Lektion 7: Arbeitsblatt 7.1 - Kalibrierung der Hinderniserkennung

Du kannst die Empfindlichkeit des Hinderniserkennungssystems von Edison einstellen. Wenn du die Empfindlichkeit des Hinderniserkennungssystems erhöhst, kann Edison weiter entfernte Hindernisse erkennen. Wenn du das System weniger empfindlich machst, kann Edison nur sehr nahe Hindernisse erkennen. Benutze dieses Arbeitsblatt, um das Hinderniserkennungssystem von Edison einzustellen.

Schritt 1: Lesen des Barcodes

1. Stelle Edison auf die rechte Seite, mit Blick auf den Barcode
2. Drücke die Aufnahmetaste (rund) dreimal
3. Edison fährt vorwärts und scannt den Barcode

Barcode - Kalibrierung der Hinderniserkennung

Schritt 2: Einstellung der maximalen Empfindlichkeit

Nachdem du den Barcode gescannt hast, stelle Edison auf einen Tisch oder Schreibtisch und entfernen alle Hindernisse vor Edison. Drücke dann die Wiedergabetaste (Dreieck). Edison befindet sich nun im Kalibrierungsmodus.

Der linke Sensor wird zuerst kalibriert.

1. Drücke wiederholt die Wiedergabetaste (Dreieck), die die Empfindlichkeit erhöht, bis die rote LED auf der linken Seite flackert.
2. Drücke wiederholt die Aufnahmetaste (rund), die die Empfindlichkeit verringert, bis die LED vollständig aufhört zu flackern.
3. Drücke die Stopptaste (Quadrat), um zur Kalibrierung der rechten Seite zu wechseln.
4. Drücke wiederholt die Wiedergabetaste (Dreieck), bis die rechte rote LED flackert. Drücke nun wiederholt die Aufnahmetaste (rund), bis die LED vollständig aufhört zu flackern.
5. Drücke die Stopptaste, um die Kalibrierung abzuschließen.

Besonderer Hinweis: Benutzerdefinierte Empfindlichkeit

Es ist möglich, die Entfernung, in der Hindernisse erkannt werden, einzustellen. Scanne dazu den Barcode "Hinderniserkennung kalibrieren", platziere ein Hindernis vor Edison in der Entfernung, in der Edison Hindernisse erkennen soll, drücke die Wiedergabetaste und wiederhole dann die Schritte 1 bis 5.

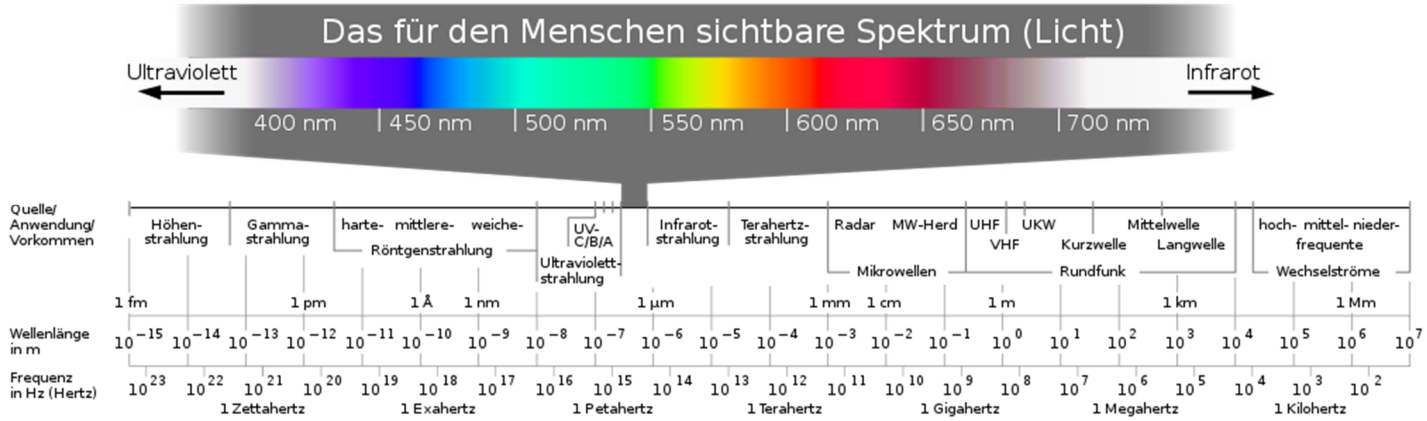
Lektion 7: Arbeitsblatt 7.1 - Infrarot-Hinderniserkennung

In dieser Aktivität erfährst du mehr über Infrarotlicht (IR) und wie Edison IR nutzen kann, um Hindernisse zu erkennen.

Was ist Infrarotlicht (IR)?

Es gibt ein breites Spektrum an Licht, von dem ein Teil für das menschliche Auge sichtbar ist und ein anderer Teil nicht. Infrarot, auch IR genannt, ist für den Menschen nicht sichtbar.

> ***Wusstest du das?*** Auch wenn Menschen es nicht sehen können, ist Infrarot eine Art von Licht. Deshalb funktioniert es auch im Dunkeln. Deshalb kannst du einen Fernseher mit einer Fernbedienung einschalten, auch wenn kein Licht im Raum brennt!



1 Horst Frank / Phrood / Anony, CC BY-SA 3.0 <http://creativecommons.org/licenses/by-sa/3.0/>, via Wikimedia Commons

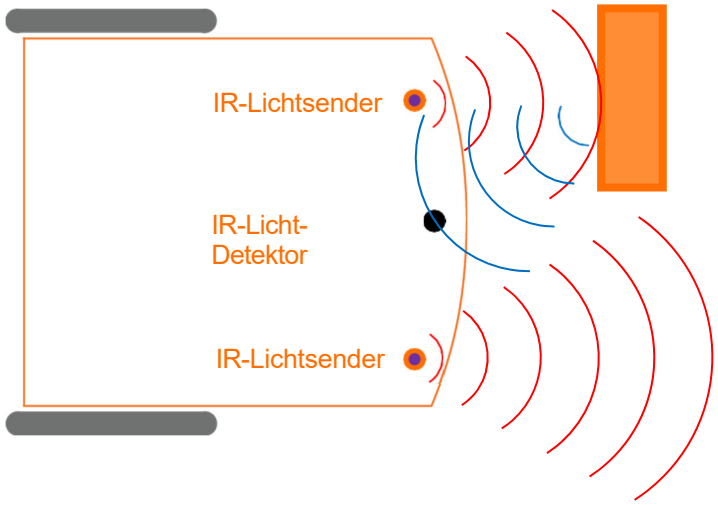
Edison und Infrarot

Der Edison-Roboter ist mit einem Infrarotsystem ausgestattet, das dem Roboter eine Art "Sehvermögen" verleiht. Dieses Infrarotsystem ermöglicht es Edison, Hindernisse in der Umgebung des Roboters zu erkennen.

Das Infrarotsystem von Edison besteht aus zwei IR-Lichtsenderdioden (oder LEDs) auf der Vorderseite. Eine befindet sich auf der linken und eine auf der rechten Seite. Edison hat auch einen IR-Sensor auf der Vorderseite, direkt in der Mitte.

Damit Edison Hindernisse erkennen kann, wird Infrarotlicht von den linken und rechten IR-LEDs nach vorne abgestrahlt. Trifft das IR-Licht auf ein Hindernis, z. B. eine Wand, wird es in Richtung Edison zurückgeworfen. Der IR-Sensor von Edison erkennt dann das reflektierte Licht.

Sieh dir die folgende Abbildung an:

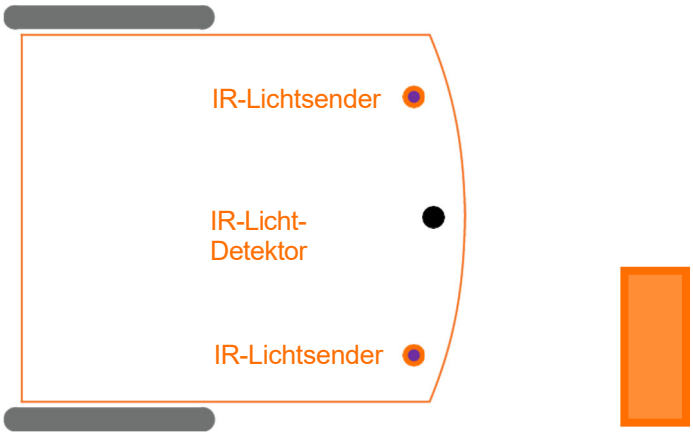


Edison sendet IR-Licht (rot dargestellt) von der linken und rechten IR-LED des Roboters aus. Reflektiertes IR-Licht (blau dargestellt) wird von Hindernissen zurückgeworfen und vom IR-Sensor von Edison erkannt.
In diesem Bild befindet sich ein Hindernis vor der linken Seite von Edison, aber nicht auf der rechten Seite. Deshalb wird nur das IR-Licht des linken Strahlers reflektiert.

Anhand des empfangenen Signals kann Edison feststellen, dass sich links ein Hindernis befindet, rechts jedoch kein Hindernis.

Du bist dran:

1. Zeichne das abgestrahlte und reflektierte IR-Licht für dieses Hindernis.



Lektion 7: Arbeitsblatt 7.2 - Ein Hindernis erkennen und anhalten

Bei dieser Aufgabe musst du ein Programm schreiben, das deinen Edison-Roboter so lange fahren lässt, bis er auf ein Hindernis stößt und dann anhält.

Sieh dir das folgende Programm an:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.ObstacleDetectionBeam(Ed.ON)
14
15 Ed.Drive(Ed.FORWARD,Ed.SPEED_5,Ed.DISTANCE_UNLIMITED)
16
17 while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
18     pass
19 Ed.Drive(Ed.STOP,1,1)
20
```

Dieses Programm weist Edison an, so lange zu fahren, bis er auf ein Hindernis stößt. Bei diesem Programm gibt es ein paar wichtige Dinge zu beachten:
Schau dir Zeile 13 des Programms an. Diese Zeile schaltet den Hinderniserkennungsstrahl von Edison ein. Wenn du Edisons Hinderniserkennungsstrahl in einem EdPy-Programm verwenden möchten, musst du den Strahl immer einschalten, bevor er im Programm verwendet wird.

Sieh dir nun Zeile 15 des Programms an. Diese Zeile setzt die Geschwindigkeit in diesem Programm auf 5. Wenn du die Hinderniserkennung verwendest, musst du eine etwas niedrigere Geschwindigkeit wählen, damit der Roboter ein Hindernis erkennen kann, bevor er mit ihm kollidiert. Wenn die Geschwindigkeit zu hoch ist, stößt der Roboter mit Hindernissen zusammen, bevor er sie erkennen kann.

Du bist dran:

Aufgabe 1: Wähle deine Hindernisse aus
Du musst gute Hindernisse für die Verwendung mit Edison auswählen. Wenn ein Hindernis zu klein ist oder nicht genug Infrarotlicht reflektiert, kann Edison es nicht erkennen. Wähle ein undurchsichtiges, aber nicht zu dunkles (z. B. nicht schwarzes) Objekt, das mindestens so groß ist wie Edison.
Für dieses Programm wäre die Wand des Raumes ein gutes Hindernis.

Aufgabe 2: Bereite deinen Edison vor
Wenn du ein Programm mit Hinderniserkennung ausführen möchtest, solltest du prüfen, ob die Hinderniserkennung deines Edison-Roboters auf die gewünschte Entfernung kalibriert ist. Verwende Arbeitsblatt 7.1, um deinen Edison zu kalibrieren. Möglicherweise musst du deinen Roboter mit einer benutzerdefinierten Empfindlichkeit kalibrieren, um sicherzustellen, dass dein Roboter das Hindernis erkennen und anhalten kann.

Aufgabe 3: Schreibe das Programm und führe es aus

Schreibe das Programm mit der EdPy-App und lade es auf deinen Edison-Roboter herunter. Führe dann das Programm aus, um zu sehen, wie es funktioniert.

Experimentiere mit verschiedenen Hindernissen, um zu sehen, was Edison erkennen kann und was nicht. Du kannst auch versuchen, die Kalibrierung der Hinderniserkennung von Edison auf verschiedene Entfernungen einzustellen, um zu sehen, was passiert.

1. Lösche die Zeile 'Ed.ObstacleDetectionBeam(Ed.ON)' aus deinem Programm. Versuche, das angepasste Programm herunterzuladen und auszuführen. Was ist passiert? Warum ist das passiert?

---

2. Überlege, wo du diese Art der unsichtbaren Erkennung in der realen Welt schon einmal gesehen haben. Beschreibe ein Beispiel.

---

3. Wo könnte diese Art von Detektionstechnologie deiner Meinung nach noch eingesetzt werden? Schreibe mindestens eine Idee auf, wie du diese Technologie einsetzen könntest.

Lektion 7: Arbeitsblatt 7.3 - Hindernisvermeidung

Bei dieser Aufgabe schreibst du ein Programm, das deinen Edison-Roboter so lange fahren lässt, bis er auf ein Hindernis stößt, dann wendet und wegfährt.

Schau dir das folgende Programm an:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.ObstacleDetectionBeam(Ed.ON)
14
15 Ed.Drive(Ed.FORWARD,Ed.SPEED_5,Ed.DISTANCE_UNLIMITED)
16
17 while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
18     pass
19 Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_5,180)
20 Ed.Drive(Ed.FORWARD,Ed.SPEED_5,10)
21
```

Dieses Programm weist Edison an, vorwärts zu fahren, bis ein Hindernis erkannt wird. Sobald Edison ein Hindernis erkennt, weist das Programm Edison an, sich um 180° zu drehen und 10 cm wegzufahren.

Du bist dran:

Aufgabe 1: Schreibe das Programm und führe es aus
Schreibe das Programm mit der EdPy-App und lade es auf deinen Edison-Roboter herunter. Führen dann das Programm aus, um zu sehen, wie es funktioniert.

Nachdem du das Programm ausgeführt hast, sieh es dir noch einmal an und überlege, wie du das Programm ändern könntest. Versuchen Sie, den Code so zu ändern, dass der Roboter sich anders verhält, wenn er ein Hindernis vor sich erkennt. (*Tipp*: Versuche es mit Geräuschen und Lichtern.)

1. Überlege, wie du das ursprüngliche Programm verbessern könntest. Was könntest du ändern, damit das Programm mehr tut, als nur umzudrehen und wegzufahren, nachdem es auf ein Hindernis gestoßen ist?

Aufgabe 2: Syntaxfehler und logische Fehler

Programmierern unterlaufen häufig Tippfehler, so genannte Syntaxfehler. Es ist wichtig, dass du deine eigenen Syntaxfehler erkennst, damit du deinen Code korrigieren kannst.

Beim Programmieren kann auch eine andere Art von Fehler auftreten, der so genannte logische Fehler. Beim Programmieren ist jeder Fehler, der kein Syntaxfehler ist, ein logischer Fehler.

Wenn ein logischer Fehler in einem Programm auftritt, verhält sich der Code nicht so, wie der Programmierer es erwartet. Wenn du in EdPy einen logischen Fehler in deinem Programm hast, wird das Programm normalerweise trotzdem auf Edison heruntergeladen. Wenn du das Programm jedoch ausführst, wird es sich nicht so verhalten, wie du es erwartest.

Ein logischer Fehler kann so einfach sein wie die Verwendung einer falschen Funktion oder das Weglassen einer Funktion. Ein Beispiel für einen logischen Fehler wäre das Schreiben eines Programms, das die Hinderniserkennung verwendet, aber den Hinderniserkennungsstrahl nicht einschaltet.

Sieh dir das folgende Programm an:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.ObstacleDetectionBeam(Ed.ON)
14
15 Ed.Drive(Ed.FORWARD,Ed.SPEED_5, Ed.DISTANCE_UNLIMITED)
16
17 While Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE
18 pass
19 Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_5,145)
20 Ed.Drive(Ed.FORWARD, Ed.SPEED_5,20)
21
```

Das Programm soll den Edison-Roboter so lange fahren lassen, bis er auf ein Hindernis stößt, dann um 135° drehen und 20 cm vom Hindernis wegfahren.

In dem Programm sind fünf Fehler enthalten. Kannst du alle fünf Fehler im Programm identifizieren und feststellen, ob es sich um Syntaxfehler oder logische Fehler handelt? Trage deine Antworten in die Tabelle auf der nächsten Seite ein.

Tipp: du kannst dieses Programm in EdPy schreiben und die Schaltfläche "Code prüfen" verwenden, um die Syntaxfehler zu finden.

Fehler #	Zeile #	Fehlertyp (Syntax oder logisch)	Fehlerbeschreibung
1			
2			
3			
4			
5			

Lektion 7: Arbeitsblatt 7.4 - Ein Hindernis als Ereignis erkennen

In dieser Übung schreibst du ein ereignisgesteuertes Programm, um deinen Edison-Roboter kontinuierlich vorwärts fahren zu lassen, während er Hindernissen ausweicht.

Ereignisgesteuerte Programmierung

In der Programmierung ist ein Ereignis etwas, das außerhalb des Programmcodes auftritt und den Programmablauf beeinflusst. Ein Ereignis kann das Drücken einer Taste oder die Übermittlung von Informationen von einem Sensor sein.

Viele Programmiersprachen, darunter auch Python, ermöglichen es Programmierern, Code zu erstellen, der auf eine Reihe von Ereignissen reagieren kann. Diese Art der Programmierung wird als "ereignisgesteuerte Programmierung" bezeichnet.

Wenn du ein Programm schreibest, das ereignisgesteuert ist, musst du auch einen Code schreiben, der diese Ereignisse verarbeitet. Es gibt zwei Möglichkeiten, dies zu tun: Entweder lässt man das Programm in einer Schleife warten oder man verwendet Unterbrechungen.

Wir unterbrechen diese Lektion, um über ... zu sprechen.

Wie du bereits weißt, bewegen sich Programme normalerweise sequentiell durch den Code, Zeile für Zeile. Es gibt jedoch Möglichkeiten, den Code anders zu bewegen, z. B. durch die Verwendung von Schleifen.

Du kannst auch die Art und Weise, wie ein Programm abläuft, beeinflussen, indem du einen "Interrupt" (Unterbrechung) verwendest.

Ein Interrupt ist ein Codeabschnitt, der das Hauptprogramm unterbricht und selbst ausgeführt wird. Sobald der Unterbrechungscode abgeschlossen ist, kehrt das Programm an die Stelle zurück, an der es im Hauptprogramm aufgehört hat.

Interrupts sind immer Funktionen, die irgendwo im Programm definiert sind. In der Regel handelt es sich um kurze Codeabschnitte, die eine bestimmte Aufgabe erfüllen sollen, ohne den Ablauf des Hauptprogramms wesentlich zu stören.

Programmierer verwenden Unterbrechungen, weil sie es einem Programm ermöglichen, jederzeit auf ein Ereignis zu reagieren, während das Programm läuft. Mit anderen Worten: Durch die Verwendung von Unterbrechungen muss ein Programmierer nicht genau vorhersagen, wann das Ereignis während eines Programms eintritt.

Event Handler und Interrupts

Bei der Verwendung von Interrupts in der ereignisgesteuerten Programmierung müssen Sie "Event-Handler"(Ereignis-Behandler) verwenden. Ein Event-Handler ist eine Möglichkeit, einen Interrupt an ein bestimmtes Ereignis zu binden.

Um einen Event-Handler zu verwenden, musst du ihn zunächst in deinem Code einrichten bzw. "registrieren". Sobald der Event-Handler registriert ist, wird er ständig auf das entsprechende Ereignis überwacht.
Immer wenn dieses Ereignis eintritt, löst der Event-Handler den Interrupt aus, der die Funktion aufruft und ausführt und dann zum Hauptprogramm zurückkehrt.

Für ein besseres Verständnis probieren wir das ganze einmal aus.

Sieh dir das folgende Programm an:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.ObstacleDetectionBeam(Ed.ON)
13 Ed.RegisterEventHandler(Ed.EVENT_OBSTACLE_AHEAD, "avoidObstacle")
14
15 while True:
16     Ed.Drive(Ed.FORWARD,Ed.SPEED_5,Ed.DISTANCE_UNLIMITED)
17
18 def avoidObstacle():
19     Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_5,180)
20     Ed.ReadObstacleDetection()
21
```

In EdPy verwenden wir die Funktion 'Ed.RegisterEventHandler', um einen Event-Handler zu registrieren.

Die Funktion "Ed.RegisterEventHandler" hat zwei Parameter. Der erste Parameter ist das Ereignis, das eintreten soll, und der zweite Parameter ist die Funktion, die aufgerufen wird, wenn das Ereignis eintritt.

Sieh dir Zeile 13 des Programms an. In dieser Zeile wird ein Event-Handler registriert, so dass die Funktion "avoidObstacle" aufgerufen wird, wenn "EVENT_OBSTACLE_AHEAD" eintritt.

Du bist dran:

Schreibe das Programm mit der EdPy-App und lade es auf deinen Edison-Roboter herunter. Führe dann das Programm aus, um zu sehen, wie es funktioniert.

1. Beschreibe, was der Roboter macht.

2. Welche Codezeilen werden in diesem Programm ausgeführt, wenn der Roboter ein Hindernis vor sich erkennt? Warum führt das Programm diese Zeilen aus?

3. Warum ist Ed.ReadObstacleDetection in diesem Programm enthalten? Was macht die Zeile 20? *Tipp:* Schau dir Arbeitsblatt 2.5 an, um einen Hinweis zu erhalten, oder versuche, Zeile 20 zu entfernen und das Programm auszuführen.

Versuche es!

Was könnte Edison sonst noch tun, wenn es ein Hindernis vor sich erkennt? Versuche, den Code so zu ändern, dass Edison ein anderes Verhalten zeigt, wenn es ein Hindernis vor sich erkennt. Experimentiere mit dem Programm, um zu sehen, was funktioniert und was nicht.

Lektion 7: Arbeitsblatt 7.5 - Hinderniserkennung rechts und links

In dieser Aufgabe schreibst du ein Programm, mit dem Edison auf Hindernisse links oder rechts vom Roboter reagieren kann. Dazu werden wir "if-Anweisungen" (Wenn-Anweisung) verwenden.

If-Anweisungen

Ein wichtiger Teil des Programmierens ist das Treffen von Entscheidungen. Die häufigste Art, dies zu tun, ist die Verwendung einer "if-Anweisung".

Eine 'if-Anweisung' fragt, ob eine Bedingung wahr oder falsch ist. Wenn das Ergebnis wahr ist, führt das Programm den auf die if-Anweisung folgenden Anweisungsblock aus. Ist das Ergebnis falsch, ignoriert das Programm die Anweisungen innerhalb der if-Anweisung und fährt mit der nächsten Codezeile außerhalb der if-Anweisung fort.

Sieh dir das folgende Programm an:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.ObstacleDetectionBeam(Ed.ON)
13
14 while True:
15     if Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE:
16         Ed.PlayBeep()
17         Ed.ReadObstacleDetection()
18
```

Dieses Programm verwendet eine if-Anweisung, um dem Roboter die Möglichkeit zu geben, Entscheidungen ohne menschliche Anleitung zu treffen. Wenn ein Roboter auf diese Weise selbständig Entscheidungen treffen kann, nennt man ihn einen autonomen Roboter.

Du bist dran:

Aufgabe 1: Piepse, wenn es ein Hindernis gibt

Schreibe das obige Programm mit der EdPy-App, lade es auf deinen Edison- Roboter herunter und führe das Programm aus. Versuche dann, ein Hindernis,
z. B. deine Hand, in und aus dem Hinderniserkennungsstrahl von Edison zu bewegen, um zu sehen, was passiert.

1. Edison kann nun unterschiedlich auf verschiedene Reize reagieren und eine "Entscheidung" treffen, was er tun soll. Bedeutet dies, dass Edison über Intelligenz verfügt? *Tipp:* Um das zu entscheiden, solltest du ein wenig über künstliche Intelligenz recherchieren.

Aufgabe 2: Piepse, wenn es ein Hindernis gibt, sonst schalte das Licht ein

Du kannst einem Programm nicht nur sagen, was es tun soll, wenn die Bedingung einer if-Anweisung wahr ist, sondern auch, was es tun soll, wenn die Bedingung falsch ist.

If, Else

Die Verwendung von if-Anweisungen mit 'else'(Sonst) ermöglicht es uns, Programme zu schreiben, die kompliziertere Entscheidungen treffen. If/else-Anweisungen sind grundsätzlich eine Möglichkeit, eine Entscheidung zwischen zwei Dingen zu treffen.

In Python lautet die Syntax für if/else:

```
if {Ausdruck}:
    Anweisung(en)
else:
    Aussage(n)
```

Das Programm läuft sequentiell von oben nach unten ab, beginnend mit der if-Bedingung. Wenn die if-Anweisung wahr ist, führt das Programm den eingerückten Code für den if-Ausdruck aus und überspringt den else-Code. Wenn die if-Anweisung jedoch falsch ist, überspringt das Programm diesen Abschnitt des eingerückten Codes und führt stattdessen den eingerückten else-Code aus.

Schreibe das folgende Programm:

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON)

while True:
    if Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE:
        Ed.LeftLed(Ed.OFF)
        Ed.PlayBeep()
        Ed.TimeWait(100, Ed.TIME_MILLISECONDS)
        Ed.ReadObstacleDetection()
    else:
        Ed.LeftLed(Ed.ON)

```

Lade das Programm herunter und führe es aus. Versuche, ein Hindernis in und aus Edisons Hinderniserkennungsstrahl zu bewegen, um zu sehen, was passiert.

Dieses Programm hat zwei Pfade: einen, wenn ein Hindernis erkannt wird, und einen, wenn kein Hindernis erkannt wird.

if, elif, else

Du kannst auch ein Programm erstellen, das eine Entscheidung unter mehr als zwei Bedingungen trifft. Dazu verwenden Sie eine andere Python-Syntaxstruktur:

```
If Ausdruck:
    Anweisung(en)
elif Ausdruck:
    Anweisung(en)
sonst:
    Aussage(n)
```

Mit 'Elif' sagt man in Python 'else if'. du kannst elif verwenden, um ein Programm mit mehreren if-Bedingungen zu schreiben.

Ein Programm, das if/elif/else verwendet, läuft immer noch sequentiell von oben nach unten ab. Sobald das Programm einen eingerückten Code innerhalb eines Teils der if-Anweisungsstruktur ausführt, überspringt es den Rest der Struktur und fährt mit der nächsten Codezeile außerhalb der Struktur fort.

Das heißt, wenn die if-Anweisung am Anfang wahr ist, führt das Programm den eingerückten Code für den if-Ausdruck aus und überspringt alle elif-Abschnitte sowie den else-Abschnitt, falls es einen solchen gibt. Wenn die if-Anweisung jedoch falsch ist,überspringt das Programm diesen Abschnitt des eingerückten Codes und geht zum ersten elif-Abschnitt über.