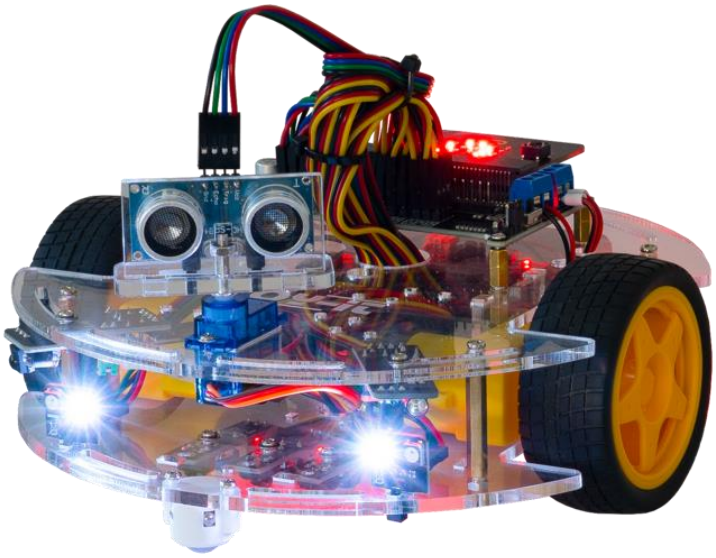
Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

JoyCar

Kursunterlagen



Name: _____

V1.0 2022

Electronics4you

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Diese Einführung und Exploration kombiniert Materialien aus verschiedenen Open Education Projekten.

Einige Einführungsaufgaben stammen aus dem iMake-IT-Projekt der Pädagogischen Hochschule Schwyz (PHSZ) und aus dem OpenSource-Material unter https://makecode.microbit.org/ .
Der Workshop wird im Rahmen des electronics4you Workshops des ISC konzipiert und organisiert.

Weitere Informationen über den PHSZ Projekt unter:
phsz-facile.ch/imake-it
imake-it.ch
Und Kontakt mit der School of Engineering Diversity unter:
oppg@zhaw.ch
diversity.engineering@zhaw.ch

Impressum

Version 1.0 (Mai 2022)
Autoren: Marina de Queiroz Tavares, Seraina Betschart, Jason Curtins, Moritz Oppliger
oppg@zhaw.ch

Bilder, Grafiken, Screenshots: PHSZ, ZHAW SoE
Icons: S.20 Misc Dice by glitch (openclipart.org), S.23 Compass Rose by Firkin (openclipart.org), S.27 Blue Robot by Scout (openclipart.org), S.30 Piano Keyboard by GDJ (openclipart.org), S.33 Rock-Paper-Scissors by uoresch (openclipart.org)

Vorwort

Der micro:bit ist ein Mikrocontroller, welchen man relativ einfach Programmieren kann. Ein Mikrocontroller ist wie ein kleiner Computer, welcher Dinge abspeichern, Sensoren auslesen, rechnen und auch Dinge anzeigen kann. Der micro:bit hat wie der Mensch einige Sinne. Zum Beispiel kann der micro:bit über ein Mikrofon hören oder mit dem kleinen Bildschirm Muster, Zahlen und Texte darstellen. Auch hat der micro:bit eine Tastatur mit 2 Tastern und einem Touch-Button.

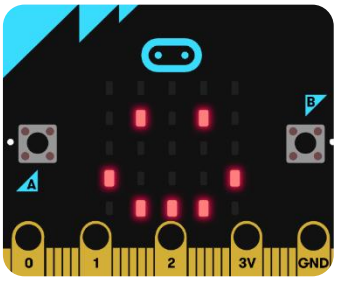
Der JoyCar ist ein zweimotoriges Fahrzeug, welches verschiedene Sensoren enthält. Der micro:bit steuert die Motoren an. Auch liest der micro:bit die Sensoren aus und entscheidet, wo der JoyCar hin fährt. Durch programmieren vom micro:bit entscheidest du, wie der Roboter handeln muss, wenn ein Objekt vor ihm steht.

Aber bevor wir mit dem Programmieren des Roboters beginnen, starten wir mit dem Zusammenbau des Roboters. Befolge dazu die beiliegende Bauanleitung. Danach wirst du mit ein paar einfachen Programmieraufgaben den micro:bit besser kennenlernen.

Gleichzeitig zum Programmieren wirst du immer neue Sensoren kennenlernen. Diese Sensoren messen Distanzen und Geschwindigkeiten auf unterschiedliche Weise.

Am Ende des Kurses weisst du, wie die wichtigsten Sensoren funktionieren und wie du einfache Programme für einen Roboter schreibst.

Viel Spass!



Website zum JoyCar: https://joycar.joy-it.net/de/

Website zum micro:bit: https://microbit.org/

Inhaltsverzeichnis

1 Grundlagen

Zuerst starten wir mit einfachen Programmen. Dadurch lernst du die wichtigsten Punkte des Programmierens kennen.

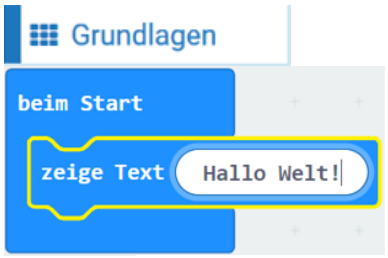
1.1 Hallo Welt

Meisten startet man beim Programmieren mit dem Programm «Hallo Welt». Der micro:bit zeigt auf dem Bildschirm den Text «Hallo Welt an». Das Programm wird eigentlich immer verwendet, um die Verbindung zum Mikrocontroller zu testen.

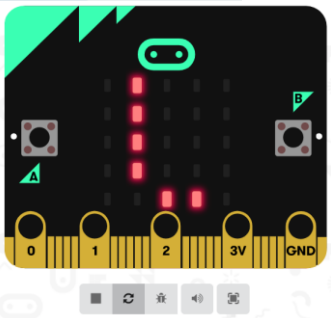
Baue den untenstehenden Code nach. Du findest die Befehle bei den verwendeten Befehlsgruppen

Verwendete Befehlsgruppen:

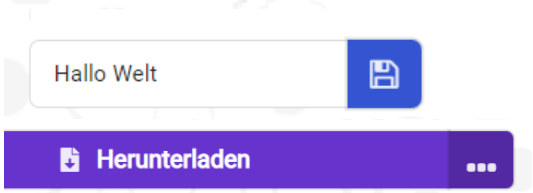
Code:



Überprüfe die Funktion im Simulator und speichere das Programm ab. Danach kannst du das Programm auf den micro:bit herunterladen



Benötigte Elemente zum Abspeichern der Programme und zum das Programm auf den micro:bit herunterzuladen.



Wenn du alles richtig gemacht hast, läuft der Text einmal auf dem micro:bit Display.

Hinweis: Wenn du das Programm neu starten willst, drücke auf die Reset-Taste auf der Rückseite des micro:bits.

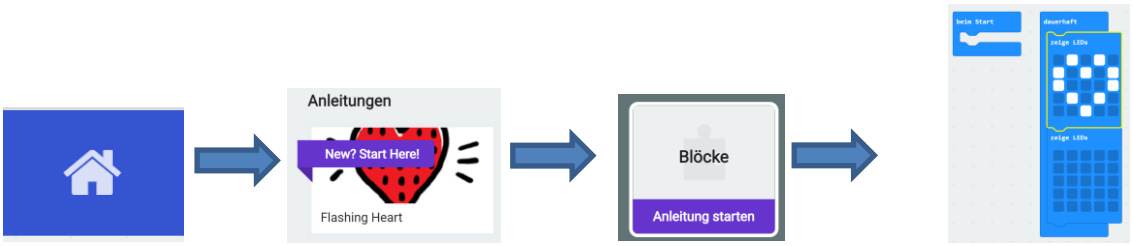
Challenge 1a

Verändere das Programm so, dass ein anderer Text auf dem LED-Display angezeigt wird.

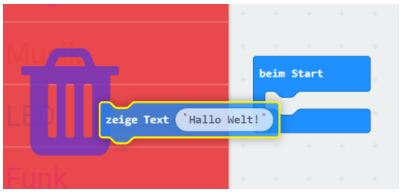
Hinweis: Um ein neues oder verändertes Programm auf dem micro:bit zu testen, muss es jedes Mal von Neuem auf den micro:bit heruntergeladen werden. Dabei wird das alte Programm überschrieben.

Challenge 1b

Wir machen jetzt ein neues Programm, aber statt eines Textes zeigen wir ein Herz, das erscheint und verschwindet. Wenn du einen Tipp brauchst, schau dir das Tutorial auf der Homepage von makecode.microbit.org an.



Hinweis: Um nicht mehr benötigte Programmierteile zu löschen, ziehe sie mit der Maus nach links, bis ein Mülleimersymbol erscheint.

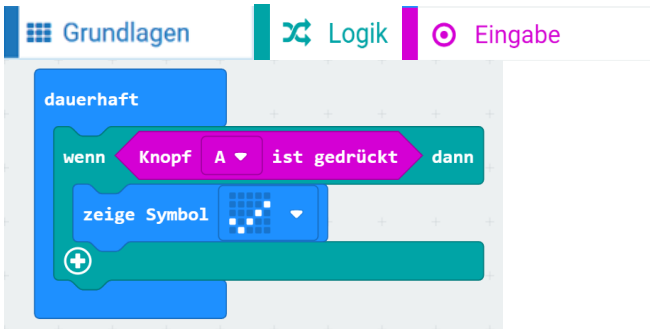


1.2 Taster A und B benutzen

Mit den Tastern A und B können Eingaben an den micro:bit gemacht werden. Dadurch kannst du den micro:bit steuern. Zusätzlich lernst du Bedingungen, wann etwas passieren soll. Dazu dient der Block: «Wenn-Dann-Sonst».

Verwendete Befehlsgruppen:

Code:



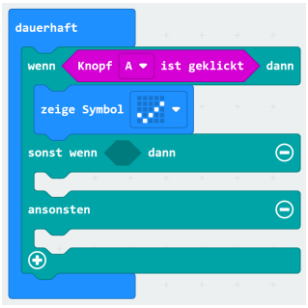
Wenn man den Taster A drückt, erscheint ein Häkchen-Symbol auf dem Display.

Hinweis: Teste dein Programm auf dem Simulator, bevor du es auf das Board herunterlädst.

Challenge 2a

Erweitere das Programm so, dass auch für den Taster B ein Symbol gezeigt wird.

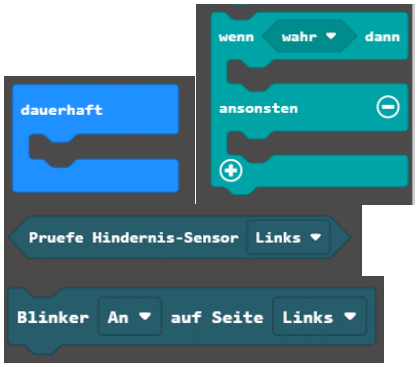
Hinweis: Drücke das «+» Zeichen innerhalb des «Wenn-Dann»-Blocks, um weitere Bedingungen zu prüfen.



Challenge 2b

Erweitere das Programm, sodass anstelle des Tasters die Objektsensoren des JoyCars Links und Rechts verwendet werden. Wenn der linke Objektsensor ein Objekt erkennt, sollte der Blinker links blinken. Für den Objektsensor rechts gilt das Gleiche

Hinweis: Du benötigst folgende Blöcke



Objektsensoren

Die Objektsensoren bestehen aus einer Infrarot-LED und einem Infrarotdetektor. Infrarot ist Licht, welches für uns Menschen nicht sichtbar ist. Einige Tiere können Infrarot sehen. Die LED strahlt das Licht aus. Wenn kein Objekt vor dem Objektsensor ist, dann kommt kein Licht zurück. Der Infrarotdetektor hat somit kein Signal. Ist ein Objekt vor dem Objektsensor, dann wird das ausgestrahlte Licht reflektiert, welches der Infrarotdetektor erkennt.

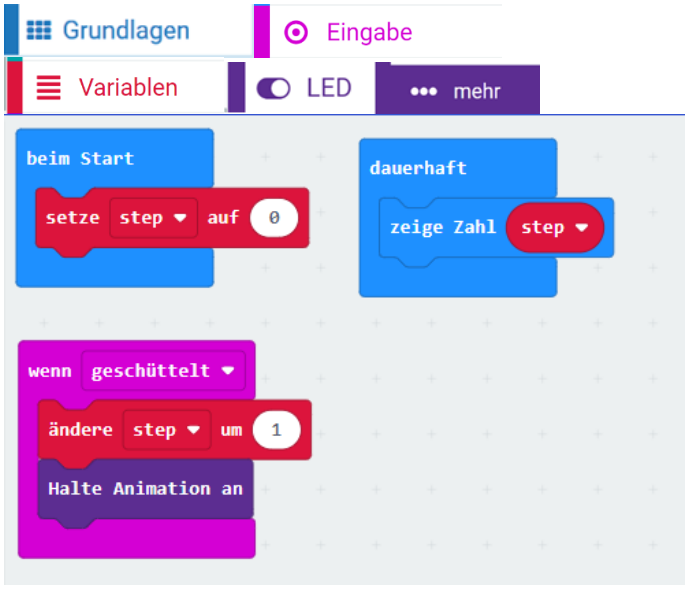
Die Objektsensoren Mit dem Potentiometer am Objektsensor kannst du die Distanz einstellen, ab wann der Objektsensor das Objekt erkennt.

1.3 Schrittzähler

Als Nächstes wollen wir den Bewegungssensor benutzen, um Schritte zu erkennen, und eine Variable, um die Schritte zu zählen. Neu benötigen wir dazu eine **Variable**. Eine Variable ist eine Zahl, welche einen Namen hat. Die Zahl kann man verändern, auf einen Wert setzen oder auch damit rechnen.

Verwendete Befehlsgruppen:

Code:

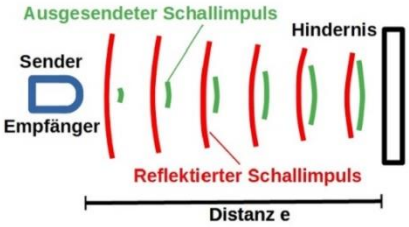


Lade den Code herunter und befestige einen Batterieblock an deinem micro:bit.
Du bist bereit, dein Programm zu testen! Stecke den micro:bit in deine Socke oder befestige ihn mit einem Band um deinen Knöchel und laufe herum!

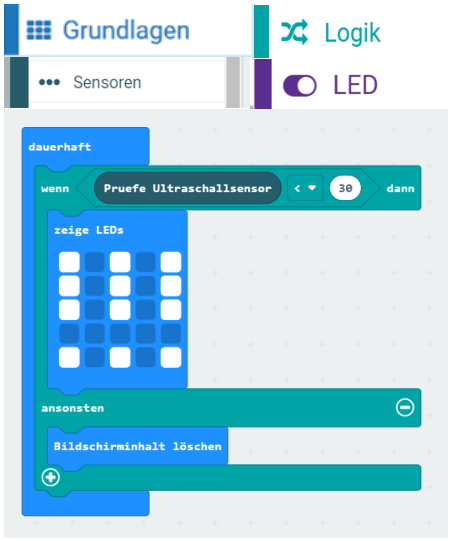
1.4 Distanzmessung

Der JoyCar misst Distanzen mit einem Ultraschallsensor (Schallwellen, in einem nicht hörbaren Bereich. Fledermäuse nutzen Ultraschall zur Orientierung.). Dieser Sensor sendet eine Schallwelle aus. Die Schallwelle wird an einem Objekt reflektiert und wieder gemessen. Aus der Laufzeit t der Schallwelle und der Schallgeschwindigkeit c=346 m/s kann die Distanz d zum Objekt berechnet werden:

$$d = \frac{t \cdot c}{2}$$



Verwendete Befehlsgruppen:



Code:

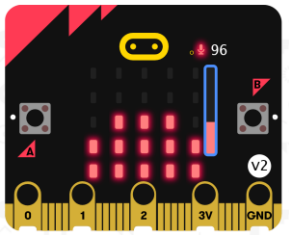
Baue den Code nach. Lade dann das Programm auf den micro:bit herunter.

Was macht das Programm?

_____

Challenge 4a

Jetzt wollen wir die gemessene Distanz im Detail verfolgen. Versuche eine grafische Darstellung mit dem Befehl "Balkendiagramm zeichnen", wie unten gezeichnet.

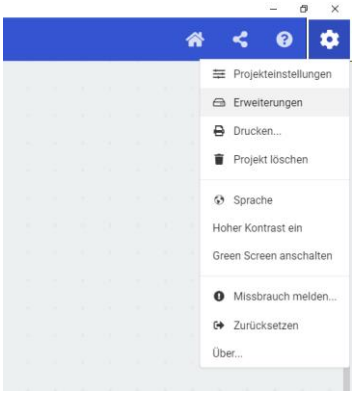


Erweitere das Programm folgendermassen:

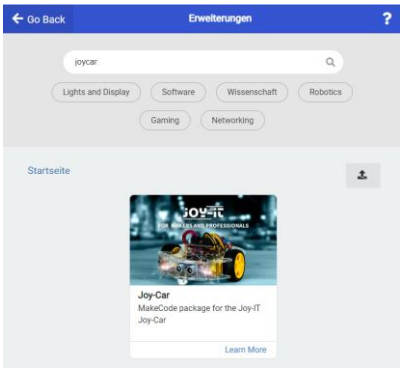
- Kein Taster gedrückt: Das Säulendiagramm wird angezeigt
- Taster A gedrückt: Die Distanz wird auf den Bildschirm geschrieben

2 JoyCar und micro:bit

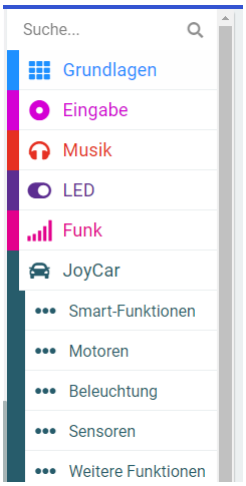
Wenn du ein neues Projekt erstellst, musst du zuerst die **Joy-Car Erweiterung** zu deinem Projekt hinzufügen, damit du die Funktionen des Joy-Cars nutzen kannst. Gehe dazu auf Einstellungen / Erweiterungen.



Suche nach JoyCar und füge die Erweiterung dem Projekt hinzu.



Alle Funktionalitäten des Joy-Cars sind für dich in dieser Erweiterung zusammengestellt. Diese Erweiterungen erscheinen nun als neue Befehlsgruppe «JoyCar».



3 Erster Parkour

Fahre einen vorgegebenen Parkour mit dem JoyCar ohne Hilfe von Sensoren. Die Steuerung erfolgt zeitlich. Das heisst, du misst die Distanz, die der JoyCar pro Zeit fährt. Danach berechnest du, wie lange der JoyCar fahren muss für die benötigte Distanz.

3.1 Gerade Strecke fahren

Lege dir die Geschwindigkeit des Fahrzeugs fest (z.b. 50%).

Fahre bei Tastendruck Taster A 1 Sekunde mit dieser Geschwindigkeit und miss die Strecke.

Wie lange musst du bei der Geschwindigkeit 50 %für genau einen Meter Distanz Fahren? (Überlege, wie du dies am einfachsten herausfindest)

_____

Kalibrier das Fahrzeug so, dass es gerade fährt, dazu muss der eine Motor leicht langsamer fahren.

Welcher Motor muss um wie viel Verzögert werden?

_____

3.2 Kurve fahren

Versuche so genau wie möglich eine 45° und eine 90° Drehung zu machen (Hinweis, ähnlich wie in Aufgabe 1).

Wie lange musst du bei der Geschwindigkeit 50 % Drehen?

90° Links: _____ 45° Links: _____

90° Rechts: _____ 45° Rechts: _____

3.3 Funktionen

Mit Funktionen musst du einen Ablauf nur einmal programmieren. Danach kannst du diesen Ablauf in eine Funktion verschachteln. Immer wenn du die Funktion aufrufst, wird der Ablauf innerhalb der Funktion ausgeführt.

Verwende Funktionen für die erstellten Drehungen und fürs geradeaus fahren (z.B. Drehung90Links mit dem Parameter Geschwindigkeit oder FahreDistanz mit den Parametern Distanz und Geschwindigkeit).

3.4 Parkour

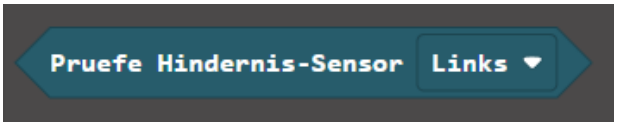
Fahre den vorgegebenen Parkour ab. Starte das Fahrzeug mit Taster A. Passe die Parameter für die Distanz und die Drehung so an, dass du so genau wie möglich auf der Strecke bleibst.

4 Automatisches Fahren

Lass den Roboter selbständig fahren. Schaue, dass er nirgendwo reinfährt. Verwende einige der Funktionen von der vorherigen Aufgabe!

4.1 Hinderniserkennung

Fahre bei Tastendruck A mit dem Auto gerade aus, bis ein Hindernissensor ein Hindernis erkennt. Stoppe das Fahrzeug.



Teste das Anhalten, wenn du gerade und / schräg auf eine Wand fährst. Hält der Roboter immer rechtzeitig?

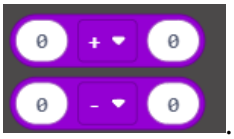
_____

Ergänze Aufgabe 1 mit dem Ultraschallsensor. Schaue, dass der Roboter nicht näher als 5 cm zu einem Objekt kommt. Achtung, der Ultraschallsensor misst die Distanz in cm.

Der Ultraschallsensor ist relativ ungenau. Welche Distanz misst der Ultraschallsensor, wenn das Objekt 5 cm vom Ultraschallsensor entfernt ist?

_____

Mit einer Variablen und Mathefunktionen kannst du, die gemessene Distanz korrigieren



4.2 Programmablauf

Überlege dir, was der Roboter machen soll, wenn ein Objekt vor dem Roboter ist.

Objekt Links: _____

Objekt Rechts: _____

Objekt Vorne: _____

Besprich deine Idee mit jemandem neben dir. Programmiere danach deine Idee und überprüfe sie.

4.3 Challenge *

Passe die Geschwindigkeit des Roboters in Abhängigkeit der Distanz zum nächsten Objekt an. Je näher der Roboter am Objekt ist, desto langsamer fährt der Roboter

5 Folge der Line

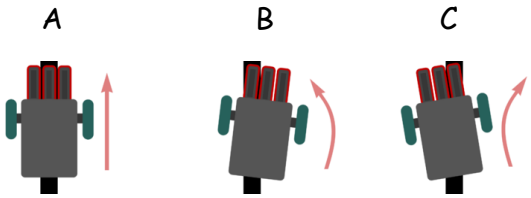
Programmiere das Fahrzeug so, dass es einer Linie folgt! Nutze dabei die 3 Liniensensoren.

5.1 Linen-Sensoren

Die Liniensensoren funktionieren gleich wie die Objekterkennungssensoren. Sie bestehen aus einem Infrarot-LED und einem Infrarotdetektor. Die LED strahlt das Licht aus. Wenn ein dunkles Objekt unterhalb des Sensors ist, wird wenig Licht reflektiert. Wenn ein helles Objekt unterhalb des Sensors ist, wird viel Licht reflektiert, welches vom Detektor erkannt wird. Die Line muss also das Licht stärker (dunkle Line) oder schwächer (helle Line) absorbieren als der Boden.

5.2 Funktionsweise

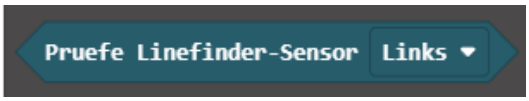
3 Sensoren sind am Boden des Fahrzeugs montiert:



A) Der mittlere Sensor erkennt dunkel: Fahre vorwärts
B) Der linke Sensor erkennt dunkel: Fahre nach _____
C) Der rechte Sensor erkennt dunkel: Fahre nach _____

5.3 Programmieraufgabe

Programmiere den Roboter so, dass er der nach Tastendruck der Line folgt. Nutze die folgende Funktion:



5.4 Challenge

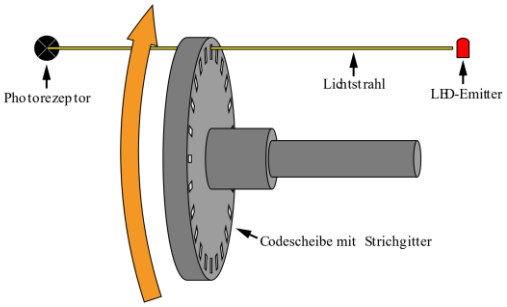
Erweitere das Programm so, dass der Roboter anhält, sobald ein Objekt im Weg ist.

6 Parkour mit Radumdrehungen

Fahre mit dem Roboter den vorgegeben Parkour automatisch ab. Nutze dabei die Drehencoder (Geschwindigkeitssensoren) des Roboters

6.1 Drehencoder

Mit dem Drehencoder kann die Umdrehung des Rades gemessen werden. Der Sensor besteht aus einem Rad mit 20 Löcher und einem Infrarot LED und Infrarotdetektor. Wenn das Rad dreht, wird immer wieder das Licht detektiert und unterbrochen. Wenn man 20-mal das Licht sieht, dann ist das Rad genau einmal vollständig gedreht.



6.2 Flankenerkennung

Für die Erkennung der Schritte des Drehencoders wird eine Flankendetektion benötigt. Durch die Flankenerkennung wird ein Ereignis (z.B. ein Tasterdruck) nur einmal bei der Betätigung ausgeführt. Eine Flankendetektion erkennt, wenn der Zustand sich zum Beispiel vom Taster A ändert. Also:

- Der Taster ist nicht gedrückt.
- Der Taster wird gedrückt. Der Übergang ist die Flanke.

Versuche eine Flankendetektion selbst zu machen. Nutze dabei:

im Dauerhaft Block in Kombination mit:

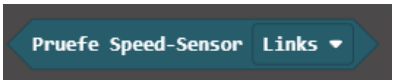
1. Zeige ein Symbol auf dem Bildschirm, sobald du den Taster A geklickt hast. (positive Flanke)
2. Lösche das Symbol auf dem Bildschirm, wenn du den Taster loslässt (negative Flanke)
3. Mache alles im Dauerhaft-Block

Hinweise: Für die Flankendetektion benötigst du eine Variable, welche den alten Zustand des Tasters abspeichert. Du musst den alten Zustand des Tasters mit dem aktuellen Zustand vergleichen. Wenn sich der Zustand geändert hat, dann hast du eine Flanke erkannt.

6.3 Distanzmessung

Miss die gefahrene Distanz. Dazu benötigst du:

1. Den Raddurchmesser: d=_____
2. Anzahl der Flanken pro Umdrehung: n=_____
3. Radumfang: U=π ·d=_____
4. Distanz pro Flanke: s=U/n=_____
5. Flankendetektion aus Aufgabe 1



gibt dir ein «1», wenn das Licht durch den Spalt kommt, und ein «0», wenn der Lichtstrahl unterbrochen ist.

Hinweis: Zum Testen der Distanzmessung: Stosse das Fahrzeug 30 cm weit. Zeige die Distanz auf dem Bildschirm an.

Fahre danach genau 50 cm weit. Miss die Distanz mit den Drehencodern.

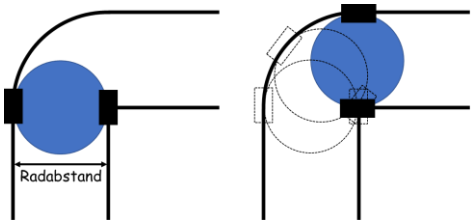
Hinweis: Fahre nicht zu schnell, sonst kommt die Messung der Schritte nicht nach (gutes Tempo ca. 20 % Geschwindigkeit)

6.4 Drehung

Drehe das Fahrzeug um 90° nach links und nach rechts. Miss die Drehung über die Drehencoder. Das eine Rad bleibt stehen, das andere dreht. Wie weit muss sich das drehende Rad drehen?

_____

Hinweis: Auf der Skizze siehst du, dass das äussere Rad $\frac{1}{4}$ Kreis fährt mit dem Radabstand $r = 15\ cm$ als Radius



6.5 Parkour

Fahre den vorgegebenen Parkour mit Hilfe der Drehencoder ab!

Hinweis: Verwende Funktionen für die erstellten Drehungen und fürs geradeaus fahren (z.B. DrehungLinks mit den Parametern Winkel und Geschwindigkeit oder FahreDistanz mit den Parametern Distanz, Geschwindigkeit)

7 Labyrinth

Fahre mit dem Roboter durch das Labyrinth! Das Labyrinth hat auf beiden Seiten Wände und teilweise Hindernisse. Erkenne die Hindernisse mit den Sensoren (Ultraschall und Objekterkennungssensor) und Entscheide, was der Roboter machen soll.

7.1 Distanzmessungen

Verwende das Servo für einen Rundumblick des Roboters. Das Servo kann den Ultraschallwandler nach links und rechts bewegen. Dadurch erkennst du, ob du nach links oder rechts fahren kannst.

1. Mach eine Funktion, bei welcher du den Winkel des Servos vorgibst, und danach eine Distanzmessung durchführst.
2. Stelle das Servo auf den Winkel 90° (nach vorne). Zeige die Distanz auf dem Bildschirm an.
3. Stelle das Servo auf den Winkel 180° (nach links). Zeige die Distanz auf dem Bildschirm an.
4. Stelle das Servo auf den Winkel 90° (nach rechts). Zeige die Distanz auf dem Bildschirm an.
5. Führe die Schritte 2,3,4 hintereinander aus, mache immer 2 Sekunden Pause nach der Anzeige. Überprüfe die Resultate beim Labyrinth.

Hinweis: Warte nach der Drehung mit dem Servo 1000 ms. Führe erst danach die Messung durch. Dadurch hat der Roboter Zeit, das Servo zu drehen.

7.2 Objekterkennung

1. Starte das Programm mit Taster A.
2. Fahre so lange geradeaus, bis ein Objekt vor dir ist (20 cm Distanz).
3. Verwende die Funktion von Aufgabe 1 um zu schauen, wo es am meisten Platz hat.
4. Fahre in die Richtung mit dem meisten Platz
5. Überlege dir, was du machst, wenn links, rechts und vorne kein Platz ist:

   _____
6. Verwende zusätzlich die Objekterkennungssensoren, falls eine Wand zu nahe ist.

Teste das Programm, indem du auf eine Wand oder in eine Ecke fährst.

7.3 Fahre durch das Labyrinth

Starte den micro:bit im Labyrinth!