temperatursensor-projekt

November 13, 2024

1 Temperatursensor-Projekt

1.1 1. Einführung

1.1.1 Projektziel

Das Ziel dieses Projekts ist es, die Temperatur mithilfe eines BME680-Sensors und eines Raspberry Pi zu messen und die Daten in Jupyter Notebook darzustellen. Dieses Projekt umfasst die Hardware-Schaltung, die Konfiguration des Raspberry Pi, die Programmierung in Python und die Visualisierung der Temperaturdaten. Es soll ein grundlegendes Verständnis für die Verwendung von Temperatursensoren, GPIO-Pins und die Datenverarbeitung in Python vermitteln.

1.1.2 Hintergrund

Der BME680 ist ein Sensor, der Temperatur, Luftfeuchtigkeit, Luftdruck und Luftqualität misst. Er ist besonders nützlich für Umweltüberwachungsprojekte.
Der Raspberry Pi ist ein kostengünstiger Einplatinencomputer, der häufig in Bildungs- und Hobbyprojekten verwendet wird. Durch die Kombination dieser beiden Komponenten können wir ein einfaches und effektives System zur Temperaturmessung und -überwachung erstellen.

1.1.3 Relevanz

Mit der zunehmenden Bedeutung der Umweltüberwachung sind kostengünstige und effektive Lösungen zur Datenerfassung und -analyse erforderlich. Das hier vorgestellte Projekt kann in verschiedenen Bereichen wie Smart Home, Wetterstationen und wissenschaftlichen Untersuchungen eingesetzt werden.

1.2 2. Grundlagen und Theorie

Der BME680-Sensor verwendet die MEMS-Technologie (Micro-Electro-Mechanical Systems), um genaue Messungen von Temperatur, Luftfeuchtigkeit, Luftdruck und Luftqualität durchzuführen. Er kommuniziert über I2C oder SPI mit dem Raspberry Pi, wobei I2C in diesem Projekt verwendet wird. Hier geht es hauptsächlich um den BME680 Sensor, wird aber mit BME280 verglichen.

1.2.1 Was ist MEMS-Technologie ?

MEMS-Technologie in Temperatursensoren kombiniert Mikroelektromechanische Systeme (MEMS) mit traditionellen Temperaturmessmethoden. Diese Technologie ermöglicht es, Temperatursensoren in sehr kleinen, energieeffizienten und kostengünstigen Formaten herzustellen. MEMS-basierte

Temperatursensoren nutzen mikrostrukturierte mechanische und elektrische Komponenten, um Temperaturänderungen präzise zu erfassen und in elektrische Signale umzuwandeln.

1.2.2 Nutzliche Links zu MEMS-Technologie

- Mikro-Elektronisch-Mechanische-Systeme einfach erklärt
- MEMS-Sensoren – Technologie für eine intelligente Welt

1.2.3 Was ist SPI und I2C Schnittstellen ?

Kommunikationsprotokolle, die verwendet werden, um Daten zwischen Mikrocontrollern und verschiedenen Peripheriegeräten wie Sensoren, Displays, Speicherchips und anderen elektronischen Bauteilen auszutauschen. Beide Schnittstellen sind in der Embedded-Entwicklung weit verbreitet, aber sie haben unterschiedliche Eigenschaften und Einsatzgebiete.

- SPI ist ein vollduplexes, synchrone Kommunikationsprotokoll, das häufig verwendet wird, wenn schnelle Datenübertragung und eine direkte Steuerung der Kommunikation zwischen Geräten erforderlich ist.
- I2C ist ein Halbduplex-Kommunikationsprotokoll, das hauptsächlich für die Verbindung von Geräten auf einem Board verwendet wird und besonders gut für die Kommunikation zwischen mehreren Slaves geeignet ist.

1.2.4 Nutzliche Link zu SPI und I2C

- SPI vs. I2C: So wählen Sie das beste Protokoll für Ihre Speicherchips
- SPI vs I2C Communication Protocols

1.3 3. Materialien und Werkzeuge

1.3.1 Zusätzliche Software

- Raspbian OS
- Python 3
- Jupyter Notebook
- Bibliotheken

1.3.2 Zusätzliche Hardware

- Raspberry Pi 5 Model B Rev 1.0
- BME680 und BME280
- Breadboard
- Jumper-Kabel
- Stromversorgung für den Raspberry Pi

1.3.3 Einführung in Jupyter Notebook und Raspberry Pi

- Klicken Sie hier, um ins Einführung-Notebbok zu springen.

1.3.4 Sensoren/Aktoren, inkl. Datenblätter

- Versorgungsspannung: 1.7V bis 3.6V
- Stromverbrauch: 3.1 µA im Forced-Modus (Temperatur und Druckmessung)
- Messbereich: -40°C bis +85°C
- Genauigkeit: ±1.0°C
- Auflösung: 0.01°C

1.3.5 Aufgabe

Wie kann man Versorgungspannung berechnen ?

Lösung der Aufgabe

Die Versorgungsspannung lässt sich druch die Multiplikation von Stromstärke und Widerstand berechnen

1.3.6 Datenblatt

- BME680

1.3.7 Bus

Der BME680-Sensor von Bosch Sensortec unterstützt zwei Kommunikationsschnittstellen: - I2C (bis zu 3.4 MHz) - SPI (bis zu 10 MHz).

Für die meisten einfachen Projekte und speziell bei der Nutzung von mehreren Sensoren gleichzeitig wird häufig die I2C-Schnittstelle bevorzugt, da sie weniger GPIO-Pins benötigt und einfach zu konfigurieren ist.

Übung Welche PINs bieten die Schnittstellen I2C und SPI?
Tipp: Gucken Sie sich das Pinot von Raspberry Pi an.

1.3.8 Pin Beschaltung

- VCC des BME680 an 3.3V des Raspberry Pi
- GND des BME680 an GND des Raspberry Pi
- SCL des BME680 an SCL des Raspberry Pi (GPIO 5)
- SDA des BME680 an SDA des Raspberry Pi (GPIO 3)

- Anschluss des Sensors BME680

1.3.9 Was ist SDA und SCL ?

- SCL (Serial Clock Line) ist die Taktleitung, die vom Master (meistens der Mikrocontroller) bereitgestellt wird. Der Master steuert die Taktfrequenz und synchronisiert so die Übertragung der Daten zwischen den Geräten.
- SDA (Serial Data Line) ist die Datenleitung, auf der sowohl der Master als auch der Slave (in diesem Fall der Temperatursensor) Daten senden und empfangen können. Über diese Leitung werden die Daten seriell bitweise übertragen.
- Weiteres zu SDA und SCL

1.3.10 Zusätzliche Berechnungen

Ein Vorwiderstand ist nicht erforderlich, da der Sensor direkt an den Raspberry Pi angeschlossen wird.

1.3.11 Komponentenauswahl

- **BME680 Sensor:** Die Auswahl des BME680 basiert auf seiner Fähigkeit, mehrere Umweltparameter mit hoher Genauigkeit zu messen.

1.4 4. Schaltungsdesign

1.4.1 Schaltplan

So sieht der einfachste Schaltplan des Sensors in Raspberry Pi aus.

Diese Schaltung wurde durch circuito.io erstellt.

Alternativ
Eine Alternativität bietet sich durch das folgende Bild, das die Schaltung und PINs deutlich und klar zeigt.

Diese Schaltung wurd durch easyeda erstellt.

1.5 5. Implementierung

1.5.1 Hardware-Aufbau

- Verbinden Sie den BME680-Sensor gemäß der oben genannten Pin-Beschaltung mit dem Raspberry Pi.
- Stellen Sie sicher, dass der Raspberry Pi durch an Ihren Rechner angeschlossenes USB mit Strom versorgt wird.

1.5.2 Software-Setup

1. Raspbian OS installieren.
2. Python und Jupyter Notebook installieren.

Zusätzliche für dieses Notebook benötigte Installationen:
smbus, adafruit und matplotlib.

```
[ ]: !pip3 install adafruit-circuitpython-bme680
```

1.5.3 Code

Bibliotheken Installieren In diesem Beispiel wird die `adafruit-circuitpython` und `smbus` benutzt.

Was ist adafruit-circuitpython ? CircuitPython ist eine einfach zu verwendende Programmiersprache, die auf Python basiert und speziell für Mikrocontroller entwickelt wurde. Adafruit hat diese Version von Python entwickelt, um die Programmierung von Mikrocontrollern (z.B. Raspberry Pi, Arduino, ESP32) zu vereinfachen.

- CircuitPython & Python

- Adafruit Library Reference
- Leg los mit CircuitPython
- CircuitPython

Was ist smbus SMBus ist ein einfaches Kommunikationsprotokoll, das auf I2C (Inter-Integrated Circuit) basiert und ursprünglich für die Systemverwaltung in Computern entwickelt wurde. Es ermöglicht die Kommunikation zwischen einem Host und verschiedenen Peripheriegeräten (z.B. Sensoren).

- Was ist ein SM-Bus-Controller?
- System Management Bus

Installation von smbus Führen Sie den folgenden Befehel hier direkt auf Jupyter, um smbus zu installieren.

[ ]:
```
!sudo apt-get install -y python3-pip python3-smbus i2c-tools
```

Installtion von adafruit: Führen Sie den folgenden Befehel hier direkt auf Jupyter, um adafruit zu installieren.

[ ]:
```
!pip3 install adafruit-circuitpython-bme680
```

Installation von board:

Führen Sie den folgenden Befehel hier direkt auf Jupyter, um board zu installieren.

[ ]:
```
!pip3 install adafruit-blinka
```

[ ]:
```
### Bibliotheken importieren
import time
from board import *
import busio
import adafruit_bme680
```

[ ]:
```
# I2C initialisieren
i2c = busio.I2C(SCL, SDA)

# BME680 Sensor initialisieren
bme680 = adafruit_bme680.Adafruit_BME680_I2C(i2c)

# Optional: Offset für die Umgebungstemperatur anpassen
bme680.sea_level_pressure = 1013.25

# Liste zum Speichern der Temperaturwerte
temperaturen = []

# Dauer für die Datensammlung in Sekunden
```

```
duration = 60
start_time = time.time()

while time.time() - start_time < duration:
    # Messe die Temperatur
    bme680.get_sensor_data()
    temperature = bme680.temperature

    # Füge die Temperatur zur Liste hinzu
    temperaturen.append(temperature)

    # Zeige die Temperatur an
    print(f"Temperatur: {temperature:.2f} °C")

    # Warte 2 Sekunden vor der nächsten Messung
    time.sleep(2)
```

Sea Level Pressure (SLP) Der Begriff Meeresspiegeldruck oder Sea Level Pressure (SLP) bezieht sich auf den Luftdruck, der auf Meereshöhe gemessen oder berechnet wird.

- SLP_Erkärung

Append In Python ist append eine Methode, die auf Listen angewendet wird. Sie fügt ein neues Element am Ende einer Liste hinzu.

- Append_Tutorial
- Append_Übung

1.6 6. Experimente und Ergebnisse

1.6.1 Ergebnisse

Darstellung in Matplotlib und Berechnung des Mittelwerts und Ungenauigkeit der BME680 Daten Um die Ergebnisse in Matplotlib darstellen zu können, werden die Bibliotheken **matplotlib** , **pandas** und **statistics** benutzt. Matplotlib und Pandas sind zwei weit verbreitete Python-Bibliotheken, die häufig für Datenanalyse und -visualisierung verwendet werden.

Die Bibliothek statistics bietet eine Sammlung von Funktionen zur Berechnung statistischer Eigenschaften von Daten. Sie ermöglicht es, grundlegende statistische Operationen durchzuführen.

1.6.2 Nutzliche Links zu Matplotlib

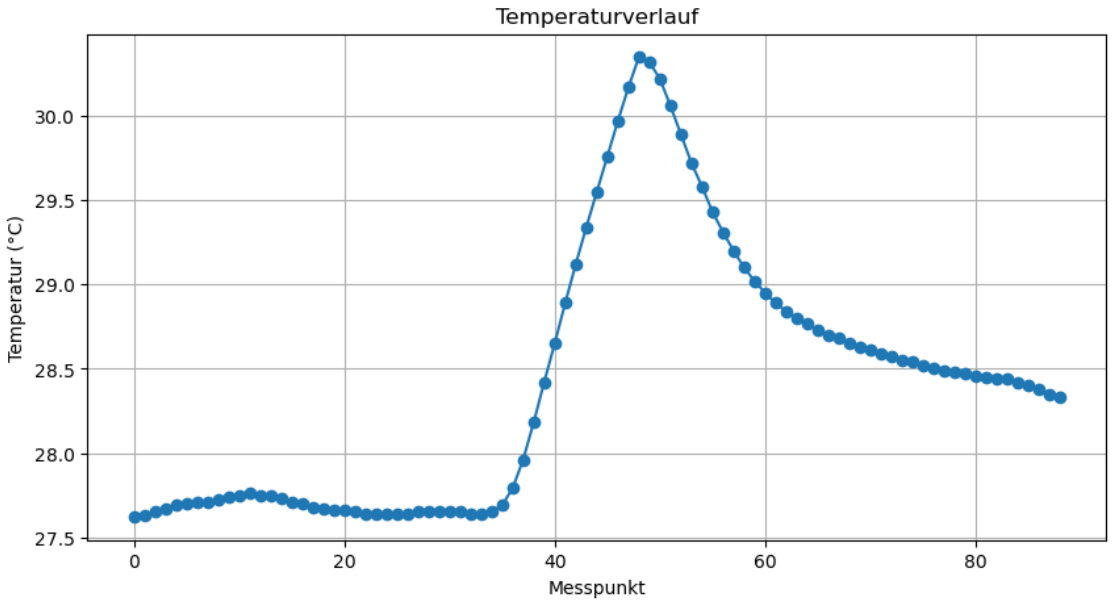
- Pyplot tutorial
- Quick start guide

1.6.3 Nutzliche Links zu Pandas

- Python pandas tutorial
- User Guide

[59]:
```
import matplotlib.pyplot as plt
import time
import statistics

# Live-Plot-Funktion mit Berechnung von Mittelwert und Standardabweichung
def zeichne_live_temperaturverlauf(temperaturen, title='Live␣
↪Temperaturverlauf', xlabel='Messungen', ylabel='Temperatur (°C)'):
    plt.ion()  # Interaktiver Modus für Live-Plotting
    fig, ax = plt.subplots(figsize=(10, 5))

    ax.set_title(title)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
    ax.grid(True)

    # Initiale x- und y-Daten
    x_werte = []
    y_werte = []

    # Durch alle Temperaturwerte iterieren und den Plot sowie Mittelwert und␣
↪Standardabweichung aktualisieren
    for i, temp in enumerate(temperaturen):
        x_werte.append(i)  # Neue x-Werte hinzufügen
        y_werte.append(temp)  # Neue Temperaturwerte hinzufügen

        # Berechnungen: Mittelwert und Standardabweichung
        mean_temperature = statistics.mean(y_werte)
        std_deviation = statistics.stdev(y_werte) if len(y_werte) > 1 else 0  #␣
↪Standardabweichung ab 2 Messungen

        ax.clear()  # Das Diagramm leeren, um es neu zu zeichnen
        ax.plot(x_werte, y_werte, marker='o', linestyle='-', color='b')  # Alle␣
↪Punkte zeichnen

        # Achsen und Titel aktualisieren
        ax.set_title(f"{title}\nMittelwert: {mean_temperature:.2f} °C,␣
↪Standardabweichung: {std_deviation:.2f} °C")
        ax.set_xlabel(xlabel)
        ax.set_ylabel(ylabel)
        ax.grid(True)

        # Plot neu zeichnen
        plt.draw()
        plt.pause(0.5)  # Kurze Pause (z.B. um Sensor-Wartezeit zu simulieren)

    plt.ioff()  # Interaktiven Modus deaktivieren
    plt.show()
```

```
zeichne_live_temperaturverlauf(temperaturen)
```



statistics.mean(): Diese Funktion berechnet den arithmetischen Mittelwert (Durchschnitt) einer Gruppe von Zahlen.

statistics.stdev(): Diese Funktion berechnet die Standardabweichung einer Gruppe von Zahlen. Die Standardabweichung ist ein Maß dafür, wie stark die Werte in einer Datenmenge um den Mittelwert streuen.

Standard_Deviation und Mean

Übung Finden Sie heraus, wie der Mittelwert und die Standardabweichung mathematisch berechnet werden können.

1.6.4 Expriment:

Vergleich zwischen den gemessenen Daten der BME680 und BME280 Sensoren

Installation der Bibliotheken in diesem Versuch wird auch die gleichen Bibliotheken benutzt aber **adafruit-circuitpython-bme280** soll statt **adafruit-circuitpython-bme680** installiert werden.

1.6.5 Code zum Testen

Führen Sie die folgenden Codezellen, um die ganze Funktion zu testen.

[ ]:
```
### Biblliotheken importieren
import time
from board import *
```

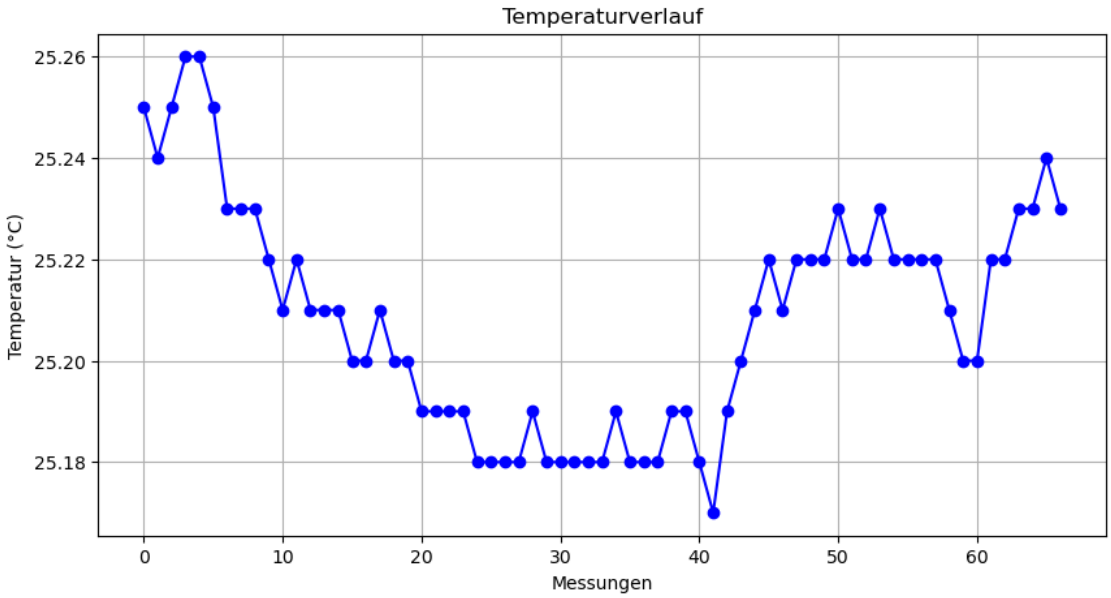
```
import busio
import adafruit_bme280
```

[ ]:
```
# I2C initialisieren
i2c = busio.I2C(board.SCL, board.SDA)

# BME280 Sensor initialisieren
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# Liste zum Speichern der Temperaturwerte
temperature_data = []

# Dauer für die Datensammlung in Sekunden
duration = 30
start_time = time.time()

while time.time() - start_time < duration:
    # Messe die Temperatur
    temperature = bme280.temperature

    # Füge die Temperatur zur Liste hinzu
    temperature_data.append(temperature)

    # Zeige die Temperatur an
    print(f"Temperatur: {temperature:.2f} °C")

    # Warte 2 Sekunden vor der nächsten Messung
    time.sleep(2)
```

1.6.6 Darstellung in Matplotlib

Mit folgendem Code können Sie die durch den Sensor abgelesenen Werte durch Matplotlib darstellen. Führen Sie die folgende Zelle aus und betrachten Sie die Darstellung der Daten.

[24]:
```
import matplotlib.pyplot as plt
import statistics

# Erstelle eine x-Achse basierend auf der Anzahl der Temperaturen
zeichne_live_temperaturverlauf(temperature_data)
```



1.6.7 Übung

Berechnen Sie den Mittelwert und die Ungenauigkeit der BME280 Daten und vergleichen Sie das Ergebnis mit dem Ergebnis von BME680.

1.7 7. Diskussion und Fazit

1.7.1 Diskussion

Die Temperaturmessung mit dem BME680-Sensor und dem Raspberry Pi ist relativ einfach und liefert genaue Ergebnisse. Der Einsatz von Jupyter Notebook ermöglicht eine interaktive und benutzerfreundliche Darstellung der Daten. Eine mögliche Herausforderung könnte die Kalibrierung und Validierung der Messdaten sein, insbesondere in Umgebungen mit extremen Temperaturen. Dieses Projekt zeigt, wie man einen Temperatursensor mit einem Raspberry Pi verwendet. Es bietet eine gute Grundlage, um die GPIO-Pins des Raspberry Pi und die Datenverarbeitung in Python kennenzulernen. Die Ergebnisse sind vielversprechend und das Projekt kann leicht erweitert werden, um komplexere Systeme zur Umweltdatenmessung zu entwickeln.