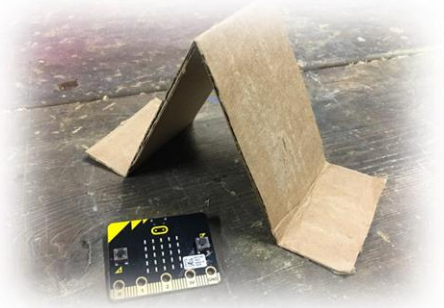
Roboter „Raupe“:

Mit dem micro:bit und dem Servo aus der letzten Übung können wir eine Raupe bauen.

Dazu schneiden wir ein Rechteck aus einem Karton, ein bisschen breiter und ca. 5x länger als der micro:bit.

Danach falten wir den Karton in der Mitte und an den Rändern:



Jetzt bekommt die Raupe vorne „Zähne“, damit sie in den Boden beißen kann, und hinten machen wir „Füße“, damit sie sich abstützen und nach vorne bewegenden kann.

Für die Füße schneiden wir mit einer Schere 5 oder 6 kleine Schnitte in den Rand des Kartons und biegen die Streifen: ein Streifen nach vorne, den nächsten nach hinten.

Jetzt müssen wir unseren micro:bit zusammen mit den Kabeln für den Servo an der Raupe befestigen. Das geht mit einem Klebeband, indem wir eine Schleife mit der klebrigen (weißen) Seite nach außen formen:

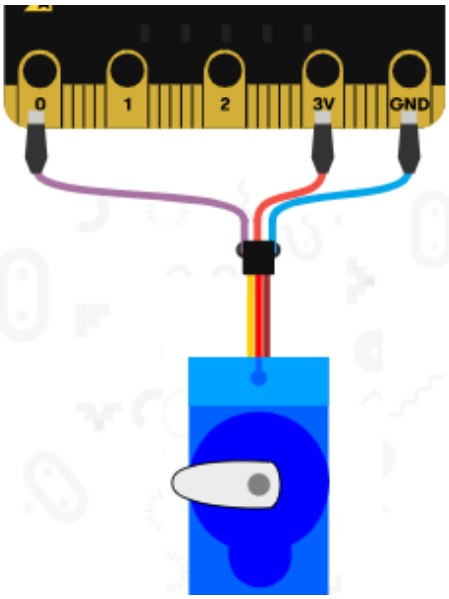
Auf der anderen Seite kleben wir den Servo so an den Karton, dass der Servo-Arm über den Rand reicht und sich frei bewegen kann:

Jetzt biegen wir eine Büroklammer gerade, stecken das eine Ende in das äußere Loch des Servo-Arms und biegen das Ende um, damit es nicht mehr aus dem Loch rutschen kann. Dann knicken wir ca. 2cm vom anderen Ende der Büroklammer im rechten Winkel ab:

Diese abgeknickte Stück der Büroklammer drücken wir jetzt in den Karton auf der Seite, wo wir den micro:bit angeklebt haben und sichern es vorsichtig mit Klebeband (es soll sich noch bewegen können, aber nicht aus dem Karton heraus rutschen):

Jetzt ist die Raupe fertig – wir kontrollieren noch einmal die Kabel zum Servo und schreiben unser Programm.

Der Servo hat 3 Anschlüsse: braun (Minus), rot (Plus) und orange (Steuer-Signal). Diese Anschlüsse müssen wir jetzt richtig mit dem micro:bit verbinden:




1. Orange mit dem Anschluss „0“ (Signal)
2. Rot mit „3V“ (3 Volt oder „Plus“)
3. Braun mit „GND“ (Masse oder „Minus“)

Damit sich die Raupe bewegt, müssen wir den Servo immer hin und her drehen: dazu setzen wir den Winkel vom Servo einmal auf den kleinsten Wert (0) und dann wieder auf den größten (180). Weil der micro:bit diese Befehle aber viel zu schnell hintereinander an den Servo schickt, würde der Servo gar keine Zeit für eine Drehung haben (er bleibt dann sogar stehen).

Wir müssen also einen „Pause“-Befehl nach jedem „Servo“-Befehl machen. Aber wie lange soll diese Pause dauern? Für einen Computer ist eine Sekunde eine sehr lange Zeit und deshalb wird die Pause in Tausendstel-Sekunden (eine Sekunde geteilt durch 1000) angegeben. Eine halbe Sekunde dauert dann 1000:2 = 500 (Millisekunden, oder „ms“).

Probiere den folgenden Code:


Den „Pause“-Befehl findest du in der blauen Bibliothek „Grundlagen“ (ziemlich weit unten).

Den „Servo“-Befehl findest du hier:

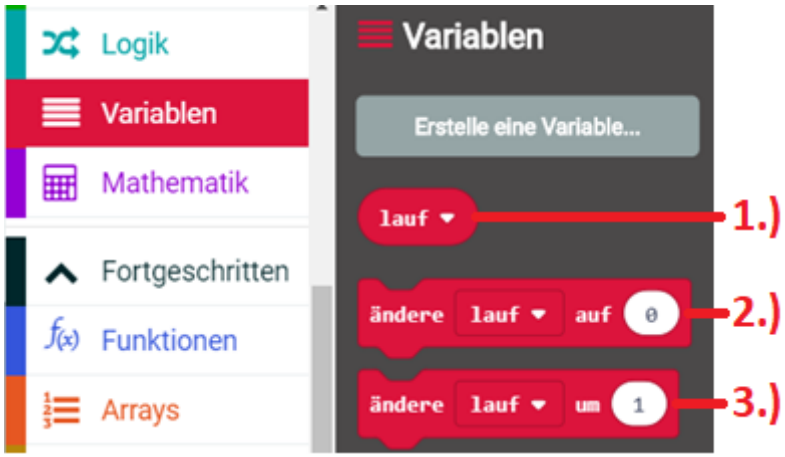
Mit dem Programm vorhin ist unsere Raupe immer nur einen Schritt gegangen. Damit sie von allein weiterläuft (und mit Knopf B auch wieder stehen bleibt), brauchen wir eine **Variable**.

Eine Variable in einem Computer-Programm ist ein „Platzhalter“ (Speicher). Die Variable merkt sich einen Wert!

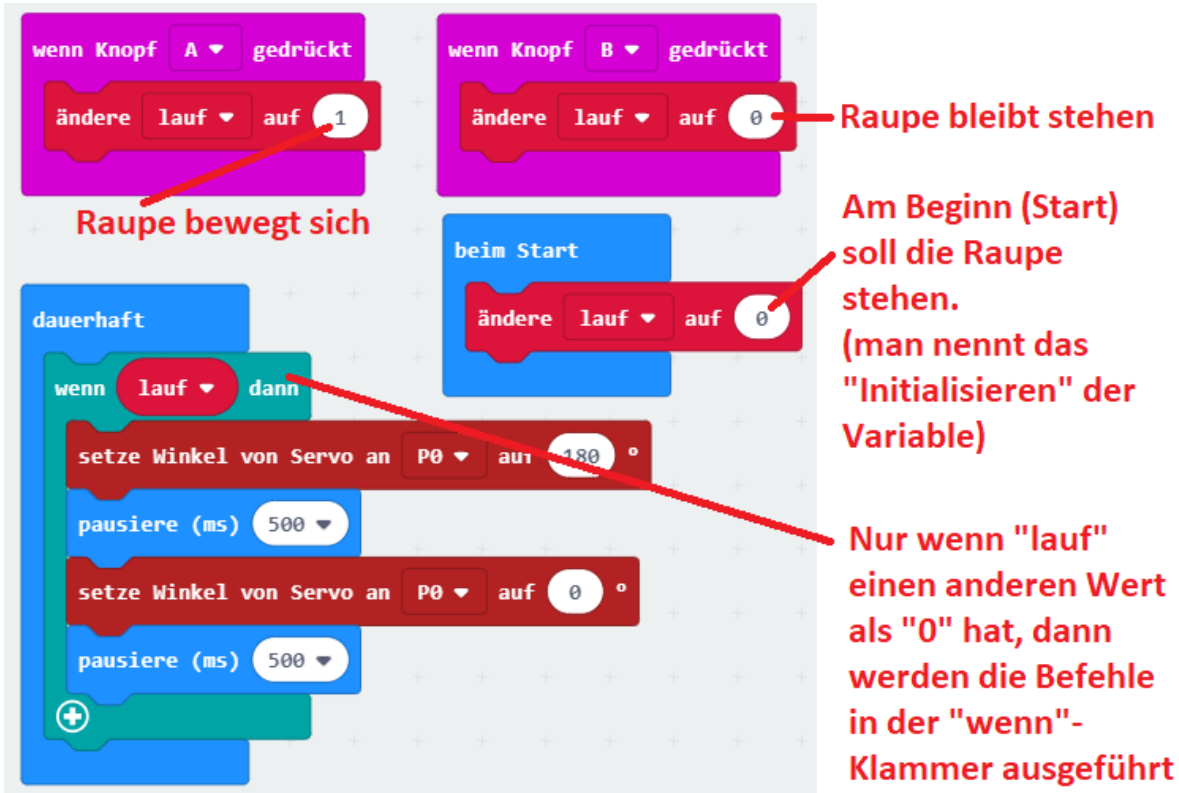
Wir nennen unsere Variable z.B. „lauf“. Wenn wir „lauf“ auf „1“ setzen, soll die Raupe sich bewegen. Wenn wir „lauf“ auf „0“ setzen, dann soll sie stehen bleiben.

In der roten Bibliothek „Variablen“ können wir auf „Erstelle eine Variable…“ drücken und unserer Variablen ihren Namen geben. Wir bekommen dann 3 Dinge:

1.) Der Block mit den runden Ecken ist der WERT der Variablen (damit können wir eine Abfrage machen)
2.) Mit diesem Block können wir den WERT der Variablen festlegen (z.B. auf „0“ oder „1“ setzen): der Block setzt WERT von lauf = Zahl
3.) Mit diesem Block können wir den WERT der Variable um eine Zahl ändern (z.B. um 1 erhöhen, also dazuzählen/addieren). Der Block rechnet: neuer WERT von lauf = alter WERT von lauf + Zahl




Unser Programm könnte damit so aussehen:

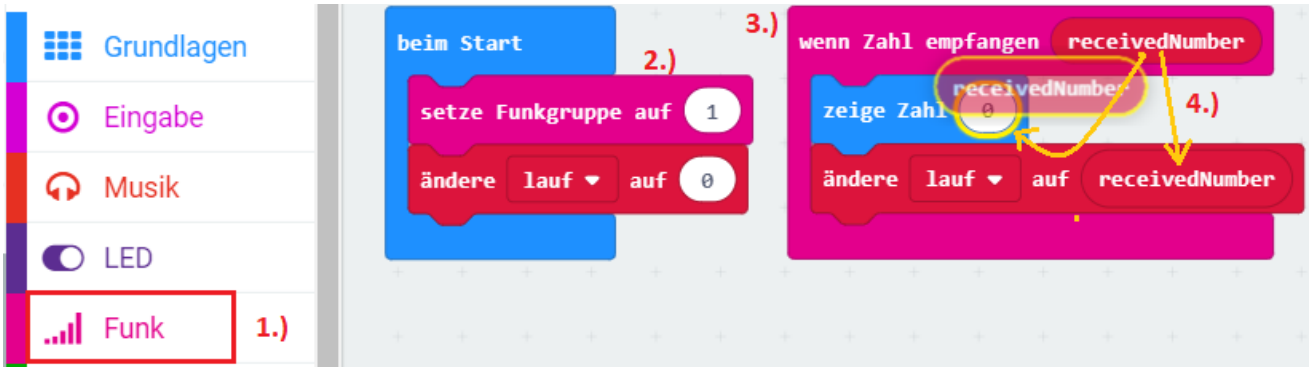


Steuerung über Funk:

Wir können unser Programm auch über Funk steuern, damit wir nicht immer die Tasten auf dem micro:bit drücken müssen (das ist schwierig, wenn die Raupe sich bewegt).

Dazu brauchen wir einen zweiten micro:bit als Fernsteuerung (den Sender).

Auf der Raupe (dem Empfänger) ändern wir das Programm so:

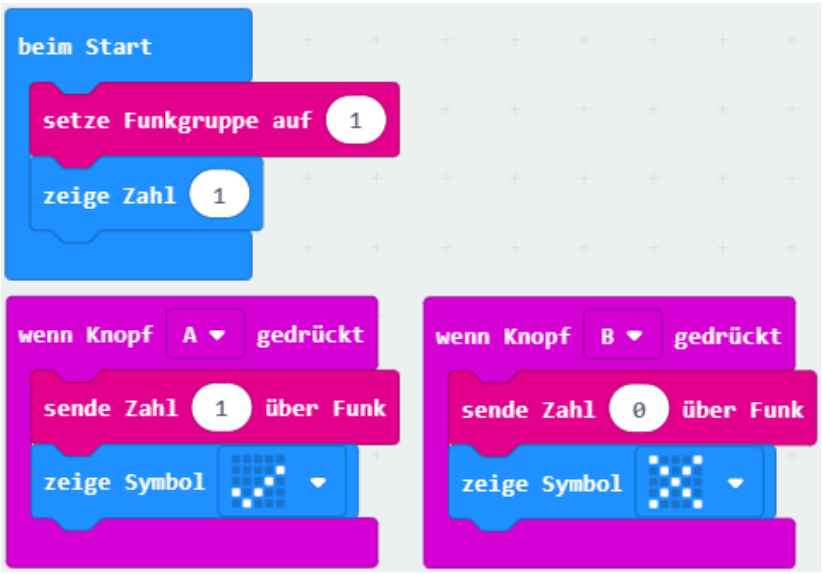



Die beiden lila Blöcke „setze Funkgruppe“ und „wenn Zahl empfangen“ findest du in der Bibliothek „Funk“.

Erklärung „Funkgruppe“: damit man auch viele Raupen über Funk fernsteuern kann und sie sich nicht gegenseitig stören, muss man beim Sender und beim Empfänger die gleiche „Funkgruppe“ einstellen. Es ist auch wichtig, dass diese Funkgruppe nicht von anderen micro:bit verwendet wird. Die Funk-Übertragung reicht meistens nur ein paar Meter (in einem Raum; man kann aber auch durch Wände funken).

Erklärung „receivedNumber“: der micro:bit macht automatisch eine Variable für die empfangene Zahl, die leider einen englischen Namen hat. Man findet diese Variable leider auch nicht in der roten Bibliothek „Variablen“. Aber man kann mit der Maus den roten Block „receivedNumber“ mit den runden Ecken auf jedes Feld für Zahlen (runder Platzhalter) ziehen.

Für den zweiten micro:bit (dem Sender) schreiben wir folgendes Programm:




Das Programm würde auch ohne die Anzeige der Zahl und der Symbole funktionieren. Aber es ist gut, wenn man zuerst die Funkgruppe anzeigt und nach dem Senden der verschiedenen Zahlen auch unterschiedliche Symbole anzeigt.

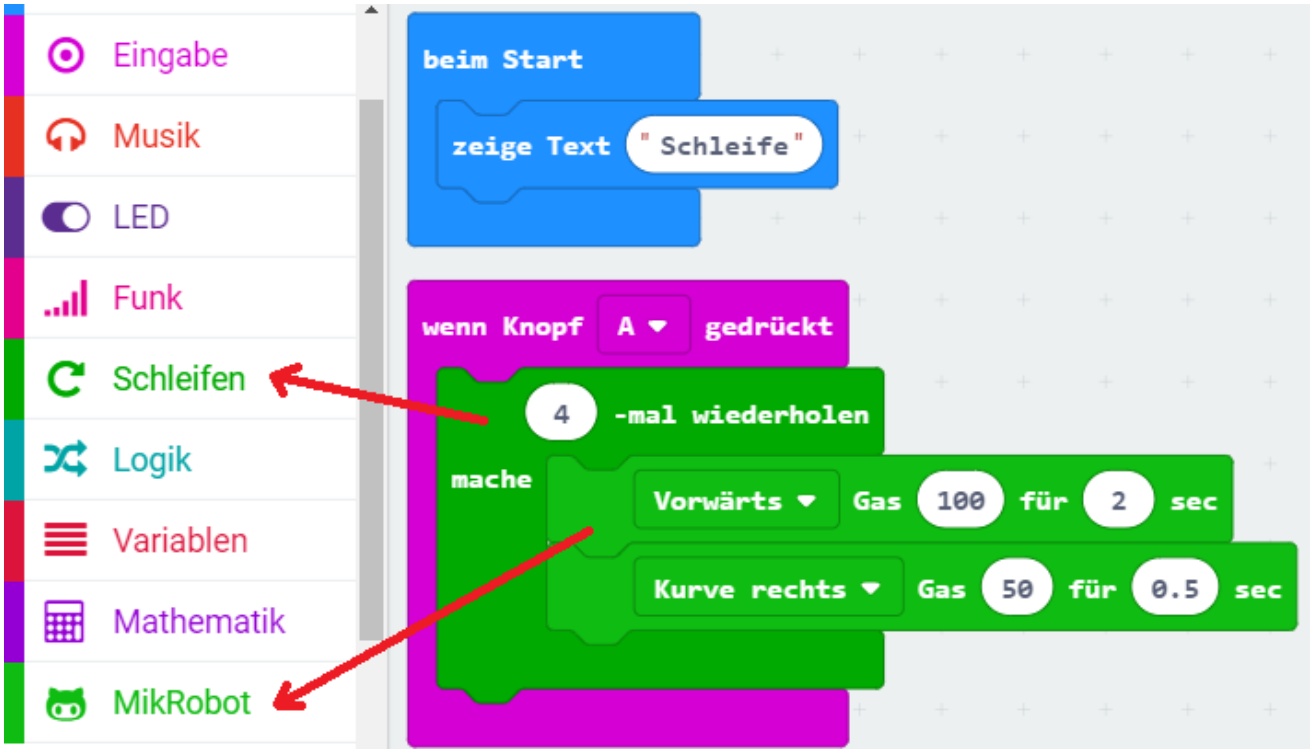
Falls die Fernsteuerung (Funk) nicht klappt, kann man kontrollieren, dass der Sender funktioniert. Mögliche Fehler beim Sender (z.B. Batterie leer oder falsches Programm) kann man damit erkennen und den Fehler eingrenzen (es ist dann wahrscheinlich etwas beim Empfänger falsch).

Ein Quadrat fahren – die Schleife:

Wenn ein Programm mehrmals dasselbe machen soll, dann verwenden wir eine Schleife. Diese findest du in der grünen Bibliothek „Schleifen“.

Wir können damit einstellen, wie oft die Blöcke in der Schleife wiederholt werden sollen. Für ein Quadrat muss der Roboter 4-mal zuerst vorwärtsfahren und dann eine Kurve machen.

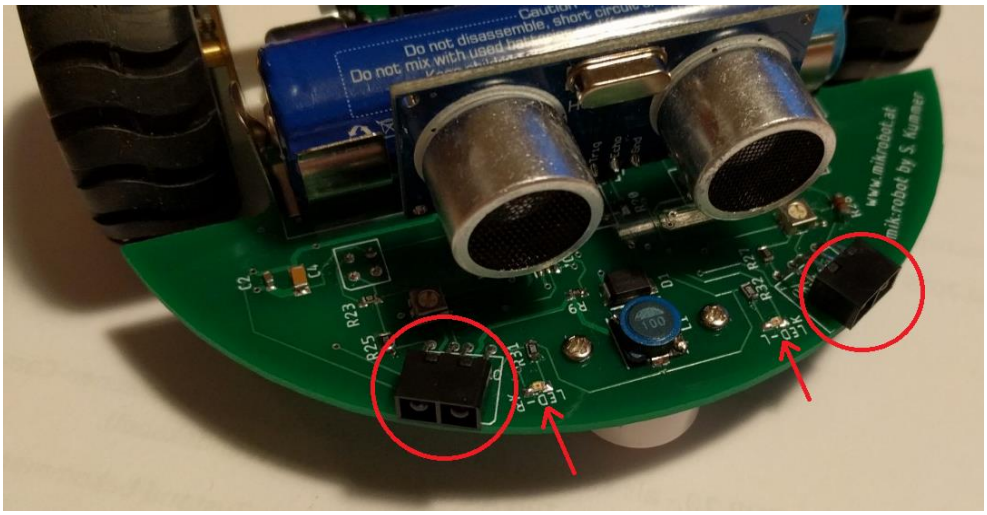
Für ein schönes Quadrat müssen wir jetzt nur noch die Geschwindigkeit (Gas) und die Zeit (in Sekunden) richtig einstellen. Halbe Sekunden bekommst du, wenn du „0.5“ eingibst (das bedeutet, Null-Komma-Fünf).




Hindernisse erkennen:

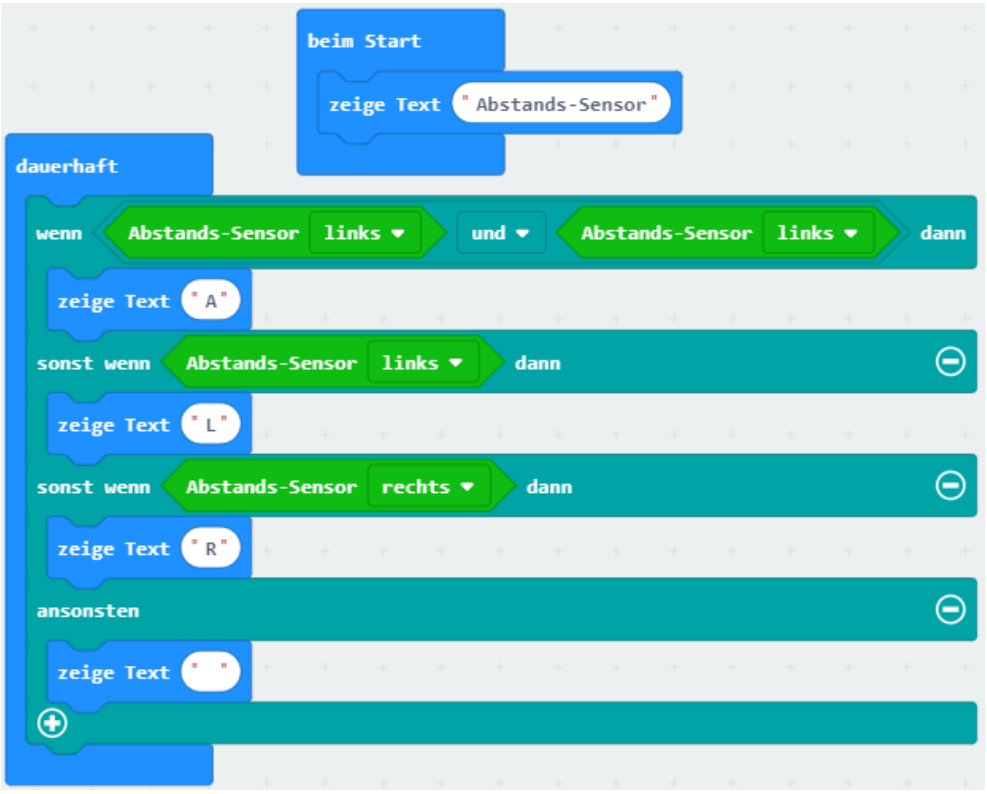
Der Roboter MIK hat 2 Infrarot-Sensoren, mit denen er Hindernisse erkennen kann. Diese Sensoren haben einen Licht-Sender (eine Leuchtdiode) und einen Licht-Empfänger (auch „Fototransistor“ genannt). Im Bild unten sind die beiden Sensoren rot eingekreist.

Wir Menschen können das Infrarot-Licht nicht sehen, aber viele Kameras (z.B. eine Handy-Kamera) können uns zeigen, dass der Sensor wirklich Licht aussendet.



Wenn das Licht auf ein Hindernis trifft, dann wird Licht zurückgestrahlt und der Licht-Empfänger erkennt das. Wenn genug Licht beim Empfänger ankommt, weil das Hindernis nahe genug ist und das Hindernis gut das Infrarot-Licht reflektiert, dann geht am Roboter ein rotes Licht an (im Bild sieht man die beiden roten Lichter bei den 2 Pfeilen).

Jetzt schreiben wir ein einfaches Programm, dass uns anzeigt, ob der Roboter das Hindernis rechts oder links oder mit beiden Sensoren erkennt:




Mit ein paar Änderungen erstellen wir ein Programm, mit dem unser Roboter vor einem Hindernis zurückweicht, dann eine zufällige Drehung tut und sich dann wieder auf den Weg vorwärts macht.


Dafür verwenden wir eine neue Art von Schleife: die „Während <wahr> mache“

Diese Schleife wiederholt die Befehle in der Klammer immer und immer wieder, bis die <Bedingung> nicht mehr wahr ist.

Wir müssen kontrollieren, ob der linke oder der rechte Sensor „nichts“ sehen. Dazu nehmen wir aus der Bibliothek „Logik“ die zwei Blöcke „oder“ und „nicht“.