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.

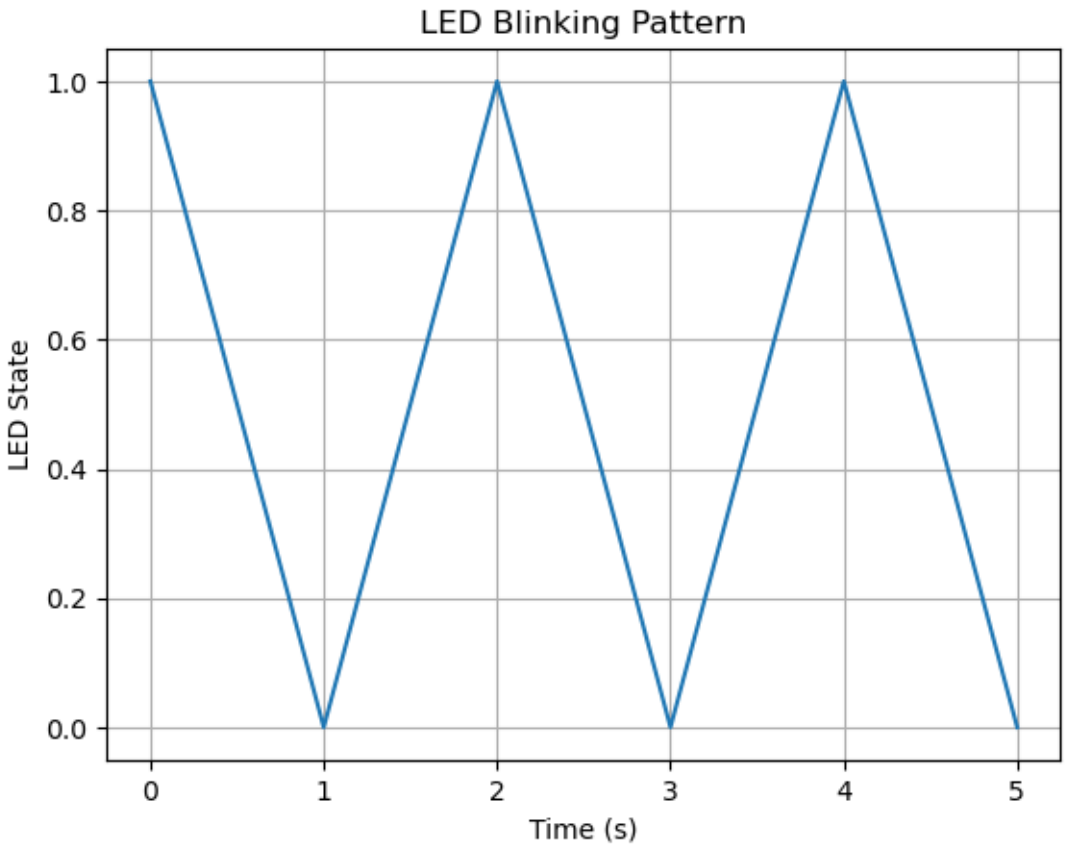
Die Anweisungen zur Installation von Matplotlib finden Sie unter Installationen.

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