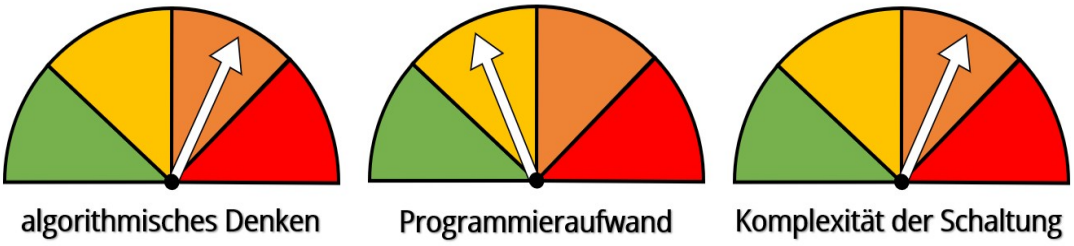
TECHNISCHE UNIVERSITÄT DRESDEN

MicroPython

Station 3 | Zeitmessung



Dirk Vorderstraße (Wikimedia Commons), „Blitzer in Hamm, Münsterstraße (10572716443)“, CC-BY-2.0

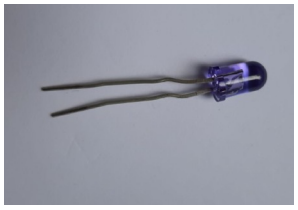
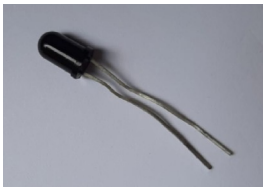
Professur für Didaktik der Informatik

UM WAS WIRD ES IN DIESER STATION GEHEN?

Lichtschranken begegnen euch überall im Alltag, zum Beispiel in Aufzügen oder Alarmanlagen. Normalerweise bemerkst du das nicht, da es sich um unsichtbares Licht handelt: dem **Infrarot-**Licht.
Wenn eine Lichtschranke durchbrochen wird, zum Beispiel weil ein Einbrecher durch das Fenster klettert, kann das einen Alarm auslösen. Im Fall des Aufzugs wird die Tür offen gehalten, wenn jemand im Eingang steht.
Auch im Sport werden Lichtschranken eingesetzt und zwar zum genauen Messen von Zeiten. Genau das wirst du in dieser Station machen.

BENÖTIGTE BAUTEILE

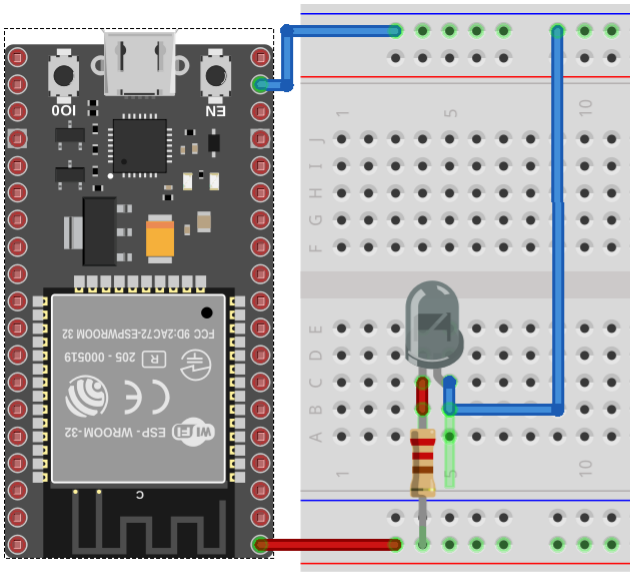
Zusätzlich zum ESP32 und dem Steckbrett brauchst du folgende Bauteile:

2 x IR-LED	Eine Infrarot-LED funktioniert wie eine LED, nur dass sie anderes Licht ausstrahlt. Und zwar unsichtbares Infrarotlicht.	
2 x IR Photodiode	Eine Infrarot-Diode kann erkennen, ob sie mit Infrarot-Licht bestrahlt wird. Zusammen mit der Infrarot-LED kannst du daraus eine Lichtschranke bauen.	
2 x 100 kΩ Widerstand	Widerstände schützen Bauteile vor Überhitzung durch zu viel Strom. Beachte die Farbe des Widerstandes, er sollte braun-schwarz-gelb sein.	
2 x 220 Ω Widerstand	Auch dieser Widerstand schützt die Bauteile. Seine Farbe ist rot-braun-gold.	

AUFGABE 1 – LICHTSCHRANKE BAUEN

Als ersten Schritt zur Zeitmessung baust du eine Lichtschranke.
Dafür erstellst du eine unsichtbare Lichtverbindung zwischen IR-LED und IR-Photodiode.

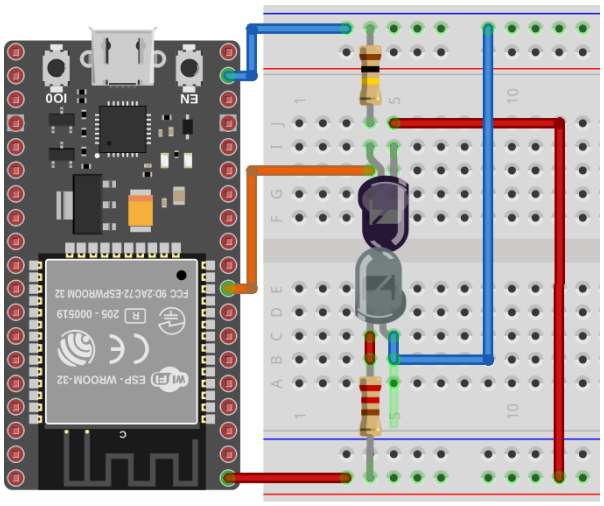
1. IR-LED anbringen



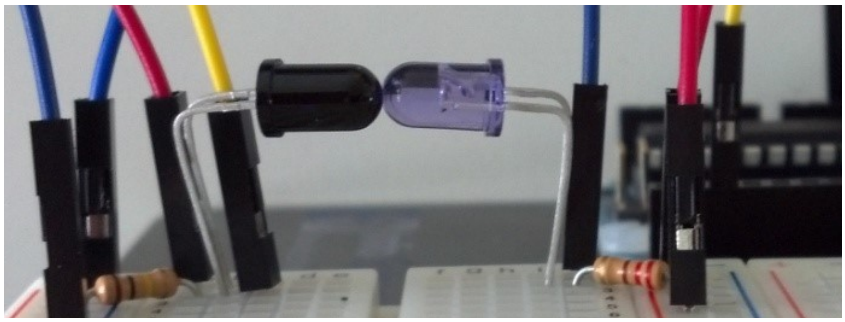
Verbinde die äußeren Leisten mit dem +Pol und –Pol. Verbinde die IR-LED über einen 220 Ω – Widerstand mit der +Leiste. Achte darauf, dass du das kurze Beinchen an die +Leiste schließt! Das kurze Beinchen der LED verbindest du über ein Kabel an die –Leiste.

2. IR-LED testen
   Wenn du den ESP32 an den Computer anschließt, wirst du sehen, dass … nichts passiert. Obwohl du das Leuchten der IR-LED nicht sehen kannst, kann sie trotzdem aktiv sein. Um das zu testen, benutze eine Handykamera und richte sie auf die LED. Beobachte das Display der Kamera. Sie kann den Lichtstrahl nämlich sichtbar machen.

3. IR-Photodiode anbringen
   Wenn die IR-LED auf dem Kamerabildschirm leuchtet, kannst du die Photodiode anbringen.



Bringe die Photodiode gegenüber von der IR-LED an. Verbinde das lange Beinchen der Photodiode über einen 100kΩ-Widerstand mit der –Leiste. Das kurze Beinchen verbindest du mit der +Leiste. Verbinde zuletzt noch das lange Beinchen mit einem Pin des ESP32-Boards.
Das Infrarotlicht der LED leuchtet jetzt nach oben, sodass die IR-Photodiode davon nichts mitbekommt. Damit die Diode das Licht empfangen kann, musst du die beiden wie im Bild zueinander biegen.



AUFGABE 2 – LICHTSCHRANKE TESTEN

Jetzt sollte eine Lichtverbindung zwischen der LED und der Diode bestehen. Da du die Verbindung leider nicht sehen kannst, überprüfen wir sie durch Programmieren. Dabei soll der ESP32 in der Konsole einen Text ausgeben, wenn die Lichtschranke unterbrochen bzw. wieder geschlossen wird.

1. Bibliotheken importieren
   Um auf die Pins am ESP32-Board zugreifen zu können, benötigt MicroPython die Bibliothek `machine`. Um den zeitlichen Ablauf des Programms steuern zu können, solltest du zusätzlich die Bibliothek `time` verwenden. Weißt du noch, wie du Bibliotheken importieren kannst? Falls nicht, schau einfach nochmal in der Einführungsstation nach.

2. Pin-Variable definieren und digitalen Pin verbinden
   Ob die Lichtschranke unterbrochen wurde, wird mithilfe der IR-Photodiode überprüft. Erstelle für sie eine Variable `photodiodeStart`.
   Die Photodiode gibt analoge Messwerte an den ESP32 weiter. Damit er sie richtig interpretieren kann, musst du sie in digitale Werte umwandeln. Nutze hierfür den `ADC()` – Befehl:

```
photodiodeStart = ADC( Pin( _____ ) )
```

Hier wird angegeben, um welchen Pin es geht.

3. Wenn, sonst – Ausdruck
   Um den Zustand der Lichtschranke ermitteln zu können, brauchst du einen **Wenn, sonst** – Ausdruck. Also auf gut Deutsch:
   Wenn die Lichtschranke durchbrochen ist, soll *„Lichtschranke durchbrochen!“* in der Konsole angezeigt werden. **Sonst** soll der ESP32 den Text *„Lichtschranke intakt“* ausgeben.
   Füge folgenden Ausdruck innerhalb einer `while True` – Schleife in deinen Code ein:

```
if [Lichtschranke durchbrochen]:
     print("Lichtschranke durchbrochen!")
else:
     print("Lichtschranke intakt")
```

4. Ist die Lichtschranke intakt?
   Wenn die Lichtschranke intakt ist, bedeutet das, dass sich kein Gegenstand zwischen IR-LED und IR-Photodiode befindet. Um herauszufinden, ob das so ist, benötigen wir die Information von der IR-Photodiode, ob sie Infrarotlicht empfängt. Falls ja, gibt sie Strom an unseren Messpin weiter. Wenn nicht, gibt sie keinen Strom weiter. Du musst am Pin also ablesen, ob an ihm ein Strom fließt.
   Setze folgenden Befehl an die passende Stelle im **Wenn, sonst** – Ausdruck:

   ```
   photodiodeStart.read() == 0
   ```

5. Wartebefehl einfügen und Lichtschranke testen
   Wie du bestimmt schon gemerkt hast, werden in der Konsole in kürzester Zeit sehr viele Zeilen ausgegeben. Um dies zu umgehen, ist es sinnvoll, zwischen den Messungen kurze Pausen einzufügen. Nutze hierfür den `sleep()` - Befehl, den du schon aus der Einführungsstation kennst:

   ```
   sleep( _____ )
   ```

   Wartezeit in Sekunden

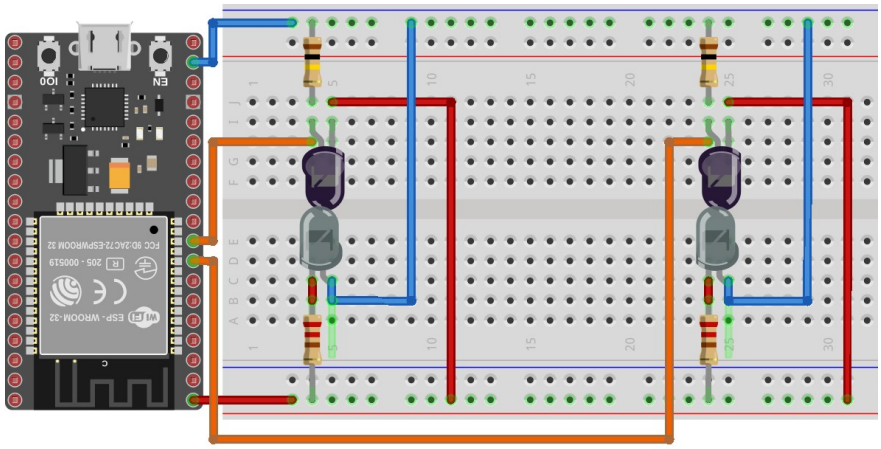
   An welchen beiden Stellen deines Codes musst du den Befehl einfügen?

   Wie groß kann die maximale Wartezeit in etwa sein, damit auch sehr kurze Unterbrechungen der Lichtschranke bemerkt werden? Teste die Lichtschranke, indem du das Papierauto zwischen den Kopf der IR-LED und der IR-Photodiode schiebst und anschließend wieder entfernst.

AUFGABE 3 – ZWEITE LICHTSCHRANKE

Du hast jetzt erfolgreich eine Lichtschranke gebaut und programmiert. Bei der Zeitmessung braucht man jedoch zwei Lichtschranken, eine am Start- und eine am Zielort. Deswegen sollst du eine weitere Lichtschranke bauen.

1. Lichtschranke anbringen



Gehe zum Abringen der zweiten Lichtschranke genauso vor, wie bei der ersten. Denke daran, die langen Beinchen der IR-LED und IR-Photodiode an die +Leiste zu schließen. Das kurze Beinchen musst du mit der –Leiste verbinden. Schließe die IR-Photodiode an einen Pin des ESP32 an. Biege die IR-LED und IR-Photodiode wieder so, dass sie eine Lichtschranke bilden.

2. Pin-Variable definieren und digitalen Pin verbinden
   Lege wie in Aufgabe 2 auch für die zweite IR-Photodiode eine Pin-Variable fest. Gib ihr den Namen `photodiodeZiel`. Verwende auch hier wieder den `ADC()` – Befehl, um die analogen Messwerte der IR-Photodiode für den ESP32 zu digitalisieren.

3. Lichtschranke 2 testen
   Um auch die zweite Lichtschranke zu testen, musst du einfach den vorigen **Wenn, sonst** - Ausdruck kopieren und darunter einfügen. Danach musst du noch die Variable im **Wenn** - Ausdruck abändern:

```
if [Zweite Lichtschranke durchbrochen]:
     print("Ziel-Lichtschranke durchbrochen!")
else:
     print("Ziel-Lichtschranke intakt")
```

Füge auch hier wieder an den beiden richtigen Stellen Wartebefehle ein, um den Abstand zwischen den Messungen zu vergrößern.
Installiere das neue Programm auf dem ESP32. Wenn du jetzt die zweite Lichtschranke mit dem Papierauto durchbrichst, sollte der obige Text auf der Konsole ausgegeben werden. Achte darauf, dass beim Test die erste Lichtschranke nicht durchbrochen wird. Das könnte sonst das Ergebnis verfälschen.

AUFGABE 4 – ZEITMESSUNG

Du hast jetzt erfolgreich zwei Lichtschranken gebaut und programmiert. Nun sollst du die Zeitmessung vornehmen.

1. Messungsvariable
   Wenn die erste Lichtschranke durchbrochen wird, soll eine Zeitmessung gestartet werden. Da der ESP32 die Befehle innerhalb der `while True` – Schleife immer wieder ausführt, würde auch das Starten einer Messung immer wiederholt. Damit das Board nicht die ganze Zeit neue Messungen startet, soll es sich merken, ob eine Messung bereits gestartet wurde. Dafür brauchst du eine Variable. Definiere eine Variable `gestartet` und ordne ihr den Wert `0` zu. `0` bedeutet hier, dass die Zeitmessung noch nicht gestartet wurde. Wenn die Messung später gestartet wird, steht dafür eine `1`.

2. Messung starten
   Wenn die erste Lichtschranke, also die Startlichtschranke, durchbrochen wird, soll die Messung gestartet werden. Der Variable `gestartet` soll also der Wert `1` zugeordnet werden. In welche Klammern im Code musst du die Zuordnung einfügen?

```
if photodiodeStart.read() == 0:
    A  _____________________
else:
    B  _____________________
if photodiodeZiel.read() == 0:
    C  _____________________
else:
    D  _____________________
```

Antwort: _____

Füge die Zuordnung an der richtigen Stelle im Code ein.

3. Messung stoppen
   Jetzt kann sich der ESP32 merken, ob die Zeitmessung gestartet wurde. Fehlt noch, dass er registriert, wenn die Zeitmessung beendet wird. Die Messung soll beendet werden, wenn die Ziellichtschranke durchbrochen wird. Ordne dafür der Variable `gestartet` an der passenden Stelle den Wert `0` zu.

4. Zeitvariablen definieren und Speichern der Uhrzeiten
   Jetzt weiß der ESP32 schon, ob eine Messung bereits gestartet oder beendet wurde. Die Zeit wird aber noch nicht gestoppt. Dafür brauchst du den Befehl `time()`. Durch ihn kannst du herausfinden, wieviel Millisekunden seit dem Start des Programms vergangen sind. Er funktioniert quasi wie eine Uhrzeit.
   Allerdings misst er die Zeit nur in ganzen Sekunden. Da wir auch sehr kurze Intervalle messen wollen, müssen wir die Zeitwerte in Millisekunden *(Tausendstelsekunden)* umrechnen. Das geht so:

   ```
   int( round( time() * 1000 ) )
   ```

   Um die Zeit auszurechnen, die von Start bis Ziel gebraucht wurde, wendest du diese Formel an:

   benötigte Zeit von Start bis Ziel = Zieluhrzeit – Startuhrzeit

   Um die benötigte Zeit später auszurechnen, musst du die Start- und Zieluhrzeit in zwei Variablen speichern. Definiere deswegen zwei Variablen `startuhrzeit` und `zieluhrzeit`.

   Speichere zum Start der Messung die Uhrzeit in der `startuhrzeit` – Variable. Das sieht so aus:

   ```
   startuhrzeit = int( round( time() * 1000 ) )
   ```

   Füge den Befehl an der richtigen Stelle im Code ein.
   Wie könnte jetzt das Speichern der Zieluhrzeit aussehen? Speichere die Zieluhrzeit in der `zieluhrzeit` – Variable, wenn die Ziellichtschranke durchbrochen wurde.

5. Berechnen der benötigten Zeit
   Definiere eine weitere Variable, um das Ergebnis der obigen Formel zu speichern. Ist die Ziellichtschranke durchbrochen, soll die gebrauchte Zeit berechnet werden. Wichtig ist, dass du das erst machst, nachdem die Zieluhrzeit gespeichert wurde.

6. Wenn, sonst – Ausdruck erweitern

Vorhin wurde bereits erwähnt, dass der ESP32 nur eine Messung starten soll,
wenn die Startlichtschranke durchbrochen ist
und noch keine Messung gestartet wurde.
Das ist wichtig, da das Board sonst bei jedem Durchlauf der `while True` – Schleife eine neue Messung starten würde.
Deswegen musst du im Wenn, sonst – Ausdruck noch einen **und**-Teil hinzufügen.

```
if [Start-Lichtschranke durchbrochen] && ____________ == 0:
    ...
else:
    ...
```

Bedeutet, dass die Messung noch nicht gestartet wurde

Steht in der Programmierung für **UND**.

Hier musst du die Variable angeben, die speichert, ob die Messung bereits gestartet wurde

Den **und**-Teil musst du auch einfügen, wenn die Messung gestoppt werden soll. Hier soll der Befehl ausgeführt werden,
wenn die Ziellichtschranke durchbrochen ist
und die Messung gestartet wurde.
Wie sieht wohl der passende Wenn-Ausdruck für die Zielschranke aus?

```
if [Ziel-Lichtschranke durchbrochen] && ____________ == __:
    ...
else:
    ...
```

Hier musst du die Variable angeben, die speichert, ob die Messung bereits gestartet wurde

Soll bedeuten, dass die Messung schon gestartet wurde.

7. Messergebnis ausgeben und Programm testen

Nutze den `print()` - Befehl, um die gemessene Zeit in der Konsole auszugeben:

```
print( ____________ )
```

Hier musst du die Variable angeben, in der das Ergebnis gespeichert ist.

Die vorherigen Ausgabebefehle benötigst du nun nicht mehr – du kannst sie einfach löschen.
Installiere dein Programm auf dem ESP32. Starte eine Messung, indem du mit dem Papierauto die Startlichtschranke durchbrichst. Wenn du anschließend die Ziellichtschranke passierst, erscheint in der Konsole das Messergebnis in Millisekunden.

Fotos: RWTH Aachen, InfoSphere
Screenshots: fritzing electronics made by easy und Arduino IDE 1.8.12 (Linux)
Alle weiteren Grafiken: Patrick Binkert, EduInf@TUD