Handreichung PhC

1. Kurzvorstellung

Das Unterrichtsmaterial beinhaltet eine praktische Programmieraufgabe, bei der die Schülerinnen und Schüler ein Reaktionszeit-Messgerät mit dem Micro:bit programmieren. Das Material ist für Schülerinnen und Schüler der 8. Klasse konzipiert und beinhaltet zwei Aufgabenstellungen: Eine, bei der der Code bereits vorgegeben ist und eine offene Aufgabe, bei der die Schülerinnen und Schüler das Messgerät selbst programmieren sollen.

2. Einordnung in die Lehrpläne

Das Material passt in den Lernbereich 3: Komplexaufgabe des Lehrplans Informatik für das Gymnasium. Dabei werden die Schülerinnen und Schüler in die Lage versetzt, eine komplexere Programmieraufgabe zu bewältigen und ihre Fähigkeiten im Umgang mit dem Micro:bit zu erweitern.

Auszug aus dem Lehrplan Informatik für das Gymnasium:

Lernbereich 3: Komplexaufgabe	6 Ustd.
Anwenden informatischer Konzepte bei der Lösung einer Komplexaufgabe	Verknüpfung von Lernbereichen der Klassenstufen 7 und 8 Gestaltung eines fächerverbindenden oder schulübergreifenden Projektes Einbeziehung externer Partner Programmierung der Hardware der Schüler Erstellung eines multimedialen Wiki oder Blogs mit einem Content Management System Einrichtung eines mobilen Endgerätes ⇒ Problemlösestrategien → Kl. 7, LB 1
- Nutzung verschiedener Anwendungen - Arbeitsmethodik	Kooperieren beim Implementieren der Lösung Reflektieren des Arbeitsprozesses

3. Lernziele

Die Lernziele des Materials sind vielfältig. Kognitive Lernziele beinhalten das Verstehen und Anwenden von Programmierkonzepten wie Schleifen, Bedingungen und Variablen. Psychomotorische Lernziele umfassen das Erlernen der Bedienung des Micro:bit und das Schreiben von Programmcode. Affektive Lernziele beziehen sich auf die Motivation und das Interesse der Schülerinnen und Schüler an der Programmierung sowie auf die Entwicklung des Problemlösungsverhaltens.

Das Erreichen folgender Lernziele kann mit Hilfe des Materials verfolgt werden:

Kognitive Lernziele:

- Die Schülerinnen und Schüler können die konkrete Funktionsweise von Variablen anhand von Code im MakeCode Editor erklären und diese in ihrem Programmcode anwenden.
- Die Schülerinnen und Schüler können Bedingungen in ihrem Programmcode formulieren und damit auf bestimmte Ereignisse reagieren (Bsp: Tasten am Micro:Bit drücken).

- Die Schülerinnen und Schüler können If bzw. If-Else Bedingungen in ihrem Programmcode nutzen, um das Auftreten unterschiedlicher Ereignisse (Messungen starten & beenden, Fehlstart erkennen) sinnvoll zu verarbeiten.
- Die Schülerinnen und Schüler können den Code eines vorgegebenen Programms verstehen, modifizieren und eigene Ideen einbringen.

Psychomotorische Lernziele:

- Die Schülerinnen und Schüler bedienen den Micro:bit und nutzen die Funktionen der Tasten und Pins zur Ein- und Ausgabe für das Reaktionszeitmessgerät und können ihre Ergebnisse auf dem Micro:bit bzw. Display anzeigen und interpretieren.
- Die Schülerinnen und Schüler können den MakeCode Editor nutzen, um ihren Programmcode zu schreiben, zu testen und auf den Micro:bit zu übertragen.
- Die Schülerinnen und Schüler bauen die verschieden Komponenten (Micro:bit, Steckbrett, Display, Extension Borad, Überbrückungs- bzw. Verbindungskabel) zusammen

Affektive Lernziele:

- Die Schülerinnen und Schüler entwickeln eine positive Einstellung gegenüber dem Programmieren und erkennen dessen Nutzen und Anwendungsmöglichkeiten.
- Die Schülerinnen und Schüler entwickeln das Vertrauen in ihre eigenen Fähigkeiten zur Problemlösung und Programmierung.
- Die Schülerinnen und Schüler entwickeln ein Verständnis für den sorgsamen und verantwortungsvollen Umgang mit technischen Geräten und Materialien und erkennen, dass dies eine wichtige Voraussetzung für einen sicheren und zuverlässigen Einsatz ist.

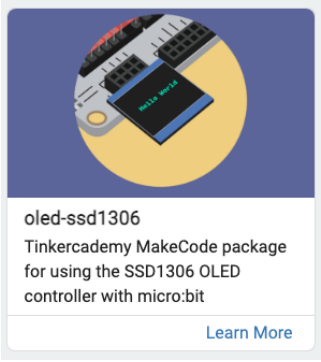
4. Voraussetzungen

- Die **fachlichen Voraussetzungen** für die Nutzung dieses Unterrichtsmaterials sind grundlegende Kenntnisse im Bereich der Programmierung, insbesondere im Umgang mit Blockprogrammierung und der Implementierung von einfachen Kontrollstrukturen. Die Schülerinnen und Schüler sollten in der Lage sein, grundlegende Konzepte wie Variablen, Bedingungen und Schleifen zu verstehen und anzuwenden. Das bedeutet, der Lernbereich 1: Algorithmen sollte bereits behandelt worden sein.

- **Technische Voraussetzungen** sind ein Micro:bit pro Schülergruppe, sowie ein Computer oder Tablet mit Internetzugang und einem aktuellen Browser (z.B. Google Chrome), auf dem der MakeCode Editor ausgeführt werden kann. Die Schülerinnen und Schüler sollten auch in der Lage sein, das USB-Kabel zum Anschließen des Micro:bit an den Computer zu verwenden.

- **Materielle Voraussetzungen** sind neben den Micro:bit-Boards und dem Computer oder Tablet folgende zusätzliche Materialien:

  - 1x Micro:bit V2
  - 1x Batteriehalter inkl. Batterien
  - 1 USB/Micro-USB Kabel
  - 1x Micro:bit Extension Board
  - 1x Steckbrett (Breadboard)
  - 1x 0.96 Inch OLED Display
  - 4x Überbrückungs- bzw. Verbindungskabel
  - benötigte Erweiterung: oled-ssd1306; nach hinzufügen erscheint folgender Menüpunkt im Editor:



5. Kurzdarstellung

Das Material besteht aus zwei Aufgabenstellungen, bei denen die Schülerinnen und Schüler entweder vorgegebenen Code ergänzen oder ein eigenes Reaktionszeit-Messgerät programmieren. Darüber hinaus gibt es Vorschläge für den Einsatz in anderen Fächern wie Mathematik und Physik. Das Material ist so konzipiert, dass es in einer Projektwoche oder fächerübergreifend eingesetzt werden kann. Es werden keine besonderen Materialien benötigt, da der Micro:bit als Hardware und der MakeCode Editor als Software verwendet werden.

1. Aufgabenstellungen:

Aufgabenbeispiel I: Code vorgeben und Reflektieren

1) Öffne den MakeCode Editor und lade das Programm „Microbit-Reaktionszeitmessung.hex“ in den Editor. Lade das Programm und probiere es aus:

   Drücke dazu den Knopf, sobald eine Raute auf dem Microbit erscheint. Drücke nach Erscheinen des Ausrufezeichens so schnell wie möglich Knopf B. Die Zahl die anschließend auf dem Display erscheint ist die Reaktionszeit in Millisekunden.

2) Führe fünf Reaktionszeitmessungen nacheinander aus. Was ist deine Bestzeit? Wie lange brauchst du im Durchschnitt, um zu reagieren? Notiere dir deine Bestzeit und deine durchschnittliche Antwortzeit

3) Schau dir nun den Code genauer an und beantworte folgende Fragen. Mache dir dazu Notizen in dein Heft:
   a. Welche Teile des Codes sind für die Ein- bzw. Ausgabe zuständig?
   b. Wie viele Variablen werden benutzt? Gib Name und Typ der Variablen an.
   c. Welche Kontrollstrukturen werden verwendet?
   d. Was bedeutet der Ausdruck „show string“?

4) Aktuell wird die Reaktionszeit in Millisekunden auf dem Display angezeigt, jedoch fehlt die Einheit. Ändere den Code so, dass nach der Zahl ein Leerzeichen folgt und die Einheit „ms“ erscheint.

5) In Teilaufgabe 2) hast du die durchschnittliche Reaktionszeit selbst berechnet. Ergänze den Code so, dass nach fünf Versuchen automatisch die durchschnittliche Reaktionszeit angezeigt wird.

Anwendungsbeispiel II: Offene Aufgabe & Spielerische Motivation:

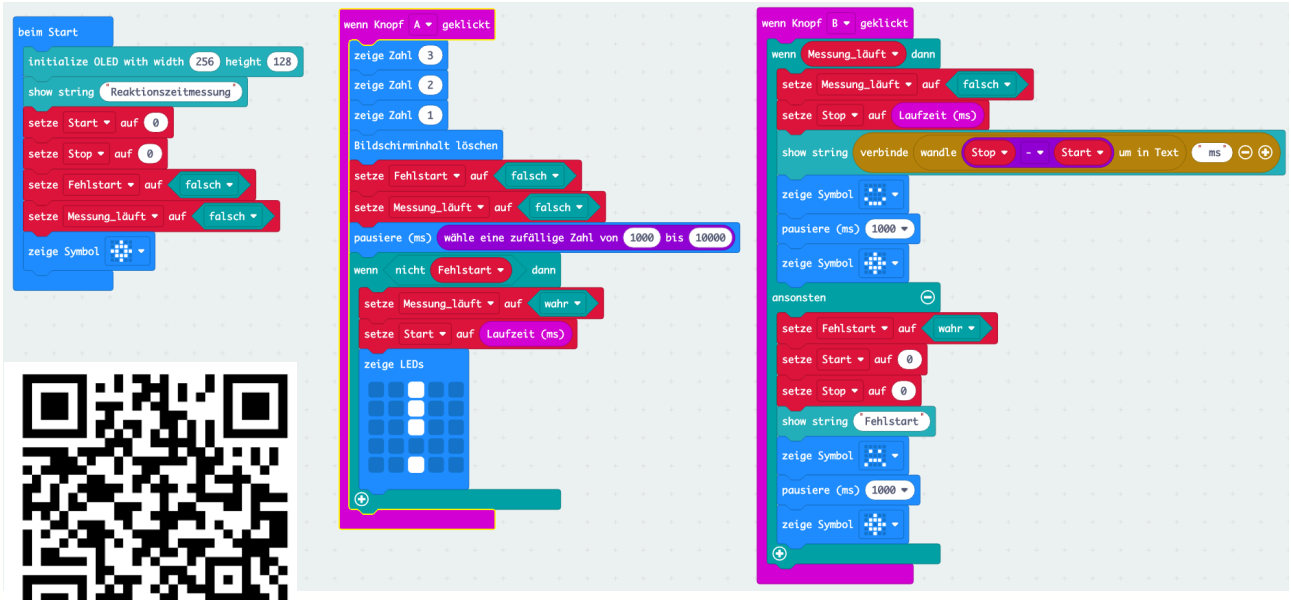
„FIFA 18, Minecraft oder nur mal schnell eine Runde Clash Royale? Video Games machen Spaß und sind manchmal eine willkommene Ablenkung vom harten Schulalltag. Und dann gibt es noch viele weitere Vorteile: etwa die Reaktionszeit, die dadurch geschult wird – oder etwa doch nicht? Du als Zockerin oder Zocker bist den Nichtzockerinnen oder Nichtzockern in deiner Reaktion doch sicher voraus?

Das schreit nach einem Praxistest! Wie das gehen soll? Du bastelst einfach ein eigenes Reaktionszeit-Messgerät und forderst Lehrerinnen und Lehrer, Eltern, Mitschülerinnen und Mitschüler heraus.“

(https://microbit.eeducation.at/wiki/Reaktionszeit-Messger%C3%A4t#:~:text=Durch%20die%20Ber%C3%BCh-rung%20wird%20%C3%BCber,f%C3%BCr%20deine%20sensationelle%20Reaktionszeit%20feiern!)

2. Lösungen:

MakeCode – Aufgabenbeispiel I – Code für Schüleraufgabe (Screenshot & JavaScript):

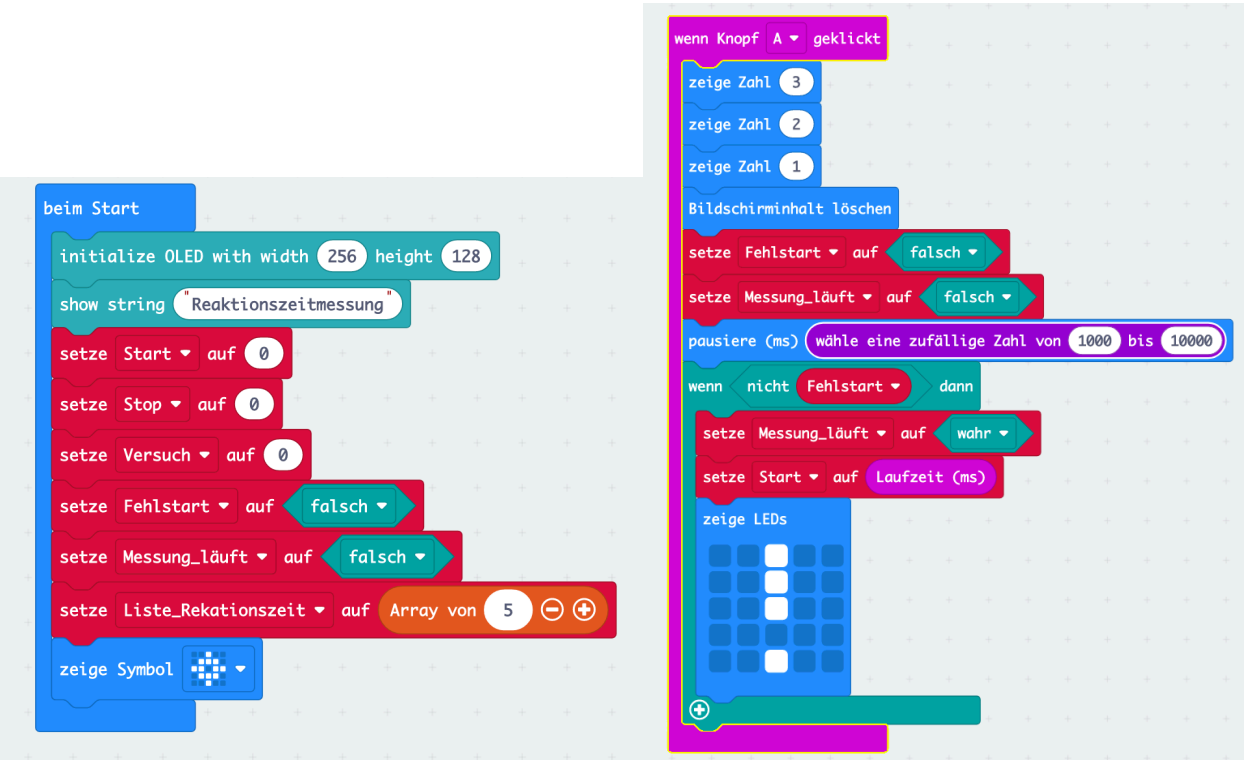


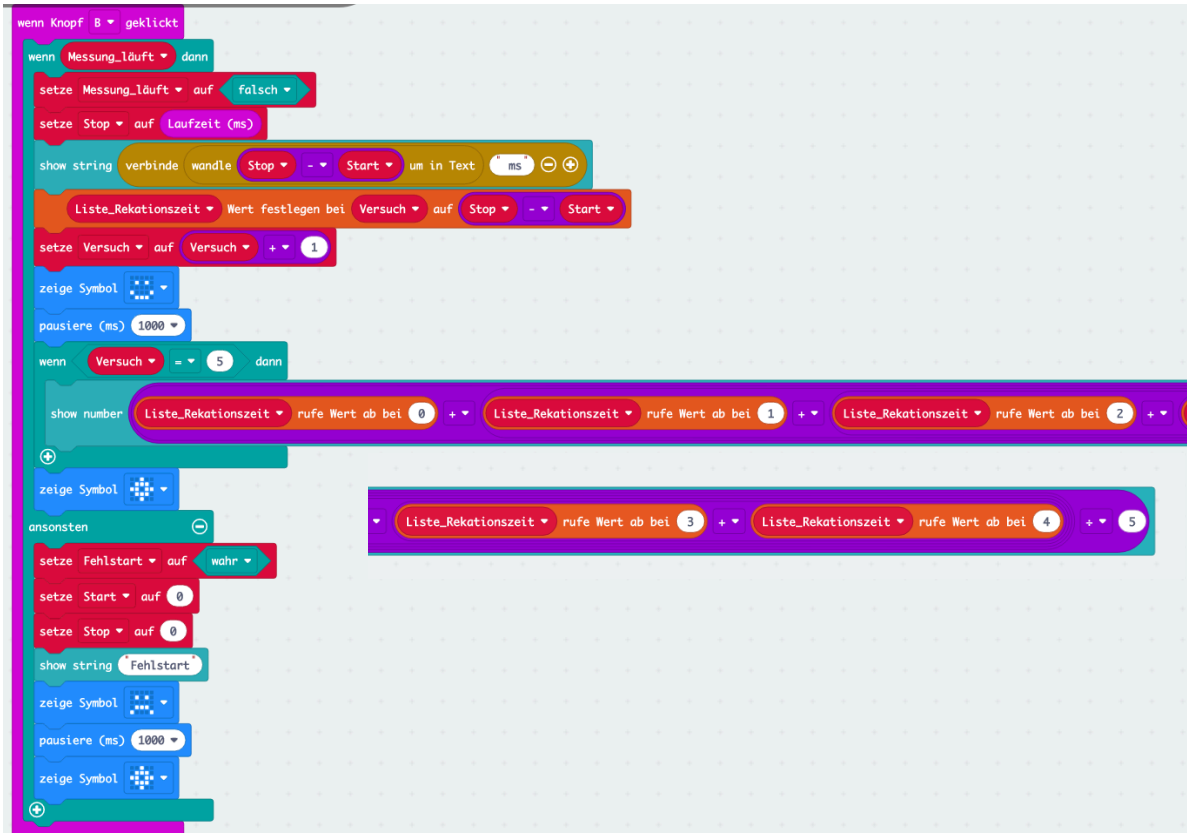

```
input.onButtonPressed(Button.A, function () {
    basic.showNumber(3)
    basic.showNumber(2)
    basic.showNumber(1)
    basic.clearScreen()
    Fehlstart = false
    Messung_läuft = false
    basic.pause(randint(1000, 10000))
    if (!(Fehlstart)) {
        Messung_läuft = true
        Start = input.runningTime()
        basic.showLeds(`
            . . # . .
            . . # . .
            . . # . .
            . . . . .
            . . # . .
            `)
    }
})
input.onButtonPressed(Button.B, function () {
    if (Messung_läuft) {
        Messung_läuft = false
        Stop = input.runningTime()
        OLED.writeStringNewLine("" + convertToText(Stop - Start) + "")
        basic.showIcon(IconNames.Happy)
        basic.pause(1000)
        basic.showIcon(IconNames.Target)
    } else {
        Fehlstart = true
        Start = 0
```

```
        Stop = 0
        OLED.writeStringNewLine("Fehlstart")
        basic.showIcon(IconNames.Sad)
        basic.pause(1000)
        basic.showIcon(IconNames.Target)
    }
})
let Messung_läuft = false
let Fehlstart = false
let Stop = 0
let Start = 0
OLED.init(256, 128)
OLED.writeStringNewLine("Reaktionszeitmessung")
Start = 0
Stop = 0
Fehlstart = false
Messung_läuft = false
basic.showIcon(IconNames.Target)
```

Code-Schnipsel als Lösung für Teilaufgabe 5:

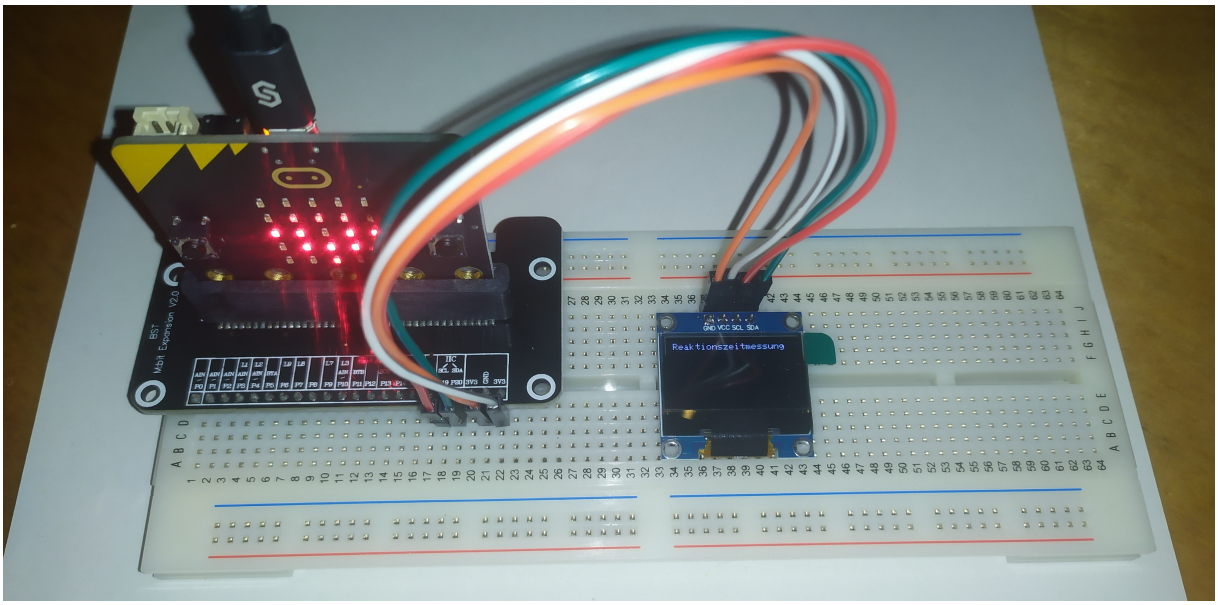
Vollständige Lösung (Screenshot & JavaScript)::





```
input.onButtonPressed(Button.A, function () {
    basic.showNumber(3)
    basic.showNumber(2)
    basic.showNumber(1)
    basic.clearScreen()
    Fehlstart = false
    Messung_läuft = false
    basic.pause(randint(1000, 10000))
    if (!(Fehlstart)) {
        Messung_läuft = true
        Start = input.runningTime()
        basic.showLeds(`
            . . # . .
            . . # . .
            . . # . .
            . . . . .
            . . # . .
            `)
    }
})
input.onButtonPressed(Button.B, function () {
    if (Messung_läuft) {
        Messung_läuft = false
        Stop = input.runningTime()
        OLED.writeStringNewLine("" + convertToText(Stop - Start) + " ms")
        Liste_Rekationszeit[Versuch] = Stop - Start
        Versuch = Versuch + 1
        basic.showIcon(IconNames.Happy)
        basic.pause(1000)
        if (Versuch == 5) {
            OLED.writeNumNewLine((Liste_Rekationszeit[0] + (Liste_Rekationszeit[1] + (Liste_Rekati-
onszeit[2] + (Liste_Rekationszeit[3] + Liste_Rekationszeit[4])))) / 5)
        }
        basic.showIcon(IconNames.Target)
    } else {
        Fehlstart = true
        Start = 0
        Stop = 0
        OLED.writeStringNewLine("Fehlstart")
        basic.showIcon(IconNames.Sad)
        basic.pause(1000)
        basic.showIcon(IconNames.Target)
    }
})
let Liste_Rekationszeit: number[] = []
let Messung_läuft = false
let Fehlstart = false
let Versuch = 0
let Stop = 0
let Start = 0
OLED.init(256, 128)
OLED.writeStringNewLine("Reaktionszeitmessung")
Start = 0
Stop = 0
Versuch = 0
Fehlstart = false
Messung_läuft = false
Liste_Rekationszeit = [5]
basic.showIcon(IconNames.Target)
```

Aufbau Aufgabe 1:



Verbindung von OLED Display und Mircobit:

Micro:bit	OLED
3V3	VCC
GND	GND
P20	SDA
P19	SCL

3. Ideen für weitere Anwendungsmöglichkeiten:

- Reflektion zur Codeoptimierung: Lassen sich bestimmte Teile des Codes verbessern?
  - Einsatz von Funktionen (insbesondere Teilaufgabe 5)
  - Codestruktur: Modularisierung
  - Zusammenarbeit mit anderen Fachbereichen:
  - Mathematik – Funktionen / Datenauswertung
  - Physik – Messung physikalischer Größen
- Wiederverwendung in älteren Klassenstufen
  - Lässt sich das Programm in anderen Programmiersprachen besser umsetzen?
- Alternative: Anwendung als Spiel: (in Anlehnung an „MICRO:BIT – Das Schulbuch“)