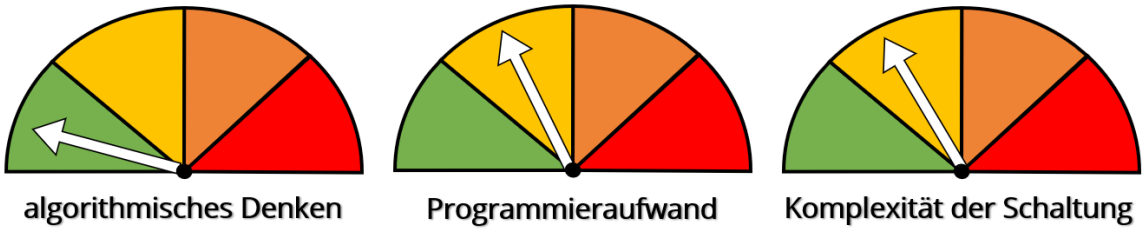
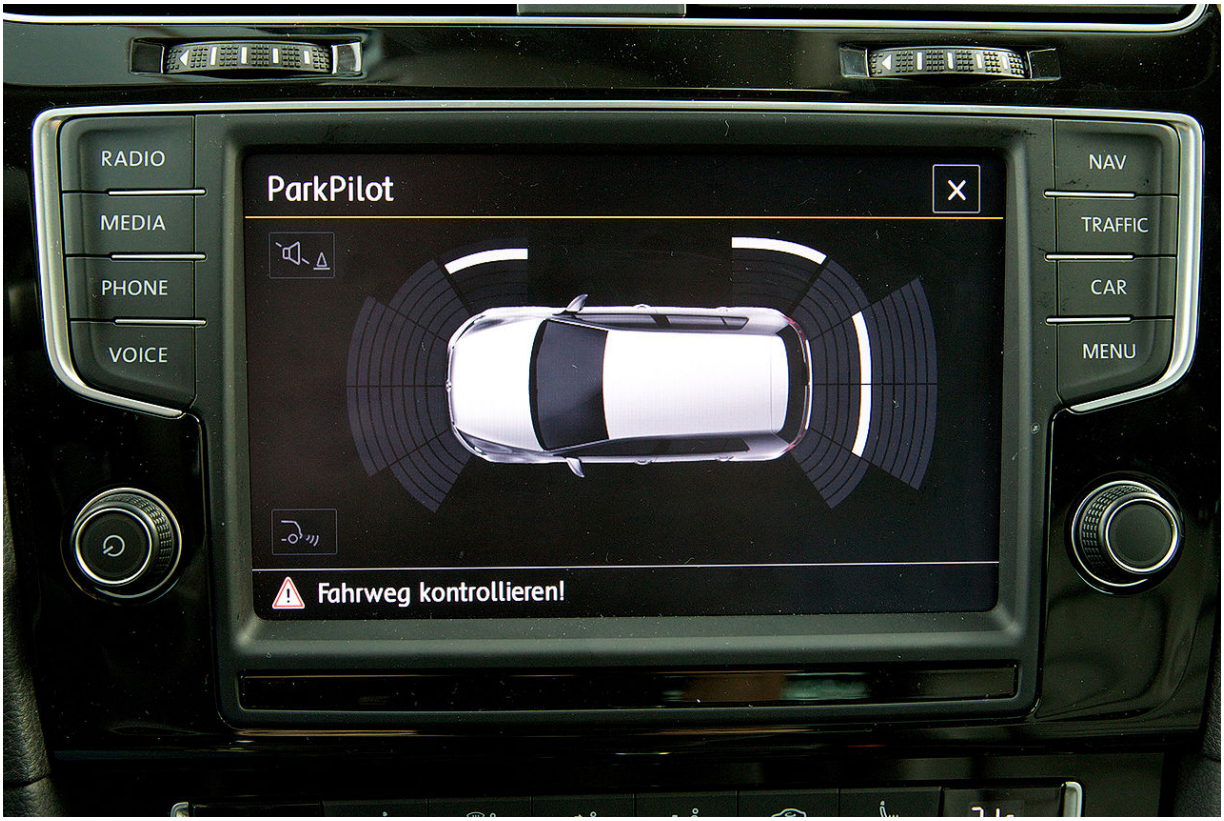
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 2 | Einparkhilfe





Nozilla, „Parking assist“, wikipedia.org , CC BY-SA 3.0

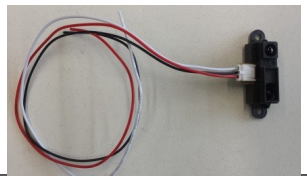
Professur für Didaktik der Informatik

UM WAS WIRD ES IN DIESER STATION GEHEN?

Jeder von euch, der demnächst in der Fahrschule seine Runden dreht, wird ein „Schreckensszenario“ kennenlernen: rückwärts einparken! Man fährt mehrmals vor und zurück, dreht ein und aus und am Ende steht man doch schief.
Was für ein Glück, dass in immer mehr Fahrzeugen Einparkhilfen eingebaut sind, die durch ihr drängendes und immer schneller werdendes Piepen vor dem Zusammenstoß warnen.
Aber wie funktioniert so eine Einparkhilfe? Wie misst das Auto den Abstand? Wie wird das Piepen erzeugt? Das wirst du in dieser Station lernen.

BENÖTIGTE BAUTEILE

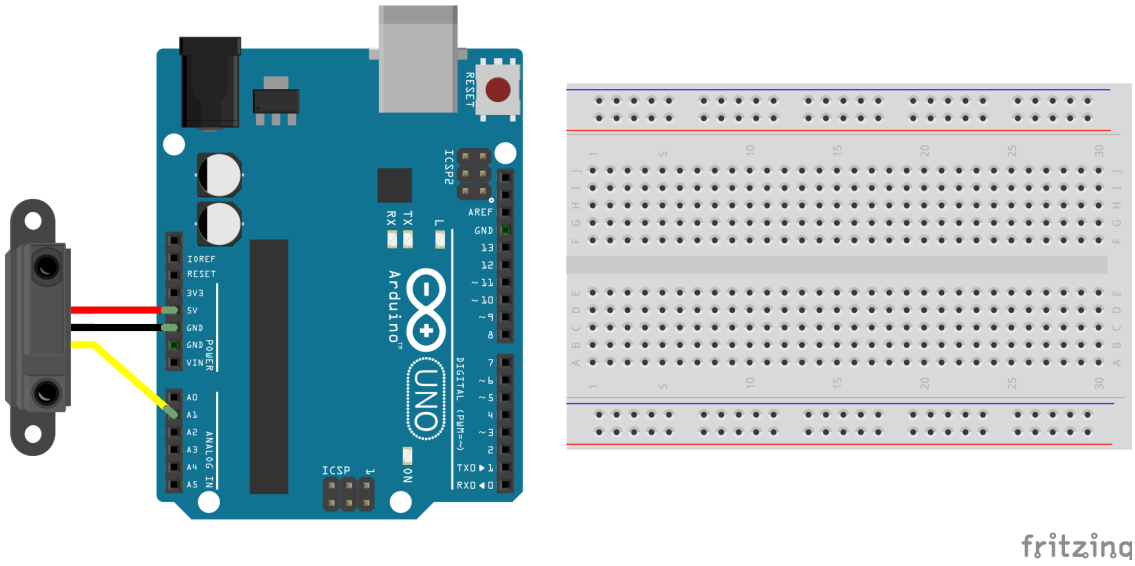
Zusätzlich zum Arduino und dem Steckbrett brauchst du folgende Bauteile:

Piezo-lautsprecher	Diesen Lautsprecher brauchst du, um das Piepsignal zu erzeugen. Beachte, dass der Lautsprecher ein kurzes und ein langes Bein hat!	
Infrarot-Distanz-Mess-Sensor	Durch diesen Sensor kannst du Abstände zwischen 10 und 80 cm messen.	

AUFGABE 1 – ABSTAND MESSEN

Als ersten Schritt sollst du den Abstand zwischen dem Infrarot-Sensor und einem Gegenstand messen. Befolge dabei die folgenden Punkte.

1. Sensor anschließen

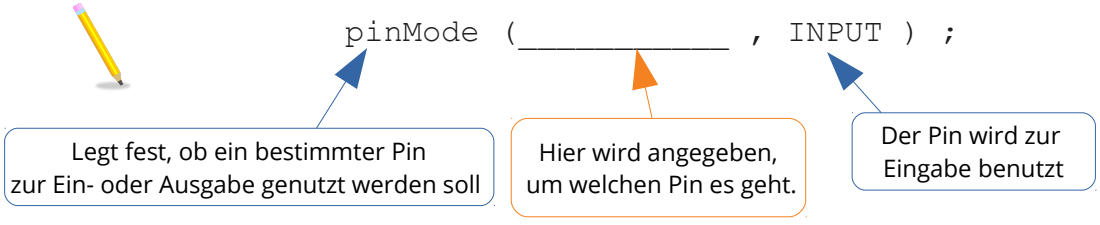


Verbinde das rote Kabel des Infrarotsensors mit 5V und das schwarze Kabel mit GND (0V). Das weiße (im Bild gelbe) Kabel verbindest du mit einem analogen Pin.

2. Pin-Variable definieren
Wie in der Einführungsstation, musst du auch hier den Arduino wissen lassen, an welchem Pin die Abstandsmessung stattfinden soll. Definiere dafür eine Variable. Das funktioniert für den analogen Pin genauso, wie für den digitalen. Falls du dafür Hilfe brauchst, schaue dir noch einmal in der Einstiegsstation Aufgabe 3 an.

3. Analogen Pin verbinden

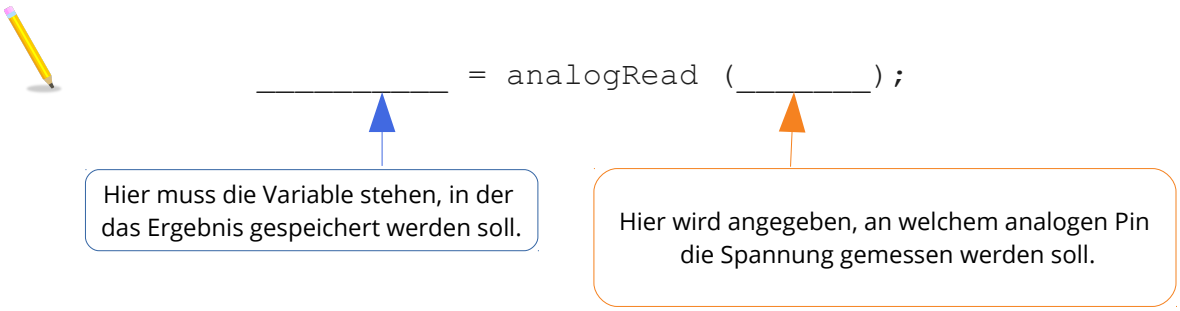
Damit der Arduino weiß, dass etwas am analogen Pin angeschlossen ist, musst du ihm das mit dem `pinMode` – Befehl im `setup()`-Teil sagen.



4. Speichern der Messergebnisse in einer Variable
   Nach der Messung des Abstands soll das Ergebnis gespeichert werden. Dafür brauchst du wieder eine Variable. Du weißt aber erst nach der Messung, welche Zahl die Variable haben soll. Deswegen musst du in der Definition der Variable noch keine Zahl zuordnen. Das kann so aussehen:

```
int abstand;
```

5. Befehl zur Messung
   Zuletzt musst du dem Arduino noch sagen, dass er die Abstandsmessung durchführen soll. Er muss also messen, welches Spannung am analogen Pin anliegt. Der Befehl dafür lautet:

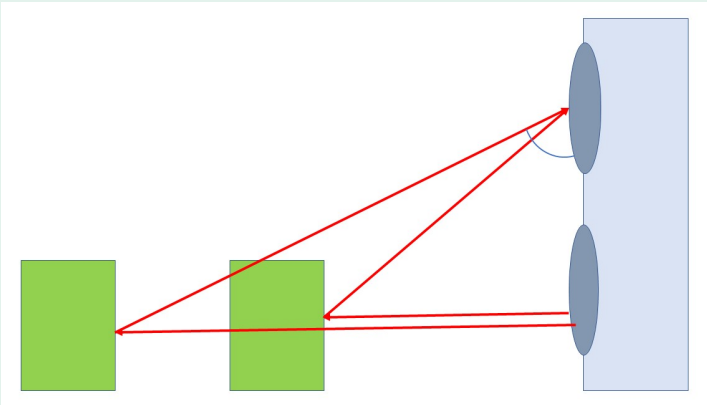


   Füge den Befehl im `loop()` – Teil des Programms ein.

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

WIE FUNKTIONIERT EIN INFRAROT-DISTANZ-MESS-SENSOR?

Um den Abstand von einem Gegenstand messen zu können, hat der Sensor zwei „Augen“. Ein Auge sendet einen infraroten (für uns unsichtbaren) Lichtstrahl aus. Wenn das Licht auf einen Gegenstand trifft, wird es reflektiert. Das funktioniert ungefähr so, wie wenn du Sonnenlicht mit einem Spiegel reflektierst. Das reflektierte Licht kommt dann beim anderen Auge des Sensors an. Um den Abstand zu ermitteln, erkennt der Sensor, in welchem Winkel das reflektierte Licht ankommt. Je kleiner der Abstand, desto größer ist der Winkel. Diese Information wird über einen analogen Pin an den Arduino weitergegeben. Und zwar, indem die Spannung verändert wird. Je größer die Spannung am Pin, desto weiter weg ist der Gegenstand vom Sensor.



AUFGABE 2 – MESSERGEBNIS ANZEIGEN

Benutze den seriellen Monitor, um dir das Ergebnis der Abstandsmessung anzeigen zu lassen. Teste dein Programm, indem du einen Gegenstand vor die Augen des Sensors stellst.
Seriellen Monitor starten

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 Messung „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 das Ergebnis gespeichert ist.

3. Installieren und testen
   Installiere das Programm auf dem Arduino. Öffne dann den seriellen Monitor in der Programmierumgebung. Beobachte die Messergebnisse. Verändere den Abstand zwischen dem Sensor und dem Gegenstand. Kannst du die Veränderung auch auf dem seriellen Monitor sehen? Wie verändert sich die Zahl?

   Je größer der Abstand ist, desto _____ wird das Ergebnis.

4. Pausen zwischen den Messungen
   Wie du vielleicht schon bemerkt hast, sieht man innerhalb kurzer Zeit hunderte Messergebnisse. Der Arduino arbeitet sehr schnell und führt den Befehl zum Messen immer wieder aus. Du brauchst natürlich nicht alle Messergebnisse. Befiehl dem Arduino deswegen nach jeder Messung eine Pause zu machen. Welchen Befehl musst du dafür verwenden? Du kennst ihn bereits!

   _____

   Installiere das Programm auf dem Arduino und öffne wieder den seriellen Monitor. Verändere die Länge der Pause, bis du die Messergebnisse auch alle lesen kannst.

Dein Programm funktioniert nicht? Kein Problem! Auf der letzten Seite findest du mögliche Fehler und wie du sie ausbessern kannst.

AUFGABE 3 – MESSERGEBNIS UMRECHNEN

Beim Testen solltest du gemerkt haben, dass die Zahl, je weiter der Sensor vom Gegenstand entfernt ist, kleiner wird. Die Zahlen an sich sehen aber noch komisch aus. Der Arduino misst den Abstand nicht in Zentimeter, oder Meter. Er hat seine eigene Einheit. In dieser Aufgabe sollst du herausfinden, welches Messergebnis für welchen Abstand steht.

WIE MISST DER INFRAROTSENSOR?

Der Infrarotsensor ist mit dem Arduino über einen Schaltkreiskreis verbunden. Die Kabel sind die einzige Verbindung über die sie Informationen austauschen können. Der Sensor schickt das Ergebnis seiner Messung also als elektrisches Signal. Je näher ein Gegenstand ist, desto größer ist die Spannung, die der Arduino am Ausgang des Sensors messen kann. Der Ausgang des Sensors nimmt dabei Werte zwischen 0V und 5V an.

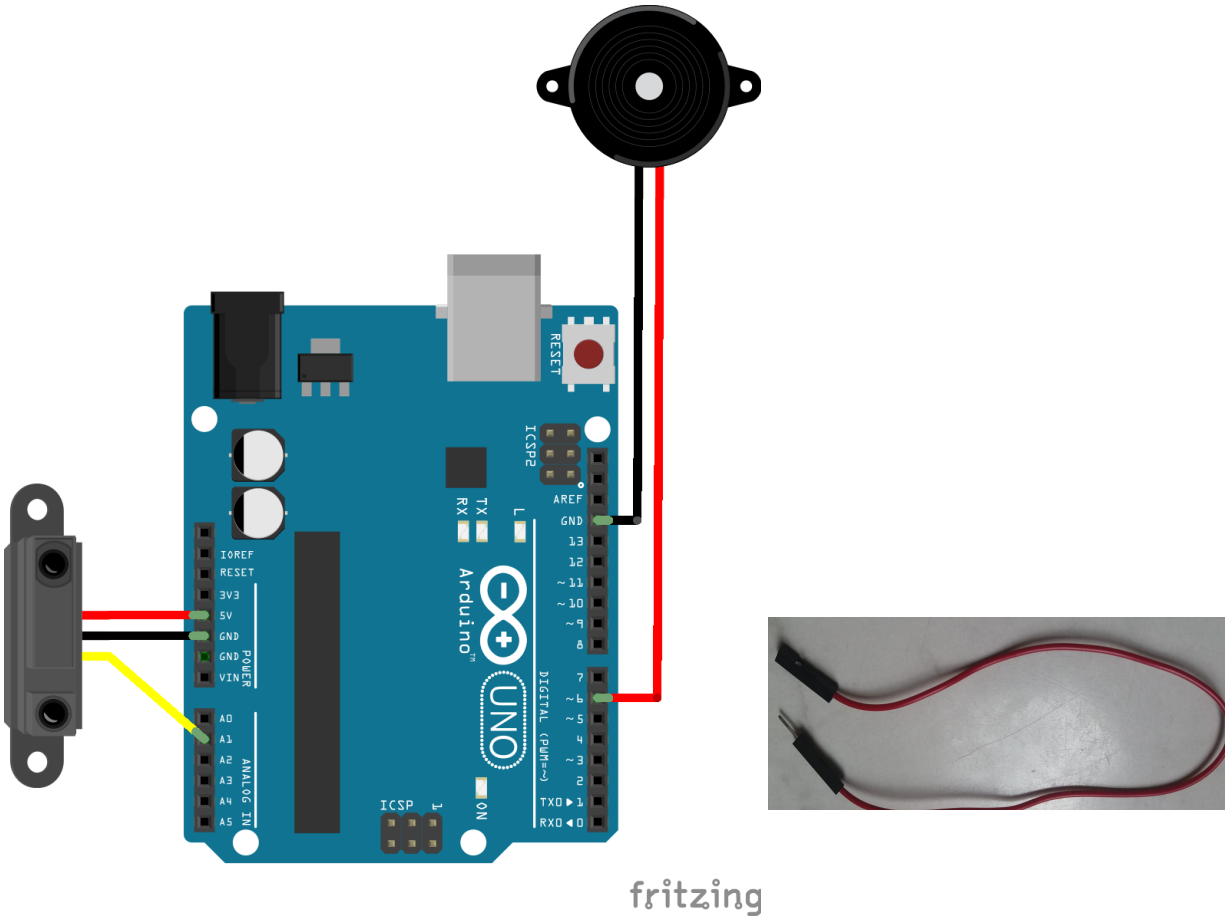
1. <u>Abstände messen</u>
   Um den Abstand in Zentimeter umzurechnen, musst du die Messergebnisse genau beobachten. Der Sensor kann Abstände zwischen 10 und 80 cm messen. Stelle ein Stück Papier in den unterschiedlichen Abständen vom Sensor entfernt hin (10cm, 20cm, etc.). Beobachte für jeden Abstand, welche Messergebnisse der Monitor anzeigt. Sie können sehr unterschiedlich sein. Schreibe das ungefähre Ergebnis in die Tabelle.

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

AUFGABE 4 – TON ERZEUGEN

Bisher hast du dem Sensor dazu gebracht den Abstand zu messen und du kannst die Ergebnisse auf dem seriellen Monitor einem ungefähren Abstand zuordnen. In dieser Aufgabe lässt du den Lautsprecher einen Ton machen, wenn der Abstand zum Gegenstand zu klein wird.

1. Lautsprecher anschließen



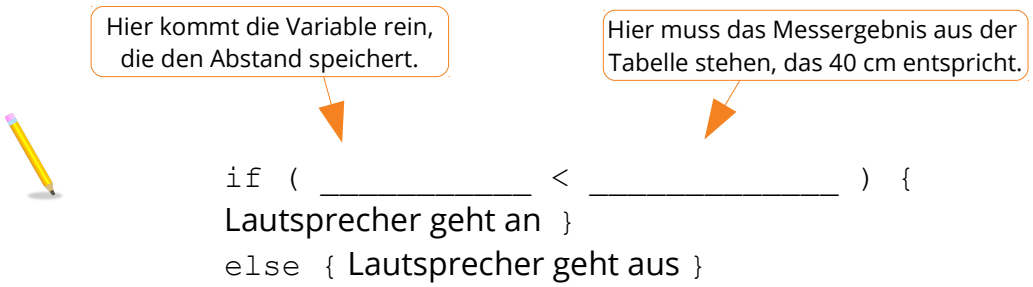
Zum Anschließen des Lautsprechers brauchst du zwei besondere Kabel (siehe Bild). In der Abbildung ist das schwarze Kabel an das kürzere Beinchen geschlossen und wird mit GND verbunden. Das lange Beinchen wird mit einem digitalen Pin verbunden.

2. Pin-Variable definieren und Pin verbinden

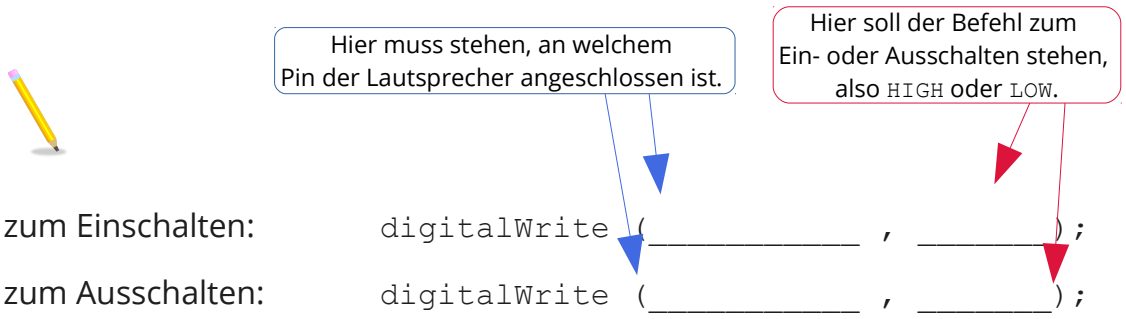
   Definiere eine Variable, die speichert, an welchem digitalen Pin der Lautsprecher angeschlossen ist. Verbinde den Pin dann als `OUTPUT`. (genaueres findest du bei Aufgabe 1, Punkt 2 und 3)

3. Wenn und sonst

Wenn der Abstand kleiner als 40 cm ist, soll der Lautsprecher angehen.
Sonst soll er ausgehen/ nicht an sein.
Um das zu programmieren, musst du folgenden Ausdruck benutzen. Du hast ihn schon einmal benutzt. Füge ihn im `loop`-Teil ein.



Jetzt fehlen noch die Befehle zum Angehen, bzw. Ausgehen. Dazu muss einfach der Lautsprecher ein-, bzw. ausgeschaltet werden. Das funktioniert ähnlich, wie das Ein- und Ausschalten einer LED.



4. Installieren und testen

Installiere das Programm auf dem Arduino und testet es mit einem Gegenstand. Der Lautsprecher sollte angehen, wenn der Gegenstand weniger als 40cm entfernt ist und ausgehen, wenn der Gegenstand wieder weiter weg ist.

AUFGABE 5 – PIEPEN

Super! Durch deine Einparkhilfe weißt du, ob ein Gegenstand näher ist, als 40 cm. Als Autofahrer ist es aber wichtig genauere Informationen zu haben. Vor allem auf engen Parkplätzen muss man anderen Gegenständen oft nahe kommen. Je näher man einem Gegenstand kommt, desto schneller sollte dann das Piepen werden. Das wirst du in dieser Aufgabe programmieren.

1. Piepen programmieren

   Bisher hast du mit einem durchgehenden Ton gearbeitet. Ein Piepen entsteht, wenn ein Ton immer angeschaltet und kurz danach wieder ausgeschaltet wird. Also wie beim Blinken einer LED. Bringe die folgenden Befehle in die richtige Reihenfolge, um ein Piepen zu programmieren.

   _____
   _____
   _____
   _____

   A `delay(300);`

   B `delay(300);`

   C `digitalWrite(_____, LOW);`

   D `digitalWrite(_____, HIGH);`

2. Abstände festlegen

   Um verschiedene Geschwindigkeiten für unterschiedliche Abstände zu programmieren, brauchst du wieder **Sonst wenn** – Ausdrücke. Füge die Ausdrücke zwischen den bereits geschriebenen `if` - und `else` – Teilen ein.

   ```
   if ( _____ < _____ ) {
   Lautsprecher geht an }

   else { Lautsprecher geht aus }
   ```

   ```
   else if ( _____ < _____ ) {
   mittleres piepen }
   else if ( _____ < _____ ) {
   langsames piepen }
   ```

   An diese Stellen muss die Variable stehen, die den Abstand speichert

   Setze hier die unterschiedlichen Messergebnisse aus der Tabelle ein. Bspw. die 30, 20 cm entsprechen.

3. Schnelligkeit des Piepen

   Je näher der Gegenstand kommt, desto schneller soll das Piepen werden. Um die Geschwindigkeit des Piepen zu verändern musst du nur eine Kleinigkeit im Code ändern. Welchen Befehl musst du wohl verändern?

   _____

   Kopiere die Piep-Befehle in die neuen Sonst wenn – Ausdrücke. Verändere dann die Geschwindigkeit des Piepen so, dass du ein mittleres und ein langsames Piepen hast.

4. Installieren und testen

   Installiere das Programm auf dem Arduino. Teste die Piepgeschwindigkeiten, indem du einen Gegenstand auf unterschiedliche Abstände hälst.

Du hast jetzt eine Einparkhilfe programmiert, wie es sie auch in einem Auto gibt.

Fotos: RWTH Aachen, InfoSphere
Screenshots: fritzing electronics made by easy und Arduino IDE 1.8.12 (windows)
Alle weiteren Grafiken: Patrick Binkert, EduInf@TUD