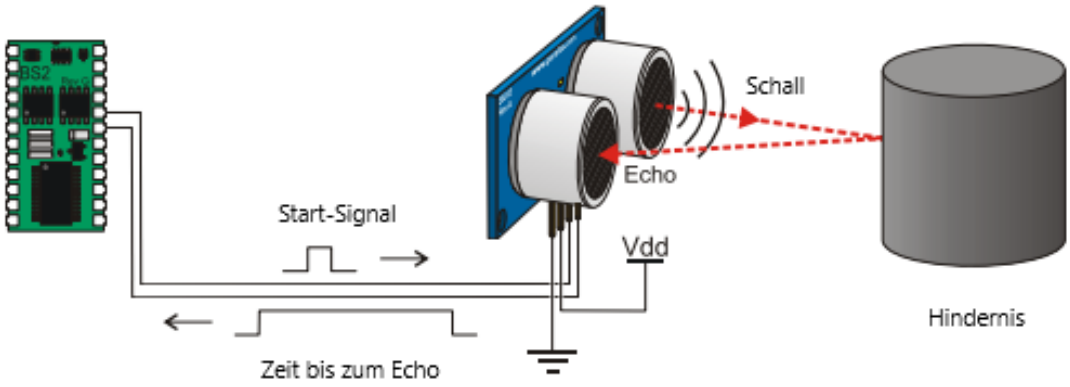
Ultraschallsensor (Parksensor oder Abstandssensor beim Auto):

Misst Entfernungen von 3cm bis 2m: die beiden runden „Augen“ sind ein Sender und ein Empfänger (wie bei uns Menschen der Mund und das Ohr).

Die ausgesendeten Schallwellen werden von Hindernissen reflektiert und laufen zum Empfänger zurück (wie ein Echo). Der Sensor errechnet aus der Zeit die Entfernung, weil der Schall ca. 3 Sekunden für einen Kilometer (1km) benötigt.



Wir hören die ausgesendeten Schallwellen nicht, weil die Frequenz zu hoch ist. Fledermäuse verwenden dasselbe Prinzip, weil sie meist im Dunklen fliegen.

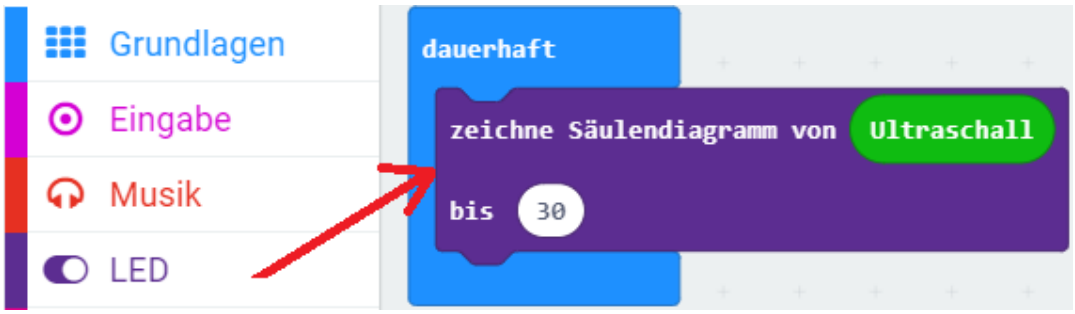
Wir können die Entfernung mit dem grünen Block „Ultraschall“ aus der MikRobot-Bibliothek messen:

Der Block „Ultraschall“ gibt die Entfernung in Zentimeter an. Weil er aber genauer als Zentimeter misst, gibt er eine Zahl mit einem Komma und viele Stellen hinter dem Komma aus. Auf dem micro:bit läuft dann immer eine lange Zahl über die Anzeige – das ist schwierig zu lesen.

Um nur die Zahl vor dem Komma anzuzeigen, können wir die „Runden“-Funktion aus der Bibliothek „Mathematik“ verwenden.

Eine andere Art der Anzeige ist das Diagramm: statt einer Zahl wird die Entfernung mit Leuchtpunkten (mit den LEDs auf dem micro:bit) angezeigt.

Wenn sich ein Messwert schnell verändert, dann können wir Menschen bei Leuchtpunkten oder einem Zeiger viel leichter den Überblick behalten. Eine Anzeige mit einer Zahl ist für uns schwieriger zu lesen und zu verstehen (deshalb macht der micro:bit bei Zahlen auch eine extra Pause, bevor er eine neue Zahl anzeigt – obwohl der Ultraschall-Sensor viele Messungen pro Sekunde macht).



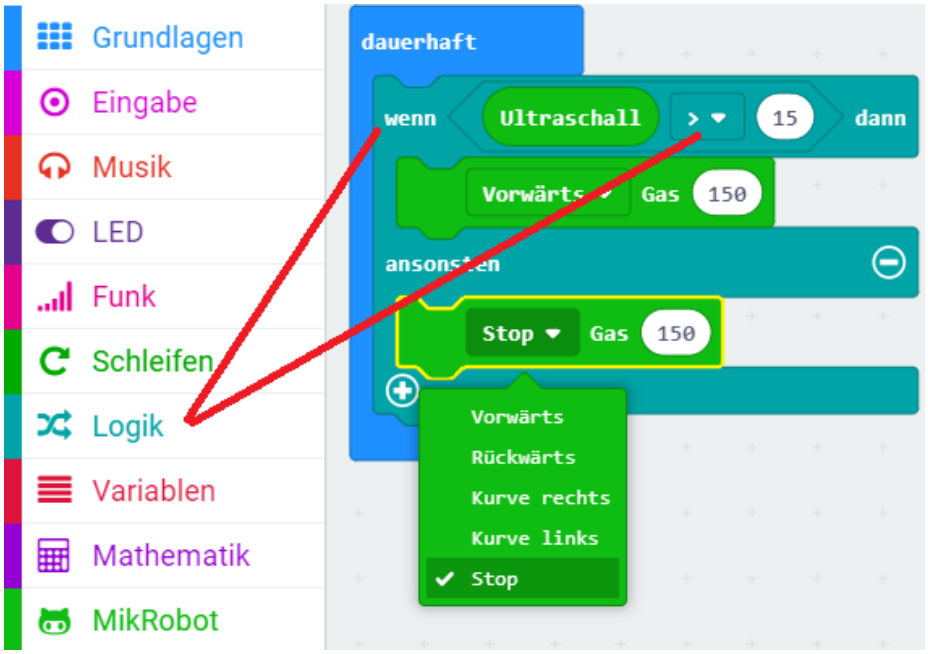
Mit der 2.ten Zahl beim Säulendiagramm (z.B. „bis 30“) stellen wir ein, bei welchem Ultraschall-Entfernungswert alle Leuchtpunkte leuchten. Wenn man nur Entfernungen bis 10cm messen will, dann stellt man „bis 10“ ein.

Auf unserem Roboter „Mik“ können wir den Ultraschall-Sensor zur Hindernis-Erkennung einsetzen. Solange der Abstand groß genug ist, lassen wir Mik geradeaus fahren.

Wenn aber der Sensor ein Hindernis erkennt, das näher als 15cm vor uns steht, dann soll Mik stehen bleiben. Was passiert, wenn das Hindernis entfernt wird? Probiere es aus…

Die Blöcke für die „Wenn…dann…ansonsten“-Bedingung und den Vergleich „>“ (größer als) findest du in der Bibliothek „Logik“.

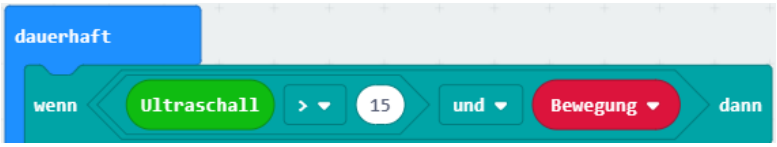
Der „Stop“-Befehl ist der „Vorwärts Gas …“-Block, den du mit dem kleinen weißen Pfeil neben „Vorwärts“ zu einem „Stop“ umkonfigurieren kannst.



Roboter-Experten erweitern das Programm:

1. Erst wenn der Knopf A gedrückt wird, soll Mik überhaupt fahren
2. Wenn der Knopf B gedrückt wird, soll Mik sowieso immer stehen bleiben
3. Beim Programmstart soll auf der Anzeige zuerst ein Name oder ein Symbol

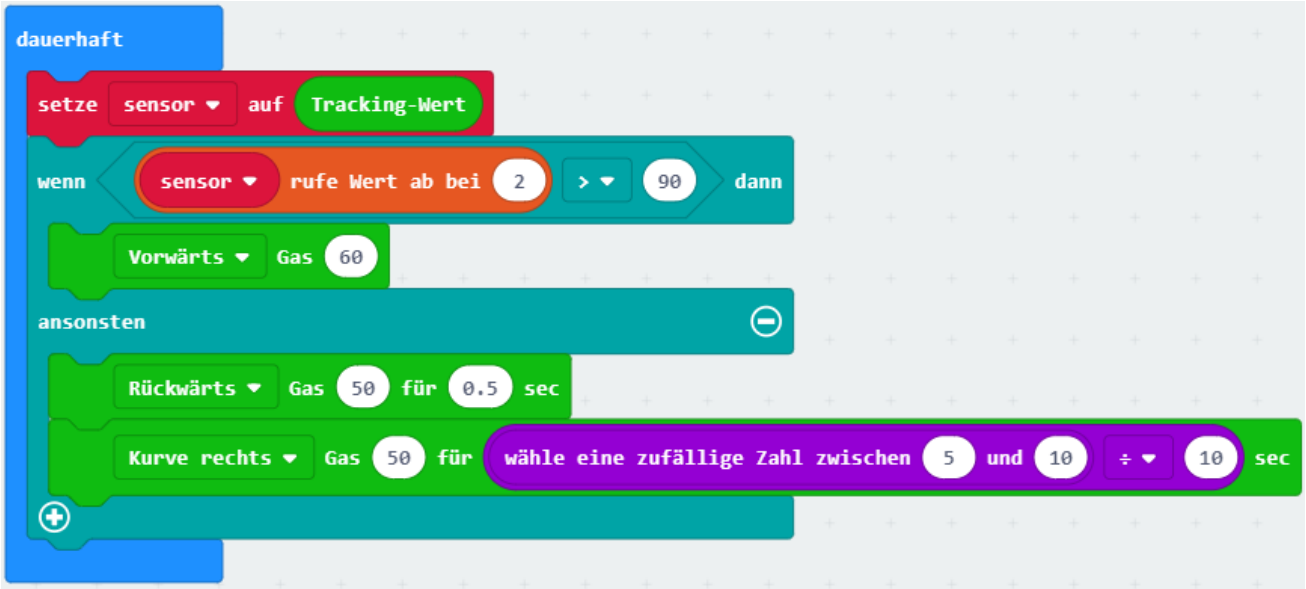
(Tipp: verwende eine Variable (z.B. „Bewegung“), die auf 1 gesetzt wird, wenn Knopf A gedrückt wird. Wenn Knopf B gedrückt wird, wird diese Variable auf 0 gesetzt. In der „Wenn…dann“-Abfrage musst du dann noch einen Logik-Block „Und“ einbauen):



Wir können auch programmieren, dass der Roboter nicht stehen bleibt, sondern eine Kurve macht:

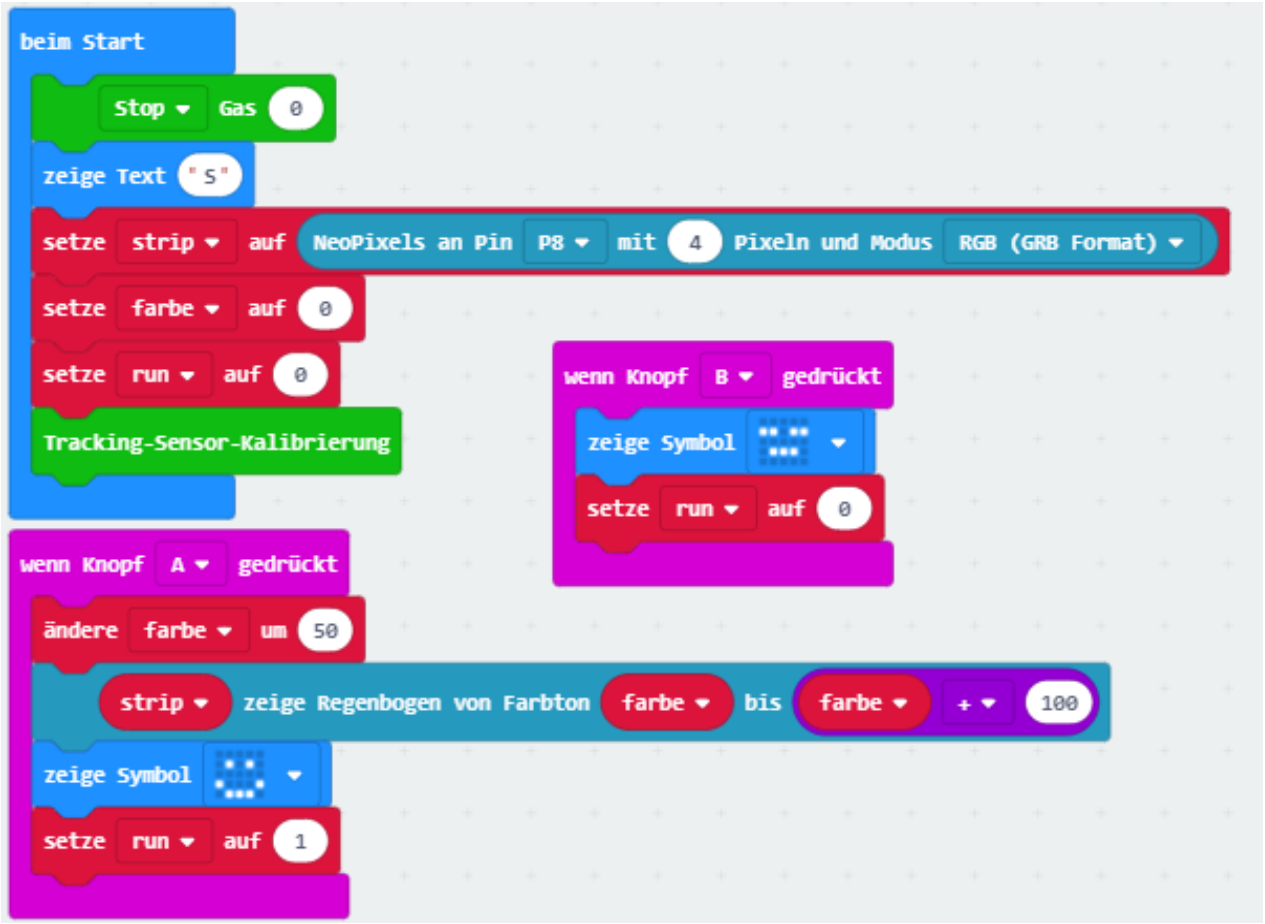
Weitere Programm-Ideen:

Rasenmäher-Roboter: Wir lassen „Mik“ nur innerhalb einer Fläche fahren, die mit einem schwarzen Klebeband eingegrenzt ist. Immer wenn er den Rand erkennt, dreht er sich um und fährt dann weiter.



Linienfolger mit Hindernis-Erkennung und Geschwindigkeits-Regelung:

Mit diesem Programm fährt „Mik“ immer der Linie nach. Die Knöpfe A und B steuern, ob „Mik“ fährt (lachender Smiley) oder schläft (schlafender Smiley). Beim Programmstart werden die bunten Lichter (Neopixels) programmiert und bei jedem Tastendruck von Knopf A ändern sich die Farben. Beim Start werden auch die Tracking-Sensoren neu eingestellt (kalibriert): dabei dreht sich „Mik“ einmal nach rechts und nach links, damit alle 5 Tracking-Sensoren die schwarze Linie sehen.



In der Schleife „dauerhaft“ wird die Position der Linie gemessen (ganz links ist der Wert 0, ganz rechts ist der Wert ~4000 – von oben gesehen). Damit wir einen guten Wert für die Steuerung der Motoren erhalten, müssen wir von der „Position der Linie“ die Zahl 2000 abziehen. Damit erhalten wir eine 0, wenn Mik die schwarze Linie genau in der Mitte sieht -> die Variable für die Differenz bei der Motorgeschwindigkeit (Variable „dif“) wird 0 und „Mik“ wird geradeaus fahren.

Weil wir die Motoren mit Zahlen bis 255 ansteuern können, müssen wir den Wert für die Differenz noch dividieren (2000:12=167). Dabei wollen wir keinen Rest (keine Kommastelle) und verwenden dafür den Block „Quadratwurzel“ aus der Mathematik-Bibliothek (den wir auf „Ganze Zahl %“ konfigurieren). Der Wert 12 funktioniert ganz gut (du kannst auch andere, ähnliche Werte verwenden).

Die Geschwindigkeit „Speed“ nehmen wir vom Ultraschallsensor: wenn wir mehr als 10cm weit kein Hindernis sehen, dann fahren wir mit der Geschwindigkeit 100 (deshalb „Ultraschall x 10“). Damit Mik nie schneller als 100 fährt, gibt es den Block „Minimal von .. und ..“ (in der Bibliothek Mathematik).

In der „Wenn…dann“-Bedingung kontrollieren wir, ob die Variable „run“ auf 1 steht UND die beiden Abstandssensoren links ODER rechts NICHT aktiv sind.

Wenn das so ist, dann kontrollieren wir mit einer 2.ten „Wenn…dann“-Bedingung, ob die berechnete Differenz größer 0 (oder „positiv“) ist: dann schalten wir auf Motor 1 die Geschwindigkeit „speed“ und auf Motor 2 den Wert „speed“ minus „dif“.

Damit dreht der Motor 1 schneller als Motor 2 und „Mik“ fährt nach rechts. Damit kommt die Linie von „zu weit rechts“ wieder zurück in die Mitte. Wenn die Linie wieder in der Mitte ist, wird auch die Variable dif wieder 0 und Mik fährt geradeaus.

Damit „dif“ nicht zu groß wird, beschränken wir es zwischen 0 und 100.

Wenn dif kleiner 0 ist („negative“ Zahl), dann schalten wir auf Motor 1 die Geschwindigkeit „speed“ plus minus dif: das bedeutet, dass von „speed“ der Wert „dif“ abgezogen wird -> der Motor 1 dreht langsamer als Motor 2 -> Linkskurve!