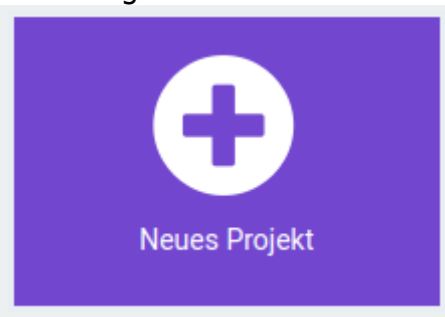
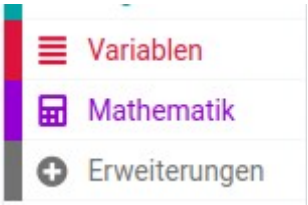
micro:bit Smart Robot Car

In dieser Einheit werden wir mit dem micro:bit ein kleines Roboterauto mittels verschiedener Sensoren steuern. Für jede der einzelnen Aufgaben ist ein kleiner Hindernis-Parcours aufgebaut. Mit deiner fertigen Lösung wird das Auto diesen jeweils bewältigen können.

Das Auto verfügt über einen eigenen Befehlssatz welchen wir zunächst als Erweiterung dem Makecode-Editor hinzufügen müssen. Erzeuge zunächst im Editor ein neues Projekt. Dieses wird als Basis für alle kommenden Aufgaben dienen.



Klicke dann im Menü auf der linken Seite auf „Erweiterungen“:



Auf der anschließenden Seite musst du nun nach folgender URL suchen:

https://github.com/keyestudio2019/MiniCar

Durch einfaches Anklicken der aufscheinenden Erweiterung wird der Befehlssatz mit dem Namen „MiniCar“ dem Editor hinzugefügt:

Jetzt bist du soweit, mit der ersten Aufgabe zu starten!

1. Manuelle Programmierung des Motors

In der ersten Aufgabe geht es zunächst darum, die Bewegungen unseres Fahrzeugs manuell zu programmieren, also zum Beispiel ein Feld vor zu fahren, sich nach links zu drehen, wieder ein Feld vorwärts zu fahren und so weiter.

Der Roboter verfügt über zwei getrennt voneinander ansteuerbare Motoren mit denen er seine zwei Reifen separat bewegen kann. Der Block zur Steuerung im Makecode-Editor ist der folgende (zu finden unter dem Befehlssatz *MiniCar*):

Wie du siehst bietet der Block insgesamt drei Einstellungs-möglichkeiten:

- die Auswahl des Motors (linker Motor A oder rechter Motor B)
- seine Drehrichtung (vorwärts/forward, rückwärts/backward)
- seine Geschwindigkeit (speed)

Auf Dauer ist dieser Befehl etwas zu kompliziert, wenn wir unser Auto zum Beispiel einfach nur kurz vorwärts fahren oder nach links drehen möchten. Wir werden uns deshalb als erstes einige Funktionen schreiben, welche die Steuerung für uns übernehmen werden. Funktionen kannst du dir als von uns selbst definierte Blöcke vorstellen, die wir beliebig oft aufrufen können. Die Möglichkeit zur Definition von Funktionen findest du im Makecode-Editor unter *Fortgeschritten → Funktionen*.

Definiere mithilfe dieses Befehls vier Funktionen *fahreVorwärts*, *fahreRückwärts*, *dreheLinks* und *dreheRechts*. Probiere aus, wie du die einzelnen Funktionen befüllen musst, damit der Roboter dauerhaft vorwärts bzw. rückwärts fährt und sich permanent nach links bzw. rechts im Kreis dreht. Du kannst die entsprechende Funktion hierzu einfach im „dauerhaft“-Block einfügen, das Auto vom USB-Kabel trennen und es anschließend starten. Achte aber bitte bei deinen Versuchen unbedingt darauf, dass das Auto nicht vom Tisch fällt.

Wir werden außerdem eine zusätzliche Funktion *stopp* benötigen, die die Geschwindigkeit beider Motoren auf 0 setzt.

Diese insgesamt fünf Funktionen werden wir auch in den nachfolgenden Aufgaben immer wieder nutzen.

Mithilfe der Funktion *stopp* können wir die Bewegung des Roboters jederzeit beenden. Damit wird es nun möglich, den Roboter nur so lange zum Beispiel links drehen zu lassen, bis er sich genau um 90 Grad gedreht hat. Die Idee ist, zunächst die jeweilige Funktion (*dreheLinks*) aufzurufen, den micro:bit anschließend für die richtige Zeitspanne pausieren zu lassen (*pausiere*) und anschließend die Funktion *stopp* aufzurufen, um den Roboter anzuhalten. Pausieren kannst du den micro:bit mit dem folgenden Befehl:

Erstelle auf diese Weise noch einmal vier Funkionen *links90*, *rechts90*, *vor1Feld* und *zurück1Feld*. Wie weit der Roboter für ein einzelnes Feld vorfahren muss (und wie lange du entsprechend *pausiere* aufrufen musst) kannst du einfach am aufgebauten Parcours ausprobieren.

Mit diesen letzten Funktionen solltest du nun in der Lage sein, ein einfaches Programm zu schreiben um den ersten Parcours zu bewältigen. Hänge dazu einfach die notwendigen Funktionsaufrufe/Blöcke (*links90*, *rechts90*, *vor1Feld*) in der richtigen Reihenfolge im „beim Start“-Block ein. Das Auto wird die vorgegebene Befehlskette dann automatisch abfahren sobald du ihn startest.

2. Line Tracking-Sensoren

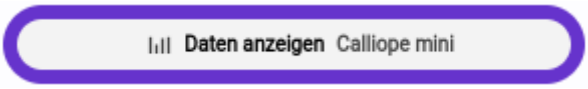
Nachdem du den ersten Parcours bewältigt hast, solltest du zur Vorbereitung für Aufgabe 2 zunächst deinen bisherigen „beim Start“-Block leeren. Lösche aber NICHT die von dir angelegten Hilfsfunktionen selbst, denn diese werden wir weiterhin brauchen! Im Notfall kannst du im Editor auch jede deiner Aktionen Schritt für Schritt rückgängig machen, falls doch einmal etwas schief gehen sollte:

Der Parcours zur zweiten Aufgabe besteht nur aus einer schwarzen Linie (englisch *line*). Der Robot Car verfügt an seiner Unterseite nämlich über zwei Line-Tracking-Sensoren, die ihn diese Linie erkennen lassen. Diese befinden sich auf der Unterseite direkt hinter dem Vorderrad und sind jeweils mit einem *Potentiometer* an der Oberseite verbunden.

Mithilfe des Potentiometers kann die Empfindlichkeit des jeweiligen Sensors mittels eines Schraubenziehers eingestellt werden. Normalerweise sollte dieser aber von den Mentoren schon richtig eingestellt worden sein.

Um den Sensor ein wenig kennenzulernen kannst du den folgenden Testcode deinem Programm hinzufügen (die dunkelblau hinterlegten Blöcke findest du unter *Fortgeschritten → Seriell*, die Variable *Line Tracking* unter *MiniCar*).

Das Programm musst du starten, während der Roboter mit dem USB-Kabel verbunden ist. Der Editor wird dir nun anzeigen, dass Daten zur Anzeige zur Verfügung stehen. Klicke auf den folgenden Button (wichtig ist, dass hier NICHT „Simulator“ steht).



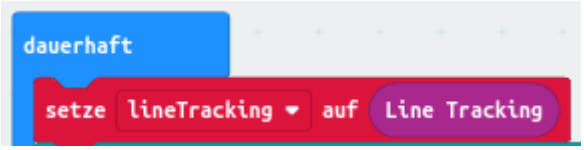
Dir werden nun kontinuierlich die Werte der Variable *Line Tracking* auf dem Bildschirm ausgegeben. Verschiebe das Auto über die folgende Testlinie und beobachte, wie sich die Werte verändern. Am besten du trägst den jeweiligen Wert in die folgende Tabelle ein:

Zustand	Sensorwert
Beide Sensoren über der Linie	
Nur rechter Sensor über der Linie	
Nur linker Sensor über der Linie	
Kein Sensor über der Linie	

Nachdem du herausgefunden hast, welcher Sensorwert für welchen Zustand steht, versuche ein Programm zu schreiben, welches den Roboter permanent der schwarzen Linie folgen lässt, bis diese endet: Die Idee sollte sein, dass

- das Auto geradeaus fährt solange beide Sensoren die Linie sehen
- in die entsprechende Richtung gedreht wird wenn ein Sensor die Linie verliert
- und das Auto anhält wenn beide Sensoren die Linie verloren haben.

Verwende hierzu die Hilfsfunktionen *fahreVorwärts*, *dreheLinks* und *dreheRechts* aus Aufgabe 1. Mit folgendem Code kannst du den Sensorwert wiederkehrend in eine Variable lesen um ihn anschließend vergleichen und korrekt regieren zu können.



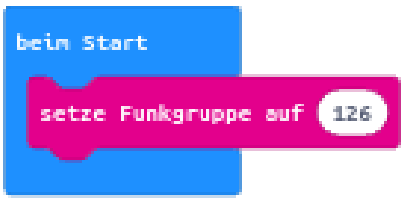
Soabld du fertig bist kannst du dich am aufgebauten zweiten Hindernis versuchen!

3. Steuerung mittels Bluetooth

In diesem Teil wollen wir versuchen, unser Auto von einem zweiten micro:bit fernsteuern zu lassen. Dies soll über die in den micro:bit verbaute Bluetooth-Schnittstelle geschehen. Der micro:bit des Roboters wird somit verschiedene Nachrichten für die Steuerung empfangen, der andere micro:bit wird diese Befehle als Fernbedienung senden.

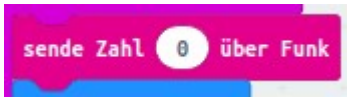
Für die Programmierung des Roboters kannst du einfach dein bestehendes Programm erweitern bzw. adaptieren. Insbesondere wirst du die Funktionen *fahreVorwärts*, *fahreRückwärts*, *dreheLinks* und *dreheRechts* benötigen. Für die Programmierung des micro:bit welcher als Fernbedienung fungiert werden wir ein neues zweites Programm erstellen.

Damit die zwei micro:bits miteinander kommunizieren können, müssen sich diese in der selben Funkgruppe befinden. Diese sollte man direkt zum Start setzen. Insgesamt stehen 256 Kanäle zur Verfügung. Damit sich die verschiedenen Fernbedienungen unserer Gruppe nicht in die Quere kommen, sollte hier jeder ein andere Zahl wählen. Du könntest die Zahl zum Beispiel von deinem Geburtstag ausgehend setzen: Jemand der am 16. Dezember Geburtstag hat könnte als Kanal zum Beispiel die 126 wählen (12 für den Monat, die 6 als letzte Ziffer des Tages).

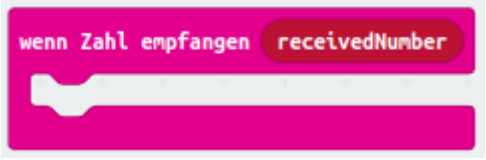


Dieser Codeblock wird in beiden Programmen benötigt.

Unsere Fernbedienung soll folgendermaßen funktionieren: Wird der micro:bit nach vorn gekippt (d.h. das micro:bit Logo zeigt nach unten), soll unser Auto vorwärts fahren, neigt sich das Logo nach links soll sich das Auto nach links drehen und so weiter. Vielleicht möchtest du die aktuelle Kipprichtung auch irgendwie mithilfe des LED-Displays der Fernbedienung anzeigen, etwa in Form eines Pfeiles. Jedenfalls sollte der micro:bit einen Befehl über Bluetooth schicken wenn er in eine bestimmte Richtung gekippt wird, z.B. eine 0 für vorwärts, 1 für links und so weiter.



Diese Zahl kann am micro:bit des Roboters empfangen werden und sollte dann eine entsprechende Bewegung auslösen, je nachdem welche Zahl empfangen wurde. Für den Empfang einer Bluetooth-Nachricht dient folgender Befehl:

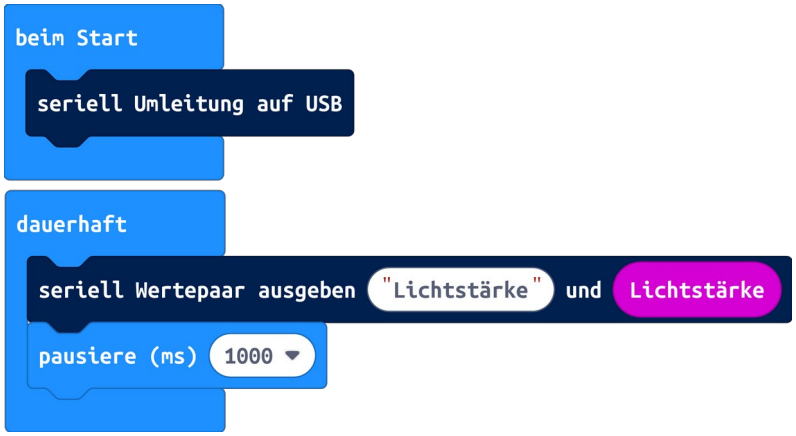


Wenn du fertig bist, kannst du versuchen ob du den Parcours aus Aufgabe 1 nun auch ferngesteuert überwinden kannst!

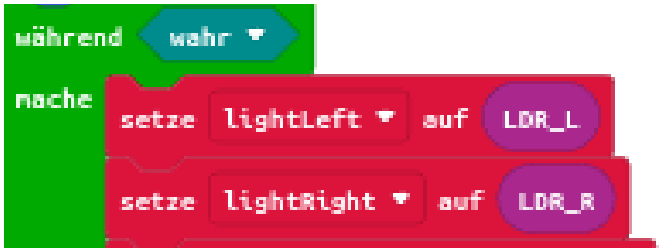
4. Photowiderstände

Unser Robot Car verfügt an der oberen Vorderseite über zwei lichtempfindliche Sensoren. Wir möchten diese im gelben Parcours dazu nutzen, den Robotor mithilfe einer Taschenlampe durch das Hindernis zu lotsen. Der Roboter soll dabei stets der Richtung des Lichts folgen.

Wie schon in Aufgabe 2 kannst du wieder ein kleines Testprogramm schreiben, um die Photowiderstände zu testen. Welche Werte werden ausgegeben, wenn die Taschenlampe die Sensoren beleuchtet und welche, wenn nur das normale Raumlicht vorhanden ist?



Nun kannst du den folgenden Code nutzen und so ausbauen, dass der Roboter mithilfe der Funktionen *fahreVorwärts*, *dreheLinks* und *dreheRechts* aus Aufgabe 1 stets der Richtung des Lichts der Taschenlampe folgt bzw. anhält und die Schleife abbricht, wenn er die Taschenlampe nicht mehr wahrnimmt.



Schaffst du es mithilfe der Taschenlampe, den Roboter durch das Hindernis zu Aufgabe 1 zu lotsen?