Station 2 – Einparkhilfe

Der Abstand macht den Ton

Eure Einparkhilfe misst jetzt Abstands-Werte 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 ja auch nicht immer erst vorsichtig den Abstand messen und die Einparkhilfe des Autos 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 Werten** des Infrarot-Sensors die Geschwindigkeit des Signaltons berechnet. Leider liefert der Sensor (wie auf Seite 4 beschrieben) nicht direkt Zentimeter-Angaben 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 und
- ... den berechneten Zentimeter-Wert direkt in eine Pieps-Geschwindigkeit umwandelt.

Eure ToDo's

1. Speichert euren Arduino-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** (auf Deutsch: **Bibliothek**) einfügen. Nutzt dazu den Befehl: `#include <Abstand.h>`

   [**Wichtig:** *Dieses Mal MIT spitzen Klammern aber OHNE Semikolon!!!*]

Bibliotheken

Das, was ihr eben eurem Sketch hinzugefügt habt, ist ein sogenannter **include**[*„füge ein“*]-**Befehl**. Er sagt dem Sketch, dass er die **Bibliothek mit dem Namen** `Abstand` einbinden soll – darin sind die Funktionen enthalten, um Abstands-Werte des Sensors in Zentimeter-Werte 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 von der **Programmiersprache C**, die zur Erstellung solcher Libraries genutzt wird.

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. im Bereich Einstellungen jetzt noch eine `int`-**Variable** hinzu, in die ihr später die berechneten Zentimeter-Werte speichern könnt.
5. Die eigentliche Arbeit passiert wieder in `loop()`:
   a. Die **Zeile zum Einlesen der Sensor-Werte** befindet sich bereits in eurem Sketch. Diese großen Werte wollen wir jetzt in Werte zwischen 10-80 cm umrechnen.
   b. Dazu müsst ihr die **Funktion** `berechneZentimeter(<Sensor-Wert>)` aus der Bibliothek benutzen. Jetzt benötigt ihr auch die **Instanz, die ihr von der Library erstellt habt**! Damit euer Sketch nämlich weiß, woher die Methode kommt, muss man ihm das durch die Variable angeben.
   c. Der Sensor-Wert ist ja bereits in einer Variablen gespeichert. Hier reicht es, einfach den Namen der Variablen **in die Klammern** zu schreiben.
   d. Um die berechneten Zentimeter-Werte jetzt auch in einer Variable zu speichern, müsst ihr nur noch den Variablennamen aus Schritt 4 mit einem **Gleichzeichen** mit dem Aufruf der Methode aus 5.b. verbinden – so als ob ihr einer int-Variablen einen einfachen Zahlenwert zuweist. **Insgesamt sieht das schematisch so aus:**

```
<Variablenname> = <NamederInstanz>.berechneZentimeter(<Sensor-Wert>);
```

6. Als letzten Schritt könnt ihr euch jetzt einmal die neuen Zentimeter-Werte **auf der Konsole ausgeben** lassen! Übertragt dazu euer Programm und platziert entsprechend euer Hindernis. Vergleicht die Werte doch einmal mit euren selbst gemessenen Zentimeter-Werten aus der Tabelle aus Seite 6.

Zentimeter	10 cm	20 cm	30 cm	40 cm	50 cm	60 cm	70 cm	80 cm
Sensor-Werte								

Nun könnt ihr endlich die Zentimeter-Angaben 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.

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

Gute Fahrt!

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

Abb. Quelle InfoSphere