Was ist ein Roboter?

Roboter „Atlas“

- MASCHINE (beweglich/mobil oder festgeschraubt)
- Von COMPUTERPROGRAMM gesteuert
- Soll den Menschen „mechanische“ ARBEIT abnehmen (Sprachassistent, wie z.B. Alexa, ist also kein Roboter)
- MECHANIK + ELEKTRONIK + INFORMATIK → Mechatronik

Industrieroboter:

- Weltweit gibt es ca. 3,5 Mio. Industrie-Roboter (1 Roboter pro 2000 Menschen, d.h. 12 Roboter in Graz – wenn gleichmäßig verteilt)
- Erster Industrie-Roboter wurde 1956 (vor 63 Jahren!) erfunden
- Schweißen, Schneiden, Lackieren, Montieren, Verpacken, …
- Wichtig: Sicherheit (viel Kraft, schnell)

Service-Roboter:

- Z.B. Staubsauger, Rasenmäher, Schwimmbadroboter, Fensterputzer…

Erkundungs-Roboter:

- Raumfahrt (z.B. am Mond oder Mars), Rettungsroboter, Tauchroboter, schlecht: Kampfroboter

Medizinische Roboter:

- Operationen (Roboter als präzise Hand des Chirurgen für mikroskopisch kleine Eingriffe, z.B. an Nerven, Gehirn, usw.), Pflege von z.B. alten Menschen, …

„Menschliche“ Roboter (griechisch: Androide): sehr kompliziert, „Künstlicher Diener“ (ein alter Traum der Menschen)

- Exoskelett (kann Menschen helfen, wieder gehen zu lernen)
- Assistenzroboter für behinderte Menschen (z.B. automatischer Rollstuhl mit Sprachcomputer)

Spielzeug-Roboter:

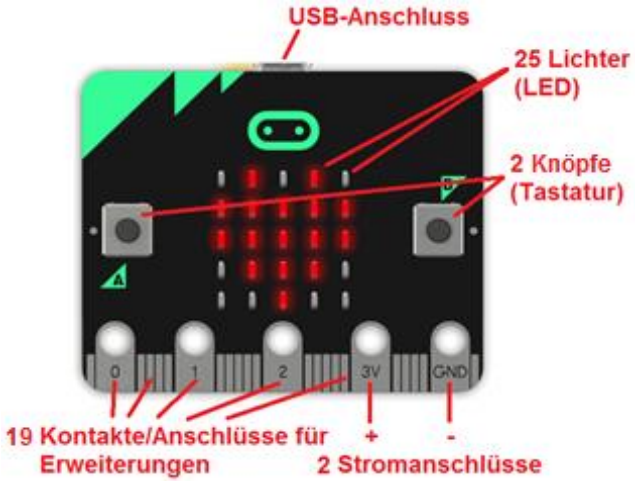
- Von einfach bis kompliziert (Anzahl der Sensoren, Anzahl der Aktoren (Motor), Rechenleistung)
- Achtung bei Spracherkennung: Oft mit dem Internet verbunden (Privatsphäre)

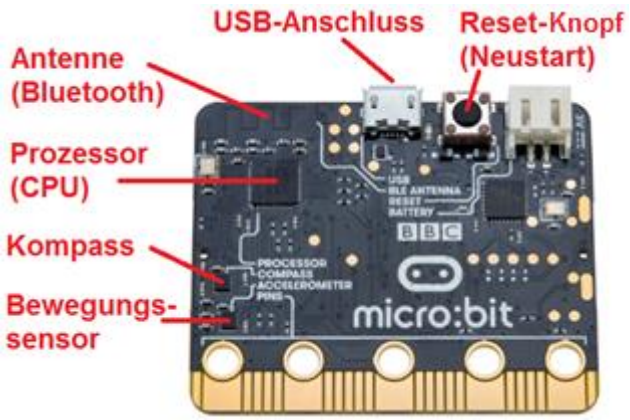
Die **Roboter-Gesetze** (von Isaac Asimov):

1. Ein Roboter darf keine Menschen verletzen oder durch Untätigkeit zulassen, dass einem Menschen Schaden zugefügt wird
2. Ein Roboter muss Menschen gehorchen – es sei denn, ein solcher Befehl verletzt Regel eins
3. Ein Roboter muss sich selbst beschützen, solange dieser Schutz nicht mit Regel eins oder zwei kollidiert

COMPUTER (Roboter-Steuerung)

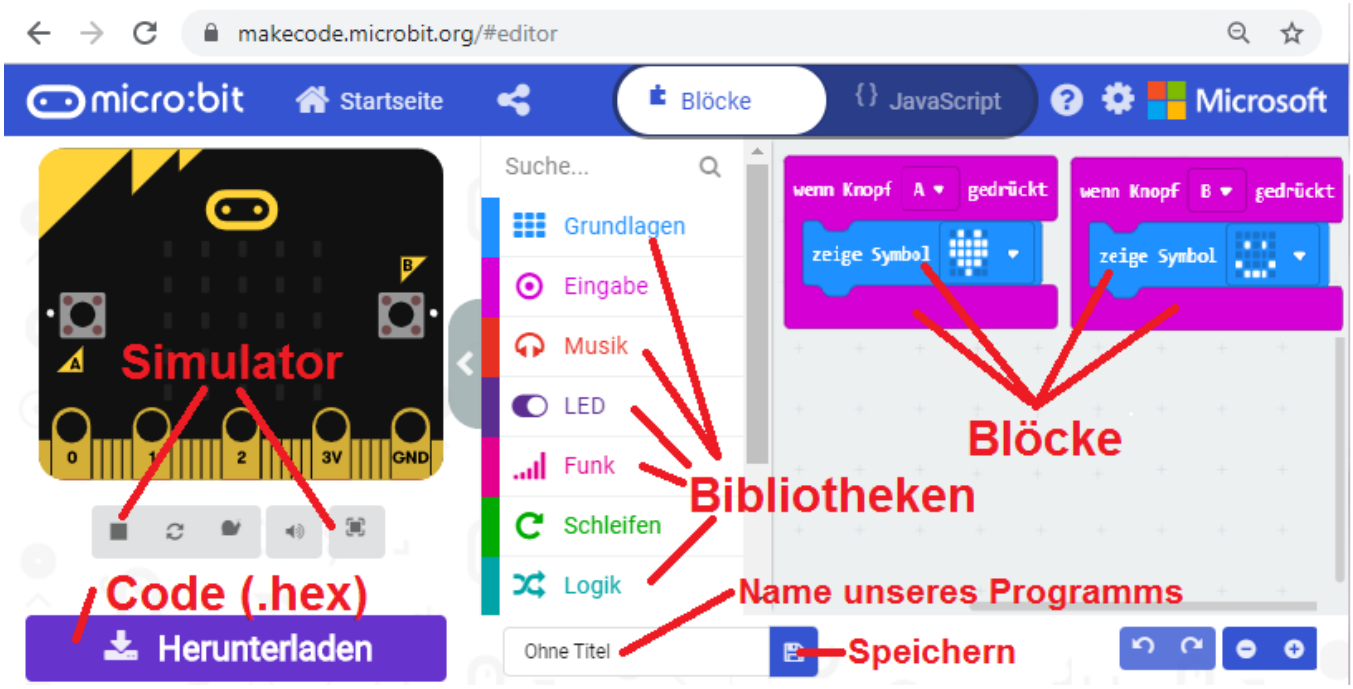
Wir verwenden als Computer den micro:bit, weil er klein und günstig ist und viele Funktionen hat:





Rückseite

Diesen Computer muss man programmieren, das heißt: wir müssen uns überlegen, welche Befehle er befolgen soll. Das Programm schreiben wir in einer „Programmierumgebung“: makecode.microbit.org



Die Befehle für den Computer bestehen in unseren Programmen aus Blöcken, die wir in die richtige Reihenfolge bringen und richtig einstellen müssen.

Es gibt viele unterschiedliche Blöcke, die in verschiedenen „Bibliotheken“ geordnet sind:

Grundlagen: Zahlen, Symbole und Text auf den LEDs (Lichtern) anzeigen, Programmpausen

Eingabe: Reaktion auf Knöpfe, Bewegungssensor, Temperatur, Licht, Kompass, …

Und viele mehr…

Wir sollten unseren Programmen einen „Titel“ (Namen) geben, der beschreibt, was das Programm macht und es nach jeder längeren Arbeit „speichern“.

Mit dem „Simulator“ können wir unser Programm ohne Übertragung auf den micro:bit ausprobieren.

Wenn das Programm fertig ist, muss man den Code auf den micro:bit übertragen („Herunterladen“)

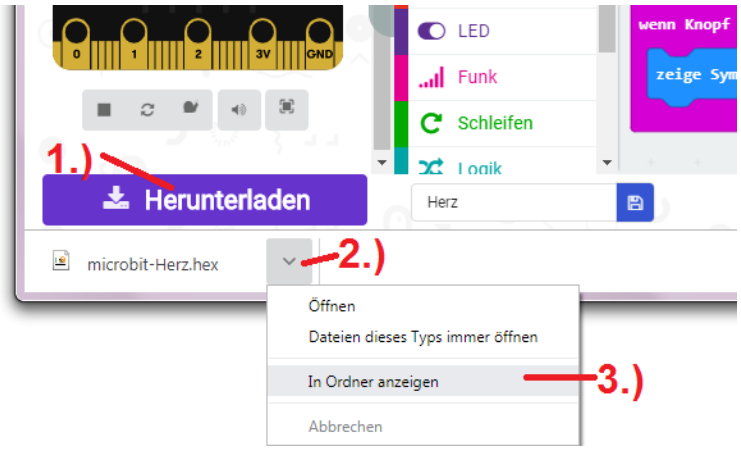
Übertragen des Programms („Herunterladen“):

Dazu muss der micro:bit über ein USB-Kabel mit dem Laptop verbunden sein:

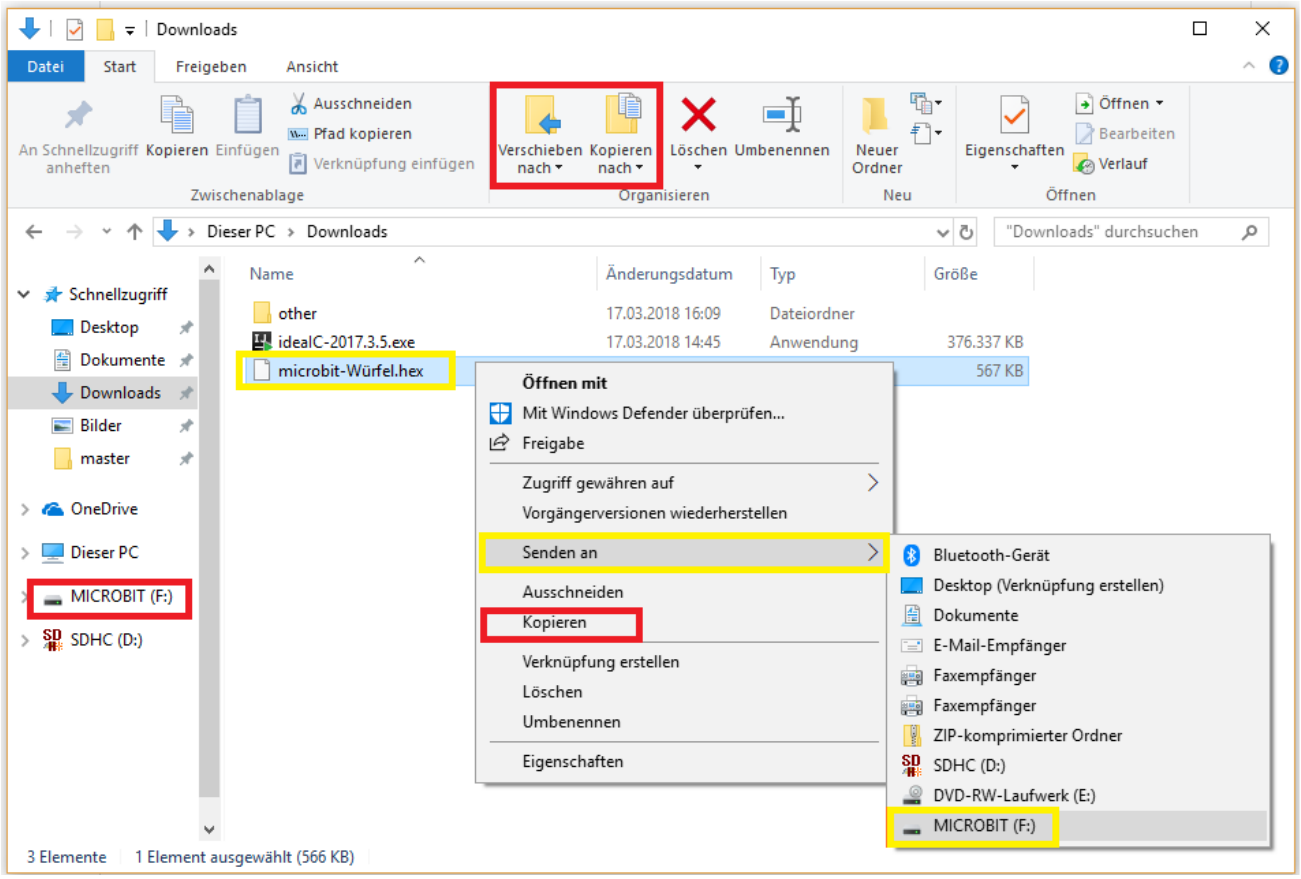
1. Lade das Programm in der Entwicklungsumgebung mit „Herunterladen“ auf den Laptop

2. Die Erklärung mit den bunten Zeichnungen (Download to your micro:bit) mit dem schwarzen x schließen

3. Öffne den Ordner, in den du die Datei (.hex) heruntergeladen hast (meist unter „Downloads“)



4. Übertrage das Programm auf den micro:bit. Klicke dazu mit der rechten Maustaste auf die Datei, um das Menu zu öffnen („Senden an“). Oder kopiere die Datei von diesem Ordner auf den MICROBIT.



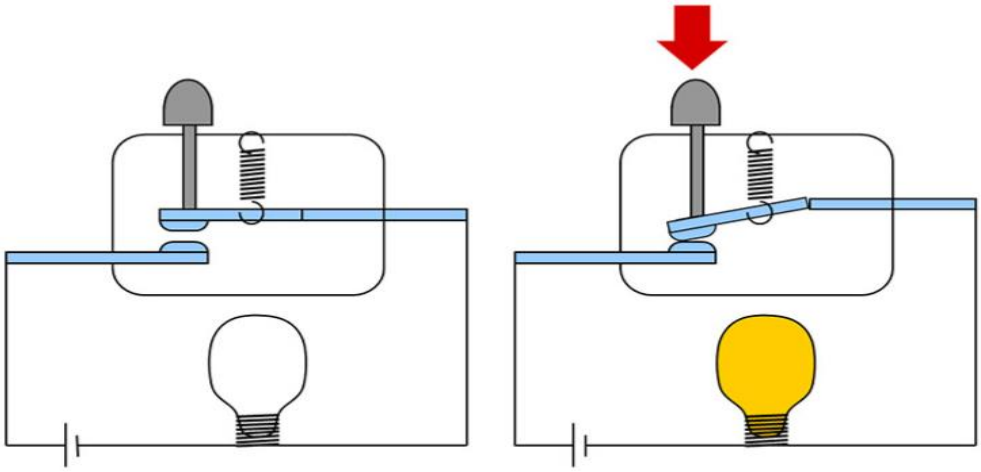
Das Übertragen des Programms dauert ein wenig. Der micro:bit blinkt während der Übertragung mit einer gelben LED (neben dem USB-Anschluss auf der Rückseite).

Wenn das Programm fertig übertragen ist, läuft es auch schon auf unserem Computer. Jetzt ist ausprobieren (testen) angesagt!

SENSOREN (Fühlen - Eingabe) und AKTOREN (Ausführen - Ausgabe)

Knopf (Berührungssensor):

Das ist ein einfacher Kontakt, der nur 2 Zustände erkennen kann: nicht gedrückt/ gedrückt. Wird der Kontakt gedrückt, dann kann der Strom fließen (im Bild leuchtet dann die Lampe).



Unser Roboter (der micro:bit) hat 2 Knöpfe, mit denen wir den Roboter zum Beispiel starten und stoppen können (linkes Bild: „wenn Knopf A gedrückt“).

Wir können aber auch bei den Erweiterungs-Anschlüssen einen Kontakt anschließen (oder mit den Fingern eine Verbindung machen) und diese abfragen (mittleres Bild: „wenn Pin P0 gedrückt“).

Das Drücken „von Pin P0“ können wir mit unseren Fingern machen: dazu müssen wir den Anschluss „GND“ (Masse) berühren und gleichzeitig den Anschluss „0“ (rechtes Bild). Der Strom, der dabei durch unsere Finger fließt, ist extrem klein und es besteht keine Gefahr.

Wenn man statt der Finger Kabel nimmt, kann man einen Berührungssensor am Roboter anschließen und der Roboter erkennt, wenn er gegen ein Hindernis gefahren ist oder etwas berührt (Hebel, Mechanik, …)

Temperatur-Sensor:

Der micro:bit kann auch die Temperatur messen. Der Fühler ist im Prozessor (in der CPU) eingebaut. Wenn ein Prozessor viel zu arbeiten hat (viel rechnet), dann wird er warm – deshalb haben die meisten Laptops einen Lüfter eingebaut. Unser micro:bit ist sehr sparsam und wird eigentlich nie warm, deshalb misst er die Temperatur der Umgebung.

Bisher haben wir nur mit Blöcken gearbeitet, die entweder eine Klammer (= Schleife) oder einen Ablauf (z.B. „zeige Symbol“) zeigen.

Den Ablauf-Block erkennt man an den „Zähnen“ für den Anfang und das Ende. Einen Ablauf-Block kann man nur an bestimmten Stellen einfügen.

Die Temperatur ist aber keine Klammer und auch kein Ablauf, sondern ein „Wert“. Diese Wert-Blöcke haben eine runde Form und können auch nur an bestimmten Stellen eingefügt werden (dort wo ein rundes „Loch“ ist). Ein passender Block für die Anzeige eines Werts (der Temperatur) ist z.B. „Zeige Zahl“ (Bibliothek „Grundlagen“).

Kompass-Sensor:

Ein Kompass kann Magnetfelder erkennen, z.B. das Magnetfeld der Erde (Nordpol – Südpol). Der Kompass im micro:bit kann die Richtung und auch die Stärke von Magnetfeldern messen.

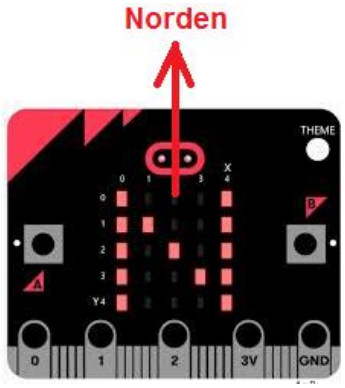
Für einen Roboter ist die Richtung wichtig. Die Himmelsrichtungen bei einem Kompass sind in „Grad“ (mit dem Zeichen °) eingeteilt:

Norden (N) = 0° (und auch 360°)

Osten (auf Englisch „East“, also E) = 90°, Süden (S) = 180° und Westen (W) = 270°

Einmal ganz rund herum sind 360° (4 mal 90 = 360). Wenn wir die 4 Himmelsrichtungen anzeigen wollen, dann müssen wir überlegen, welche Grad-Werte z.B. nach Norden zeigen, denn Norden reicht von „Nordost“ (NE) bis „Nordwest“ (NW). Das heißt, dass alle Richtungen die kleiner als 45° (0+90:2) und alle die größer als 315° (360-90:2 = 360-45) sind, „Norden“ bedeuten.

Wir können nun ein Programm schreiben, dass die Himmelsrichtung anzeigt. Für eine richtige Messung muss der micro:bit liegen und der USB-Anschluss zeigt uns die gemessene Richtung.



Für das Programm brauchen wir neue Blöcke:

In der Bibliothek „Variablen“ erstellen wir eine neue Variable. Das ist ein Platzhalter mit einem Namen (z.B. „richtung“). Dieser Platzhalter merkt sich dann den Wert von „Kompassausrichtung“, wenn wir ihn in den Block „ändere richtung auf…“ geben.

Diesen Änderungs-Block geben wir in die Schleife „dauerhaft“, damit der Computer immer wieder eine neue Messung der Himmelsrichtung macht.

Für die Überprüfung der Richtung brauchen wir 3 neue Blöcke aus der Bibliothek „Logik“: eine Bedingung (wenn wahr dann…), einen Vergleich (0 < 0) und eine ODER-Verknüpfung (Boolean oder).

Achte auf die genaue Anordnung diese vielen Blöcke! Den Wert „Richtung“ finden wir unter „Variablen“.

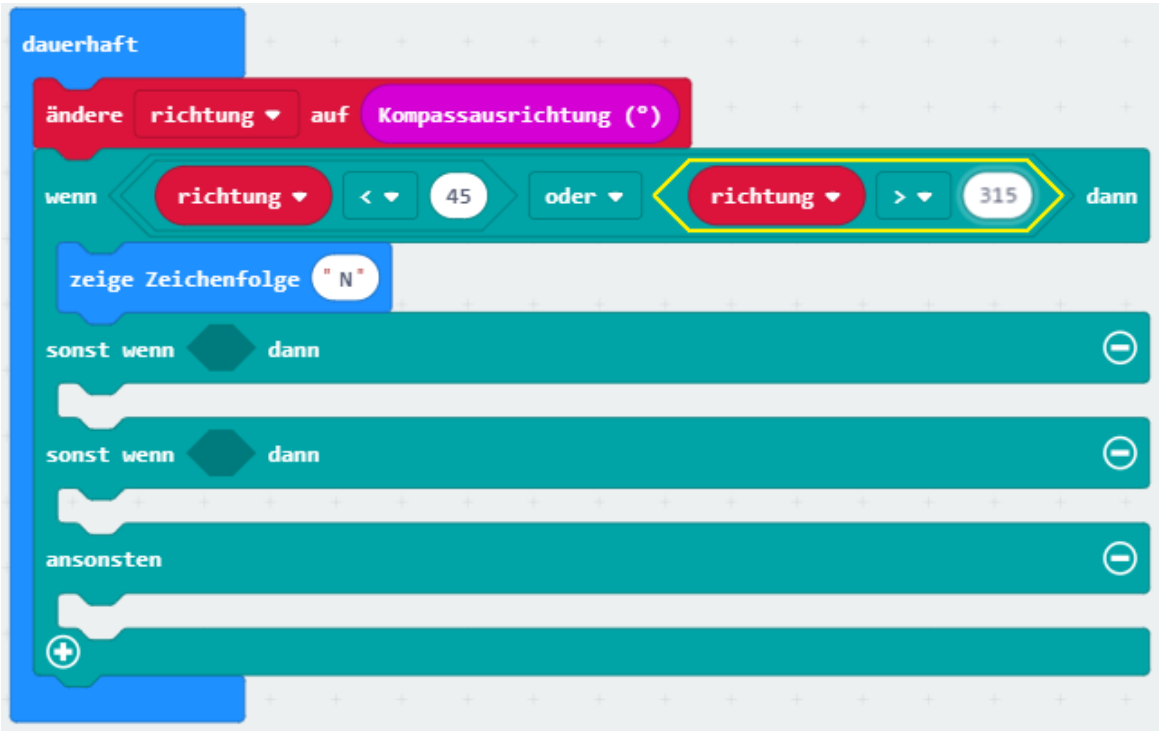
Die Bedingung (wenn…dann) ist wie eine Klammer. Wenn die Bedingung erfüllt (wahr) ist, dann wird der umklammerte Block (in unserem Beispiel „zeige Zahl“) ausgeführt. Wenn die Bedingung nicht erfüllt (falsch) ist, dann wird der umklammerte Block nicht ausgeführt und das Programm geht bei  weiter.

Wenn wir den Kompass das erste Mal (oder nach einer langen Pause wieder) verwenden, dann muss der micro:bit den Kompass-Sensor „kalibrieren“. Das bedeutet, dass wir den micro:bit in alle Richtungen drehen müssen und der Computer dabei viele Messungen des Magnetfeldes macht (bis alle roten Lichter leuchten).

Aus den vielen unterschiedlichen Messungen rechnet der micro:bit dann aus, wo der höchste und der niedrigste Wert ist. Damit teilt der Computer dann die 360° neu auf.

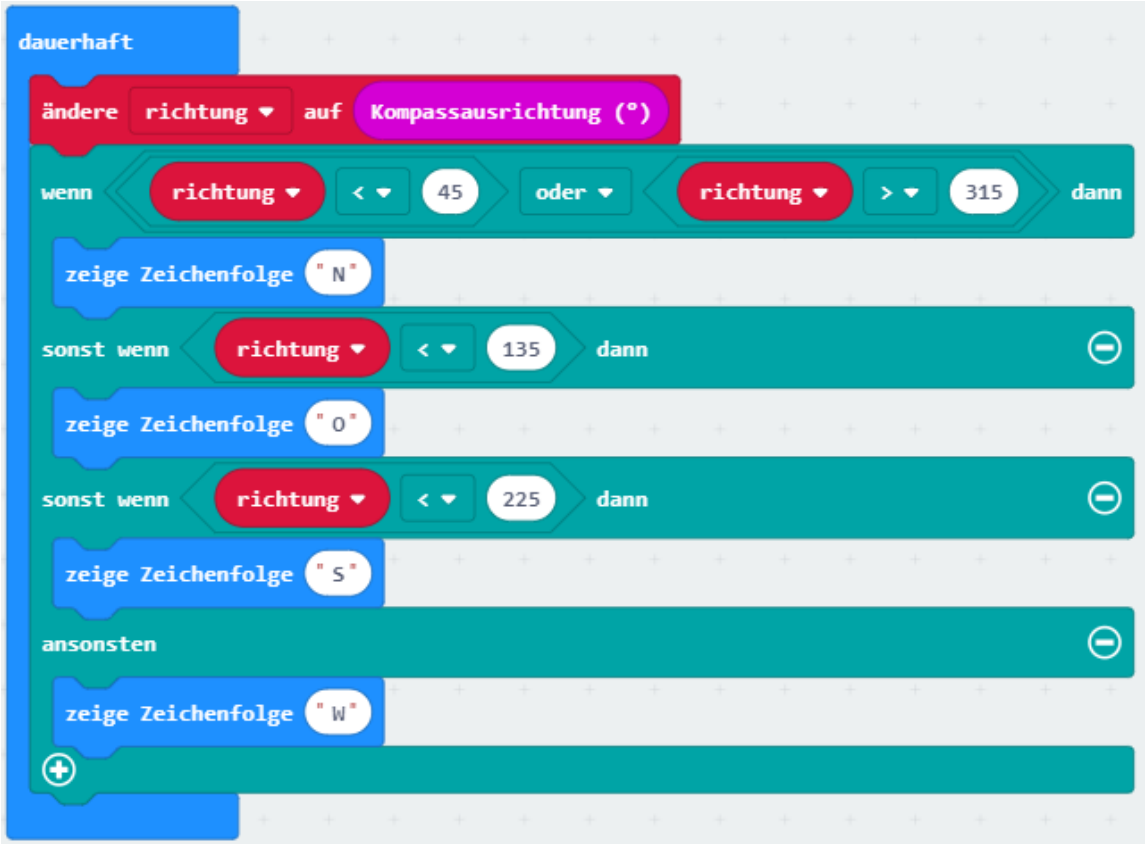
Bisher haben wir nur die Richtung Norden erkannt. Wir können aber unser Programm noch für die anderen Himmelsrichtungen erweitern.

Dazu muss unser Programm in der „wenn – dann“-Bedingung (türkiser Block) noch 3 andere Bedingungen abfragen. Dazu drücken wir 3x auf das ⊕-Zeichen und bekommen:

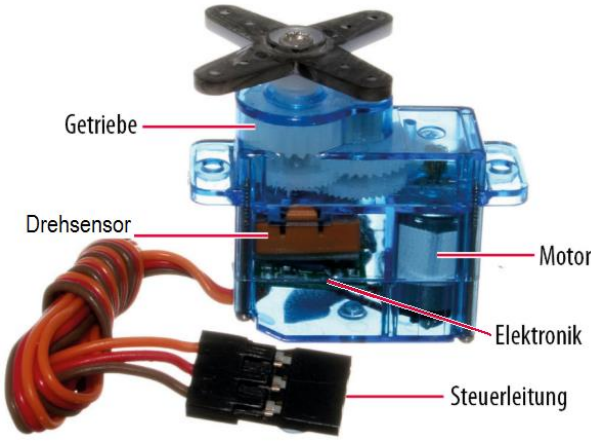


Jetzt müssen wir **der Reihe nach** (die Grad ° werden immer größer) noch die anderen Himmelsrichtungen abfragen. Zuerst kommt Osten, weil hier die Grad von 45° (90-45) bis 135° (90+45) gehen. Das bedeutet, wir sagen: „sonst wenn richtung < 135 dann“ und „zeige Zeichenfolge „O“ “.

Dann kommt der Süden von 135° (180-45) bis 225° (180+45) und wenn es weder Norden, noch Osten, noch Süden ist, dann muss es für den letzten Teil der Westen sein. Das bedeutet, dass wir für den Westen gar keine Grad mehr kontrollieren müssen:



Servo-Aktor:

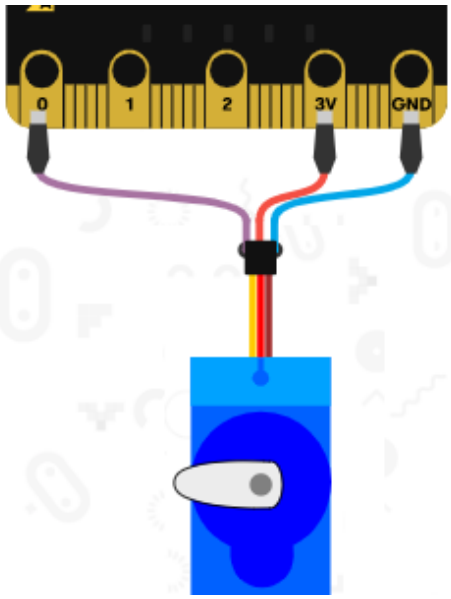


Ein Servo besteht aus einem kleinen Motor und einem Dreh-Sensor. Wenn der Servo vom micro:bit den Befehl bekommt, seinen Arm (Hebel) in eine bestimmte Richtung zu drehen, dann läuft der Motor so lange in diese Richtung, bis der Dreh-Sensor erkennt, dass diese Richtung erreicht ist.

Die Richtung wird – wie beim Kompass – in Grad (°) angegeben. Eine halbe Umdrehung hat 180°, eine Viertel-Umdrehung 90°.

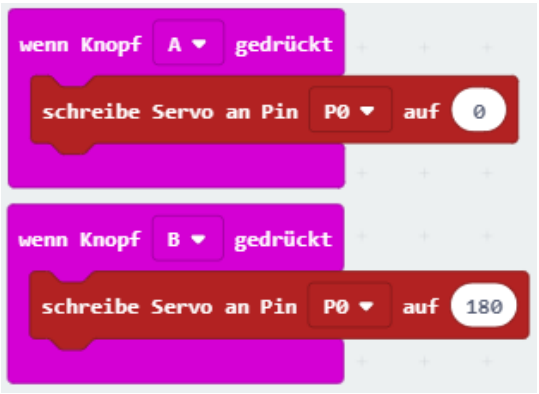
Damit der Servo genug Kraft hat, gibt es auch ein Getriebe: der Motor dreht sehr schnell mit wenig Kraft und das Getriebe wandelt die schnelle Drehung in eine Langsame mit viel Kraft.

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“)

Das Programm für die Ansteuerung des Servos ist einfach:



Den Block „schreibe Servo an Pin“ finden wir in der Bibliothek „Pins“. Dazu müssen wir den Abschnitt „Fortgeschritten“ aufklappen:

Licht-Sensor (Infrarot-Sensor):

Dieser Sensor sendet (für das menschliche Auge unsichtbares) Licht aus und empfängt das reflektierte Licht.

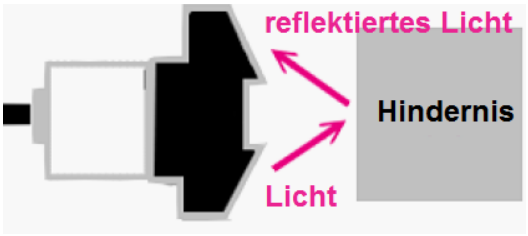
Wenn das Hindernis sehr hell ist (z.B. weiße Farbe an der Wand), wird viel Licht reflektiert und der Empfänger im Sensor zeigt einen hohen Wert an. Wenn das Hindernis dunkel ist, dann ist der Wert niedrig.

Der Sensorwert kann vom Computer (oder Roboter) auf 2 Arten verwendet werden:

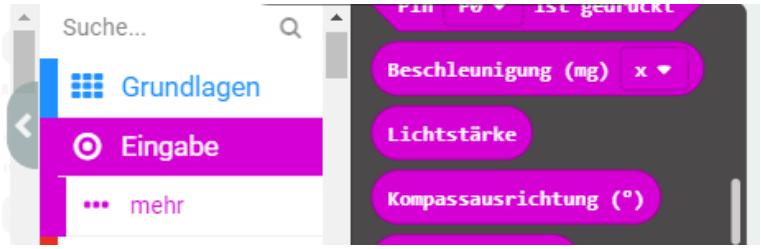
1.) Digital: „Weiß“ oder „Schwarz“. Es gibt nur diese zwei Zustände, die man auch mit „EIN“ und „AUS“ bzw. 1 und 0 bezeichnen kann.

2.) Analog: viele Abstufungen zwischen „Weiß“ und „Schwarz“, sogenannte Grau-Stufen. Damit kann der Roboter z.B. auch erkennen, wie nahe er einem (reflektierenden) Hindernis ist (je näher, desto mehr reflektiertes Licht kommt zum Empfänger zurück und desto höher der Wert). Das funktioniert bei unseren Sensoren bis zu einer Reichweite von mehreren Zentimetern.

Das Infrarot-Licht wird auch für viele Fernbedienungen (z.B. Fernseher) und bei Überwachungskameras verwendet und kann z.B. von Schlagen gesehen werden. Auch viele Kameras (z.B. im Smartphone) können Infrarot-Licht erkennen (damit kann man z.B. kontrollieren, ob eine Fernbedienung funktioniert).



Der micro:bit kann mit seinen LEDs nicht nur Licht machen, sondern auch die „Lichtstärke“ der Umgebung messen.



Der Block „Lichtstärke“ liefert einen Wert zwischen 0 (absolut dunkel) und 255 (sehr hell). Wenn wir die Lichtstärke auf der Anzeige des micro:bit ausgeben, dann laufen die 3 Stellen immer über die Anzeige – und das ist schwer zu lesen. Es würde reichen, wenn wir nur einen Wert zwischen 0 und 9 für die Lichtstärke hätten.

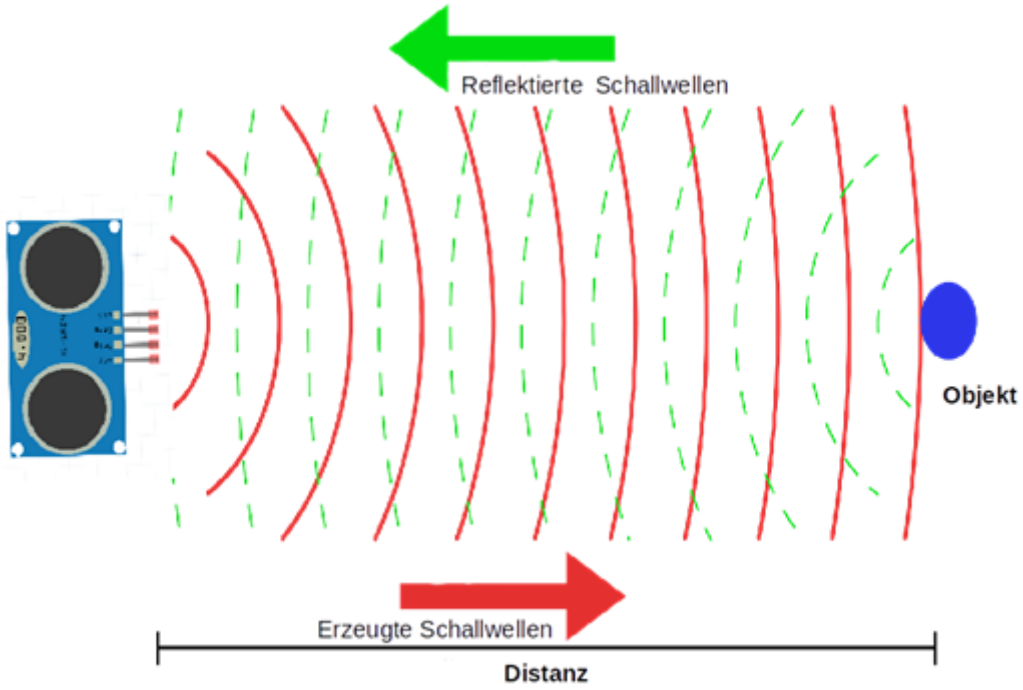
Und das können wir ganz einfach programmieren! Wir rechnen uns aus, durch welche Zahl wir 255 (der größte mögliche Wert) dividieren müssen, damit eine 9 entsteht. 255:9 = 28 (mit 3 Rest)

Zum Dividieren verwenden diesen Block:

Ein Computer rechnet sehr genau und der Rest wird als Zahl hinter dem Komma angegeben. Wieder würde eine lange Zahl über die Anzeige laufen. Wir können aber auch programmieren, dass das Ergebnis „gerundet“ wird:

Ultraschallsensor (wie bei den Parksensoren eines Autos):

Misst Entfernungen (von 3cm bis 2m), das linke „Auge“ ist ein Lautsprecher/Sender, das rechte ein Empfänger. Die ausgesendeten Schallwellen werden von Hindernissen reflektiert und laufen zum Empfänger zurück (wie ein Echo). Der Sensor errechnet aus der Zeit die Entfernung (Schallgeschwindigkeit ist ca. 343m pro Sekunde). Das Echo eines 10cm entfernten Hindernisses kommt nach 0,6ms (Millisekunden) zurück. Das sind 6 Sekunden geteilt durch 1000!



Bewegungssensor:

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

Bei einem drehenden Kreisel wirken 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 Sensor gemessen und der Sensor berechnet aus der Kraft und der Zeit (je länger verdreht wird, desto größer ist der Drehwinkel) die Winkeländerung.

Leider messen die meisten Kreiselsensoren nicht sehr genau und oft ändert sich der angezeigte Winkel auch ohne Drehung innerhalb ein paar Sekunden um 1-2° (man nennt das „Drift“). Deshalb muss man diesen Sensor auch vor jeder Messung zurücksetzen.

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!!!