1 Allgemeines

1.1 Inhalt

1.2 Hilfreiche Internetseiten

- Vorwiderstand für LED berechnen:
  - https://www.elektronik-kompendium.de/sites/grd/1006011.htm
- GPIO-Pin-Belegung des Raspberry Pis:
  - https://www.elektronik-kompendium.de/sites/raspberry-pi/1907101.htm
- Ansteuerung einer LED:
  - https://sensorkit.joy-it.net/de/sensors/ky-011
- Dokumentation des Python-Pakets gpiozero:
  - https://gpiozero.readthedocs.io/en/stable/index.html
- Ansteuerung der Sensoren:
  - https://sensorkit.joy-it.net/de/
- Hinweise zu Python:
  - https://quickref.me/python.html

1.3 Python-Programm ausführen

Das Python-Programm kannst du auf zwei verschiedene Weisen starten:

- Wenn du die Datei, die den Programm-Code enthält, mit thonny geöffnet hast, kannst du ganz einfach auf “Run” oben im Fenster klicken. Das Programm kannst du danach über die Schaltfläche “Stop” stoppen.
- Wenn du die Datei, die den Programm-Code enthält, nicht mit thonny geöffnet hast, kannst du das Programm direct vom Terminal aus starten. Navigiere dazu mit dem Befehl “cd” zu der Datei, die den Programm-Code enthält. Darauf führst du den Befehl “python main.py” aus. “main.py” ist hierbei der Name deiner Datei.

1.4 Polung einer LED

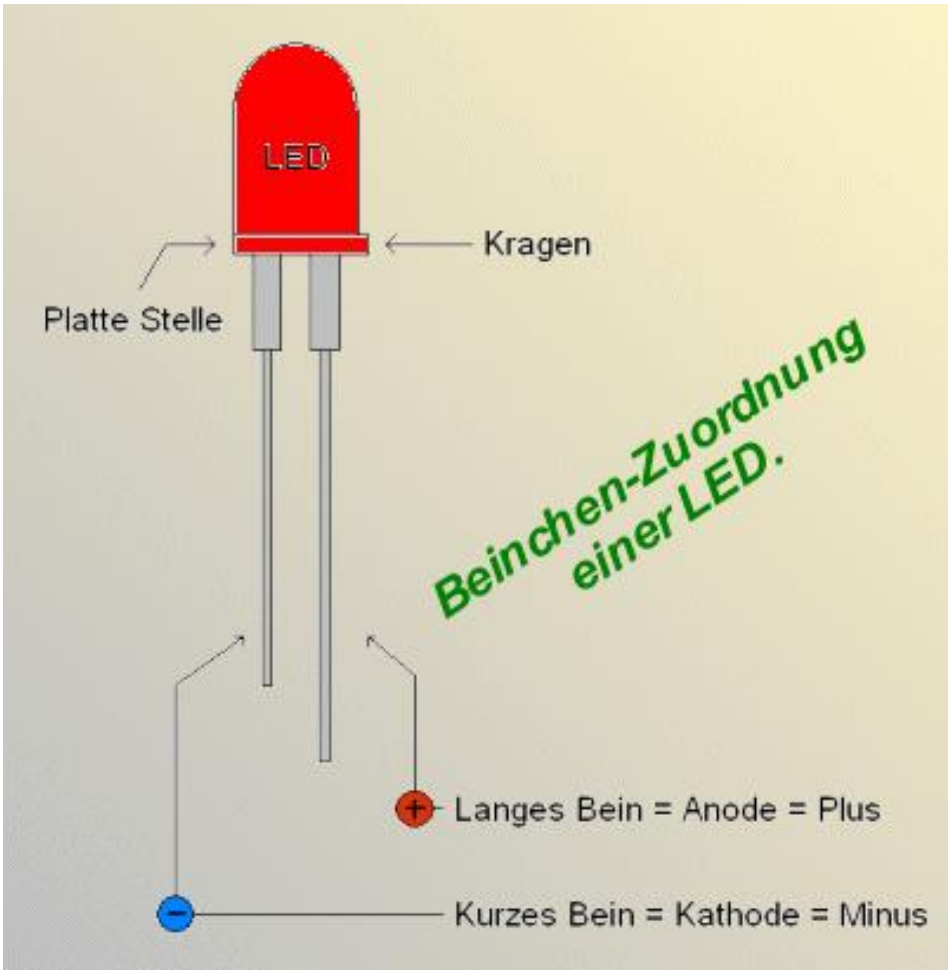

Abbildung 1: Polung einer LED

2 Aufgabe 1a: LED

In dieser Aufgabe wirst du ein Python-Programm schreiben, das eine LED zum Blinken bringt. Die LED soll in einer Dauerschleife eine Sekunde lang eingeschaltet und eine Sekunde lang ausgeschaltet sein. Verbinde dazu eine LED mit dem Raspberry Pi. Baue zudem einen Vorwiderstand für die LED ein!

Hinweise:

- Verwende „LED“ aus der Bibliothek „gpiozero“ für die Steuerung der LED:
  - https://gpiozero.readthedocs.io/en/stable/api_output.html#regular-classes:~:text=15.1.1.-,LED,-%EF%83%81

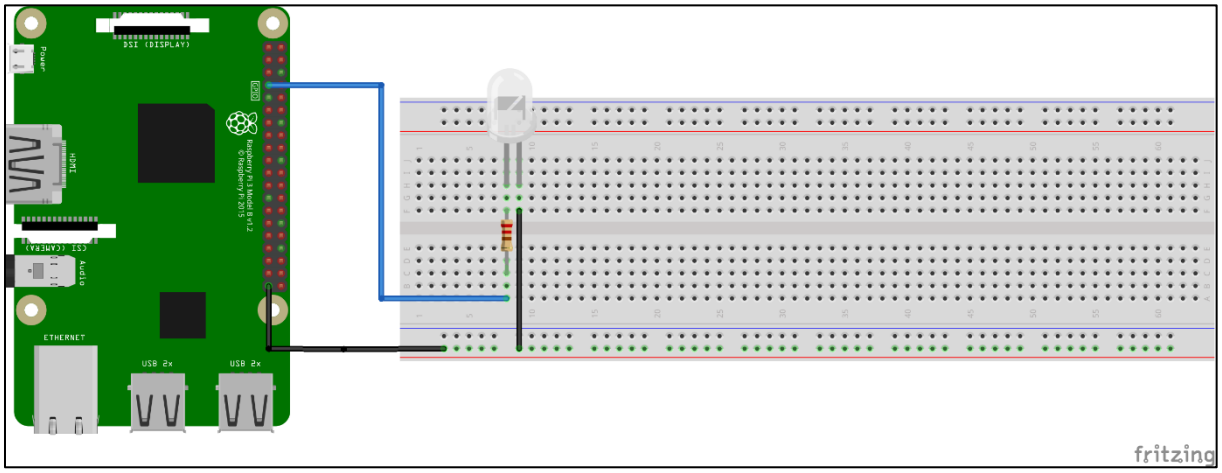

Abbildung 2: Schaltbild Aufgabe 1a

3 Aufgabe 2a: Frequenzgenerator

In dieser Aufgabe wirst du ein Python-Programm schreiben, das einen passiven Buzzer (KY006) ansteuert. Der Buzzer soll in einer Dauerschleife den Ton A3 eine Sekunde lang wiedergeben und eine Pause eine von einer Sekunde machen. Verbinde dazu den passiven Buzzer mit dem Raspberry Pi.

Hinweise:

- Verwende TonalBuzzer aus der Bibliothek gpiozero:
  - https://gpiozero.readthedocs.io/en/stable/api_output.html#gpiozero.TonalBuzzer:~:text=15.1.5.-,TonalBuzzer,-%EF%83%81
- Verwende für das Abspielen der Töne die Funktion „play()“ von TonalBuzzer und für die Definition der Tonhöhe „Tone“ aus gpiozero.
- Orientiere dich gern an der Dokumentation zum passiven Buzzer.

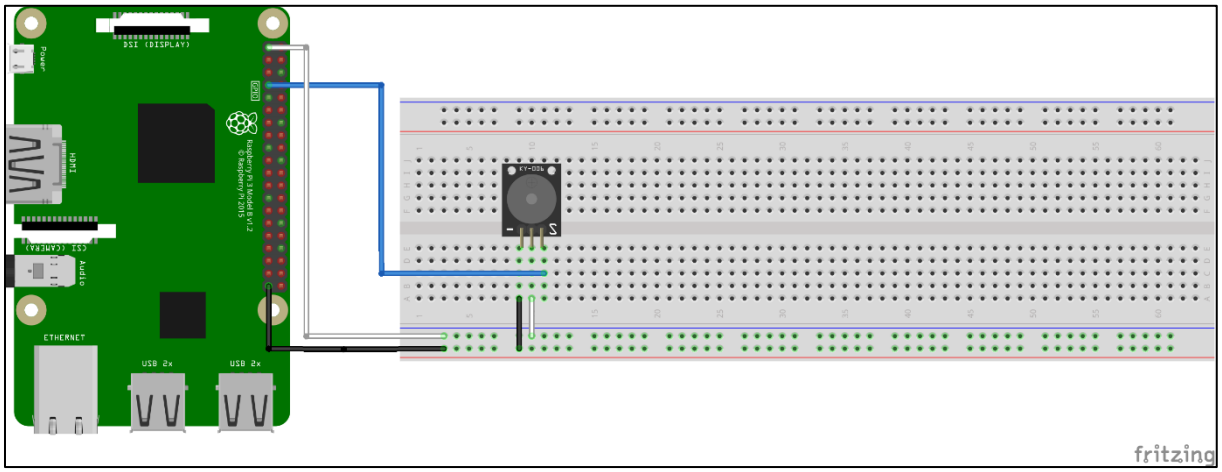

Abbildung 3: Schaltbild Aufgabe 2a

4 Aufgabe 2b: Lied

In dieser Aufgabe wirst du ein Python-Programm schreiben, das einen passiven Buzzer (KY006) ansteuert. Der Buzzer soll das Lied „Alle meine Entchen“ abspielen. Die Noten findest du in Abbildung 4.

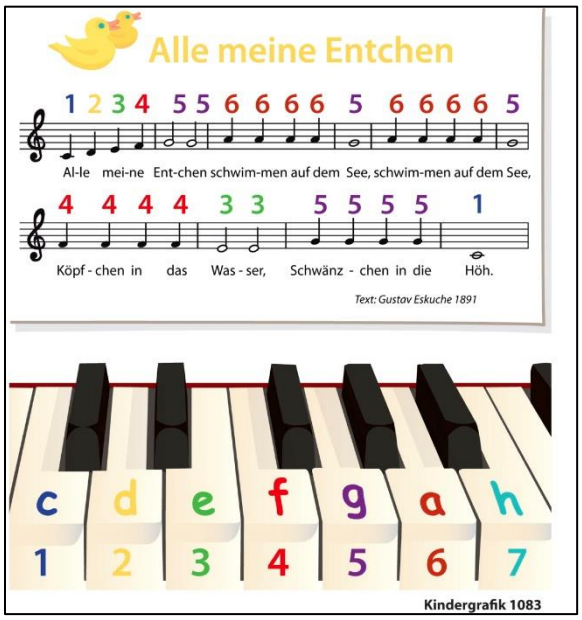

Abbildung 4: Noten für Aufgabe 2b

Hinweise:

- Verwende TonalBuzzer aus der Bibliothek gpiozero:
  - https://gpiozero.readthedocs.io/en/stable/api_output.html#gpiozero.TonalBuzzer:~:text=15.1.5.-,TonalBuzzer,-%EF%83%81
- Verwende für das Abspielen der Töne die Funktion „play()“ von TonalBuzzer und für die Definition der Tonhöhe „Tone“ aus gpiozero.
- Orientiere dich gern an der Dokumentation zum passiven Buzzer.
- Speichere die Töne und Tonlängen jeweils in einer Liste und spiele die Töne mit ihrer Länge der Reihe nach in einer For-Schleife ab. Beachte, dass der Buzzer erst ab der Tonhöhe C4 funktioniert. Verwende daher folgende Eingabe für die einzelnen Töne der Tonleiter: C4, D4, E4, F4, G4, A4, H4, C5

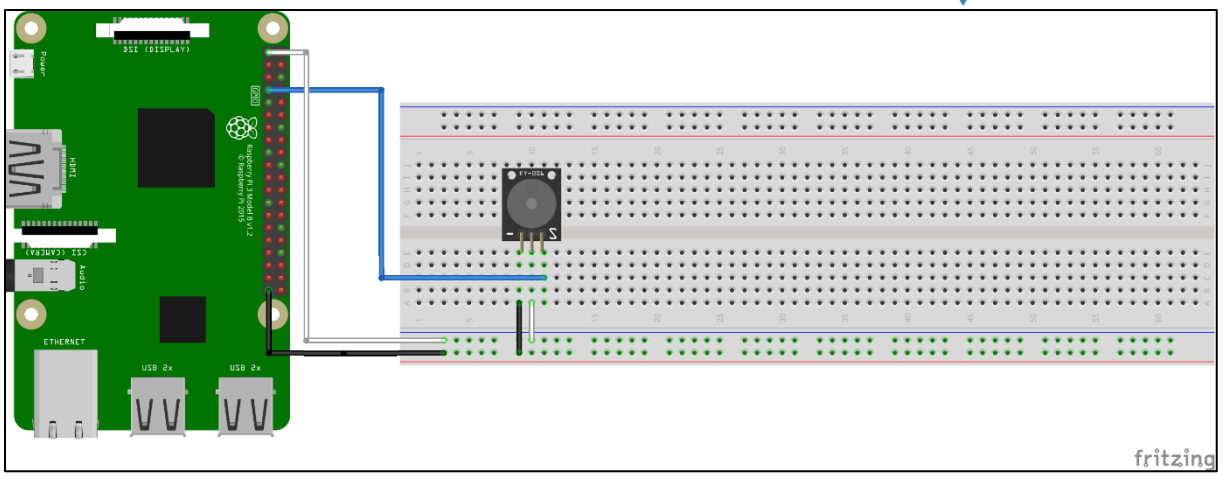

Abbildung 5: Schaltbild Aufgabe 2b

5 Aufgabe 2c: Alarm

In dieser Aufgabe wirst du ein Python-Programm schreiben, das einen passiven Buzzer (KY006) und eine Lichtschranke (KY010) ansteuert. Der Buzzer soll einen Alarmton ausgeben, wenn die Lichtschranke unterbrochen wird.

Hinweise:

- Wenn der Alarm ausgelöst wird, soll die Tonfolge „A3 – Pause – A4 – Pause“ einmal abgespielt werden.
- Verwende TonalBuzzer aus der Bibliothek gpiozero:
  - https://gpiozero.readthedocs.io/en/stable/api_output.html#gpiozero.TonalBuzzer:~:text=15.1.5.-,TonalBuzzer,-%EF%83%81
- Verwende für das Abspielen der Töne die Funktion „play()“ von TonalBuzzer und für die Definition der Tonhöhe „Tone“ aus gpiozero.
- Orientiere dich gern an der Dokumentation des passiven Buzzer und der Lichtschranke.

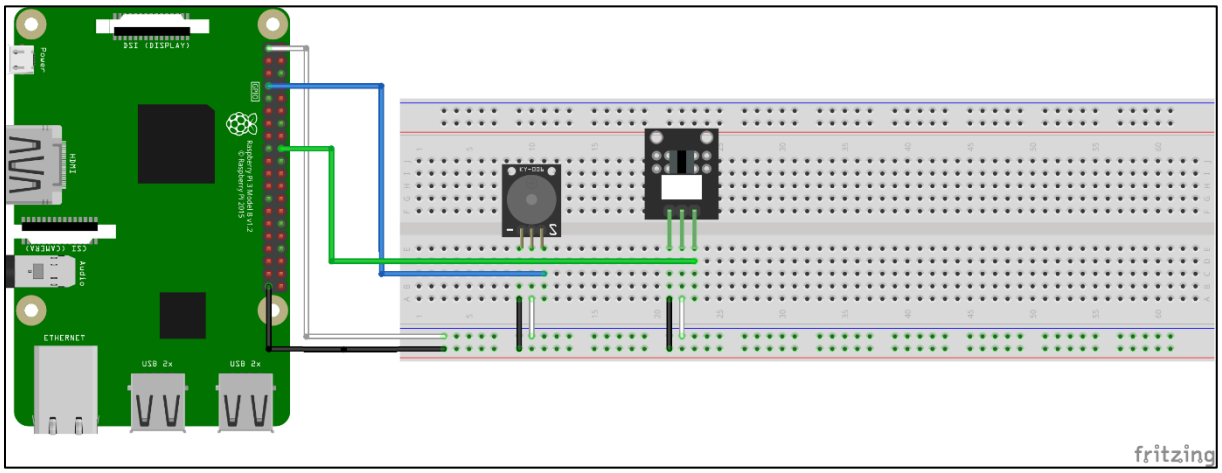

Abbildung 6: Schaltbild Aufgabe 2c

6 Aufgabe 3a: Lichtschalter

In dieser Aufgabe wirst du ein Python-Programm schreiben, das eine LED umschaltet, wenn ein Taster (KY004) gedrückt wird. Wenn die LED aus ist und der Taster gedrückt wird, so soll die LED leuchten. Wenn die LED bereits leuchtet, soll sie bei Tasterdruck ausgeschaltet werden. Verbinde dazu eine LED und einen Taster mit dem Raspberry Pi. Baue zudem einen Vorwiderstand für die LED ein!

Hinweise:

- Dieses Programm arbeitet Event-basiert. Das heißt, die Hauptprogrammschleife ist eine Dauerschleife und es wird eine bestimmte Funktion aufgerufen, wenn der Taster gerückt wird. Dazu implementierst du eine Funktion „onPressed“, die die LED umschaltet. Diese Funktion verbindest du mit dem Event „when_pressed“ des Tasters:

```
#Die Funktion onPressed wird an das Event "when_pressed" des Tasters gebunden.
button.when_pressed = onPressed
```

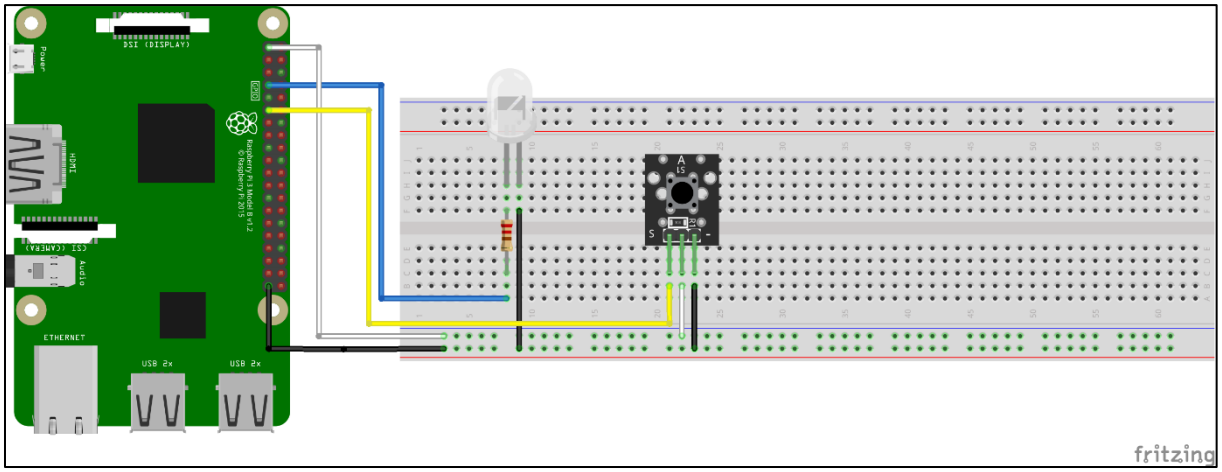

Abbildung 7: Schaltbild Aufgabe 3a

7 Aufgabe 4a: Thermometer

In dieser Aufgabe wirst du ein Python-Programm schreiben, das einen Temperatur-Sensor (KY001) ansteuert. Orientiere dich an der Dokumentation zu dem Sensor, um die Kommunikation mit diesem zu implementieren. (https://sensorkit.joy-it.net/de/sensors/ky-001)

Die Hauptprogrammschleife soll eine Ausgabe der Temperaturen ermöglichen. Jede Sekunde soll die Temperatur in Grad Celsius und Grad Fahrenheit ausgegeben werden. Zudem soll das Programm den Mittelwert der letzten fünf Temperaturmessungen ausgeben. Speichere daher die Temperaturwerte in einer Liste und implementiere eine Funktion für die Berechnung des Mittelwertes.

Damit der Raspberry Pi über den One-Wire-Bus mit dem Temperatur-Sensor kommunizieren kann, muss er zunächst aktiviert werden. Führe dazu den Befehl “sudo raspi-config” aus. Navigiere auf “Interface Options” und aktiviere die One-Wire-Schnittstelle. Danach ist ein Neustart des Raspberry Pis notwendig. Das geschieht automatisch oder manuell mit dem Befehl “sudo reboot”.

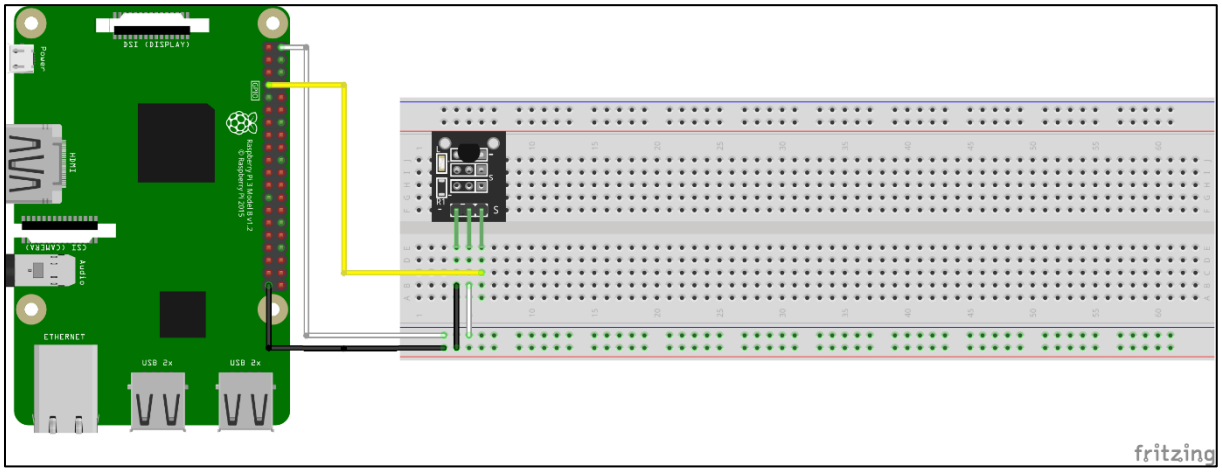

Abbildung 8: Schaltbild Aufgabe 4a

8 Aufgabe 4b: Temperatur-Licht

In dieser Aufgabe wirst du ein Python-Programm schreiben, das einen Temperatur-Sensor (KY001) und drei LEDs (rot, grün, blau) ansteuert. Orientiere dich an der Dokumentation zu dem Temperatur-Sensor, um die Kommunikation mit diesem zu implementieren. (https://sensorkit.joy-it.net/de/sensors/ky-001)

Die Temperatur soll jede Sekunde einmal ausgelesen werden. Darauf soll das Programm die drei LEDs wie folgt ansteuern:

- Wenn die Temperatur größer als 25°C ist, soll nur die rote LED leuchten.
- Wenn die Temperatur kleiner als 23°C ist, soll nur die blaue LED leuchten.
- Liegt die Temperatur zwischen beiden Werten, soll die grüne LED leuchten.

Damit der Raspberry Pi über den One-Wire-Bus mit dem Temperatur-Sensor kommunizieren kann, muss er zunächst aktiviert werden. Führe dazu den Befehl “sudo raspi-config” aus. Navigiere auf “Interface Options” und aktiviere die One-Wire-Schnittstelle. Danach ist ein Neustart des Raspberry Pis notwendig. Das geschieht automatisch oder manuell mit dem Befehl “sudo reboot”.

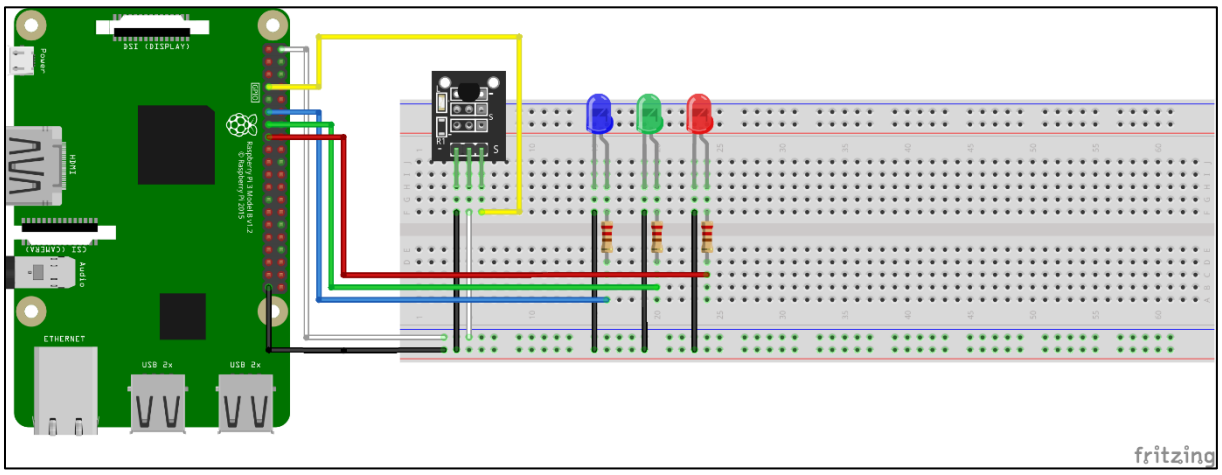

Abbildung 9: Schaltbild Aufgabe 4b

9 Aufgabe 5a: Helligkeit

In dieser Aufgabe wirst du ein Python-Programm schreiben, das eine LED mit einem Taster (KY004) steuert. Wenn der Taster gedrückt wird, soll die Helligkeit der LED bis zu ihrem maximalen Wert (1.0) erhöht werden. Wenn der Taster weiter gedrückt wird, dann soll die LED wieder schrittweise verdunkelt werden. Ist die LED dann aus, soll sie wieder schrittweise erleuchtet werden usw. Verbinde dazu eine LED und einen Taster mit dem Raspberry Pi. Baue zudem einen Vorwiderstand für die LED ein!

Hinweise:

- Verwende in dieser Anwendung die Event-basierte Programmierung. Die Hauptprogrammschleife ist dabei eine Endlosschleife. Zudem ist eine Funktion „onPressed“ notwendig, die abgearbeitet wird, wenn der Taster gedrückt wird. In der Funktion wird die LED angesteuert. Die Funktion wird wie folgt mit dem Event „Taster gedrückt“ verknüpft:

```
#Die Funktion onPressed wird an das Event "when_pressed" des Tasters gebunden.
button.when_pressed = onPressed
```

- Um die Helligkeit der LED steuern zu können solltest du im Programm statt „LED“ die Klasse „PWMLED“ verwenden:
  - https://gpiozero.readthedocs.io/en/stable/api_output.html#gpiozero.PWMLED
  - Wenn du im Programm die Variable „led“ erstellt hast, kannst du mit „led.value“ auf den Helligkeitswert zugreifen und diesen bearbeiten wie eine Variable. Der Wert darf allerdings nur zwischen 0.0 und 1.0 sein.

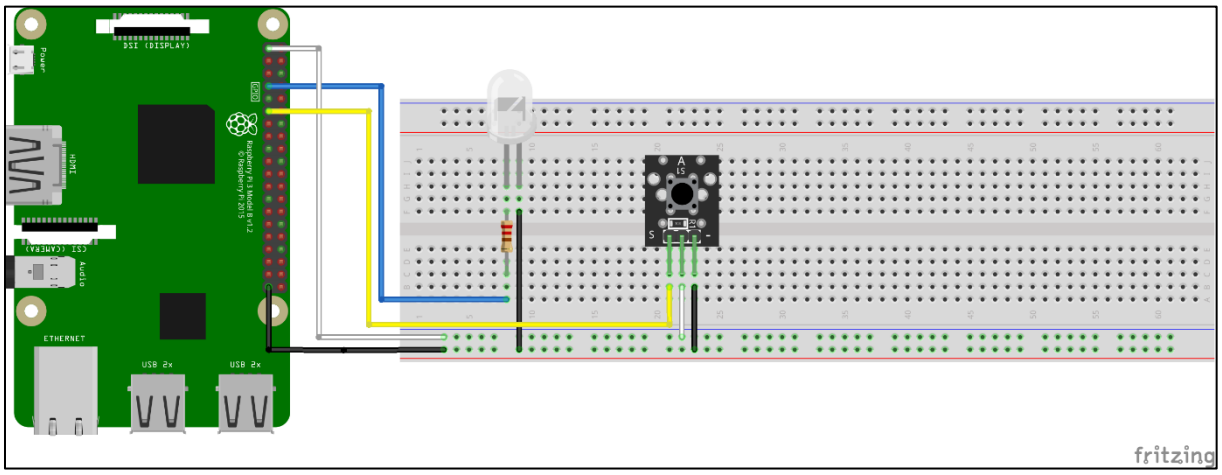

Abbildung 10: Schaltbild Aufgabe 5a

10 Aufgabe 5b: Dimmer

In dieser Aufgabe wirst du ein Python-Programm schreiben, das eine LED mit einem Drehsensor (KY040) steuert. Wenn der Drehsensor nach rechts gedreht wird, so soll die LED heller werden. Dreht man am Sensor nach links, soll sie wieder dunkler werden. Beachte dabei, dass der Helligkeitswert zwischen 0 und 1 liegen muss. Verbinde dazu eine LED und einen Drehsensor mit dem Raspberry Pi. Baue zudem einen Vorwiderstand für die LED ein!

Hinweise:

- Für die Ansteuerung des Drehsensors kannst du dich an der Dokumentation orientieren:
  - https://sensorkit.joy-it.net/de/sensors/ky-040
- Um die Helligkeit der LED steuern zu können solltest du im Programm statt „LED“ die Klasse „PWMLED“ verwenden:
  - https://gpiozero.readthedocs.io/en/stable/api_output.html#gpiozero.PWMLED
  - Wenn du im Programm die Variable „led“ erstellt hast, kannst du mit „led.value“ auf den Helligkeitswert zugreifen und diesen bearbeiten wie eine Variable. Der Wert darf allerdings nur zwischen 0.0 und 1.0 sein.
- Erstelle eine Funktion, die als Parameter einen Wert erhält, der die Richtung der Drehung des Sensors erhält. Auf Grundlage der Drehrichtung kannst du die LED steuern.
- Die Funktion für das Auslesen des Drehsensors soll unterscheiden können, ob eine Rechts- oder Links-Drehung ausgeführt wurde. Darauf soll die Funktion für die Steuerung der LED mit dem entsprechenden Parameter aufgerufen werden.

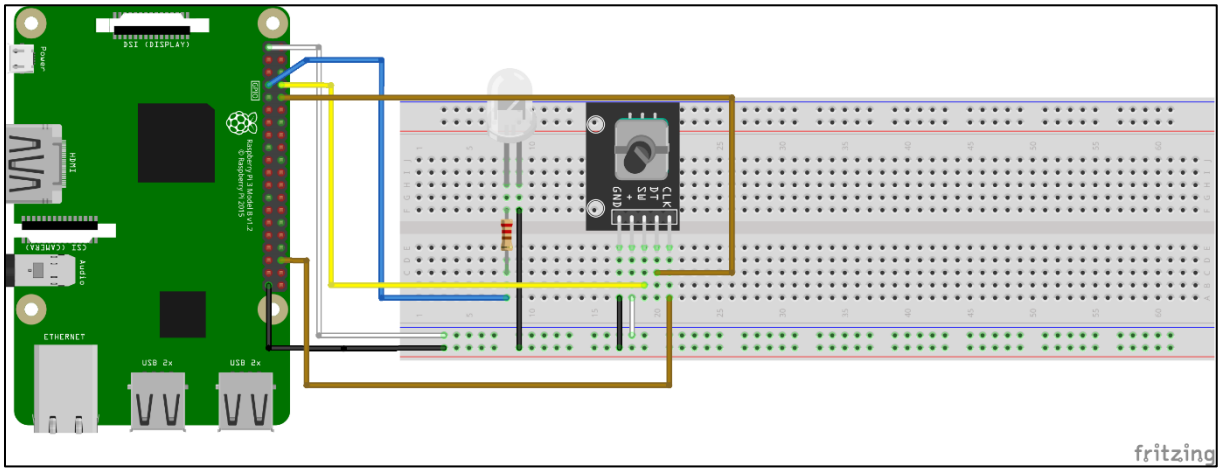

Abbildung 11: Schaltbild Aufgabe 5b

11 Aufgabe 5c: Farbspektrum

In dieser Aufgabe wirst du ein Python-Programm schreiben, das eine RGB-LED (KY009) mit einem Drehsensor (KY040) steuert. Die RGB-LED besteht aus drei LEDs, die rotes, grünes und blaues Licht emittiert. Wenn man die Helligkeit der einzelnen Farben variiert, werden diese „gemischt“ und stellen zusammen eine Farbe dar. Wenn der Drehsensor gedreht wird, sollen die Helligkeiten variiert werden, sodass ein Farbverlauf (siehe Abbildung 12) durchlaufen wird.

Abbildung 12: Farbspektrum

Hinweise:

- Um die Helligkeit der LED steuern zu können solltest du im Programm statt „LED“ die Klasse „PWMLED“ verwenden:
  - https://gpiozero.readthedocs.io/en/stable/api_output.html#gpiozero.PWMLED
  - Wenn du im Programm die Variable „led“ erstellt hast, kannst du mit „led.value“ auf den Helligkeitswert zugreifen und diesen bearbeiten wie eine Variable. Der Wert darf allerdings nur zwischen 0.0 und 1.0 sein.
- Globale Variable für Farbverlauf:
  - Erstelle eine globale Variable, die den Farbverlauf speichert. Dazu benötigst du eine Liste, die die einzelnen Helligkeitswerte zwischen 0.0 und 1.0 speichert. Diese Liste wird Tupel mit jeweils drei Werten enthalten, die die Helligkeitswerte für die RGB-Komponenten repräsentieren. Ein Beispiel für einen solchen Tupel ist (1,0,0), wobei die Werte für (Rot, Grün, Blau) stehen.
  - Erstelle ebenso eine globale Variable für den Index der Liste. Der Index ist notwendig, um auf die einzelnen Elemente der Liste zuzugreifen. Der Index steht bildlich gesehen für die Position im Farbspektrum (siehe Abbildung 12).
- Funktion zur Erstellung des Farbspektrums:
  - Schreibe eine Funktion, die das Farbspektrum in einer Liste speichert.
  - Diese Funktion besteht aus sechs for-Schleifen. Jede for-Schleife bildet einen Übergang zwischen zwei Farben ab:
    - Rot zu Gelb (Rot bleibt an, Grün geht von 0 zu 1)
    - Gelb zu Grün (Grün bleibt an, Rot geht von 1 zu 0)
    - Grün zu Cyan (Grün bleibt an, Blau geht von 0 zu 1)
    - Cyan zu Blau (Blau bleibt an, Grün geht von 1 zu 0)
    - Blau zu Magenta (Blau bleibt an, Rot geht von 0 zu 1)

        - Magenta zu Rot (Rot bleibt an, Blau geht von 1 zu 0)
    - In den For-Schleifen werden die Tupel mit den Helligkeitswerten der Liste hinzugefügt. (liste.append((1,0,0))
    - Die Farbwerte, die in einer For-Schleife einen Übergang von 0 zu 1 bzw. von 1 zu 0 durchlaufen, sollen in einer bestimmten Schrittweite bearbeitet werden. Das heißt, die Farbe Grün soll nicht direkt von 0 auf 1 übergehen, sondern bspw. den Verlauf „0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1“ haben. Auf diese Weise entsteht ein weicher Übergang zwischen den Farben.
- Funktion zur Einstellung der LED-Farbwerte:
    - Schreibe eine Funktion zur Anpassung der LED-Helligkeit, die drei Argumente (r, g, b) akzeptiert und die Helligkeitswerte der entsprechenden LEDs setzt.
- Farbindex erhöhen oder verringern:
    - Implementiere Funktionen (incrementColor() und decrementColor()), um den Index der Liste (globale Variable) zu erhöhen oder zu verringern. „incrementColor“ wird aufgerufen, wenn der Drehsensor nach rechts gedreht wird. Damit soll der Farbverlauf in eine Richtung durchlaufen werden. „decrementColor“ soll bei einer Links-Drehung in die entgegengesetzte Richtung steuern.
- Für die Ansteuerung des Drehsensors und der RGB-LED kannst du dich an der Dokumentation orientieren:
    - https://sensorkit.joy-it.net/de/sensors/ky-040
    - https://sensorkit.joy-it.net/de/sensors/ky-009

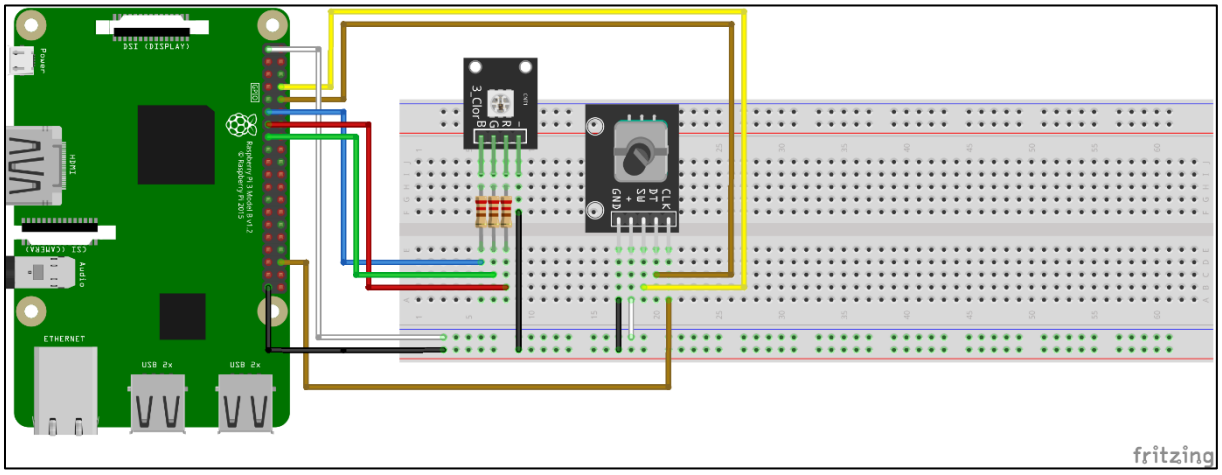

Abbildung 13: Schaltbild Aufgabe 5c

12 Aufgabe 5e: LED-Kette

In dieser Aufgabe wirst du ein Python-Programm schreiben, das drei LEDs mit einem Drehsensor (KY040) steuert. Die LEDs sind in einer Reihe aufgebaut und zu Beginn sind alle aus. Dreht man den Drehsensor nach rechts, so werden die LEDs schrittweise von links nach rechts angeschaltet. Dreht man nach links, sollen die LEDs von rechts nach links schrittweise wieder ausgeschaltet werden. (Schrittweise heißt hier LED für LED). Verbinde dazu drei LEDs und einen Drehsensor mit dem Raspberry Pi. Baue zudem Vorwiderstände für die LEDs ein.

Hinweise:

- Für die Ansteuerung des Drehsensors kannst du dich an der Dokumentation orientieren:
  - https://sensorkit.joy-it.net/de/sensors/ky-040
- Verwende eine globale Variable, die die Anzahl der leuchtenden LEDs repräsentiert.
- Definiere eine Funktion, welche die Drehrichtung des Sensors als Parameter erhält. Anhand der Richtung und der bereits leuchtenden LEDs kannst du in mehreren If-Blöcken die LEDs ansteuern.

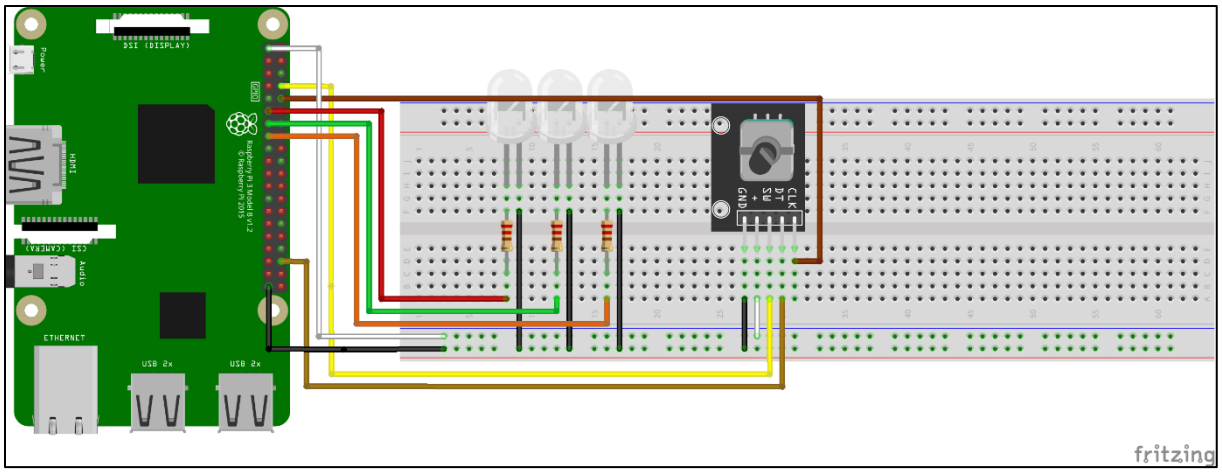

Abbildung 14: Schaltbild Aufgabe 5e

13 Aufgabe 6a: Morse-Code (Ziffern)

In dieser Aufgabe wirst du ein Python-Programm schreiben, das einen Taster (KY004) ansteuert. Der Taster soll als Eingabegerät für einen Morse-Code dienen. Das Programm soll den Morse-Code empfangen, dekodieren und anzeigen. Die Anzeige soll den eingegebenen Code, sowie die dekodierte Ziffer enthalten.

Hinweise:

- Eine Erklärung des Morse-Codes für Ziffern findest du hier:
  - https://de.wikipedia.org/wiki/Morsecode#:~:text=Lateinische%20Buchstaben-,Ziffern,-Umlaute%2C%20Ligaturen
  - In dieser Aufgabe soll lediglich die Verarbeitung von Ziffern und keine Buchstaben und sonstigen Zeichen beinhalten. Das vereinfacht die Aufgabe, da die Morse-Codes für Ziffern gleichlang (5 Signale) sind.
- Speichere den Morse-Code für die Ziffern in einer Liste, sodass das i-te Element der Liste die Ziffer i im Morse-Code repräsentiert. Den Morse-Code kannst du dabei in „1“ für „lang“ und „0“ für „kurz“ kodieren.
  - Dies kannst du beispielsweise so umsetzen:

```
morse_code = [
        (1,1,1,1,1),        #0
        (0,1,1,1,1),        #1
        (0,0,1,1,1),        #2
        (0,0,0,1,1),        #3
        (0,0,0,0,1),        #4
        (0,0,0,0,0),        #5
        (1,0,0,0,0),        #6
        (1,1,0,0,0),        #7
        (1,1,1,0,0),        #8
        (1,1,1,1,0),        #9
]
```

- Den eingegebenen Morse-Code kannst du ebenso in einer Liste speichern.
- Speichere zudem immer die Länge der aktuellen Eingabe, um nach jedem fünften Zeichen die Eingabe zu dekodieren und zurückzusetzen.
- Verwende in diesem Beispiel die Event-orientierte Programmierung, um Funktionen aufzurufen, wenn der Taster gedrückt bzw. losgelassen wird. Die Funktionen sollen dabei helfen, bei einer Eingabe zwischen „kurz“ und „lang“ zu unterscheiden. Dazu ist eine Zeitmessung notwendig. Erstelle also zwei Funktionen „onPressed()“ und „onRelease()“ und verbinde sie mit den zugehörigen Events:

```
#Funktionen und Ereignisse (Events) verbinden
button.when_pressed = onPressed
button.when_released = onRelease
```

- Die Funktion „onPressed“ speichert die aktuelle Zeit in Nanosekunden in einer globalen Variable „start_time“.
- Die Funktion „onRelease“ wird die Zeit beim Loslassen des Tasters in „end_time“ hinterlegen und die Differenz berechnen. Wenn die Differenz (also die Dauer des Tasterdrucks) größer als bspw. 0,35 Sekunden ist, dann wird die Eingabe als „lang“, andernfalls als „kurz“ gespeichert.

- Das Hauptprogramm soll eine Dauerschleife beinhalten, die bei jeder fünften Eingabe den Morse-Code und die Ziffer ausgibt. Für die Wandlung des Codes in eine Ziffer implementierst du eine Funktion. Bei jeder fünften Eingabe soll zudem die Variable für die Zählung der eingegebenen Zeichen sowie die Eingabeliste zurückgesetzt / geleert werden.

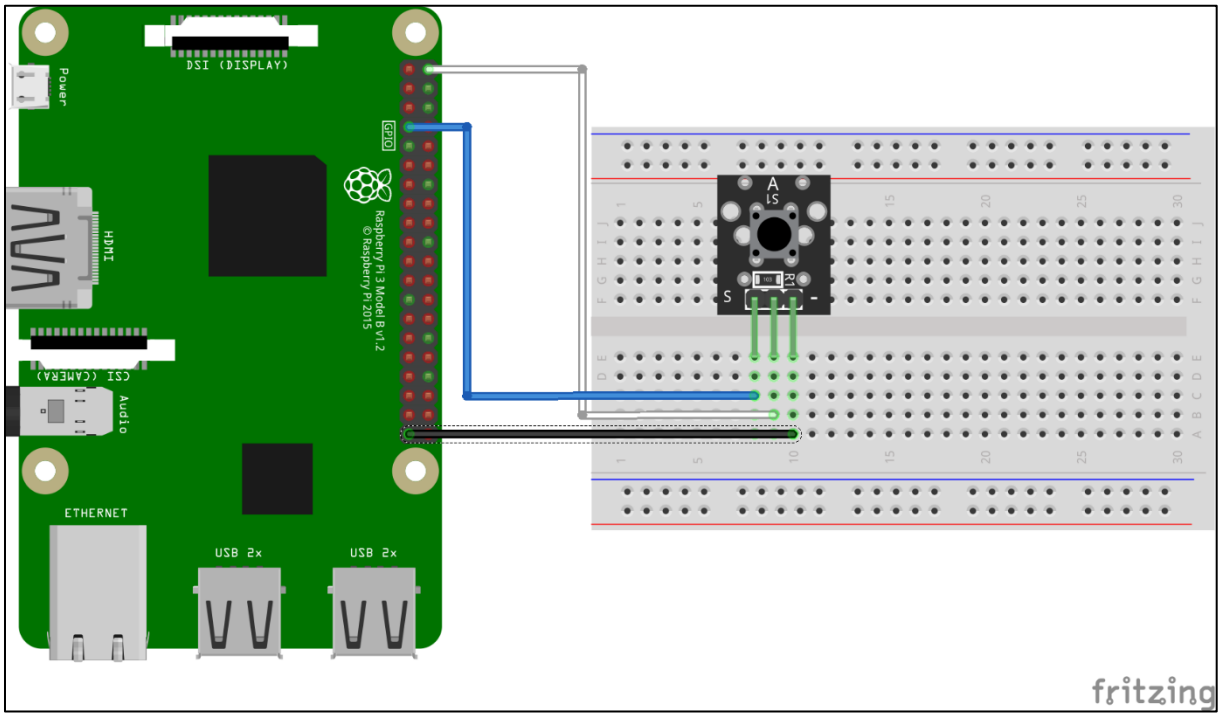

Abbildung 15: Schaltbild Aufgabe 6a