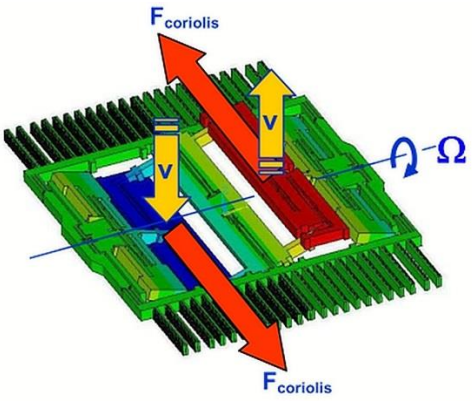
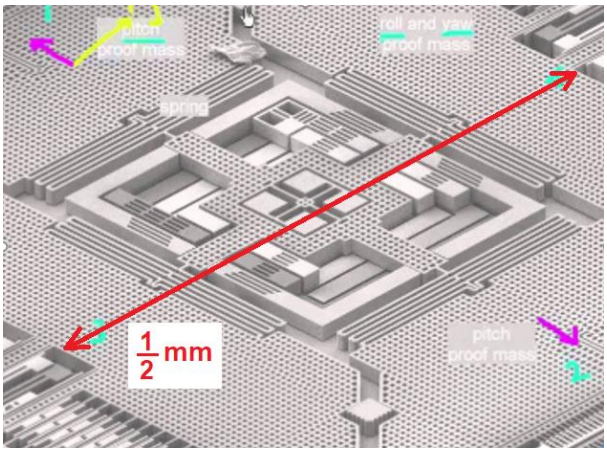
Bewegungssensor:

Ist auch in jedem Smartphone eingebaut, um Drehungen erkennen zu können (Umschaltung zwischen Hoch- und Querformat). Wie funktioniert dieser komplizierte Kreisel-Sensor?

Bei einem drehenden Kreisel entstehen Kräfte, die das Umfallen verhindern. Seine Drehachse bleibt stabil. Wenn man nun einen Kreisel frei beweglich aufhängt und die Halterung verdreht, dann entstehen Kräfte. Diese werden vom Kreisel-Sensor gemessen und aus der Kraft und der Zeit (je länger verdreht wird, desto größer ist der Drehwinkel) wird die Winkeländerung berechnet.

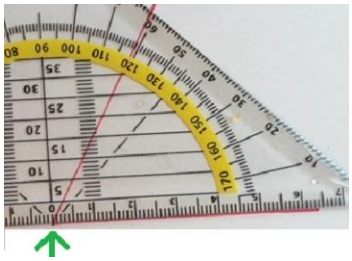
Die meisten Miniatur-Kreiselsensoren verwenden keinen Kreisel (der wäre zu groß und es ist kompliziert, die Kreiseldrehung immer aufrecht zu erhalten) sondern eine Art „Stimmgabel“, die dauernd schwingt (wie in einer Quarzuhr). Auch diese „Stimmgabel“ erzeugt beim Verdrehen eine kleine Kraft, die der Sensor messen kann. Die gesamte Mechanik im Sensor ist nur ca. einen halben Millimeter groß!!!

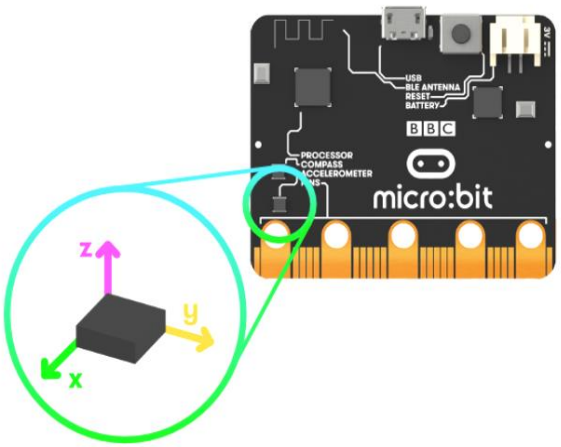




Wir haben schon im Kurs 9 bei der Funkfernsteuerung mit diesem Sensor im micro:bit gearbeitet und erinnern uns, dass es 3 Achsen gibt: x, y und z

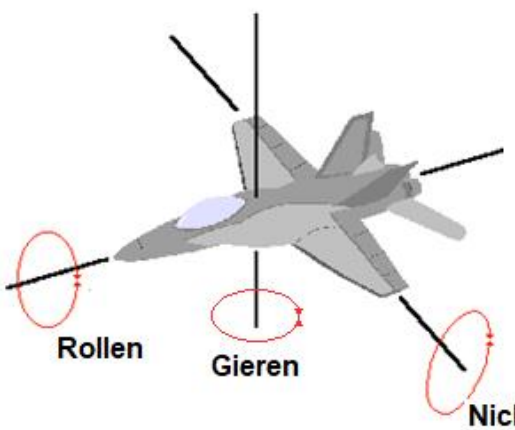
Der Kreisel-Sensor kann den Drehwinkel um diese Achsen auch in Grad (°) messen:

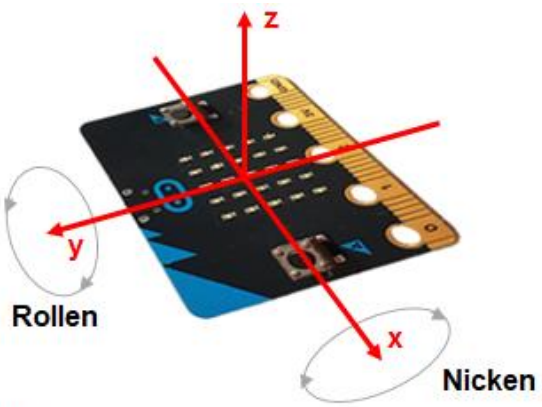


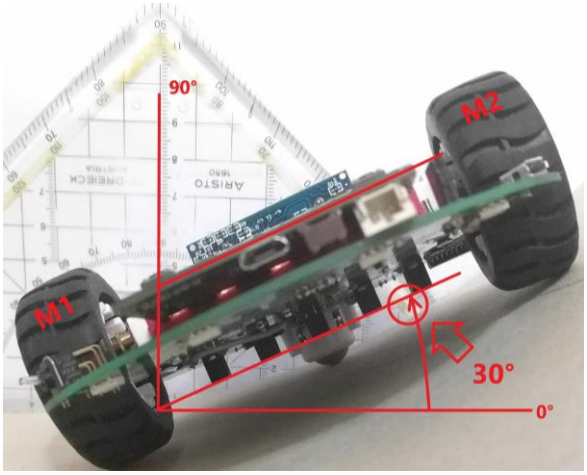


Bei Fahrzeugen (Flugzeug, Schiff, Auto…) nennt man die Bewegungen um diese 3 Achsen auch

- Nicken (x-Achse)
- Rollen (y-Achse)
- Gieren (z-Achse)





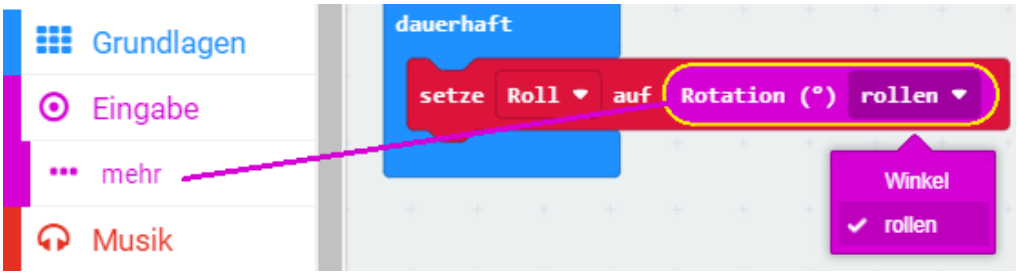


In Makecode können wir den Drehwinkel mit der Funktion „Rotation“ aus der Eingabe-Bibliothek (unter „…mehr“) messen. Wir müssen noch die richtige Achse auswählen:

- „Winkel“ bedeutet Nicken (ein Fehler in Makecode)
- „rollen“ bedeutet Rollen 😊

Wenn der Roboter MIK wie im Bild mit der linken Seite nach unten „gerollt“ ist, dann ist der Winkel positiv (+30°). Wenn die rechte Seite unten ist, ist der Winkel negativ (-30°).

Damit MIK im Bild wieder gerade steht, muss der Motor M2 schneller drehen, als Motor M1.



In unserem Programm müssen wir eine Variable „Roll“ anlegen (und eine Variable „Fahr“, damit unser Programm den Roboter auch wieder stoppen kann).

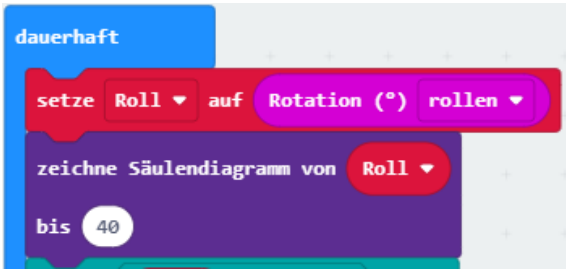
Beim Programmstart setzen wir „Fahr“ auf 0 und wenn Knopf A gedrückt wird, setzen wir „Fahr“ auf 300.

In der „dauerhaft“-Schleife werden wir mit einer „wenn… dann… ansonsten“-Abfrage überprüfen, ob „Fahr“ größer als 0 ist:

- Wenn „Fahr“ größer als 0 ist, dann setzen wir das Gas für die Motoren M1 und M2 (z.B. auf 40) und die Variable „Fahr“ um -1 ändern (-1 bedeutet, dass unser Programm „Fahr“ bei jedem Durchlauf der Schleife um eins kleiner macht). Wir müssen aber bei einem Motor etwas dazu rechnen (addieren +) und beim anderen Motor das abziehen (subtrahieren -).
- Wenn „Fahr“ bis auf 0 herunter gezählt hat („ansonsten“-Teil der Abfrage), dann soll MIK stehen bleiben

Damit wir auch sehen, welche Werte der Sensor misst, können wir die Variable „Roll“ mit den LEDs auf dem micro:bit anzeigen:

- Weil die Funktion „zeige Zahl“ aber eine Pause macht (weil wir Menschen zum Lesen und Erkennen einer Zahl ein bisschen Zeit brauchen), sollten wir die Funktion „Säulendiagramm“ aus der Bibliothek „LED“ verwenden (sonst reagiert der Roboter viel zu langsam):



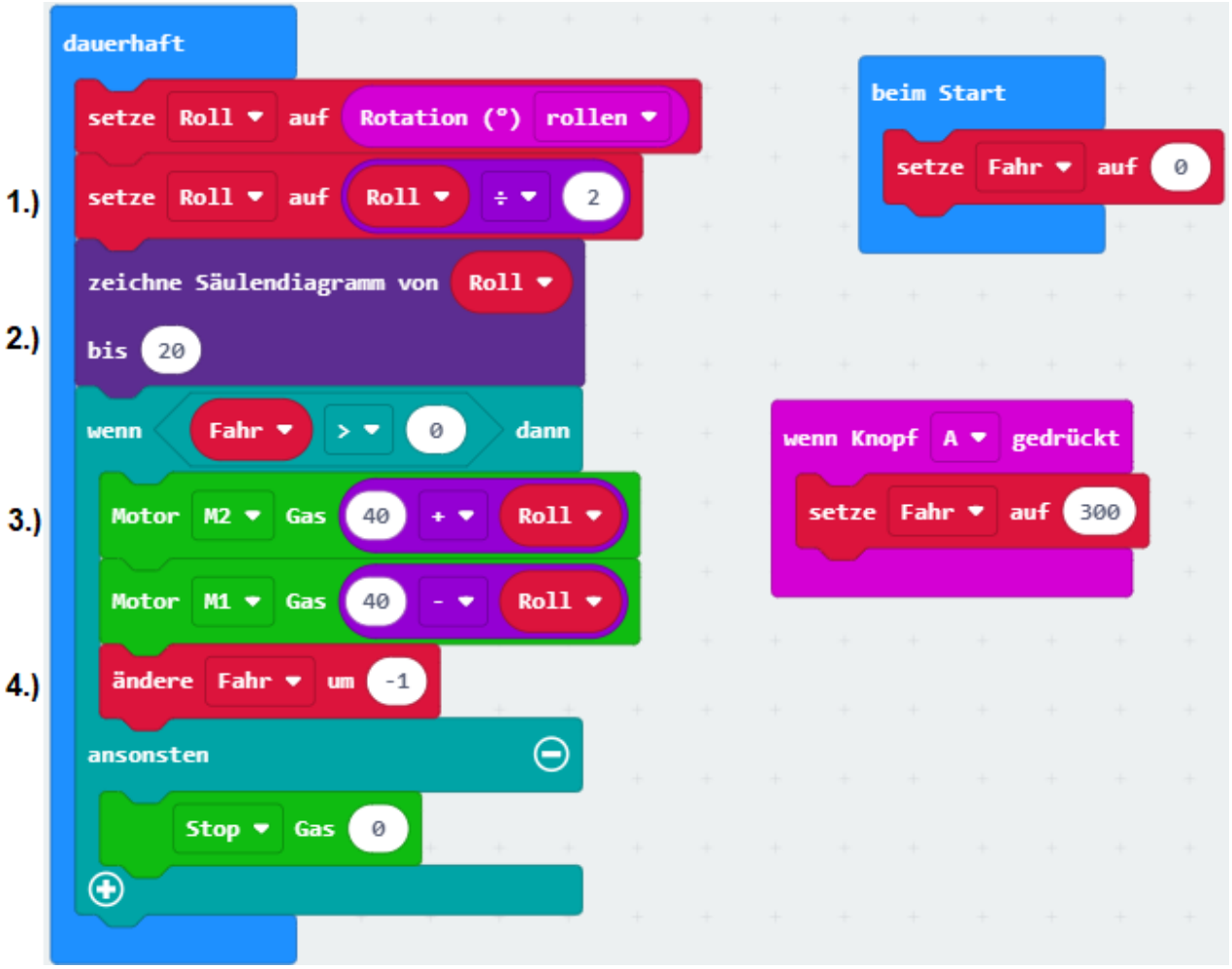
Die Angabe „bis 40“ legt fest, dass bei einem Winkel von 40° alle Lichter auf dem micro:bit leuchten.

Der Nachteil von „Säulendiagramm“ ist, dass negative Zahlen ganz gleich wie positive gezeichnet werden.

Ihr könnt also auch ein Testprogramm mit „zeige Zahl“ machen, um herauszufinden wie der Winkel positiv und wie er negativ wird.

Probiere das auch im Simulator aus, indem du das Bild vom micro:bit mit der Maus bewegst! Du wirst sehen, wie bei unterschiedlichen Drehwinkel mehr oder weniger Lichter leuchten.

So könnte das fertige Programm aussehen:



Erklärung:

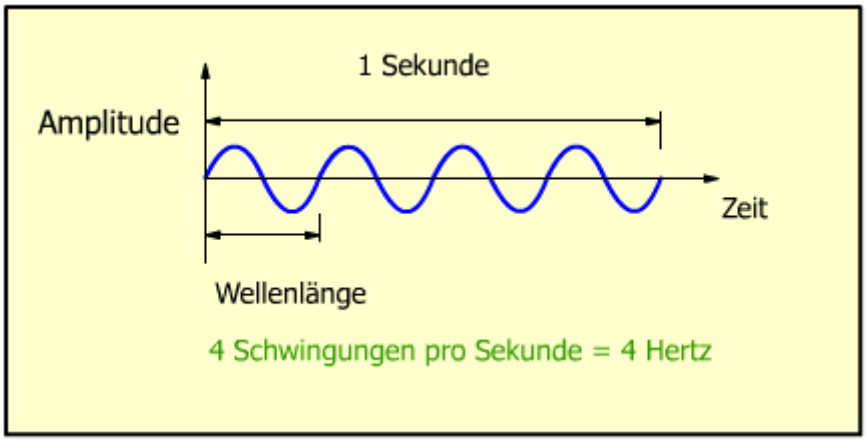
1.) Der Winkel, der in der Variablen „Roll“ gespeichert ist, wird durch 2 dividiert. Probiere auch andere Zahlen aus (z.B. dividiert durch 3), weil sich damit das Verhalten des Roboters ändert. Wird ein zu großer Wert errechnet, dann reagiert der Roboter zu stark und pendelt in der Röhre hin und her. Hier wird das Ergebnis der Division wieder in der Variablen „Roll“ gespeichert, es ist also nur mehr der halbe Winkel (das macht man beim Programmieren oft, dass Werte in Variablen angepasst werden und gleich wieder in derselben Variablen gespeichert werden).

2.) Wenn das Säulendiagramm „bis 20“ eingestellt ist, dann leuchten bei Roll = 20 alle Lichter (also bei einem Winkel von 40°)

3.) Wenn „Roll“ eine positive Zahl ist (Rad mit Motor M2 ist höher als M1), dann muss M2 schneller laufen, deshalb 40 + Roll. Motor M1 läuft dann langsamer: 40 – Roll
Ist „Roll“ negativ (Rad M1 ist höher als M2), dann muss M1 schneller laufen. Wenn der Winkel zum Beispiel -20° ist (damit ist Roll = -10), dann lautet die Rechnung für M1: 40 – (-10). 2mal minus ist in der Mathematik wieder ein Plus! M1 = 40 + 10 und M2 = 40 + (-10) = 40 – 10 (plus und minus ergeben minus).

4.) Damit der Roboter von alleine stehen bleibt, ändern wir die Variable „Fahr“ bei jedem Schleifen-Durchlauf um -1. Das bedeutet, dass wir immer die Zahl 1 abziehen, bis „Fahr“ wieder 0 ist.
Wenn der Roboter weiter fahren soll, dann kannst du die Zahl in „wenn Knopf A gedrückt“ größer machen.

Eine andere Möglichkeit unser Programm während des Ablaufs zu überprüfen, ist der Lautsprecher:
Mit der Funktion „Klingelton“ kann man einen Wert als Ton ausgeben. Ein Ton ist umso höher, je höher der Wert ist. Unser Ohr hört Töne von ca. 20Hz bis 15.000Hz (20 Schwingungen pro Sekunde bis 15.000/s).



Deshalb müssen wir den Wert von „Roll“ mit z.B. 100 multiplizieren. Ein Winkel von 2° (z.B. Roll=1) erzeugt dann einen Ton mit 100Hz (tiefer Ton), ein Winkel von 20° (Roll=10) einen Ton mit 1.000Hz (hoher Ton).

Weil negative Werte für den Ton nicht verwendet werden können und unser Winkel auch negativ sein kann, sollten wir die Funktion „Absolute von“ aus der Mathematik-Bibliothek verwenden (sonst hörst du bei negativen Winkeln nichts).
„Absolute“ bedeutet, dass das Minus-Vorzeichen bei einer negativen Zahl (z.B. -3) verschwindet und aus -3 eine 3 wird (ein +3 bleibt 3).

Welche anderen Ideen hast du, um ein Programm während der Ausführung zu kontrollieren?

Eine Kontrolle der Werte (Variablen) ist wichtig, um Fehler im Programm zu finden. Auch Profi-Programmierer sind oft auf der Suche nach Fehlern und überprüfen ihre Programme immer wieder. Sie nennen das „Debugging“ (das bedeutet: Entfernen von Fehlern/Bugs). Ein gutes Programm braucht viele Versuche und Überprüfungen – das ist ganz normal!

Die Profis verwenden dafür neben Lichtern (wie wir es mit dem Säulendiagramm gemacht haben) meistens eine Kabel-Verbindung zwischen dem Mikro-Computer (micro:bit) und dem großen Computer und lassen sich die Werte der Variablen anzeigen.

Dazu kannst du die Bibliothek „Seriell“ verwenden, wie wir das in Kurs 4 (Licht) bereits gemacht haben.
Auf dem PC benötigt man zum Empfangen ein eigenes Programm, das die Profis „Terminal“ oder „Konsole“ nennen. Dieses kann dann Text oder Zahlen übersichtlich und schnell anzeigen. Auch der Simulator kann eine Konsole anzeigen – probiere es aus!