debo-sens-9axis-sensor-notebook

November 13, 2024

1 DEBO SENS 9AXIS Sensor Steuerung mit Raspberry Pi und Jupyter Notebook

1.1 1. Einführung

Was ist der DEBO SENS 9AXIS Sensor?

Der DEBO SENS 9AXIS Sensor ist ein hochpräziser Sensor, der zur Messung der Beschleunigung und der Winkelgeschwindigkeit in drei Dimensionen konzipiert ist. Er kombiniert einen 3-Achsen-Beschleunigungssensor und einen 3-Achsen-Gyroskopsensor in einem einzigen Modul. Diese Sensoren sind in der Robotik, in der Fahrzeugnavigation und in der Bewegungserfassung sehr nützlich. Der Sensor kann Bewegungen und Neigungen in Echtzeit verfolgen, was ihn ideal für Anwendungen in der Spieleentwicklung, Augmented Reality (AR) und der Überwachung von Bewegungen macht.

Führen Sie die folgenden zwei Zellen, um Videos zu 9AXIS Sensor in einem neuen Browser-Tab zu öffnen.
Hinweis: Beide Videos sind auf Englisch

[16]:
```
import webbrowser
url = "https://youtu.be/a37xWuNJsQI?feature=shared"
webbrowser.open_new_tab(url)
```

[16]:
```
True
```

[15]:
```
url = "https://youtu.be/ciX3L3nnNHg?feature=shared"
webbrowser.open_new_tab(url)
```

1.1.1 Projektziel

In dieser Einführung erfahren Sie, wie Sie den DEBO SENS 9AXIS Sensor mit einem Raspberry Pi steuern und die gesammelten Daten in einem Jupyter Notebook analysieren können. Wir werden die grundlegenden Schritte durchgehen, um den Sensor korrekt anzuschließen, die erforderlichen Bibliotheken zu installieren und ein einfaches Programm zu schreiben, um die Sensordaten zu erfassen und darzustellen.

1.1.2 Relevanz

Dieses Projekt kombiniert Grundlagen der Elektronik und Programmierung mit Hardware-Steuerung und gibt einen praxisorientierten Einstieg in das Auslesen von Sensordaten.

1.1.3 Ressourcen

Quellen für den Sensor:
- DEBO SENS 9AXIS Sensor
- Raspberry Pi

1.2 2. Grundlagen und Theorie

1.2.1 DEBO SENS 9AXIS Sensor

Der DEBO SENS 9AXIS Sensor kombiniert Beschleunigungsmesser, Gyroskop und Magnetometer in einem Chip. Dieser Sensor wird für Bewegungserkennung, Lagebestimmung und Schrittmessung eingesetzt.

- **Beschleunigungsmesser:** Misst die Beschleunigung in x, y und z Richtung.
- **Gyroskop:** Misst die Winkelgeschwindigkeit in drei Achsen.
- **Magnetometer:** Misst das Magnetfeld in drei Dimensionen.

1.2.2 I2C Protokoll

Der DEBO SENS 9AXIS verwendet das I2C-Kommunikationsprotokoll zur Datenübertragung. Für weiteres gucken Sie sich dieses Notebook

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 Software & Hardware

Software	Hardware
Raspbian OS	Raspberry Pi 5 Model B
Python 3	DEBO SENS 9AXIS Sensor
Jupyter Notebook	Jumper-Kabel

1.3.2 Datenblätter

- DEBO SENS 9AXIS Sensor

1.4 4. Schaltungsdesign

1.4.1 Verdrahtung

Sensor-Pin	Raspberry Pi GPIO Pin	Funktion
SDA	GPIO2 (SDA)	Datenleitung
SCL	GPIO3 (SCL)	Taktsignal
VCC	3.3V	Stromversorgung
GND	Ground	Masse

Zur Verdeutlichung gucken Sie sich das Pinout-Bild

1.4.2 Aussehen des Sensors

Das folgende Bild stellt dar, wie der Sensor aussieht.

1.4.3 Aussehen der Schaltung

Der folgende Schaltungsplan zeigt, wie man den Sensor mit RaspPi verbindet.

1.5 5. Implementierung

1.5.1 Python Bibliotheken

Um den DEBO SENS 9AXIS Sensor anzusprechen, werden folgende Bibliotheken benötigt: - `smbus` - `matplotlib`
Falls noch nicht geschehen, installieren Sie die Bibliothek `smbus` mit:

[11]:
```
!pip3 install smbus2
```

```
Requirement already satisfied: smbus2 in c:\python312\lib\site-packages (0.4.3)


[notice] A new release of pip is available: 23.2.1 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Für die Installation von `Matplotlib` gucken Sie sich dieses Notebook.

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

[ ]:
```
import time  # Importiert die Zeitbibliothek, um Zeitfunktionen zu nutzen
from smbus2 import SMBus  # Importiert die SMBus-Klasse für die␣
 ↪I2C-Kommunikation
import matplotlib.pyplot as plt  # Importiert matplotlib für das Zeichnen von␣
 ↪Plots
```

[7]:
```
# Definiert die Adresse des Geräts und die Register für die Beschleunigungsdaten
DEVICE_ADDRESS = 0x68  # Beispieladresse des Beschleunigungssensors (kann␣
 ↪variieren)
ACCEL_X_REG = 0x3B  # Registeradresse für die X-Beschleunigungsdaten
ACCEL_Y_REG = 0x3D  # Registeradresse für die Y-Beschleunigungsdaten
ACCEL_Z_REG = 0x3F  # Registeradresse für die Z-Beschleunigungsdaten
```

[ ]:
```
# Öffnet den I2C-Bus (Standardbus für Raspberry Pi)
bus = SMBus(1)  # Bus 1 wird für die meisten Raspberry Pi-Modelle verwendet
```

[ ]:
```
# Funktion zur Kalibrierung des Sensors
def calibrate_sensor(samples=100):
    # Initialisiert die Offset-Werte für jede Achse
    x_offset = 0
    y_offset = 0
    z_offset = 0
    # Führt eine Schleife aus, um mehrere Messungen zur Kalibrierung zu sammeln
    for _ in range(samples):
        # Liest die Beschleunigungsdaten von den jeweiligen Registern
        x_accel = bus.read_byte_data(DEVICE_ADDRESS, ACCEL_X_REG)
        y_accel = bus.read_byte_data(DEVICE_ADDRESS, ACCEL_Y_REG)
        z_accel = bus.read_byte_data(DEVICE_ADDRESS, ACCEL_Z_REG)

        # Addiert die Werte zur Offset-Berechnung
        x_offset += x_accel
        y_offset += y_accel
        z_offset += z_accel
        time.sleep(0.1)  # Wartet 0,1 Sekunden zwischen den Messungen

    # Berechnet den Durchschnittswert für jede Achse als Offset
    x_offset /= samples
    y_offset /= samples
    z_offset /= samples

    return x_offset, y_offset, z_offset  # Gibt die Offset-Werte zurück
```

[ ]:
```
# Kalibrierung durchführen und die Offset-Werte speichern
offsets = calibrate_sensor()
print(f"Offsets - X: {offsets[0]:.2f}, Y: {offsets[1]:.2f}, Z: {offsets[2]:.
 ↪2f}")
```

[ ]:
```
# Listen zur Speicherung der Zeit und der Beschleunigungsdaten erstellen
times = []  # Liste zur Speicherung der Zeitstempel
accel_x = []  # Liste zur Speicherung der X-Beschleunigungsdaten
accel_y = []  # Liste zur Speicherung der Y-Beschleunigungsdaten
accel_z = []  # Liste zur Speicherung der Z-Beschleunigungsdaten
```

[ ]:
```
# Werte ablesen und die Offsets abziehen
try:
    start_time = time.time()  # Startzeit erfassen, um Zeitdifferenzen zu␣
 ↪berechnen
    while True:  # Unendliche Schleife, um kontinuierlich Daten zu lesen
        # Liest die Beschleunigungsdaten und subtrahiert die Offsets für genaue␣
 ↪Werte
        x_accel = bus.read_byte_data(DEVICE_ADDRESS, ACCEL_X_REG) - offsets[0]
        y_accel = bus.read_byte_data(DEVICE_ADDRESS, ACCEL_Y_REG) - offsets[1]
        z_accel = bus.read_byte_data(DEVICE_ADDRESS, ACCEL_Z_REG) - offsets[2]
```

```
        # Aktuelle Zeit erfassen, relativ zur Startzeit
        current_time = time.time() - start_time
        # Speichert die Zeit und die Beschleunigungswerte in den Listen
        times.append(current_time)
        accel_x.append(x_accel)
        accel_y.append(y_accel)
        accel_z.append(z_accel)

        # Gibt die aktuellen Beschleunigungswerte in der Konsole aus
        print(f"Beschleunigung - X: {x_accel:.2f} g, Y: {y_accel:.2f} g, Z:␣
 ↪{z_accel:.2f} g")
        time.sleep(1)  # Wartet 1 Sekunde vor der nächsten Messung

# Beendet die Schleife und fängt die KeyboardInterrupt-Ausnahme (Strg+C) ab
except KeyboardInterrupt:
    print("Messung beendet.")
```

1.5.2 Darstellung in Matplotlib

Nutzen Sie Matplotlib zur Visualisierung der Sensordaten.

[ ]:
```
# Plotten der Daten nach dem Beenden der Messung
plt.figure(figsize=(10, 6))  # Erstellt eine neue Figure mit einer bestimmten␣
 ↪Größe
# Plottet die X-Beschleunigungsdaten
plt.plot(times, accel_x, label='Acceleration X (g)', color='blue')
# Plottet die Y-Beschleunigungsdaten
plt.plot(times, accel_y, label='Acceleration Y (g)', color='green')
# Plottet die Z-Beschleunigungsdaten
plt.plot(times, accel_z, label='Acceleration Z (g)', color='red')
plt.xlabel('Time (s)')  # Beschriftung der x-Achse
plt.ylabel('Acceleration (g)')  # Beschriftung der y-Achse
plt.title('Accelerometer Data over Time')  # Titel des Plots
plt.grid(True)  # Aktiviert das Gitter im Plot
plt.legend()  # Zeigt die Legende an
plt.show()  # Zeigt den Plot an
```

1.5.3 Übung

- Änderen Sie die GPIO-Pins, um mit anderen I2C-Schnittstellen zu arbeiten.
- Lösung einblinden!

  Bild-Quelle

1.6 6. Ergebnisse und Ausblick

Der DEBO SENS 9AXIS Sensor lieferte genaue Messwerte zu Beschleunigung, Gyroskop und Magnetometer in Echtzeit. Diese Daten können zur Lageerkennung und Bewegungsverfolgung verwendet werden.

Weitere Ressourcen: - DEBO SENS 9AXIS Sensor - Jupyter Notebook Grundlagen