2.12.2024

Studienarbeit

Praktische Anwendungen von Jupyter zur IoT-Steuerung und Datenerfassung: Sensor- und Aktorsteuerung auf dem Raspberry Pi

AUTOREN:

ASHRAF YAHYA
AHMED SAJAD FAIZ

BETREUUNG:

PROF. DR. WOLFGANG ALBRECHT

Einführung

Diese Studienarbeit dokumentiert verschiedene Projekte zur Steuerung und Datenerfassung mithilfe eines Raspberry Pi. Ein zentrales Werkzeug dabei ist Jupyter Notebook, das in jedem Projekt genutzt wird, um den Code übersichtlich darzustellen und Schritt für Schritt zu erklären. Durch die interaktive Umgebung von Jupyter können theoretische Erläuterungen, Code, Diagramme und Ergebnisse direkt nebeneinander dargestellt werden, was das Lernen und Experimentieren erheblich erleichtert.

Alle Projekte folgen einer einheitlichen Gliederung, die sicherstellt, dass sie strukturiert und vergleichbar aufgebaut sind; ein Muster dafür ist im Anhang zu finden. Der Hauptteil der Studienarbeit besteht aus den erstellten Jupyter Notebooks, die in der gedruckten Version als Anhänge hintereinander aufgeführt werden. Aus diesem Grund folgt an dieser Stelle auch direkt das Abschlusskapitel. Die jeweilige Literatur ist in den einzelnen Notebooks zu finden.

Jupyter erlaubt es uns, die Funktionsweise jedes Projekts transparent und nachvollziehbar darzustellen, und macht es Anwendern einfach, Anpassungen vorzunehmen und die Ergebnisse sofort zu sehen. Die Projekte umfassen die Ansteuerung von LEDs, die Steuerung eines Servomotors und die Nutzung von Sensoren zur Erfassung von Temperatur und Bewegung. Jupyter ermöglicht es, diese Projekte dynamisch zu präsentieren, indem Diagramme und Visualisierungen in Echtzeit aktualisiert werden, z.B. durch die Verwendung der Matplotlib-Bibliothek zur Darstellung von Sensordaten.

Darüber hinaus wird die Literatur und Online-Referenzen, die in der Arbeit verwendet wurden, direkt in den Notebooks als Links gespeichert, sodass Leser leicht auf zusätzliche Ressourcen zugreifen können. Jupyter ist daher nicht nur ein Tool zur Programmierung, sondern auch eine Plattform zur Dokumentation und Weiterbildung, die dieses Projekt zu einem wertvollen, interaktiven Lernwerkzeug macht.

Ziel der Studienarbeit

Das Ziel dieser Studienarbeit ist es, die Grundlagen der Elektronik, Sensortechnik und Aktorsteuerung auf dem Raspberry Pi zu vermitteln. Jedes Projekt wurde so gestaltet, dass es praktische Erfahrungen in der Arbeit mit elektronischen Komponenten bietet und einen fundierten Einstieg in die Entwicklung und Umsetzung von IoT-Anwendungen gibt. Zudem wird der Einsatz von Jupyter Notebooks als effektives Werkzeug für die Dokumentation und Ausführung von Code hervorgehoben.

Die Studienarbeit kann sowohl als Grundlage für Einsteiger in die Elektronik als auch als Referenz für fortgeschrittene Anwender dienen, die Interesse an IoT-Projekten und der Arbeit mit dem Raspberry Pi haben.

Verzeichnis

LED-Steuerung (Gemeinsam)

Steuerung von LEDs mit PWM-Signalen auf dem Raspberry Pi. Ziel ist es, die grundlegenden Konzepte der GPIO-Steuerung und PWM-Anwendung für LED-Helligkeitsregulierung zu verstehen.

Notebook: LED_Projekt

Temperatursensor (Gemeinsam)

Misst die Umgebungstemperatur und stellt die Daten in Echtzeit dar. Mithilfe der Matplotlib-Bibliothek werden die Daten als Diagramm visualisiert, um Veränderungen anschaulich darzustellen.

Notebook: TempSensor_Projekt

Servo Motor (YAHYA)

Ein Servomotor wird über den Raspberry Pi gesteuert und bewegt sich je nach Vorgabe in Winkelstellungen zwischen 0 und 180 Grad. Dieses Projekt vermittelt die Anwendung der Pulsweitenmodulation (PWM) zur präzisen Steuerung von Servomotoren.

Notebook: ServoMotor_Projekt

Temperatursensor mit Buzzer und LED (YAHYA)

Mithilfe des BME680-Sensors wird die Umgebungstemperatur überwacht. Erreicht die Temperatur einen bestimmten Wert, sendet eine LED und ein Buzzer ein SOS-Signal, wodurch praxisorientierte Anwendungen der Temperatursteuerung demonstriert werden.

Notebook: BME680_Buzzer_LED_Projekt

DEBO SENS 9AXIS Sensor (YAHYA)

In diesem Projekt wird der DEBO SENS 9AXIS Sensor genutzt, um Bewegungs- und Lageinformationen zu erfassen und darzustellen. Die aufgezeichneten Daten werden mit geeigneten Funktionen analysiert und visualisiert.

Notebook: DEBO_SENS_9AXIS

LED Display (FAIZ)

Steuerung einer 8x8-LED-Matrix zur Darstellung verschiedener Leuchtmuster. Hierbei wird das Freenove Projects Kit verwendet, um kreative Muster und Animationen auf der Matrix anzuzeigen.

Notebook: LED_Display

Motion Sensor (FAIZ)

Mit dem HC-SR501-Bewegungssensor wird Bewegung erkannt und über eine blinkende LED signalisiert. Dieses Projekt verdeutlicht die Grundlagen der Bewegungserkennung mithilfe des des Raspberry Pi.

Notebook: Motion_Sensor

Allgemeine Notebooks

RaspberryPi_Jupyter_Grundlagen.ipynb:

Grundlagen zur Nutzung von Jupyter Notebooks auf dem Raspberry Pi, inklusive Tipps zur Code-Entwicklung und -Ausführung.

Rasp_Pi_Einrichtung.ipynb:

Einrichtung und Konfiguration des Raspberry Pi für die Nutzung mit Jupyter Notebooks und weiteren Projekten.

Installationen.ipynb:

Notwendige Installationsanweisungen für Bibliotheken und Abhängigkeiten auf dem Raspberry Pi, um die Projekte auszuführen.

Raspberry_Pi_Kommunikation.ipynb:

Einführung in die Kommunikation und Steuerung zwischen dem Raspberry Pi und externen Geräten.

Einführung in Freenove Kit.ipynb:

Einführung in die Nutzung des Freenove Kits, das in verschiedenen Projekten zur Steuerung und Sensorik verwendet wird.

Einführung in Matplotlib.ipynb:

Einführung in die Matplotlib-Bibliothek zur Datenvisualisierung, die in Projekten zur Anzeige von Sensordaten verwendet wird.

Raspberry_Pi_AccessPoint_Setup.md:

Anleitung zur Konfiguration des Raspberry Pi als Access Point, um eine drahtlose Netzwerkverbindung für die Projekte bereitzustellen.

Strukturmuster für ein Projekt

Projektname: Servo Motor

0. Bilder-Ordner
   Dieser Ordner enthält klärende Bilder zu den Projektbauteilen sowie zur Schaltungen, die im Projekt genutzt und gezeigt werden.

1. Einleitung
   Beschreibung der Ziele und Relevanz des Projekts, z.B. wie die Steuerung eines Servomotors nützlich ist, um die Prinzipien der Pulsweitenmodulation (PWM) zu verstehen.

2. Materialien
   Liste der erforderlichen Komponenten (Servomotor, Raspberry Pi, Jumper-Kabel).

3. Schaltplan
   Darstellung des Aufbaus der Schaltung, die den Servomotor mit dem Raspberry Pi verbindet, beispielsweise als Bild oder Diagramm.

4. Setup und Installation
   Anweisungen zur Installation benötigter Bibliotheken und Software (z.B. GPIO-Bibliothek für Python).

5. Code-Implementierung
   Detaillierter Python-Code zur Steuerung des Servomotors mit Erklärungen zu den Schritten, wie z.B. dem Einsatz von PWM für die präzise Bewegungssteuerung.

6. Erweiterungen und Variationen
   Vorschläge zur Projektweiterentwicklung, z.B. Steuerung des Motors basierend auf Eingabewerten oder Sensor-Daten.

7. Fazit
   Zusammenfassung der Lernergebnisse und Ideen für weitere Anwendungen des erlernten Wissens in anderen Projekten.

Abschlusskapitel

In dieser Studienarbeit wurden verschiedene Projekte zur Steuerung und Datenerfassung mit dem Raspberry Pi erfolgreich umgesetzt und dokumentiert. Insgesamt hat sich die Nutzung von Jupyter Notebooks als wertvolles Werkzeug erwiesen, um komplexe Zusammenhänge und Programmierungen übersichtlich darzustellen. Die interaktive Struktur von Jupyter hat es möglich gemacht, Code, Daten und Ergebnisse direkt miteinander zu verknüpfen, was sowohl für die Entwicklung als auch für die Dokumentation der Projekte von Vorteil war.

Besonders gut hat die Visualisierung von Sensordaten mit der Matplotlib-Bibliothek funktioniert, da sie es erlaubte, Messwerte in Echtzeit und ansprechend zu präsentieren. Allerdings stellte die Einrichtung des Raspberry Pi als Access Point einen besonders aufwendigen Schritt dar, da dabei einige Herausforderungen bei der Netzwerkkonfiguration überwunden werden mussten. Auch die Integration verschiedener Sensoren erforderte detaillierte Anpassungen und Tests, um eine zuverlässige und genaue Datenerfassung zu gewährleisten.

Insgesamt hat das Projekt gezeigt, dass der Raspberry Pi eine vielseitige und leistungsstarke Plattform für experimentelles Lernen und die Anwendung von Steuerungs- und Datenerfassungsmethoden ist. Jupyter Notebooks haben diesen Prozess zusätzlich vereinfacht und zu einem umfassenden, interaktiven Lernerlebnis beigetragen.

erry-pi-led-buzzer-bme680-notebook

November 13, 2024

1 LED-Buzzer-TemperaturSensor Projekt

1.1 1. Einführung

Dieses Jupyter Notebook führt in die praktische Anwendung der Steuerung einer LED, eines Buzzers und eines BME680-Sensors mit einem Raspberry Pi ein. Es kombiniert Elektronik und Python-Programmierung, um eine Überwachung der Temperatur zu erstellen und bei Überschreitung eines Schwellenwerts ein SOS-Signal zu senden. Das Projekt zeigt, wie man ein digitales Signal einsetzt, um die Funktionalität verschiedener Hardware-Komponenten zu steuern.

Projektskomponente im Blick:

Bilder-Quellen: Raspberry Pi 5, BME680, Buzzer, LED

Weiteres zu diesen Komponenten erfahren Sie gleich unten.

1.1.1 Projektziel

Das Ziel dieses Projekts ist es, eine LED und einen Buzzer basierend auf der Temperaturüberwachung eines BME680-Sensors zu steuern und bei bestimmter Temperatur ein SOS-Signal zu senden.

1.1.2 Relevanz

Dieses Projekt ist relevant, da es die Grundlagen der Elektronik und Programmierung kombiniert und ein grundlegendes Verständnis für die Steuerung von Aktoren in Abhängigkeit von Sensordaten entwickelt.

1.2 2. Grundlagen und Theorie

Komponente	Funktionsweise	Steuerung	Link
BME680 Sensor	Der BME680 ist ein Umweltsensor, der Temperatur, Luftfeuchtigkeit, Druck und Luftqualität misst.	Die Kommunikation mit dem BME680 erfolgt über das I2C-Protokoll.	Datasheet

Komponente	Funktionsweise	Steuerung	Link
Buzzer	Ein Buzzer ist ein elektronisches Bauteil, das einen akustischen Ton erzeugt, wenn Strom durch ihn fließt.	In diesem Projekt verwenden wir einen passiven Buzzer, der über ein PWM-Signal gesteuert wird, um Töne zu erzeugen. Die Frequenz des PWM-Signals bestimmt den Ton.	Buzzer

1.2.1 GPIO-Steuerung

- **Definition:** GPIO steht für General Purpose Input/Output. Die Pins des Raspberry Pi können entweder als Eingabe- oder Ausgabe-Pins konfiguriert werden.
- **PWM-Anwendung:** PWM steht für Pulsweitenmodulation und wird hier verwendet, um die Lautstärke des Buzzers zu steuern.

Pinout des Raspberry Pi folgt unter Schaltungsdesign.

PWM-Signale für Buzzer

- **Definition:** Wie bei Servomotoren wird auch beim Buzzer PWM verwendet, um das Signal zu modulieren. Hier variiert jedoch die Frequenz des PWM-Signals, um unterschiedliche Töne zu erzeugen.
- **Anwendung:** Der Buzzer in diesem Projekt wird mit einer PWM-Frequenz von 1000 Hz angesteuert, um einen konstanten Ton zu erzeugen.

Komponente	Funktionsweise	Steuerung
Hardware-PWM (HW-PWM)	HW-PWM wird durch spezielle Hardware-Register des Mikrocontrollers (z. B. Raspberry Pi) gesteuert.	Bietet eine präzise und stabile PWM-Signalausgabe, unabhängig von anderen Prozessen. Verwendet für zeitkritische Anwendungen wie Motorsteuerung.
PINs HW-PWM	PWM0: GPIO 12 (Pin 32), GPIO 18 (Pin 12) PWM1: GPIO 13 (Pin 33), GPIO 19 (Pin 35)	
Software-PWM (SW-PWM)	SW-PWM wird in der Software implementiert, indem Pins durch Programmierung mit `sleep()` ein- und ausgeschaltet werden.	Geringere Präzision und Stabilität, kann durch andere laufende Prozesse beeinflusst werden. Nützlich für einfache Anwendungen oder wenn HW-PWM nicht verfügbar ist.

Komponente	Funktionsweise	Steuerung
PINs SW-PWM	GPIO 18: Häufig genutzt, da viele Tutorials darauf verweisen. GPIO 23: Beliebter Pin für SW-PWM. GPIO 24: Geeignet für Projekte mit mehr PWM-Kanälen.	

Arten von PWM: **Hilfreiche Links**
Buzzer-Datenblatt
Buzzer verwenden
PWM-Signals How to Use Active and Passive Buzzers

1.2.2 SOS Signal

- **Morsecode:** SOS ist ein internationaler Notruf im Morsecode, dargestellt durch drei kurze Signale (Punkte), gefolgt von drei langen Signalen (Striche) und wiederum drei kurzen Signalen (Punkte).

- Für weitere über Morsecode klicken Sie hier.

1.3 3. Materialien und Werkzeuge

Kategorie	Komponenten
Software	Raspbian OS Python 3 Jupyter Notebook
Hardware	Raspberry Pi 5 Model B Rev 1.0 Breadboard Jumper-Kabel BME680 Sensor LED Buzzer Widerstand X Ohm

Übung Berechnen Sie den Widerstandswert, der für die Schaltung mit Red-LED gedacht ist.

Lösung R = {3,3V - 2V}/{0,02A} = 65 ohm.

```
Ein Widerstand von 65 Ohm oder dem nächsthöheren verfügbaren Wert (z. B. 68 Ohm) wäre geeignet
```

Tipp: gucken Sie sich dieses Notebook.

Komponente	Verbindung
LED	GPIO 17 des Raspberry Pi
Buzzer	GPIO 18 des Raspberry Pi
BME680 Sensor	Über I2C mit dem Raspberry Pi verbunden

Übung Überlegen Sie sich, wie der Widerstand verdrahtet werden kann.

Tipp!

```
Gucken Sie sich den Schaltplan unten.
```

1.4 4. Schaltungsdesign

1.4.1 Raspberry Pi 5 GPIO-Pinout

Das folgende Bild zeigt die PINs und Protokollen eines Rasüpberry Pi und die wichtigsten Funktionen, die mit PINs verbunden sind.

Weiteres zu diesem Bild

1.4.2 Zu BME680 Sensor

BME680 Schaltung
Aussehen des BME680 Temperatursensors

Weiteres zu diesem Bild

So sieht die Schaltung des BME680 Sensors mit Rasp Pi PINs aus.

Erstellt durch easyeda

Alternatives Bild
Dieses Alternatives Bild zeigt eine klare Oberfläche der Komponentenschaltung.

Erstellt durch circuito.io

1.4.3 Schaltplan

Kompletter Schaltplan
Kompletter Schaltplan mit klarer Oberfläche, der durch circuito.io erstellt wurde.

Alternativ
Kompletter Schaltplan mit Pasp Pi Pins, der durch easyeda erstellt wurde.

Aussehen in Realität
So könnte Ihre Schaltung in der Realität aussehen.

1.5 5. Implementierung

1.5.1 Hardware-Aufbau

1. Komponenten auf dem Breadboard platzieren und verkabeln.
2. Raspberry Pi aufstellen und mit Strom durch an PC angeschlossenes USB versorgen.

1.5.2 Software-Setup

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

1.5.3 Installieren Sie die erforderlichen Bibliotheken:

Führen Sie die folgenden zwei Zeile direkt auf Jupyter aus, um die weitre notwendige Bibliotheken zu diesem Notebook zu installieren.

Installation von `adafruit-circuitpython-bme680`

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

Die Adafruit_CircuitPython_BME680-Bibliothek ist eine Python-Bibliothek für die Verwendung mit dem BME680-Sensor von Adafruit. Die Bibliothek ermöglicht eine einfache Integration des Sensors in Projekte, die auf der CircuitPython-Plattform basieren, und bietet Funktionen zur Abfrage der gemessenen Werte. Klicken Sie hier, um weiteres zu adafreuit-circuitpython zu erfahren.

[ ]:
```
!sudo apt-get install python3-rpi.gpio
```

`RPi.GPIO` wurde unter dem Schritt 4 hier in dem Notebook erklärt.

[ ]:
```
#importieren der notwendigen Bibliotheken
import board
```

`board`: Diese Bibliothek ermöglicht den Zugriff auf die Pins des Raspberry Pi, sodass sie in CircuitPython-Programmen verwendet werden können. Sie definiert eine Vielzahl von Board-spezifischen Konstanten, die in der Hardware-Programmierung genutzt werden können.
Board-Dokumentation

[ ]:
```
import busio
```

`busio`: Diese Bibliothek stellt Klassen zur Verfügung, die serielle Kommunikationsschnittstellen wie I2C und SPI unterstützen. Sie wird häufig in Kombination mit Sensoren verwendet, um Daten zwischen dem Mikrocontroller und den angeschlossenen Geräten auszutauschen.
Busio-Dokumentation

[ ]:
```
import adafruit_bme680
import RPi.GPIO as GPIO
import time
```

`adafruit_bme680`: Diese Bibliothek ist speziell für die Verwendung mit dem BME680-Sensor von Adafruit entwickelt worden. Sie ermöglicht die einfache Integration und Abfrage von Temperatur-, Luftfeuchtigkeits-, Luftdruck- und Luftqualitätsdaten.
Intro zu Adafruit_bme680
Weiteres zu Time-Bibliothek

[ ]:
```
import sys
```

`sys`: Diese Standard-Python-Bibliothek bietet Zugang zu System-spezifischen Parametern und Funktionen. Sie ermöglicht z.B. das Beenden des Programms oder den Zugriff auf Kommandozeilenargumente.
Sys-Dokumentation

[ ]:
```
# GPIO Pins definieren
LED_PIN = 17
BUZZER_PIN = 18

# GPIO-Modus (BCM) und Pins konfigurieren
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(BUZZER_PIN, GPIO.OUT)

# PWM für den Buzzer einrichten
pwm = GPIO.PWM(BUZZER_PIN, 1000)  # 1000 Hz Frequenz
pwm.start(0)  # Start PWM mit 0% Duty Cycle (Buzzer aus)

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

Übung Für BWM, also Buzzer, wurde der PIN 18 genutzt. Überlegen Sie sich, ob Sie den Buzzer an einem anderen PIN anschließen können.
Ist es noch möglich? Wenn ja, an welchem PIN denn?

Lösung! Ja, es ist möglich, den Buzzer an einen anderen Pin anzuschließen, solange dieser Pin PWM unterstützt. Andere PWM-fähige Pins auf dem Raspberry Pi sind:
- GPIO 12 (Pin 32)
- GPIO 13 (Pin 33)
- GPIO 19 (Pin 35)

Zum Beispiel könnten Sie GPIO 12 (Pin 32) verwenden.
Gucken Sie sich das Pinout-Bild.

[ ]:
```
# BME680 Sensor initialisieren
try:
    bme680 = adafruit_bme680.Adafruit_BME680_I2C(i2c)
    bme680.sea_level_pressure = 1013.25
    print("Sensor erfolgreich initialisiert!")
except Exception as e:
    print(f"Fehler beim Initialisieren des Sensors: {e}")
    GPIO.cleanup()
    exit(1)
```

Übung Für die Kommunikation mit dem Temperatursensor wurde dir Kommunikation-schnittstelle i2c benutzt. Überlegen Sie sich, ob sie eine andere Stelle verwenden können.

Lösung!

Ja, der BME680 unterstützt auch die SPI-Schnittstelle als Alternative zu I2C. Bei SPI sind die benötigten Pins:
- MOSI (Master Out, Slave In) - MISO (Master In, Slave Out) - SCK (Serial Clock) - CS (Chip Select)

Wenn Sie den Sensor über SPI anschließen möchten, müssen Sie die Verkabelung und den Code entsprechend anpassen, um die SPI-Schnittstelle zu verwenden.

[ ]:
```
def dot():
    GPIO.output(LED_PIN, GPIO.HIGH)
```

```
    pwm.ChangeDutyCycle(50)  # Buzzer an
    time.sleep(1)  # LED 1 Sekunde an (Punkt)
    GPIO.output(LED_PIN, GPIO.LOW)
    pwm.ChangeDutyCycle(0)  # Buzzer aus
    time.sleep(1)  # LED 1 Sekunde aus (Pause zwischen Zeichen)

def dash():
    GPIO.output(LED_PIN, GPIO.HIGH)
    pwm.ChangeDutyCycle(50)  # Buzzer an
    time.sleep(3)  # LED 3 Sekunden an (Strich)
    GPIO.output(LED_PIN, GPIO.LOW)
    pwm.ChangeDutyCycle(0)  # Buzzer aus
    time.sleep(1)  # LED 1 Sekunde aus (Pause zwischen Zeichen)

def letter_space():
    time.sleep(3)  # Pause zwischen Buchstaben (3 Sekunden)

def word_space():
    time.sleep(7)  # Pause zwischen Wörtern (7 Sekunden)

def send_sos():
    dot(); dot(); dot()  # S
    letter_space()
    dash(); dash(); dash()  # O
    letter_space()
    dot(); dot(); dot()  # S
    word_space()
```

Dieser Code-Abschnitt definiert bzw. implementiert die SOS-Funktion.

Übung Recherschieren Sie, wie das Wort “Help” durch Morsecode dargestellt werden kann.

Lösung!

Das Wort “HELP” wird im Morsecode folgendermaßen dargestellt:
- H: …. (vier kurze Signale)
- E: . (ein kurzes Signal)
- L: .-.. (ein kurzes, ein langes, zwei kurze Signale)
- P: .–. (ein kurzes, zwei lange, ein kurzes Signal)

[ ]:
```
try:
    while True:
        # Temperatur lesen
        temperature = bme680.temperature
        if temperature is not None:
            # Temperatur ausgeben und Konsole aktualisieren
            sys.stdout.write(f"\rAktuelle Temperatur: {temperature:.2f} °C")
            sys.stdout.flush()
```

```
            # Temperaturüberprüfung
            if temperature >= 30:
                # Temperatur >= 30°C: LED und Buzzer einschalten
                GPIO.output(LED_PIN, GPIO.HIGH)
                pwm.ChangeDutyCycle(50)  # Buzzer an
                send_sos()  # SOS-Morsecode senden
            else:
                # Temperatur < 30°C: LED und Buzzer ausschalten
                GPIO.output(LED_PIN, GPIO.LOW)
                pwm.ChangeDutyCycle(0)  # Buzzer aus

        else:
            print("Fehler beim Lesen der Temperatur")

        # Verzögerung vor der nächsten Messung
        time.sleep(1)

except KeyboardInterrupt:
    print("\nProgramm beendet")

finally:
    pwm.stop()  # Stoppe PWM
    GPIO.cleanup()  # GPIO-Pins zurücksetzen
```

Dieser Code-Abschnitt ist sozusagen unsere main-Funktion. Hier wird der ganze Code und andere Funktionen aufgerufen und die PINs gesteuert.

1.6 6. Experimente und Ergebnisse

1.6.1 Versuchsaufbau

- Der oben beschriebene Hardware-Aufbau wurde verwendet.
- Der Code wurde in einem Jupyter Notebook ausgeführt.
- Der Sensor überwachte die Temperatur, und die LED und der Buzzer reagierten entsprechend.

1.7 7. Zusätzliche Ressourcen

Für besseres Verstand und für die Klarheit gucken Sie sich die Notebooks bzw. folgenden Projekts: - Grundlagen-Notebook - BME680-Notebook - LED-Notebook

- CircuitPython & Python

- Adafruit Library Reference
- Leg los mit CircuitPython
- CircuitPython

1.8 Übung

Überlegen Sie sich wie sie einen Motor mit einer Fan bzw. Ventilator verbinden können, so dass die Ventilator eingeschaltet wird, wenn die SOS-Hilfe aufgerufen wird, also wenn LED leuchtet und der Buzzer pipt.

Heraus zu finden: - Notwendige PINs - Transistorenart - Verdrahtung in der obigen Schaltung

Lösung!

Um den Motor in die Schaltung zu integrieren, könnte ein NPN-Transistor verwendet werden, um den Motor an den Raspberry Pi anzuschließen. Der Transistor wird benötigt, weil der Raspberry Pi nicht genug Strom liefert, um den Motor direkt anzutreiben.

- **Notwendige PINs:** Ein GPIO-Pin (z. B. GPIO 23) für das Steuersignal zum Transistor.
- **Transistor:** Ein NPN-Transistor wie der 2N2222 könnte verwendet werden.
- **Verdrahtung:**
  - Der Kollektor des Transistors wird mit dem Minuspol des Motors verbunden.
  - Der Emitter wird mit der Masse verbunden.
  - Der GPIO-Pin wird über einen Widerstand an die Basis des Transistors angeschlossen.
  - Der Pluspol des Motors wird an eine externe Spannungsquelle (z. B. 5V oder 12V) angeschlossen.

Wenn die LED leuchtet und der Buzzer piept, wird der GPIO-Pin aktiviert, wodurch der Transistor durchschaltet und den Motor aktiviert.

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

LED_Matrix mithilfe des Freenove FNK0054 "Projects Kit for Raspberry Pi"

1. Einführung

Quelle

Quelle

Projektziel

Das Ziel dieses Projekts ist es, eine LED-Matrix mithilfe eines Raspberry Pi und des Freenove FNK0054 "Projects Kit for Raspberry Pi" zu steuern. Dabei wird zwei 74HC595-Schieberegister verwendet, um die Anzahl der benötigten GPIO-Pins zu minimieren und verschiedene Muster, Zeichen und Symbole auf der LED-Matrix anzuzeigen.

Hintergrund

Das Freenove FNK0054 "Projects Kit for Raspberry Pi" ist ein umfangreiches Kit, das verschiedene Komponenten und Module enthält, darunter LEDs, Widerstände, Schieberegister und andere Bauelemente. Dieses Kit ermöglicht es, eine Vielzahl von Projekten durchzuführen, um die Fähigkeiten des Raspberry Pi in der Elektroniksteuerung zu demonstrieren. Eine LED-Matrix, gesteuert durch ein 74HC595-Schieberegister, zeigt, wie digitale Elektronik und Python-Programmierung kombiniert werden können, um visuelle Ausgaben zu erzeugen.

Relevanz

Das Projekt zeigt praxisnah, wie man eine LED-Matrix mit einem Raspberry Pi und Standardkomponenten aus einem Elektronik-Kit steuern kann. Es verdeutlicht, wie einfache Hardware-Schnittstellen programmiert werden können und ist ein exzellentes Beispiel für den Einsatz von Schieberegistern in der Elektronik.

2. Grundlagen und Theorie

LED-Matrix:

Eine LED-Matrix besteht aus einer Gruppe von LEDs, die in einem rechteckigen Gitter angeordnet sind. Jede LED kann individuell angesteuert werden, um komplexe Muster und Zeichen darzustellen.

74HC595 Schieberegister:

Der 74HC595 Schieberegister ist ein 8-Bit-Schieberegister mit seriellen Eingängen und parallelen Ausgängen. Es ermöglicht die Steuerung mehrerer LEDs mit einer geringen Anzahl von Steuerleitungen. Durch das Schieberegister wird die Anzahl der benötigten GPIO-Pins am Raspberry Pi reduziert, da die Daten seriell in das Register geschrieben werden und dann parallel an die LEDs ausgegeben werden.

- Schieberegister: Eine Übersicht und Definition

Freenove "Projects Kit for Raspberry Pi"

Das Freenove FNK0054 "Projects Kit for Raspberry Pi" ist ein umfassendes Elektronik-Kit, das speziell entwickelt wurde, um Anwendern zu helfen, Projekte mit einem Raspberry Pi durchzuführen. Es ist besonders gut geeignet für Einsteiger, die die Grundlagen der Elektronik und Programmierung erlernen möchten. Das Kit enthält eine Vielzahl von Komponenten und Bauteilen, mit denen eine breite Palette von Projekten realisiert werden kann.

- Offizielle Webseite
- PDF-Tutorial
- Video-Tutorial

3. Materialien und Werkzeuge

Software

- Raspbian OS
- Python 3
- Jupyter Notebook
- Python 3 Bibliotheken wie **gpiozero**

Hardware

- Raspberry Pi
- Freenove Projects Board für Raspberry Pi
- LED-Matrix
- GPIO Ribbon Kabel

- Freenove-Tutorial

Sensor/(Aktor)en, inkl. Datenblätter

der 74HC595 Schieberegister wird als Aktor benutzt und die Daten aus Datenblatt lautet:

Betriebsspannung (Vcc): 2V bis 6V, typischerweise 5V.

Maximaler Ausgangsstrom pro Pin: 35 mA.

Maximaler Gesamtstrom (alle Ausgänge kombiniert): 70 mA.

Berechnung des Vorwiderstands

der Vorwiderstand der LED kann wie folgt berechnet werden:

```
# Beispielcode zur Berechnung des Vorwiderstands:
V_supply = 5.0  # Versorgungsspannung in Volt
V_f = 2.0  # Vorwärtsspannung der LED in Volt
I_f = 0.02  # Vorwärtsstrom der LED in Ampere

R = (V_supply - V_f) / I_f
print("Erforderlicher Vorwiderstand: {} Ohm".format(R))
```

```
Erforderlicher Vorwiderstand: 150.0 Ohm
```

Pin Beschaltung

Die Pin-Beschaltung in Freenove Projects Kit für Raspberry Pi ist wie flogt:

74HC595 Schieberegister	GPIO Pin des Raspberry Pi
DS(PIN 14)	GPIO 22
SR_CP(PIN 12)	GPIO 27
SH_CP(PIN 11)	GPIO 17

Freenove-Tutorial

4. Schaltungsdesign

Schmatischer Entwurf

- Schaltung-Aufbau
- Freenove-Tutorial

Hardware-Verbindung

Freenove-Tutorial

5. Implementierung

Hardware-Aufbau

Der Hardware soll wie obbiges Bild aufgabeut werden.

- Raspberry einschalten und durch das Ribbon Kabel mit Freenove Projects Kit für Raspberry Pi verbunden.
- Power-Taste von Freenove Projects Kit für Raspberry Pi einschalten.

- Um LED-Matrix 8*8 nutzen zu können, sollen wir den Schalter 7 des Boards einschalten.

Software-Setup

1. Raspbian OS installieren.
2. Python und Jupyter Notebook installieren: `sh sudo apt-get update sudo apt-get install python3 jupyter`
3. Installation der benötigten Bibliotheken: ```sh sudo apt-get install python-gpiozero

```

## 6. Experimente und Ergebnisse

## Beispiel 1

### Installieren Sie gpiozero auf Ihrem Rechner

```
pip install RPi.GPIO
```

```
pip install gpiozero
```

```
pip install luma.led_matirx
```

```
pip install pillow
```

in diesem Beispiel wird Herz-Symbol gezeigt

```
# importieren von Bibliotheken
from gpiozero import OutputDevice

# importieren von time
import time

# LSB und MSB
LSBFIRST = 1
MSBFIRST = 2

# Pin-Definition
dataPin = OutputDevice(22)
latchPin = OutputDevice(27)
clockPin = OutputDevice(17)

# Testmuster für Herz
pic = [0x0C, 0x1E, 0x3F, 0x7F, 0x7F, 0x3E, 0x1C, 0x08]

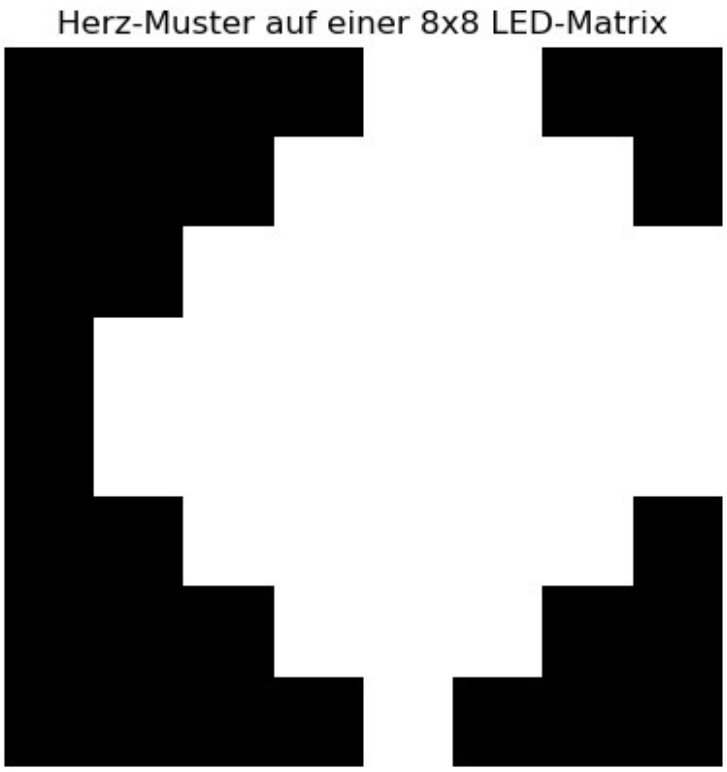
# Funktion, um 8 Bits in der angegebenen Reihenfolge an das
Schieberegister zu senden
def shiftOut(order, val):
    for i in range(0, 8):
        clockPin.off()
```
```

```
        if order == LSBFIRST:
            dataPin.on() if (0x01 & (val >> i) == 0x01) else
dataPin.off()
        elif order == MSBFIRST:
            dataPin.on() if (0x80 & (val << i) == 0x80) else
dataPin.off()
        clockPin.on()

# Funktion, um das Herzmuster kontinuierlich auf der LED-Matrix
anzuzeigen
def loop():
    while True:
        for j in range(0, 500):  # Wiederholt das Muster zur
Stabilisierung der Anzeige
            x = 0x80
            for i in range(0, 8):  # Durchläuft jede Spalte des
Musters
                latchPin.off()
                shiftOut(MSBFIRST, pic[i])  # Überträgt die
Zeilendaten
                shiftOut(MSBFIRST, ~x)  # Überträgt die Spaltenauswahl
                latchPin.on()
                time.sleep(0.001)  # Verzögerung für eine flüssigere
Anzeige
                x >>= 1  # Verschiebt die Aktivierung zur nächsten
Spalte

# Funktion zum Löschen der LED-Matrix und Freigeben der Ressourcen
def destroy():
    latchPin.off()
    shiftOut(MSBFIRST, 0x00)
    shiftOut(MSBFIRST, 0x00)
    latchPin.on()
    dataPin.close()
    latchPin.close()
    clockPin.close()

# Hauptprogramm-Ausführung
if __name__ == '__main__':
    print('Programm startet...')
    try:
        loop()  # Startet die Anzeigeschleife
    except KeyboardInterrupt:
        print("Programm wird beendet")
    finally:
        destroy()  # Stellt sicher, dass die Ressourcen beim Beenden
freigegeben werden
```

Quelle: Freenove-Tutorial

Darstellenung in Matplotlib

```
import matplotlib.pyplot as plt
import numpy as np

# Muster zur Anzeige eines Herzsymbols (wie in deinem ursprünglichen
Code)
pic = [0x0C, 0x1E, 0x3F, 0x7F, 0x7F, 0x3E, 0x1C, 0x08]

# Funktion, um das Herzmuster zu extrahieren und als 8x8-Array
darzustellen
def convert_to_matrix(pic):
    # Wir erzeugen ein 8x8 NumPy Array, um das Muster zu speichern
    matrix = np.zeros((8, 8), dtype=int)
    for i in range(8):
        # Wandelt das Hex-Wert-Muster in binäre Werte um und speichert
sie in der Matrix
        for j in range(8):
            if (pic[i] & (1 << (7 - j))) != 0:
                matrix[i, j] = 1
    return matrix

# Herzmuster in eine Matrix umwandeln
matrix = convert_to_matrix(pic)

# Visualisierung mit Matplotlib
plt.imshow(matrix, cmap='gray', interpolation='nearest')
plt.axis('off')  # Keine Achsen anzeigen
plt.title("Herz-Muster auf einer 8x8 LED-Matrix")
plt.show()
```



Was sind LSB und MSB ?

MSB (Most Significant Bit): das Bit mit Index Null hat den geringsten Wert.

LSB (Least Significant Bit): das Bit mit Index Null hat den höchsten Wert.

Aufgabe1:

Wie lautet das Schachbrettmuster ?

Aufgabe2

Versuchen Sie das obige Programm so anzupassen, dass Schachbrettmuster auf die LED-Matrix angezeigt wird.

Beispiel 2

Installieren Sie gpiozero auf Ihrem Rechner

```
sudo apt-get install python-gpiozero
```

in diesem Beispiel wird Hallo auf den LED-Matrix gezeigt

```
# importieren von Bibliotheken
from gpiozero import OutputDevice
```

```
# importieren von time
import time

# LSB und MSB
LSBFIRST = 1
MSBFIRST = 2

# Pin-Definition
dataPin = OutputDevice(22)
latchPin = OutputDevice(27)
clockPin = OutputDevice(17)

# Testmuster
H = [0x7F, 0x49, 0x49, 0x49, 0x49, 0x49, 0x49, 0x00]
A = [0x3E, 0x51, 0x49, 0x49, 0x49, 0x51, 0x3E, 0x00]
L = [0x7F, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00]
O = [0x3E, 0x41, 0x41, 0x41, 0x41, 0x41, 0x3E, 0x00]

data = H + [0x00] * 8 + A + [0x00] * 8 + L + [0x00] * 8 + L + [0x00] *
8 + O  # Kombiniert die Muster der Buchstaben mit Leerraum

# Funktion, um 8 Bits in der angegebenen Reihenfolge an das
Schieberegister zu senden
def shiftOut(order, val):
    for i in range(8):
        clockPin.off()

        if order == LSBFIRST:
            dataPin.on() if (0x01 & (val >> i)) else dataPin.off()
        elif order == MSBFIRST:
            dataPin.on() if (0x80 & (val << i)) else dataPin.off()
        clockPin.on()

# Funktion, um die Buchstaben von "HALLO" nacheinander auf der Matrix
anzuzeigen
def loop():
    while True:
        for k in range(0, len(data) - 8):
            for j in range(0, 20):
                x = 0x80
                for i in range(k, k + 8):
                    latchPin.off()
                    shiftOut(MSBFIRST, data[i])
                    shiftOut(MSBFIRST, ~x)
                    latchPin.on()
                    time.sleep(0.001)
                    x >>= 1

def destroy():
    latchPin.off()  # Setzt den Latch-Pin auf niedrig, um das
Schieberegister zu deaktivieren
```

```
    shiftOut(MSBFIRST, 0x00)  # Überträgt 0x00, um alle Zeilen zu
deaktivieren
    shiftOut(MSBFIRST, 0x00)  # Überträgt 0x00, um alle Spalten zu
deaktivieren
    latchPin.on()  # Setzt den Latch-Pin auf hoch, um die Ausgänge zu
aktualisieren
    dataPin.close()  # Schließt den Daten-Pin, um die Ressourcen
freizugeben
    latchPin.close()  # Schließt den Latch-Pin
    clockPin.close()  # Schließt den Takt-Pin

# Hauptprogramm-Ausführung
if __name__ == '__main__':
    print('Displaying "HALLO"...')
    try:
        loop()
    except KeyboardInterrupt:
        print("Ending program")
    finally:
        destroy()
```

```
Displaying "HALLO"...
Ending program
```

Quelle: Freenove-Tutorial

Darstallung in Matplotlib

Aufgabe4:

Zeigen Sie Wellcome auf den LED-Matrix.

Der Testmuster dafür lautet:

[

```
0x1E, 0x36, 0x36, 0x36, 0x36, 0x36, 0x00, 0x00,  # W

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # E

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # L

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # C

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # O

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # M

0x7C, 0x08, 0x08, 0x7C, 0x00, 0x00, 0x00, 0x00,  # E
```

]

7. Diskussion und Fazit

Diskussion

Das Projekt demonstriert die Steuerung einer LED-Matrix mit einem Raspberry Pi und einem Schieberegister, unter Verwendung des Freenove FNK0054 Kits. Die erfolgreiche Implementierung zeigt die Vielseitigkeit des Kits und des Raspberry Pi und die Möglichkeiten der digitalen Elektronik.

Fazit

Das Projekt war erfolgreich und hat das Verständnis für die Steuerung von LED-Matrizen mit dem Raspberry Pi verbessert. Die Implementierung ist einfach, aber effektiv und kann als Grundlage für weiterführende Projekte dienen.

led-projekt

November 13, 2024

1 LED Projekt mit Raspberry Pi und Jupyter Notebook

1.1 1. Einführung

Dieses Jupyter Notebook führt in die praktische Anwendung von GPIO-Steuerungen mit einem Raspberry Pi ein. Es kombiniert Elektronik und Python-Programmierung, um grundlegende Schaltungen zu erstellen und zu testen. Das Projekt zeigt, wie man LEDs mit verschiedenen Bibliotheken steuert, Widerstände berechnet und Schaltkreise aufbaut. Mit einer schrittweisen Anleitung und praxisnahen Experimenten bietet dieses Notebook eine solide Grundlage für Einsteiger in die Welt der Elektronik und die Nutzung des Raspberry Pi für Hardware-Projekte.

1.1.1 Projektziel

Das Ziel dieses Projekts ist es, den Raspberry Pi mithilfe von Jupyter Notebook zu steuern und dabei verschiedene Schaltungen unter Berücksichtigung von Widerständen zu erstellen und zu testen.

1.1.2 Relevanz

Dieses Projekt ist relevant, da es die Grundlagen der Elektronik und Programmierung kombiniert und ein grundlegendes Verständnis für die Verwendung von GPIO-Pins und die Programmierung in Python entwickelt.

1.2 2. Grundlagen und Theorie

1.2.1 Elektrische Grundlagen

- **Spannung (V):** Der elektrische Potentialunterschied.
- **Strom (I):** Der Fluss von Elektronen durch einen Leiter.
- **Widerstand (R):** Der Widerstand gegen den Stromfluss, gemessen in Ohm (Ω).

1.2.2 Schaltkreise

- **Serienschaltung:** Komponenten sind in einer Reihe geschaltet.
- **Parallelschaltung:** Komponenten sind parallel zueinander geschaltet.

1.2.3 Widerstände

- **Funktion:** Begrenzen den Stromfluss und teilen Spannungen.
- **Berechnung:** Ohm'sches Gesetz: (U = I * R)

1.3 3. Materialien und Werkzeuge

1.3.1 Software

- Raspbian OS
- Python 3
- Jupyter Notebook

1.3.2 Hardware

- Raspberry Pi 5 Model B Rev 1.0

- Breadboard
- Jumper-Kabel
- LEDs
- Widerstände (verschiedene Werte)

1.3.3 Sensor/(Aktor)en, inkl. Datenblätter

In diesem Versuch wird GPIO des Pi und der LED als Aktor benutzt.

Komponente	Spannung (V)	Strom (mA)
LED-Rot	1.8	20
LED-Gelb	2.0	20
LED-Grün	2.8	10
LED-Blau	3.0	10
Widerstand	220 ohm	5% Toleranz

1.3.4 Links zu den Datenblättern

LED-Datenblatt
Widerstand-Datenblatt

1.3.5 Berechnung des Vorwiderstands

[3]:
```
### als Beispiel wird hier LED-Rot benutzt
Vcc = 3.3  # Spannung des GPIO-Pins
Vf = 1.8   # Vorwärtsspannung der LED
If = 0.02  # Vorwärtsstrom der LED

R = (Vcc - Vf) / If
print("%ld",R)
```

```
%ld 74.99999999999999
```

Für die Verdeutlichung zur Berechnung des Vorwiderstands klicken Sie hier.

1.3.6 Übung

Im obigen Beispiel wurde der Vorwiderstand des roten LED berechnet.
Berechnen Sie den Vorwiderstand der grünen LED und zeigen Sie dem Dozent Ihre Rechnung.

1.3.7 Pin Beschaltung

- Für dieses Projekt nutzen wir GPIO 17.
- Die Anode der LED wird mit GPIO17 von Pi verbunden.
- Die Kathode der LED wird mit dem Vorwiderstand verbunden.
- Der Vorwiderstand wird mit GND des Pi verbunden.

1.4 4. Schaltungsdesign

1.4.1 Schaltplan 1

Folgende Schaltung wurde durch circuito erstellt.

1.4.2 Schaltplan 2 LEDS

Der Schaltplan für zwei LEDs mit einem gemeinsamen Kathodenanschluss könnte wie folgt aussehen:

1.4.3 Komponentenauswahl

- **Widerstände:** 220Ω für LED-Schutz.
- **LED:** Standard 5mm LED.

1.5 5. Implementierung

1.5.1 Hardware-Aufbau

1. Komponenten auf dem Breadboard platzieren und verkabeln.
2. Raspberry Pi aufstellen und durch an PC angeschlossenes USB mit Strom versorgen.

1.5.2 Software-Setup

1.5.3 Software-Setup

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

1.5.4 Installiere die RPi.GPIO Bibliothek:

Die Installation von RPi.GPIO finden Sie unter Installationen.

1.5.5 Code zum LED-Blinken

Der Code für LED-Blinken könnte folgendermaßen so aussehen:

“‘python import RPi.GPIO as GPIO import time

2 Setup

GPIO.setmode(GPIO.BCM) GPIO.setup(17, GPIO.OUT)

3 LED Blinken

try: while True: GPIO.output(17, GPIO.HIGH) time.sleep(1) GPIO.output(17, GPIO.LOW) time.sleep(1) except KeyboardInterrupt: GPIO.cleanup()

Dieser Code finden Sie unten zum Ausführen. Er ist in mehreren Abschnitten geteilt und erklärt.

3.0.1 Links zur Vorbereitung und Implementierung des Versuchs:

Für weiteres können Sie folgende Datenblätter durchgehen und gucken. - LED einschalten und ausschalten - LED ansteuern (GPIO)

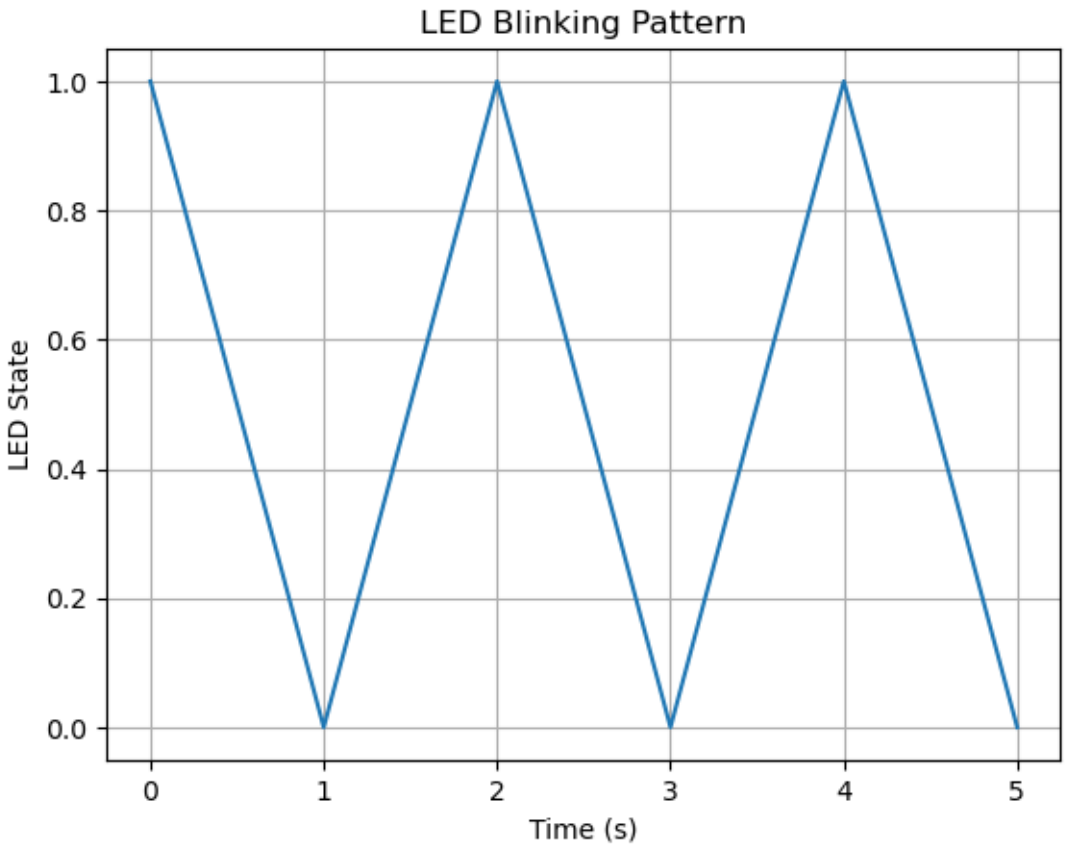
3.0.2 Darstellung in Matplotlib

Matplotlib ist eine weit verbreitete Bibliothek in Python, die für die Erstellung von statischen, animierten und interaktiven Grafiken verwendet wird. Sie bietet eine Vielzahl von Funktionen zum Plotten von Diagrammen, wie Linien-, Balken-, Kreis-, Streu- und Histogrammdiagrammen. Matplotlib ist besonders beliebt, weil sie flexibel und anpassbar ist, was sie zu einem mächtigen Werkzeug für die Visualisierung von Daten in Wissenschaft, Technik und Datenanalyse macht.

Die Anweisungen zur Installation von Matplotlib finden Sie unter Installationen.

Beispiel zu Matplotlib:

[1]:
```
import matplotlib.pyplot as plt

times = [0, 1, 2, 3, 4, 5] #wie oft bzw. wie lang eine led ein und ausleuchtet
states = [1, 0, 1, 0, 1, 0]# States einer led, wobei 1 an und 0 aus bedeutet.

plt.plot(times, states)
plt.xlabel('Time (s)')
plt.ylabel('LED State')
plt.title('LED Blinking Pattern')
plt.grid(True)
plt.show()
```



3.0.3 Nutzliche Links zu Matplotlib

- Pyplot Tutorial
- Matplotlib interactive examples

3.0.4 Übung:

Im obigen Beispiel mit Matpoltlip wurden zwei Arrays bzw. Vektoren behandelt. In diesem Beispiel ging es nur um LED ein und auschalten. Betrachten Sie die times-Array als Stunden, also erste Stunde, zweite Stunde usw. und Ergänzen Sie das states-Array durch sinnvolle Temperatur-Werte. Lassen Sie das Diagramm davon durch Matplotlib generiert werden. Anschließend zeigen Sie dem Dozent Ihre Lösung.

3.1 6. Experimente und Ergebnisse

3.1.1 Versuchsaufbau

- Der oben beschriebene Hardware-Aufbau wurde verwendet.
- Der Code wurde in einem Jupyter Notebook ausgeführt.
- Die LED leuchtete auf, wie im Code vorgesehen.

3.2 LED Beispiel 1

In diesem Beispiel geht es darum, eine LED mit der Bibliothek gpiozero ein und auszuschalten.

Die gpiozero Bibliothek ist ideal für die LED-Steuerung, da sie eine einfache und intuitive API bietet, die den Einstieg in die Hardwareprogrammierung erleichtert. Sie abstrahiert die komplexen Details der GPIO-Steuerung und ermöglicht es, mit wenigen Zeilen Code präzise Hardware-Interaktionen zu realisieren.

3.2.1 Installation von Bibliotheken

Installieren Sie die Bibliothek `gpiozero` auf Ihrem Rechner falls noch nicht geschehen.
Die Anweisungen dafür finden Sie unter Installationen

Abhilfe Wenn Ihr Notebook die Bibliothek immer noch nicht findet, dann führen Sie den folgenden Befehl einfach hier auf Jupyter. Es wird dann direkt auf Ihrem Rechner installiert und Sie können weiter arbeiten.

[ ]:
```
!pip3 install gpiozero
```

Importieren Sie die notwendigen Bibliotheken:

[7]:
```
# importieren von gpiozero
from gpiozero import LED

# importieren von time
from time import sleep
```

Implementierung für ein und ausschalten einer an PIN 17 angesclossenen LED.

[ ]:
```
# Code für ein- und ausschalten
roteled = LED(17)
while True:
    roteled.on()
    sleep(1)
    roteled.off()
    sleep(1)
```

Ihre an Port bzw. PIN 17 angeschloßene LED soll jetzt ein- und ausblinken

Übung: Die LED soll mit doppelter Frequenz leuchten. Überlegen Sie sich, wie das geht und tragen Sie Ihre Änderungen direkt im Code. Testen Sie schließlich den Code nochmal. Leuchtet die LED mit doppelter Frequenz? Erklären Sie Ihrem Kollege, wie Sie den Code angepasst haben.

3.3 LED Beispiel 2

RPi.GPIO In diesem Beispiel wird im Prinzip nichts anderes als das, was im Beispiel 1 gemacht wurde, außer dass wir jetzt mit einer anderen Bibliothek arbeiten werden, nähmlich `RPi.GPIO` Bibliothek.

Die RPi.GPIO Bibliothek ist eine robuste Wahl für die LED-Steuerung, da sie direkten Zugriff auf die GPIO-Pins des Raspberry Pi ermöglicht und eine feingranulare Kontrolle über die Hardware bietet. Sie ist ideal für Nutzer, die eine tiefergehende Kontrolle und Verständnis der GPIO-Programmierung anstreben.

Installation von RPi:GPIO Die Anweisungen dafür finden Sie unter Installationen

Führen Sie den folgenden Code Schritt für Schritt.

[ ]:
```
#importieren von RPi.GPIO
import RPi.GPIO as GPIO
import time
```

[ ]:
```
# das GPIO-Modus setzen
GPIO.setmode(GPIO.BCM)
```

[ ]:
```
# den GPIO-Pin 17 als Ausgang setzen
GPIO.setup(17, GPIO.OUT)
```

[ ]:
```
# LED anschalten
GPIO.output(17, GPIO.HIGH)
time.sleep(1) # Eine Sekunde Warten
```

[ ]:
```
# LED ausschalten
GPIO.output(17, GPIO.LOW)
time.sleep(1) # Eine Sekunde warten
```

[ ]:
```
# Die GPIO-Einstellungen cleanen
GPIO.cleanup()
```

3.3.1 LED Beispiel 3

In diesem Beispiel geht es darum, mehrere LEDs blinken zu lassen. Hierbei wird die Bibliothek gpiozero genutzt.

3.3.2 Die Schaltung

Das Einbinden zweier LEDs sieht folgendermaßen aus:

Klarer könnte es so aussehen:

3.3.3 Installation der notwendigen Bibliotheken

Falls Sie gpiozero noch nicht installiert haben, können es mit folgendem Befehl installieren.

[ ]:
```
!pip3 install gpiozero
```

Importieren Sie die Bibliotheken als erstes.

[3]:
```
# importieren von gpiozero
from gpiozero import LED

# importieren von time
from time import sleep
```

Implementierung der Funktion, wie die LEDs ein und ausgeschaltet werden können.

[ ]:
```
# Code für ein- und ausschalten
led1 = LED(17)
led2 = LED(27)

try:
    while True:
        led1.on()
        led2.off()
        sleep(1)
        led1.off()
        led2.on()
        sleep(1)
except KeyboardInterrupt:
    led1.off()
    led2.off()
```

Im obigen Beispiel wurden die Pins 17 und 27 für die zwei in Reihe geschaltetenen LEDs genutzt. Eine While-Schleife dient dazu, dass die LEDs dauerhaft blinken bzw. bis sie unterbrochen werden.

3.3.4 Übung:

Überlegen Sie sich, wie Sie eine dritte LED in Reihe schalten können. Überlegen Sie sich, wie Sie die LEDs parallel schalten können.

3.4 7. Zusätzlich

Gucken Sie sich folgende Notebooks - Grundlagen-Notebook - Bib-Installationen - Raspbian OS

Bewegungserkennung mit Infrarot Motion Sensor

1. Einführung

PIR

PIR und LED

Projektziel

Das Ziel dieses Projekts ist es, mit dem PIR-Sensor HC-SR501 einen Bewegungssensor zu erstellen. Dieser Bewegungssensor aktiviert eine LED, wenn eine Bewegung in der Nähe erkannt wird. Das Projekt integriert sowohl Hardware (den PIR-Sensor und die LED) und Raspberry Pi als auch Software (Python-Programmierung über Jupyter Notebook) und bietet eine Einführung in die Programmierung und Steuerung von Sensoren mit dem Raspberry Pi.

Hintergrund

Der PIR-Sensor HC-SR501 (Passiver Infrarot-Sensor) wird häufig zur Bewegungserkennung verwendet. Er erkennt die von Menschen ausgestrahlte Infrarotstrahlung, die als Wärmesignatur detektiert wird. Wenn der Sensor eine plötzliche Änderung der Wärmestrahlung bemerkt, sendet er ein elektrisches Signal, um die Bewegung anzuzeigen.

Relevanz

Solche PIR-Sensoren werden in Anwendungen wie Sicherheitssystemen, automatischen Beleuchtungen und energiesparenden Einrichtungen eingesetzt. Dieses Projekt zeigt die grundlegende Funktionsweise eines Bewegungssensors und seine Integration in ein computergesteuertes System, was für viele Heimautomatisierungssysteme von Bedeutung ist.

2. Grundlagen und Theorie

PIR-Sensor HC-SR501:

Ein passiver Infrarotsensor, der Bewegungen durch Erkennung von Änderungen in der Infrarotstrahlung wahrnimmt. Der HC-SR501 verfügt über ein breites Erfassungsfeld (120 Grad) und kann Bewegungen in einem Bereich von 3 bis 7 Metern erkennen. Er gibt ein digitales Signal an den Raspberry Pi aus, wenn eine Bewegung erkannt wird.

Wie benutzt man diesen Sensor?

Der HC-SR501 PIR-Sensor kann entweder im nicht wiederholbaren Modus (L) betrieben werden, bei dem nach einer Bewegungserkennung ein High-Signal ausgegeben wird und keine weiteren Bewegungen erkannt werden, bis das Signal auf Low wechselt, oder im wiederholbaren Modus (H), bei dem Bewegungen kontinuierlich erfasst werden, solange das Objekt im Erfassungsbereich bleibt. Mit den Widerständen R1 und R2 lassen sich die Haltezeit des High-Signals (1,2 bis 320 Sekunden) und die Erfassungsreichweite (3 bis 5 Meter) anpassen.

- Freenove-Tutorial

LED:

Eine Leuchtdiode (LED) wird als visuelles Signal verwendet. Wenn der Sensor eine Bewegung registriert, leuchtet die LED als Indikator auf. Ein Widerstand wird verwendet, um den Stromfluss durch die LED zu begrenzen, um Schäden zu vermeiden.

Vorwiderstand:

Der Vorwiderstand für die LED begrenzt den Strom, um zu verhindern, dass die LED zu stark leuchtet oder beschädigt wird. Dieser wird auf Basis der LED-Daten berechnet.

3. Materialien und Werkzeuge

Software

- Raspbian OS
- Python 3
- Jupyter Notebook

Hardware

- Raspberry Pi
- PIR-Sensor HC-SR501
- LED
- Widerstand (330Ω)
- Breadboard und Jumperkabel
- GPIO Ribbonkabel

Sensor/(Aktor)en, inkl. Datenblätter

der HC-SR501 PIR Sensor wird als Sensor benutzt und die Daten im Datenblatt des Sensors lautet:

Spezifikation	HCR501 Sensor
Betriebsspannung	5V bis 20V
Strom	65 mA
Reichweite	3 bis 7 Meter
Erfassungswinkel	ca. 120°

Spezifikation	HCR501 Sensor

Berechnung des Vorwiderstands

der Vorwiderstand der LED kann wie folgt berechnet werden:

```
# Beispielcode zur Berechnung des Vorwiderstands:
# Werte: Versorgungsspannung 3.3V, LED-Spannung 2V, Strom 20mA
V_supply = 3.3  # Versorgungsspannung in Volt
V_f = 2.0  # Vorwärtsspannung der LED in Volt
I_f = 0.02  # Vorwärtsstrom der LED in Ampere

R = (V_supply - V_f) / I_f
print("Erforderlicher Vorwiderstand: {} Ohm".format(R))
```

```
Erforderlicher Vorwiderstand: 64.99999999999999 Ohm
```

Pin Beschaltung

Die Pin-Beschaltung kann für Beispeil wie flogt aussehen:

PIR Bewegungssensor	GPIO Pin des Raspberry Pi
VCC	Pin2 (5V)
GND	Pin6 (GND)
OUT	GPIO23, 17 oder 24

tutorials-raspberrypi.de

Die Beschaltung für Beispiel2 soll zusätzlich zu obiger Beschaltung soll wie folgt aussehen:

LED	GPIO Pin des Raspberry Pi
Anode	Pin11 GPIO 17
Kathode	Pin6 (GND)

Die Beschaltung für Beispiel3 soll zusätzlich zu obiger Beschaltungen soll wie folgt aussehen:

Buzzer	GPIO Pin des Raspberry Pi
Positiver Pin	Pin13 GPIO 27
Negativer Pin	GND

4. Schaltungsdesign

Schmatischer Entwurf

- Freenove-Tutorial

Hardware-Verbindung

tutorials-raspberrypi.de

Hardware-Verbindung für Beispiel 2

In diesem Bild ist der Anode von LED mit Pin 3 des Raspberry Pi verbunden.

Quelle

Hardware-Verbindung Beispiel 3

die HWVerbindung für Beispeil 3 kann sowas in Realität aussehen:

5. Implementierung

Hardware-Aufbau

Der Hardware soll wie obbiges Bild aufgabeut werden.

- HC SR501 Sensor mit dem Raspberry Pi verbinden.
- Raspberry Pi einschalten.

Software-Setup

1. Raspbian OS installieren.
2. Python und Jupyter Notebook installieren: ```sh sudo apt-get update sudo apt-get install python3 jupyter
3. Installation der Bibliotheken wie gpiozero und RPi.GPIO.

6. Experimente und Ergebnisse

Installation der Bibliotheken

```
!pip install ipython
```

ipython

```
pip install RPi.GPIO
```

RPi.GPIO

```
pip install gpiozero
```

gpiozero

Beispiel 1

Dieser Code gibt eine Meldung aus, wenn der PIR-Sensor eine Bewegung erkennt.

```
# Importieren Sie die Bibliotheken
from gpiozero import MotionSensor
from signal import pause

# Initialisiere den PIR-Sensor an GPIO 23 (BCM-Modus)
pir = MotionSensor(23)

# Funktionen, die auf Bewegung und keine Bewegung reagieren
def bewegung_erkannt():
    print("Bewegung erkannt!")

def keine_bewegung():
    print("Keine Bewegung mehr...")

# Binde die Funktionen an die Ereignisse
pir.when_motion = bewegung_erkannt
pir.when_no_motion = keine_bewegung

# Ausgabe im Jupyter-Notebook
print("Warte auf Bewegung...")

# Halt das Programm, damit es weiter auf Bewegung warten kann
pause()
```

Darstellung in Matplotlib

Aufgabe 1

Was bedeutet BCM-Modus ?

Aufgabe 2

Wie unterscheidet sich die Pin-Deklaration bei BCM und Board-Modus ?

Beispiel 2

Dieser Code schaltet die LED ein, wenn der PIR-Sensor eine Bewegung erkennt, und gibt eine Meldung aus.

Installation der Bibliotheken

```
# Importieren Sie die Bibliotheken
from gpiozero import MotionSensor, LED
from signal import pause
```

```
# Initialisiere den PIR-Sensor an GPIO 23 (BCM-Modus)
pir = MotionSensor(23)

# Initialisiere die LED an GPIO 17 (BCM-Modus)
led = LED(17)

# Funktionen, die auf Bewegung und keine Bewegung reagieren
def bewegung_erkannt():
    print("Bewegung erkannt!")
    led.on()  # LED einschalten, wenn Bewegung erkannt wird

def keine_bewegung():
    print("Keine Bewegung mehr...")
    led.off()  # LED ausschalten, wenn keine Bewegung erkannt wird

# Binde die Funktionen an die Ereignisse
pir.when_motion = bewegung_erkannt
pir.when_no_motion = keine_bewegung

# Ausgabe im Jupyter-Notebook
print("Warte auf Bewegung...")

# Halt das Programm, damit es weiter auf Bewegung warten kann
pause()
```

Beispiel 3

In diesem Beispiel wird neben der LED auch einen Buzzer verwendet. Der Buzzer wird aktiviert, wenn Bewegung erkannt wird, und deaktiviert, wenn keine Bewegung mehr erkannt wird.

```
# Importieren Sie die Bibliotheken
from gpiozero import MotionSensor, LED, Buzzer
from signal import pause

# Initialisiere den PIR-Sensor an GPIO 23 (BCM-Modus)
pir = MotionSensor(23)

# Initialisiere die LED an GPIO 17 (BCM-Modus)
led = LED(17)

# Initialisiere den Buzzer mit PWM an GPIO 27 (BCM-Modus)
buzzer = PWMOutputDevice(27)

# Funktionen, die auf Bewegung und keine Bewegung reagieren
def bewegung_erkannt():
    print("Bewegung erkannt!")
    led.on()  # LED einschalten
    buzzer.frequency = 2000  # Frequenz des Buzzers (2 kHz für
lauteren Ton)
    buzzer.value = 0.8  # Lautstärke erhöhen (0.8 entspricht 80% der
```

```
maximalen Leistung)

def keine_bewegung():
    print("Keine Bewegung mehr...")
    led.off()  # LED ausschalten
    buzzer.value = 0  # Buzzer ausschalten

# Binde die Funktionen an die Ereignisse
pir.when_motion = bewegung_erkannt
pir.when_no_motion = keine_bewegung

# Ausgabe im Jupyter-Notebook
print("Warte auf Bewegung...")

# Halt das Programm, damit es weiter auf Bewegung warten kann
pause()
```

7. Diskussion und Fazit

Diskussion

Der HC-SR501 PIR-Sensor hat eine zuverlässige Erkennungsreichweite und ist für viele Heimautomatisierungsanwendungen geeignet. Ein Punkt, der diskutiert werden könnte, ist die Möglichkeit, die Empfindlichkeit des Sensors zu ändern, um bestimmte Bewegungen besser oder schlechter zu erkennen.

Verbesserungsmöglichkeiten

- Integration eines Alarms oder einer Kamera für eine umfassendere Überwachung.
- Nutzung zusätzlicher PIR-Sensoren für eine breitere Bewegungserkennung.

Fazit

Das Projekt zeigt, wie PIR-Sensoren erfolgreich in einem einfachen Bewegungserkennungssystem integriert werden können. Der Raspberry Pi ermöglicht eine flexible Steuerung und Auswertung der Daten in einer komfortablen Entwicklungsumgebung wie Jupyter Notebook.

servomotor-projekt

November 13, 2024

1 Servo-Motor Steuerung mit Raspberry Pi und Jupyter Notebook

1.1 1. Einführung

Dieses Jupyter Notebook führt in die praktische Anwendung der Steuerung von Servomotoren mit einem Raspberry Pi ein. Es kombiniert Elektronik und Python-Programmierung, um grundlegende Steuerungen für Servomotoren zu erstellen und zu testen. Das Projekt zeigt, wie man einen analogen Servomotor mit verschiedenen Bibliotheken steuert, die Grundlagen des PWM-Signale erklärt und den Aufbau der Schaltung beschreibt. Mit einer schrittweisen Anleitung und praxisnahen Experimenten bietet dieses Notebook eine solide Grundlage für Einsteiger in die Welt der Elektronik und die Nutzung des Raspberry Pi für Hardware-Projekte.

1.1.1 Projektziel

Das Ziel dieses Projekts ist es, einen Servomotor mit einem Raspberry Pi zu steuern und dabei die Prinzipien der Pulsweitenmodulation (PWM) anzuwenden.

1.1.2 Relevanz

Dieses Projekt ist relevant, da es die Grundlagen der Elektronik und Programmierung kombiniert und ein grundlegendes Verständnis für die Steuerung von Motoren, insbesondere Servomotoren, entwickelt.

Blink in Projektskomponenten:

Quellen: Servo Motor, Raspberry Pi 5

Führen Sie die folgenden zwei Zellen, um Videos zu Servo in einem neuen Browser-Tab zu öffnen.

[2]:
```
import webbrowser
url = "https://www.youtube.com/embed/4RGCWnhy888?start=4"
webbrowser.open_new_tab(url)
```

[2]:
```
True
```

[3]:
```
url = "https://www.youtube.com/embed/xHDT4CwjUQE?start=10"
webbrowser.open_new_tab(url)
```

[3]: True

1.2 2. Grundlagen und Theorie

1.2.1 Servomotoren

- **Funktionsweise:** Servomotoren sind Aktoren, die eine präzise Steuerung der Winkelposition ermöglichen. Sie bestehen aus einem Motor, einem Potentiometer und einer Steuerungselektronik.
- **Steuerung:** Servomotoren werden typischerweise mit einem PWM-Signal gesteuert, wobei die Pulsbreite die Position des Servos bestimmt.

Weiteres zu Servomotoren

1.2.2 PWM-Signale

- **Definition:** PWM steht für Pulsweitenmodulation, eine Technik zur Steuerung der Leistung in elektronischen Schaltungen, indem das Ein/Aus-Verhältnis eines Signals variiert wird.
- **Anwendung bei Servos:** Die Position des Servos wird durch die Breite des PWM-Signals gesteuert, wobei typische Servos Pulse zwischen 1 ms (Min-Position) und 2 ms (Max-Position) erwarten.

PWM-Typ	Beschreibung	PINs
Hardware-PWM	Verwendung von Hardware zur Generierung von PWM-Signalen	PWM0: GPIO 12 (Pin 32), GPIO 18 (Pin 12) PWM1: GPIO 13 (Pin 33), GPIO 19 (Pin 35)
Software-PWM	Nutzung von Software zur Erzeugung von PWM-Signalen	GPIO 18: Häufig genutzt, viele Tutorials GPIO 23: Beliebter Pin für SW-PWM GPIO 24: Gut geeignet für mehr PWM-Kanäle

Arten von PWM: Hardware-PWM (HW-PWM)
- Definition: HW-PWM wird durch spezielle Hardware-Register des Mikrocontrollers (z. B. Raspberry Pi) gesteuert.
- Vorteile: Bietet eine präzise und stabile PWM-Signalausgabe. Unabhängig von anderen Prozessen, die auf dem Gerät laufen.
- Verwendung: Ideal für zeitkritische Anwendungen wie Motorsteuerung, bei denen genaue Pulsweiten erforderlich sind.

Software-PWM (SW-PWM)
- Definition: SW-PWM wird in der Software implementiert, indem Pins durch Programmierung mit sleep() ein- und ausgeschaltet werden.
- Nachteile: Geringere Präzision und Stabilität; kann durch andere laufende Prozesse beeinflusst werden.
- Verwendung: Nützlich für einfache Anwendungen oder wenn Hardware-PWM nicht verfügbar ist.

Weiteres zu PWM-Signals

1.3 3. Materialien und Werkzeuge

1.3.1 Software und Hardware

Kategorie	Komponenten
Software	Raspbian OS
	Python 3
	Jupyter Notebook
Hardware	Raspberry Pi 5 Model B Rev 1.0
	Breadboard
	Jumper-Kabel
	GRV Servo Motor
	5V Stromversorgung für den Servo

1.3.2 Datenblätter

Servo-Datenblatt:
- Drehmoment: 21.0/25.2oz.in - Geschwindigkeit: 0.12/10.1 s/60° - Spannung: 4.8-6V

Links zum Datenblatt Sowie Servo Motor:
- Servo Datenblatt - Gravo Servo

1.3.3 Verdrahtung

Verbindung	Raspberry Pi Pin	Servo-Anschluss
Steuerungs-Pin des Servos	GPIO 18 (PWM)	Steuerungs-Pin (Signal)
Stromversorgung (Vcc)	-	Externe 5V-Quelle
Masse (GND)	GND	GND

1.3.4 PWM: GPIO-Pins

PWM Kanal	GPIO Pin	Pin Nummer
PWM1	GPIO 12	Pin No 36
PWM2	GPIO 16	Pin No 33
PWM1	GPIO 18	Pin No 12
PWM2	GPIO 19	Pin No 35

Gucken Sie die komplette Erklärung von Pins hier

Übung In diesen Beispielen wurde der GPIO18 genutzt. Bauen Sie die Schaltung um, um PWM2 (GPIO13) zu nutzen. Zeigen Sie dem Dozent Ihre neue Schaltung und testen Sie es.

Tipp!

Gucken Sie sich das Pinout-Bild.

1.4 4. Schaltungsdesign

1.4.1 Raspberry P i5 GPIO-Pinout

Das folgende Bild zeigt die PINs und Protokollen eines Rasüpberry Pi und die wichtigsten Funktionen, die mit PINs verbunden sind.

1.4.2 Zu Servo Motor

Dieses Bild stellt die PINs des Servos sowie einige Spezifikationen dar.

1.4.3 Schaltplan

So könnte die Schaltung eines Servos mit Rasp Pi ausshen.

Erstellt durch circuito.io

Alternativ
Dieses Bild druckt die PINs-Schaltung deutlisch klar aus.

Erstellt durch easyeda

1.4.4 Geschwindigkeit messen

Servo Motor kann mit einem Zeiger als Geschwindigkeit-Messer genutzt werden.
Im folgenden Bild ist die Anzeige der Geschwindigkeit genauso groß wie der Bewegungsgrad des Servo Motors, nämlich 180.

1.4.5 Komponentenauswahl

Komponente	Beschreibung
Servomotor	GRV Servo 21.0/25.2oz.in 0.12/10.1 s/60°
Jumper-Kabel	Für die Verbindung der GPIO-Pins mit dem Servo
Breadboard	Für die Schaltungsaufbauten

Für Weiteres über die Verwendung von Servo mit Rasp Pi klicken Sie hier.

1.5 5. Implementierung

1.5.1 Hardware-Aufbau

1. Komponenten auf dem Breadboard wie oben beschrieben platzieren und verkabeln.
2. Raspberry Pi aufstellen und durch USB-Kabel an Rechner mit Strom versorgen.

1.5.2 Software-Setup

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

1.5.3 Installiere die RPi.GPIO und gpiozero Bibliotheken:

Für die Installationsanweisungen dieser Bibliotheken Klicken Sie hier

[ ]:
```
from gpiozero import Servo
from time import sleep

servo = Servo(18) #GPIO 18

try:
    while True:
        servo.mid()   # Mittelposition
        sleep(2)
        servo.min()   # Minimum Position
        sleep(2)
        servo.max()   # Maximum Position
        sleep(2)
except KeyboardInterrupt:
    pass
```

1.5.4 Funktionen für den Servomotor

Funktion	Beschreibung
`servo.mid()`	Stellt den Servomotor in die mittlere Position, also auf den mittleren Punkt seines Bewegungsbereichs.
`servo.min()`	Bewegt den Servomotor in die minimale Position, also den unteren Extrempunkt seines Bewegungsbereichs.
`servo.max()`	Bewegt den Servomotor in die maximale Position, also den oberen Extrempunkt seines Bewegungsbereichs.

Für weiteres über Servo-Bibliotheken und Funktionen klicken Sie hier.

Diese Impelimentierung lässt der Zeiger dies Servos auf drei Positionen bewegen, nämlich Mittelposition, Minimum Position und Maximum Position. Auch hier wurde der GPIO 18 benutzt.

Übung

- Mid, Min und Max wurden oben behandelt. Welche weitere Funktionen bietet uns Servo?
- Schreiben Sie diesen Beispiel nochmal, aber dieses Mal mit der Nutzung von RPi.GPIO Bibliothek.
- Anschließend testen Sie Ihren Code mit Rasp Pi. Funktioniert es noch wie vorhin? Zeigen Sie dem Dozent Ihre Änderungen.
- Welcher Wert (in [ms]) für die mittlere Position steht?

Lösung!

- servo.detach(): Dies deaktiviert den Servo und spart Energie, wenn er nicht benutzt wird.

- servo.value: Ein Attribut, das den aktuellen Wert des Servos (zwischen -1 und 1) zurückgibt, wo:

  -1 für die minimale Position,
  0 für die mittlere Position und
  1 für die maximale Position steht.

Für die mittlere Position eines Standard-Servomotors liegt der Puls bei ca. 1,5 ms. Dies entspricht typischerweise einem Duty Cycle von 7,5% bei einer 50 Hz PWM-Frequenz.

```
Min-Position: 1 ms (2% Duty Cycle)
Max-Position: 2 ms (12% Duty Cycle)
Mid-Position: 1,5 ms (7,5% Duty Cycle)
```

“‘bash import RPi.GPIO as GPIO from time import sleep

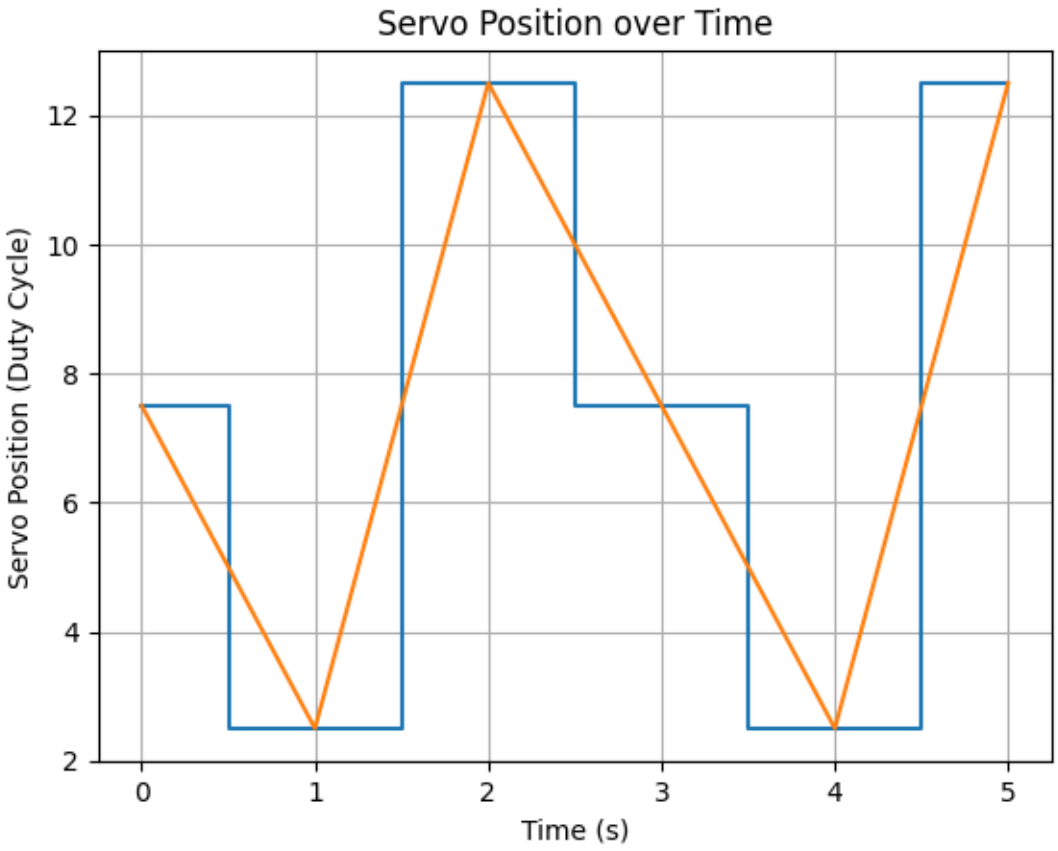
```
# Konfiguration der GPIO-Pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)  # Pin 18

# Servo initialisieren
pwm = GPIO.PWM(18, 50)  # 50 Hz
pwm.start(0)

# Funktion zur Steuerung des Servowinkels
def set_angle(angle):
    duty = angle / 18 + 2
    GPIO.output(18, True)
    pwm.ChangeDutyCycle(duty)
    sleep(1)  # Warte, bis der Servo sich bewegt hat
    GPIO.output(18, False)
    pwm.ChangeDutyCycle(0)

try:
    while True:
        angle = float(input("Geben Sie den gewünschten Winkel (0-180) ein: "))
        set_angle(angle)

except KeyboardInterrupt:
    pwm.stop()
    GPIO.cleanup()
```

1.5.5 Darstellung in Matplotlib

Matplotlib kann verwendet werden, um die PWM-Signale oder die Bewegungsposition des Servos über die Zeit darzustellen.
Falls ie diese Bibliothek noch nicht installiert haben, Gucken Sie sich die Installation hier.

[8]:
```
import matplotlib.pyplot as plt
```

```
times = [0, 1, 2, 3, 4, 5]
positions = [7.5, 2.5, 12.5, 7.5, 2.5, 12.5]

plt.step(times, positions, where='mid')  # 'mid', 'pre', or 'post' kann␣
 ↪verwendet werden
plt.plot(times, positions)
plt.xlabel('Time (s)')
plt.ylabel('Servo Position (Duty Cycle)')
plt.title('Servo Position over Time')
plt.grid(True)
plt.show()
```



Die `step`-Funktion in Matplotlib ermöglicht es, die Linien zwischen den Datenpunkten entweder vertikal oder horizontal verlaufen zu lassen, anstatt diagonal. Sie können den Stil der Treppenlinien mit dem Parameter where steuern, wobei `mid`, `pre`, und `post` verschiedene Varianten darstellen.

1.5.6 Schritte in einem Diagramm

Schritt	Beschreibung
`pre`	Der Schritt in einem Diagramm beginnt direkt vor dem x-Wert.

Schritt	Beschreibung
`post`	Der Schritt beginnt direkt nach dem x-Wert.
`mid`	Der Schritt erfolgt in der Mitte zwischen zwei x-Werten.

Weiteres zu `step`- und Matplotlib-Funktionen klicken Sie hier.

1.5.7 Übung:

- Finden Sie heraus, welche andere Funktionen die Matplotlib anbietet?
- Im obigen Beispiel wurde der Servo zwischen drei Positionen bewegt.
  Erweitern Sie das Beispiel, um eine langsamere Bewegung zwischen den Positionen zu simulieren.

  Tipp!
  Nutzen Sie de Funktion sleep(x), wobei x die Schlafzeit ist.

1.6 6. Experimente und Ergebnisse

1.6.1 Versuchsaufbau

- Der oben beschriebene Hardware-Aufbau wurde verwendet.
- Der Code wurde in einem Jupyter Notebook ausgeführt.
- Der Servo bewegte sich entsprechend den vorgegebenen Positionen.

1.6.2 Beispiel 1: Schrittweise-Bewegung nach Angabe von Winkel

In diesem Beispiel wurde die Bibliotheken RPi.GPIO und time, die es schon mit Python installiert.

Sehen Sie sich auch The Top 5 Libraries for Every Developer

[ ]:
```
#Falls noch nicht installiert führen Sie diesen Befehl aus.
!pip3 install RPi.GPIO
```

[ ]:
```
#Bibliotheken importieren
import RPi.GPIO as GPIO
from time import sleep
```

[ ]:
```
# Konfiguration der GPIO-Pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)  # Pin 18
```

[ ]:
```
# Servo initialisieren
pwm = GPIO.PWM(18, 50)  # 50 Hz
pwm.start(0)
```

Da wir die GPIO.PWM-Klasse verwenden, nutzen wir Hardware-PWM (HW-PWM). Diese Methode ist stabiler und ermöglicht präzisere Steuerungen im Vergleich zur Software-PWM.

[ ]:
```
#Motor-Bewegung durch Eingabe von Winkel
def set_angle(angle):
    duty = angle / 18 + 2
    GPIO.output(18, True)
    pwm.ChangeDutyCycle(duty)
    sleep(1)  # Warte, bis der Servo sich bewegt hat
    GPIO.output(18, False)
    pwm.ChangeDutyCycle(0)
```

ChangeDutyCycle ist eine Methode, die in der Bibliothek RPi.GPIO verwendet wird, um die Duty-Cycle-Eigenschaft eines PWM-Signals zu ändern. Die Duty-Cycle beschreibt das Verhältnis von Ein- zu Ausschaltzeiten eines PWM-Signals (Pulsweitenmodulation). Für weiteres zu `RPi.GPIO` klicken Sie hier

[ ]:
```
try:
    while True:
        try:
            # Benutzereingabe für den gewünschten Winkel
            angle = float(input("Geben Sie den gewünschten Winkel (0-180) ein␣
 ↪(oder 'exit' zum Beenden): "))

            # Eingabewert überprüfen
            if angle < 0 or angle > 180:
                print("Ungültiger Winkel. Bitte geben Sie einen Wert zwischen 0␣
 ↪und 180 ein.")
                continue

            # Servo auf den angegebenen Winkel bewegen
            set_angle(angle)

        except ValueError:
            user_input = input("Ungültige Eingabe. Geben Sie 'exit' zum Beenden␣
 ↪ein: ")
            if user_input.lower()== 'exit':
                break
            else:
                print("Ungültige Eingabe. Bitte geben Sie einen Wert zwischen 0␣
 ↪und 180 ein.")

except KeyboardInterrupt:
    print("Programm wurde unterbrochen.")

finally:
    pwm.stop()
    GPIO.cleanup()
```

In diesem Beispiel wird man bei jedem Schritt danach gefordert, durch die Tastatur einen Winkel einzugeben. Der Servo bewegt sich dann aus der aktuellen Posiotion auf die neue durch den Winkel

angegebene Posiotion. Falls der Nutzer einen Wert größer 180, was der größtmögliche Winkel des Motors ist, oder kleiner 0, was das Minimum des Servo ist, kriegt man eine Ausgabe auf Jupyter, dass eine ungültige Eingabe erfolgt wurde. Motor macht dann nichts und wartet auf eine neue Eingabe.

Übung - Welcher Wert (in [ms]) steht für einen Winkel von 80 Grad?

Lösung! Der Wert für 80 Grad kann folgendermaßen berechnet werden:

Formel für den Duty Cycle: Duty=(angle/18)+2 Duty=18angle+2

Für 80 Grad: Duty=(80/18)+2=6.44 ms

1.6.3 Beispiel 2: dauerhafte Bewegung

Wir nutzen die gleiche Bibliotheken und PIN-Einstellungen wie oben. Merken Sie sich hierbei den Unterschied zwischen dem obigen und dem folgenden Code.

[ ]:
```
if __name__ == '__main__':
    degree = 180 #Maximum von Servo
    direction = 0 #Richtung
    try:
        while True: #Das heißt: dauerhafte Bewegung
            set_angle(degree) #Aufrufen der obigen Funktion im Beispiel 1
            print(degree) #Ausgabe, für Klarheit
            # Einstellung von Bewegungsrichtung anhand des Winkels.
            if degree == 180:
                direction = 1
            elif degree == 0:
                direction = 0

            if direction == 0:
                degree += 10
            else:
                degree -= 10
    except KeyboardInterrupt:
        pwm.stop()
        GPIO.cleanup()
```

In diesem Beispiel bewegt sich der Zeiger bzw. dreht sich der Motor ständig ohne anzuhalten. Der Servo kann nur zwischen Winkel 0 und 180. In diesem Fall bewegt sich er von 0 auf 180. Wenn der Servo 180 erreicht, bewegt sich zurück auf 0 usw. . Im Beispiel kann man bemerken, dass die Variable degree sich immer um 10 erhöht oder verkleinert. Dies ist die Größe eines Schritts des Servos. 10 wurde mit Absicht eingegeben, damit sich der Servo während Experimenten nicht sehr langsam bewegt.

1.6.4 Übung:

- Suchen Sie sich einen anderen Pin, wo Sie den Servo mit einem anderen PWM-Pin verbinden und passen Sie den Code an. Funktioniert der Servo immer noch?

- Tipp!

  Gucken Sie sich das Pinout-Bild.

- Ändern Sie den Code so, dass der Servo nur 4 Schritte zwischen 0 und 180 machen kann. Wie machen Sie das?

- Lösung!

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

pwm = GPIO.PWM(18, 50)  # 50 Hz
pwm.start(0)

def set_angle(angle):
    duty = angle / 18 + 2
    GPIO.output(18, True)
    pwm.ChangeDutyCycle(duty)
    sleep(1)
    GPIO.output(18, False)
    pwm.ChangeDutyCycle(0)

try:
    steps = [0, 60, 120, 180]
    while True:
        for angle in steps:
            print(f"Bewege Servo auf {angle} Grad")
            set_angle(angle)
            sleep(2)

except KeyboardInterrupt:
    pwm.stop()
    GPIO.cleanup()
```

“‘

1.6.5 Zusätzliche Ressourcen

Für besseres Verstand und für die Klarheit gucken Sie sich die folgenden Notebooks:
- Grundlagen-Notebook - Bib-Installationen - Raspbian OS

Sowie die Webseiten: - jupyter.org - mybinder

Temperatursensor-Projekt

1. Einführung

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.

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.

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.

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.

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.

Nutzliche Links zu MEMS-Technologie

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

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.

Nutzliche Link zu SPI und I2C

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

3. Materialien und Werkzeuge

Zusätzliche Software

- Raspbian OS
- Python 3
- Jupyter Notebook
- Bibliotheken

Zusätzliche Hardware

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

Einführung in Jupyter Notebook und Raspberry Pi

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

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

Aufgabe

Wie kann man Versorgungspannung berechnen ?

Datenblatt

- BME680

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.

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

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

Zusätzliche Berechnungen

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

Komponentenauswahl

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

4. Schaltungsdesign

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.

5. Implementierung

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.

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
```

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
    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

6. Experimente und Ergebnisse

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.

Nutzliche Links zu Matplotlib

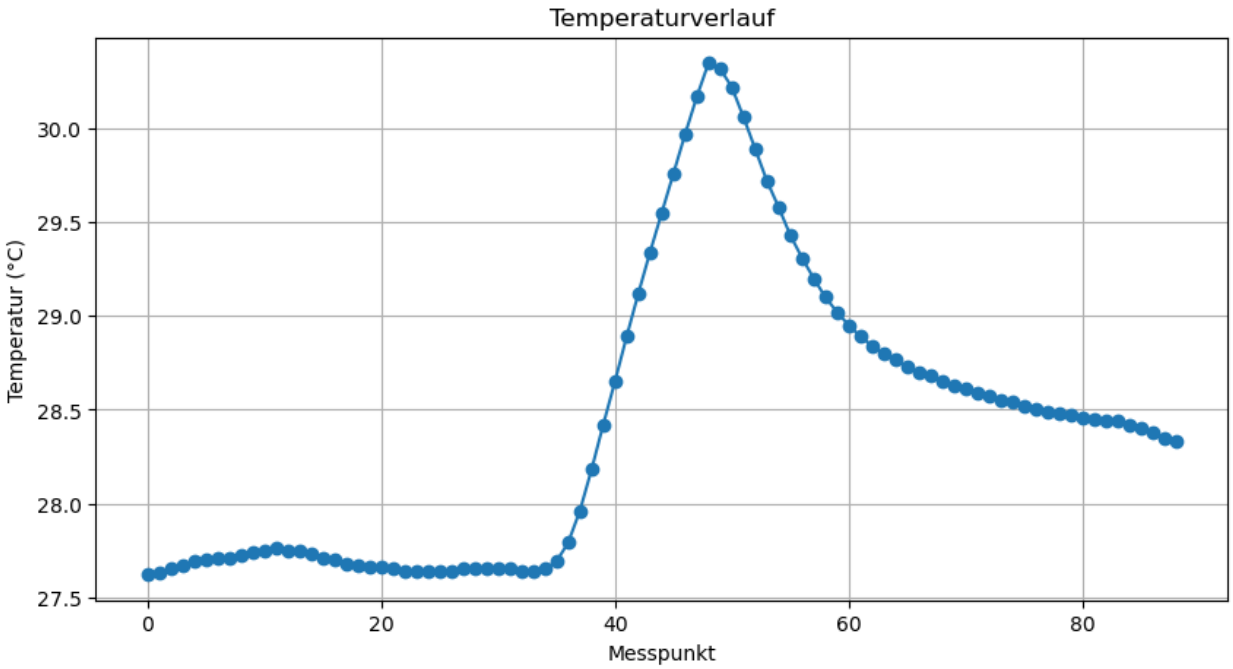
- Pyplot tutorial
- Quick start guide

Nutzliche Links zu Pandas

- Python pandas tutorial
- User Guide

```
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.

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.

Code zum Testen

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

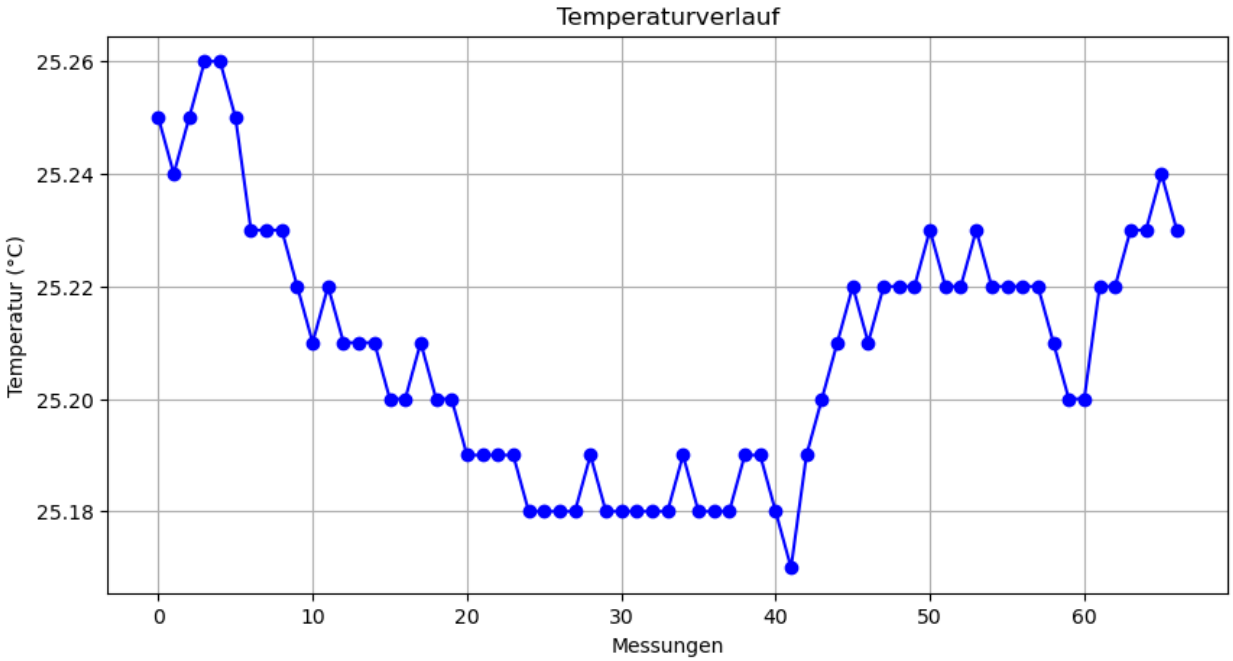
```
### Biblliotheken importieren
import time

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)
```

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.

```
import matplotlib.pyplot as plt
import statistics
```

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



Übung

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

7. Diskussion und Fazit

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.

raspberrypi-jupyter-grundlagen

November 13, 2024

1 Einführung in Jupyter Notebook und Raspberry Pi

1.1 Jupyter Notebook

Jupyter Notebook ist eine Open-Source-Webanwendung, die es ermöglicht, Dokumente zu erstellen und zu teilen, die Live-Code, Gleichungen, Visualisierungen und erläuternde Texte enthalten. Es wird häufig in wissenschaftlichen und datenwissenschaftlichen Bereichen verwendet, da es eine interaktive Umgebung bietet, in der Benutzer Code schreiben und sofortige Rückmeldungen erhalten können.

1.1.1 Funktionen von Jupyter Notebook

- Interaktive Entwicklung: Ermöglicht das Schreiben und Ausführen von Codezellen, wodurch sofortige Ergebnisse sichtbar werden.
- Integration von Text, Code und Visualisierungen: Ermöglicht das Einbetten von Diagrammen, Grafiken und formatiertem Text, um Ergebnisse klar zu präsentieren.
- Unterstützung mehrerer Programmiersprachen: Obwohl es hauptsächlich für Python verwendet wird, unterstützt Jupyter Notebook auch andere Sprachen wie R, Julia und mehr durch die Verwendung von Kernels.

1.1.2 Vorteile von Jupyter Notebook

- Leicht zu bedienen: Die Benutzeroberfläche ist intuitiv und erleichtert das Erlernen und Verwenden.
- Kollaborativ: Dokumente können geteilt und gemeinsam bearbeitet werden, was die Zusammenarbeit fördert.
- Erweiterbar: Eine Vielzahl von Erweiterungen und Plugins kann installiert werden, um die Funktionalität zu erweitern.

1.1.3 Nützliche Links für Jupyter Notebook

Offizielle Jupyter Webseite
Einführung zu Jupyter Notebooks auf GitHub
Jupyter Notebook-Dokumentation

1.2 Raspberry Pi

1.2.1 Was ist ein Raspberry Pi?

Der Raspberry Pi ist ein kostengünstiger, kreditkartengroßer Einplatinencomputer, der für Bildungszwecke entwickelt wurde, sich aber auch als leistungsfähiges Werkzeug für zahlreiche Elektronikprojekte etabliert hat. Er wird häufig verwendet, um Grundlagen der Programmierung, Elektronik und Robotik zu lehren und zu lernen.

1.2.2 Spezifikationen des Raspberry Pi 5

- Prozessor: Quad-Core 64-bit ARM Cortex-A72
- Arbeitsspeicher: 4GB RAM
- Anschlüsse: GPIO-Pins, USB-Anschlüsse, HDMI, Audio, Ethernet, Kamera und Display Schnittstellen
- Betriebssystem: Raspbian OS (ein Debian-basiertes Linux-Betriebssystem)

1.2.3 Vorteile des Raspberry Pi

- Erschwinglich: Kostengünstig und daher ideal für Bildungszwecke und Hobbyprojekte.
- Vielseitig: Kann in einer Vielzahl von Projekten verwendet werden, von einfachen Elektronikschaltungen bis hin zu komplexen IoT-Anwendungen.
- Große Community: Eine aktive Gemeinschaft bietet Unterstützung, Tutorials und Projekte, die es einfach machen, Hilfe zu finden und zu lernen.

1.2.4 Einsatzmöglichkeiten des Raspberry Pi

- Bildung: Programmier- und Elektronikunterricht.
- Hausautomation: Steuerung von Haushaltsgeräten und Überwachungssystemen.
- Mediacenter: Einsatz als Mediaplayer mit Software wie Kodi.
- Prototyping: Schnelles und kostengünstiges Entwickeln und Testen von Prototypen für verschiedene Anwendungen.

1.2.5 Nützliche Links für Raspberry Pi

Offizielle Raspberry Pi Webseite
Raspberry Pi Projekte auf GitHub
Raspberry Pi Dokumentation

1.3 Fazit

Das Zusammenspiel von Jupyter Notebook und Raspberry Pi bietet eine leistungsstarke Kombination für die Entwicklung und Durchführung von Elektronik- und Programmierprojekten. Während Jupyter Notebook eine interaktive Plattform für die Entwicklung und Visualisierung von Code bietet, stellt der Raspberry Pi die Hardwarebasis bereit, um reale physikalische Schaltungen zu steuern und zu testen. Durch die Kombination dieser beiden Werkzeuge können Lernende und Entwickler ein tieferes Verständnis für die Programmierung und Elektronik erlangen und innovative Projekte umsetzen.

rasp-pi-einrichtung

November 13, 2024

1 Anleitung zur Installation von Raspberry Pi OS

Diese Anleitung beschreibt die Schritte zur Installation von Raspberry Pi OS (ehemals Raspbian OS) auf einem Raspberry Pi. Sie werden lernen, wie Sie das Betriebssystem herunterladen, auf einer SD-Karte installieren und den Raspberry Pi einrichten.

1.1 1. Voraussetzungen

- **Raspberry Pi**: Ein Raspberry Pi Modell (z. B. Raspberry Pi 5 Model B Rev 1.0).
- **SD-Karte**: Mindestens 16 GB, aber 32 GB oder mehr wird empfohlen. Stellen Sie sicher, dass die SD-Karte leer ist oder sicheren Sie Ihre Daten vorher.
- **SD-Kartenleser**: Ein Adapter oder integrierter Leser, um die SD-Karte mit Ihrem Computer zu verbinden.
- **Computer**: Windows, macOS oder Linux-Computer, um die SD-Karte vorzubereiten.
- **Stromversorgung und Kabel**: Um den Raspberry Pi mit Strom zu versorgen.
- **Monitor und HDMI-Kabel**: Zum Anzeigen der Raspberry Pi-Ausgabe.
- **Tastatur und Maus**: Zum Einrichten des Raspberry Pi.

1.2 2. Herunterladen des Raspberry Pi Imager

1. **Besuchen Sie die offizielle Raspberry Pi-Website**: Gehen Sie zu Raspberry Pi Downloads.
2. **Raspberry Pi Imager herunterladen**: Wählen Sie das passende Installationsprogramm für Ihr Betriebssystem (Windows, macOS oder Ubuntu) und laden Sie es herunter.
3. **Installieren Sie den Raspberry Pi Imager**: Folgen Sie den Anweisungen des Installationsprogramms, um den Raspberry Pi Imager auf Ihrem Computer zu installieren.

1.3 3. SD-Karte vorbereiten

1. **Starten Sie den Raspberry Pi Imager**: Führen Sie das Programm auf Ihrem Computer aus.
2. **Betriebssystem auswählen**: Klicken Sie auf “Choose OS” und wählen Sie “Raspberry Pi OS (32-bit) oder (64-bit), abhängig von Ihrem Computer”. Es gibt auch Lite- und andere spezialisierte Versionen, je nach Bedarf.
3. **SD-Karte auswählen**: Klicken Sie auf “Choose Storage” und wählen Sie Ihre SD-Karte aus.
4. **Schreiben starten**: Klicken Sie auf “Write”, um das Betriebssystem auf die SD-Karte zu schreiben. Bestätigen Sie alle Warnungen und warten Sie, bis der Vorgang abgeschlossen ist.

1.4 4. SD-Karte in den Raspberry Pi einlegen

1. **Entfernen Sie die SD-Karte**: Sobald der Imager meldet, dass der Schreibvorgang abgeschlossen ist, entfernen Sie die SD-Karte sicher von Ihrem Computer.
2. **SD-Karte einlegen**: Stecken Sie die SD-Karte in den entsprechenden Slot Ihres Raspberry Pi.

1.5 5. Raspberry Pi starten und konfigurieren

1. **Verkabelung**: Verbinden Sie den Raspberry Pi mit dem Monitor, der Tastatur und der Maus. Schließen Sie die Stromversorgung an.
2. **Erster Start**: Schalten Sie den Raspberry Pi ein. Das Betriebssystem wird nun gebootet.
3. **Ersteinrichtung**: Folgen Sie den Anweisungen auf dem Bildschirm, um Sprache, Zeitzone und Tastaturlayout einzustellen.
4. **Netzwerkverbindung**: Stellen Sie eine Verbindung zu Ihrem WLAN her oder verwenden Sie ein Ethernet-Kabel.
5. **Aktualisierungen durchführen**: Während der Einrichtung werden Sie aufgefordert, das System zu aktualisieren. Es wird empfohlen, diese Updates durchzuführen.

1.6 6. Zusätzliche Einstellungen

- **SSH aktivieren**: Gehen Sie zu "Raspberry Pi Configuration" und aktivieren Sie SSH, wenn Sie den Raspberry Pi ohne Monitor oder Tastatur fernsteuern möchten.
- **VNC aktivieren**: Sie können auch VNC aktivieren, um eine Remote-Desktop-Verbindung herzustellen.
- **Paketmanager verwenden**: Verwenden Sie `apt` im Terminal, um zusätzliche Software zu installieren.

1.6.1 Virtuelle Umgebung (venv) erstellen und aktivieren

Falls Sie noch keine virtuelle Umgebung erstellt haben, können Sie dies mit den folgenden Schritten tun:

- Öffnen Sie das Terminal auf Ihrem Raspberry Pi.
- Erstellen Sie eine neue virtuelle Umgebung mit Python:
- `python3 -m venv venv` Hierbei wird die virtuelle Umgebung in einem Ordner namens venv erstellt.
- Aktivieren Sie die virtuelle Umgebung:
- `source venv/bin/activate` Sobald die virtuelle Umgebung aktiviert ist, wird der Name der Umgebung (venv) in Ihrem Terminal angezeigt. Sie arbeiten nun innerhalb dieser Umgebung.

Installieren Sie Jupyter Notebook (falls noch nicht installiert):

- `pip install jupyter`

1.7 7. Abschließende Schritte

- **Sicherung der SD-Karte**: Es kann hilfreich sein, ein Backup der SD-Karte zu erstellen, nachdem Sie die Grundkonfiguration abgeschlossen haben.

- **Dokumentation lesen**: Besuchen Sie die offizielle Raspberry Pi-Dokumentation, um mehr über die Nutzung Ihres Raspberry Pi zu erfahren.

Mit diesen Schritten sollte Ihr Raspberry Pi mit Raspberry Pi OS erfolgreich eingerichtet und bereit sein, für verschiedene Projekte verwendet zu werden.

raspberry-pi-kommunikation

November 13, 2024

1 Kommunikation mit Raspberry Pi

1.1 Steuerung durch SSH Verbindung

Schritt 1:

SSH auf dem Raspberry Pi aktivieren
- Verbinden Sie sich direkt mit Ihrem Raspberry Pi (Monitor und Tastatur) oder greifen Sie über ein anderes Medium zu.
- Öffnen Sie das Terminal und geben Sie ein:

```bash
sudo raspi-config
```

- Navigieren Sie zu **Interfacing Options > SSH** und wählen Sie **Enable**.

Schritt 2:

SSH-Client auf Windows verwenden
Auf Ihrem Windows Rechner gehen Sie die folgenden Anweisungen durch.

- CMD öffnen

- Folgendes im Terminal bzw. im CMD eingeben:
  `bash ssh <username>@<raspPi-IP>`
  In meinem Fall ist das:
  `bash ssh ashraf@192.168.0.104`

- Passwort eingeben

- Navigieren Sie zu Ihrem Notebook
- Folgendes im Terminal bzw. im CMD eingeben:
  `bash ipython`
  `bash %run your_script.ipynb`
  In meinem Fall ist das dann:
  `bash %run 1_LED_Morsecode.ipynb`
  So wird der Code auf Ihrem Raspberry Pi ausgeführt.

1.2 Steuerung durch Netzwerkfreigabe

Steuern Sie Ihr Jupyter Notebook und somit das Raspberry Pi aus Ihrem eigenen Rechner, der sich im gleichen Netzwerk befindet.
Gehen Sie die folgenden Schritte durch.

- Zur virtuellen Umgebung wechseln, indem Sie im Terminal folgendes eingeben:
  `bash  cd venv`
  Falls Sie keine Virtuelle Umgebung installiert haben, gucken Sie sich die Anweisungen hier unter zusätzliche Einstellungen `bash  source bin/activate`

- Nutze den Befehl für die Freigabe im Netzwerk:
  `bash  jupyter notebook --ip <your_LAN_ip> --port 8888`
  In meinem Fall ist das:
  `bash  jupyter notebook --ip 192.168.0.104 --port 8888`

- So ermitteln Sie Ihre IP in RaspPi Terminal:
  `bash  hostname -I`

- Im eigenen Rechner in Browser folgendes eingeben:
  `plaintext  http://your_LAN_ip:8888`

- Passwort eingeben, falls nicht vorhanden, geben Sie im Terminal von RaspPi folgendes ein:
  `bash  jupyter notebook password`
  - Passwort eingeben und bestätigen

- Gehen Sie zurück zu Ihrem Browser und geben Sie das Passwort ein.

- Für weiteres folgendes gucken:
  Stack Overflow

1.3 Steuerung durch Kombination von SSH und Freigabe im Netzwerk

Gehen Sie bitte die folgenden Schritte durch:

- CMD öffnen

- Folgendes im Terminal bzw. im CMD eingeben:
  `bash  ssh <username>@<raspPi-IP>`

- Passwort eingeben

- Zur virtuellen Umgebung wechseln, indem Sie im Terminal folgendes eingeben:
  `bash  cd venv`
  `bash  source bin/activate`

- Nutze den Befehl für die Freigabe im Netzwerk:
  `bash  jupyter notebook --ip <your_LAN_ip> --port 8888`

- Im eigenen Rechner in Browser folgendes eingeben:
  `plaintext  http://your_LAN_ip:8888`
  In meinem Fall ist das:
  `plaintext  http://192.168.0.104:8888`

Vergewissern Sie sich, dass die virtuelle Umgebung `venv` heißt oder passen Sie anhand dessen die obigen Schritte an.

2 Steuerung durch Access Point

Gucken Sie sich dieses Notebook

einf195188hrung-in-matplotlib

November 13, 2024

1 Einführung in Matplotlib

1.0.1 1. Was ist Matplotlib ?

Matplotlib ist eine weit verbreitete Python-Bibliothek zur Visualisierung von Daten. Es ermöglicht Benutzern, ansprechende und leicht interpretierbare Diagramme zu erstellen, die dabei helfen, komplexe Daten besser zu verstehen. Matplotlib ist besonders bei Wissenschaftlern, Ingenieuren und Datenanalysten beliebt, da es eine einfache Möglichkeit bietet, Daten zu visualisieren.

1.0.2 2. Vorteile von Matplotlib

Vorteil	Beschreibung
Einfach zu erlernen	Matplotlib hat eine intuitive Syntax, die besonders für Anfänger leicht verständlich ist.
Vielseitigkeit	Von einfachen Linienplots bis hin zu komplexen 3D-Diagrammen unterstützt Matplotlib eine Vielzahl von Diagrammtypen.
Anpassbarkeit	Benutzer können fast jedes Detail eines Diagramms anpassen, von Farben über Achsenbeschriftungen bis hin zu Linienbreiten.
Kompatibilität	Matplotlib funktioniert hervorragend mit anderen Python-Bibliotheken wie NumPy, Pandas und Seaborn, was die Datenvisualisierung für komplexe Analysen erleichtert.
Vielseitiges Ausgabeformat	Plots können in verschiedenen Formaten gespeichert werden, was nützlich für Publikationen oder Präsentationen ist.

1.0.3 3. Installation und erste Schritte mit Matplotlib

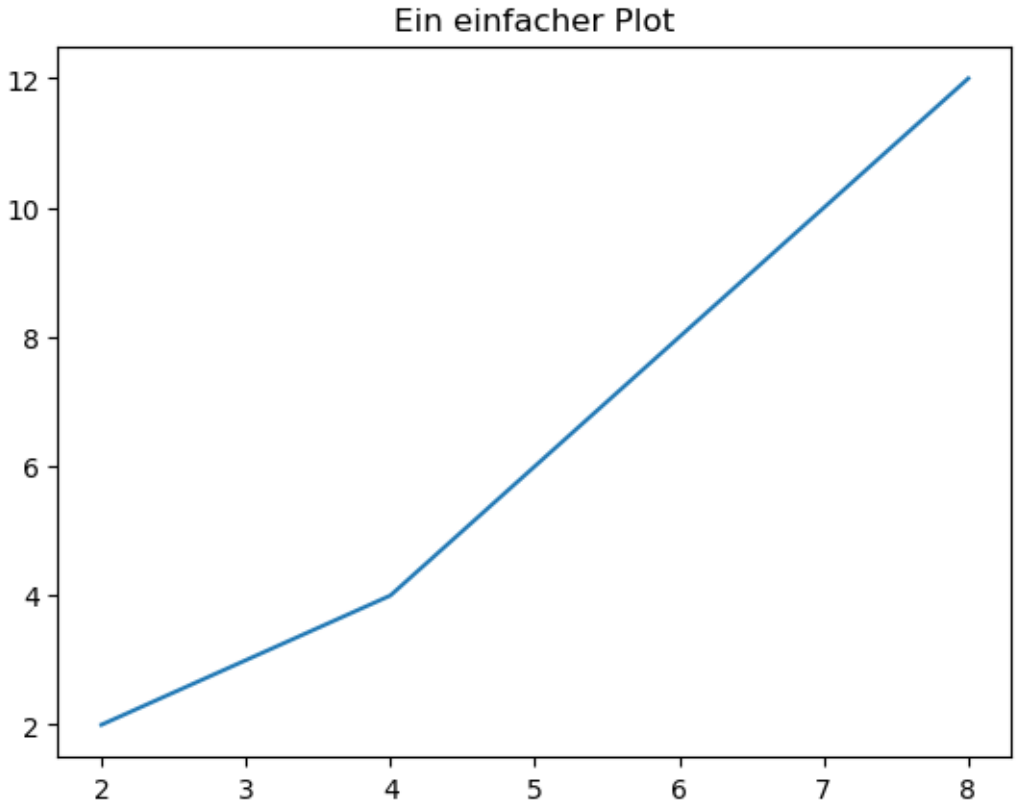
3.1 Installation in Jupyter Notebook:
in einem Jupyter Notebook kann Matplotlib mit folgendem Befehl installiert werden: `!pip install matplotlib`

3.2 Installation in Python:
Matplotlib kann in Python mit folgendem Befehl installiert werden: `pip3 install matplotlib`

Bevor Sie den `pip` Befehl nutzen, stellen Sie sicher, das Sie `pip` mit folgendem Befehl `sudo apt install python3-pip` installiert haben.

3.3 erster Schritt in Matplotlib:
Es ist ein einfaches Beispiel für Darstellung eines Plots in Matplotlib:

[44]:
```
### Imporieren Sie die Bibliotheken
import matplotlib.pyplot as plt

# Einfache Linie zeichnen
plt.plot([2, 4, 6, 8], [2, 4, 8, 12])
plt.title("Ein einfacher Plot")
plt.show()
```



1.0.4 3.4 Erklärung des Codes

Code	Beschreibung
`import matplotlib.pyplot as plt`	Importiert die pyplot-Funktionen von Matplotlib, um einfache Plot-Befehle zu verwenden.
`plt.plot()`	Erstellt einen Linienplot basierend auf den angegebenen x- und y-Daten.
`plt.title()`	Fügt einen Titel zum Diagramm hinzu.

Übung

finden Sie heraus, was plt.show() im Code macht !

1.0.5 4.Funktionen von Matplotlib

Die typischen Funktionen von Matplotlib können wie folgt gelistet werden:

Funktion	Beschreibung
Vielfältige Diagrammtypen	Matplotlib unterstützt eine Vielzahl von Diagrammtypen, darunter Linien-, Balken-, Streu- und Kreisdiagramme sowie 3D-Visualisierungen.
Anpassung	Die meisten Aspekte eines Diagramms können detailliert angepasst werden (z.B. Titel, Achsen, Farben).
Mehrere Plots	Mit `subplots()` können mehrere Diagramme auf einer Abbildung erstellt werden, was es einfach macht, verschiedene Visualisierungen nebeneinander darzustellen.
Interaktive Funktionen	In Kombination mit Jupyter Notebooks bietet Matplotlib interaktive Diagramme, bei denen Nutzer zoomen und Datenpunkte hervorheben können.

1.0.6 6. Arten von Plots in Matplotlib

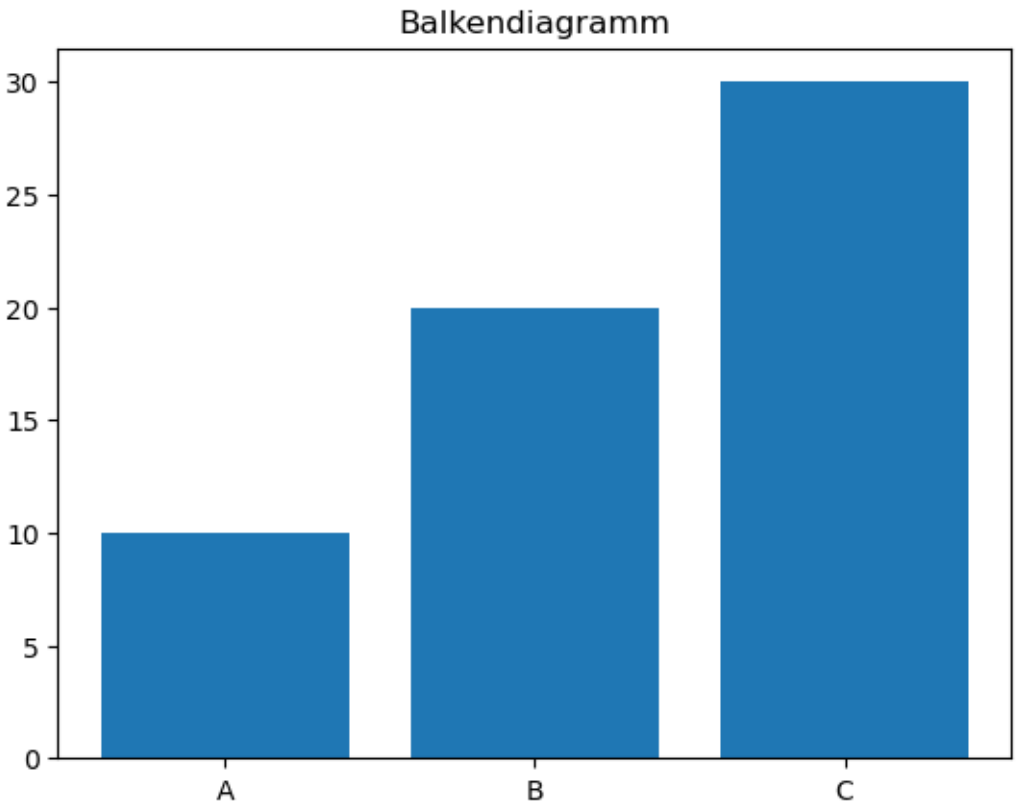
Plot-Typ	Beschreibung	Funktion
Linienplot	für zeitliche Trends und kontinuierliche Daten.	`plot()`
Streudiagramm	für die Visualisierung von Datenpunkten und deren Verteilung.	`scatter()`
Balkendiagramm	für den Vergleich von kategorischen Daten.	`bar()`
Histogramm	für die Darstellung von Häufigkeitsverteilungen.	`hist()`
Kuchendiagramm	für prozentuale Anteile an einem Ganzen.	`pie()`
Boxplot	für die Darstellung von Verteilungen und Ausreißern.	`boxplot()`
Flächendiagramm	für die Darstellung kumulativer Werte.	`fill_between()`
Dichteplot	für die Visualisierung der Wahrscheinlichkeitsdichte.	`hist()` mit der Option `density=True`
Heatmap	für die Visualisierung von Matrizen und Korrelationen.	`imshow()`
3D-Plot	für dreidimensionale Visualisierungen.	`plot_surface()` aus dem Modul `mpl_toolkits.mplot3d`

1.0.7 6.1 Beispiele zu Plots

hier werden einpaar Beispiele zu manschen der Plots gezeigt.

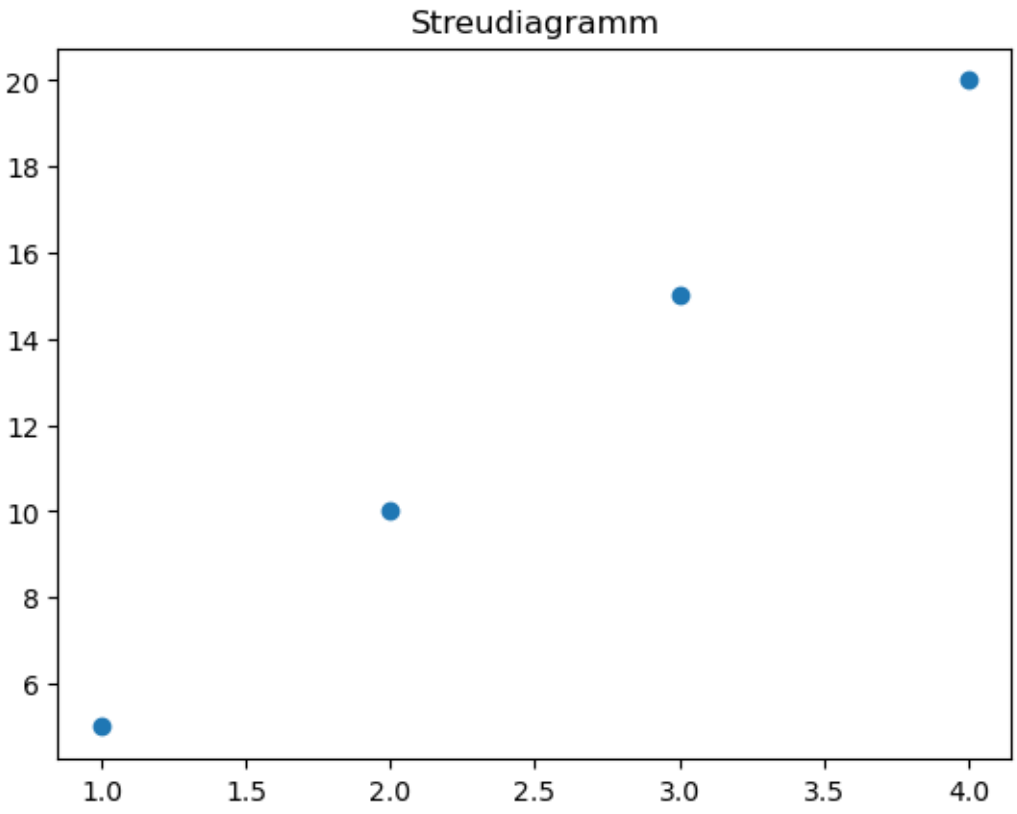
6.1.1 Beispiel zum Balkendiagramm

[82]:
```
plt.bar(['A', 'B', 'C'], [10, 20, 30])
plt.title("Balkendiagramm")
plt.show()
```



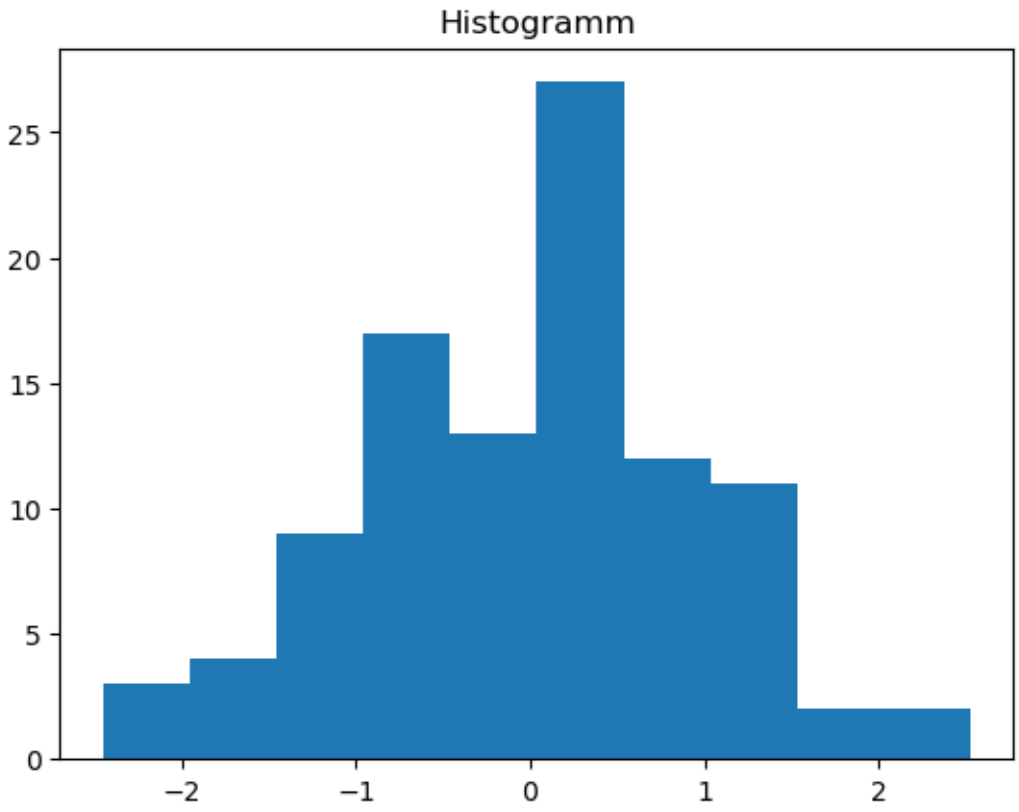
6.1.2 Beispiel zum Streudiagramm

[84]:
```
plt.scatter([1, 2, 3, 4], [5, 10, 15, 20])
plt.title("Streudiagramm")
plt.show()
```



6.1.3 Beispiel zum Histodiagramm

[80]:
```
import numpy as np
data = np.random.randn(100)
plt.hist(data, bins=10)
plt.title("Histogramm")
plt.show()
```



Aufgabe

versuchen Sie, Daten in 3 Diagrammarten Ihrer Wahl darzustellen !

1.0.8 7. Matplotlib im Vergleich zu anderen Bibliotheken

Seaborn: Baut auf Matplotlib auf, bietet jedoch eine einfachere API und speziellere Diagramme für statistische Daten. Nutzen Seaborn, wenn Sie besonders ästhetische und komplexe statistische Plots erstellen möchten.
Plotly: Interaktive Diagramme und Dashboards. Es ist ideal, wenn Sie Diagramme benötigen, die durch Benutzerinteraktionen dynamisch werden (z.B. in Webanwendungen).

1.0.9 8. häufige Probleme und deren Lösungen

Beim Erstellen von Diagrammen können Sie möglicherweise folgende Fehler sehen:

Problem	Lösung
Kein Plot wird angezeigt	Fügen Sie `%matplotlib inline` in Jupyter Notebooks hinzu, um Diagramme korrekt anzuzeigen.
Diagramme sind zu klein oder unleserlich	Verwenden Sie `figsize` in `plt.figure()`, um die Größe der Abbildung anzupassen.

[ ]:
```
### Beispiel, um Diagramme groß und leserlich zu machen
plt.figure(figsize=(10, 6))
```

Übung

versuchen Sie, den Befehl auf erstem Diagramm des Notebooks zu implementieren !

1.0.10 9. Zusammenfassung

Matplotlib ist eine äußerst flexible Bibliothek für die Erstellung statischer, interaktiver und 3D-Diagramme in Python. Mit einer leicht verständlichen Syntax und vielen Anpassungsoptionen ist sie ideal für Datenanalyse, Berichte und wissenschaftliche Visualisierungen. Durch die Kombination mit anderen Bibliotheken wie NumPy oder Pandas wird die Visualisierung großer Datenmengen zum Kinderspiel.

Einführung in Freenove Kit for Raspberry Pi Projects

Wichtige Links zu dieser Einführung:

- GitHub-Repository
- Offizielle Webseite
- PDF-Tutorial
- Video-Tutorial

1. Einführung

Freenove

Freenove bietet weltweit Open-Source-Elektronikprodukte und -dienstleistungen an. Freenove engagiert sich dafür, Kunden bei der Ausbildung in den Bereichen Robotik, Programmierung und elektronische Schaltungen zu unterstützen, damit sie ihre kreativen Ideen in Prototypen und neue, innovative Produkte umsetzen können.

Zweck von Freenove

- Bildungs- und unterhaltsame Projektkits für Roboter, Smart Cars und Drohnen.
- Bildungskits zum Erlernen von Robotersoftware-Systemen für Arduino und Raspberry Pi.
- Elektronik-Bauteilsortimente, elektronische Module und spezialisierte Werkzeuge.
- Produktentwicklungs- und Anpassungsdienstleistungen.

Freenove Kit for Raspberry Pi Projects

Das Freenove Projects Kit for Raspberry Pi ist ein umfassendes Elektronik-Bausatz, das entwickelt wurde, um Anwendern den Einstieg in die Programmierung und Elektronik mit dem Raspberry Pi zu erleichtern. Das Kit enthält eine Vielzahl von Komponenten, die es ermöglichen, verschiedene Projekte zu realisieren und gleichzeitig die Grundlagen der Elektronik und Programmierung zu erlernen.

Wichtige Eigenschaften des Freenove Projects Kit

Breites Spektrum an Komponenten: Das Kit enthält LEDs, Widerstände, Sensoren, Motoren, Relais, ein Breadboard, Jumper-Kabel und viele weitere Bauteile. Es ist ideal für Projekte, die von einfachen Schaltungen bis hin zu komplexeren Anwendungen wie Sensormessungen und Motorsteuerungen reichen.

Vielfältige Projekte: Das Kit wird mit detaillierten Anleitungen geliefert, die Schritt-für-Schritt-Erklärungen zu verschiedenen Projekten enthalten. Zu den typischen Projekten gehören:

- Blinkende LEDs und Lauflichter
- Temperatur- und Feuchtigkeitsmessungen

- Steuerung von Servomotoren und DC-Motoren
- Messung von Entfernungen mit einem Ultraschallsensor
- Anzeige von Text und Daten auf einem LCD-Display

Programmiersprachen: Die Projekte können in Python und C programmiert werden, was das Kit sowohl für Anfänger als auch für fortgeschrittene Nutzer attraktiv macht.

Kompatibilität: Das Kit ist mit verschiedenen Raspberry Pi Modellen kompatibel, einschließlich Raspberry Pi 4, 3B+, 3 und Zero.

Lernmaterialien und Support

Das Freenove Projects Kit enthält nicht nur die Hardware, sondern auch umfassende Lernressourcen wie Tutorials, Schaltpläne und Beispielcodes, die online zugänglich sind. Diese Anleitungen helfen beim Einstieg und bieten praktische Übungen, die den Lernprozess unterstützen:

- GitHub-Repository
- Offizielle Webseite
- PDF-Tutorial
- Video-Tutorial

Projects Board for Raspberry Pi

das Freenove Projects Board für Raspberry ist das speziell für die Durchführung von Elektronik- und Programmierprojekten entwickelt. Dieses Board ist eine zentrale Komponente des Freenove Project Kits und bietet viele integrierte Bauteile und Module, die den Aufbau von Projekten erleichtern.

Wichtige Komponenten und deren Funktionen:

Dieses Board enthält verschiedene Komponenten wie GPIO-Pins zur Steuerung von Sensoren, LEDs und Displays zur Anzeige von Daten, sowie Module wie Relais, Buzzer und Bewegungssensoren zur Interaktion mit der Umgebung. Es gibt Eingabemöglichkeiten wie Joysticks, Schalter und Berührungssensoren sowie Ausgabegeräte wie eine LED-Matrix, ein 7-Segment-Display und ein LED-Bar-Graph. Zusätzlich sind Module wie RFID-Leser, ADCs und Potentiometer für erweiterte Steuerungen und Messungen vorhanden.

Board mit allen Komponenten

Board

Board

2. Aufbau

der Aufbau soll in vier Schritten implementiert werden.

- Brass Standoffs installieren.
- Den Raspberry Pi montieren.

- Das Acrylteil anbringen.
- Fertiges Produkt.

Brass Standoffs installieren

Brass

Nachdem Installation soll das Board wie folgt aussehen:

BFinish

Den Raspberry Pi montieren

RPI

Das Acrylteil anbringen

Acrylic

Fertiges Produkt

FProdukt

3. Anleitung zum Herunterladen des Projektcodes

Bevor Sie diese Anleitung verfolgen, sollen Sie Raspberry Pi OS installieren und durch VNC-Viewer mit Raspberry Pi verbunden sein. Nach dem Sie es gemacht haben sollen Sie python3 mit diesem Befehl `sudo apt install python3` auf den Raspberry Pi installieren.

Offizielle Quellen

Sie können den Code von der offiziellen Freenove-Webseite oder von ihrem GitHub-Repository herunterladen. Der Code ist sowohl in C als auch in Python verfügbar, sodass Sie die Programmiersprache wählen können, die Ihnen besser liegt.

Schritte zum Herunterladen des Codes

- Öffnen Sie das Terminal auf ihrem Raspberry Pi.
- Geben Sie den folgenden Befehl `cd ~` ein, um in das Home-Verzeichnis zu wechseln.
- Geben Sie anschließend diesen Befehl `git clone https://github.com/Freenove/Freenove_Projects_Kit_for_Raspberry_Pi.git` ein, um das Repository von GitHub herunterzulade.
- da der Name sehr lang ist, können Sie mit diesem Befehl `mv Freenove_Projects_Kit_for_Raspberry_Pi/ Freenove_Kit/` auf *Freenove_Kit* ändern.:

Ausführung der Projekte

Nachdem Sie das Code heruntergelden haben, können Sie wie folgt unterschiedliche Projekte (hier zum Beispiel LED-Projekt) ausführen:

- Schalten Sie den Netzschalter. Der Netzschalter sollte bei allen Projekten eingeschaltet sein.
- Schalten Sie den Schalter Nr. 5 (Toggle-Schalter) ein.
- Gehen Sie mit dem Befehl `cd ~/Freenove_Kit/Code/Python_GPIOZero_Code/1_Blink` in den entsprechenden Ordner.
- Führen das Python-Skript mit dem Befehl `python3 Blink.py` aus.
- Um Raspberry Pi mit dem Jupyter Notebook steuern zu können, soll das Code entsprechend geändert werden.

4. Fazit

Das Freenove Projects Kit for Raspberry Pi ist eine ausgezeichnete Wahl für alle, die in die Welt der Elektronik und Programmierung eintauchen möchten. Es bietet eine strukturierte und praxisorientierte Möglichkeit, mit dem Raspberry Pi zu experimentieren und dabei wertvolle Fähigkeiten zu erlernen.

installationen

November 13, 2024

1 Installationen

Dieses Notebook führt Sie durch die Installation von Python, Jupyter Notebook und wichtigen Bibliotheken.

1.1 Schritt 1: Installation von Python

1.1.1 Auf Windows

1. Besuchen Sie die offizielle Python-Website.
2. Laden Sie die neueste Python-Version herunter und führen Sie das Installationsprogramm aus.
3. Stellen Sie sicher, dass die Option "Add Python to PATH" während der Installation ausgewählt ist.

1.1.2 Auf Rasp Pi OS

Führen Sie die folgenden Befehle, um Pythos auf OS zu installieren.

[ ]:
```
!sudo apt update
!sudo apt install python3
```

Nach der Installation können Sie die Python-Version in der Eingabeaufforderung überprüfen:

[ ]:
```
!python --version
```

1.2 Schritt 2: Installation von Pip

Pip ist ein Paketverwaltungsprogramm, das verwendet wird, um Python-Pakete und -Bibliotheken zu installieren, zu verwalten und zu deinstallieren. Es ist der Standardpaketmanager für Python und macht es einfach, externe Bibliotheken und Tools zu integrieren, die nicht in der Standardbibliothek von Python enthalten sind.

Installation von Pip auf OS:

[ ]:
```
!sudo apt install python3-pip
```

Pip wird normalerweise mit Python installiert. Überprüfe die Pip-Version:

[ ]:
```
!pip3 --version
```

Falls Pip nicht installiert ist, können Sie auch mit dem folgenden Skript installieren:

[ ]:
```
!python -m ensurepip --upgrade
```

[ ]:
```
#Alternativität zu obiger Code-Zeile
!python.exe -m pip install --upgrade pip
```

1.3 Schritt 3: Installation von Jupyter Notebook

Installieren Sie Jupyter Notebook mit Pip:

[ ]:
```
!pip3 install jupyter
```

1.4 Schritt 4: Installtion von zusätzlichen Bibliotheken

1.4.1 Darstellung in Matplotlib

Matplotlib ist eine weit verbreitete Bibliothek in Python, die für die Erstellung von statischen, animierten und interaktiven Grafiken verwendet wird. Sie bietet eine Vielzahl von Funktionen zum Plotten von Diagrammen, wie Linien-, Balken-, Kreis-, Streu- und Histogrammdiagrammen. Matplotlib ist besonders beliebt, weil sie flexibel und anpassbar ist, was sie zu einem mächtigen Werkzeug für die Visualisierung von Daten in Wissenschaft, Technik und Datenanalyse macht.

Installation von Matplotlib:

[ ]:
```
!pip3 install -U matplotlib
```

1.4.2 RPi.GPIO

RPi.GPIO

Die RPi.GPIO-Bibliothek ermöglicht die direkte Steuerung der GPIO-Pins (General Purpose Input/Output) des Raspberry Pi aus Python heraus. Sie bietet Funktionen, um Pins als Ein- oder Ausgänge zu konfigurieren und digitale Signale zu lesen oder zu senden.

Die `RPi.GPIO`-Bibliothek wird normalerweise auf einem Raspberry Pi verwendet und ist unter Windows nicht direkt verfügbar, da sie speziell für die Raspberry Pi-Hardware entwickelt wurde. Wenn Sie GPIO-Funktionen auf einem Raspberry Pi simulieren möchten, können Sie andere Bibliotheken wie `gpiozero` verwenden.

Hier ist, wie Sie sie installieren können:

[ ]:
```
!pip3 install RPi.GPIO
```

1.4.3 GPIOZero

GPIOZero ist eine einfachere und benutzerfreundlichere Bibliothek zur Steuerung der GPIO-Pins des Raspberry Pi. Sie baut auf RPi.GPIO auf und bietet eine höhere Abstraktionsebene, was die Programmierung von Hardwarekomponenten für Anfänger erleichtert.
Hier ist, wie Sie sie installieren können:

[ ]:
```
!pip3 install gpiozero
```

1.5 Schritt 5: Verifizierung

Um sicherzustellen, dass alles korrekt installiert ist, können Sie die folgenden Befehle in einer neuen Jupyter-Zelle ausführen:

[ ]:
```
!pip show gpiozero
```

1.6 Zusammenfassung

Damit ist die grundlegende Installation auf deinem Windows-System abgeschlossen. Sie können nun Jupyter Notebook verwenden, um Python-Skripte zu entwickeln.