Station 3 - Geschwindigkeitsmessung

Unsichtbares Licht und Geschwindigkeit

Lichtschranken begegnen euch überall im Alltag, zum Beispiel in Aufzügen oder bei automatischen Türklingeln in Geschäften.

Auch wenn ihr ihnen ständig über den Weg lauft, merkt ihr es häufig nicht, denn sie arbeiten mit unsichtbarem Licht: dem **Infrarot-Licht (kurz IR-Licht)**.

Lichtschranken funktionieren im Prinzip wie Schalter. Die Einsatzzwecke sind dabei sehr vielfältig. Heute werdet ihr zwei IR-Lichtschranken einsetzen, um Geschwindigkeiten zu messen. Anwendung findet das zum Beispiel bei der Zeitmessung im Sport.

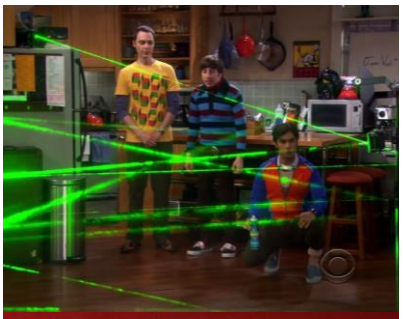


Abb. 1: Szene aus „The Big Bang Theory“
Abb. 2: Aufzugtür
Abb. 3: Foto eines Läufers

Der Aufbau der Schaltung

Jede IR-Lichtschranke besteht aus zwei Elementen: Einem Sender und einem **Empfänger**. Der Sender, hier eine IR-LED, sendet Licht aus und der Empfänger, hier eine IR-Photodiode "sieht" genau dieses Licht. Wenn die IR-Fotodiode IR-Licht registriert leitet sie Strom. Ist die Sichtlinie unterbrochen, fließt kein Strom.
Ihr werdet zwei Lichtschranken bauen, also braucht ihr folgende Bauteile (neben Steckbrett und Arduino):

- **2x 220 Ω Widerstand**
- **2x 100 kΩ Widerstand**
- **2x IR-Photodiode**
- **2x IR-LED**
- **2 gelbe, 3 blaue und 3 rote lange Steckkabel**

Abb. 4: Verschiedene Widerstände

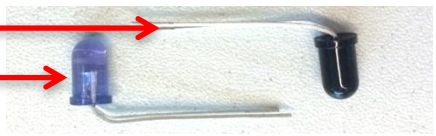


Abb. 5: IR-Photodiode und IR-LED

Station 3 - Geschwindigkeitsmessung

IR-LEDs und IR-Photodioden

IR-LEDs funktionieren im Grunde wie die ganz normalen LEDs aus dem Einstiegsprojekt. Allerdings haben sie einen kleinen Nachteil: Mit bloßem Auge sieht man nicht, ob sie an oder aus sind. Hier hilft euch ein kleiner Trick. Im Gegensatz zu unserem Auge erfassen die meisten digitalen Kameras auch Infrarot-Licht, nutzt also einfach eine Kamera oder ein Smartphone, um zu kontrollieren, ob die IR-LED eingeschaltet ist!

Eine **IR-Photodiode** ist so aufgebaut, dass sie in einer Richtung nur Strom leitet, wenn sie mit dem richtigen Licht beschienen wird. Eine IR-Photodiode leitet also in einer Richtung nur dann Strom, wenn man sie zum Beispiel mit einer IR-LED beleuchtet.

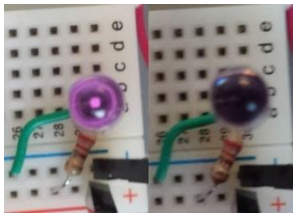


Abb. 6: Schaltungen mit IR-Photodiode und IR-LED

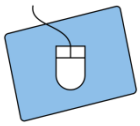
*ACHTUNG: In der anderen Richtung leitet sie **immer** Strom.*

Jetzt könnt ihr euch ans Zusammenbauen begeben! Zuerst wird nur eine Lichtschranke aufgebaut. Auf den Bildern seht ihr, wie die Schaltung am Ende aussehen soll.

1. Zunächst arbeitet ihr nur mit dem unteren Steckbrett, verbindet hierfür die „+“-Leiste mit dem **5V**-Anschluss, die „-“-Leiste mit einem **GND**-Pin.
2. Steckt die **IR-LED** und die **IR-Photodiode** ganz links ins Steckbrett (wie auf dem Schema abgebildet).
3. Für die Lichtschranke muss man die IR-LED nicht ein- oder ausschalten können. Verbindet sie also mit dem 220 Ω **Widerstand** und einem Kabel mit der „+“- und „-“-Leiste.

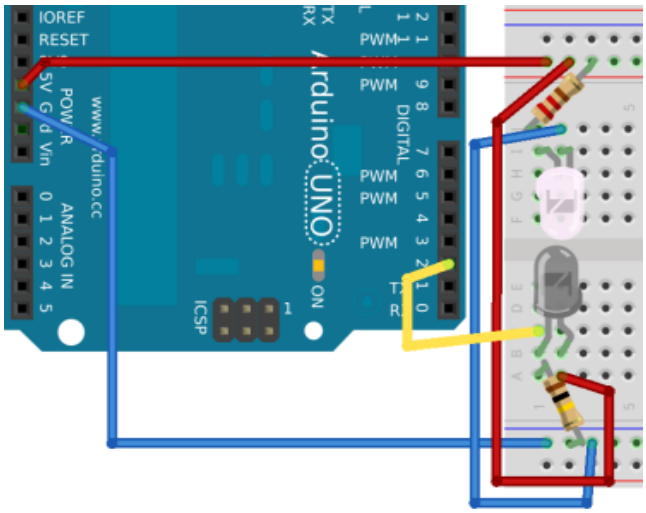
Wichtig: Bei der IR-LED sind die Beinchen vertauscht. Diesmal muss das kurze Beinchen an die „+“-Leiste angeschlossen werden.

4. Testet die IR-LED (siehe Info-Box). Schaut euch die LED mit Hilfe einer **Kamera** an. Zieht einmal das Kabel und steck es wieder ein, um einen Unterschied zu sehen. Leuchtet sie wirklich? Wenn nicht, dann tauscht einmal +- und --Pol, oder testet eure Schaltung mit einer normalen LED, deren Licht ihr erkennen könnt.



5. Sowohl IR-LED als auch IR-Photodiode können nur in einem sehr kleinen Winkel Licht ausstrahlen bzw. aufnehmen. Biegt beide Bauteile so, dass sie sich gegenseitig „angucken“ (wie auf der rechten Abbildung). Am besten liegen sich die Spitzen der Köpfe exakt gegenüber.
6. Die IR-Photodiode wird genauso angeschlossen. Das kurze Beinchen wird mit einem Kabel mit der „+“-Leiste verbunden. Das lange Beinchen wird mit dem **digitalen Pin 2** auf dem Arduino verbunden, der euer Eingang werden soll. Zusätzlich wird dieses Beinchen mit dem 100 kΩ **Widerstand** mit der „-“-Leiste verbunden.

Hinweis: *Wenn ihr die IR-Photodiode falsch anschließt, leitet sie immer Strom!*



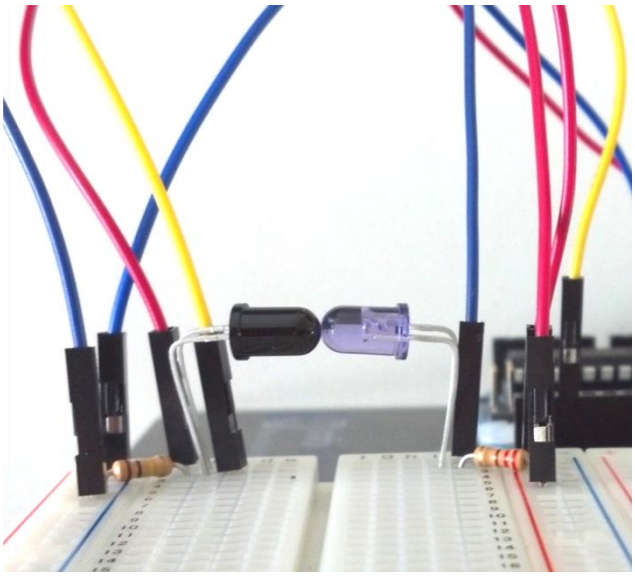


Abb. 7 und 8: Komplette Schlatung (Fritzing und Foto)

Der erste Test

Gut, eure Schaltung steht. Nun geht es ans Programmieren! Ihr habt bisher zwar keine Möglichkeit etwas auszugeben, aber hier haben die Macher vom Arduino zum Glück mitgedacht. Der **Pin 13** ist mit einer **LED** auf dem Arduino intern verbunden, wird Pin 13 eingeschaltet (also auf HIGH gesetzt), leuchtet die LED ohne das Kabel gesteckt werden müssen. Das Foto zeigt euch, wo ihr die eingebaute LED findet.

Eure Aufgabe ist jetzt, die LED genau dann einzuschalten, wenn die Lichtschranke unterbrochen wird!

Eure ToDo's

1. Öffnet einen neuen Arduino-Sketch und speichert ihn unter einem sinnvollen Namen. Denkt dabei an das Grundgerüst!
2. Erstellt nun alle Variablen, die ihr braucht. Gebt ihnen sinnvolle Namen und tragt diese rechts ein:
   a. eine `int`-Variable für die **Arduino-LED** mit dem Wert 13
   b. eine `int`-Variable für die **IR-Photodiode** mit dem Wert 2

<Arduino-Pin>:	_____
<IR-Pin>:	_____

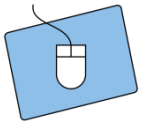
3. Nun zu `setup()`: Als erstes muss der Typ der Pins über den Befehl

   `pinMode(<Pin-Name>, <Pin-Typ>)`

   angegeben werden. Den Pin-Typ für einen Ausgangspin, kennt ihr schon aus dem Einstiegsprojekt. Für einen Eingangspins benutzt man stattdessen **INPUT**.
   a. Benutzt den `pinMode`-Befehl, um den Pin für die Arduino-LED als **Ausgangspin** festzulegen.

b. Benutzt den `pinMode`-Befehl ein zweites Mal, um den IR-Pin als **Eingangspin** festzulegen.

4. Als erstes bringt ihr die Arduino-LED in `loop()` zum Blinken. Das kennt ihr bereits aus dem Einstiegsprojekt. Dazu braucht ihr die Befehle `delay()` und `digitalWrite()`.
5. Testet euer Programm. Blinkt sie? Super! Auf der nächsten Seite lernt ihr, wie ihr die Lichtschranke testen könnt. Wenn sie nicht blinkt, dann könnt ihr ruhig noch einmal im Einstiegsblatt nachlesen, wie man `loop()` dazu ausfüllen muss.



Eingangspin Lesen

Einen Eingangspin kann man mit dem Befehl `digitalRead(<Pin-Name>);` auslesen. Um später mit dem gemessenen Wert weiter arbeiten zu können, könnt ihr diesen in einer Variablen speichern.
Dies funktioniert so:

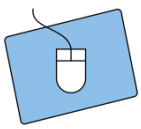
```
<Variablenname> = digitalRead(<Pin-Name>);
```

Jetzt wird `loop()` so erweitert, dass die Lichtschranke getestet werden kann. Ihr wollt die LED natürlich nicht immer einschalten, sondern nur wenn die Lichtschranke unterbrochen ist. Dafür braucht ihr eine `if-else`-Anweisung, die für euch ja nichts Neues ist. Werft ruhig nochmal einen Blick auf das Einstiegsblatt, wenn ihr euch nicht mehr ganz genau erinnert.

1. Ihr könnt den `digitalRead`-Befehl direkt in der `if`-Bedingung benutzen, um zu überprüfen, ob Strom ankommt (**HIGH**) oder ob keiner ankommt (**LOW**):

```
if (digitalRead(<Pin-Name>) == <HIGH/LOW>) {…}
```

2. Wenn also der digitale Eingang ein **LOW** ausspuckt, wird die LED eingeschaltet. Mit einem „else“ in der nächsten Zeile könnt ihr festlegen, was sonst passiert. Dazu müsst ihr nun eure Befehle, die die Arduino-LED blinken lassen an die richtigen Stellen verschieben.
3. Entfernt alle `delays`, die noch übrig sind. Diese benötigt ihr jetzt nicht mehr.
4. Es ist an der Zeit, den Sketch zu testen! Benutzt das offizielle Lichtschrankenunterbrechungsautomobil (oder einen beliebigen schmalen Gegenstand), um die Lichtschranke zu unterbrechen. Überprüft dabei, ob sich die LED auf dem Arduino richtig verhält.



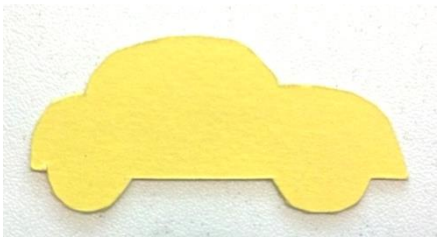


Abb. 7: Schablone eines Autos

5. Wenn es funktioniert, könnt ihr diese Aufgabe überspringen. Ansonsten ist es ganz normal, wenn es nicht auf Anhieb funktioniert. Das passiert auch den besten Informatikern. Das einzige was zählt ist, dass man in aller Ruhe nach dem Fehler sucht und ihn am Ende behebt. Überprüft einmal, ob eure if-Bedingung wirklich stimmt, ob beide Pins korrekt in `setup()` als `INPUT` oder `OUTPUT` definiert wurden und ob ihr `digitalWrite()` und `digitalRead()` richtig benutzt. Testet auch eure LED noch einmal mit einer Kamera. Vielleicht ist auch eure IR-Photodiode falsch herum angeschlossen, sodass sie immer Strom leitet.

6. Speichert euren Sketch, denn ihr werdet ihn in den folgenden Aufgaben noch benötigen.

Die zweite Lichtschranke

Der Aufbau erfolgt analog zur ersten Lichtschranke, allerdings benutzt ihr jetzt die Zeilen 29 und 30 des gleichen Steckbrettes. Als Eingang für diese IR-Photodiode bietet sich der Pin Nummer 3 des Arduino an. Eure Schaltung sollte jetzt in etwa so aussehen:

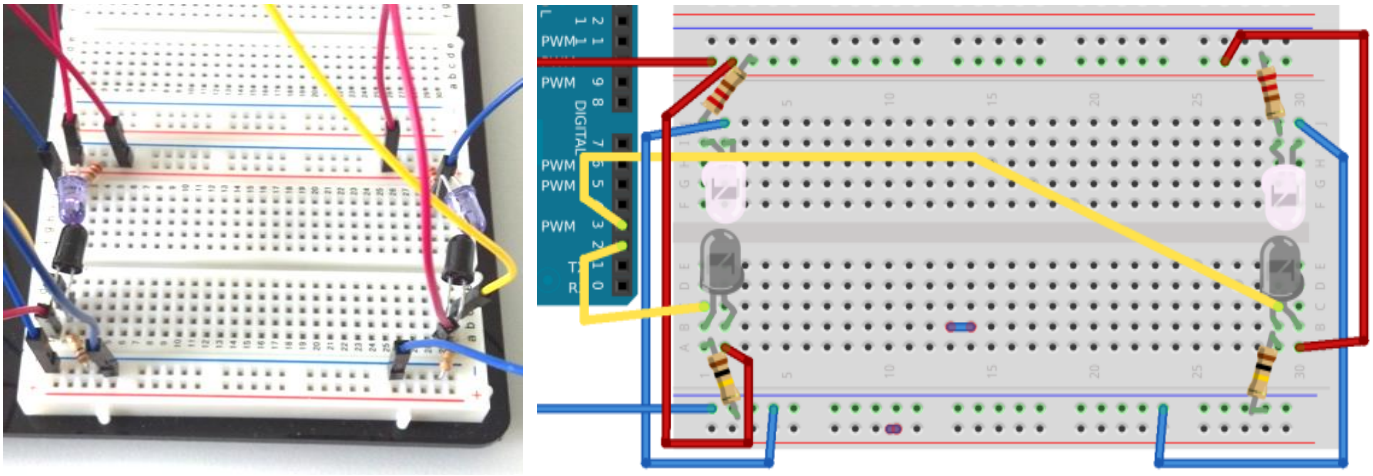
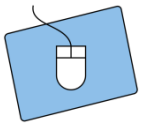


Abb. 8 und 9: Komplette Schaltung (inkl. Fritzing-Bild)

Eure ToDo's:

- Wie könnt ihr jetzt schnell und einfach die Funktion testen? Nehmt den Sketch von vorhin, ändert aber oben den Eingangspin auf die 3. Leuchtet die LED jetzt sobald die zweite Lichtschranke durchbrochen wurde?



Funktioniert alles? Gut, jetzt habt ihr schon alles beisammen, um Geschwindigkeiten zu messen! Allerdings müssen dafür noch ein paar Vorbereitungen getroffen werden. Wenn die erste Lichtschranke durchbrochen wurde, soll eine Messung gestartet werden und wenn die zweite Lichtschranke passiert wurde, soll die Messung beendet werden.

Station 3 - Geschwindigkeitsmessung

Die ToDo's erklären euch, wie ihr euren Sketch um die nötigen `if`-Anweisungen erweitern könnt.

Eure ToDo's:

1. Ändert den Wert des IR-Pin wieder auf 2. Dann speichert den Sketch unter einem neuen Namen, damit ihr ihn jetzt verändern könnt, ohne euer bisheriges Ergebnis zu löschen.

2. Als erstes müsst ihr eine **Variable** für die zweite **IR-Photodiode** erstellen und den Pin-Typ in `setup()` festlegen. Tragt beides rechts ein.

```
<IR-Pin-2>:
```

3. Ihr müsst in eurem Sketch speichern, ob die Messung schon begonnen hat, oder nicht. Dafür könnt ihr eine weitere **int**-Variable benutzen. Eine „**0**" soll dabei bedeuten, dass die Messung noch nicht gestartet wurde. Umgekehrt soll eine „**1**" bedeuten, dass die Messung bereits läuft. Im Bereich Einstellungen muss diese Variable als mit dem Wert 0 angelegt werden.

```
<          Messungsvariable          >:
```

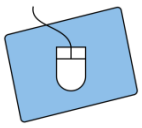
4. Nun benötigt ihr eine `if`-Anweisung, die erkennt, dass die Messung beginnen soll. Passt dazu eure `if`-Anweisung so an, dass sie eure Messungs-Variable auf **1** setzt und über den **Seriellen Monitor** einen Text ausgibt, der besagt, dass die Messung begonnen hat. Dies soll genau dann passieren, wenn die erste Lichtschranke durchbrochen wurde und die Messung nicht bereits läuft. Schaut euch dazu den rot markierten Teil des Diagramms unten an. Ihr braucht kein `else` in dieser Aufgabe.

if-Anweisungen mit mehreren Bedingungen

Mit „`if`" könnt ihr auch mehr als eine Bedingung gleichzeitig abfragen. Verknüpft diese Bedingungen dazu mit „&&". Dabei müsst ihr aber um die einzelnen Bedingungen Klammern schreiben. So wird die Anweisung nur ausgeführt, wenn beide Bedingungen zutreffen.

```
if ( ( <BedingungA>) && ( <BedingungB> ) )
```

5. Testet euer Programm. Startet den Seriellen Monitor und unterbrecht die Lichtschranken. Dabei sollte euer Text nur genau einmal angezeigt werden, denn die Messung ist ja dann gestartet. Wenn der Text immer wieder angezeigt wird, müsst ihr nochmal über die Bedingungen nachdenken.



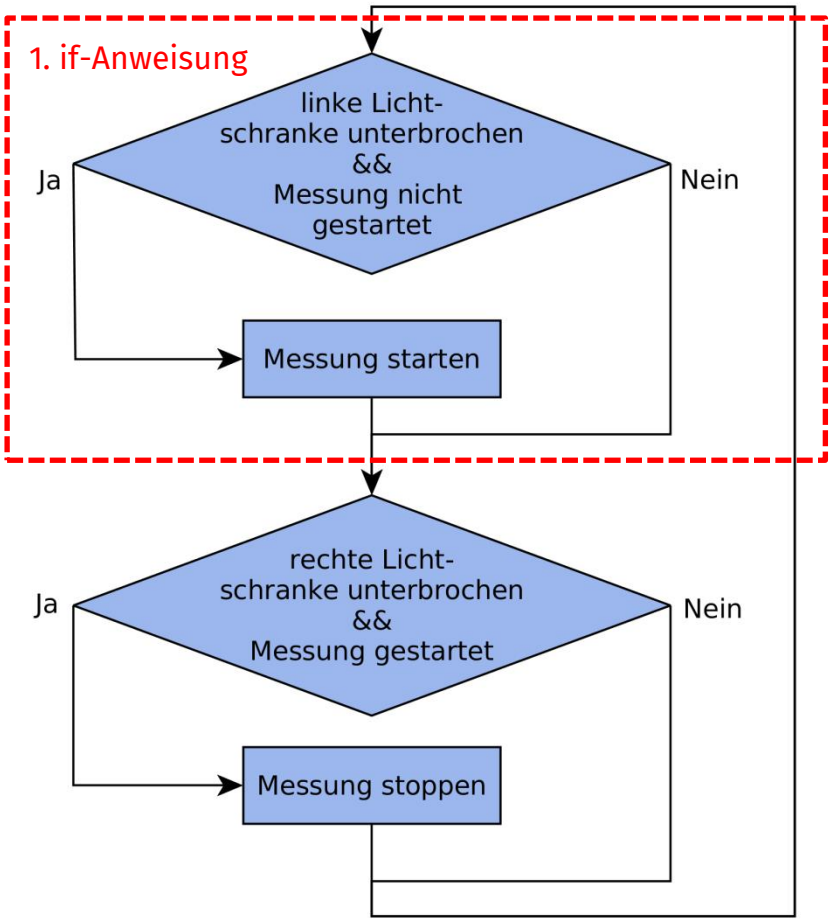
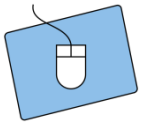


Abb. 10: Programmablaufplan

6. Erstellt nun eine zweite `if`-Anweisung direkt darunter (ohne `else`). Diese soll genau anders herum funktionieren. Sie soll die „Messung-gestartet“-Variable auf **0** setzen und über den Seriellen Monitor ausgeben, dass die Messung beendet wurde (später soll hier auch die Geschwindigkeit ausgegeben werden). Schaut euch dazu wieder das Diagramm unten an.

7. Testet euren Sketch wieder. Nun sollten die Texte „Messung beginnt“ und „Messung beendet“ abwechselnd auf dem Monitor angezeigt werden, wenn ihr abwechselnd die Lichtschranken unterbrecht.



[**Hinweis:** *Passt auf, dass ihr dabei eure IR-LEDs und IR-Photodioden nicht versehentlich verschiebt oder dreht. Die Kabel können bei der Messung stören. In dem Fall, klebt sie einfach mit Tesafilm oder ähnlichem zur Seite.*]

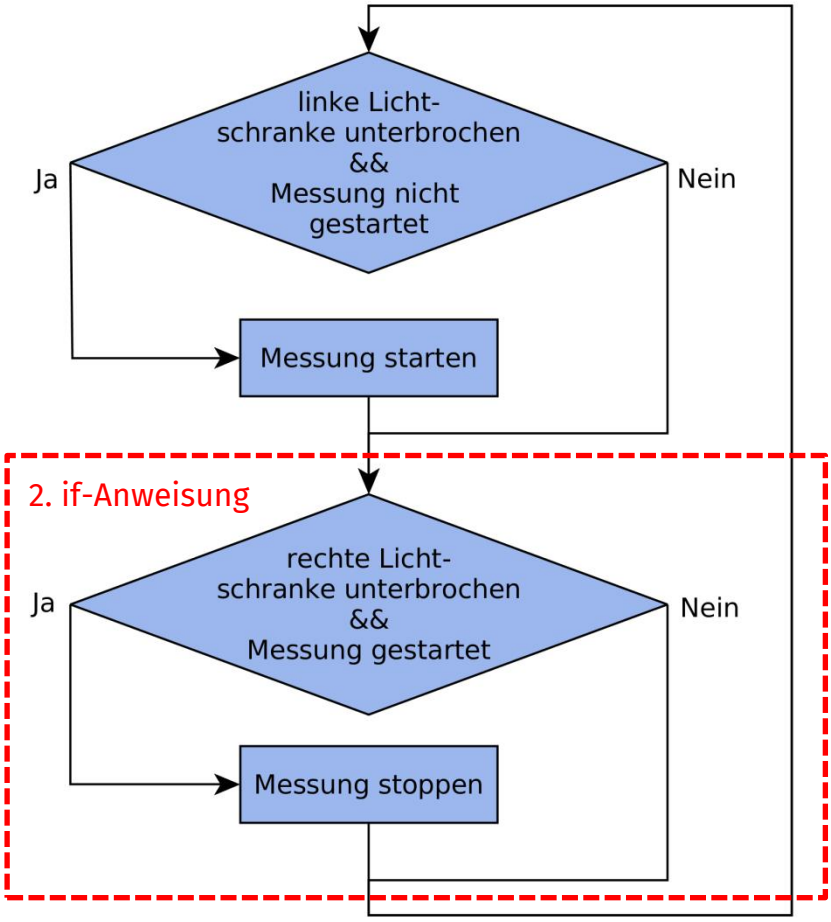


Abb. 11: Programmablaufplan

Jetzt habt ihr alles vorbereitet, um die Zeitmessung einzubauen.

Zeitmessung

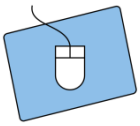
Die Funktion `millis()` liefert euch die Anzahl der Millisekunden, die, seit ihr euer Programm auf den Arduino geladen habt, vergangen sind. Diese Zeit soll gespeichert werden. Eine `int`-Variable kann jedoch nur Zahlen bis zu einer bestimmten Größe speichern. Die Anzahl der Millisekunden kann jedoch sehr groß werden. Darum benötigt ihr eine neue Art von Variablen: `unsigned long`. Das ist ebenfalls eine ganze Zahl, allerdings können größere Zahlen gespeichert werden.

Eure ToDo's

1. Erstellt im Bereich Einstellungen zwei neue Variablen vom Typ `unsigned long`. Diese sollen den Start- und Endzeitpunkt der Messung speichern. Gebt ihnen sinnvolle Namen.

```
unsigned long <Variablen-Name>;
```

2. Überlegt euch, wann die Zeit gemessen werden muss. Speichert mit Hilfe des Befehls `millis()` an den richtigen Stellen in eurem Programm die Zeiten in eure beiden Variablen.
3. Erweitert die if-Anweisungen in eurem Programm so, dass sie über den Seriellen Monitor die Start- und die Endzeit ausgeben.
   Hinweis: hier ohne „“.



4. Testet euer Programm. Nicht wundern, die Zahlen werden sehr schnell sehr groß. Aber das ist bei der Angabe in Millisekunden ja auch nicht verwunderlich.
5. Klappt die Zeitausgabe? Super! Jetzt müsst ihr nur noch berechnen, wie lange die Messung gedauert hat. Dazu braucht ihr eine dritte `unsigned long`-Variable, die die Dauer der Messung speichern soll.

Rechnungen

In eurem Programm könnt ihr ganz einfach ausführen, indem ihr zwei Variablen oder Zahlen mit **+**, **-**, ***** oder **/** verbindet und das Ergebnis dann mit = in eine dritte Variable speichert. Eine einfache Subtraktion sieht zum Beispiel so aus:

```
<Ergebnisvariable> = <Zahl/Variable> - <Zahl/Variable>;
```

6. Benutzt eure Variablen und eine einfache Subtraktion, um die Messungsdauer zu berechnen und in eurer Messungsdauer-Variablen zu speichern. Füllt dazu diese Box aus:

```
<Dauer>     = <Endzeitpunkt> - <Startzeitpunkt> ;
__________________ = ___________________ - ___________________ ;
```

7. Gebt das Ergebnis beim Messungsende über den Seriellen Monitor aus.

Super! Jetzt wird euch auf dem Seriellen Monitor angezeigt wie lange das Auto gebraucht hat, um die Strecke zwischen den beiden Lichtschranken zurückzulegen.
Eine Zeit, das ist doch noch keine Geschwindigkeit? Richtig, der Rest ist Physik! Wenn ihr diese Herausforderung auch noch meistern wollt, fragt die Betreuer nach dem **Bonus-Blatt**.

Quellenverzeichnis:

Abb. 1 – *Quelle: The Big Bang Theory (CBS)*
Abb. 2 – *Quelle: Bild von Tate Johnson*
Abb. 3 – *Quelle: Bild von Mark Sadowksi*
Abb. 4, 9 – *Quelle: Screenshots der Fritzing-Software (http://fritzing.org)*
Abb. 5, 6, 7, 8, 10, 11 – *Quelle: InfoSphere*

Alle weiteren Grafiken/Icons – Quelle: InfoSphere