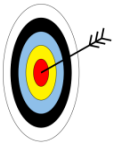
Station 2 – Einparkhilfe – Bonus

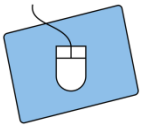
Der Abstand macht den Ton

Eure Einparkhilfe misst jetzt Abstandswerte mit dem Sensor und piepst sogar schon in verschiedenen Abstufungen. Ein großer Nachteil dabei ist allerdings, dass ihr zuerst Werte gemessen habt, dann von Hand Distanz-Bereiche eingeteilt habt und für diese Bereiche auch von Hand die Pieps-Geschwindigkeiten vorgegeben habt. Für den Alltag ist das leider sehr unpraktisch. Mit einem Fahrzeug kann man schließlich auch nicht immer zunächst vorsichtig den Abstand messen und die Einparkhilfe dann erst programmieren. Die Lösung: Der Sensor misst die Veränderung des Abstands ständig und berechnet sofort automatisch den Warnton.

Kurz gesagt: Ihr wollt, dass eure Einparkhilfe **direkt aus den gemessenen Sensorwerten** die Geschwindigkeit des Signaltons berechnet. Leider liefert der Sensor nicht direkt Zentimeterangaben und aus den Werten lassen sich diese auch nicht einfach berechnen.



In dieser zweiten Stufe der Optimierung lernt ihr,...
- wie ihr vorgegebene Funktionen in euren Sketch einbindet und nutzt.
- den berechneten Zentimeterwert direkt in eine Pieps-Geschwindigkeit umzuwandeln.

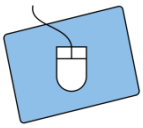


1. Speichert euren Sketch ein letztes Mal unter einem anderen, sinnvollen Namen.
2. Um die Funktion zur Umrechnung der Sensor-Werte benutzen zu können, müsst ihr ganz am Anfang des Sketches, als ersten Befehl, eine sogenannte **Library/Bibliothek** einfügen. Nutzt dazu den Befehl: `#include <Abstand.h>`
   Wichtig: Diesen Befehl schreibt ihr mit **spitzen Klammern** aber **ohne** Semikolon!

Bibliotheken

Das, was ihr eben in euren Sketch geschrieben habt, ist ein sogenannter `include`-Befehl („füge ein“-Befehl). Er sagt dem Sketch, dass er die **Bibliothek mit dem Namen** `Abstand` einbinden soll; darin sind die Funktionen enthalten, um Abstandswerte des Sensors in Zentimeterwerte umzurechnen, z. B. die `berechneZentimeter()`-Funktion, die ihr später benutzen werdet. Die Raute (#) vor dem Befehl gibt an, dass es sich um einen besonderen Befehl handelt, der vor allem anderen unbedingt zuerst ausgeführt werden muss. Die Endung `.h` stammt aus der **Programmiersprache C**, die in Arduino zur Erstellung solcher Bibliotheken genutzt wird.

Station 2 – Einparkhilfe – Bonus



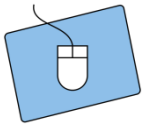
3. Alles, was zur Berechnung nötig ist, habt ihr also jetzt eingebunden. Damit ihr aber mit dieser Bibliothek arbeiten könnt, müsst ihr ein **Exemplar dieser Bibliothek** erzeugen. Stellt euch das so vor, als ob ihr eine Variable anlegt. Der Typ (z. B. `int` bei Zahlenwerten) ist hier der **Name der Bibliothek** (also `Abstand`), gefolgt von einem beliebigen Namen.
4. Fügt im Bereich **Einstellungen** jetzt noch eine `int`–Variable hinzu, in der ihr später die berechneten Zentimeterwerte speichern könnt.
5. Die eigentliche Arbeit passiert wieder in **`loop()`:**
   a. Die Zeile zum **Einlesen der Sensorwerte** befindet sich bereits in eurem Sketch. Diese großen Werte werdet ihr jetzt in Werte zwischen 10 und 80 cm umrechnen. Dazu müsst ihr die **Funktion** `berechneZentimeter(sensorWert)` aus der Bibliothek benutzen. Jetzt benötigt ihr auch noch die **Instanz, die ihr von der Bibliothek** erstellt habt. Damit euer Sketch nämlich weiß, woher die Methode kommt, muss man ihm das durch die Variable angeben.
   b. Der Sensorwert ist ja bereits in einer Variablen gespeichert. Hier reicht es, einfach den Namen der Variablen in die **Klammern** zu schreiben.
   c. Um die berechneten Zentimeterwerte jetzt auch in einer Variablen zu speichern, müsst ihr nur noch den Variablennamen aus Schritt 4 mit einem **Gleichzeichen** mit dem Aufruf der Methode aus 5.a. verbinden – so, als ob ihr einer `int`-Variablen einen einfachen Zahlenwert zuweist.
   Insgesamt sieht das also so aus:
   ```
   zentimeterWert = libraryInstanz.berechneZentimeter(sensorWert);
   ```
6. Als letzten Schritt könnt ihr euch jetzt die neuen Zentimeterwerte auf der Konsole ausgeben lassen. Übertragt dazu euer Programm und platziert euer Hindernis entsprechend. Vergleicht die Werte doch einmal mit euren selbst gemessenen Zentimeterwerten aus der Tabelle.

Abstand	10 cm	20 cm	30 cm	40 cm	50 cm	60 cm	70 cm	80 cm
Sensorwert								

Nun könnt ihr endlich die Zentimeterangaben nutzen und daraus **direkt** die **Verzögerung** zwischen den Piep-Tönen bestimmen! Ihr benötigt keine **`if`-Anweisung** mehr, da ihr keine Bereiche mehr per Hand einteilt und abfragen müsst.

Zum Endspurt geht's auf der nächsten Seite…

Station 2 – Einparkhilfe – Bonus



1. Setzt einmal den **Piezo** auf **high** (gefolgt von einem `delay`) und einmal auf **low** (ebenfalls gefolgt von einem `delay`).
2. Überlegt, woher ihr die Werte für die `delay`-Befehle nehmt.
   Erinnerung: Genau diese Verzögerung durch `delay` soll über die berechneten Zentimeterwerte bestimmt werden.
   Hinweis: Was bedeutet das für den Fall, dass ihr jetzt Zentimeterwerte zwischen 10 und 80 cm bekommt? Passt die Werte ggf. sinnvoll an und testet euer Programm.

Herzlichen Glückwunsch! Ihr habt es geschafft, eure Einparkhilfe sogar noch zu optimieren! Gute Fahrt! 😉

Quellenverzeichnis:

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