MICROPYTHON PROGRAMMIERUNGSREFERENZ FÜR DEN FISCHERTECHNIK RX CONTROLLER

Importieren von Bibliotheken

Diese Bibliotheken sind für die Programmierung von Modellen mit dem RX Controller erforderlich.

1. **import asyncio:**
    - **Was ist das?:** asyncio ist eine Python-Bibliothek, die Infrastruktur für das Schreiben von parallel ausführbarem Code bereitstellt.
    - **Wofür ist das gut?:** Die Bibliothek ermöglicht die gleichzeitige und effiziente Ausführung mehrerer Aufgaben (wie das Lesen von Sensoren oder das Aktivieren von Aktoren). Sie ist nützlich in Szenarien, in denen auf verschiedene Ein-/Ausgabeoperationen gewartet werden muss (wie z. B. auf die Antwort eines Sensors), ohne die Ausführung anderer Aufgaben zu blockieren.
2. **import fischertechnik.factories as rx_factory:**
    - **Was ist das?:** Diese Zeile importiert das Modul factories aus der fischertechnik-Bibliothek und weist es dem Alias rx_factory zu, um die Referenzierung zu erleichtern.
    - **Wofür ist das gut?:** Die Bibliothek fischertechnik.factories stellt die erforderlichen Funktionen zur Verfügung, um die verschiedenen Arten von Sensoren und Aktoren, die mit dem RX Controller kompatibel sind, zu erstellen und zu verwalten.
3. **from fischertechnik.logging import log as print:**
    - **Was ist das?:** Diese Zeile importiert die Funktion log aus dem Modul logging von fischertechnik und weist sie dem Alias print zu.
    - **Wofür ist das gut?:** Dadurch kann print verwendet werden, um Debugging- oder Informationsmeldungen zu protokollieren, ähnlich wie die Funktion print() in Python, jedoch speziell für die Fischertechnik-Bibliothek mit zusätzlichen Protokollierungsfunktionen.

Initialisierung der Komponenten

Nach den Importen erfolgt die Initialisierung der Komponenten unter Verwendung verschiedener Funktions. Diese Initialisierungen sind entscheidend, um das System korrekt zu konfigurieren und sicherzustellen, dass jeder Sensor und Aktor betriebsbereit ist. Ohne diese Initialisierungen könnte das Programm nicht mit den Hardwarekomponenten kommunizieren, und Operationen wie das Auslesen von Sensoren oder das Aktivieren von Aktoren wären nicht möglich.

1. **Initialisierung der Factories:**
    - Um Sensoren und Aktoren nutzen zu können, müssen zunächst die entsprechenden Factories initialisiert werden, die das System vorbereiten, um Objekte zu erstellen, die die Hardwarekomponenten repräsentieren.

```
rx_factory.init_controller_factory()
rx_factory.init_input_factory()
rx_factory.init_output_factory()
rx_factory.init_motor_factory()
rx_factory.init_i2c_factory()
```

    - **rx_factory.init_controller_factory():** Initialisiert den Controller, der für die Verwaltung der Kommunikation mit der Steuerelektronik erforderlich ist.
    - **rx_factory.init_input_factory():** Initialisierung, die die Erstellung und Verwaltung von Sensoren (Eingängen) ermöglicht.
    - **rx_factory.init_output_factory():** Initialisierung der Ausgänge, die die Erstellung und Verwaltung von Aktoren (Ausgängen) ermöglicht.
    - **rx_factory.init_motor_factory():** Initialisiert die Motoren und spezialisiert sich auf deren Verwaltung.
    - **rx_factory.init_i2c_factory():** Initialisiert I2C-Geräte, die zur Verwaltung von Sensoren oder Aktoren verwendet werden, die über das I2C-Protokoll verbunden sind.

2. **Initialisierung von Sensoren und Komponenten**

Der fischertechnik RX Controller ermöglicht den Anschluss und die Trennung verschiedener Sensoren und Aktoren an unterschiedlichen Ports und bietet damit große Flexibilität beim Design und Bau von Projekten. Im Folgenden werden die Arten von Sensoren beschrieben, die angeschlossen werden können, wie sie konfiguriert und initialisiert werden, und wofür sie verwendet werden.

- **Fotowiderstand**
    - **Eingangsport:** Jeder digitale Eingangsport.
    - **Initialisierung:**

```
RX_I5_photo_resistor = rx_factory.input_factory.create_photo_resistor(controller, 5)
```

- **Konfiguration:** Dieser Sensor misst die Lichtintensität. Sein Widerstand variiert je nach Lichtmenge; mehr Licht, weniger Widerstand und weniger Licht, mehr Widerstand.
- **Verwendung:** Kann verwendet werden, um Änderungen der Lichtverhältnisse zu erkennen, z.B. zum automatischen Einschalten einer Lampe, wenn es dunkel wird.

- **Ultraschallsensor**
  - **Eingangsport:** Jeder digitale Eingangsport.
  - **Initialisierung:**

    ```
    RX_I7_ultrasonic_distance_meter = rx_factory.input_factory.create_ultrasonic_distance_meter(controller, 7)
    ```

  - **Konfiguration:** Dieser Sensor verwendet Schallwellen, um die Entfernung zu einem Objekt zu messen. Die Entfernung wird durch Messen der Zeit bestimmt, die die Schallwelle benötigt, um das Objekt zu erreichen und zurückzukehren.
  - **Verwendung:** Wird in Anwendungen wie der Vermeidung von Kollisionen bei Robotern oder zur Erkennung der Anwesenheit von Objekten in einem Bereich verwendet.
- **Farbsensor**
  - **Eingangsport:** Jeder digitale Eingangsport.
  - **Initialisierung:**

    ```
    RX_I6_color_sensor = rx_factory.input_factory.create_color_sensor(controller, 6)
    ```

  - **Konfiguration:** Dieser Sensor erkennt die Farbe eines Objekts vor ihm, indem er die Intensität des reflektierten Lichts in den RGB-Farbkanälen (Rot, Grün, Blau) misst.
  - **Verwendung:** Nützlich in Anwendungen, die eine Farberkennung erfordern, wie z.B. bei Sortierprozessen oder der Liniendetektion in der Robotik.
- **NTC-Widerstand**
  - **Eingangsport:** Jeder digitale Eingangsport.
  - **Initialisierung:**

    ```
    RX_I1_ntc_resistor = rx_factory.input_factory.create_ntc_resistor(controller, 1)
    ```

  - **Konfiguration:** Dies ist ein Temperatursensor, der einen NTC-Thermistor (Negative Temperature Coefficient) verwendet, dessen Widerstand bei steigender Temperatur abnimmt.
  - **Verwendung:** Wird zur Messung der Umgebungstemperatur oder der Temperatur von Objekten verwendet.
- **IR-Spursensor**

- **Eingangsport:** Jeder digitale Eingangsport.
- **Initialisierung:**

```
RX_I3_trail_follower = rx_factory.input_factory.create_trail_follower(controller, 3)
RX_I4_trail_follower = rx_factory.input_factory.create_trail_follower(controller, 4)
```

- **Konfiguration:** Wird verwendet, um einer Linie auf einer Oberfläche zu folgen, häufig unter Verwendung von Infrarotlicht zur Erkennung des Kontrasts zwischen der Linie und dem Hintergrund. Beachten Sie, dass ein einzelner Sensor an zwei Eingängen angeschlossen werden muss.
- **Verwendung:** Ideal für Linienfolger-Roboter, die einem vorgegebenen Pfad folgen müssen.

- **Fototransistor**
  - **Eingangsport:** Jeder digitale Eingangsport.
  - **Initialisierung:**

```
RX_I4_photo_transistor = rx_factory.input_factory.create_photo_transistor(controller, 4)
```

  - **Konfiguration:** Dieser Sensor ist lichtempfindlich und ermöglicht die Erkennung der Lichtintensität oder des Vorhandenseins von Licht in einem bestimmten Bereich.
  - **Verwendung:** Kann verwendet werden, um Änderungen der Umgebungsbeleuchtung oder das Vorhandensein von Licht in Sicherheitsanwendungen zu erkennen.

- **Mini-Taster**
  - **Eingangsport:** Jeder digitale Eingangsport.
  - **Initialisierung:**

```
RX_I8_mini_switch = rx_factory.input_factory.create_mini_switch(controller, 8)
```

  - **Konfiguration:** Dies ist ein einfacher Schalter, der erkennen kann, ob er offen oder geschlossen ist.
  - **Verwendung:** Kann als Endschalter zur Erkennung der Anwesenheit oder Abwesenheit eines Objekts oder als Steuerknopf verwendet werden.

- **Reed-Kontakt**
  - **Eingangsport:** Jeder digitale Eingangsport.
  - **Initialisierung:**

```
RX_I2_reed_switch = rx_factory.input_factory.create_mini_switch(controller, 2)
```

  - **Konfiguration:** Ein magnetischer Sensor, der bei Vorhandensein eines Magnetfelds aktiviert wird und anzeigt, ob das Feld vorhanden oder abwesend ist.
  - **Verwendung:** Wird häufig in Sicherheitssystemen oder zur Erkennung der Position von beweglichen Objekten verwendet.

- **I2C-Sensoren:** Sensoren, die über das I2C-Protokoll angeschlossen sind, erfordern eine spezielle Initialisierung, da dieses Protokoll die Kommunikation mit mehreren Geräten über einen einzigen Datenbus ermöglicht.
    - **RGB-Gestensensor**
        - **I2C-Port:** Kann an jeden verfügbaren I2C-Port angeschlossen werden.

        RX_I2C_1_gesture_sensor = rx_factory.i2c_factory.create_gesture_sensor(controller, 1)
        - **Konfiguration:** Dieser Sensor kann Bewegungen wie Winken oder Handbewegungen erkennen. Er ist auch in der Lage, Entfernungen, Lichtintensität und dessen Zusammensetzung zu messen.
        - **Verwendung:** Wird zur berührungslosen Steuerung von Geräten verwendet, wie in Benutzeroberflächen oder Robotersteuerungen.
    - **Umweltsensor**
        - **I2C-Port:** Kann an jeden verfügbaren I2C-Port angeschlossen werden.
        - **Initialisierung:**

            RX_I2C_2_environment_sensor = rx_factory.i2c_factory.create_environment_sensor(controller, 2)
        - **Konfiguration:** Misst Umgebungsparameter wie Temperatur, Luftfeuchtigkeit, Luftqualität und Luftdruck.
        - **Verwendung:** Ideal zur Überwachung der Umgebungsbedingungen in einem bestimmten Bereich.
    - **Kombisensor 10-polig**
        - **I2C-Port**: Kann an jeden verfügbaren I2C-Port angeschlossen werden.
        - **Initialisierung:**

            RX_I2C_3_combined_sensor = rx_factory.i2c_factory.create_combined_sensor(controller, 3)
        - **Konfiguration:** Beinhaltet mehrere Sensoren wie Beschleunigungsmesser, Gyroskop und Magnetometer, die die Erkennung von Bewegung, Orientierung und Magnetfeld ermöglichen.
        - **Verwendung:** Wird in Anwendungen zur Bewegungserkennung oder Stabilisierung verwendet.

Zusammenfassung: Für jeden Sensortyp muss der richtige Port basierend auf der Anwendungskonfiguration ausgewählt und der Sensor mithilfe der bereitgestellten Factories initialisiert werden. Diese Anfangskonfiguration ist wesentlich, um sicherzustellen, dass die Sensoren korrekt funktionieren und genaue Daten liefern. Im nächsten Teil werden wir behandeln, wie diese Sensoren verwendet und ihre Daten interpretiert werden.

Detaillierte Beschreibung der Informationen, die von digitalen und analogen Sensoren im Fischertechnik RX Controller gewonnen werden können

Die an den fischertechnik RX Controller angeschlossenen Sensoren können je nach Art des generierten Signals und der bereitgestellten Informationen in digitale und analoge Sensoren eingeteilt werden. Im Folgenden wird beschrieben, welche Informationen von diesen Sensoren gewonnen werden können und wie diese Daten genutzt werden können.

Digitale Sensoren: Digitale Sensoren liefern diskrete Werte, normalerweise "ein" oder "aus" (1 oder 0), "offen" oder "geschlossen" oder eine andere Form von binärem Wert.

- **Mini-Taster**
  - **Gewonnene Informationen:**
    - **is_open():** Gibt True zurück, wenn der Schalter geöffnet ist, False wenn er geschlossen ist.
    - **is_closed():** Gibt True zurück, wenn der Schalter geschlossen ist, False wenn er geöffnet ist.
    - **get_state():** Gibt den aktuellen Zustand des Schalters zurück (offen oder geschlossen).
  - **Verwendung:** Kann verwendet werden, um die Anwesenheit eines Objekts zu erkennen oder als Steuertaste in einem System, z.B. um einen Prozess zu starten oder zu stoppen.
- **Reed-Kontakt**
  - **Gewonnene Informatione**n:
    - **is_open():** Gibt True zurück, wenn der Schalter geöffnet ist, False wenn er geschlossen ist.
    - **is_closed():** Gibt True zurück, wenn der Schalter geschlossen ist, False wenn er geöffnet ist.
    - **get_state():** Gibt den aktuellen Zustand des Schalters zurück (offen oder geschlossen).
  - **Verwendung:** Wird häufig verwendet, um das Vorhandensein eines Magnetfeldes zu erkennen, nützlich in Sicherheitsanwendungen oder zur Positionsbestimmung.
- **IR-Spursensor**
  - **Gewonnene Informationen:**
    - **get_state():** Gibt einen Wert zurück, der angibt, ob der Sensor die Linie erkennt (0 für nicht erkannt, 1 für erkannt).
  - **Verwendung:** Wichtig für Linienfolger-Roboter, die einer vorgegebenen Linie auf einer Oberfläche folgen müssen.

- **Photo Transistor**
  - **Gewonnene Informationen:**
    - **is_bright():** Gibt True zurück, wenn das erkannte Licht hoch ist.
    - **is_dark():** Gibt True zurück, wenn das erkannte Licht niedrig ist.
    - **get_state():** Gibt den aktuellen Zustand basierend auf dem erkannten Licht (hell oder dunkel) zurück.
  - **Verwendung:** Kann verwendet werden, um das Vorhandensein von Licht oder Dunkelheit in einem Bereich zu erkennen, nützlich bei der Tag/Nacht-Erkennung oder der Objekterkennung.

Analoge Sensoren: Analoge Sensoren liefern kontinuierliche Werte, die innerhalb eines bestimmten Bereichs variieren können und physikalische Größen wie Licht, Temperatur, Entfernung usw. darstellen.

- **Fotowiderstand**
  - **Gewonnene Informationen:**
    - **get_resistance():** Gibt den Widerstandswert zurück, der umgekehrt proportional zur Lichtintensität ist (mehr Licht, weniger Widerstand).
  - **Verwendung:** Wird verwendet, um die Umgebungslichtintensität zu messen, z.B. zur automatischen Steuerung von Beleuchtungssystemen.
- **Ultraschallsensor**
  - **Gewonnene Informationen:**
    - **get_distance():** Gibt die Entfernung zu einem Objekt in Einheiten von 0 bis 1024 zurück, wobei 0 die minimale und 1024 die maximale Entfernung darstellt.
  - **Verwendung:** Wird zur Messung der Entfernung zu Objekten verwendet, nützlich in Anwendungen wie Hindernisvermeidung bei Robotern.
- **Farbsensor**
  - **Gewonnene Informationen:**
    - **get_voltage():** Gibt den Spannungswert zurück, der mit der Intensität der erkannten RGB-Farbkomponenten korreliert.
  - **Verwendung:** Ermöglicht die Farberkennung von Objekten, nützlich bei Sortierprozessen oder der Erkennung farbiger Linien.

- **NTC Resistor**
  - **Gewonnene Informationen:**
    - **get_resistance():** Gibt den Widerstand des Thermistors zurück, der umgekehrt zur Temperatur variiert (mehr Temperatur, weniger Widerstand).
  - **Verwendung:** Wird zur Messung der Umgebungstemperatur oder von Objekten verwendet, geeignet für Anwendungen zur thermischen Überwachung.

Zusammenfassung: Digitale und analoge Sensoren liefern eine Vielzahl von Daten, die zur Interaktion mit der Umgebung, zur Steuerung automatischer Systeme oder zur Datenerfassung für Analysen verwendet werden können. Digitale Sensoren liefern binäre Werte, die für einfache Steuerentscheidungen nützlich sind, während analoge Sensoren kontinuierliche Werte liefern, die eine präzise Messung physikalischer Größen ermöglichen. Die korrekte Interpretation dieser Daten ist entscheidend für das effiziente Funktionieren jedes auf dem fischertechnik RX Controller basierenden Systems.

RGB-Gestensensor

Der RGB-Gestensensor ist ein fortschrittliches Gerät, das mehrere Funktionen in einem einzigen Chip vereint. Es kann Gesten erkennen sowie Umgebungslicht, RGB-Licht (rot, grün, blau) und die Nähe messen. Nachfolgend finden Sie eine detaillierte Beschreibung der Fähigkeiten des Sensors, der verfügbaren Funktionen und der abrufbaren Werte.

Initialisierung des Sensors: Um den RGB-Gestensensor zu verwenden, muss er zunächst korrekt initialisiert werden. Dies geschieht durch die Erstellung einer Instanz des Sensors über die I2C-Fabrik.

```
RX_I2C_1_gesture_sensor = rx_factory.i2c_factory.create_gesture_sensor(controller, 1)
```

Dieser Code erstellt das Objekt RX_I2C_1_gesture_sensor, das dem I2C-Port 1 zugeordnet ist. Nach der Initialisierung kann mit der Konfiguration der verschiedenen Funktionen begonnen und Daten vom Sensor abgerufen werden.

Gestenerkennung: Der RGB-Gestensensor kann verschiedene Handbewegungen wie Auf-, Ab-, Links- und Rechtsbewegungen erkennen. Diese Funktion ist nützlich zur Steuerung von Geräten ohne physischen Kontakt.

- **Aktivierung der Gestenerkennung:**

  ```
  RX_I2C_1_gesture_sensor.enable_gesture()
  ```

- **Abrufen erkannter Gesten:**

  ```
  gesture = RX_I2C_1_gesture_sensor.get_gesture()
  print(gesture)
  ```

- **Deaktivierung der Gestenerkennung:**

```
RX_I2C_1_gesture_sensor.disable_gesture()
```

Messung von RGB-Licht: Der Sensor kann die Intensität des Lichts in den RGB-Farbkomponenten erkennen und so eine genaue Darstellung der Umgebungslichtfarbe liefern.

- **Aktivierung des Lichtsensors:**

```
RX_I2C_1_gesture_sensor.enable_light()
```

- **Abrufen von RGB-Daten:**

```
rgb_values = RX_I2C_1_gesture_sensor.get_rgb()
print(f"Rot: {rgb_values[0]}, Grün: {rgb_values[1]}, Blau: {rgb_values[2]}")
```

  - **get_rgb():** Gibt ein Tupel mit den Intensitätswerten für die Komponenten Rot, Grün und Blau zurück. Jeder Wert kann normalerweise von 0 bis 255 variieren, abhängig von der Sensoreinstellung und den Lichtbedingungen.

- **Abrufen von HEX-Werten:**

```
hex_value = RX_I2C_1_gesture_sensor.get_hex()
print(f"Farbe im HEX-Format: {hex_value}")
```

  - **get_hex()**: Gibt eine HEX-Darstellung der erkannten Farbe zurück, nützlich für Anwendungen, die einen Farbidentifikator benötigen.

- **Abrufen von HSV-Werten:**

```
hsv_values = RX_I2C_1_gesture_sensor.get_hsv()
print(f"Farbton: {hsv_values[0]}, Sättigung: {hsv_values[1]}, Wert: {hsv_values[2]}")
```

  - **get_hsv():** Liefert die Werte für Farbton (Hue), Sättigung und Wert (Helligkeit), die nützlich sind zur Farbbeschreibung in grafischen und Designanwendungen.

- **Deaktivierung des Lichtsensors:**

```
RX_I2C_1_gesture_sensor.disable_light()
```

Messung des Umgebungslichts: Neben den RGB-Messungen kann der Sensor eine Messung des Umgebungslichts liefern, was nützlich ist, um die allgemeinen Lichtverhältnisse zu bestimmen.

- **Abrufen des Umgebungslichts:**

```
ambient_light = RX_I2C_1_gesture_sensor.get_ambient()
print(f"Umgebungslicht: {ambient_light}")
```

  - **get_ambient():** Gibt einen Wert zurück, der die Intensität des Umgebungslichts repräsentiert. Dieser Wert kann verwendet werden, um die Helligkeit von Displays oder Beleuchtung automatisch anzupassen.

Nähe-Erkennung: Der RGB-Gestensensor kann die Nähe von Objekten basierend auf der Menge des reflektierten Infrarotlichts erkennen.

- **Aktivierung der Nähe-Erkennung:**

```
RX_I2C_1_gesture_sensor.enable_proximity()
```

- **Abrufen von Näherungsdaten:**

```
proximity = RX_I2C_1_gesture_sensor.get_proximity()
print(f"Proximität: {proximity}")
```

- **get_proximity():** Gibt einen Wert zwischen 0 und 255 zurück, wobei 0 bedeutet, dass sich kein Objekt in der Nähe befindet und 255, dass das Objekt sehr nah am Sensor ist.

- **Deaktivierung der Nähe-Erkennung:**

```
RX_I2C_1_gesture_sensor.disable_proximity()
```

Zusammenfassung: Der RGB-Gestensensor ist ein vielseitiges Gerät, das die Gestenerkennung, die Messung von RGB-Licht und Umgebungslicht sowie die Nähe-Erkennung in einem einzigen Chip kombiniert. Dies macht ihn ideal für Anwendungen mit fortschrittlichen Benutzeroberflächen, automatischer Lichtanpassung und Sicherheitssystemen. Der Sensor bietet eine reiche Schnittstelle zur Interaktion mit der Umgebung durch detaillierte und präzise Daten.

Umweltsensor

Der BME680 ist ein integrierter Umweltsensor, der verschiedene Parameter wie Temperatur, relative Luftfeuchtigkeit, Luftdruck und Luftqualität messen kann. Diese Eigenschaften machen ihn ideal für Anwendungen zur Umweltüberwachung, HVAC-Systeme, Wetterstationen und andere IoT-Geräte (Internet der Dinge). Im Folgenden werden die Fähigkeiten des Sensors, seine Initialisierung und die Erfassung der verschiedenen Daten beschrieben.

Initialisierung des Umweltsensors: Um den Umweltsensor zu verwenden, muss er zunächst korrekt initialisiert werden. Dies geschieht durch die Erstellung einer Instanz des Sensors über die I2C-Fabrik.

```
RX_I2C_2_environment_sensor = rx_factory.i2c_factory.create_environment_sensor(controller, 2)
```

Dieser Code erstellt das Objekt RX_I2C_2_environment_sensor, das dem I2C-Port 2 zugeordnet ist. Nach der Initialisierung können die vom Sensor bereitgestellten Umweltdaten abgerufen werden.

Funktionen des Sensors

- **Temperaturmessung**
  - **Abrufen der Temperatur:**

```
temperatur = RX_I2C_2_environment_sensor.get_temperature()
print(f"Temperatur: {temperatur} °C")
```

    - **get_temperature():** Gibt die Umgebungstemperatur in Grad Celsius (°C) zurück. Diese Information ist nützlich für die Überwachung des Innen- oder Außenklimas sowie für Anwendungen, die eine präzise Temperaturregelung erfordern.

- **Messung der relativen Luftfeuchtigkeit**
  - **Abrufen der relativen Luftfeuchtigkeit:**

```
feuchtigkeit = RX_I2C_2_environment_sensor.get_humidity()
print(f"Relative Luftfeuchtigkeit: {feuchtigkeit} %")
```

- **get_humidity():** Liefert die relative Luftfeuchtigkeit in Prozent (%). Diese Messung ist wichtig für die Kontrolle der Luftqualität, die Vermeidung übermäßiger Trockenheit oder hoher Luftfeuchtigkeit und für Anwendungen im Komfort- und Gesundheitsbereich.

- **Messung des Luftdrucks**
  - **Abrufen des Luftdrucks:**

```
druck = RX_I2C_2_environment_sensor.get_pressure()
print(f"Luftdruck: {druck} hPa")
```

- **get_pressure():** Gibt den Luftdruck in Hektopascal (hPa) zurück. Diese Daten sind in meteorologischen Anwendungen unerlässlich, da sie helfen, Wetteränderungen wie Klima und Stürme vorherzusagen.

- **Messung der Luftqualität (IAQ):** Der Umweltsensor kann die Luftqualität in Innenräumen (IAQ) durch die Erkennung flüchtiger organischer Verbindungen (VOC) in der Luft messen.
  - **Abrufen der Luftqualität als Zahl:**

```
iaq_number = RX_I2C_2_environment_sensor.get_indoor_air_quality_as_number()
print(f"Luftqualität (IAQ): {iaq_number}")
```

- **get_indoor_air_quality_as_number():** Gibt eine Zahl zurück, die die Luftqualität in Innenräumen darstellt. Dieser Wert kann interpretiert werden, um die allgemeine Luftqualität zu bestimmen und Maßnahmen zu ergreifen, falls eine Verbesserung erforderlich ist.

  - **Abrufen der Luftqualität als Text:**

```
iaq_text = RX_I2C_2_environment_sensor.get_indoor_air_quality_as_text()
print(f"Luftqualität: {iaq_text}")
```

- **get_indoor_air_quality_as_text():** Liefert eine textliche Beschreibung der Luftqualität (z.B. "Gut", "Mäßig", "Schlecht" usw.), was die Interpretation der Ergebnisse für Endnutzer erleichtert.

Kalibrierung des Sensors: Der Umweltsensor kann eine Kalibrierungsphase erfordern, um die Genauigkeit der Messungen der Luftqualität zu gewährleisten.

- **Kalibrierung des Sensors:**

```
RX_I2C_2_environment_sensor.calibrate()
```

  - **calibrate():** Dieser Befehl startet die Kalibrierung des Sensors, um sicherzustellen, dass die Messungen der Luftqualität präzise sind.

- **Überprüfung der Notwendigkeit einer Kalibrierung:**

```
needs_calibration = RX_I2C_2_environment_sensor.needs_calibration()
if needs_calibration:
    print("Der Sensor benötigt eine Kalibrierung.")
```

- **needs_calibration():** Gibt einen booleschen Wert zurück, der anzeigt, ob der Sensor eine Kalibrierung benötigt.

Zusammenfassung: Der Umweltsensor bietet ein umfassendes Spektrum an Umweltmessungen, darunter Temperatur, Luftfeuchtigkeit, Luftdruck und Luftqualität. Diese Daten sind für eine Vielzahl von Anwendungen von entscheidender Bedeutung, von der Umweltüberwachung bis hin zu intelligenten HVAC-Systemen. Die Fähigkeit, die Luftqualität zu messen, ist besonders wichtig in Umgebungen, in denen die Luftqualität in Innenräumen die Gesundheit und das Wohlbefinden beeinträchtigen kann.

Kombisensor

Der Kombisensor, der einen Beschleunigungsmesser, ein Gyroskop und ein Magnetometer umfasst. Dieses Gerät liefert Messungen zu Bewegung, Orientierung und Magnetfeld und eignet sich ideal für Anwendungen in der Robotik, Bewegungserfassung, Stabilisierung von Geräten und Navigation. Nachfolgend finden Sie eine detaillierte Beschreibung der Funktionen, Initialisierung und Datenerfassung dieses Sensors.

Initialisierung Kombisensors Um den Kombisensor zu verwenden, müssen alle seine Komponenten (Beschleunigungsmesser, Magnetometer, Gyroskop) mit den gewünschten Einstellungen initialisiert werden.

```
controller.RX_I2C_3_combined_sensor.init_accelerometer(2, 8, False)
controller.RX_I2C_3_combined_sensor.init_magnetometer(25)
controller.RX_I2C_3_combined_sensor.init_gyrometer(250, 12, False)
```

- **Beschleunigungsmesser:**
  - **init_accelerometer(range, bandwidth, high_res):**
    - **range:** Definiert den Messbereich der Beschleunigung. In diesem Fall gibt 2 einen Bereich von ±2g an.
    - **bandwidth:** Definiert die Bandbreite des Ausgangsfilters. Hier gibt 8 die Bandbreite in Hz an.
    - **high_res:** Ein boolescher Wert, der angibt, ob der Hochauflösungsmodus verwendet wird.

- **Magnetometer:**
  - **init_magnetometer(data_rate):**
    - **data_rate**: Definiert die Datenrate (in Hz) für das Magnetometer. 25 gibt eine Datenrate von 25 Hz an.
- **Gyroskop:**
  - **init_gyrometer(range, bandwidth, high_res):**

- **range:** Definiert den Messbereich der Rotationsgeschwindigkeit. Hier gibt 250 einen Bereich von ±250 °/s an.
- **bandwidth:** Definiert die Bandbreite des Ausgangsfilters. 12 gibt die Bandbreite in Hz an.
- **high_res:** Ein boolescher Wert, der angibt, ob der Hochauflösungsmodus verwendet wird.

Datenerfassung des Kombisensor

1. **Beschleunigungsmesser:** Der Beschleunigungsmesser misst die lineare Beschleunigung in den drei Richtungen (x, y, z), was nützlich ist, um die relative Position, Geschwindigkeit und Bewegungserkennung zu bestimmen.
   - **Abrufen der Beschleunigung:**

```
acc_x = controller.RX_I2C_3_combined_sensor.get_acceleration_x()
acc_y = controller.RX_I2C_3_combined_sensor.get_acceleration_y()
acc_z = controller.RX_I2C_3_combined_sensor.get_acceleration_z()
print(f"Beschleunigung - X: {acc_x} g, Y: {acc_y} g, Z: {acc_z} g")
```

     - **get_acceleration_x():** Gibt die Beschleunigung auf der X-Achse in Einheiten von g zurück.
     - **get_acceleration_y():** Gibt die Beschleunigung auf der Y-Achse in Einheiten von g zurück.
     - **get_acceleration_z():** Gibt die Beschleunigung auf der Z-Achse in Einheiten von g zurück.
2. **Magnetometer:** Das Magnetometer misst die Stärke des Magnetfelds in den drei Richtungen und hilft, die Orientierung in Bezug auf das Erdmagnetfeld zu bestimmen.
   - **Abrufen des Magnetfelds:**

```
mag_x = controller.RX_I2C_3_combined_sensor.get_magnetic_field_x()
mag_y = controller.RX_I2C_3_combined_sensor.get_magnetic_field_y()
mag_z = controller.RX_I2C_3_combined_sensor.get_magnetic_field_z()
print(f"Magnetfeld - X: {mag_x} µT, Y: {mag_y} µT, Z: {mag_z} µT")
```

     - **get_magnetic_field_x():** Gibt die Stärke des Magnetfelds auf der X-Achse in Mikrotesla (µT) zurück.
     - **get_magnetic_field_y():** Gibt die Stärke des Magnetfelds auf der Y-Achse in Mikrotesla (µT) zurück.
     - **get_magnetic_field_z():** Gibt die Stärke des Magnetfelds auf der Z-Achse in Mikrotesla (µT) zurück.
3. **Gyroskop:** Das Gyroskop misst die Rotationsgeschwindigkeit in den drei Richtungen und liefert wesentliche Daten zur Steuerung von Orientierung und Stabilisierung.

- **Abrufen der Rotationsgeschwindigkeit:**

```
rot_x = controller.RX_I2C_3_combined_sensor.get_rotation_x()
rot_y = controller.RX_I2C_3_combined_sensor.get_rotation_y()
rot_z = controller.RX_I2C_3_combined_sensor.get_rotation_z()
print(f"Rotationsgeschwindigkeit - X: {rot_x} °/s, Y: {rot_y} °/s, Z: {rot_z} °/s")
```

  - **get_rotation_x():** Gibt die Rotationsgeschwindigkeit auf der X-Achse in Grad pro Sekunde (°/s) zurück.
  - **get_rotation_y():** Gibt die Rotationsgeschwindigkeit auf der Y-Achse in Grad pro Sekunde (°/s) zurück.
  - **get_rotation_z():** Gibt die Rotationsgeschwindigkeit auf der Z-Achse in Grad pro Sekunde (°/s) zurück.

Zusammenfassung: Der Kombisensor bietet eine umfassende Lösung zur Messung von Bewegung, Orientierung und Magnetfeld. Mit Fähigkeiten zur Messung von Beschleunigung, Magnetfeld und Rotationsgeschwindigkeit ist dieser Sensor äußerst nützlich für Anwendungen in der Robotik, Navigation, Stabilisierung von Geräten und anderen Bewegungskontrollsystemen. Jede der erfassten Messungen ermöglicht ein detailliertes Verständnis des Zustands und der Bewegung des angeschlossenen Geräts und erleichtert eine breite Palette von Anwendungen, von der Sturzerkennung bis zur Positions- und Orientierungserfassung.