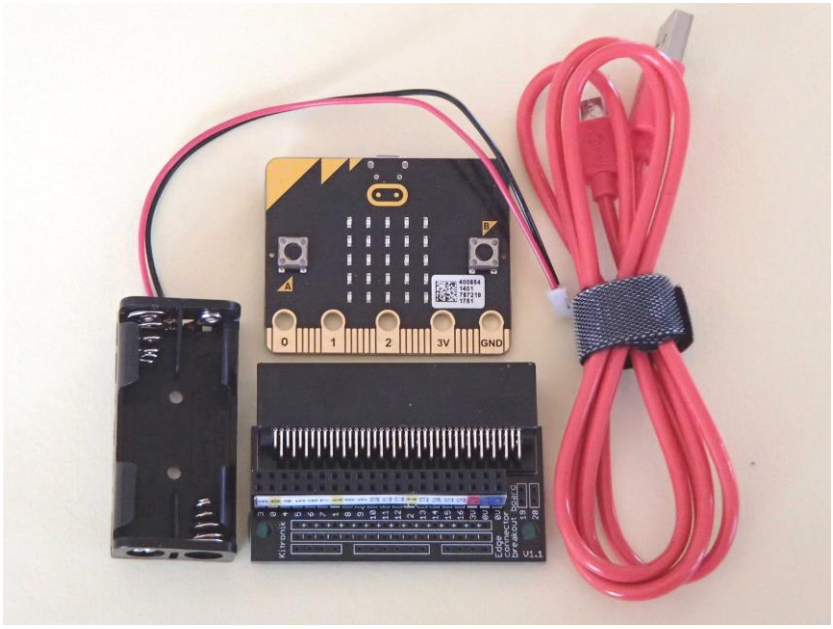
Digitalelektronik

Inhalt

Grundlagen:	Microbit; Programme schreiben und starten; Kaufen siehe Abschnitt 3
Digitale Signale:	Eingabe; Logik; Signale aus der Welt; Alarmanalage; Variablen; Stoppuhr
Analoge Signale:	Helligkeit; Lichtschranke; Farbcode für Widerstände; Löten

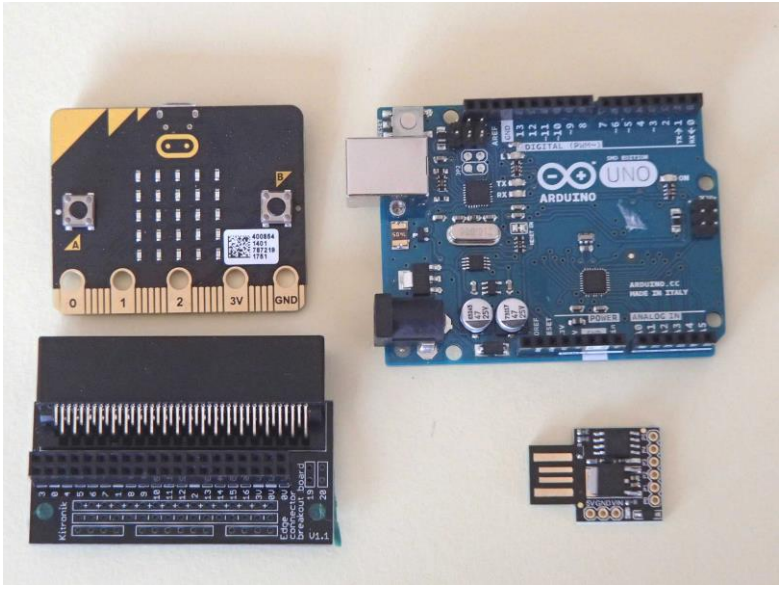
Microbit

Mikrocontroller sind **Einplatinencomputer**. Unser BBC micro:bit ist zwar kein kleiner PC, aber wir können Daten ein- und ausgeben, zum Beispiel über Tasten oder Leuchtdioden. Wir können allerlei Elektronik an seine Pins anschließen. Und vor allem: Wir können ihn programmieren.
Ein neuer Microbit wird bereits mit einem kurzen Programm geliefert: Sobald er mit Strom versorgt ist, stellt er ein paar seiner Funktionen vor: Die Anzeige mit der Laufschrift, die zwei Eingabeknöpfe, den Beschleunigungs- und den Lagesensor.



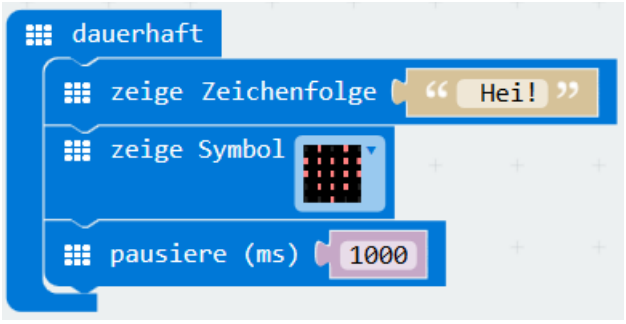
Beim **Kaufen** lohnt es sich, Preise auf eBay zu vergleichen und auch an USB-Kabel und Batteriehalterung oder evtl. ein Breakout Board zu denken. Bemerke, dass wir keine Pins, sondern eine 2x23-polige Buchsenreihe auf das Breakout Board löten. Siehe auch den Abschnitt „Einkaufen“.

Es gibt noch viele andere Mikrocontroller. Bei Bastlern verbreitet ist der **Arduino**. Um ihn zu programmieren, muss auf dem PC erst eine Entwicklungsumgebung installiert werden, meist die „Arduino IDE“; sie läuft dann auch ohne Internetanschluss. Sie funktioniert mit Befehlen über die Tastatur, nicht mit der Maus. Mit dem Arduino vergleichbar und auf gleiche Weise zu programmieren ist der **Digispark**. Er hat weniger Anschlüssen und eine geringere Leistung, aber er reicht für den Elektronikkurs vollkommen aus. Über chinesische Händler kostet er weniger als 1,30 Euro. **Calliope** ist von der Programmierumgebung her mit allen Vor- und Nachteilen das Gleiche wie der BBC micro:bit, nur sechseckig, mit wenig zusätzlichem Schnickschnack mehr als doppelt so teuer.



Programme schreiben und starten

Unter https://makecode.microbit.org/# starten wir die Entwicklungsumgebung für den Microbit. Dort ziehen wir mit der Maus aus den verschiedenen Kategorien die gewünschten Befehle in das Programmierfenster. Klicke mit der Maus auf die blaue Kategorie „Grundlagen“ und ziehe ein paar Anweisungen nach rechts, zum Beispiel das klammerförmige dauerhaft oder zeige Zeichenfolge „…“, das du in die Klammer von dauerhaft setzt. Etwa so ähnlich wie im Bild.



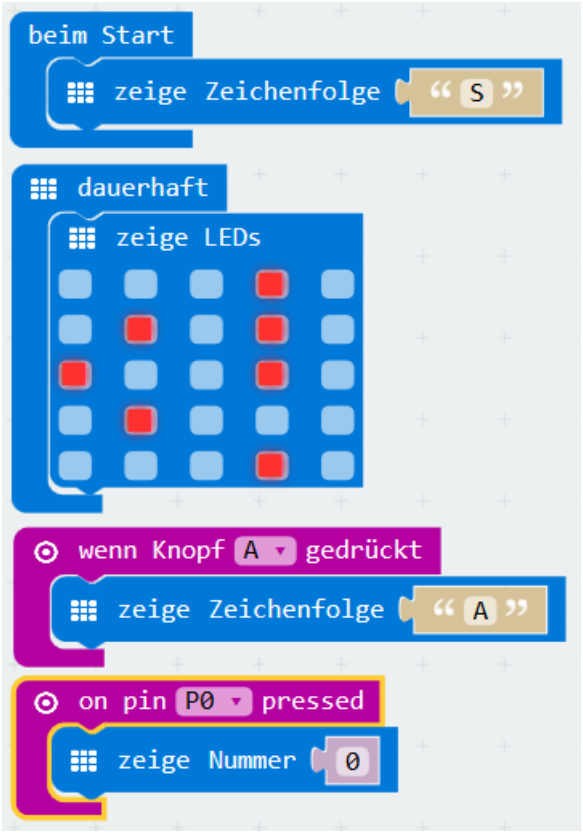
Gib deinem Programm im Schriftfeld rechts von „Herunterladen“ einen Namen und speichere es ab. Dann liegt dein Werk im Downloadordner. Sein Code *.hex ist für uns zwar nicht verständlich, aber das macht nichts. Schließe den Microcontroller über ein Micro-USB-Kabel an deinen PC an. Dann erscheint im Windows-Explorer das Laufwerk MICROBIT. Dorthin verschiebst du deinen hex-Code, und nach etwas Blinken auf der Rückseite läuft auch schon dein Programm auf ihm.

Probiere auch mit den anderen Befehlen aus der Kategorie Grundlagen herum. Was macht die Zeitangabe in pausiere? Klicke in die 5x5 Kästchen bei zeige LEDs!

Eingabe – digitale Werte

In der zweiten Kategorie findest du Befehle für Eingaben.

Wenn Knopf A gedrückt sollte wohl besser heißen „wenn Knopf A wieder losgelassen wird“, aber auf das Sprachgefühl der Entwickler haben wir so wenig Einfluss wie auch bei on pin P0 pressed: Da ist nichts zu pressen, sondern es geht um die Reihe der messingfarbenen Streifen an der Unterkante. Fünf davon sind breiter und haben Namen. Stelle einen Kontakt her zwischen Pin 0 und GND (Ground = Erde). Dafür kannst du ein Stück Draht oder eine Krokodilklemme nehmen. Vielleicht reichen auch schon zwei feuchte Finger. Spiele auch damit herum.
Schaffst du es, dass er statt der Zahl 0, wie im Bild rechts, die Temperatur ausgibt?

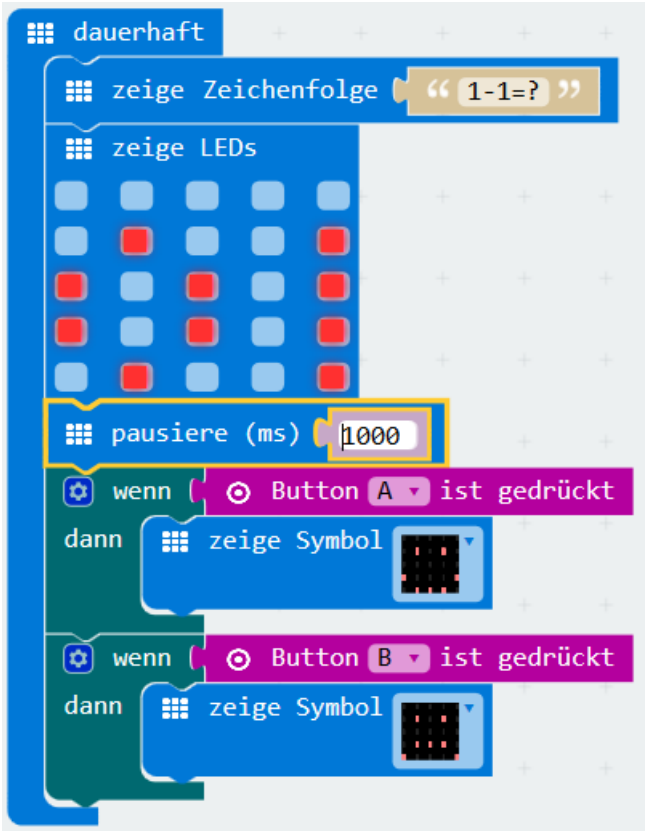


Logik

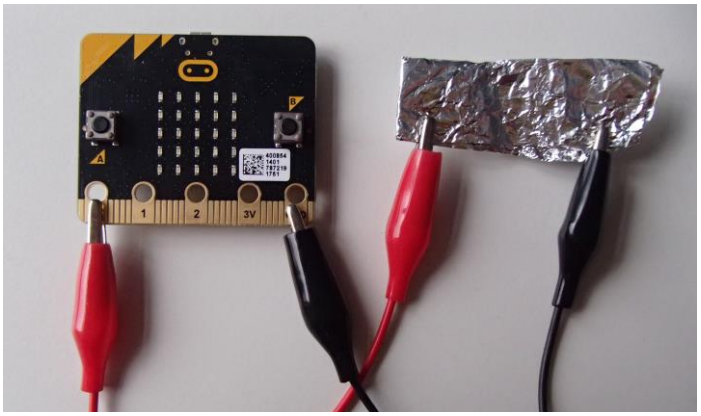
Wir programmieren ein Rechenspiel:
Erst wird eine Aufgabe gestellt. Dann schlägt er zwei Ergebnisse vor für Taste A und Taste B. Und dann soll er erkennen, ob du eine Taste gedrückt hast. Dazu brauchen wir einen Befehl aus der Kategorie Logik, nämlich wenn … dann. Mit Button ist der Knopf gemeint. Anschließend soll er wieder von vorne anfangen.

Beachte, dass die Knöpfe bei unserem Programm in dem Moment gedrückt sein müssen, wenn die Pause von 1000 Millisekunden gerade vorbei ist. Übrigens sind 1000 ms genau 1 s, so ähnlich wie 1000 mm genau 1 m sind.

Zum Testen musst du übrigens nicht immer das Programm auf den Microbit übertragen, es reicht oft auch, deine Idee in der Programmierumgebung auf dem PC laufen zu lassen. Die Tasten A und B beim virtuellen Microbit auf dem Bildschirm kannst du mit der Maus anklicken.



Beschleunigung, Magnetfeld



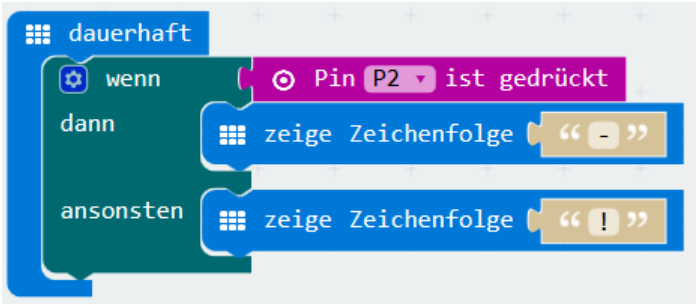
Probiere die Befehle selber aus und stelle sie uns in der nächsten Stunde vor!

. . . . . .

. . . . . .

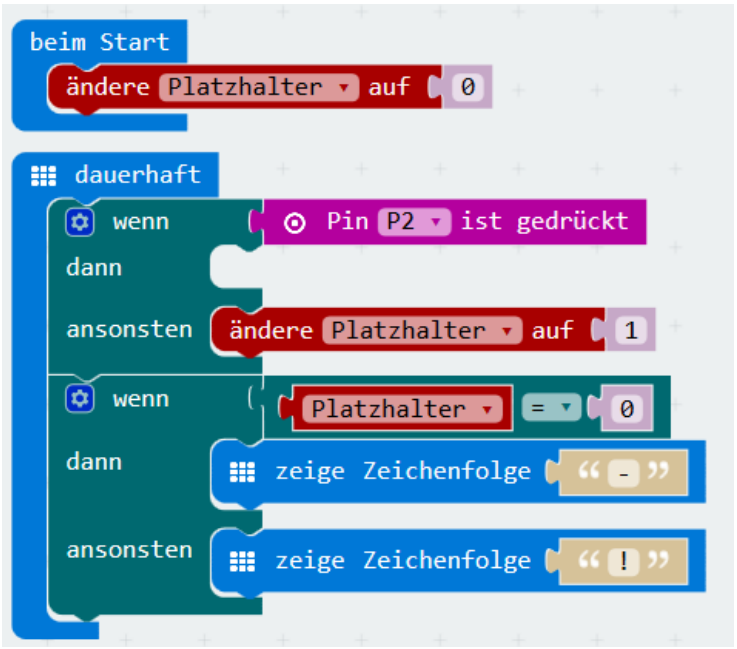
. . . . . .

Projekt: Alarmanalage



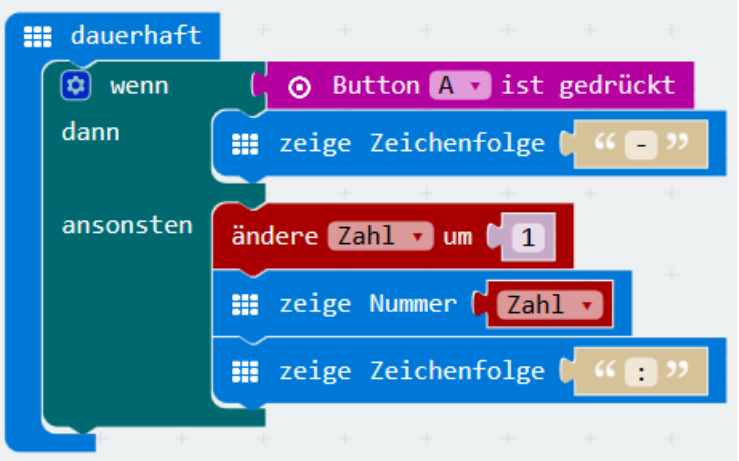
An die fünf großen Pins kannst du mit Krokodilklemmen Elektronik anschließen. Den passenden Befehl wie on pin 0 pressed kennen wir schon. Bau dir eine Alarmanlage, die anzeigt, wenn ein Schokoladenpapier zerrissen wurde! Mit diesem könntest du zum Beispiel deine Türe versiegeln.

Variablen



Jetzt soll der Alarm gespeichert bleiben, auch wenn das Schokoladenpapier wieder geflickt wird. Das heißt, der Mikrocontroller soll sich etwas merken können. Wir speichern die Information in eine Variable: Anfangs soll sie auf 0 gesetzt werden. Wenn der Kontakt geschlossen bleibt, soll sich daran nichts ändern, aber wenn er geöffnet wird, soll die Variable auf 1 gesetzt werden.
Die Anzeige soll dann nicht mehr (wie oben) vom Schokoladenpapier abhängen, sondern von der Variable: Bei 0 gibt es keinen Alarm, bei 1 schon. Kannst du das programmieren?
Beim nächsten Mal wählst du einen treffenderen Namen für die Variable, zum Beispiel DaWarWer.

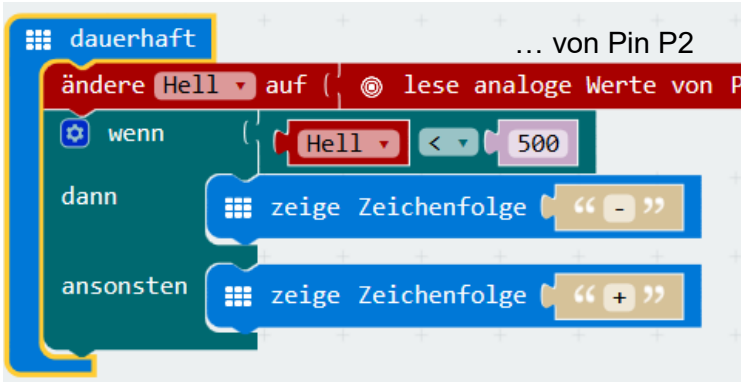
Projekt Stoppuhr



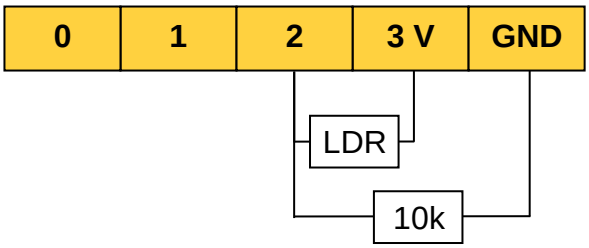
Programmiere eine Art Stoppuhr; sie soll hochzählen, solange die Türe geöffnet ist. Dabei hilft dir der Befehl ändere Zahl um 1. Der Takt muss für’s erste nicht jede Sekunde sein.

Helligkeit – ein analoger Wert

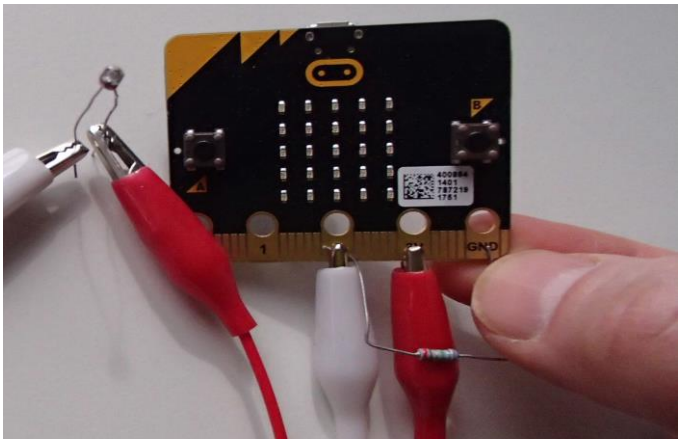
Helligkeit können wir mit einem LDR messen, das ist ein lichtempfindlicher Widerstand. Bei Helligkeit hat er ein paar Dutzend Ohm, bei Dunkelheit viele Hunderttausend. Um ihn zu verwenden, müssen wir ein wenig Elektronik lernen.
Bisher hat der Mikrocontroller nur erkennen sollen, ob ein Kontakt geschlossen war oder nicht, 1 oder 0, wahr oder falsch. Das nennt der Microbit *digital*. Er kann aber auch Zwischenwerte erkennen, nämlich Werte von 0 (= GND) bis 3 V in Stufen von 0 bis 1023. Das nennt er *analog*.



Jetzt bauen wir die Schaltung auf:
Wir verbinden GND und Pin 2 mit einem Widerstand von ungefähr 10 000 Ohm*.
Und wir verbinden Pin 2 und 3 V mit dem LDR.
Dann liefert Pin2 bei Helligkeit einen größeren Wert und bei Dunkelheit einen kleineren. Der Wert liegt zwischen 0 und 1023; wir vergleichen ihn mit einer voreingestellten Schwelle, zum Beispiel 500, und lassen anzeigen, ob er darüber oder darunter liegt, ob es also hell oder dunkel ist.



Aufgabe: Probiere verschiedene Werte für die Schwelle und verschiedene Helligkeiten aus!



Projekt: Bau einer Lichtschranke

Der Mikrocontroller soll Alarm geben, wenn jemand durch die offene Türe geht. Vielleicht brauchst du dazu noch eine Lampe?

*** Farbcode für die Widerstände**

Die vier Farbringe auf den Bauteilen geben den Wert der Widerstände an. Halte den goldenen Ring nach rechts. Dann zeigt der linke Ring die erste Ziffer, der nächste die zweite. Der dritte Ring gibt die Anzahl der folgenden Nullen an. Der goldene ist für uns nicht wichtig, er zeigt die Genauigkeit.

Beispiele:
bn gr bn = 18 · 10 = 180 [Ohm] (180 Ω)
bn sw or = 10 · 10^3 = 10 000 [Ohm] („10k“).

Bei mehr Ringen hältst du den breiteren nach rechts. Dann stehen die linken drei für die ersten drei Ziffern, der vierte für die Anzahl der Nullen.

Farbe	1. Ring Ziffer	2. Ring Ziffer	3. Ring Multiplikator
schwarz	—	0	10^0 = 1
braun	1	1	10^1 = 10
rot	2	2	10^2 = 100
orange	3	3	10^3 = 1.000
gelb	4	4	10^4 = 10.000
grün	5	5	10^5 = 100.000
blau	6	6	10^6 = 1.000.000
violett	7	7	10^7 = 10.000.000
grau	8	8	10^8 = 100.000.000
weiß	9	9	10^9 = 1.000.000.000