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.