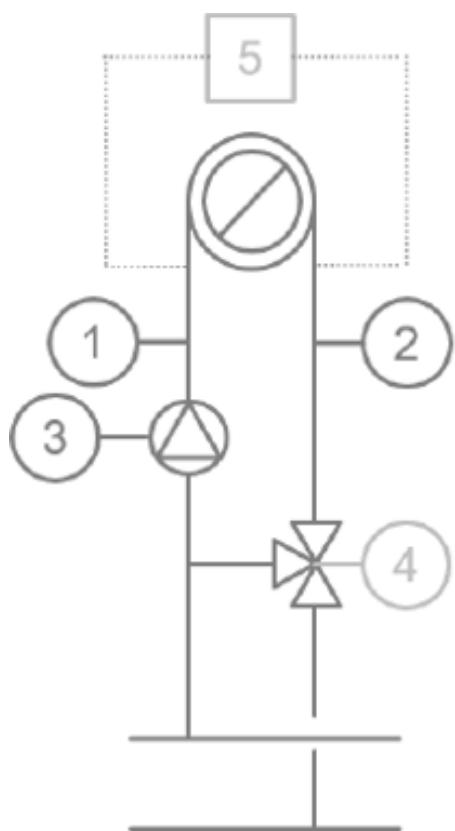


Abbildung 3.20: Anordnung der Messpunkte im Heizkreis (Quelle: AMEV)

4 Erfassung – Messtechnik Topologie

Die Erfassung von Sensordaten lässt sich in Analogie zur Gebäudeleittechnik in drei Ebenen unterteilen (siehe Abbildung 4.1). Innerhalb der Monitoringebene werden die relevanten Daten durch Sensoren generiert. Die Messdaten werden in der Datenebene gesammelt, gebündelt, eventuell zwischengespeichert und weitergeleitet. In der Prozessebene werden die Daten schließlich gespeichert.

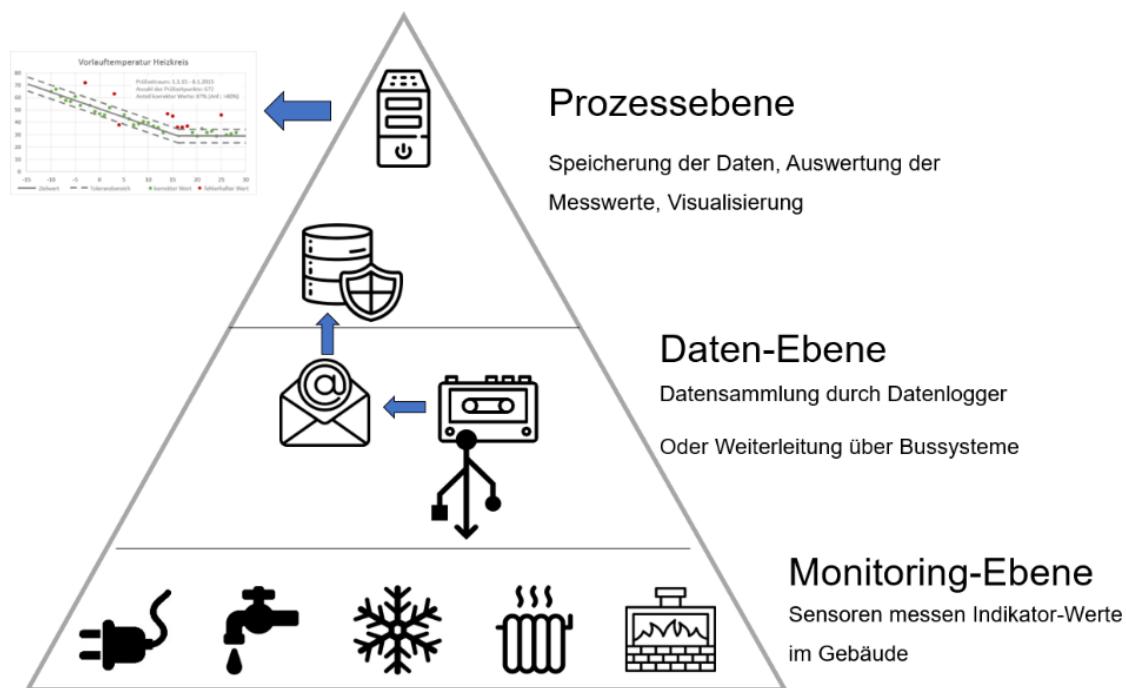


Abbildung 4.1: Topologie eines Monitoringsystems

Einen Hauptaspekt in diesem Kapitel bildet die Datenebene, welche im TMon eine wichtige Stellung einnimmt. Die Datenebene dient zur Weiterleitung der Daten über größere Entfernungen bis hin zum Datenserver, der die Daten in die Datenbank schreibt. Oftmals werden dezentrale „Konzentratoren“ eingesetzt, die mehrere Sensorwerte bündeln und in Datenpakete zusammenfassen. Dies ist immer stark von den räumlichen und technischen Gegebenheiten der Monitoringaufgabe abhängig.

4.1 Messkette

Die Messkette stellt eine Aneinanderreihung von Maßnahmen dar, um einen physikalischen Vorgang digital verarbeiten zu können (siehe Abbildung 4.2). Die physikalische Messgröße wird dabei von einem Sensor aufgenommen, der ein elektrisches Signal gibt. Je nach Messgerät werden elektrische oder elektronische Signale erzeugt. Elektrische Signale werden im Messumformer (Messverstärker) auf ein normiertes Messsignal umgeformt. Normierte analoge Signalbereiche sind bspw. 0-10 V oder 4-20 mA. Mittels Analog-Digital-Umsetzer wird der elektrische Signalbereich in elektronischen Signalen mit einer Zeichenlänge von beispielsweise 32 Bit abgebildet.

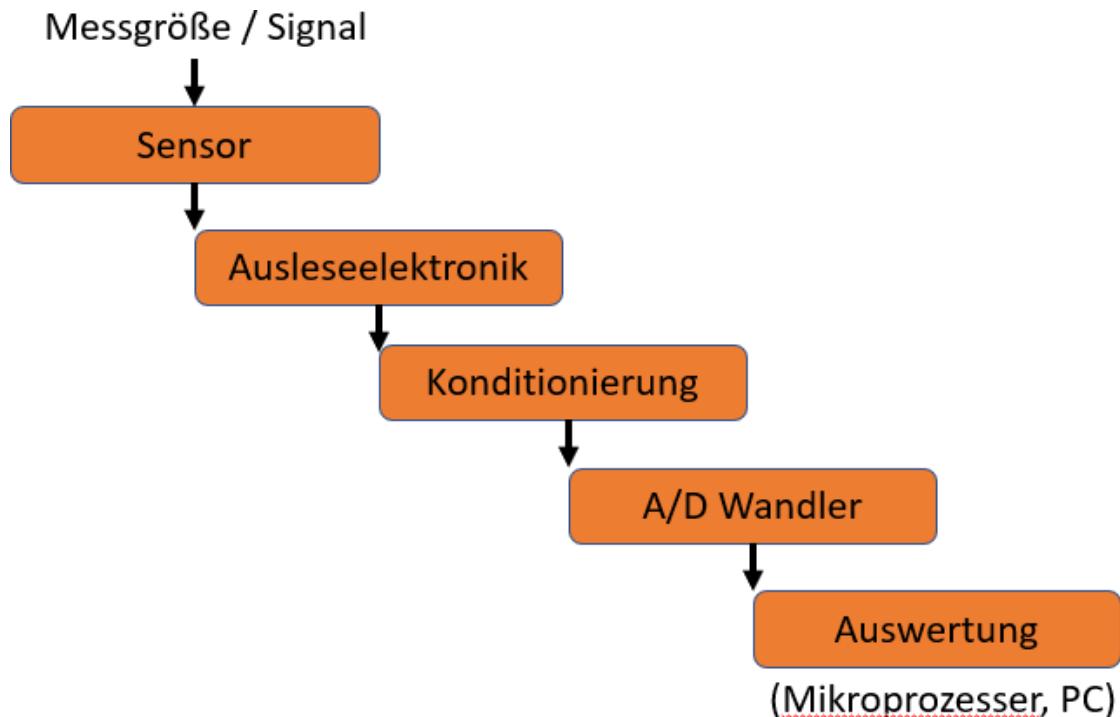


Abbildung 4.2: Messkette innerhalb des Sensors

Der derzeitige Stand der Messtechnik hat eine Vielzahl an Sensoren mit bereits integrierter Elektronik hervorgebracht. Diese Sensoren erzeugen bereits ein digitalisiertes Signal und enthalten meistens eine Anbindung an einen Low-Level Bus (z.B. One-Wire, I2C) oder auch an einen höherwertigen Bus (Modbus RTU/TCP, M-Bus).

4.2 Übertragungsstrecken

Die Übertragungsstrecke beschreibt den Weg der Daten vom Sensor zum Datenspeicherort. Die Übertragungsstrecke lässt sich in der Regel in mindestens zwei Strecken mit unterschiedlichen Aufgaben aufteilen. In Abbildung 4.3 ist dies schematisch dargestellt.

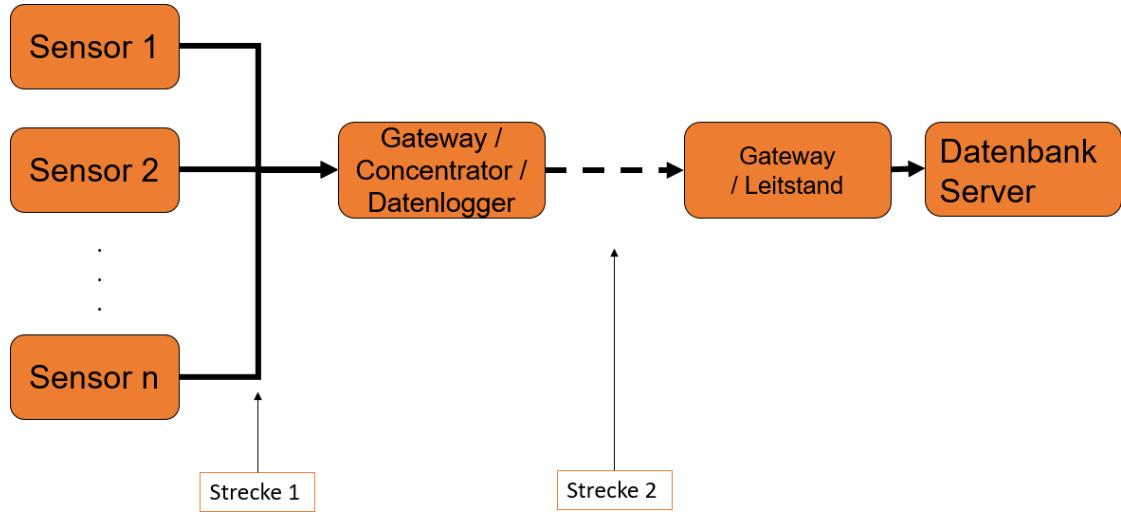


Abbildung 4.3: Schematische Darstellung von Übertragungsstrecken

Als Grundstruktur wird ein System mit zwei Übertragungsstrecken angesehen. Es sind folgende Komponenten vorhanden:

- **Strecke 1:** Die erste Übertragungsstrecke dient der Zentralisierung der Sensor-knoten. Für diese Übertragung werden meistens Bussysteme verwendet, sofern diese durch Kabel realisiert werden. Eine Funkübertragung über WLAN, LoRaWan, XBee und weiteren Möglichkeiten ist ebenfalls möglich. Auch die Anbindung von Sensoren über analoge Signale ist möglich.
- **Gateway:** Der Knotenpunkt der ersten Übertragungsstrecke kann je nach Aus-führung verschiedene Aufgaben übernehmen. So dient es als Gateway als reiner Übersetzer der Daten und reicht diese über die zweite Übertragungsstrecke weiter.
- **Concentrator:** Als Concentrator wird der Knotenpunkt bezeichnet, wenn er Daten von vielen Sensoren bzw. Messsystemen zusammenfasst. Das weiterleitende Proto-koll nach dem Concentrator ist in der Regel anders als vor dem Concentrator.
- **Datenlogger:** Die Möglichkeit einer Zwischenspeicherung der Daten, zum Beispiel zu Redundanzzwecken, wird als Datenlogger bezeichnet.

Die oben genannten Teilaufgaben werden heutzutage meistens zusammengefasst und angeboten. Für kleinere Systeme, vor allem wenn nur zeitlich begrenzt gemessen werden soll, ist es möglich, das Gesamtsystem hier zu beenden. Für alle weiteren Zwecke kommen die folgenden weiteren Komponenten hinzu:

- **Strecke 2:** Diese Übertragungsstrecke dient dem Weiterleiten der Daten an den Leitstand. Da sich dieser meistens in einem anderen Gebäude oder noch weiter weg befindet und auf dieser Strecke auch deutlich mehr Daten übertragen werden

müssen, sind die Anforderungen an diese Strecke erheblich anders als die der ersten Strecke. In der Regel wird dies über eine Internetverbindung, zumindest aber über eine LAN-Verbindung realisiert.

- **Leitstand:** Beim Leitstand handelt es sich um einen physischen Computer (bzw. eine virtuelle Maschine in einer Serverumgebung), der die Daten empfängt und diese an den Datenbankserver weiterleitet. Beide Rechner befinden sich im gleichen lokalen Netzwerk.
- **Datenbankserver:** Auf dem Datenbankserver werden die Daten schließlich in die Datenbank geschrieben. Aus Sicherheitsgründen und Netzwerkarchitekturgründen können Leitstand und Datenbankserver getrennt werden. Meistens jedoch befinden sich beide auf dem gleichen physischen Computer bzw. Server.

4.3 BUS-Systeme

Die Kommunikation zwischen Monitoring- und Datenebene geschieht in den meisten Fällen über ein Bussystem. Dieses Binary Unit System ist dabei für die Übertragung der Daten zuständig und ermöglicht das Senden getrennter Datenpakete über einen gemeinsamen Übertragungsweg. Bussysteme werden überall dort eingesetzt, wo größere Datenmengen zu erwarten sind, und bieten dabei einen verringerten Aufwand gegenüber klassischer Verdrahtung. Da über eine Busleitung binäre Signale gesendet werden, können auch große Leitungslängen ohne Datenverlust realisiert werden. Zudem können auch mehrere Teilnehmer an einem Kabel angeschlossen werden. Die Anzahl der möglichen Teilnehmer variiert je nach verwendetem System, hat aber in der Regel eine Höchstgrenze von ca. 250 Teilnehmern. Die Teilnehmer werden durch einzigartige Adressen identifiziert.

4.3.1 Physikalische Busebene

Die Busleitung besteht aus zwei Drähten, die je nach Bussystem verschiedenen Anforderungen nach Stabilität, Abschirmung, Querschnitt, Anschlusstechniken genügen müssen. Für die Übertragung werden in der Regel Spannungsgrenzen bestimmt, in welchen die Signale in binärer Form (0 bzw. 1) übertragen werden. In Abbildung 4.4 ist diese Übertragung dargestellt. Aufgrund dieser Grenzen können auch (geringe) Schwankungen in der Übertragungsspannung ausgeglichen werden. Dabei ist es nötig, die Übertragungsgeschwindigkeit von Sender und Empfänger zu synchronisieren, damit die Übertragung reibungslos funktioniert. Diese als Baudrate bezeichnete Einheit gibt die Anzahl der übertragenen Symbole pro Sekunde an. 1 Baud ist dabei die Übertragung eines Symbols pro Sekunde. Gängige Baudraten sind unter anderem 320, 9600, 115200 Baud.

In der Regel wird bei Bussystemen eine Master/Slave-Architektur verwendet (siehe Abbildung 4.5). Der Master übernimmt die Koordination über die Kommunikation, indem er Anfragebefehle sendet, auf welche die angesprochenen Slaves entsprechend der Anfrage antworten.

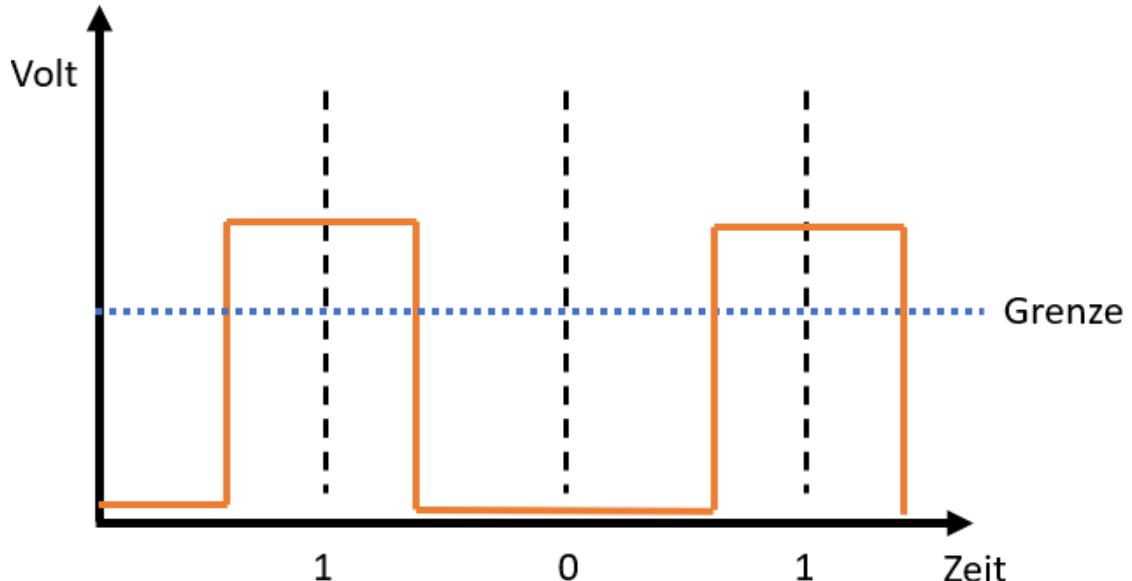


Abbildung 4.4: Darstellung der Signalübertragung in Bussystemen

Als Master gibt es verschiedenste Ausführungen. Der einfachste Master ist ein sogenannter Pegelwandler. Dieser transformiert die Eingangsspannung auf die entsprechende Busspannung. Dabei muss jedoch eine zusätzliche Computer- oder Microcontrollereinheit die Signale vorgeben.

Das in Abbildung 4.6 dargestellte System besteht aus einem Computer mit integrierter serieller Schnittstelle, die mit einer Übertragungsspannung von ± 5 Volt arbeitet. Für den Betrieb des Bussystems wird in diesem Fall ± 10 Volt benötigt. Dies entspricht der Spannung des weit verbreiteten Modbus. Dazwischen sitzt der Pegelwandler. Dieser arbeitet rein physikalisch, indem er die Grenze zwischen den verschiedenen Spannungen darstellt. Die Daten werden dabei nicht verändert.

Kommerziell können Pegelwandler auch durch zusätzliche Funktionen erweitert werden:

- **Datenlogger:** Busmaster mit eingebauter Datenspeicherung.
- **Gateway:** Verbindung von Bussystemen mit unterschiedlichen Signal- bzw. Datenstrukturen, z.B. Bus auf LAN/Internet.

Als Busteilnehmer oder Knoten werden alle Komponenten bezeichnet, die an der Busleitung angeschlossen sind. Um die Kommunikation mit mehreren Feldgeräten zu ermöglichen, wird auf normierte Protokolle zurückgegriffen. Innerhalb des Protokolls dienen Syntax (Kennung), Semantik (Befehl) und Initiative (Übertragungsrate) der geregelten Kommunikation. Jedem Busteilnehmer wird eine eindeutige Adresse zugewiesen, über die er angesprochen werden kann. Diese Adresse wird je nach Bussystem automatisch über eine Software vergeben, vom Hersteller voreingestellt oder wird manuell am Busteilnehmer vergeben. Für eine Parametrisierung stehen verschiedene Softwarelösungen

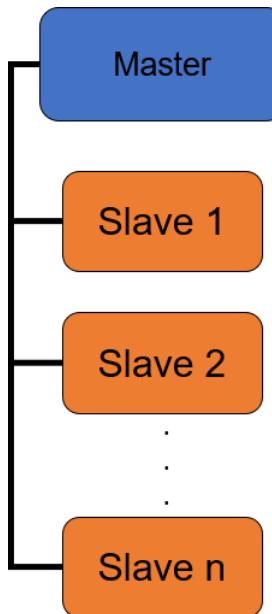


Abbildung 4.5: Master/Slave Schema

zur Verfügung. Für eine durchgängige Kompatibilität zwischen verschiedenen Systemen stehen Buskoppler, Analog-Digitalwandler und Gateways zur Verfügung. Für die kabelgebundene Anbindung der Sensoren bietet es sich an, die einzelnen Stränge nicht in Reihe anzuordnen. Empfehlenswert ist eine Sternstruktur mit dem Master als zentrales Glied. Vorteile hierbei sind die geringere Störanfälligkeit sowie der verminderter Aufwand beim Nachrüsten von weiteren Sensoren bzw. Messgeräten.

Die am häufigsten verwendeten Bussysteme in der Datenerfassung, Gebäudeautomation und Gebäudeleittechnik sind:

- **M-Bus:** Wird als Standard für Verbrauchsdatenerfassung durch Zähler verwendet.
- **Modbus:** Ist entweder als reines Bussystem als Modbus RTU oder über Ethernetverkabelung als Modbus TCP im Einsatz.
- **KNX:** Herstellerübergreifendes Bussystem. Wird hauptsächlich zur Steuerung von Licht, Jalousien und ähnlichem verwendet. Es ist jedoch auch eine Datenaufzeichnung über manche Geräte möglich.

Die gängigsten Systeme im TMon sind M-Bus und Modbus. Beide bieten den Vorteil, große Kabellängen zu unterstützen und die Messgeräte über die Verkabelung mit Energie versorgen zu können. In der Gebäudetechnik kommt häufig der M-Bus als Kommunikationssystem zum Einsatz. Für dieses System ist die Kommunikation über die DIN EN 13757 einheitlich festgelegt. Von Vorteil ist die Verkabelung mit zweiadrigem Kabeln, die simpel und kostengünstig ist. Allerdings benötigt das Kommunikationsprotokoll, je nach Topologie, einige Zeit (wenige Sekunden bis mehrere Minuten) zur Abfrage der Messdaten

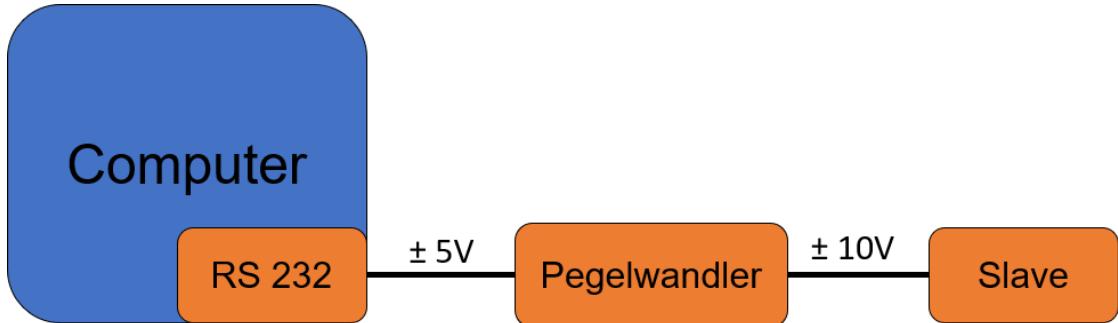


Abbildung 4.6: Schematische Darstellung eines Pegelwandlers

und eignet sich daher nicht für hohe Messfrequenzen. Der Modbus hingegen ermöglicht durch ein anderes Kommunikationsprotokoll und die Nutzung des Leitsystems RS485 deutlich kürzere Kommunikationsintervalle. Die Infrastruktur des Modbus benötigt jedoch vieradrige Kabel mit Abschirmung, die in der Anschaffung mit deutlich höheren Kosten verbunden sind. Die Wahl des Bussystems sollte daher an die Anforderungen des TMon angepasst erfolgen.

4.3.2 Protokollebene

Als Protokollebene wird die Sprache des Bussystems bezeichnet. Es ist eine festgelegte Vereinbarung, nach der Daten zwischen den Komponenten ausgetauscht werden. Die Daten werden stets in einer dem Protokoll entsprechenden Form in Datenpaketen übertragen. Diese Datenpakete enthalten dabei die Informationen für die Kommunikation innerhalb des Busses.

Betrachtet man die Aufgabe des Monitorings, so werden in der Regel Sensordaten über den Bus abgefragt. Im Folgenden wird als Beispiel die Abfrage eines Modbus Sensors näher betrachtet. Prinzipiell läuft dies auch in anderen Systemen ähnlich und muss entsprechend angepasst werden.

Bei Modbus-RTU handelt es sich um eine Master/Slave-Kommunikation. Dies bedeutet, dass der Sensor (Slave) nur dann eine Nachricht schickt (Response), wenn er vom Master dazu angewiesen wurde (Query). Für die Anforderung eines Messwertes schickt der Master dazu eine Nachricht, welche folgende Inhalte enthält:

- **Adresse:** Jede Komponente in einem Bussystem besitzt eine eindeutige Adresse. Jeder Slave reagiert nur auf Nachrichten, die seine Adresse enthalten. Meistens können in einem Busstrang bis zu 250 Komponenten enthalten sein.
- **Funktionscode:** Der Funktionscode gibt an, was der Teilnehmer tun soll. Für die Abfrage von Daten gibt man den Funktionscode „03“ an. Der Funktionscode „03“ bedeutet in der Modbussprache „Read Coil“, da früher Daten auf Speicherspulen gelagert wurden.

- **Datenabfrage:** Es wird angegeben, welche Daten abgefragt werden sollen. Modbus Sensoren speichern die Messdaten in sogenannten Registern ab. Für die Abfrage wird angegeben, bei welchem Register begonnen werden soll und wie viele Register abgefragt werden sollen. Die Angabe, welche Register welche Daten enthalten, sind in der Regel in der Bedienungsanleitung des Sensors zu finden.
- **Prüfsumme:** Jede Nachricht enthält eine Prüfsumme, die nach einem bestimmten Muster aus den vorhergehenden Inhalten gebildet wird und dient als Redundanzprüfung, ob die Nachricht korrekt übertragen wurde.

Zusätzlich muss noch der Start und das Ende der Nachricht bestimmt werden. Beim Modbus-RTU erfolgt dies durch eine Pause von 3,5 Zeichen (Baud). Eine andere Möglichkeit ist die Bestimmung eines fest definierten Zeichens. So beginnt eine XBee-Nachricht immer mit dem Zeichen 0x7E (Hex-Schreibweise).

Als Beispiel lesen wir einen fiktiven Sensor aus. Vom Modbus Master wird folgender Befehl gesendet:

11 03 006B 0003 7687

Modbus-Befehle sind binär und werden in der Regel im Hexadezimalsystem dargestellt. Zur Kennzeichnung des Hexadezimalsystems sieht man auch häufig folgende Schreibweise 0x10. Es handelt sich dabei um die Zahl 10 im Hexadezimalsystem, welche im Dezimalsystem dem Wert 16 entspricht. Die Nachricht ist in die folgenden, bereits besprochenen Bestandteile aufgeteilt:

- **11:** Die Adresse des Slaves hat die Nummer 17, die im Hexadezimalsystem 11 entspricht.
- **03:** Der Funktionscode zum Auslesen von Daten.
- **006B:** Dies ist das erste auszulesende Register 107. Dabei gilt es zu beachten, dass Register in vier Tabellen aufgeteilt sind. Die Tabelle mit Sensordaten ist in der Regel die Tabelle 4. Weiterhin gilt zu beachten, dass diese Tabellen einen Offset enthalten. So beginnt das erste Register der 4. Tabelle mit dem Wert 40001. In unserem Beispiel ist das erste auszulesende Register die Nummer 40108.
- **0003:** Die Anzahl der zu lesenden Register, es werden demnach die Register von 40108 bis 40110 ausgelesen.
- **7687:** Die Prüfsumme, die nach einer bestimmten Vorschrift aus den vorherigen Bestandteilen der Nachricht berechnet wird.

Das entsprechende Gerät gibt als Antwort folgende Nachricht zurück:

11 03 06 AE41 5652 4340 49AD

- **11:** Die Adresse des Slaves.
- **03:** Der Funktionscode.

- **06:** Die Länge der Daten in Byte. Ein Register ist immer 16 Bit groß, was 2 Byte entspricht. Da in der Anfrage 3 Register ausgelesen werden, ist der Inhalt (Payload) 6 Bytes groß.
- **AE41:** Die Werte des ersten Registers, aufgeteilt in einen High-Byte-Wert (0xAE) sowie einen Low-Byte-Wert (0x41). Die einfachste Interpretation dieses Registers ist das Übersetzen des Wertes 0xAE41, der im Dezimalsystem dem Wert 44609 entspricht. Diese Interpretation entspricht der Vorschrift „Big Endian“. Die Werte werden (umgangssprachlich gesagt) von vorne nach hinten eingelesen. Im Gegensatz dazu muss man beim „Little Endian“ die Zahl von hinten nach vorne auslesen.
- **5652:** Die Werte des zweiten Registers.
- **4340:** Die Werte des dritten Registers.
- **49AD:** Die Prüfsumme.

Zur richtigen Interpretation der Werte ist es noch nötig, den Datentyp der Messpunkte zu kennen. Diese sind der Registertabelle des Sensors zu entnehmen.

Im obigen Beispiel wurde die Zahl 0xAE41 in den Wert 44609 übersetzt. Dies entspricht einem 16-Bit-Ganzzahl (Integer) Wert ohne Vorzeichen. Der Wertebereich dieses Datentyps liegt zwischen 0 und 65535. Weitere Datentypen sind unter anderem wie folgt:

- 16-Bit-Integer mit Vorzeichen. Der Wertebereich liegt zwischen -32768 und 32768.
- 32-Bit-Integer ohne Vorzeichen. Der Wertebereich liegt zwischen 0 und 4.294.967.295. Hier werden zwei Register zu einer Zahl zusammengefasst.

Zum Übersetzen der Zahlenwerte existieren in sämtlichen Programmiersprachen entsprechende Funktionen.

4.4 Funkübertragungsstrecken

Sofern die Monitoringaufgabe das Verlegen von Kabeln nicht oder nur bedingt zulässt, ist die Übertragung der Daten auch auf Basis von Funktechnologien ausführbar. Dies kann zum Beispiel bei Bestandsgebäuden nicht sinnvoll oder bei zeitlich begrenzten Monitorings nicht wirtschaftlich sein.

Im Bereich der Funkübertragung sind derzeit unter anderem folgende Technologien verfügbar:

4.4.1 Wireless LAN (WLAN)

Eine Vielzahl von Sensoren und Messsystemen können direkt ins WLAN eingebunden werden. Weiterhin sind WLAN-Gateways für Sensordaten erhältlich und die IoT-Computer wie der Raspberry Pi enthalten bereits WLAN. Die Vorteile sind die Nutzung einer, in den allermeisten Fällen, vorhandenen Infrastruktur. Aufgrund von Datenschutzgründen ist es

aber besser, ein separates Netzwerk aufzubauen. Aufgrund der weiten Verbreitung der Technologie sind die dazu benötigten Komponenten günstig erwerbbar sowie deren Einrichtung sehr einfach. Weiterhin können auch höhere Kommunikationsprotokolle einfach umgesetzt werden. Die Nachteile von WLAN sind der hohe Stromverbrauch, weshalb es meistens nur für netzbetriebene Sensoren geeignet ist, sowie die beschränkte Reichweite.

4.4.2 LoRaWAN

Long Range Wide Area Network (LoRaWAN) ist eine relativ neue Technologie, die es ermöglicht, Sensordaten über sehr weite Entferungen (bis zu mehreren Kilometern bei direkter Sichtverbindung) zu senden. Eine Vielzahl an Herstellern bietet LoRaWAN-Sensoren an. Aufgrund des Open-Source-Charakters der Technologie sind Gateways öffentlich zugänglich. Ist ein Gateway in der Nähe, können die Sensoren direkt mit diesem verbunden werden. Ein weiterer Vorteil ist der sehr geringe Stromverbrauch der Sensoren, die meistens mit einer Akkuladung bis zu 10 Jahre funktionieren können. Nachteile sind hingegen die geringe Datenrate sowie die beschränkte Übertragungsmenge (Anzahl der Übertragungen pro Stunde bzw. Tag).

4.4.3 XBee

XBee sind Funkmodule, mit denen ein serielles Signal per Funk übertragen werden kann. Ein Vorteil dieser Technologie ist der Aufbau eines Mesh-Netzwerks, welches die Übertragung über mehrere in Reichweite befindliche Module (sogenanntes Hopping) ermöglicht. Daher ist diese Technologie im Gebäudebereich, vor allem

bei Raumüberwachungen, sehr breit einsetzbar. Vorteile sind die bereits genannte Mesh-Fähigkeit, der geringe Energieverbrauch sowie eine bereits eingebaute Verschlüsselung bei der Übertragung. Nachteilig hingegen ist der hohe Aufwand bei der Installation und Wartung des Netzwerks.

4.4.4 Mobilfunk (5G)

Eine weitere Möglichkeit der Funkübertragung ist die Nutzung des Handynetzes. Hierzu stehen verschiedene Router zur Verfügung, die als Gateway zwischen WLAN und Handynetz fungieren. Vorteile sind die hohen Datenraten. Nachteile hingegen sind die hohen Kosten aufgrund der Notwendigkeit eines Mobilfunkvertrags.

4.5 Speicherprogrammierbare Steuerung

Möchte man ein Gebäude monitoren, welches über eine Gebäudeleittechnik (GLT) verfügt, ist es sinnvoll, die Daten direkt aus der GLT herauszuziehen. Bei der Hardware einer GLT handelt es sich um eine speicherprogrammierbare Steuerung (SPS oder PLC – Programmable Logic Controller). Diese dient zur Steuerung von Anlagen oder dem Abfragen und Loggen von Messwerten. SPS bestehen in der Regel aus einem digital programmierbaren Mikroprozessor, mehreren Eingängen zum Anschluss von Sensoren, Ausgängen zur

Regelung und einem Speichermedium. Die Bauarten reichen dabei von sehr kompakten Bauweisen bis zu PC-Panels und PCs und sind von verschiedenen Herstellern erhältlich.

SPS von führenden Herstellern sind heutzutage modular erhältlich. Ein Controller übernimmt die Aufgabe der Steuerung, während verschiedene Module die Eingänge und Ausgänge (sowohl in digitaler als auch analoger Form) je nach Bedarf hinzufügbar sind. In Abbildung 4.7 ist der Controller auf der linken Seite zu erkennen, und die Module rechts daneben.

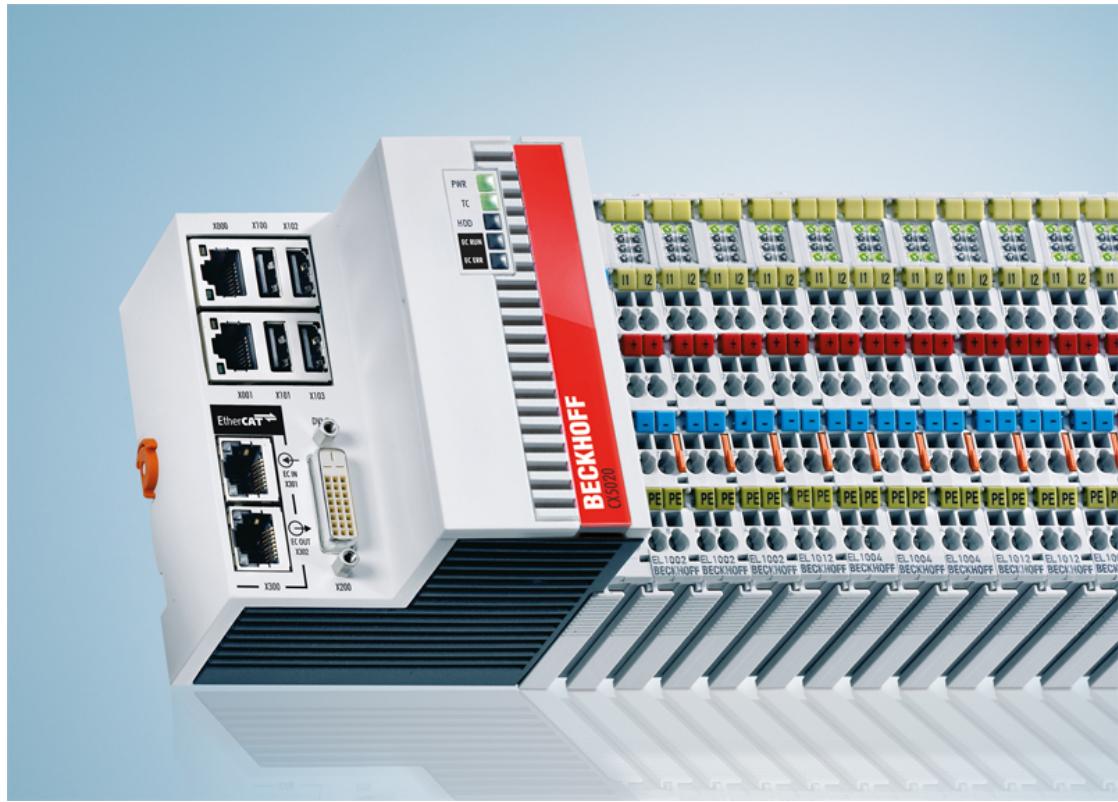


Abbildung 4.7: Beispielhafte Abbildung einer SPS [Beckhoff]

Die Module sind unter anderem auch spezialisiert erhältlich. So gibt es zum Beispiel Eingangsklemmen, die direkt auf Thermoelemente ausgelegt sind. Diese können direkt in die Klemme eingesteckt werden und haben nach der korrekten Konfiguration bereits die Kennlinie enthalten.

Neben den vielen Vorteilen bieten SPS jedoch auch den Nachteil, dass diese in einer speziellen Programmierungsumgebung des Herstellers konfiguriert werden müssen. Somit entsteht ein Mehraufwand.

5 Datenspeicherung

Die Speicherung der Daten kann auf verschiedenste Art und Weisen erfolgen. Je nach Größe des Gesamtsystems können heutzutage mehrere Tausend Datenpunkte zusammenkommen. Die Speicherung der Messdaten selbst stellt aufgrund günstig verfügbarer Festplatten keine große Herausforderung mehr dar. Viel wichtiger ist es, den Datensatz in einer vernünftigen Struktur zur Verfügung zu haben, die ein zügiges Filtern der Daten zulässt. Weitere Punkte, auf die man achten sollte, sind eine schnelle Verfügbarkeit sowie die Möglichkeit, die Daten zur Überprüfung darstellen zu können.

Am wichtigsten ist jedoch, stets die Rohdaten zu speichern und diese in keiner Weise, auch wenn sie nicht plausibel erscheinen, zu verändern. Alle Auswertungen müssen auf Basis des Rohdatensatzes erfolgen. Zusätzlich sollte man über die gesamten Messdaten stets die gleichen Einheiten für die physikalischen Größen verwenden.

Im Folgenden werden verschiedene Methoden zur Speicherung der Monitoringdaten aufgezeigt.

5.1 CSV-Datei

Eine immer noch weit verbreitete Lösung ist das Speichern der Messdaten in einer comma separated file (CSV). Bei einer CSV-Datei handelt es sich im Prinzip um eine einfache Textdatei, deren spezielles Format die Daten in einer für andere Programme (z.B. Microsoft Excel) lesbaren Form darstellt. Die Daten sind durch Kommas oder Semikolons getrennt. Die Wahl des Trennungszeichens hängt mit der Schreibweise von Kommabehaf-teten Werten zusammen. Während in Deutschland die Schreibweise mit Komma gängig ist, wird als Trennzeichen das Semikolon gewählt. Ist die Darstellung des Kommas als Punkt gängig (amerikanisches Format), wird das Komma als Trennzeichen verwendet.

Ein Beispiel für eine Zeile in einer kommaseparierten Datei sieht wie folgt aus:

01.01.2022 01:00:00;7,5;52;0

Betrachtet man diese Datei in Tabellenform, stellt sich dies wie folgt dar:

01.01.2022	01:00:00	7,5	52	0
------------	----------	-----	----	---

Die Zeile stellt dabei einen Zeitpunkt dar, welcher drei Messwerte enthält. Um diese Messpunkte zu benennen, werden vor dem ersten Messwert weitere Zeilen, der sogenannte Header, mit den Bezeichnungen für die Messpunkte in die Datei geschrieben.

Eine zulässige Darstellung sieht laut der AMEV Richtlinie Nr. 158 in Tabelle 5.1 zu entnehmen.

Tabelle 5.1: Datenpunkte und ihre Werte

Datenpunktadresse	DP001 Außenlufttemperatur °C	DP002 Stellung Ventil %	DP003 Betriebsmeldung WP3
Klartext			-
Einheit			0
Min	-10	0	
Max	50	100	1
01.01.2022 00:00:00	5,3	0	0
01.01.2022 00:15:00	6,5	0	1
01.01.2022 00:30:00	7,2	25	1
01.01.2022 00:45:00	7,3	37	1
01.01.2022 01:00:00	7,5	52	0

Wie zu erkennen ist, enthält der Header neben dem Datenpunktnamen auch weitere Informationen. Diese zusätzlichen Daten werden als Metadaten bezeichnet und stellen zusätzliche Informationen wie die Einheit, die Grenzen des Messwertes dar. Diese Metadaten können noch weitere Informationen, wie zum Beispiel den Einbauort und Ähnliches enthalten.

5.2 Datenbanken

Datenbanken sind spezialisierte Programme zur Speicherung und Verwaltung von Daten. Ein wesentlicher Vorteil von Datenbanken ist die hohe Automatisierbarkeit beim Einfügen und Abrufen von Daten. In den letzten Jahren haben sich verschiedene Datenbanktypen für unterschiedliche Anwendungszwecke etabliert.

Monitoringdaten stellen eine Form von Zeitreihendaten dar, bei denen in regelmäßigen Abständen ein oder mehrere Messwerte erfasst werden. Für diese Art von Daten haben sich verschiedene Datenbanken bewährt. Ein prominentes Beispiel ist die frei verfügbare InfluxDB, die sich besonders durch ihre hohe Geschwindigkeit beim Schreiben und Auslesen von Daten auszeichnet. Dank spezieller Schnittstellen können hier mehrere Tausend Messwerte pro Sekunde verarbeitet werden. Diese Eigenschaften machen InfluxDB besonders sinnvoll für technisches Monitoring, etwa in der Überwachung von Servern, Netzwerken oder Anwendungen, wo zeitnahe Entscheidungen auf Basis von aktuellen Leistungsdaten getroffen werden müssen.

Eine weitere leistungsstarke Option für die Speicherung von Zeitreihendaten ist TimescaleDB. Diese auf PostgreSQL basierende Datenbank wurde speziell für Zeitreihendaten entwickelt und kombiniert die Vorteile relationaler Datenbanken mit einer effizienten Verwaltung großer Datenmengen. TimescaleDB nutzt Hypertables, die eine einfache Handhabung und automatische Partitionierung von Zeitreihendaten ermöglichen. Diese Eigenschaften machen TimescaleDB besonders geeignet für technisches Monitoring, insbesondere in der Überwachung von IT-Infrastrukturen, wo eine umfangreiche historische Analyse und Aggregation von Leistungsdaten erforderlich sind. Die Unterstützung von

SQL und Funktionen zur Aggregation und Kompression von Daten tragen dazu bei, dass Nutzer sowohl schnelle Abfragen als auch eine effiziente Datenspeicherung durchführen können.

MySQL hingegen ist ein weit verbreitetes relationales Datenbankmanagementsystem, das SQL zur Verwaltung und Abfrage von Daten verwendet. Es speichert Daten in Tabellen, die durch Beziehungen miteinander verbunden sind, was komplexe Abfragen und Datenintegrität ermöglicht. Obwohl MySQL nicht speziell für Zeitreihendaten optimiert ist, kann es dennoch im technischen Monitoring eingesetzt werden, beispielsweise zur Speicherung von Log-Daten oder zur Überwachung von Anwendungsmetriken, wo Zeitstempel eine Rolle spielen.

Insgesamt bieten InfluxDB, TimescaleDB und MySQL unterschiedliche Ansätze zur Speicherung und Verwaltung von Daten. Die Wahl der richtigen Datenbank hängt von den spezifischen Anforderungen und Anwendungsfällen ab. Während InfluxDB und TimescaleDB besonders für die effiziente Verarbeitung von Zeitreihendaten im technischen Monitoring geeignet sind, bietet MySQL eine flexible Lösung für eine Vielzahl von Datenbankanforderungen.

Datenbanken sind spezialisierte Programme zur Speicherung und Verwaltung von Daten. Ein wesentlicher Vorteil von Datenbanken ist die hohe Automatisierbarkeit beim Einfügen und Abrufen von Daten. In den letzten Jahren haben sich verschiedene Datenbanktypen für unterschiedliche Anwendungszwecke etabliert. Monitoringdaten stellen eine Form von Zeitreihendaten dar, bei denen in regelmäßigen Abständen ein oder mehrere Messwerte erfasst werden. Für diese Art von Daten haben sich derzeit verschiedene Datenbanken bewährt. Ein bekannter Vertreter ist die frei verfügbare InfluxDB, die sich besonders durch ihre hohe Geschwindigkeit beim Schreiben und Auslesen von Daten auszeichnet. Dank spezieller Schnittstellen können hier mehrere Tausend Messwerte pro Sekunde verarbeitet werden. Zudem ermöglicht die Syntax zur Datenabfrage eine Voraggregierung, etwa durch das Abrufen von Stundenmittelwerten. Diese Berechnungen erfolgen intern während der Anfrage an die Datenbank, was die Auslesegeschwindigkeit erheblich steigert.

5.3 Datenpunktbezeichnungen

Mit steigender Größe eines Datensatzes wird die Verwaltung der Einzeldaten umso wichtiger. Ein wichtiger Punkt bei der Verwaltung ist die Bezeichnung der Datenpunkte. Jeder Datenpunkt muss dabei zwingend eine eindeutige Bezeichnung besitzen, mit der die Messdaten dem Sensor bzw. Messgerät zugeordnet werden können.

Datenpunktbezeichnungen können grundsätzlich frei gewählt werden. Dabei gilt, je mehr Datenpunkte vorhanden sind, desto vorteilhafter sind Metadaten zur Bezeichnung der Datenpunkte. Metadaten können beispielsweise die Messgröße, den Einbauort und Ähnliches beschreiben. Neben der Beschreibung des Datenpunktes ermöglichen Metadaten automatisierbare Auswertungen. Eine Datenanalysesoftware kann durch eine eindeutige Bezeichnung oder Metadaten jede Zeitreihe automatisch erkennen und entsprechende Datenverarbeitungsschritte automatisch durchführen. Beispielsweise kann die Plausibili-

sierung der Messwerte für alle Außentemperaturmessungen in einer Datenbank automatisiert durchgeführt werden, wenn die entsprechenden Zeitreihen von der Software erkannt und gefunden werden können. So lassen sich diese auch bei vielen Messpunkten leicht zuordnen.

Für eine einfache Handhabung der Messdaten beim Inbetriebnahmemonitoring ist eine frühe Festlegung der Datenpunktbezeichnungen bzw. eines Metadatensystems wichtig. Bei großen und wachsenden Projekten sollte daher zu Beginn eine konsistente und eindeutige Datenpunktbezeichnung gewählt werden. Es ist sinnvoll, die Datenpunkte in einer Liste, der sogenannten Messpunktliste, zu speichern.

Für Datenpunktbezeichnungen gibt es viele Konzepte wie die GA-Funktionsliste der DIN EN ISO 16484-3, die VDI 3841, nach AMEV oder das BUDO-Schema. Datenpunktbezeichnungen werden auch Anlagenkennschlüssel (AKS), Datenpunktschlüssel oder Datenpunktadressierungsschlüssel genannt, da sie aus zusammengesetzten Abkürzungen, sogenannten Schlüsseln, bestehen.

Als ein Beispiel für ein solches Schema zur Bezeichnung der Datenpunkte ist das Building Unified Data point naming schema for operation Management (BUDO). BUDO ist ein Open Source Schema, welches in einer Zusammenarbeit von verschiedenen Instituten entwickelt wurde. Es ist auf der folgenden Seite erhältlich: <https://github.com/RWTH-EBC/BUDO>

Das BUDO Schema besteht aus zwei Komponenten, dem Bezeichnungsschema und dem dazugehörigen Vokabelbuch. In dem Vokabelbuch sind nahezu alle gebäudetechnischen Anlagen und Objekte, die für ein Gebäudemonitoring benötigt werden, aufgelistet und es wird ein Abkürzungsschlüssel benannt. So wird zum Beispiel ein Brennwertkessel mit der Abkürzung „COND“ (engl. condensing boiler) und eine Wärmepumpe als „HP“ (engl. Heat pump) bezeichnet.

Das Bezeichnungsschema ist in Abbildung 5.1 dargestellt.

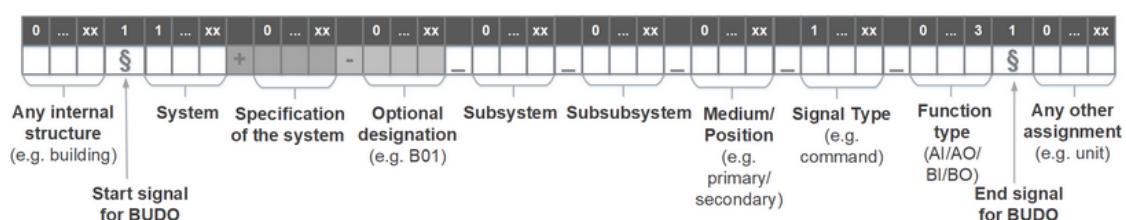


Abbildung 5.1: Struktur des BUDO-Schemas [?]

Wie in Abbildung 5.1 zu sehen, besteht das Bezeichnungsschema aus einer festgelegten Reihenfolge an Informationen. Die Informationen sind durch festgelegte Zeichen getrennt und somit eindeutig zuordbar. Weiterhin ermöglicht es diese Festlegung auch, das maschinelle Lesen und Schreiben des Schlüssels. Weitere Informationen über das BUDO Schema sind auf der oben genannten GitHub Seite zu entnehmen.

5.4 Metadatensystem

In Abschnitt 5.3 wurde die Datenpunktbezeichnung als Benennung von Messpunkten eingeführt. Dieses System hat bei größeren Projekten den Nachteil, dass die Bezeichner oft unnötig lang werden. Damit wird es auch unübersichtlich. Daher gibt es mit dem Metadatensystem eine weitere Möglichkeit der Datenpunktbezeichnung.

Beim Metadatensystem werden die Metadaten nicht direkt im Messpunktnamen gespeichert. Während also die Datenpunktbezeichnung aus aneinander gereihten Zeichen besteht, die Informationen über den Datenpunkt enthalten, wird dies beim Metadatensystem entkoppelt. Das Metadatensystem ist also deutlich flexibler. Dies ist darauf zurückzuführen, dass hier die Informationen nicht starr in den Messpunktschlüssel eingeschrieben sind. Stattdessen werden die Metadaten als Zusatzinformation in der Datenbank oder einer parallelen Datenbank gespeichert und mit der Datenreihe der Messdaten verknüpft.

Es gibt unterschiedliche Ansätze von Metadatensystemen:

- **Tabellarisch:** (bspw. mondas Metadatenstruktur)
- **Graphen mit Tags:** (bspw. Haystack, das auf Haystack basierende Brick Schema oder PLCont)

Graphen können gegenüber Tabellen einen höheren Informationsgehalt durch die Darstellung von Zusammenhängen besitzen. Der Aufwand bei der Pflege und dem Erstellen der Metadaten ist bei Graphen jedoch höher. Bisherige Ansätze, Metadaten automatisiert bspw. mit machine learning zu generieren, waren wenig erfolgreich. Bei guter maschinenlesbarer Informationsgrundlage, wie einheitlichen Datenpunktbezeichnern mit Beschreibungen oder durch BIM-Modelle, lassen sich viele Schritte bei der Erstellung der Metadaten durch Automatisierung erleichtern.

Ein Metadatensystem sollte die Identifizierung einzelner Sensoren ermöglichen. Die Unabhängigkeit von der Datenpunktbezeichnung ermöglicht ein standardisiertes „Finden“ von Sensoren bei unterschiedlichen Datenquellen. Werden beispielsweise Liegenschaften mit unterschiedlichen Datenpunktbezeichnungen überwacht, ist ein einheitliches Metadatensystem sehr hilfreich zum Identifizieren der einzelnen Sensoren. Durch ein vereinheitlichtes Metadatensystem lassen sich außerdem Auswertungen und Darstellungen automatisieren. So können beispielsweise leicht alle Vorlauftemperaturen von Heizkreisen gefunden und überwacht werden. Ein einheitliches, von der Datenpunktbezeichnung unabhängiges Metadatensystem ist somit für modernes Monitoring unabdingbar.

In Abbildung 5.2 wird beispielhaft die mondас Metadatenstruktur gezeigt.

Die mondас Metadatenstruktur entspringt aus dem Kontext des Fraunhofer-ISE. Das Vokabular der Metadatenstruktur ist daher dem des BUDO-Schemas sehr ähnlich, das von der RWTH Aachen entwickelt wurde und zum Teil auf dem Fraunhofer Verfahren basiert. Die Metadatenstruktur findet in der Praxis bereits Anwendung bei einer Vielzahl von Anlagentypen, wie PV-Anlagen, Fernwärmenetzen, TGA, Produktionsbetrieben. Es ist tabellarisch aufgebaut. Zusammengesetzte Komponenten, ähnlich eines AKS, sind aus einzelnen Tabellenspalten möglich. Verfügbar ist ein erweiterbarer Satz

The screenshot shows the 'Metadata' section of the mondus administration interface. At the top, there are tabs for 'Raw' (selected), 'Virtual', and 'Selector'. Below the tabs are buttons for 'Import', 'Export', 'Actions', 'Clear search', 'Reset', 'Save', and a search bar with placeholder 'Show sensors (315)'. A toolbar on the right includes icons for file operations like save, delete, and copy.

The main area contains a search form with two AND clauses:

- DP equals MEA.T
- POS equals SUP

Below the search form is a table header with columns: DP, SYS, SUBSYS1, SUBSYS1_EXT, Threshold, Unit, and Description. The table lists four sensor entries:

DP	SYS	SUBSYS1	SUBSYS1_EXT	Threshold	Unit	Description	
AT2 (E101, E103)-S...	MEA.T	SCOL	WC.H	SOLAR	40	°C	Solarkreis WMZ VL-Temp.
AT2 (E101, E103)-K...	MEA.T	SCOL	WC.H	COLLECTOR	40	°C	Kollektorkreis VL-Temp.
AT2 (E101, E103)-H...	MEA.T	DHW	WC.H		50	°C	Heizkreis WMZ VL-Temp.
AT2 (E101, E103)-W...	MEA.T	DHW	WC.H		60	°C	WWB WMZ VL-Temp.

At the bottom of the table, there is a page navigation bar showing 'Showing 100 of 315 sensors' and a page number '1'.

Abbildung 5.2: Anwendungsbeispiel der mondus Metadatenstruktur

an Spaltennamen wie „SYS“, sogenannten Metadefinitionen und das dazugehörige Vokabular. Durch die Flexibilität der Metadefinitionen sind viele Zusatzinformationen zu den Sensoren speicherbar. Es gibt jedoch eine fest definierte Basis an Metadefinitionen und Vokabeln. Auch sind die Vokabeln wie DHW für Trinkwasser den Basis Metadefinitionen zugeordnet. Erfüllbar ist somit der Anspruch an automatisierte Auswertungen durch standardisierte Metadaten. Bei gleichzeitiger Flexibilität, um auf individuelle Anlageninformationen zu verknüpfen.

In Abbildung 5.2 ist beispielhaft eine Filtermöglichkeit der Metadaten zu sehen. Es werden alle Vorlauftemperaturen von den in der Plattform verfügbaren Sensoren gefunden. Außerdem ist ein individueller „Threshold“ festgelegt, welcher das automatische Überwachen der jeweiligen Vorlauftemperatur ermöglicht.

6 Redaktion

Herausgeber: Technische Hochschule Rosenheim, Technical University of Applied Sciences,
Hochschulstraße 1, 83024 Rosenheim, Telefon +49 8031 805-0, Fax +49 8031 805-2105,
www.th-rosenheim.de

Verantwortlich i.S.d.P.: Prof. Heinrich Köster, Präsident

Redaktion: Claudia Neuner, Studienberatung

Literaturverzeichnis

- [1] VDI-Richtlinie 3789. Umweltmeteorologie - wechselwirkungen zwischen atmosphäre und oberflächen - berechnung der spektralen kurz- und der langwelligen strahlung, 2019.
- [2] VDI-Richtlinie 6041. Facility-management - technisches monitoring von gebäuden und gebäudetechnischen anlagen, 2017.

Gefördert durch:



Bundesministerium
für Wirtschaft
und Klimaschutz

aufgrund eines Beschlusses
des Deutschen Bundestages

Technische
Hochschule
Rosenheim

 **MonSec**

The MonSec logo features a stylized 'M' composed of four horizontal bars in red, green, blue, and orange, followed by the word 'MonSec' in a bold, sans-serif font.

Datenaufnahme – Monitoringschauwand

IoT-Messtechnik – Raspberry Pi

Anleitung

Technische Hochschule Rosenheim

Bearbeiter: Markus Hartmann

27.06.2022

Inhaltsverzeichnis

Inhaltsverzeichnis	1
Abbildungsverzeichnis	2
1 Einleitung	3
2 Konfiguration der Datenabfrage	4
2.1 Konfiguration des Mbus Connectors	4
2.2 Konfiguration der Hilfsfunktion	6
2.3 Konfiguration der Parser Funktionen.....	7
2.4 Konfiguration des InfluxDB Connectors	8
3 Konfigurierung der Visualisierung	10

Abbildungsverzeichnis

Abbildung 1: Darstellung der Monitoringschauwand	3
Abbildung 2: Übersicht des NodeRed Flows	4
Abbildung 3: Konfiguration Mbus Connector	5
Abbildung 4. Code der Hilfsfunktion	7
Abbildung 5: Code einer Parsing Funktion	7
Abbildung 6: Konfiguration des InfluxDb Connectors	9
Abbildung 7: Konfiguration der Abfragequery	10

1 Einleitung

In dieser Dokumentation wird der Aufbau der Sensorabfrage, Datenspeicherung und Visualisierung der Monitoringschauwand am Rosenheimer Technologiezentrum für Energie und Gebäude (roteg) beschrieben.

Bei der Monitoringschauwand handelt es sich um einen simulierten Heizkreis mit verschiedenen, in der Gebäudetechnik üblichen, Sensoren und Messgeräten.

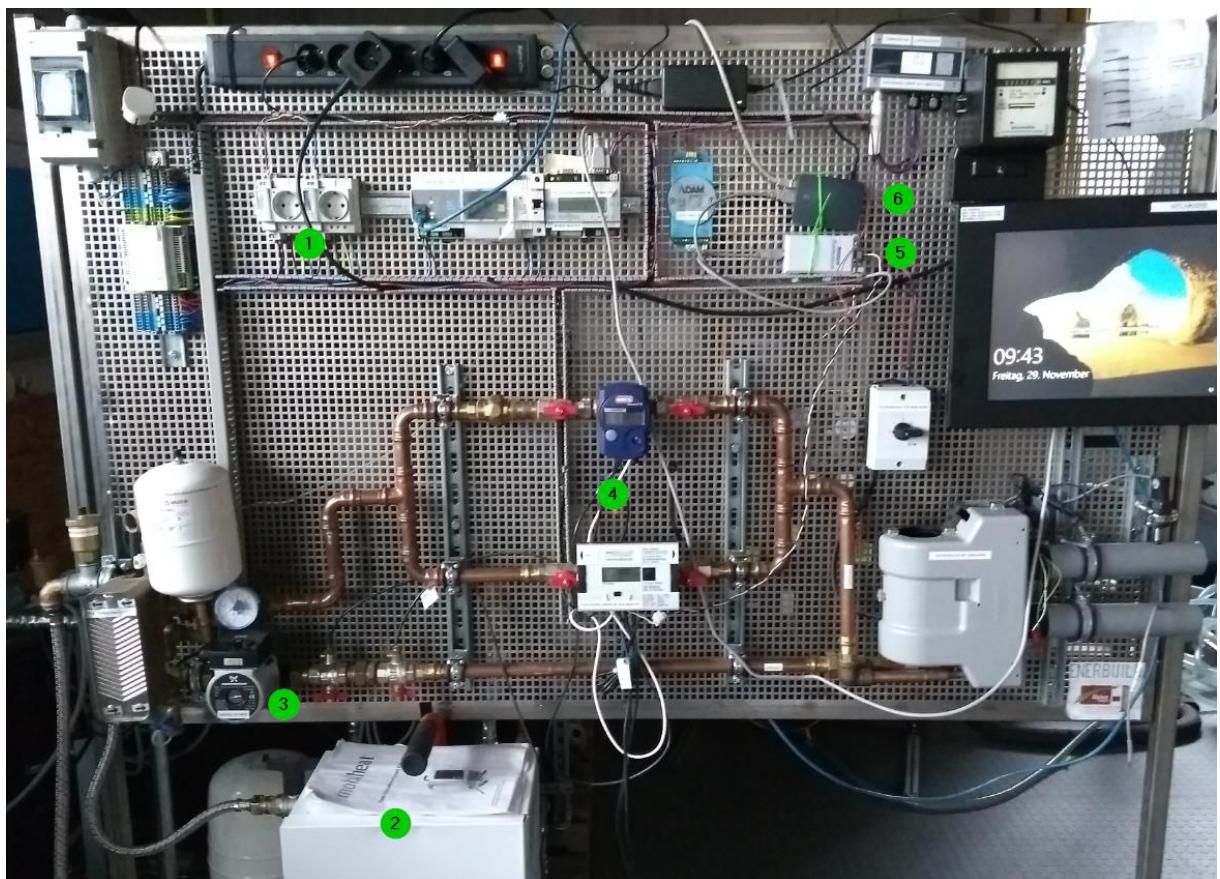


Abbildung 1: Darstellung der Monitoringschauwand

In dieser Doku wird vor allem auf die Datenabfrage der MBus Zähler eingegangen. Diese werden mit dem IoT Device Raspberry Pi (6) (per NodeRed) ausgelesen und in eine lokale Datenbank (InfluxDB) geschrieben. Eine Visualisierungsumgebung für die Daten ist in Form des Programms Grafana ebenfalls auf dem RaPi installiert.

Bei den hier relevanten Sensoren handelt es sich um zwei Energiemessgeräte (1) an welchen die elektrische Heizung (2) und die Heizkreispumpe (3) separat gemessen werden. In dem Heizkreis sind parallel zwei Wärmemengenzähler (4) verbaut.

2 Konfiguration der Datenabfrage

Die Datenabfrage erfolgt über einen USB- Pegelwandler, der die MBus Zähler direkt mit dem Raspberry Pi verbindet.

Die Datenabfrage erfolgt über NodeRed. NodeRed wird unter Angabe der IP des Raspberry Pimit dem Port 1880 im lokalen Netzwerk über den Browser erreicht.

Hierfür werden die folgenden „Paletten“ benötigt. Die Paletten sind die Bibliotheken in NodeRed, welche den Standardumfang um zusätzliche Programmierblöcke erweitern.

- Mbus Abfrage
<https://flows.nodered.org/node/node-red-contrib-m-bus>
- Speichern in der InfluxDB
<https://flows.nodered.org/node/node-red-contrib-influxdb>

Der grundsätzliche Flow für die Datenabfrage sieht wie folgt aus:

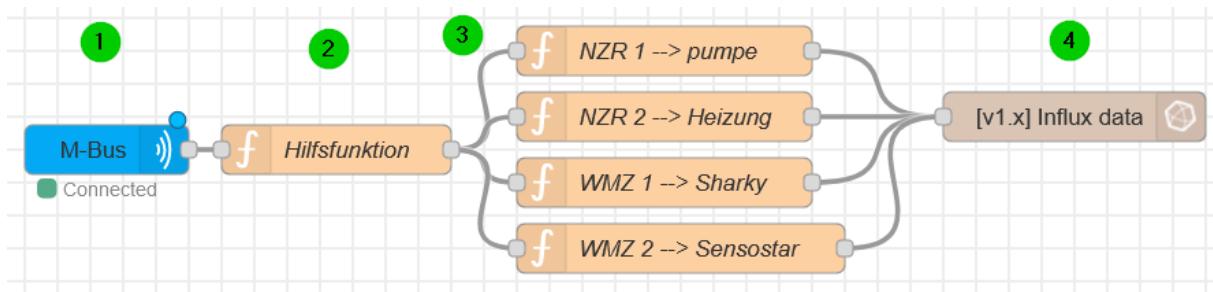


Abbildung 2: Übersicht des NodeRed Flows

2.1 Konfiguration des Mbus Connectors

Für das erstmalige einrichten wird der Beispiel Flow der contrib-m-bus Palette benötigt, um das Busnetz nach verfügbaren Geräten zu scannen und die Geräte-Ids herauszulesen. Danach werden die Geräte Zyklisch durch den Connector abgefragt. Zur Konfiguration des Connectors werden folgende Einstellungen gewählt.

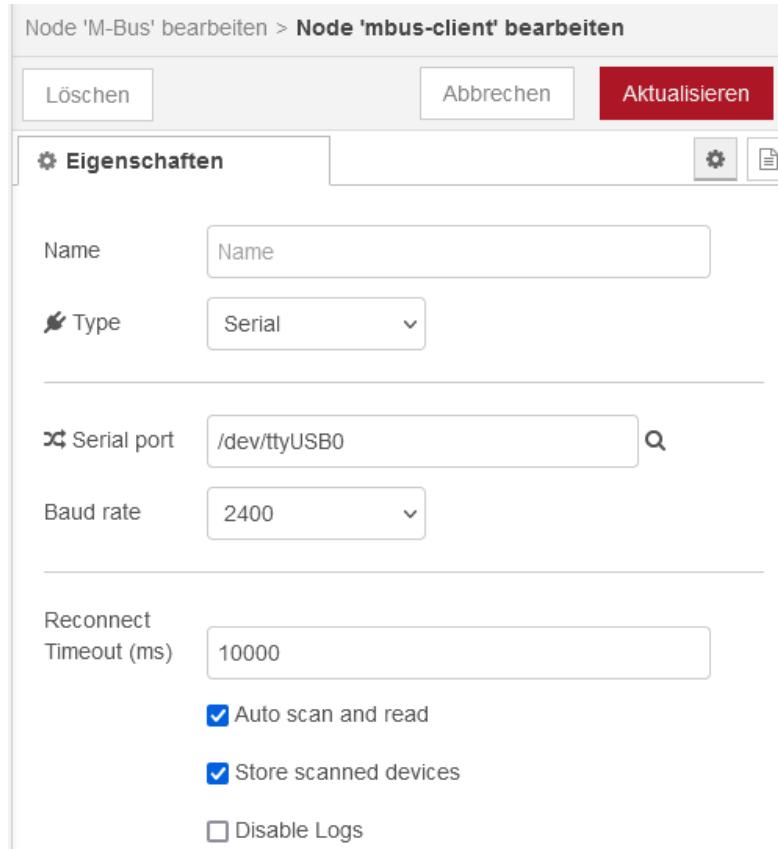


Abbildung 3: Konfiguration Mbus Connector

Für den Serial Port wird der USB-Port des RaPi angegeben.

Die Baud rate ist abhängig von Einstellungen an den Geräten und dem Pegelwandler.

Die Reconnect Timeout Einstellung ist auch gleichzeitig der Zyklus für die Abfrage der Geräte und wurde für die Demonstration der Schauwand auf 10 Sekunden eingestellt. Das Automatisierte Auslesen wird durch die Option „Auto scan and read“ eingeschaltet. Die Option „Store scanned decices“ erlaubt es den Connector ohne scannen des gesamten MBus zu verwenden.

Es gilt zu beachten, dass beim Mbus jedes Gerät separat abgefragt werden muss. Von daher ist auch das gleichzeitige Auslesen nicht möglich. Daher sollte für die Abfrage genügend Zeit (>2 Sekunden) eingeplant werden.

Wird ein Gerät abgefragt, so entsteht eine NodeRed Message im JSON (JavaScript Object Notation) Format ausgegeben. Ein Beispiel für eine solche Nachricht sieht, in stark gekürzter Form wie folgt aus:

```
"topic": "mbDeviceUpdated",
"payload": {
  "SlaveInformation": {
    "Id": 30102556,
  },
  "DataRecord": [
    {
      "Unit": "Energy (Wh)",
      "Value": 3455620,
    },
    {
      "Unit": "1e-1 V",
      "Value": 2334,
    },
    {
      "Unit": "1e-1 A",
      "Value": 0,
    },
    {
      "Unit": "Power (W)",
      "Value": 34,
    }],
  }
}
```

Die Nachricht erhält weitaus mehr Informationen, welche aber für den weiteren Gebrauch nicht relevant sind und aus Gründen der Übersichtlichkeit entfernt wurden.

Die verwendeten Bestandteile der Nachricht sind wie folgt:

- Payload.SlaveInformation.Id → Die Sekundäradresse des Messgerätes
- Payload.DataRecord[].Unit → Die Einheit des Messpunktes
- Payload.DataRecord[].Value → Der Wert des Messpunktes

2.2 Konfiguration der Hilfsfunktion

In der Hilfsfunktion werden die Daten vorsortiert, um den Code der Parser Funktionen zu reduzieren. Eine IF Abfrage stellt noch sicher, dass nur Daten aus aktualisierten Sensordaten weiterverarbeitet werden. Der Code für die Funktion sieht wie folgt aus:

```

1 // Filtern der Nachrichten nach "Device Updated"
2 // Umsortieren der payload für leichteres Erreichen
3 if (msg.topic == "mbDeviceUpdated"){
4 msg.Device = msg.payload.SlaveInformation.Id;
5 msg.payload = msg.payload.DataRecord;
6 return msg;
7 };

```

Abbildung 4. Code der Hilfsfunktion

Es werden die ID des Gerätes sowie die Messdaten extrahiert und weitergegeben.

2.3 Konfiguration der Parser Funktionen

Die Parser Funktionen haben die Aufgabe, die Messdaten aus den Speicherregistern zu entnehmen. Für das Einspeichern der Daten in die InfluxDB müssen die Messdaten noch in eine entsprechende Form gebracht werden.

Grundsätzlich gibt es beim Parsen von Bus Daten in NodeRed mehrere Möglichkeiten. In diesem Fall wird für jedes Gerät eine eigenständige Funktion verwendet, welche die Nachrichten nach dem jeweiligen Gerät filtert. Ebenfalls kann die Funktionalität durch einen komplexeren Code vereinfacht und stark automatisiert werden. Im Sinne der Aufgabe einen Einstieg in die IoT-Messdatenaufnahme zu zeigen wurden die Funktionen bewusst einfach und rudimentär gehalten.

In der Folgenden Abbildung ist die Parser Funktion für einen der beiden Strommessgeräte (entspricht dem JSON Objekt aus Kapitel 2.1) dargestellt.

```

1 if (msg.Device == 30102556){
2   msg.payload = [
3     Energy: msg.payload[0].Value,
4     Power: msg.payload[4].Value,
5     Voltage: msg.payload[2].Value/10,
6     Amps: msg.payload[3].Value/10,
7   ],
8   [
9     Device: "Heizung"
10   ];
11 }
12 }

```

Abbildung 5: Code einer Parsing Funktion

Die Nachricht wird durch eine IF Funktion anhand der sekundären Geräte Id gefiltert.

Die Messdaten des Gerätes werden anhand des Datenblattes, aus welchem die Registertabelle sowie die Einheiten und Skalierungsfaktoren entnommen werden, umgeformt.

So steht im Register Nr. Null der Wert für den Energiezähler mit einem Skalierungsfaktor von Eins. Dieser wird in ein Array der Message Payload mit der Variablen „Energy“ verschoben. Die restlichen Werte werden entsprechend umgewandelt, wobei bei den Werten für Volt und Ampere ein Skalierungsfaktor von $1 \cdot 10^{-1}$ verwendet werden muss.

Einschub:

Alle verwendeten Geräte liefern die Messwerte im Datenformat Integer. Somit ist keine Umwandlung des Datentyps nötig. Andere Geräte können Daten in Form von z.B. Float Zahlen liefern. Für diesen Fall muss der Messwert noch umgewandelt werden.

Das Einspeichern in die Influx erlaubt es, dem Messwerten noch Metadaten (Tag) mitzuliefern, welche im zweiten Array der Payload enthalten sein muss. So wird hier noch der Gerätetyp in Form des Tags „Device“ als String mitgeliefert. Dies erlaubt ein einfaches Filtern der Datenpunkte in der Grafana Visualisierungssoftware (siehe Kapitel 3)

2.4 Konfiguration des InfluxDB Connectors

Für das Einspeichern der Daten in die InfluxDB Datenbank wird der Connector wie folgt konfiguriert

Node 'influxdb out' bearbeiten > **Node 'influxdb' bearbeiten**

Löschen Abbrechen Aktualisieren

Eigenschaften

Name: Influx

Version: 1.x

Host: 127.0.0.1 **Port:** 8086

Database: [REDACTED]

Benutzername: [REDACTED]

Passwort: [REDACTED]

Enable secure (SSL/TLS) connection

Abbildung 6: Konfiguration des InfluxDb Connectors

Da die Datenbank auf dem RaPi selbst läuft wird für die IP des Hosts die lokale IP 127.0.0.1 angegeben, der Port 8086 ist der Standard Influx Port.

Für die Database wird der Name einer erstellten Datenbank benötigt.

Es empfiehlt sich zur Absicherung der Datenbank einen Benutzer mit Passwort einzurichten.

Die Daten werden nach Vorgaben des Lineprotocols der InfluxDB entsprechend im JSON Format dem Koton zur Verfügung gestellt.

3 Konfigurierung der Visualisierung

Zur Visualisierung wird die Software Grafana verwendet, welche ebenfalls lokal auf dem RaPi läuft. Grafana wird unter Angabe der IP mit dem Port 3000 im lokalen Netzwerk über den Browser erreicht.

Für die Erstinstallation wird die Einstellung der Datenbank benötigt. Hierfür wird die IP der Datenbank (localhost) der Datenbankname sowie Benutzer und Passwort benötigt.

Zur Konfiguration der Abfragen aus der Datenbank (Querys) enthält Grafana einen einfachen Editor über welchen die Querys einfach konfiguriert werden können. In der folgenden Abbildung ist beispielhaft für einen Datenpunkt die Eingaben der Query dargestellt.

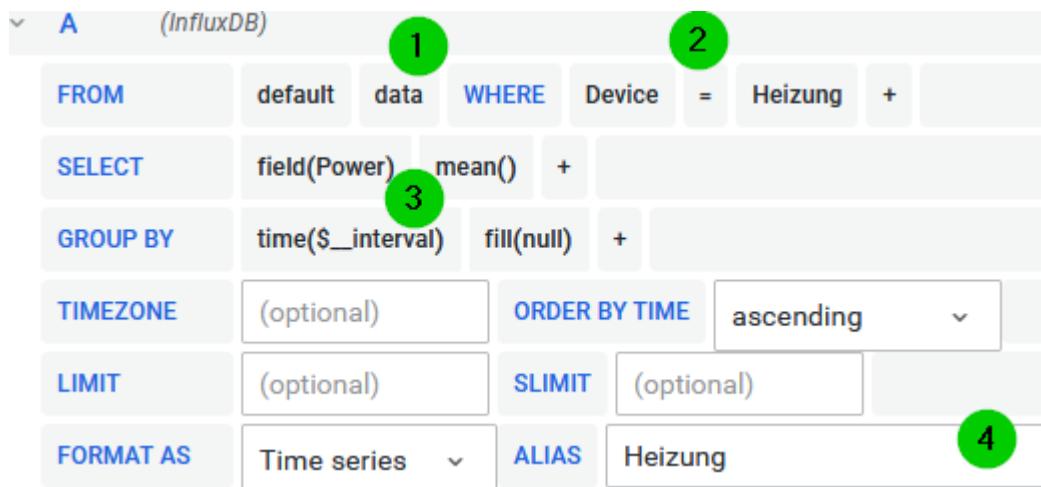


Abbildung 7: Konfiguration der Abfragequery

Es wird der Messpunkt „Leistung“ des Stromzählers der Heizung abgefragt. Hierzu sind folgende Eingaben nötig:

FROM:

Eingabe der Datenbank (default) und des Measurements (1) der Datenpunkte Durch den zusätzlichen Tag (2), „Device“ werden die Datenpunkte nach dem Messgerät gefiltert.

SELECT:

Als field (3) wird der Name des Messpunktes eingegeben. Es wird eine Aggregierungsweise Selektor Funktion benötigt.

GROUP By:

Enthält zusätzlich Funktionen zur Auswahl des Zeitraums. Durch den Standard Parameter erfolgt die Zeiteinstellung über das Dashboard in welchem die Visualisierung enthalten ist.

ALIAS:

Durch Eingabe des Alias wird der Legendenname eingegeben.

Gefördert durch:



Bundesministerium
für Wirtschaft
und Klimaschutz

aufgrund eines Beschlusses
des Deutschen Bundestages

Technische
Hochschule
Rosenheim



 **MonSec**

Datenabfrage Fronius Wechselrichter mit IoT-Technik

Im Projekt MonSEC- Monitoring Secure

TH Rosenheim

Bearbeiter: Markus Hartmann

Stand: 03.08.2022

Inhaltsverzeichnis

Inhaltsverzeichnis	1
Abbildungsverzeichnis	2
1 Einleitung	3
2 Vorbereitung Fronius Wechselrichter	4
3 Installation und Einrichtung des RaspberryPi	5
3.1 Einrichtung von Influx	5
3.2 Einrichten von NodeRed	5
3.3 Einrichtung von Grafana.....	8
4 Abfrage Smartmeter über Wechselrichter	10

Abbildungsverzeichnis

Abbildung 1: Konfiguration der Datenausgabe des Fronius Wechselrichters	4
Abbildung 2: Übersicht der Programmierung in NodeRed.....	5
Abbildung 3: Eingabe der Registerabfrage.....	6
Abbildung 4: Erstellung der Modbus Abfrage	6
Abbildung 5: Konfiguration des Modbus Flex Getter	6
Abbildung 6: Parsing Funktion 1	7
Abbildung 7: Parsing Funktion 2	7
Abbildung 8: Einstellungen zum Schreiben in Datenbank	7
Abbildung 9: Grafana Einstellungen für Verlaufsgraphen	8
Abbildung 10: Einstellung für Zahlenwerte	9

1 Einleitung

In dieser Anleitung wird beschrieben wie man einen Fronius Wechselrichter über Modbus TCP ausliest. Die Daten werden dabei in einer Influx Datenbank gespeichert und über Grafana visualisiert. Zum Programmieren der Abfrage und Speichern der Daten in der Datenbank wird die Programmierumgebung NodeRed verwendet.

2 Vorbereitung Fronius Wechselrichter

Damit der Fronius Wechselrichter die Daten auf der Schnittstelle zur Verfügung stellt, ist diese im Konfigurationsmenü einzustellen.

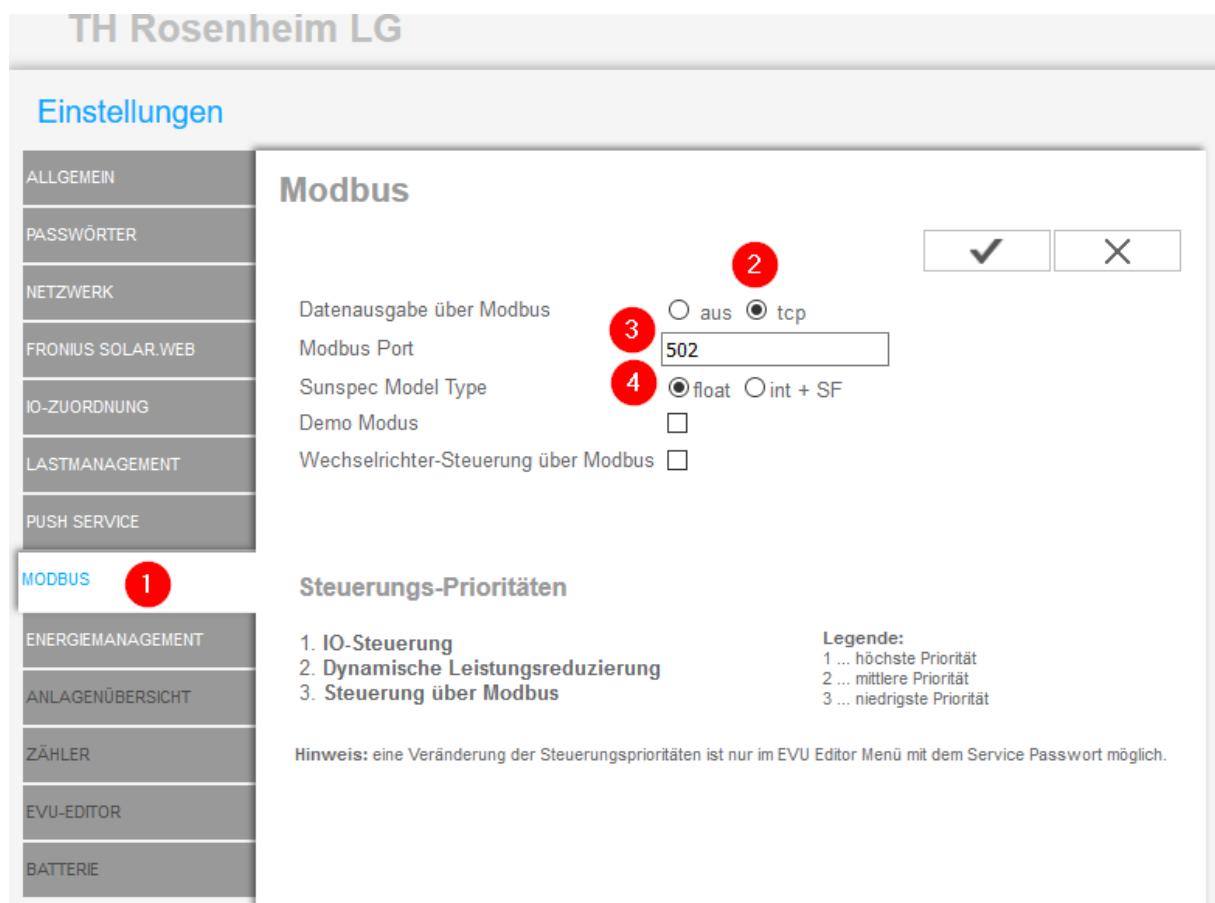


Abbildung 1: Konfiguration der Datenausgabe des Fronius Wechselrichters

Im Einstellungsfenster des Wechselrichters auf den Reiter „MODBUS“ (1) wird die Datenausgabe aktiviert (2). Der Modbus Port (3) ist default auf 502 gesetzt, ebenso die ausgegebenen Werte als float (4).

3 Installation und Einrichtung des RaspberryPi

Es werden die Programme NodeRed, Influx sowie Grafana benötigt. Die Installation der Programme selbst erfolgt nach bekanntem Vorgang.

3.1 Einrichtung von Influx

Bei Influx muss am wenigsten Aufwand betrieben werden. Neben einem Nutzer wird nur die Datenbank selbst benötigt.

3.2 Einrichten von NodeRed

Für die Programmierung der Datenabfrage und Speicherung der Daten in der InfluxDB werden die folgenden Paletten (Zusatzbibliotheken mit Programmbausteinen) benötigt:

- Node-red-contrib-influxdb
- Node-red-contrib-modbus

Die Paletten können in NodeRed direkt gesucht und heruntergeladen werden.

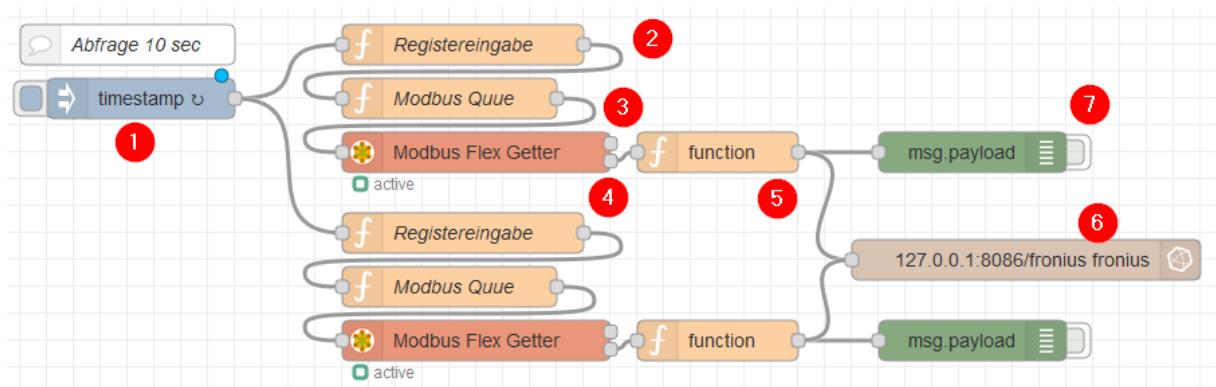


Abbildung 2: Übersicht der Programmierung in NodeRed

In Abbildung 2 ist der Programmcode als Übersicht dargestellt. Da die benötigten Daten auf dem Fronius in weit auseinander liegenden Registern bereit gestellt wird, werden 2 Anfragen parallel gestellt (oben / unten).

Der Reihe nach sind die Blöcke wie folgt:

1. Initialisierung und erzeugen eines zyklischen Startsignals (Inject)

Hier wird das Intervall der Abfrage eingegeben.

2. Funktion zur Eingabe der abzufragenden Modbus Register

```

1 msg.register = 40071;
2 msg.quantity = 38;
3 return msg;

```

Abbildung 3: Eingabe der Registerabfrage

In Abbildung 3 ist der Programmcode zur Einstellung der Registerabfrage zu erkennen. Es werden 38 Register ab der Registernummer 40072 abgefragt (Wichtig, Abfrage -1). Die zweite Abfrage enthält 48 Register ab der Nummer 40265.

3. Erstellung der Abfrage für den „Modbus Flex Getter“

```

1 msg.payload = {
2   value: msg.payload,
3   'fc': 3,
4   'unitid': 1,
5   'address': msg.register ,
6   'quantity': msg.quantity } ;
7 return msg

```

Abbildung 4: Erstellung der Modbus Abfrage

Der Code ist für beide Abfragen identisch.

4. Funktionsblock der Modbus Abfrage (Modbus-Flex-Getter)

In diesem Block wird die IP und der Port des Fronius Wechselrichters eingegeben.

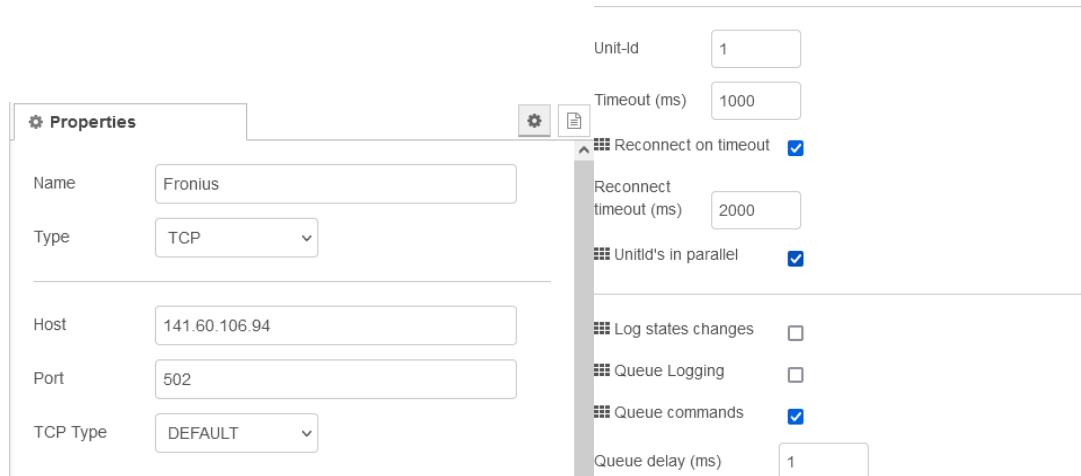


Abbildung 5: Konfiguration des Modbus Flex Getter

5. Parsing Funktion

```

1 msg.payload = {
2     P_PV_AC : msg.payload.buffer.readFloatBE(40,41,42,43),
3     P_PV_DC : msg.payload.buffer.readFloatBE(72,73,74,75),
4     E_PV_AC : msg.payload.buffer.readFloatBE(60,61,62,63)/1000,
5 }
6 return msg;

```

Abbildung 6: Parsing Funktion 1

```

1 var Scale_Factor_I = msg.payload.buffer.readInt16BE(0,1);
2 var Scale_Factor_U = msg.payload.buffer.readInt16BE(2,3);
3 var Scale_Factor_P = msg.payload.buffer.readInt16BE(4,5);
4 var Scale_Factor_E = msg.payload.buffer.readInt16BE(6,7);
5 msg.payload =
6 {
7     E_PV_DC_String_1 : msg.payload.buffer.readUInt32BE(40,41,42,43)*Math.pow(10.0, Scale_Factor_E)/1000,
8     E_PV_DC_String_2 : msg.payload.buffer.readUInt32BE(80,81,82,83)*Math.pow(10.0, Scale_Factor_E)/1000,
9     I_PV_DC_String_1 : msg.payload.buffer.readUInt16BE(34,35)*Math.pow(10.0, Scale_Factor_I),
10    I_PV_DC_String_2 : msg.payload.buffer.readUInt16BE(74,75)*Math.pow(10.0, Scale_Factor_I),
11 }
12 return msg;

```

Abbildung 7: Parsing Funktion 2

Hier werden die empfangenen Daten (in Hex) durch Parsingfunktionen in die richtige Form gebracht und den Variablen zugeordnet.

6. Schreiben in Datenbank

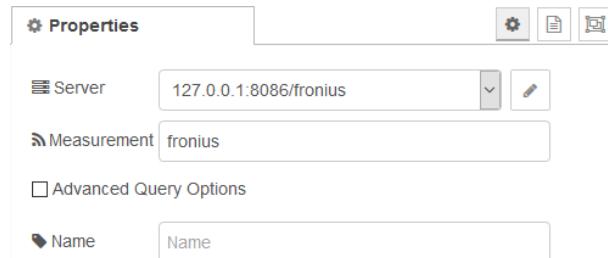


Abbildung 8: Einstellungen zum Schreiben in Datenbank

Es werden die Datenbank Informationen – IP / Datenbankname sowie eine Bezeichnung der Messreihe benötigt.

7. Debugging

Diente zur Ansicht der Modbus Antworten zur Entwicklung des Codes.

3.3 Einrichtung von Grafana

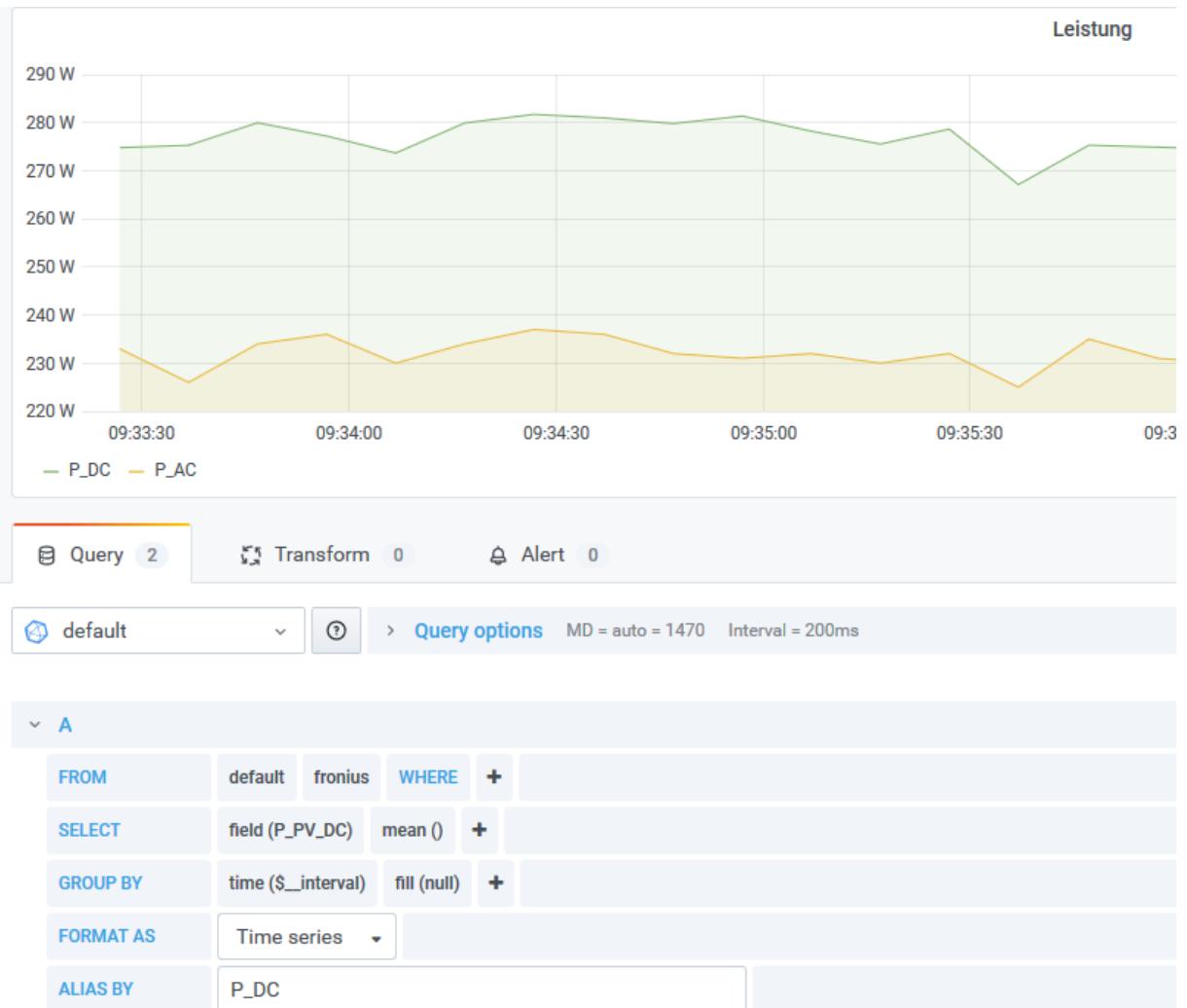


Abbildung 9: Grafana Einstellungen für Verlaufsgraphen

Verlaufsgraphen werden in Grafana wie in Abbildung 9 dargestellt abgefragt.

Die Query ist dabei abhängig, wie die Daten in Influx vorliegen.

Der untere Bereich in der Abbildung ist der s.g. Query Editor mit dem Anfragen zusammengestellt werden können.

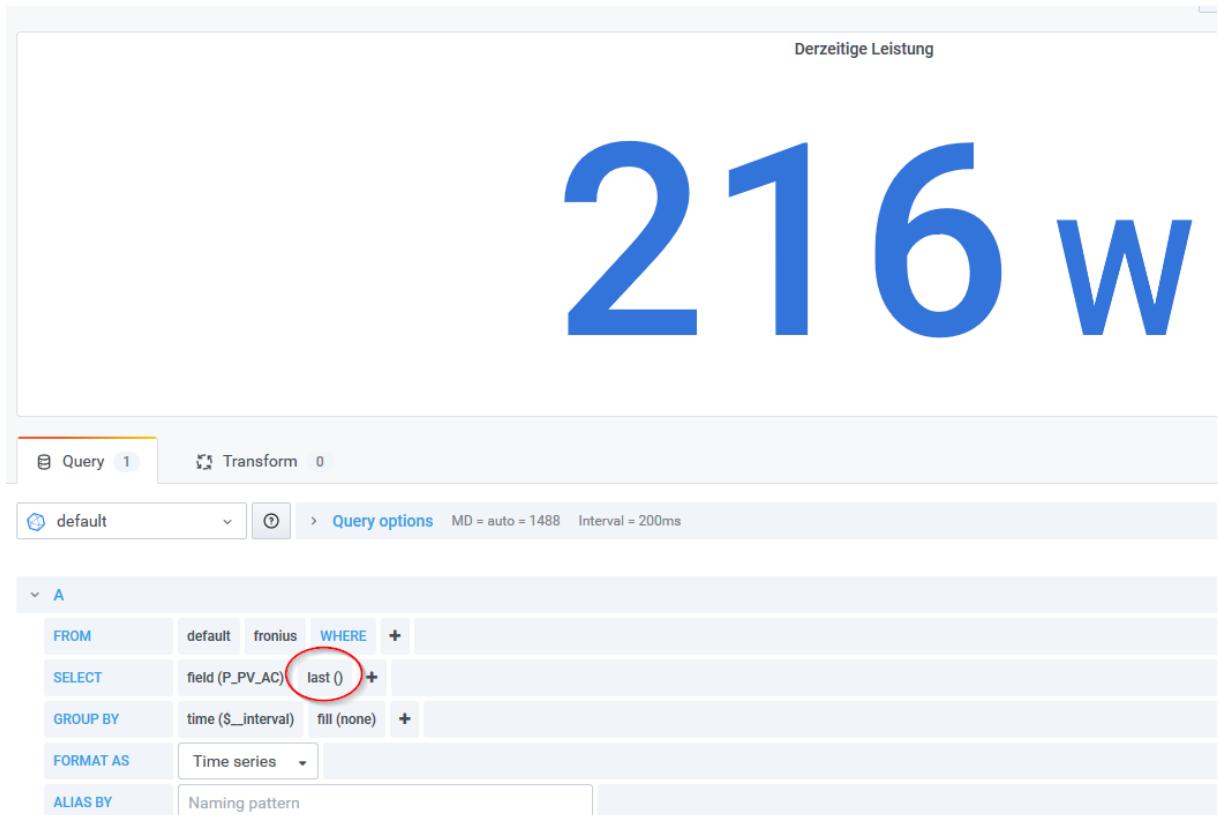


Abbildung 10: Einstellung für Zahlenwerte

Zur Darstellung von Zahlenwerten wird die Visualisierung von Graph auf Stat geändert. Die Abfrage aus der Datenbank zieht dabei den letzten Wert.

Da sich jedoch diese Abfrage auf das angezeigte Intervall bezieht, wird also nur der letzte Wert aus dem Intervall angezeigt. Damit auch tatsächlich der derzeit aktuelle Wert angezeigt wird, muss die Abfrage aus der Datenbank mit einem festen Zeitintervall eingestellt werden. Dazu muss aber die Abfrage in Textform eingegeben werden. Die obige Abfrage sieht in Textform wie folgt aus:

```
SELECT last("P_PV_AC") FROM "fronius" WHERE $timeFilter GROUP BY time($interval)
```

Dies wird wie folgt geändert:

```
SELECT last("P_PV_AC") FROM "fronius" WHERE time > now() -2m and time < now()
```

Grafana fragt also immer den letzten Wert der vergangenen 2 Minuten ab.

4 Abfrage Smartmeter über Wechselrichter

Beim Fronius werden die Smartmeter über Modbus RTU an den Datamanager angebunden.

Es muss die GerätetID des Smartmeters ermittelt werden (sollte 240) sein, konnte bisher noch nicht getestet werden.

Mit der unteren Abfrage kann man die Device IDs herauslesen.

http://141.60.106.94/solar_api/v1/GetActiveDeviceInfo.cgi?DeviceClass=System

Gefördert durch:



Bundesministerium
für Wirtschaft
und Klimaschutz

aufgrund eines Beschlusses
des Deutschen Bundestages

Technische
Hochschule
Rosenheim



 **MonSec**

Datenabfrage eines MBus Wärmemengenzähler mit IoT-Technik

Im Projekt MonSEC- Monitoring Secure

TH Rosenheim

Bearbeiter: Markus Hartmann

Stand: 27.10.2022

Inhaltsverzeichnis

Inhaltsverzeichnis	1
Abbildungsverzeichnis	2
1 Einleitung	3
2 Hardware	4
2.1 Aufbau des Messkasten	4
2.2 Einrichten der Seriellen Schnittstelle.....	4
3 Konfiguration des Raspberry Pi	6
3.1 Konfiguration der Datenabfrage in Node-Red.....	6
3.1.1 Konfiguration des Mbus Connectors	6
3.1.2 Konfiguration der Hilfsfunktion	8
3.2 Konfiguration der Parser Funktionen.....	9
3.2.1 Konfiguration des InfluxDB Connectors	10
4 Konfigurierung der Visualisierung	11

Abbildungsverzeichnis

Abbildung 1: Aufbau und Beschreibung des Messkastens	4
Abbildung 2: Raspberry Pi mit RS232 Erweiterung	5
Abbildung 3: Übersicht des NodeRed Flows	6
Abbildung 4: Konfiguration des MBus Connectors	7
Abbildung 5: Code der Hilfsfunktion.....	8
Abbildung 6: Code einer Parsing Funktion.....	9
Abbildung 7: Konfiguration des InfluxDb Connectors	10
Abbildung 8: Konfiguration der Abfragequery.....	11

1 Einleitung

Diese Dokumentation beschreibt den Aufbau eines Datenabfrage eines Wärmemengenzählers und Stromzählers über MBus durch einen Raspberry Pi. Es werden hierbei die IoT-Komponenten Nodered zum Programmieren, eine Influx Datenbank zum Speichern der Daten sowie das Visualisierungstool Grafana verwendet.

Die Anwendung für dieses System ist das Monitoring einer Wärmepumpe. Hierzu wird auf der nach der Wärmepumpe ein Wärmemengenzähler (Typ Sharky 775) in den Hydraulikkreislauf eingebaut und vor der Wärmepumpe ein Stromzähler in den Stromkreislauf eingebaut. Beide Geräte verfügen über M-Bus schnittstellen.

2 Hardware

2.1 Aufbau des Messkasten

Die Hardware, umgesetzt in einem Messkasten, besteht aus einem Raspberry 3B+ mit angeschlossener RS232 Platine sowie einem MBus Pegelwandler.

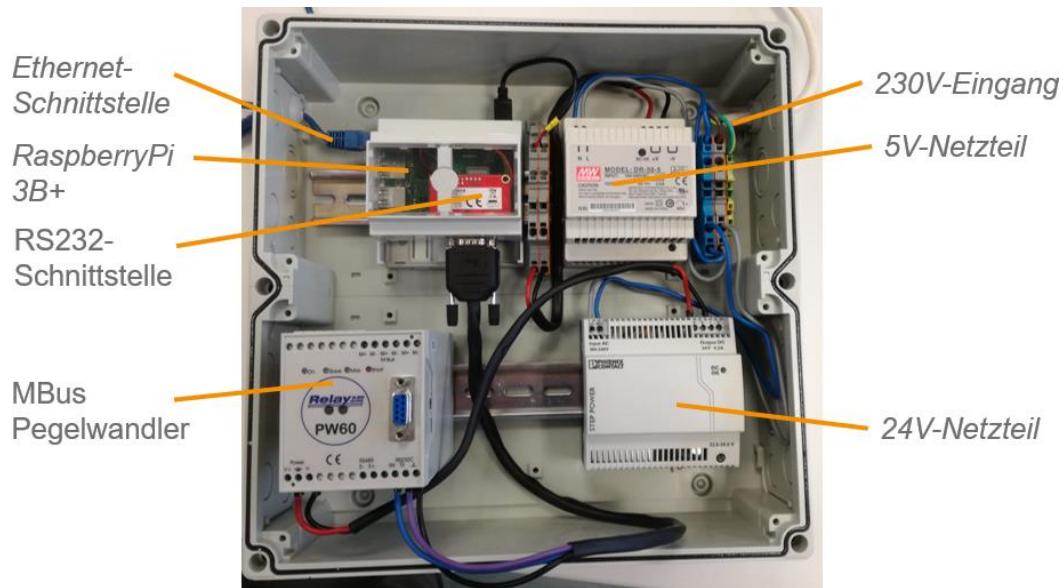


Abbildung 1: Aufbau und Beschreibung des Messkastens

Dabei dient das 24V Netzteil für die Versorgung des Pegelwandlers, das 5V Netzteil für die Versorgung des Raspberry Pi.

2.2 Einrichten der Seriellen Schnittstelle

Für die Serielle Schnittstelle wird die Erweiterungsplatine Renkforce RF-4011279 verwendet. Diese wird über das mitgelieferte Verbindungskabel an die GPIOs 02,06,08,10 angeschlossen

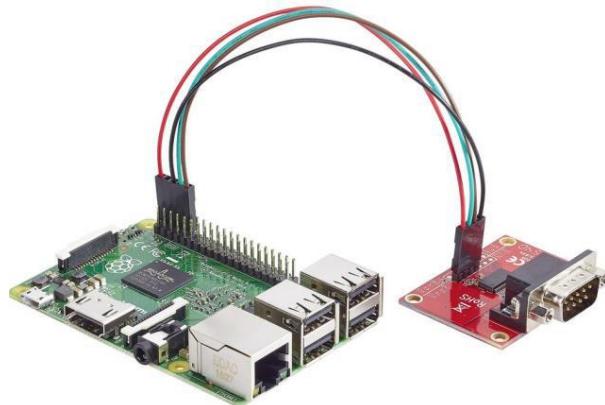


Abbildung 2: Raspberry Pi mit RS232 Erweiterung

Auf dem Raspberry muss die Serielle Schnittstelle erst eingerichtet und konfiguriert werden. Hierzu über den Befehl „sudo raspi-config“ das Konfigurationsmenü aufrufen und in den Interfacing Options → Serial zuerst bei der Frage nach dem „loginShell = Nein“ und bei der Frage nach Verwendung der „serial port hardware = Yes“ einstellen. Anschließend muss noch das vorhandene Bluetooth Modul deaktiviert und die UART Schnittstelle eingeschalten werden. Hierzu über den Befehl „sudo nano /boot/config.txt“ die Konfigurationsdatei bearbeiten. Am Ende der Datei muss die folgende Zeile eingefügt werden.

```
dtoverlay=pi3-miniuart-bt
```

An die RS232 Schnittstelle wird der Pegelwandler angeschlossen.

3 Konfiguration des Raspberry Pi

Als Betriebssystem des Raspberry Pi wird Raspbian Buster lite in der neuesten Version verwendet. Es werden die Programme Node-Red, InfluxDB und Grafana benötigt.

3.1 Konfiguration der Datenabfrage in Node-Red

Die Datenabfrage erfolgt über NodeRed. NodeRed wird unter Angabe der IP des Raspberry Pi mit dem Port 1880 im lokalen Netzwerk über den Browser erreicht.

Hierfür werden die folgenden „Paletten“ benötigt. Die Paletten sind die Bibliotheken in NodeRed, welche den Standardumfang um zusätzliche Programmierblöcke erweitern.

- Mbus Abfrage
<https://flows.nodered.org/node/node-red-contrib-m-bus>
- Speichern in der InfluxDB
<https://flows.nodered.org/node/node-red-contrib-influxdb>

Der grundsätzliche Flow für die Datenabfrage sieht wie folgt aus:

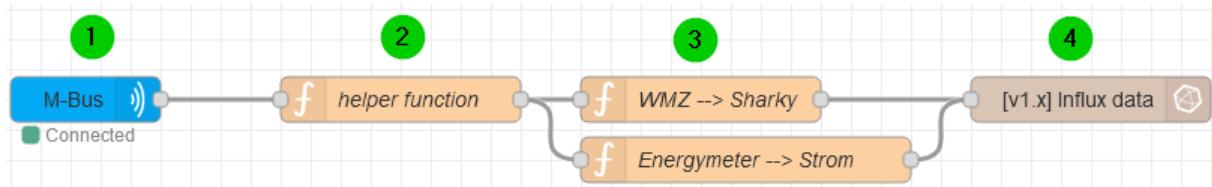


Abbildung 3: Übersicht des NodeRed Flows

3.1.1 Konfiguration des Mbus Connectors

Für das erstmalige einrichten wird der Beispiel Flow der contrib-m-bus Palette benötigt, um das Busnetz nach verfügbaren Geräten zu scannen und die Geräte-Ids herauszulesen. Danach werden die Geräte zyklisch durch den Connector abgefragt. Zur Konfiguration des Connectors werden folgende Einstellungen gewählt.

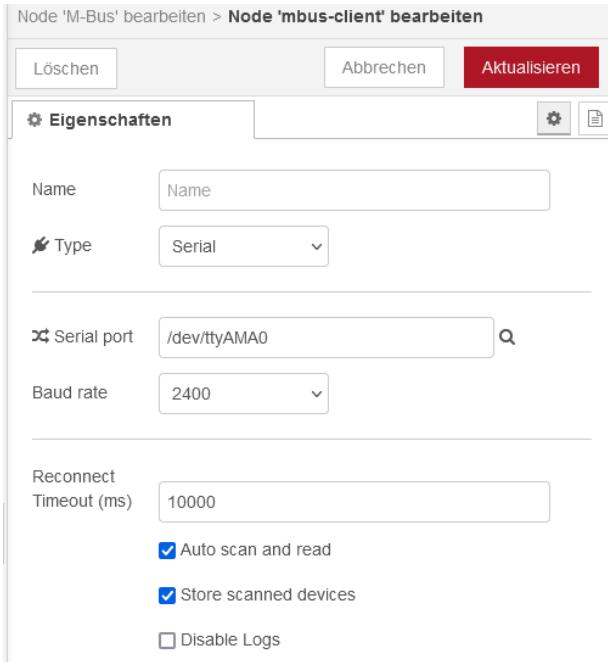


Abbildung 4: Konfiguration des MBus Connectors

Für den Serial Port wird die UART Schnittstelle angegeben. An dieser ist die RS232 Erweiterungsplatine softwareseitig angeschlossen

Die Baud rate ist abhängig von Einstellungen an den Geräten und dem Pegelwandler.

Die Reconnect Timeout Einstellung ist auch gleichzeitig der Zyklus für die Abfrage der Geräte. Für die Anwendung der Überwachung der Wärmepumpe wurde eine Zykluszeit von 10 Sekunden gewählt. Das Automatisierte Auslesen wird durch die Option „Auto scan and read“ eingeschaltet. Die Option „Store scanned decices“ erlaubt es den Connector ohne scannen des gesamten MBus zu verwenden.

Es gilt zu beachten, dass beim Mbus jedes Gerät separat abgefragt werden muss. Von daher ist auch das gleichzeitige Auslesen nicht möglich. Daher sollte für die Abfrage genügend Zeit (> 2 Sekunden) eingeplant werden.

Wird ein Gerät abgefragt, so entsteht eine NodeRed Message im JSON (JavaScript Object Notation) Format ausgegeben. Ein Beispiel für eine solche Nachricht sieht, in stark gekürzter Form wie folgt aus:

```

"topic": "mbDeviceUpdated",
"payload": {
  "SlaveInformation": {
    "Id": 30102556,
  },
  "DataRecord": [
    {
      "Unit": "Energy (Wh)",
      "Value": 3455620,
    },
    {
      "Unit": "1e-1 V",
      "Value": 2334,
    },
    {
      "Unit": "1e-1 A",
      "Value": 0,
    },
    {
      "Unit": "Power (W)",
      "Value": 34,
    }],
  },
}

```

Die Nachricht erhält weitaus mehr Informationen, welche aber für den weiteren Gebrauch nicht relevant sind und aus Gründen der Übersichtlichkeit entfernt wurden.

Die verwendeten Bestandteile der Nachricht sind wie folgt:

- Payload.SlaveInformation.Id → Die Sekundäradresse des Messgerätes
- Payload.DataRecord[].Unit → Die Einheit des Messpunktes
- Payload.DataRecord[].Value → Der Wert des Messpunktes

3.1.2 Konfiguration der Hilfsfunktion

In der Hilfsfunktion werden die Daten vorsortiert, um den Code der Parser Funktionen zu reduzieren. Eine IF Abfrage stellt noch sicher, dass nur Daten aus aktualisierten Sensordaten weiterverarbeitet werden. Der Code für die Funktion sieht wie folgt aus:

```

1 // Filtern der Nachrichten nach "Device Updated"
2 // Umsortieren der payload für leichteres Erreichen
3 if (msg.topic == "mbDeviceUpdated"){
4   msg.Device = msg.payload.SlaveInformation.Id;
5   msg.payload = msg.payload.DataRecord;
6   return msg;
7 }

```

Abbildung 5: Code der Hilfsfunktion

3.2 Konfiguration der Parser Funktionen

Die Parser Funktionen haben die Aufgabe, die Messdaten aus den Speicherregistern zu entnehmen. Für das Einspeichern der Daten in die InfluxDB müssen die Messdaten noch in eine entsprechende Form gebracht werden.

Grundsätzlich gibt es beim Parsen von Bus Daten in NodeRed mehrere Möglichkeiten. In diesem Fall wird für jedes Gerät eine eigenständige Funktion verwendet, welche die Nachrichten nach dem jeweiligen Gerät filtert. Ebenfalls kann die Funktionalität durch einen komplexeren Code vereinfacht und stark automatisiert werden. Im Sinne der Aufgabe einen Einstieg in die IoT-Messdatenaufnahme zu zeigen wurden die Funktionen bewusst einfach und rudimentär gehalten.

In der Folgenden Abbildung ist die Parser Funktion für einen der beiden Strommessgeräte (entspricht dem JSON Objekt aus Kapitel 3.1.1) dargestellt.

```
if (msg.Device == 30102556){  
    msg.payload = [{  
        Energy: msg.payload[0].Value,  
        Power: msg.payload[4].Value,  
        Voltage: msg.payload[2].Value/10,  
        Amps: msg.payload[3].Value/10,  
    },  
    {  
        Device: "Energymeter"  
    }];  
    return msg;  
}
```

Abbildung 6: Code einer Parsing Funktion

Die Nachricht wird durch eine IF Funktion anhand der sekundären Geräte Id gefiltert.

Die Messdaten des Gerätes werden anhand des Datenblattes, aus welchem die Registertabelle sowie die Einheiten und Skalierungsfaktoren entnommen werden, umgeformt.

So steht im Register Nr. Null der Wert für den Energiezähler mit einem Skalierungsfaktor von Eins. Dieser wird in ein Array der Message Payload mit der Variablen „Energy“ verschoben. Die restlichen Werte werden entsprechend umgewandelt, wobei bei den Werten für Volt und Ampere ein Skalierungsfaktor von $1 \cdot 10^{-1}$ verwendet werden muss.

Einschub:

Alle verwendeten Geräte liefern die Messwerte im Datenformat Integer. Somit ist keine Umwandlung des Datentyps nötig. Andere Geräte können Daten in Form von z.B. Float Zahlen liefern. Für diesen Fall muss der Messwert noch umgewandelt werden.

Das Einspeichern in die Influx erlaubt es, dem Messwerten noch Metadaten (Tag) mitzuliefern, welche im zweiten Array der Payload enthalten sein muss. So wird hier noch der Gerätetyp in Form des Tags „Device“ als String mitgeliefert. Dies erlaubt ein einfaches Filtern der Datenpunkte in der Grafana Visualisierungssoftware (siehe Kapitel 4)

3.2.1 Konfiguration des InfluxDB Connectors

Für das Einspeichern der Daten in die InfluxDB Datenbank wird der Connector wie folgt konfiguriert

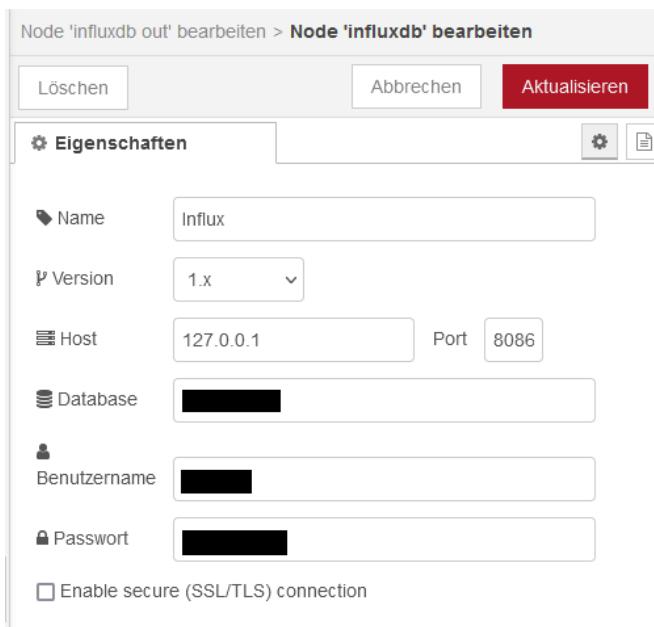


Abbildung 7: Konfiguration des InfluxDb Connectors

Da die Datenbank auf dem RaPi selbst läuft wird für die IP des Hosts die lokal IP 127.0.0.1 angegeben, der Port 8086 ist der Standard Influx Port.

Für die Database wird der Name einer erstellten Datenbank benötigt.

Es empfiehlt sich zur Absicherung der Datenbank einen Benutzer mit Passwort einzurichten.

Die Daten werden nach Vorgaben des Lineprotocols der InfluxDB entsprechend im JSON Format dem Koton zur Verfügung gestellt.

4 Konfigurierung der Visualisierung

Zur Visualisierung wird die Software Grafana verwendet, welche ebenfalls lokal auf dem RaPi läuft. Grafana wird unter Angabe der IP mit dem Port 3000 im lokalen Netzwerk über den Browser erreicht.

Für die Erstinstallation wird die Einstellung der Datenbank benötigt. Hierfür wird die IP der Datenbank (localhost) der Datenbankname sowie Benutzer und Passwort benötigt.

Zur Konfiguration der Abfragen aus der Datenbank (Querys) enthält Grafana einen einfachen Editor über welchen die Querys einfach konfiguriert werden können. In der folgenden Abbildung ist beispielhaft für einen Datenpunkt die Eingaben der Query dargestellt.

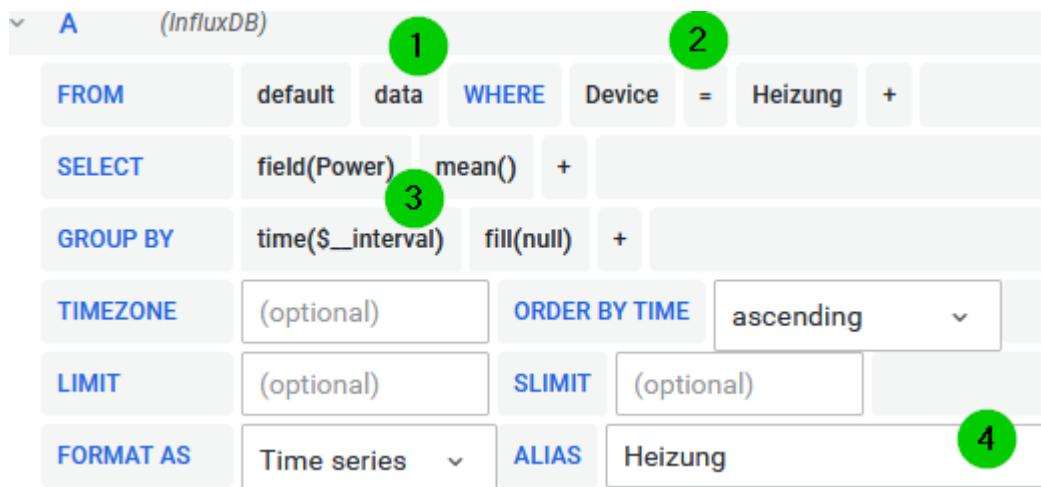


Abbildung 8: Konfiguration der Abfragequery

Es wird der Messpunkt „Leistung“ des Stromzählers der Heizung abgefragt. Hierzu sind folgende Eingaben nötig:

FROM:

Eingabe der Datenbank (default) und des Measurements (1) der Datenpunkte Durch den zusätzlichen Tag (2), „Device“ werden die Datenpunkte nach dem Messgerät gefiltert.

SELECT:

Als field (3) wird der Name des Messpunktes eingegeben. Es wird eine Aggregierungsweise Selektor Funktion benötigt.

GROUP By:

Enthält zusätzlich Funktionen zur Auswahl des Zeitraums. Durch den Standard Parameter erfolgt die Zeiteinstellung über das Dashboard in welchem die Visualisierung enthalten ist.

ALIAS:

Durch Eingabe des Alias wird der Legendenname eingegeben.

Gefördert durch:



Bundesministerium
für Wirtschaft
und Klimaschutz

aufgrund eines Beschlusses
des Deutschen Bundestages

Technische
Hochschule
Rosenheim



 **MonSec**

Setup Temperatur / Luftfeuchte Sensor DHT22 am Raspberry Pi

Im Projekt MonSEC- Monitoring Secure

TH Rosenheim

Bearbeiter: Markus Hartmann

Stand: 27.10.2022

Inhaltsverzeichnis

Inhaltsverzeichnis	2
Abbildungsverzeichnis	3
1 Einleitung	4
1.1 Sensor – DHT22.....	4
1.2 Raspberry Pi - Konfiguration	5
1.2.1 Konfiguration der Node-RED Knoten	5
2 Anhang	11
2.1 Arduino Code.....	11
2.1.1 Code für BME280	11
2.1.2 Code für DHT22	12

Abbildungsverzeichnis

Abbildung 1: DHT11 und DHT22 [Adafruit]	4
Abbildung 2: Anschlusschema DHT 22 an RaPi.....	5
Abbildung 3: Darstellung des Flows zur Abfrage des DHT22	6
Abbildung 4: Einstellungen des XBee RX Knoten	7
Abbildung 5: Beispieldaten (JSON) aus dem XBee RX Knoten	9
Abbildung 6: Datenstruktur der Payload.....	9
Abbildung 7: Konfiguration der Parsingfunktion.....	10

1 Einleitung

In dieser Dokumentation wird beschrieben, wie man einen DHT22 Sensor an einem Raspberry Pi anschließt und diesen per NodeRed in eine InfluxDB Datenbank einschreibt.

1.1 Sensor – DHT22

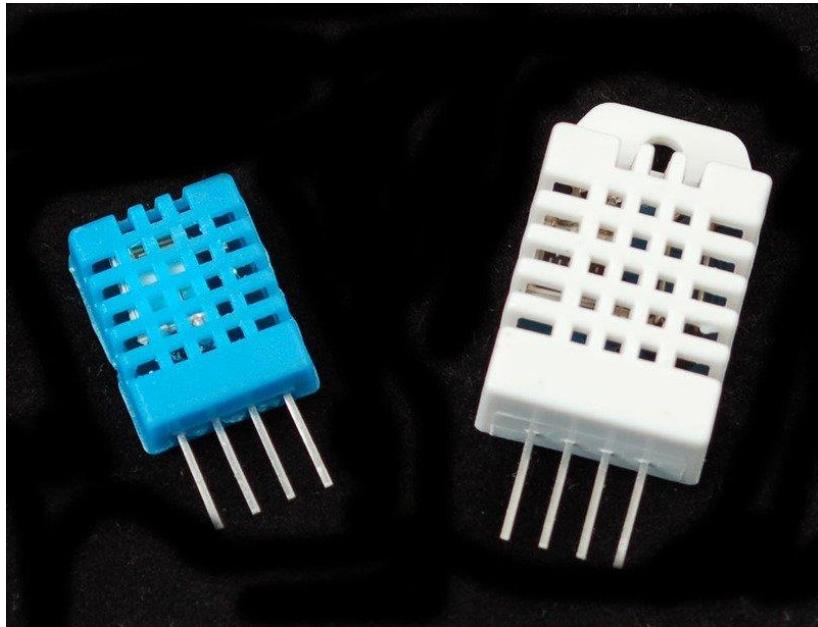


Abbildung 1: DHT11 und DHT22 [Adafruit]

Sowohl der DHT11 als auch der DHT22 sind Sensoren für Temperatur und Luftfeuchte. Die Genauigkeit des DHT22 ist jedoch höher. Die Sensoren sind vom Anschluss her gleich und verwenden den OneWire Bus. Sie werden nach folgendem Schema an den Raspberry Pi angeschlossen.

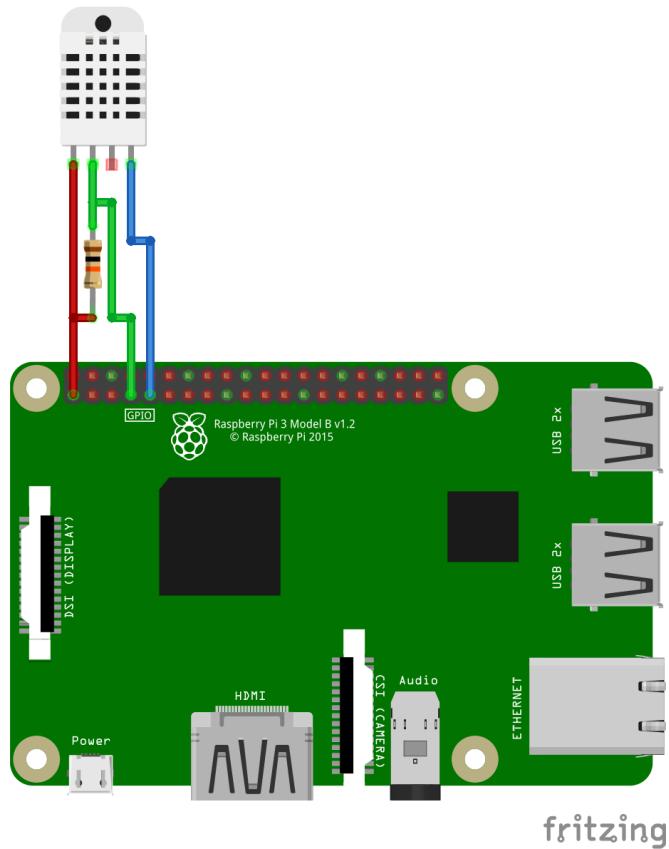


Abbildung 2: Anschlusschema DHT 22 an RaPi

Wie in der Abbildung zu erkennen wird der DHT 22 an die folgenden Pins angeschlossen

DHT 22	Raspberry Pi
Pin# 1 - VCC	Pin# 01 - 3,3V
Pin#2 - Data	Pin# 07 – GPIO 04
Pin#3	-
Pin#4 – Ground	Pin# 09 - Ground

1.2 Raspberry Pi - Konfiguration

Als Empfänger wird ein Raspberry Pi (3B+) verwendet. Das Betriebssystem ist dabei Raspian Buster lite in der neuesten Version.

1.2.1 Konfiguration der Node-RED Knoten

Die Abfrage des DHT 22 und das Einschreiben der Messdaten in die Datenbank ist über Node-RED realisiert. Als Datenbank dient eine InfluxDB die ebenfalls auf dem RaPi läuft.

In folgenden wird die Konfiguration der NodeRed Knoten erklärt.

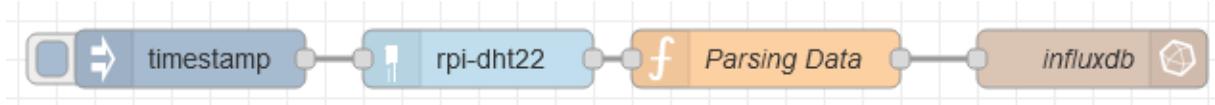


Abbildung 3: Darstellung des Flows zur Abfrage des DHT22

Es werden 4 Knoten benötigt:

- **Inject:**

Durch den Inject wird der Nachrichtenflow gestartet. Hier wird das Intervall der Messung eingegeben. Der Knoten startet dann die Messung automatisch nach Ablauf des festgelegten Zeitschritts

- **Rpi-dht22**

Dieser Knoten ist aus der Zusatzpalette „node-red-contrib-dht-sensor“ und stellt die Verbindungsbibliothek zum Sensor dar.

- **XBee_Parsing:**

Diese Funktion entschlüsselt die Payload Daten und verknüpft diese mit den Variablen für das Einspeichern in die Datenbank

- **Influx / Data:**

Anbindung an die Datenbank

Die Konfiguration des XBee Eingangknoten erfolgt nach dem folgenden Bild.

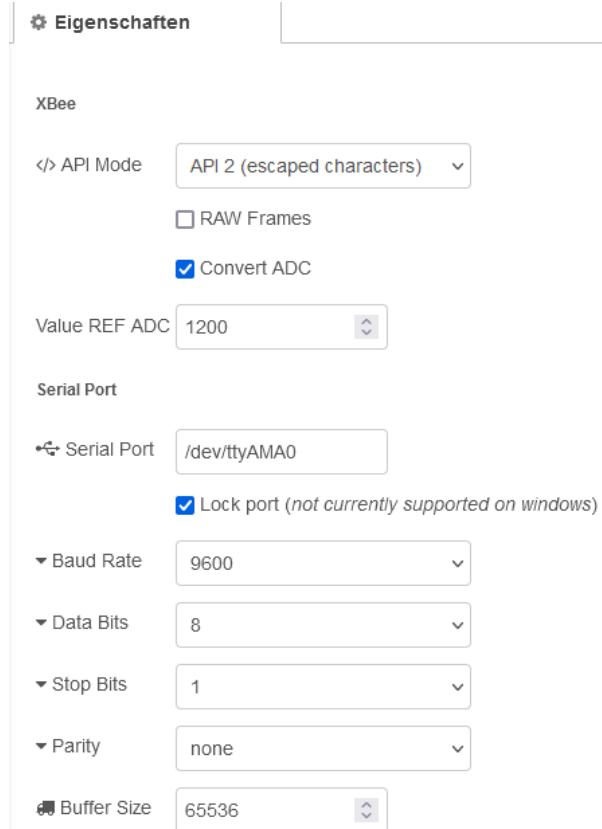


Abbildung 4: Einstellungen des XBee RX Knoten

rpi-dht22 Node bearbeiten

Löschen Abbrechen Fertig

Properties

Topic: rpi-dht22

Sensor model: DHT22

Pin numbering: Physical pins (rev. 1)

Pin number: 7

Name: Name

Der Knoten erzeugt bei eingehenden XBee Frame (Erkennung des Startwertes 0x7E) und gibt folgende Nachricht weiter.

```

4.3.2022, 11:12:09 node: bdc67596295f3265
msg : Object
  ▼ object
    ▼ payload: object
      type: 144
      remote64: "0013a20041d5bf55"
      remote16: "ffff"
      receiveOptions: 193
      ▼ data: buffer[8] raw
        0: 0x41
        1: 0x9a
        2: 0x66
        3: 0x67
        4: 0x42
        5: 0x1e
        6: 0x0
        7: 0x0
      _msgid: "e9309ad6b89907ea"

```

Abbildung 5: Beispielnachricht (JSON) aus dem XBee RX Knoten

In dieser Anwendung werden die Daten auf dem Arduino in ein UInt8-Array nach folgendem Schema übergeben.

MacAddress XBee:	0013a20041d5c40a				0013a20041d5bf55			
Sensor:	BME280				DHT22			
length payload	12				8			
Payload	Datotyp	Measurement	Unit	Influx	Datotyp	Measurement	Unit	Influx
0					0			
1					1			
2					2			
3	float	Temperature	[°C]	Temperature01	3	float	Temperature	[°C]
4					4			
5					5			
6					6			
7	float	Humidity	[%]	Humidity01	7	float	Humidity	[%]
8								
9								
10								
11	float	Pressure	[hPa]	Pressure01				

Abbildung 6: Datenstruktur der Payload

Die Konfiguration des Parsing Knoten erfolgt nach der folgenden Abbildung.

```
1 var Source = msg.payload.remote64;
2 msg.measurement = "home";
3 // Mapping für BME280
4 if (Source == "0013a20041d5c40a"){
5     msg.payload = {
6         Temperature01: msg.payload.data.readFloatBE(0),
7         Humidity01: msg.payload.data.readFloatBE(4),
8         Pressure01: msg.payload.data.readFloatBE(8),
9     };
10 }
11 // Mapping für DHT22
12 if (Source == "0013a20041d5bf55"){
13     msg.payload = {
14         Temperature02: msg.payload.data.readFloatBE(0),
15         Humidity02: msg.payload.data.readFloatBE(4),
16     };
17 }
18 return msg;
```

Abbildung 7: Konfiguration der Parsingfunktion

Durch die IF Bedingungen werden die Nachrichten nach der MAC Adresse der Sender XBee-Module sortiert und entsprechend

Der Ausgangsknoten für die Datenbank bedarf ausschließlich Standardeinstellungen.

2 Anhang

2.1 Arduino Code

2.1.1 Code für BME280

```
*****  
*  
* Programm zum Auslesen eines BME280 Sensors und senden der Daten über XBEE  
  
Markus Hartmann  
24.02.2022  
  
BME wird über I2C ausgelesen --> Wenn andere Pins verwendet werden, muss  
das Programm angepasst werden  
  
*****  
/  
#include <Wire.h>  
#include <SPI.h>  
#include <XBee.h>  
#include <Adafruit_Sensor.h>  
#include <Adafruit_BME280.h>  
  
#define SEALEVELPRESSURE_HPA (1013.25)  
// initialize BME280  
Adafruit_BME280 bme; // I2C  
// Create XBee object  
XBee xbee = XBee();  
// Initialize payload (für BME 12 bytes)  
uint8_t ui8Payload[12] =  
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};  
//setup xbee request  
// Adresse Coordinator  
XBeeAddress64 addr64 = XBeeAddress64(0x0013a200, 0x41d5c29a);  
ZBTxRequest zbTx = ZBTxRequest(addr64, ui8Payload, sizeof(ui8Payload));  
// Intervall für die Messungen  
int delayTime = 6000;  
  
void setup() {  
    Serial.begin(9600);  
    xbee.setSerial(Serial);  
    delay(2000);  
    bme.begin();  
    delay(2000);  
}  
  
void loop() {  
    //read Temperature [°C]  
    uint32_t ui32Temperature = 0;  
    float fTemperature = bme.readTemperature();  
    ui32Temperature = (reinterpret_cast<uint32_t&>(fTemperature));  
    //read Humidity [%]  
    uint32_t ui32Humidity = 0;
```

```

float fHumidity = bme.readHumidity();
ui32Humidity = (reinterpret_cast<uint32_t&>(fHumidity));
//read Pressure
uint32_t ui32Pressure = 0;
float fPressure = bme.readPressure()/100.0F;
ui32Pressure = (reinterpret_cast<uint32_t&>(fPressure));
// Mapping to Payload
//Temperature
ui8Payload[0] = (byte) ( (ui32Temperature >>24) &0x000000FF );
ui8Payload[1] = (byte) ( (ui32Temperature >>16) &0x000000FF );
ui8Payload[2] = (byte) ( (ui32Temperature >>8) &0x000000FF );
ui8Payload[3] = (byte) ( ui32Temperature &0x000000FF );
//Humidity
ui8Payload[4] = (byte) ( (ui32Humidity >>24) &0x000000FF );
ui8Payload[5] = (byte) ( (ui32Humidity >>16) &0x000000FF );
ui8Payload[6] = (byte) ( (ui32Humidity >>8) &0x000000FF );
ui8Payload[7] = (byte) ( ui32Humidity &0x000000FF );
//Pressure
ui8Payload[8] = (byte) ( (ui32Pressure >>24) &0x000000FF );
ui8Payload[9] = (byte) ( (ui32Pressure >>16) &0x000000FF );
ui8Payload[10] = (byte) ( (ui32Pressure >>8) &0x000000FF );
ui8Payload[11] = (byte) ( ui32Pressure &0x000000FF );
xbee.send(zbTx);
delay(delayTime);
}

```

2.1.2 Code für DHT22

```

/*****
*
* Programm zum Auslesen eines DHT22 Sensors und senden der Daten über XBEE
*
Markus Hartmann
24.02.2022
*****
/
#include <XBee.h>
#include "DHT.h"

#define DHTPIN 2           // Digital pin connected to the DHT sensor
#define DHTTYPE DHT22    // DHT 22  (AM2302, AM2321

// Initialize DHT sensor.
DHT dht(DHTPIN, DHTTYPE);
// Create an XBee object at the top of your sketch
XBee xbee = XBee();
// Initialize payload (für DHT22 8 bytes)
uint8_t ui8Payload[8] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
//setup xbee request
XBeeAddress64 addr64 = XBeeAddress64(0x0013a200, 0x41d5c29a);
ZBTxRequest zbTx = ZBTxRequest(addr64, ui8Payload, sizeof(ui8Payload));

// Intervall für die Messungen
int delayTime = 6000;

void setup() {

```

```
Serial.begin(9600);
xbee.setSerial(Serial);
delay(1000);
dht.begin();
delay(1000);
}

void loop() {
//read Temperature [°C]
uint32_t ui32Temperature = 0;
float fTemperature = dht.readTemperature();
ui32Temperature = (reinterpret_cast<uint32_t&>(fTemperature));
//read Humidity [%]
uint32_t ui32Humidity = 0;
float fHumidity = dht.readHumidity();
ui32Humidity = (reinterpret_cast<uint32_t&>(fHumidity));

// Mapping to Payload
//Temperature
ui8Payload[0] = (byte)( (ui32Temperature >>24) &0x000000FF );
ui8Payload[1] = (byte)( (ui32Temperature >>16) &0x000000FF );
ui8Payload[2] = (byte)( (ui32Temperature >>8)  &0x000000FF );
ui8Payload[3] = (byte)( ui32Temperature        &0x000000FF );
//Humidity
ui8Payload[4] = (byte)( (ui32Humidity >>24) &0x000000FF );
ui8Payload[5] = (byte)( (ui32Humidity >>16) &0x000000FF );
ui8Payload[6] = (byte)( (ui32Humidity >>8)   &0x000000FF );
ui8Payload[7] = (byte)( ui32Humidity        &0x000000FF );

xbee.send(zbTx);
delay(delayTime);
}
```

Gefördert durch:



Bundesministerium
für Wirtschaft
und Klimaschutz

aufgrund eines Beschlusses
des Deutschen Bundestages

Technische
Hochschule
Rosenheim



 MonSec

XBee – Setup (API-Mode)

Im Projekt MonSEC- Monitoring Secure

TH Rosenheim

Bearbeiter: Markus Hartmann

Stand: 06.12.2022

Inhaltsverzeichnis

Inhaltsverzeichnis	2
Abbildungsverzeichnis	3
1 Einleitung	4
1.1 Quellen	4
1.1.1 Offizielle Quellen	4
1.1.2 Blog Artikel und sonstiges für Entwicklung.....	4
2 Aufbau und Setup.....	5
2.1 XBee Module	5
2.1.1 Coordinator.....	5
2.1.2 Endknoten	6
2.2 Arduino	8
2.2.1 Sensor – BME280	9
2.2.2 Sensor – DHT22.....	10
2.3 Raspberry Pi - Empfängerknoten	11
2.3.1 Einrichten der Seriellen Schnittstelle.....	12
2.3.2 Konfiguration der Node-RED Knoten	13
3 Anhang	17
3.1 Arduino Code.....	17
3.1.1 Code für BME280	17
3.1.2 Code für DHT22	18

Abbildungsverzeichnis

Abbildung 1: Parameter des Coordinators	5
Abbildung 2: Konfiguration der Endknoten	7
Abbildung 3: XBee Shield für Arduino [DFRobot]	8
Abbildung 4: BME280 [Adafruit]	9
Abbildung 5: Anschlusschema BME280 an Arduino [http://cactus.io/]	9
Abbildung 6: DHT11 und DHT22 [Adafruit]	10
Abbildung 7: Anschlusschema DHT22 [Adafruit]	11
Abbildung 8: Raspberry Pi mit RS232 Erweiterung	12
Abbildung 9: XBee Explorer Serial Entwicklungsplatine	13
Abbildung 10: Darstellung XBee Parsing zur Datenbank	13
Abbildung 11: Einstellungen des XBee RX Knoten	14
Abbildung 12: Beispielnachricht (JSON) aus dem XBee RX Knoten	15
Abbildung 13: Datenstruktur der Payload.....	15
Abbildung 14: Konfiguration der Parsingfunktion.....	16

1 Einleitung

Diese Dokumentation beschreibt den Aufbau von mobilen Sensorknoten zum Messen von Raumluftdaten (Temperatur, Luftfeuchte sowie Luftdruck) und Übertragung der Daten über XBee Funkmodule (Firma Digi). Die Sender werden auf Basis eines Arduino Uno mit zwei verschiedenen Sensoren (Bosch BME280 sowie DHT22) realisiert. Als Empfänger wird ein Raspberry Pi verwendet.

1.1 Quellen

1.1.1 Offizielle Quellen

XBee API Mode

https://www.digi.com/resources/documentation/Digidocs/90001942-13/containers/cont_api_mode.htm?tocpath=XBee%20API%20mode%7C_0

XBee Reichweitentest

<https://de.digi.com/blog/post/wireless-communication-range-testing-with-digi-xct>

1.1.2 Blog Artikel und sonstiges für Entwicklung

<https://www.instructables.com/How-to-Use-XBee-Modules-As-Transmitter-Receiver-Ar/>

<https://www.ardumotive.com/how-to-use-xbee-modules-as-transmitter--receiver-en.html>

2 Aufbau und Setup

2.1 XBee Module

Die XBee Module werden über die von Digi bereitgestellte Software XTCU konfiguriert.

2.1.1 Coordinator

Der Connector stellt den Sammelknoten dar, der als Koordinator des Netzwerkes und der Sammlung der Daten entspricht. Die beiden wichtigen Einstellungen sind in der folgenden Darstellung rot markiert.

The screenshot shows the XTCU software interface for configuring an XBee module. The configuration is divided into several sections:

- Networking & Security**: Contains fields for CH Channel (C), ID Network ID (2015), MT Broadcast Multi-Transmits (3), PL TX Power Level (Highest [4]), PM Power Mode (Boost Mode Enabled [1]), RR Unicast Retries (A), and CA CCA Threshold (0). The 'ID Network ID' field is highlighted with a red border.
- Diagnostic - MAC Statistics and Timeouts**: Shows various MAC statistics and timeouts, all of which are currently at zero. The entire section is highlighted with a red border.
- Network**: Contains fields for CE Coordinator/End-Device Mode (Indirect Msg Coordinator [1]), BH Broadcast Hops (0), DM DigiMesh Options (0), and NH Network Hops (7). The 'CE Coordinator/End-Device Mode' dropdown is highlighted with a red border.
- Serial Interfacing**: Contains fields for BD Baud Rate (9600 [3]), NB Parity (No Parity [0]), RO Packetization Timeout (3), FT Flow Control Threshold (51), AP API Enable (API Mode With Escapes [2]), and AO API Options (API Rx Indicator - 0x90 [0]). The 'AP API Enable' dropdown and the 'AO API Options' dropdown are highlighted with red borders.

Abbildung 1: Parameter des Coordinators

Es sollte auch die Adresse des Coordinators ausgelesen werden, damit die Endknoten diese als Empfänger der Nachricht erkennen.

2.1.2 Endknoten

Die Endknoten sind die Sensorknoten. Diese stellen die Daten dem Coordinator zu Verfügung.

Die Endknoten müssen im gleichen Netzwerk – ID sein, damit diese Nachrichten an den Coordinator verschicken können. Die Parametrierung ist in der folgenden Darstellung abgebildet.

Networking & Security
Modify networking settings

i CH Channel	C
i ID Network ID	2015
i MT Broadcast Multi-Transmits	3
i PL TX Power Level	Highest [4]
i PM Power Mode	Boost Mode Enabled [1]
i RR Unicast Retries	A
i CA CCA Threshold	0 -dBm

Diagnostic - MAC Statistics and Timeouts
MAC Statistics and Timeouts.

i BC Bytes Transmitted	D8
i DB Last Packet RSSI	0
i GD Good Packets Received	0
i EA MAC ACK Failure Count	0
i EC CCA Failure Count	0
i TR Transmission Failure Count	0
i UA Unicasts Attempted Count	0
i %H MAC Unicast One Hop Time	19
i %B MAC Broadcast One Hop Time	2C

Network
Change DigiMesh Network Settings

i CE Coordinator/End-Device Mode	Non-Routing Module [2]
i BH Broadcast Hops	0
i DM DigiMesh Options	0 Bitfield
i NH Network Hops	7 Hops

Addressing
Change Addressing Settings

i SH Serial Number High	13A200
i SL Serial Number Low	41F5B2E6
i DH Destination Address High	0
i DL Destination Address Low	FFFF
i TO Transmit Options	C0 Bitfield
i NI Node Identifier	ED4
i NT Network Discovery Back-off	82 * 100 ms
i NO Network Discovery Options	0 Bitfield
i CI Cluster ID	11

Serial Interfacing
Change module interfacing options

i BD Baud Rate	9600 [3]
i NB Parity	No Parity [0]
i RO Packetization Timeout	3 * character times
i FT Flow Control Threshold	51 Bytes
i AP API Enable	API Mode With Escapes [2]
i AO API Options	API Rx Indicator - 0x90 [0]

Abbildung 2: Konfiguration der Endknoten

Der Parameter NI Node Identifier ist für jeden Knoten verschieden zu vergeben

2.2 Arduino

Für dieses Projekt wird ein Arduino Uno verwendet.

Für die Verbindung zwischen Arduino und Xbee wird ein Shild der Firma DF Robot verwendet.

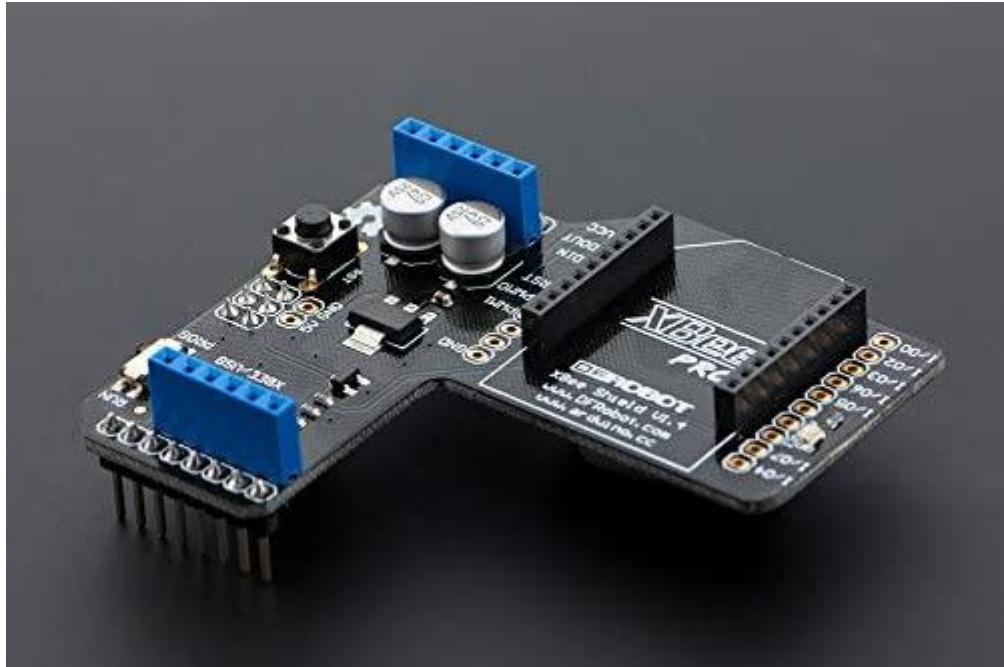


Abbildung 3: XBee Shield für Arduino [DFRobot]

Die Anleitung für das Shild ist auf der folgenden Seite einzusehen:

https://wiki.dfrobot.com/Xbee_Shield_For_Arduino_no_Xbee_SKU_DFR0015

Als Bibliothek für die XBee Kommunikation wird auf dem Arduino die folgende Bibliothek verwendet:

<https://github.com/andrewrapp/xbee-arduino>

2.2.1 Sensor – BME280

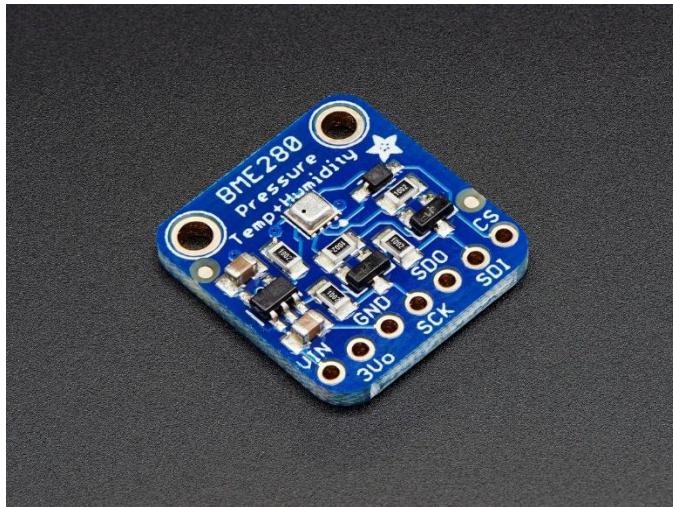


Abbildung 4: BME280 [Adafruit]

Als Sensor wird ein BME280 von der Firma Bosch verwendet. Es handelt sich um einen Kombinierten Sensor für Temperatur, Luftfeuchte und Luftdruck.

Für den leichteren Einsatz wird eine fertig Bestückte Platine von der Firma Adfruit verwendet. Der Sensor wird über den I2C Bus an den Arduino verbunden. Er wird nach folgendem Schema an den Arduino angeschlossen:

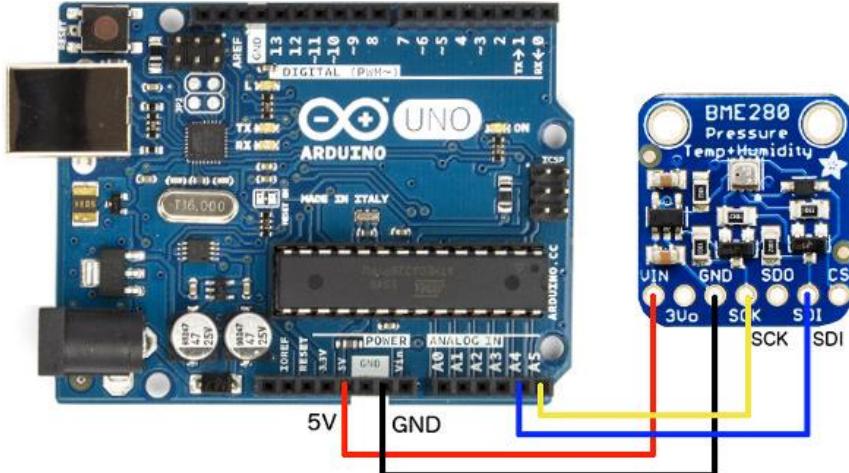


Abbildung 5: Anschlusschema BME280 an Arduino [<http://cactus.io/>]

Die Bibliothek für den Sensor ist auf Github zu finden.

https://github.com/adafruit/Adafruit_BME280_Library

Der Sensor wird durch das Programm abgefragt und die Daten werden in Binärer Form an das XBee Empfängermodul gesendet. Der Code auf dem Arduino ist im Anhang (Kapitel 3.1.1) zu finden.

2.2.2 Sensor – DHT22

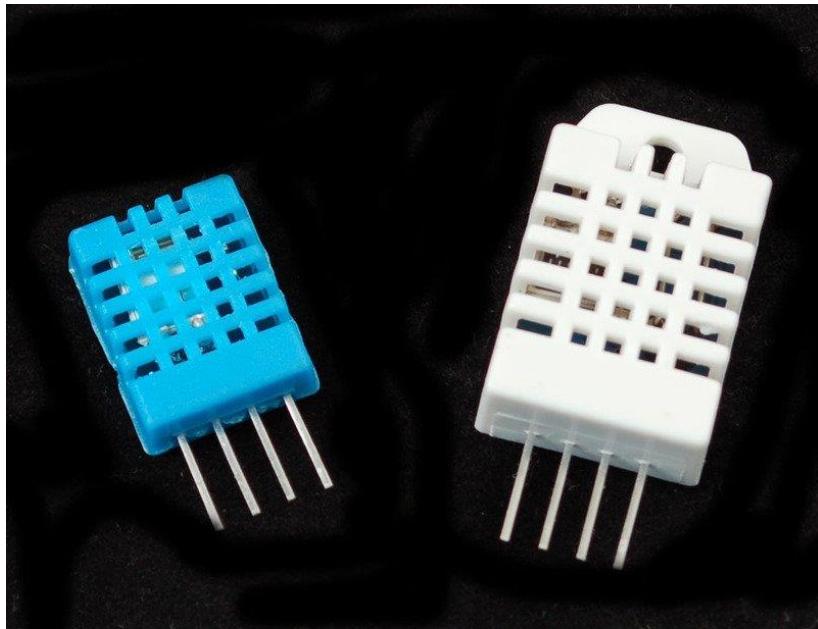


Abbildung 6: DHT11 und DHT22 [Adafruit]

Sowohl der DHT11 als auch der DHT22 sind Sensoren für Temperatur und Luftfeuchte. Die Genauigkeit des DHT22 ist jedoch höher. Die Sensoren sind vom Anschluss her gleich und verwenden den OneWire Bus. Sie werden nach folgendem Schema an den Arduino angeschlossen.

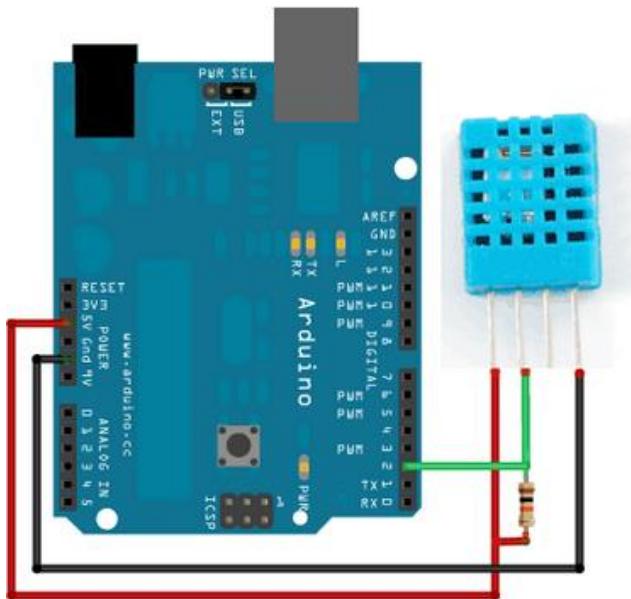


Abbildung 7: Anschlusschema DHT22 [Adafruit]

Wie in Abbildung 7 zu erkennen muss ein Widerstand (Pullup, 10 K Ω) zwischen die Datenleitung und der Spannungsversorgung eingebaut werden (Sofern noch nicht vorhanden).

Die Bibliothek ist direkt in der Arduino IDE unter dem Stichwort DHT zu finden. Bei neuen Versionen muss noch die Adafruit Unified Sensor Bibliothek zusätzlich eingebunden werden.

Der Code auf dem Arduino ist im Anhang (Kapitel 3.1.2) zu finden.

2.3 Raspberry Pi - Empfängerknoten

Als Empfänger wird ein Raspberry Pi (3B+) verwendet. Das Betriebssystem ist dabei Raspian Buster lite in der der neuesten Version. An dem RaspberryPi ist der XBee Coordinator über eine Serielle Schnittstelle (RS232) angeschlossen ist. Die Schnittstelle wird über eine Erweiterungsplatine realisiert.

2.3.1 Einrichten der Seriellen Schnittstelle

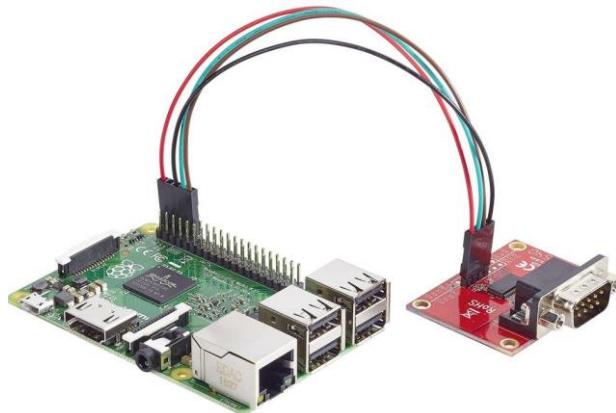


Abbildung 8: Raspberry Pi mit RS232 Erweiterung

Auf dem Raspberry muss die Serielle Schnittstelle erst eingerichtet und konfiguriert werden. Hierzu über den Befehl „sudo raspi-config“ das Konfigurationsmenü aufrufen und in den Interfacing Options → Serial zuerst bei der Frage nach dem „loginShell = Nein“ und bei der Frage nach Verwendung der „serial port hardware = Yes“ einstellen. Anschließend muss noch das vorhandene Bluetooth Modul deaktiviert und die UART Schnittstelle eingeschalten werden. Hierzu über den Befehl „sudo nano /boot/config.txt“ die Konfigurationsdatei bearbeiten. Am Ende der Datei muss die folgende Zeile eingefügt werden.

```
dtoverlay=pi3-miniuart-bt
```

An die RS232 Schnittstelle wird das XBee Explorer Serial mit XBee Modul per D-Sub Kabel angeschlossen.

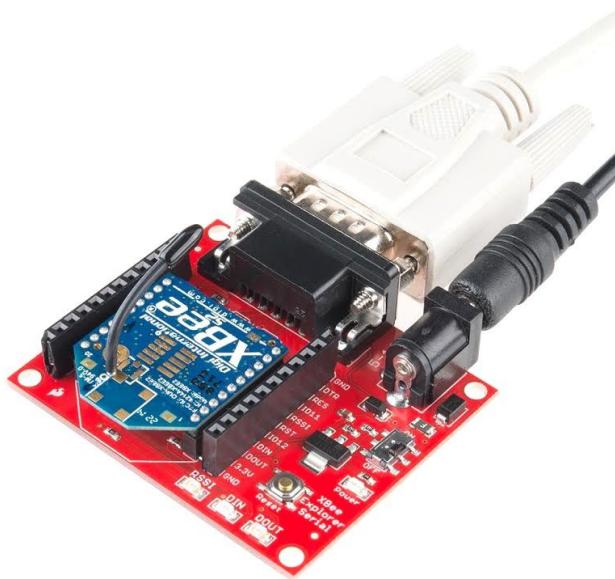


Abbildung 9: XBee Explorer Serial Entwicklungsplatine

2.3.2 Konfiguration der Node-RED Knoten

Die Abfrage der Seriellen Schnittstelle und das Einschreiben der Messdaten in die Datenbank ist über Node-RED realisiert. Als Datenbank dient eine InfluxDB die ebenfalls auf dem RaPi läuft.

In folgenden wird die Konfiguration der NodeRed Knoten erklärt.

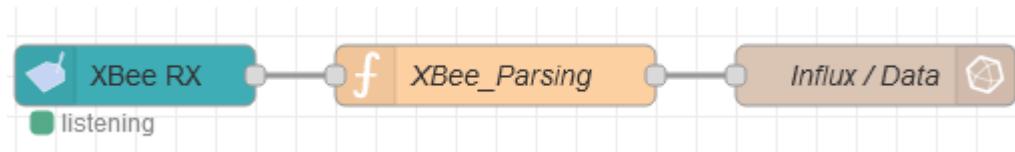


Abbildung 10: Darstellung XBee Parsing zur Datenbank

Es werden 3 Knoten benötigt:

- XBee RX:

Dieser Knoten beobachtet die Serielle Schnittstelle auf eingehende XBee Nachrichten.

Dieser Knoten entstammt aus der Palette „node-red-contrib-xbee“

- XBee_Parsing:

Diese Funktion entschlüsselt die Payload Daten und verknüpft diese mit den Variablen für das Einspeichern in die Datenbank

- Influx / Data:

Anbindung an die Datenbank

Die Konfiguration des XBee Eingangknoten erfolgt nach dem folgenden Bild.

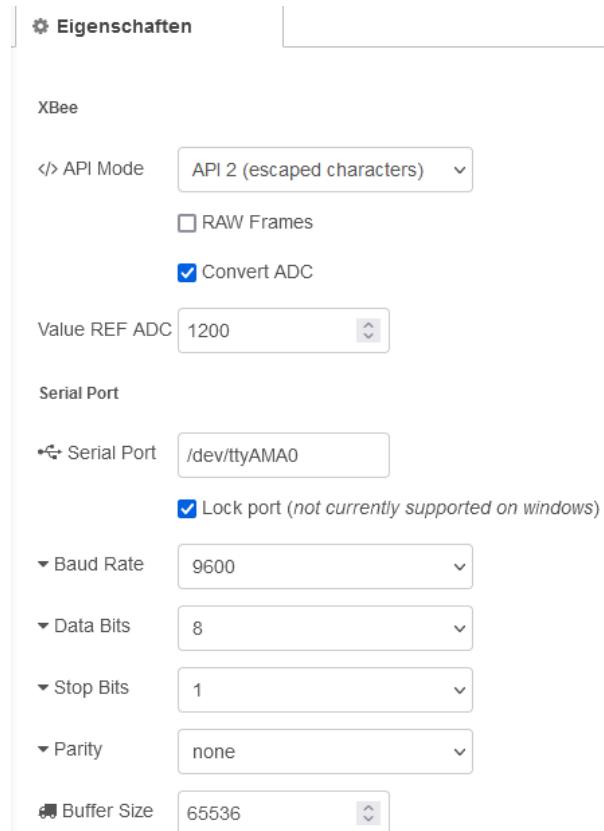


Abbildung 11: Einstellungen des XBee RX Knoten

Der Knoten erzeugt bei eingehenden XBee Frame (Erkennung des Startwertes 0x7E) und gibt folgende Nachricht weiter.

```

4.3.2022, 11:12:09 node: bdc67596295f3265
msg : Object
  ▼ object
    ▼ payload: object
      type: 144
      remote64: "0013a20041d5bf55"
      remote16: "ffff"
      receiveOptions: 193
      ▼ data: buffer[8] raw
        0: 0x41
        1: 0x9a
        2: 0x66
        3: 0x67
        4: 0x42
        5: 0x1e
        6: 0x0
        7: 0x0
      _msgid: "e9309ad6b89907ea"

```

Abbildung 12: Beispielnachricht (JSON) aus dem XBee RX Knoten

In dieser Anwendung werden die Daten auf dem Arduino in ein UInt8-Array nach folgendem Schema übergeben.

MacAddress XBee:	0013a20041d5c40a				0013a20041d5bf55			
Sensor:	BME280				DHT22			
length payload	12				8			
Payload	Datotyp	Measurement	Unit	Influx	Datotyp	Measurement	Unit	Influx
0					0			
1					1			
2					2			
3	float	Temperature	[°C]	Temperature01	3	float	Temperature	[°C]
4					4			
5					5			
6					6			
7	float	Humidity	[%]	Humidity01	7	float	Humidity	[%]
8								
9								
10								
11	float	Pressure	[hPa]	Pressure01				

Abbildung 13: Datenstruktur der Payload

Die Konfiguration des Parsing Knoten erfolgt nach der folgenden Abbildung.

```
1 var Source = msg.payload.remote64;
2 msg.measurement = "home";
3 // Mapping für BME280
4 if (Source == "0013a20041d5c40a"){
5     msg.payload = {
6         Temperature01: msg.payload.data.readFloatBE(0),
7         Humidity01: msg.payload.data.readFloatBE(4),
8         Pressure01: msg.payload.data.readFloatBE(8),
9     };
10 }
11 // Mapping für DHT22
12 if (Source == "0013a20041d5bf55"){
13     msg.payload = {
14         Temperature02: msg.payload.data.readFloatBE(0),
15         Humidity02: msg.payload.data.readFloatBE(4),
16     };
17 }
18 return msg;
```

Abbildung 14: Konfiguration der Parsingfunktion

Durch die IF Bedingungen werden die Nachrichten nach der MAC Adresse der Sender XBee-Module sortiert und entsprechend

Der Ausgangsknoten für die Datenbank bedarf ausschließlich Standardeinstellungen.

3 Anhang

3.1 Arduino Code

3.1.1 Code für BME280

```
*****  
*  
* Programm zum Auslesen eines BME280 Sensors und senden der Daten über XBEE  
  
Markus Hartmann  
24.02.2022  
  
BME wird über I2C ausgelesen --> Wenn andere Pins verwendet werden, muss  
das Programm angepasst werden  
  
*****  
/  
#include <Wire.h>  
#include <SPI.h>  
#include <XBee.h>  
#include <Adafruit_Sensor.h>  
#include <Adafruit_BME280.h>  
  
#define SEALEVELPRESSURE_HPA (1013.25)  
// initialize BME280  
Adafruit_BME280 bme; // I2C  
// Create XBee object  
XBee xbee = XBee();  
// Initialize payload (für BME 12 bytes)  
uint8_t ui8Payload[12] =  
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};  
//setup xbee request  
// Adresse Coordinator  
XBeeAddress64 addr64 = XBeeAddress64(0x0013a200, 0x41d5c29a);  
ZBTxRequest zbTx = ZBTxRequest(addr64, ui8Payload, sizeof(ui8Payload));  
// Intervall für die Messungen  
int delayTime = 6000;  
  
void setup() {  
    Serial.begin(9600);  
    xbee.setSerial(Serial);  
    delay(2000);  
    bme.begin();  
    delay(2000);  
}  
  
void loop() {  
    //read Temperature [°C]  
    uint32_t ui32Temperature = 0;  
    float fTemperature = bme.readTemperature();  
    ui32Temperature = (reinterpret_cast<uint32_t&>(fTemperature));  
    //read Humidity [%]  
    uint32_t ui32Humidity = 0;
```

```

float fHumidity = bme.readHumidity();
ui32Humidity = (reinterpret_cast<uint32_t&>(fHumidity));
//read Pressure
uint32_t ui32Pressure = 0;
float fPressure = bme.readPressure()/100.0F;
ui32Pressure = (reinterpret_cast<uint32_t&>(fPressure));
// Mapping to Payload
//Temperature
ui8Payload[0] = (byte) ( (ui32Temperature >>24) &0x000000FF );
ui8Payload[1] = (byte) ( (ui32Temperature >>16) &0x000000FF );
ui8Payload[2] = (byte) ( (ui32Temperature >>8) &0x000000FF );
ui8Payload[3] = (byte) ( ui32Temperature &0x000000FF );
//Humidity
ui8Payload[4] = (byte) ( (ui32Humidity >>24) &0x000000FF );
ui8Payload[5] = (byte) ( (ui32Humidity >>16) &0x000000FF );
ui8Payload[6] = (byte) ( (ui32Humidity >>8) &0x000000FF );
ui8Payload[7] = (byte) ( ui32Humidity &0x000000FF );
//Pressure
ui8Payload[8] = (byte) ( (ui32Pressure >>24) &0x000000FF );
ui8Payload[9] = (byte) ( (ui32Pressure >>16) &0x000000FF );
ui8Payload[10] = (byte) ( (ui32Pressure >>8) &0x000000FF );
ui8Payload[11] = (byte) ( ui32Pressure &0x000000FF );
xbee.send(zbTx);
delay(delayTime);
}

```

3.1.2 Code für DHT22

```

/*****
*
* Programm zum Auslesen eines DHT22 Sensors und senden der Daten über XBEE
*
Markus Hartmann
24.02.2022
*****
/
#include <XBee.h>
#include "DHT.h"

#define DHTPIN 2          // Digital pin connected to the DHT sensor
#define DHTTYPE DHT22    // DHT 22  (AM2302, AM2321

// Initialize DHT sensor.
DHT dht(DHTPIN, DHTTYPE);
// Create an XBee object at the top of your sketch
XBee xbee = XBee();
// Initialize payload (für DHT22 8 bytes)
uint8_t ui8Payload[8] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
//setup xbee request
XBeeAddress64 addr64 = XBeeAddress64(0x0013a200, 0x41d5c29a);
ZBTxRequest zbTx = ZBTxRequest(addr64, ui8Payload, sizeof(ui8Payload));

// Intervall für die Messungen
int delayTime = 6000;

void setup() {

```

```
Serial.begin(9600);
xbee.setSerial(Serial);
delay(1000);
dht.begin();
delay(1000);
}

void loop() {
//read Temperature [°C]
uint32_t ui32Temperature = 0;
float fTemperature = dht.readTemperature();
ui32Temperature = (reinterpret_cast<uint32_t&>(fTemperature));
//read Humidity [%]
uint32_t ui32Humidity = 0;
float fHumidity = dht.readHumidity();
ui32Humidity = (reinterpret_cast<uint32_t&>(fHumidity));

// Mapping to Payload
//Temperature
ui8Payload[0] = (byte)( (ui32Temperature >>24) &0x000000FF );
ui8Payload[1] = (byte)( (ui32Temperature >>16) &0x000000FF );
ui8Payload[2] = (byte)( (ui32Temperature >>8)  &0x000000FF );
ui8Payload[3] = (byte)( ui32Temperature        &0x000000FF );
//Humidity
ui8Payload[4] = (byte)( (ui32Humidity >>24) &0x000000FF );
ui8Payload[5] = (byte)( (ui32Humidity >>16) &0x000000FF );
ui8Payload[6] = (byte)( (ui32Humidity >>8)   &0x000000FF );
ui8Payload[7] = (byte)( ui32Humidity        &0x000000FF );

xbee.send(zbTx);
delay(delayTime);
}
```