Station 3 – Geschwindigkeitsmessung

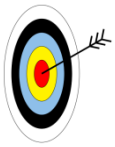

[1]

Unsichtbares Licht und Geschwindigkeit

Lichtschranken begegnen euch überall im Alltag, z. B. in Aufzügen oder bei automatischen Türklingeln in Geschäften. Aber auch wenn ihr ihnen ständig über den Weg lauft, merkt ihr dies häufig nicht, weil sie mit unsichtbarem Licht arbeiten: dem **Infrarot-Licht** (kurz: **IR**-Licht).

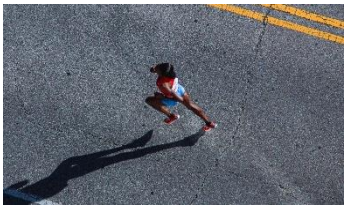

[2]

Lichtschranken funktionieren im Prinzip wie Schalter. Die Einsatzzwecke sind dabei sehr vielfältig.



In diesem Arbeitsblatt werdet ihr…
- zwei IR-Lichtschranken verbauen.
- lernen, wie man mit ihnen Geschwindigkeit misst.

Anwendung findet diese Konstruktion z. B. bei der Zeitmessung im Sport.


[3]

Aufbau der Schaltung

Jede IR-Lichtschranke besteht aus zwei Elementen: einem **Sender** und einem **Empfänger**. Der Sender, hier eine IR-LED, sendet Licht aus. Der Empfänger, hier eine IR-Fotodiode „sieht“ genau dieses Licht. Wenn die IR-Fotodiode IR-Licht registriert, leitet sie Strom. Ist die Sichtlinie jedoch unterbrochen, fließt kein Strom.

Ihr werdet zwei Lichtschranken bauen, also braucht ihr folgende Bauteile (neben Steckbrett und Arduino):

- 2x 220 Ω Widerstand (Abbildung 4)
- 2x 100 kΩ Widerstand (Abbildung 5)
- 2x IR-Fotodiode (Abbildung 6, schwarz)
- 2x IR-LED (Abbildung 6, blau)
- 2 gelbe, 3 blaue, 3 rote lange Steckkabel


[4]


[5]


[6]

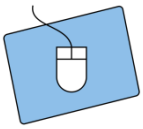
Station 3 – Geschwindigkeitsmessung

IR-LEDs und ihre Fotodioden

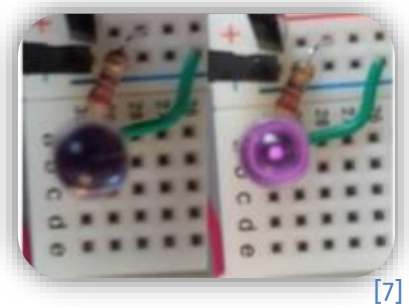
IR LEDs funktionieren im Grunde wie die ganz normalen LEDs aus dem Einstiegsprojekt. Allerdings haben sie einen kleinen Nachteil: **Mit bloßem Auge sieht man nicht, ob sie an oder aus sind.** Hier hilft euch ein kleiner Trick: Um zu überprüfen, ob eure Schaltung korrekt und der Stromkreis geschlossen ist, könnt ihr eure IR-LED und Fotodiode durch normale LEDs ersetzen. Wenn diese leuchten, dann funktioniert eure Schaltung soweit. Eine IR-Fotodiode ist so aufgebaut, dass sie in eine Richtung nur Strom leitet, wenn sie mit dem richtigen Licht beschienen wird. Eine IR-Fotodiode leitet also in eine Richtung nur dann Strom, wenn man sie z. B. mit einer IR-LED beleuchtet.

Die IR-Fotodiode leitet in eine Richtung nur dann, wenn sie z. B. von einer IR-LED beschienen wird. In die andere Richtung leitet sie immer Strom.

Jetzt könnt ihr mit dem Zusammenbau starten. Zuerst wird nur eine Lichtschranke aufgebaut. Auf der nächsten Seite seht ihr, wie die Schaltung schlussendlich aussehen soll.



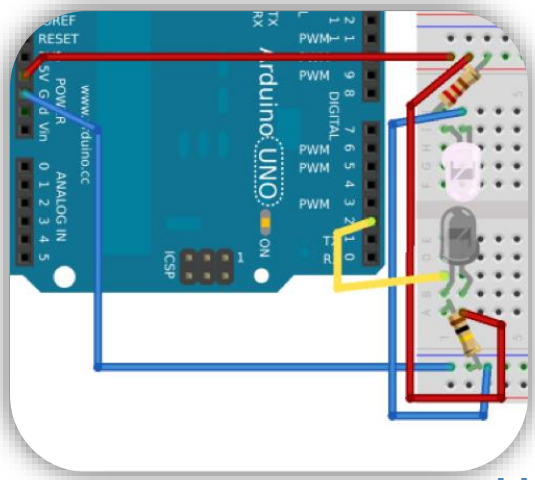
1. Zunächst arbeitet ihr nur mit dem unteren Teil des Steckbrettes. Verbindet hierfür die Plus-Leiste mit dem **5V**-Anschluss und die Minus–Leiste mit einem **GND**-Pin.
2. Steckt die **IR-LED** und die **IR-Fotodiode** ganz links ins Steckbrett.
3. Für die Lichtschranke muss man die IR-LED nicht ein- oder ausschalten können. Verbindet sie also mit dem **220 Ω Widerstand** mit der Plus-Leiste und mit einem Kabel mit der Minus-Leiste. Eine Orientierung gibt euch Abbildung 7:



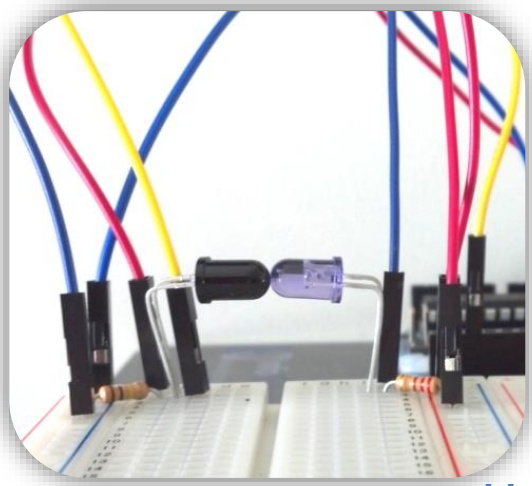
[7]

4. Testet die IR-LED (wie das geht wurde euch am Anfang der Seite erklärt).
5. Sowohl die IR-LED als auch die IR-Fotodiode können nur in einem sehr kleinen Winkel Licht ausstrahlen bzw. aufnehmen. Biegt die Bauteile so, dass sie sich gegenseitig „anschauen“. Am besten liegen sich die Spitzen der Köpfe exakt gegenüber.
6. Die IR-Fotodiode wird genauso angeschlossen. Das kurze Beinchen wird mit einem Kabel mit der Plus-Leiste verbunden. Das lange Beinchen wird mit dem **digitalen Pin 2** des Arduinos verbunden, der euer Eingang werden soll. Zusätzlich wird das lange Beinchen auch mit einem **100 Ω Widerstand** mit der Minus-Leiste verbunden.
   Hinweis: Wenn ihr die IR-Fotodiode falsch anschließt, leitet sie immer Strom!

Station 3 – Geschwindigkeitsmessung



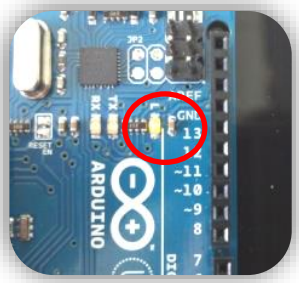
[8]



[9]

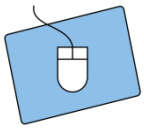
Der erste Test

Gut, eure Schaltung steht. Nun geht es weiter mit der Programmierung! Ihr habt bisher zwar keine Möglichkeit, etwas auszugeben, aber hier haben die Macher von Arduino zum Glück mitgedacht. **Pin 13** ist mit einer **LED** auf dem Arduino intern verbunden (siehe Abbildung 10). Wird Pin 13 eingeschaltet (also auf high gesetzt), leuchtet die LED, ohne dass Kabel gesteckt werden müssen.



[10]

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



1. Öffnet einen neuen Sketch und speichert ihn unter einem sinnvollen Namen.
2. Erstellt nun alle Variablen, die ihr braucht. Gebt auch ihnen sinnvolle Namen:
   a. eine `int`-Variable für die **Arduino-LED** mit dem Wert 13
   b. eine `int`-Variable für die **IR-Fotodiode** mit dem Wert 2
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 **Eingangspin** benutzt man stattdessen `INPUT`.
   a. Benutzt den **`pinMode`**-Befehl, um den Pin für die Arduino-LED als **Ausgangspin** festzulegen.
   b. Benutzt den **`pinMode`**-Befehl, um den Pin für 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 die LED? Super! Wenn sie nicht blinkt, dann könnt ihr ruhig noch einmal im Einstiegsblatt nachlesen, wie man **`loop()`** dazu ausfüllen muss.

Station 3 – Geschwindigkeitsmessung

Eingangspin lesen

Einen Eingangspin kann man mit dem Befehl `digitalRead(pin-name);` lesen. Um später mit dem gemessenen Wert weiterarbeiten 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 noch einmal einen Blick auf das Einstiegsblatt, wenn ihr euch nicht mehr 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) == Strom)
   { … }
   ```
2. Wenn der digitale Eingang **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 richtige Stelle 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.

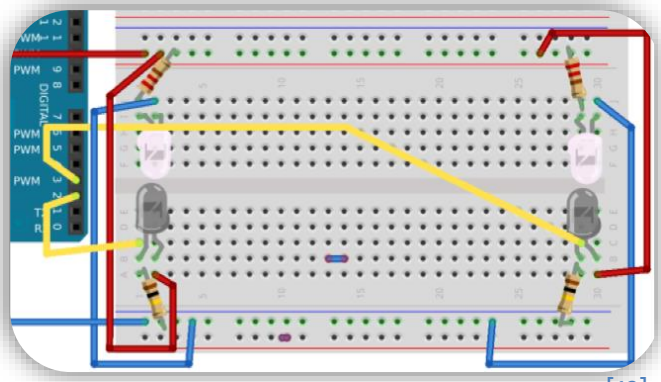
[11]

5. Wenn es funktioniert, könnt ihr diesen Punkt überspringen.
   Ansonsten ist es ganz normal, wenn es nicht auf Anhieb funktioniert. Das passiert auch den besten Programmierenden. 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 sind und ob ihr **`digitalWrite()`** und **`digitalRead()`** richtig benutzt. Testet auch eure LED noch einmal. Vielleicht ist auch eure Fotodiode falsch herum angeschlossen, sodass sie immer Strom leitet.
6. Speichert euren Sketch, denn ihr werdet ihn in den folgenden Aufgaben noch brauchen.

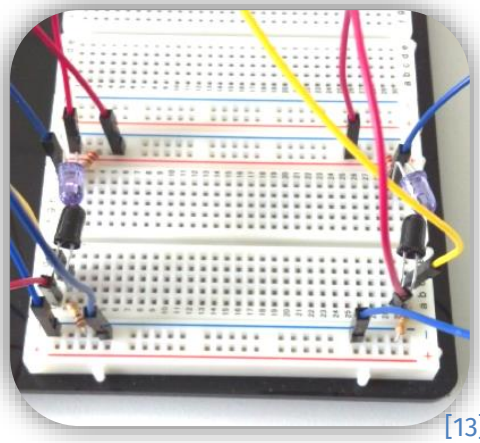
Station 3 – Geschwindigkeitsmessung

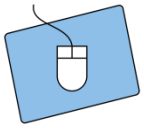
Die zweite Lichtschranke

Der Aufbau erfolgt analog zur ersten Lichtschranke. Allerdings benutzt ihr jetzt die Zeilen 29 und 30 des gleichen Steckbrettes. Als Eingang für diese IR-Fotodiode bietet sich Pin 3 des Arduinos an. Eure Schaltung sollte jetzt in etwas so aussehen:



[12]


[13]

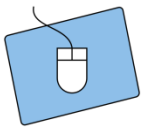


Wie könnt ihr jetzt schnell und einfach die Funktion testen?

1. Nehmt den Sketch von vorhin.
2. Ändert den Eingangspin auf 3.

Leuchtet die LED jetzt, sobald die zweite Lichtschranke durchbrochen wurde?

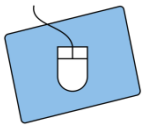
Funktioniert alles? Gut, jetzt habt ihr schon alles beisammen, um Geschwindigkeiten zu messen! Allerdings müssen dafür noch ein paar Vorbereitungen getroffen werden. Wenn die erste Lichtschranke durchbrochen wurde, soll eine Messung gestartet werden und wenn die zweite Lichtschranke passiert wurde, soll die Messung beendet werden. Die folgenden Punkte erklären euch, wie ihr euren Sketch um die nötigen **`if`-Anweisungen** erweitern müsst.



1. Ändert den Wert des IR-Pins wieder auf **2**. Speichert dann 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-Fotodiode** erstellen und den Pin-Typ in **`setup()`** festlegen.
3. Ihr müsst außerdem 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 bedeutet eine `1`, dass die Messung bereits läuft. Im **Bereich Einstellungen** muss diese Variable also mit dem Wert `0` angelegt werden.

Weiter geht's auf der nächsten Seite…

Station 3 – Geschwindigkeitsmessung

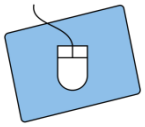


4. Nun benötigt ihr eine **`if`-Anweisung**, die erkennt, dass die Messung beginnen soll. Passt dazu eure `if`-Anweisung so an, dass sie eure Messungs-Variable auf `1` setzt und über den **Serial Monitor** einen Text ausgibt, der besagt, dass die Messung begonnen hat. Dies soll genau dann passieren, wenn die erste Lichtschranke durchbrochen wurde und die Messung nicht bereits läuft. Schaut euch dazu das Diagramm unten an. **Hinweis:** 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.

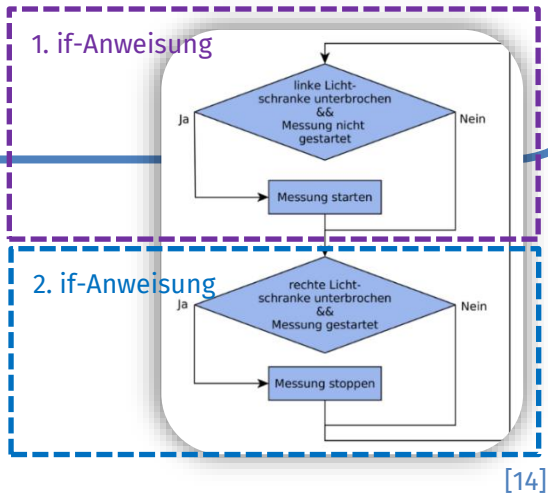
```
if ( (bedingungA) && (bedingungB) )
```



5. Testet euer Programm. Startet den Serial 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 Bedingung nachdenken.
6. Erstellt nun eine **zweite `if`-Anweisung** direkt darunter (ohne `else`). Diese soll genau andersherum funktionieren. Sie soll die **Messung-gestartet**-Variable auf `0` setzen und über den Serial Monitor ausgeben, dass die Messung beendet wurde (später soll hier auch die Geschwindigkeit ausgegeben werden). Schaut euch dazu 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-Fotodioden nicht versehentlich verschiebt oder dreht. Die Kabel können bei der Messung stören. Klebt sie einfach mit Tesafilm oder ähnlichem zur Seite.

Jetzt habt ihr alles vorbereitet, um die Zeitmessung einzubauen.



[14]

Station 3 – Geschwindigkeitsmessung

Zeitmessung

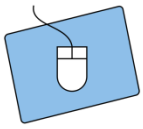
Die Funktion **`millis()`** liefert euch die Anzahl der Millisekunden, die, seit ihr euer Programm auf den Arduino geladen habt, vergangen sind. Diese Zeit soll gespeichert werden. Eine **`int`**-Variable kann jedoch nur Zahlen bis zu einer bestimmten Größe speichern. Da die Anzahl der Millisekunden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

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 sinnvoller_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 euren beiden Variablen.
3. Erweitert die `if`-Anweisungen in eurem Programm so, dass sie über den Serial Monitor die Start- und die Endzeit ausgeben.
4. Testet euer Programm. Nicht wundern: Die Zahlen werden sehr schnell sehr groß. Aber das ist bei der Angabe der 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 Rechnungen ganz einfach ausführen, indem ihr zwei Variablen oder Zahlen mit **+**, **–**, ***** oder **/** verbindet und das Ergebnis dann mit **=** in einer dritten Variablen speichert.
Eine einfache Subtraktion sieht z. B. so aus:

```
ergebnisvariable = zahl1/variable1 - zahl2/variable2;
```



6. Benutzt eure Variablen und eine einfache Subtraktion, um die Messungsdauer zu berechnen und speichert sie in eurer Messungsdauer-Variablen ab.
7. Gebt das Ergebnis beim Messungsende über den Serial Monitor aus.

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

Quellenverzeichnis:

Abb. 1 – *Quelle: pxhere.com (https://pxhere.com/de/photo/981592), CC0 1.0 Universal (CC0 1.) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 06.07.2022*

Abb. 2 – *Quelle: pxhere.com (https://pxhere.com/de/photo/927145), CC0 1.0 Universal (CC0 1.) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 06.07.2022*
Abb. 3 – *Quelle: pxhere.com (https://pxhere.com/de/photo/53576), CC0 1.0 Universal (CC0 1.) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 06.07.2022*
Abb. 4, 5, 8, 12 – *Quelle: Screenshot der Fritzing-Software (http://fritzing.org), CC BY-SA 3.0 Attribution-ShareAlike 3.0 Unported (https://creativecommons.org/licenses/by-sa/3.0/), abgerufen am: 14.06.2022.*
Abb. 6, 7, 9-11, 13, 14 – *Quelle: InfoSphere, CC BY-SA 4.0 Attribution-ShareAlike 4.0 International (https://creativecommons.org/licenses/by-sa/4.0/)*

– *Quelle: InfoSphere, CC BY-SA 4.0 Attribution-ShareAlike 4.0 International (https://creativecommons.org/licenses/by-sa/4.0/)*