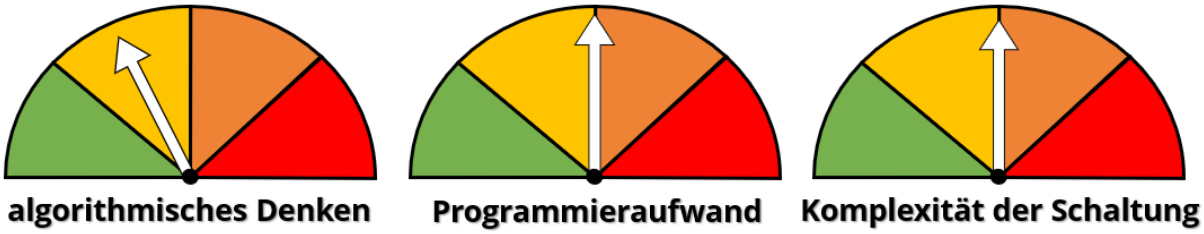
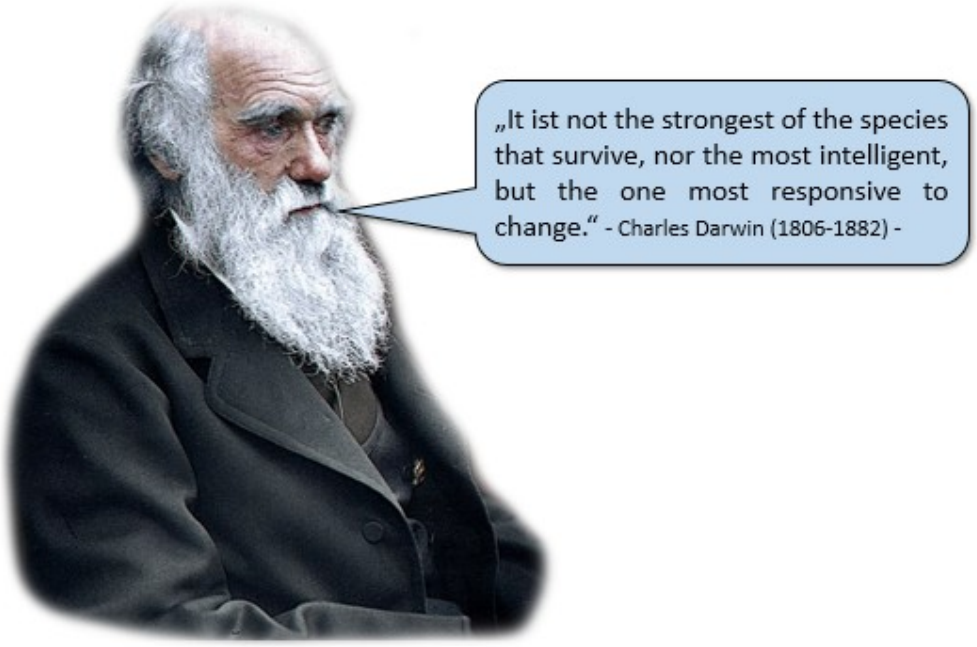
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 3 - ZUSATZ| It‘s the survival of the fittest!





Professur für Didaktik der Informatik

SYLBER Synergetische Lehrerbildung im exzellenten Rahmen

Das Maßnahmenpaket „TUD-Sylber² – Synergetische Lehrerbildung im exzellenten Rahmen“ wird im Rahmen der gemeinsamen „Qualitätsoffensive Lehrerbildung“ von Bund und Ländern aus Mitteln des Bundesministeriums für Bildung und Forschung gefördert.

GEFÖRDERT VOM Bundesministerium für Bildung und Forschung

ZUSATZAUFGABE – TEIL 1 – FEHLER FINDEN UND BEHEBEN

Du hast noch kein Problem gefunden? Nicht schlimm – probiere einfach folgendes aus: Nach dem Ausschalten der roten LED drückt zunächst Spieler 1 und erst danach Spieler 2! Welches Verhalten kannst du beobachten? **Schreibe es auf!**

Wir brauchen also irgendeine Lösung, welche es verhindert, dass die Verzweigungsbedingungen `wahr` beziehungsweise `true` werden, obwohl ein Spieler oder eine Spielerin bereits gewonnen hat. Das nutzen wir eine sogenannte **boolesche Variable**.

BOOLESCHE VARIABLEN

Übersetzt bedeutet das Wahrheitswert, welche entweder `TRUE` (wahr) oder `FALSE` (falsch) sein kann, beziehungsweise in der Sprache eines Computers 1 oder 0.

Das heißt jetzt im Umkehrschluss, wir überprüfen bei den Verzweigungen zusätzlich das noch keiner gewonnen hat. Sobald jemand gewinnt, ändern wir den Variable `sieg` auf den Wert `true`, damit merkt sich das Programm das es bereits einen Sieg gibt. Nachfolgend wird dir das an einem Beispiel gezeigt, du musst dies dann nur noch auf die anderen Prozeduren und Verzweigungen anwenden.

Initialisierung:
```
boolean sieg = false;
```

Prozedur:
```
void siegSpielerEins () {
     sieg = true;
}
```

Verzweigung:
```
if(digitalRead(ButtonSpielerEins) == HIGH
&& sieg == false) {
     siegSpielerEins();
}
```

ZUSATZAUFGABE 1 - PROBLEMLÖSUNG

Initialisiere die Variable, ändere die Prozeduren und ergänze die Verzweigungsbedingungen. Teste im Anschluss ob das Problem behoben wurde!

ZUSATZAUFGABE – TEIL 2 – ZEITMESSUNG

Nachdem das Programm nun einwandfrei funktioniert, wollen wir noch eine kleine Zusatzfunktion einbauen. Und zwar wollen wir die Zeit zwischen ausschalten der roten LED und drücken des erstens Buttons messen. Dies soll dann über den seriellen Monitor ausgegeben werden.

Wie können wir das realisieren? Wenn wir wissen, wie lange die `loop()`-Methode benötigt, könnten wir einfach zählen wie oft diese sich wiederholt. Dies könnten wir dann mit der Zeitdauer multiplizieren und wir hätten unsere Zeit.

WIE LANG BRAUCHT DIE `LOOP()`-METHODE FÜR EINEN DURCHLAUF?

Um diese Frage näherungsweise zu beantworten müssen wir einen Blick auf das Herzstück des Arduinos werden – den Prozessor. Dieser hat eine Taktfrequenz von $16\,MHz$. Das bedeutet, der Mikrocontroller verarbeitet **16 Millionen** Instruktionen / Befehle pro Sekunde.

Das heißt, die Verarbeitung ist so unglaublich schnell, dass die wenigen Befehle in kürzester Zeit abgearbeitet sind. Wie können wir jetzt aber die Zeit messen? Dazu halten wir das Programm für 1 Millisekunde an. Das wir dies mit der `delay()`-Funktionen machen können weißt du ja bereits. Wir benötigen als eine Variable in der wir die Zeit in Sekunden speichern. Es gilt als eine Zahl mit Nachkommastellen zu speichern.

ACHTUNG

Du darfst die Variable nicht, wie wir es die ganze Zeit gemacht haben, als Ganzzahl mit `int` definieren. Du brauchst eine Gleitkommazahl mit Nachkommastellen. Nutze daher den Datentyp `float`.

Diese Variable initialisieren wir mit dem Startwert 0. Innerhalb der `loop()`-Methode stoppen wir das Programm für 1 Millisekunden und addieren eine Millisekunde zu der aktuellen hinzu. Dies funktioniert wie folgt:

```
zeit = zeit + 0.001; //Addieren von 1 Millisekunde
delay(1); //1 Millisekunde warten
```

Achte bei Kommazahlen darauf, dass Programmieren der Punkt und nicht das Komma verwendet wird! Was jetzt noch fehlt ist nur noch die **Ausgabe** im seriellen Monitor. Die Prozeduren geben ja bereits aus, welcher Spieler gewonnen hat. Dies musst du nur noch ergänzen um die Ausgabe der Sekunden. Beispielhaft ist dies unten gezeigt.

```
Serial.print("Sieg für beide Spieler! Zeitdauer: ");
Serial.print(zeit);
//Hier wird der Inhalt der Variable ausgegeben
Serial.println(" Sekunden.");
```

ZUSATZAUFGABE 2 – ZEITMESSUNG ERGÄNZEN

Nun gilt es dein Programm anzupassen: Initialisiere die Variable, schreibe den notwendigen Programmcode in die `loop()`-Methode und ergänze die Prozeduren um die Ausgabe der Zeitdauer.

Teste dein Programm aus und vergleicht eure Reaktionszeiten!

Grafik auf dem Deckblatt: *Charles Darwin, Juliuas Jääskeläinen, CC BY-SA 2.0, https://creativecommons.org/licenses/by/2.0/deed.en, https://commons.wikimedia.org/wiki/File:Charles_Darwin,_English_naturlist,_colored.jpg*

Alle weiteren Grafiken: *Patrick Binkert, EduInf@TUD*