Sensoren auslesen und Aktoren ansteuern mit dem Raspberry Pi®

Inhalt

1 Ziele

In diesem Modul lernen Sie den Umgang mit dem Raspberry PI® und die GPIO[1] Programmierung in Java kennen, um Signale einzulesen und auszugeben. Die Programmierung der GPIO Pins mit Java geschieht mit Hilfe der Pi4J-Bibliothek[2]. Sie schreiben Programme, die einerseits Output Pins steuern, um beispielsweise LEDs zum Leuchten zu bringen und andererseits Input Pins verwenden, sodass Taster-Signale Aktionen auslösen. Sie lernen ferner das Auslesen von Sensorwerten über verschiedene Bussysteme kennen. In jeder Aufgabe wird die Funktionsweise eines Steckbretts (engl. breadboard) für die einzelnen Anwendungen erklärt. Am Ende des Moduls wissen Sie, wie die GPIO Pins des Raspberry Pi mit Java-Programmen ansteuert werden.

[1] GPIO steht für General Purpose Input/Output
[2] https://pi4j.com/1.2/index.html

2 Aufgabe 1: Blinkendes Licht

In dieser Aufgabe bringen Sie eine LED zum Blinken:

1. Für die LED-Steuerung bauen Sie einen elektronischen Schaltkreis zusammen.
2. Mit Hilfe der Projektvorlage *BlinkingLED* implementieren Sie die LED-Ansteuerung.

2.1 Aufbau

Bauen Sie den Schaltkreis mit folgenden Schritten zusammen:

1. LED und Widerstand (220 Ω) auf dem Steckbrett anbringen.
2. Realisieren Sie Steckverbindungen:
    a. Kabel 1 (grün): *LED Pin* – mit *PinNo 6 (Ground)*
    b. Kabel 2 (rot): *LED Pin* + mit *12 (GPIO 1)*

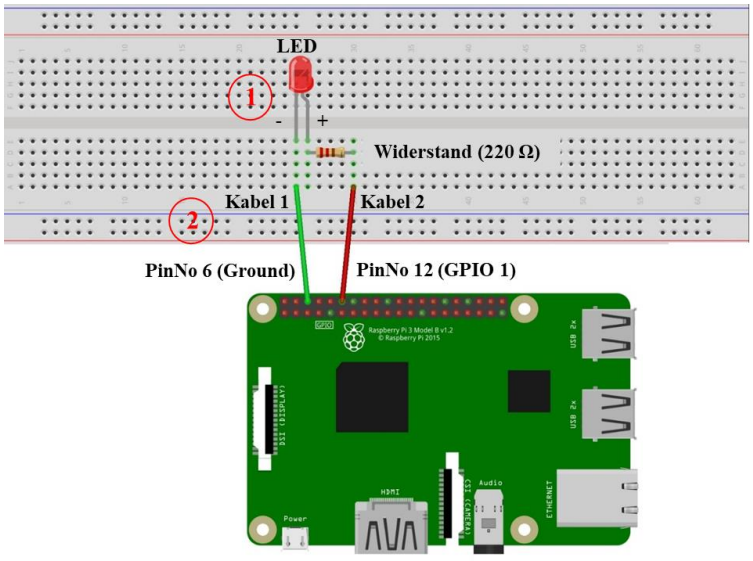

Abbildung 1: LED-Anbindung mit Widerstand und Kabel auf dem Steckbrett

Hinweis: Beim Zusammenbau ist zu beachten, dass die Leuchtdiode mit richtiger Polarität eingesetzt wird. Die LED hat zwei unterschiedlich lange Enden (Anode (+) und

Kathode (-)), wobei das längere Ende, die Anode, mit Pin 12 (GPIO 1) verbunden wird. Der Widerstand dient dazu, die Diode vor zu grossen Strömen zu schützen.

2.2 Implementierung und Test des Steuerprogramms

1. Öffnen Sie mit *BlueJ* aus dem Projektordner */home/pi/documents/gpio* die Projektvorlage *BlinkingLED*.
2. Kompilieren Sie die Klasse `BlinkingLED`.
3. Instanzieren Sie `BlinkingLED` und steuern Sie die LED mit den Methoden `on` und `off`. Zum Schluss schalten Sie den GPIO Controller mit `cleanup` aus.
4. Vervollständigen Sie die Methode `blink` mit folgender Signatur:

   ```
   public void blink(int n) throws Exception
   ```

   Der Parameter *n* gibt vor wie oft die LED blinken soll. Für die Blinkgeschwindigkeit ist eine Wartezeit von 1000 Millisekunden (`Thread.sleep`) vorgesehen. Verwenden Sie die Methoden `on` und `off`.

   Testen Sie ihre Implementierung mit der Methode `main`.

2.3 Lessons Learned

In dieser Aufgabe lernten Sie mit einem GPIO Pin eine LED zu steuern. Sie haben konkret ein Output-Signal generiert, um die Stromversorgung der LED zu kontrollieren.

3 Aufgabe 2: Temperaturmessung mit Eindraht-Bus

In dieser Aufgabe messen Sie die Temperatur mit einem Sensor, der am *Eindraht-Bus* (One-Wire oder 1-Wire) mit dem Raspberry Pi verbunden ist:

1. Für die Messung bauen Sie einen elektronischen Schaltkreis zusammen.
2. Aktivieren Sie die 1-Wire Schnittstelle des Raspberry Pi.
3. Mit Hilfe der Projektvorlage *TemperatureOneWire* lernen Sie den Temperatursensor auszulesen.

3.1 Aufbau

Bauen Sie den Schaltkreis mit folgenden Schritten zusammen:

1. Temperatursensor Typ *KY-001* auf dem Steckbrett platzieren.
2. Widerstand (4,7k Ω) mit dem Signal-Pin (S) des Temperatursensors verbinden.
3. Realisieren Sie Steckverbindungen:
    a. Kabel 1 (rot): *Temperatursensor Pin +V* mit *PinNo 1 (3.3 VDC Power)*
    b. Kabel 2 (schwarz): *Temperatursensor Pin GND* mit *PinNo 6 (Ground)*
    c. Kabel 3 (grün): *Temperatursensor Pin Signal* mit *PinNo 7 (GPIO 7)*

Hinweis: Nach dem Aufbau und korrekten Anschluss des Temperatursensors leuchtet eine Sensor-LED in regelmässigen Abständen kurz rot auf.

Kommentiert [RP1]: In regelmässigen Abständen oder in regelmässigem Abstand

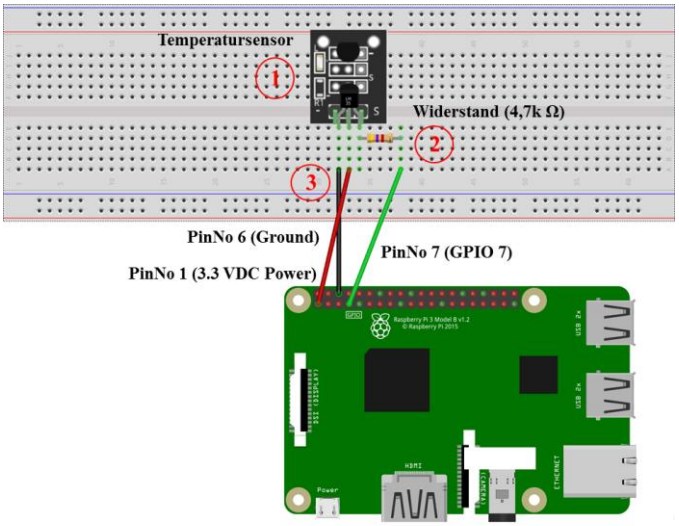

Abbildung 2: Anbindung des Temperatursensors mit Eindraht-Bus

3.2 Eindraht-Bus aktivieren

Um den 1-Wire Bus benützen zu können, ist in der Raspberry Pi Konfigurationsdatei ein Eintrag vorzunehmen:

1. Öffnen Sie dazu das Terminalfenster und geben Sie den folgenden Befehl ein: `sudo nano /boot/config.txt`.
2. Ergänzen Sie die geöffnete Konfigurationsdatei am Ende um die folgende Zeile: `dtoverlay=w1-gpio,gpiopin=4`.
3. Speichern Sie die Datei mit der Tastenkombination *STRG+O* und verlassen Sie den Editor mit der Kombination *STRG+X*.
4. Starten Sie den Raspberry Pi mit `sudo reboot` neu.

3.3 Implementierung der Software und Test des Aufbaus

Für das Auslesen der Temperatur wird der 1-Wire Bus verwendet. Dieser Bus ist seriell und kommt, wie der Name andeutet, mit einer Leitung aus, die sowohl für die Stromversorgung als auch für das Senden und den Empfang von Daten genutzt wird.

1. Öffnen Sie mit *BlueJ* aus dem Projektordner */home/pi/documents/gpio* die Projektvorlage *TemperatureOneWire*.
2. Kompilieren Sie die Klasse `TemperatureOneWire`.
3. Instanzieren Sie `TemperatureOneWire` und zeigen Sie die aktuelle Temperatur mit `displayTemperature` an.
4. Vervollständigen Sie nun die Methode `polling` mit folgender Signatur:

   ```
   public void polling(long duration)
   ```

   Verwenden Sie die lokale Variable `sensor` und rufen Sie in regelmässigen Abständen von 500 Millisekunden (`Thread.sleep`) die Methode `getTemperature` auf und vergleichen Sie den Rückgabewert mit `previousTemperature`. Bei einer Temperaturänderung geben Sie die Uhrzeit (`getTime`), die aktuelle Temperatur und die Differenz zum letztgespeicherten Temperaturwert auf der Konsole aus. Dazu Beispiel: *Time: 13:49:20, Current Temperature: 25.8ºC, Change: 0.1ºC*. Aktualisieren Sie `previousTemperature`.

   Testen Sie nun die Erweiterung mit der Methode `main`.

Kommentiert [RP2]: Abständen

3.4 Lessons Learned

In dieser Aufgabe erlernten Sie das Auslesen eines Temperatursensors über einen einfachen Bus.

4 Aufgabe 3: Sequentieller Web-Server

In dieser Aufgabe lernen Sie mit einem sequentiellen Web-Server zu kommunizieren und rufen mit einen HTTP-Request einen Text ab.

4.1 Implementierung und Test

3. Öffnen Sie mit *BlueJ* aus dem Projektordner */home/pi/documents/web* die Projektvorlage *SeqWebServer*.
4. Kompilieren Sie die Klasse `WebServer`.
5. Studieren Sie im Code der Klasse `WebServer` den Konstruktor und die Methoden `run`, `processRequest`, `processRequestHeader`, `generateResponseHeader` und `generatePage`.
6. Starten Sie nun in *BlueJ* den Web-Server über die Methode `main`.

   Hinweis: Der gestartete Web-Server kann in BlueJ mit der Tastenkombination *Ctrl+Shift+R* (Alternativ: Menuauswahl *Tools/Reset Java Virtual Machine*) wieder beendet werden.
7. Testen Sie den Web-Server mit dem Webbrowser indem Sie folgenden Aufruf machen: `http://localhost:6789`. Der Webbrowser sendet einen Request an den Web-Server, welcher als Antwort (engl. Response) den Inhalt der Webseite zurücksendet. Was zeigt der Browser nach dem Aufruf an?

   Hinweis: Rufen Sie im Webbrowser mit der Tastenkombination *Ctrl+Shift+I* die Entwicklertools auf, öffnen die Rubrik *Network* und laden dann mit *Ctrl+R* die Seite neu. Danach wählen die den Namen *localhost* aus und öffnen die Rubrik *Headers/Response Headers*. Welche Länge wird unter *Content-Length* angegeben?
8. Zeigen Sie einen eigenen Text an, indem Sie in der Methode `generatePage` eine eigene Web-Seite zusammenstellen. Testen Sie Anzeige mit dem Webbrowser[3].

[3] Auf https://www.w3schools.com/html/default.asp finden Sie ein HTML-Tutorial, falls Sie noch kein HTML beherrschen.

4.2 Lessons Learned

In dieser Aufgabe lernten Sie einen einfachen Web-Server zu benützen und HTTP-Requests mit einem Webbrowser auszuführen.

Hinweis: Befinden sich weitere Geräte (z.B. Windows-PC oder Smartphone) im selben Netzwerk wie das Raspi, dann können Sie auch mit einem Webbrowser auf diesen Geräten den Web-Server auf dem Raspi erreichen. Im Aufruf des Webbrowsers ist *localhost* durch die IP-Netzwerkadresse (`http://[IP-Adresse]:6789`) des Raspi zu ersetzen. Diese kann im ▀ Terminalfenster durch Eingabe von *ifconfig* ermittelt werden.

5 Aufgabe 4: Sequentieller Web-Server mit Temperaturanzeige

In dieser Aufgabe rufen Sie über einen HTTP-Request die aktuelle Temperatur ab. Wählen Sie dabei den Aufbau von der Temperaturmessung mit Eindraht-Bus (siehe Aufgabe 2).

5.1 Implementierung und Test

1. Zur Messung der Temperatur benützen Sie den Eindraht-Bus: Verwenden Sie den Aufbau Aufgabe 2 – Temperaturmessung mit Eindraht-Bus.
2. Öffnen Sie mit *BlueJ* aus dem Projektordner */home/pi/documents/web* die Projektvorlage *TempWebServer*.
3. Kompilieren Sie die Klassen `WebServer` und `TemperatureOneWire`.
4. Testen Sie in *BlueJ* die Temperaturklasse durch die Instanzierung von `TemperatureOneWire` und zeigen Sie mit `displayTemperature` die aktuelle Temperatur an.
5. Vervollständigen Sie nun die Methode `generatePage`, indem Sie die aktuelle Temperatur abfragen und den Wert in den vorhandenen HTML Code integrieren. Verwenden Sie die Instanz-Variable `sensor`. Ergänzen Sie den Rückgabewert mit dem Zeitstempel und mit der Temperatur von Instanz-Variable `sensor`.
6. Starten Sie in *BlueJ* den Web-Server mit der Methode `main`.
7. Testen Sie mit dem Webbrowser den Web-Server für die Temperaturmessung, indem Sie folgenden Aufruf machen: `http://localhost:6789`.

5.2 Lessons Learned

In dieser Aufgabe lernten Sie einen Web-Server mit Temperaturmessung zu implementieren. Nach dem Start des Web-Servers lassen sich mit einem Webbrowser die HTTP-Requests ausführen.

6 Aufgabe 5: Sequentieller Web-Server mit LED-Steuerung

In dieser Aufgabe steuern Sie über einen HTTP-Request mit Query-String eine LED.

6.1 Exkurs: Query-String

Ein Query-String (Abfrage-Zeichenkette)[4] ist Bestandteil eines URL und wird vom Web-Server ausgewertet. Er beginnt mit einem «`?`», dann folgt ein Schlüssel-Wert-Paar mit *Parametername* und *-wert*, welches sich durch «`=`» trennt. Nach einem &-Zeichen können weitere Schlüssel-Werte-Paare folgen.

Beispielsweise enthält der URL

```
http://localhost:6789/?p=1
```

den Query-String `p=1`, wobei dem Parameter `p` der Wert `1` zugeordnet ist.

6.2 Implementierung und Test

Benützen Sie für die LED-Steuerung den Aufbau von Aufgabe 1.

1. Öffnen Sie mit *BlueJ* aus dem Projektordner */home/pi/documents/web* die Projektvorlage *LEDWebServer*.
2. Kompilieren Sie die Klassen `WebServer` und `BlinkingLED`.
3. Testen Sie in *BlueJ* die LED-Steuerung durch die Instanzierung von `BlinkingLED` und rufen Sie die Methoden `on`, `off` und `toggle` auf.
4. Vervollständigen Sie in der Klasse `WebServer` die Methode `switchLED` mit folgender Signatur:

   ```
   public void switchLED(String ledState) throws Exception
   ```

   Der Parameter `ledState` gibt den gewünschte LED-Zustand vor: `0` für LED aus, `1` für LED an und `2` für LED umschalten. Implementieren Sie die Zustandsbestimmung mit Hilfe einer switch-Anweisung und verwenden Sie die Methoden `off`, `on` und `toggle` der Instanz-Variablen `blinkingLED`.

[4] https://de.wikipedia.org/wiki/Query-String

5. Starten Sie in *BlueJ* den Web-Server mit der Methode `main`.
6. Testen Sie mit dem Webbrowser den Web-Server mit LED-Steuerung, indem Sie folgende Aufrufe machen:

   http://localhost:6789?state=0,
   http://localhost:6789?state=1 und
   http://localhost:6789?state=2.

6.3 Lessons Learned

In dieser Aufgabe lernten Sie HTTP-Requests mit Query-Strings kennen. Nach dem Start des Web-Servers lässt sich die LED mit einem Webbrowser über das Netzwerk ein- und ausschalten.

7 Anhang

7.1 Hardware und Zubehör

7.1.1 Raspberry Pi

Der Raspberry Pi 400 enthält einen 40-Pin-Erweiterungs-Header mit der Bezeichnung J8, der den Zugriff auf 28 GPIO-Pins ermöglicht.

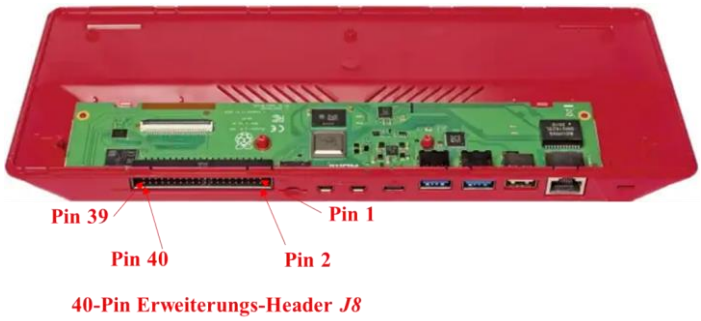

Abbildung 3: Raspberry Pi 400 mit 40-Pin Erweiterungs-Header *J8*

7.1.2 GPIO

Der GPIO (**G**eneral **P**urpose **I**nput/**O**utput) ist eine Schnittstelle, an die

- Sensoren, wie Temperaturfühler,
- Eingabegeräte, wie Taster oder Schalter und
- Ausgabegeräte, wie LED

angeschlossen werden können.

Das folgende Diagramm zeigt die GPIO Pin Belegung des *J8* Headers.

Abbildung 4: GPIO Pin Belegung des Raspberry Pi 400

7.1.3 Steckbrett

Mit Hilfe eines Steckbretts (engl. breadboard) können Schaltungen rasch ohne Löten aufgebaut werden.

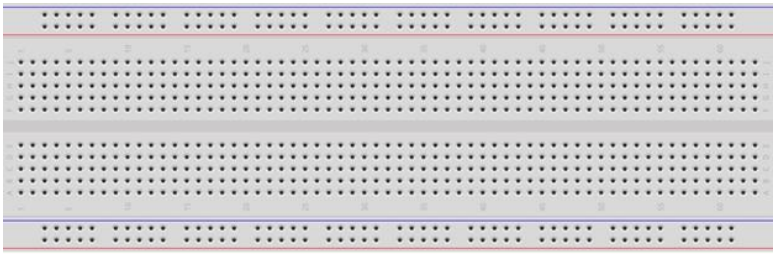

Abbildung 5: Steckbrett

In der obigen Abbildung zeigen die roten horizontalen und vertikalen Linien die Verbindungen der einzelnen Löcher auf.

7.2 Software

7.2.1 Betriebssystem

Auf dem Raspberry Pi 400 ist das Debian-basierte Betriebssystem Raspbian installiert. Das Öffnen des Terminalfensters erfolgt über die Tastenkombination **Strg + Alt + T**. Die gängigen Befehle sind in der Dokumentation für Linux Commands zu finden.

7.2.2 BlueJ

BlueJ () ist eine integrierte Entwicklungsumgebung (DIE) für Java und speziell für Ausbildungszwecke konzipiert.

7.2.3 Pi4J Bibliothek

Pi4J ist eine Java-Bibliothek, um die GPIO-Ports des Raspberry Pi in Java-Programmen zu verwenden.

