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

Abbildungsverzeichnis

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

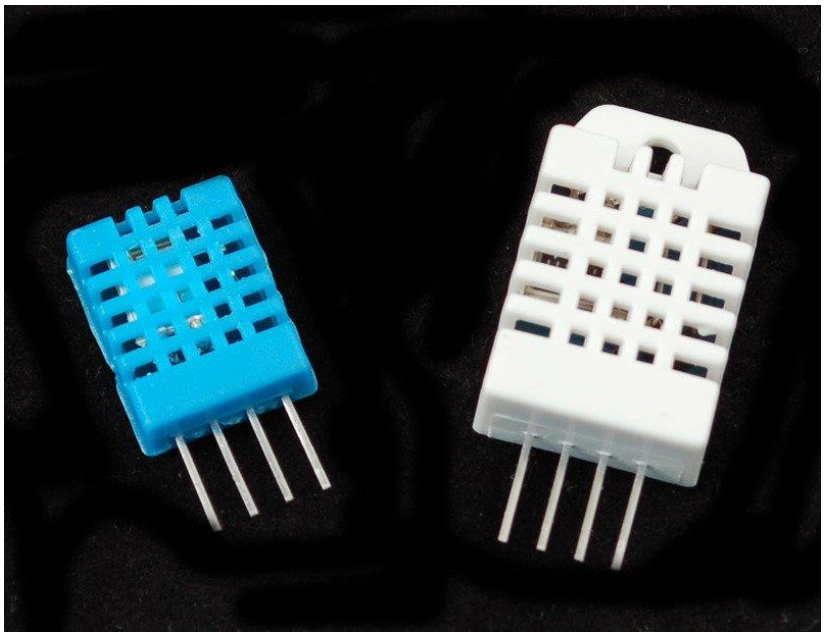


Abbildung 1: DHT11 und DHT22 [Adafruit]

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.

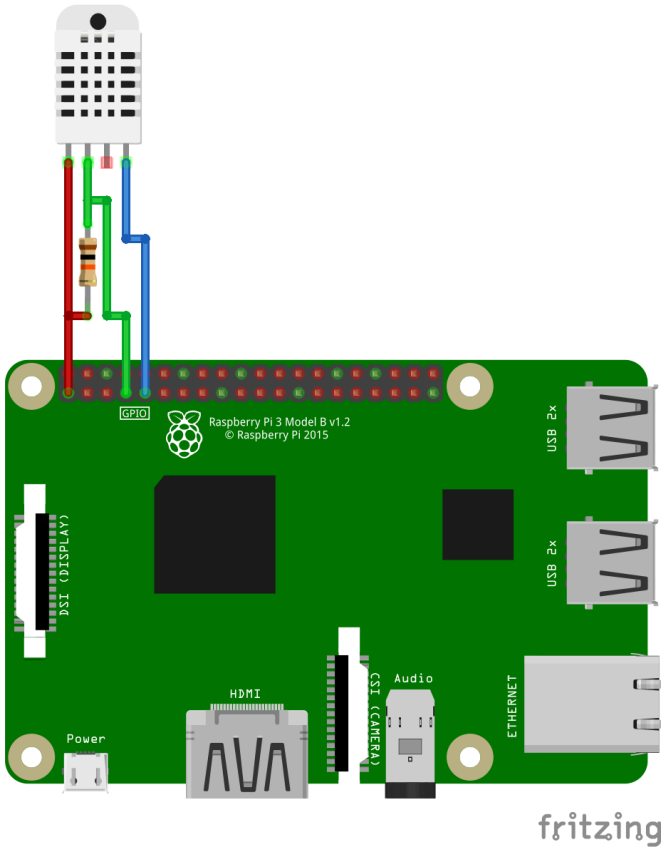


Abbildung 2: Anschlussschema DHT 22 an RaPi

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 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.

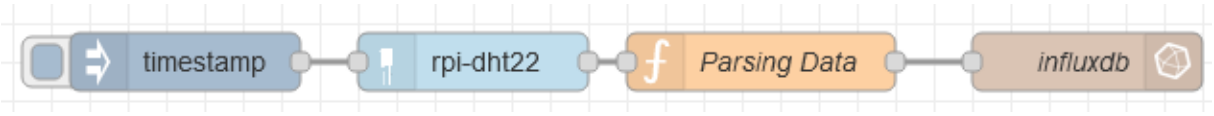


Abbildung 3: Darstellung des Flows zur Abfrage des DHT22

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 Verbindungsbiliothek 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.

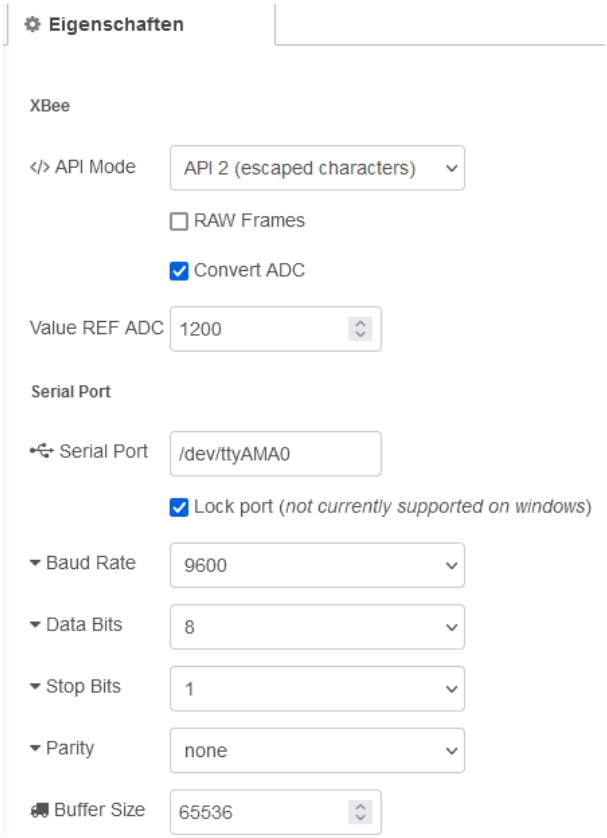
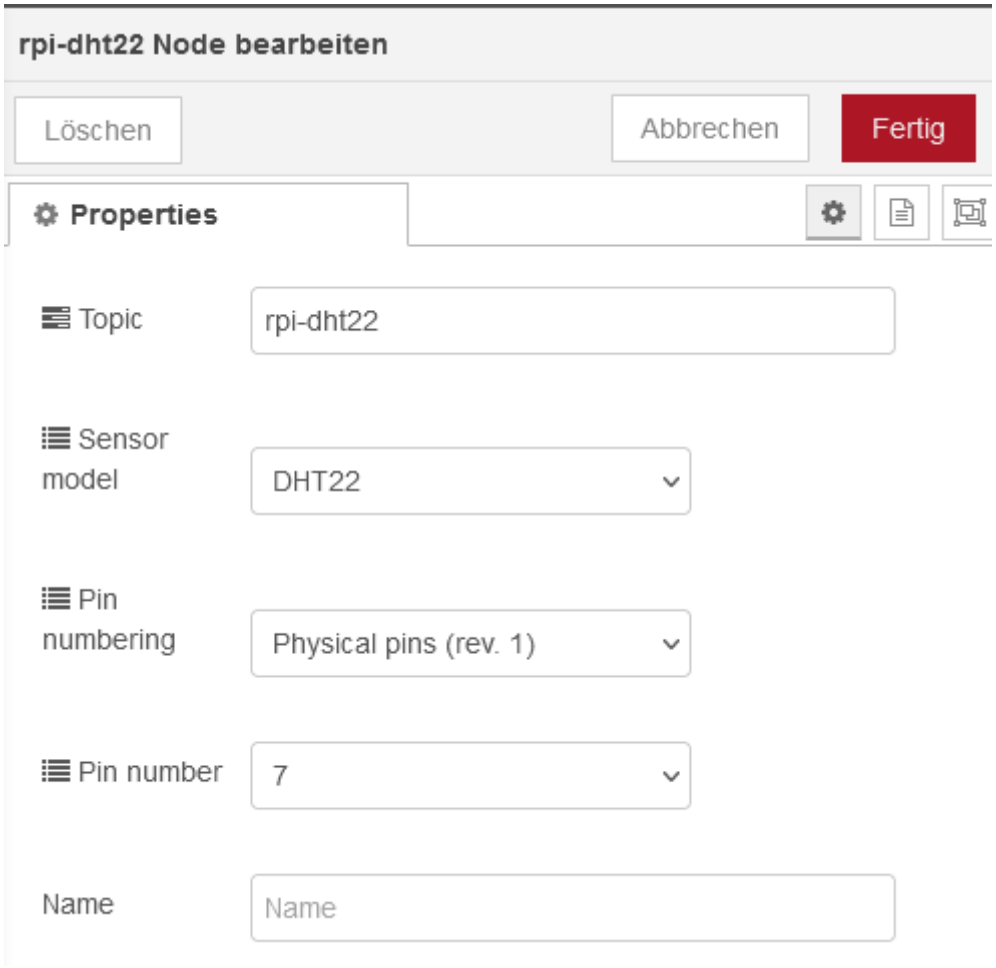


Abbildung 4: Einstellungen des XBee RX Knoten



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

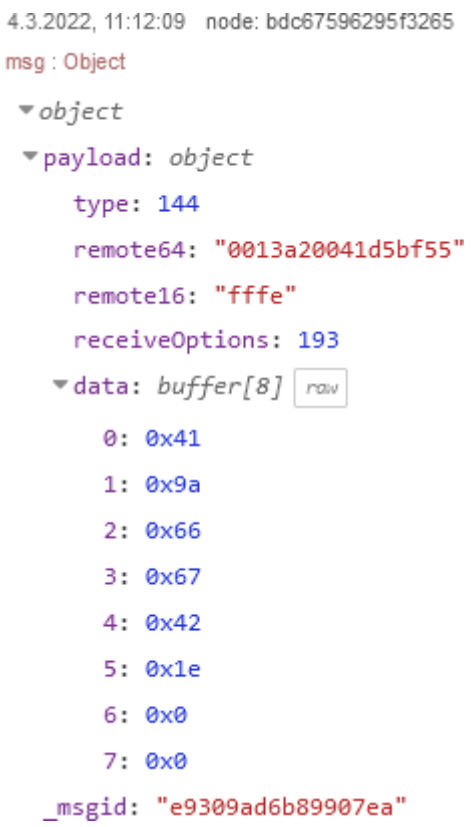


Abbildung 5: Beispielnachricht (JSON) aus dem XBee RX Knoten

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	Datatyp	Measurement	Unit	Influx		Datatyp	Measurement	Unit	Influx
0					0				
1					1				
2					2				
3	float	Temperature	[°C]	Temperature01	3	float	Temperature	[°C]	Temperature02
4					4				
5					5				
6					6				
7	float	Humidity	[%]	Humidity01	7	float	Humidity	[%]	Humidity02
8									
9									
10									
11	float	Pressure	[hPa]	Pressure01					

Abbildung 6: Datenstruktur der Payload

Die Konfiguration des Parsing Knoten erfolgt nach der folgenden Abbildung.

```
var Source = msg.payload.remote64;
msg.measurement ="home";
// Mapping für BME280
if (Source == "0013a20041d5c40a"){
    msg.payload = {
    Temperature01: msg.payload.data.readFloatBE(0),
    Humidity01:  msg.payload.data.readFloatBE(4),
    Pressure01: msg.payload.data.readFloatBE(8),
    };
}
// Mapping für DHT22
if (Source == "0013a20041d5bf55"){
    msg.payload = {
    Temperature02: msg.payload.data.readFloatBE(0),
    Humidity02:  msg.payload.data.readFloatBE(4),
    };
}
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] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
//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);
}
```