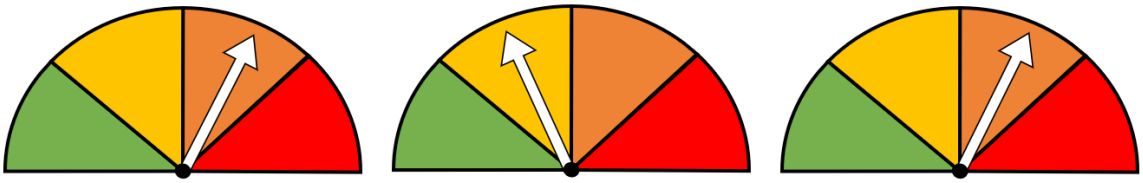
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 3 | Zeitmessung



Valentin_Photography, pixabay.com , Pixabay Lizenz

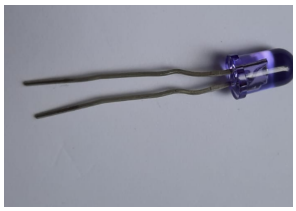
Professur für
Didaktik der Informatik

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 ein Alarm auslöst werden. Im Fall eines Aufzugs wird die Tür offen gehalten, wenn jemand im Eingang steht.
Auch im Sport werden Lichtschranken zum genauen Messen von Zeiten eingesetzt. Genau das wirst du in dieser Station machen.

BENÖTIGTE BAUTEILE

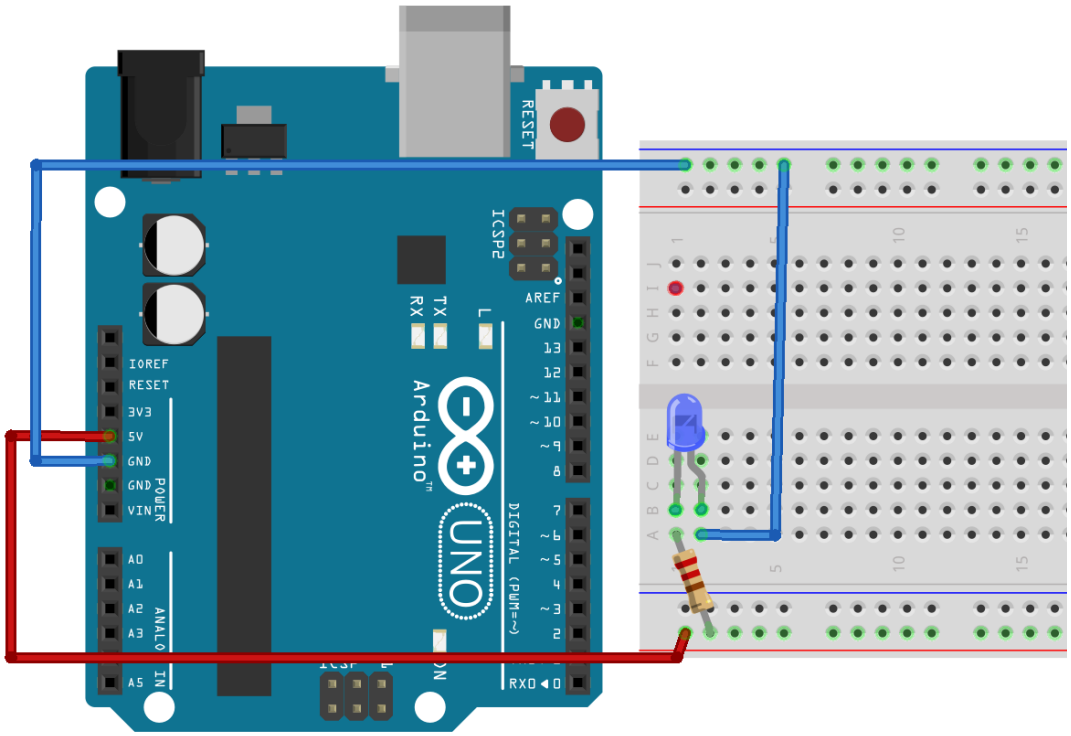
Zusätzlich zum Arduino und dem Steckbrett brauchst du folgende Bauteile:

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 die Lichtschranke.
Dafür richtest du den Sender des Infrarotlichts und den Empfänger aufeinander aus. Dadurch entsteht eine unsichtbare Lichtverbindung zwischen IR-LED und IR-Photodiode.

1. IR-LED anbringen

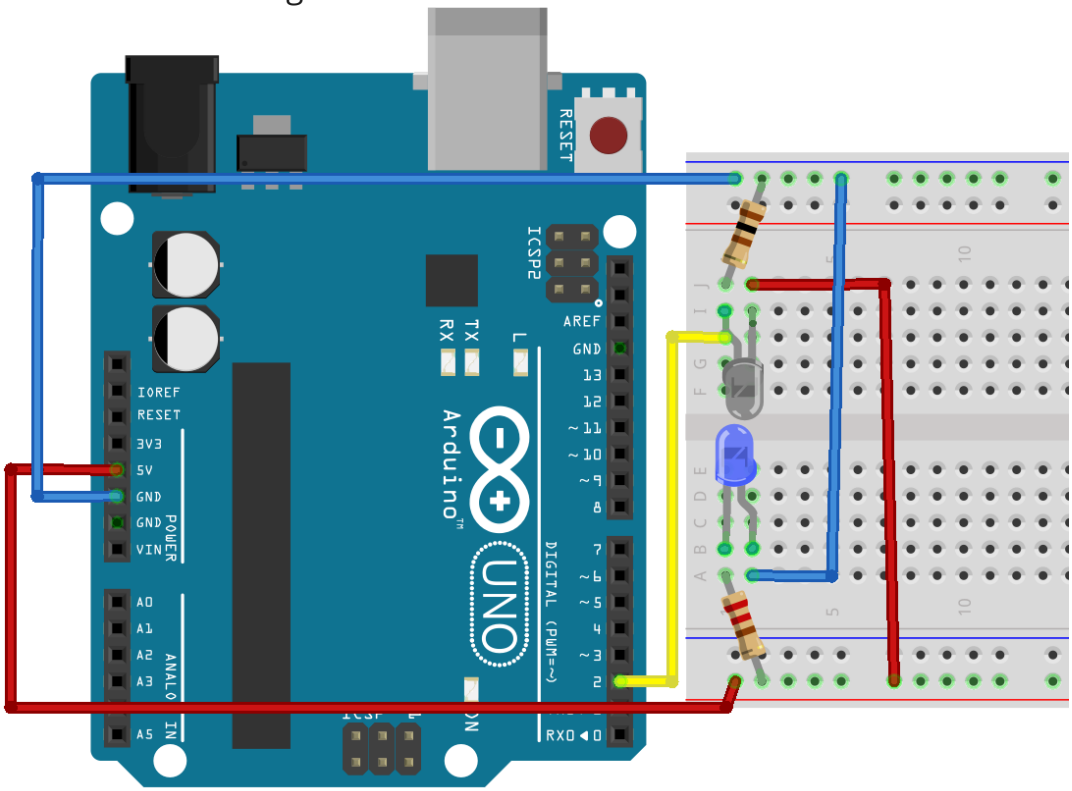


Verbinde die äußeren Leisten mit dem +Pol und -Pol. Verbinde die IR-LED über einen 220 Ω – Widerstand mit der +Leiste. Achte darauf, dass du das kurze Beinchen an die +Leiste schließt! Das kurze Beinchen der LED verbindest du über ein Kabel an die -Leiste.

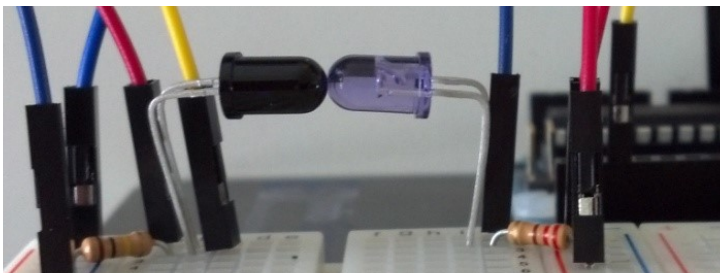
2. IR-LED testen
   Wenn du den Arduino an den Computer anschließen wirst du sehen, dass … nichts passiert. Obwohl du das Leuchten der IR-LED nicht sehen kannst, kann sie trotzdem aktiv sein. Um das zu testen, benutze eine Handykamera und richte sie auf die LED. Beobachte den Display der Kamera. Sie 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 dem digitalen Pin 2. 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 knicken.

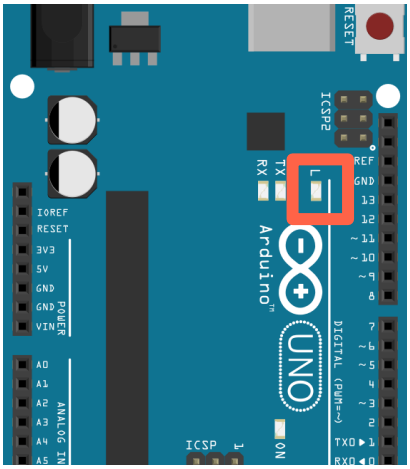


AUFGABE 2 – LICHTSCHRANKE TESTEN

Jetzt sollte es eine Lichtverbindung zwischen der LED und der Diode geben. Da du die Verbindung leider nicht sehen kannst, überprüfen wir das durch Programmieren. Dabei sollst du eine LED leuchten lassen, wenn die Lichtschranke durchbrochen ist. Wenn die Lichtschranke wieder intakt ist, soll die LED aufhören zu leuchten.

1. <u>LED und Dioden-Variable definieren</u>

   Die LED, die wir dafür benutzen, muss dieses Mal nicht extra angeschlossen werden, sondern ist bereits im Arduino eingebaut. Du findest sie neben dem Buchstaben L. Sie wird auch als Pin13 bezeichnet. Das heißt, sie wird behandelt, als wäre sie an Pin 13 angeschlossen.
   Wie in der Einführungsstation, musst du den Arduino wissen lassen, an welchem Pin die LED und die IR-Photodiode angeschlossen sind. Definiere dafür zwei Variablen. Falls du dafür Hilfe brauchst, schaue dir noch einmal in der Einstiegsstation Aufgabe 3 an.



2. <u>Digitale Pins verbinden</u>

   Außerdem muss der Arduino wissen, dass an Pin 13 etwas angeschlossen ist, das angeschaltet werden soll. Lege deswegen den `pinMode` der LED im `setup()`-Teil auf `OUTPUT` fest.
   Am Pin der Photodiode wird gelesen, ob die Lichtschranke intakt ist. Den `pinMode` vom Pin der IR-Photodiode musst du deswegen auf `INPUT` festlegen. Das funktioniert fast, wie bei `OUTPUT`:

```
pinMode (_____________ , INPUT ) ;
```

Legt fest, ob ein bestimmter Pin zur Ein- oder Ausgabe genutzt werden soll

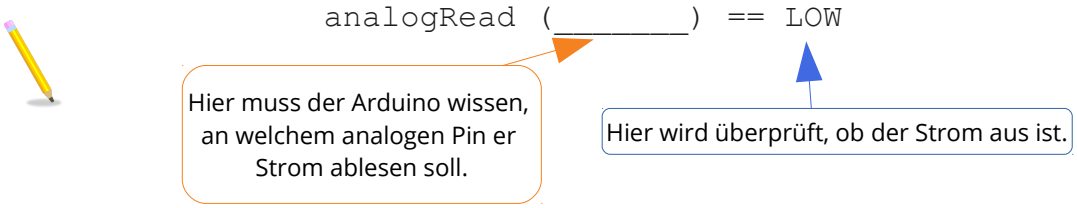
Hier wird angegeben, um welchen Pin es geht.

Der Pin wird zur Eingabe benutzt

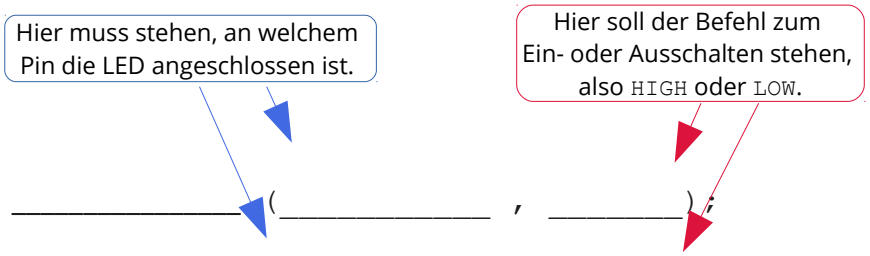
1. Wenn, sonst – Ausdruck
   Um das Ein- und Ausschalten der LED zu programmieren brauchst du einen **Wenn, sonst -** Ausdruck. Auf gut Deutsch:
   Wenn die Lichtschranke durchbrochen ist, soll die LED leuchten. **Sonst** leuchtet sie nicht.
   Füge diesen Ausdruck in den `loop()` - Teil ein.

```
if ( Lichtschranke durchbrochen ) {
LED leuchtet }
else {
LED leuchtet nicht }
```

2. Ist die Lichtschranke intakt?
   Wenn die Lichtschranke intakt ist, bedeutet das, dass sich kein Gegenstand zwischen IR-LED und IR-Photoiode befindet. Um das herauszufinden brauchen wir die Information von der IR-Photodiode, ob sie Infrarotlicht empfängt. Falls ja, gibt sie Strom an den Pin weiter. Falls nicht, gibt sie keinen Strom weiter. Du musst also am Pin ablesen, ob dort Strom fließt. Füge folgenden Befehl an die passende Stelle im **Wenn, sonst** - Ausdruck ein.



3. An- und Ausschalten der LED
   Damit die LED leuchtet oder nicht, musst du sie An- und Ausschalten. Die Befehle dafür kennst du bereits. Wie lauten sie?



Einschalten: _____(_____ , _____);

Ausschalten: _____ (_____ , _____);

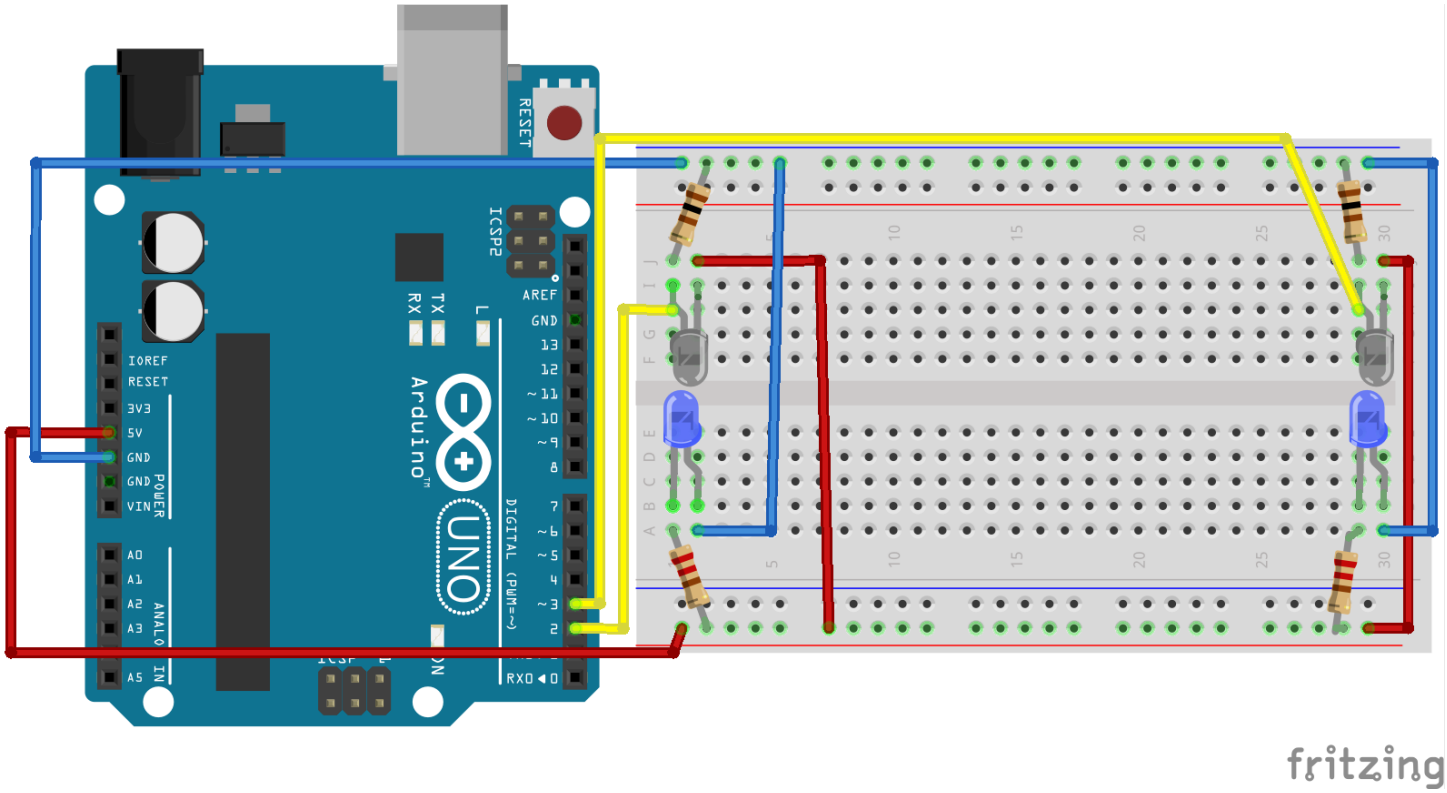
Füge die Befehle an die passenden Stellen des Wenn, sonst-Ausdrucks ein.

4. Lichtschranke testen
   Installiere das Programm auf dem Arduino. Teste das Programm, indem du das Papierauto zwischen den Kopf der IR-LED und der Diode schiebst. Jetzt sollte die LED angehen. Wenn du das Papier wieder entfernst, sollte die LED wieder ausgehen.

AUFGABE 3 – ZWEITE LICHTSCHRANKE

Du hast jetzt erfolgreich eine Lichtschranke gebaut und programmiert. Bei der Zeitmessung braucht man jedoch 2 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 Bein musst du mit der -Leiste verbinden. Schließe die IR-Photodiode an digitalen Pin 3 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. Gebe ihr einen anderen Namen. Setze außerdem ihren `pinMode` auf `OUTPUT`.

3. Lichtschranke 2 testen

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

   ```
   if ( Lichtschranke2 durchbrochen ) {
   LED leuchtet }
   else {
   LED leuchtet nicht }
   ```

   Installiere das neue Programm auf dem Arduino. Wenn du jetzt die zweite Lichtschranke mit dem Papierauto durchbrichst, sollte die LED leuchten. Achte darauf, dass im Test die erste Lichtschranke nicht durchbrochen wird. Das könnte sonst das Testergebnis fälschen.

AUFGABE 4 – ZEITMESSUNG

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

1. Löschen der else-Anweisungen
   Auch wenn die else-Anweisungen praktisch waren, um die Lichtschranken zu testen, wirst du sie ab jetzt nicht mehr brauchen. Lösche sie deswegen, um später den Überblick im Programm nicht zu verlieren.

```
if ( Lichtschranke durchbrochen ) {
LED leuchtet }
else {
LED leuchtet nicht }
if ( Lichtschranke2 durchbrochen ) {
LED leuchtet }
else {
LED leuchtet nicht }
```

2. Messungsvariable
   Wenn die erste Lichtschranke durchbrochen wird, soll eine Zeitmessung gestartet werden. Da der Arduino die Befehle im loop() - Teil immer wieder ausführt, würde auch das Starten einer Messung immer wiederholt. Damit der Arduino nicht die ganze Zeit neue Messungen macht, soll er sich merken, wenn eine Messung bereits gestartet wurde. Das soll eine Variable erledigen. Wenn keine Messung läuft, soll sie der Zahl 0 zugeordnet sein. Ist eine Messung gerade am Laufen, soll ihr die Zahl 1 zugeordnet sein. Am Anfang des Programms wurde die Messung noch nicht gestartet. Definiere eine Variable `gestartet` und ordne ihr die Zahl 0 zu.

```
int gestartet = 0;
```

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

A `if (digitalRead(Lichtschranke 1) == LOW ) { _____ }`
B `if (digitalRead(Lichtschranke 2) == LOW) { _____ }`

Antwort: _____

Füge die Zuordnung an der richtigen Stelle im Code ein.

4. Messung stoppen
   Jetzt kann sich der Arduino merken, ob die Zeitmessung gestartet wurde. Fehlt noch, dass er sich merkt, wenn die Zeitmessung beendet wird. Der Variable `gestartet` musst du dann wieder die Zahl 0 zuordnen. An welcher Stelle im Code musst du diese Zuordnung einfügen?

A `if (digitalRead(Lichtschranke 1) == LOW ) { _____ }`
B `if (digitalRead(Lichtschranke 2) == LOW) { _____ }`

Antwort: _____

Füge die Zuordnung an der richtigen Stelle im Code ein.

5. Zeitvariablen definieren
   Jetzt kann sich der Arduino merken, ob eine Messung schon gestartet oder beendet wurde. Die Zeit wird aber noch nicht gestoppt. Dafür brauchst du den Befehl `millis()`. Durch ihn kannst du herausfinden, wieviel Millisekunden seit dem Start des Programms vergangen sind. Er funktioniert quasi wie eine Uhrzeit. Um die Zeit auszurechnen, die von Start bis Ziel gebraucht wurde, wendest du diese Formel an:

   gebrauchte Zeit von Start bis Ziel = Zieluhrzeit – Startuhrzeit

   Um die gebrauchte Zeit später auszurechnen, musst du die Start- und Zieluhrzeit in zwei Variablen speichern. Definiere deswegen drei Variablen `startuhrzeit`, `zieluhrzeit` und `gebrauchtezeit`. Verwende dieses mal nicht `int`, sondern `unsigned long`, um die Variablen zu definieren. `Unsigned long` kommt zum Einsatz, wenn besonders hohe Zahlen gespeichert werden.

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

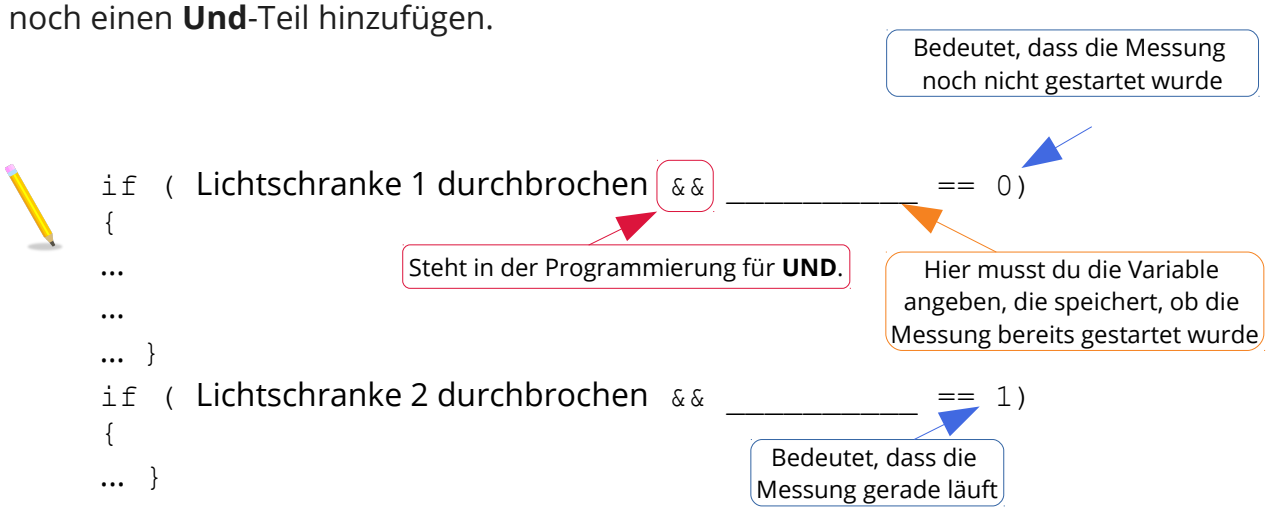
   ```
   startuhrzeit = millis();
   ```

   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.

7. Berechnen der gebrauchten Zeit
   Ist die Ziellichtschranke durchbrochen, soll die gebrauchte Zeit berechnet werden. Benutze dafür die obenstehende Formel. Füge die Berechnung ein, nachdem die Zieluhrzeit gemessen wurde.
   Überprüfe dann, ob du alle Bestandteile in deinem Code umgesetzt hast:

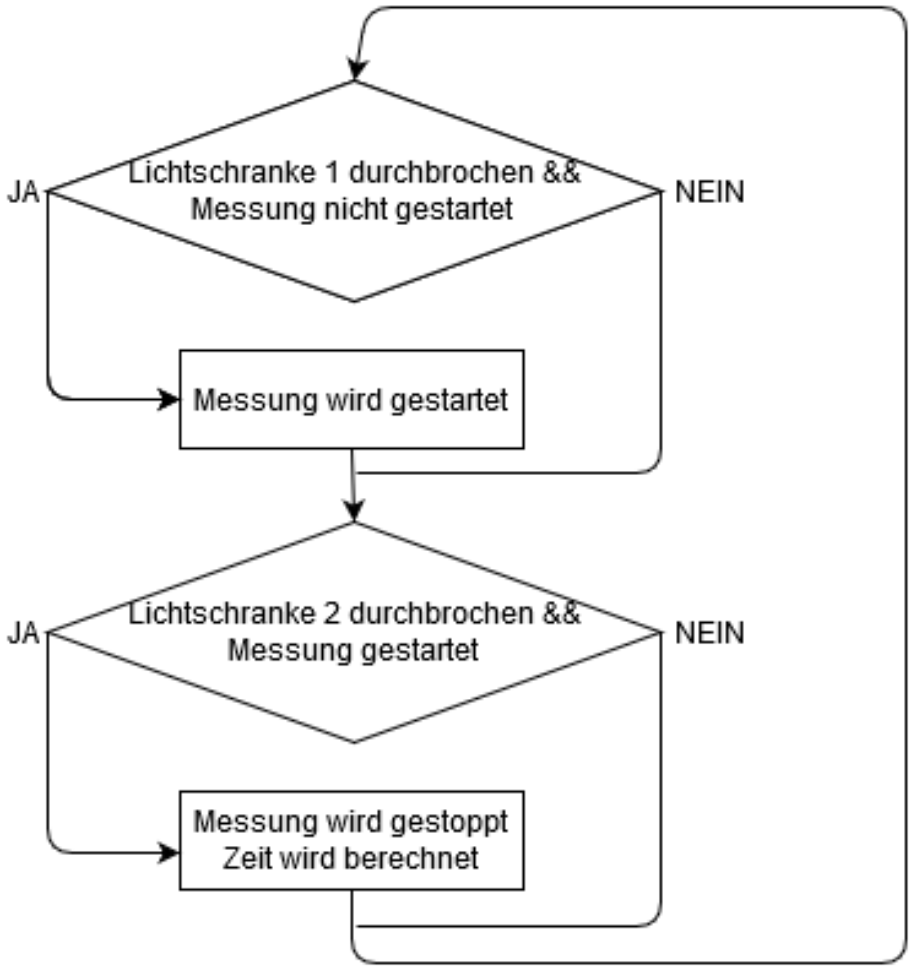
```
if ( Lichtschranke durchbrochen ) {
LED leuchtet
Variable gestartet wird 1 zugeordnet
Startuhrzeit wird gemessen }
if ( Lichtschranke2 durchbrochen ) {
LED leuchtet
Variable gestartet wird 0 zugeordnet
Zieluhrzeit wird gemessen
gebrauchte Zeit wird berechnet }
```

8. Wenn,sonst - Ausruck erweitern
   Vorhin wurde bereits erwähnt, dass der Arduino nur eine Messung starten soll,
   wenn die Startlichtschranke durchbrochen ist
   und noch keine Messung gestartet wurde.
   Das ist wichtig, da sonst der Arduino bei jedem Durchlauf des `loop()` - Teils neue Messungen starten würde.
   Damit das tatsächlich funktioniert, musst du im Wenn, sonst-Ausdruck noch einen **Und**-Teil hinzufügen.



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.

Das bisherige Programm hat diesen Ablauf:



Du hast das Programm erfolgreich installiert, aber es passiert nichts? Keine Sorge! Der Arduino misst die Geschwindigkeit. Er hat nur noch keine Möglichkeit uns das Ergebnis mitzuteilen. Schließlich hat er keinen Bildschirm, um uns das Ergebnis anzuzeigen. Oder doch?

AUFGABE 5 – MESSERGEBNIS ANZEIGEN

Benutze den seriellen Monitor, um dir das Ergebnis der Zeitmessung anzeigen zu lassen. Teste dein Programm.

WAS IST EIN SERIELLER MONITOR?

Tatsächlich hat der Arduino einen Bildschirm, auf dem er uns Ergebnisse anzeigen kann. Diesen kannst du öffnen, wenn du in der Programmierumgebung rechts oben auf *Serieller Monitor* klickst.

Wie immer macht der Arduino aber nichts alleine. Du musst ihm also befehlen, das Ergebnis anzuzeigen.

1. Seriellen Monitor starten
   Damit der Arduino weiß, dass du den seriellen Monitor benutzen willst, musst du am Anfang des setup()-Teils folgenden Befehl schreiben:

```
void setup() {
Serial.begin(9600);
...
```

Startet den seriellen Monitor.

mit einer Übertragungsrate von 9600 Baud.

2. Ergebnis durch seriellen Monitor anzeigen
   Um das Ergebnis sehen zu können, muss der serielle Monitor es nach der Berechnung der gebrauchten Zeit „drucken“. Dafür brauchst du folgenden Befehl:

```
Serial.println(__________);
```

Der serielle Monitor druckt das, was in der Klammer steht.

Hier musst du die Variable angeben, in der die gebrauchte Zeit gespeichert ist.

3. Installieren und testen
   Installiere das Programm auf dem Arduino. Öffne dann den seriellen Monitor in der Programmierumgebung. Durchbreche mit dem Papierauto zuerst die Startlichtschranke und dann die Ziellichtschranke. Auf dem seriellen Monitor sollte jetzt das Messergebnis in Millisekunden erscheinen.

Super! Du hast es geschafft, eine Zeitmessung zu programmieren.