Handreichung Physical Computing

-Ein Merkspiel mit dem Arduino Mikrocontroller-

Ferdinand Probst

1. Kurzvorstellung

Bei dem hier dargestellten Physical Computing-Projekt handelt es sich um ein Merkspiel. Dabei muss das Blinken von LEDs in der korrekten Reihenfolge nachgeahmt werden. Nach jeder erfolgreichen Runde verlängert sich die zu wiederholende Sequenz. Wird ein Fehler gemacht, startet das Spiel von vorn.

2. Einordnung in die Lehrpläne

Schulart	**Jahrgangsstufe**	**LB**	**Inhalt**
Gymnasium	10	1	-Beherrschen der Implementierung der algorithmischen Grundstrukturen -Kennen des Prinzips der Modularisierung
	11/12 GK	2	-Beherrschen der Implementierung strukturierter Datentypen in einer Programmierumgebung -Beherrschen der Arbeit mit Unterprogrammen
Fachoberschule	12	2	-Anwenden ausgewählter Problemlösungsstrategien in einer höheren Programmiersprache
Berufliches Gymnasium	12/13	3	-Beherrschen ausgewählter Problemlösestrategien in einer höheren Programmiersprache

3. Lernziele

Kognitive Lernziele

Die Schüler*innen vollziehen die Implementierung algorithmischer Grundstrukturen anhand eines Merkspiels mit einem Arduino nach.

Die Schüler*innen lernen das Prinzip der Modularisierung an einem Merkspiel mit einem Arduino kennen.

Psychomotorische Lernziele

Die Schüler*innen implementieren algorithmische Grundstrukturen, indem sie ein Merkspiel auf einem Arduino programmieren.

Die Schüler*innen implementieren den strukturierten Datentyp „Feld“ in einem Merkspiel auf einem Arduino.

Die Schüler*innen verwenden Unterprogramme, um ein Merkspiel auf einem Arduino zu implementieren.

Affektive Lernziele

Die Schüler*innen nehmen den Arduino-Mikrocontroller als ein interessantes und vielfältig einsetzbares Medium wahr.

Die Schüler*innen werden zur Umsetzung eigener Projekte auf dem Arduino-Mikrocontroller motiviert.

4. Voraussetzungen

Um das Projekt umzusetzen, wird neben einem Computer zur Programmierung folgendes benötigt:

1x Arduino Mikrocontroller (mind. 8 digital Input/Output)

1x Steckplatine

31x Kabel (Farbe und Länge ähnlich Schaltplan unter **5. Kurzdarstellung**, um Übersichtlichkeit zu erhöhen)

8x Widerstand 10kOhm

8x LED (2x rot, gelb, grün, blau)

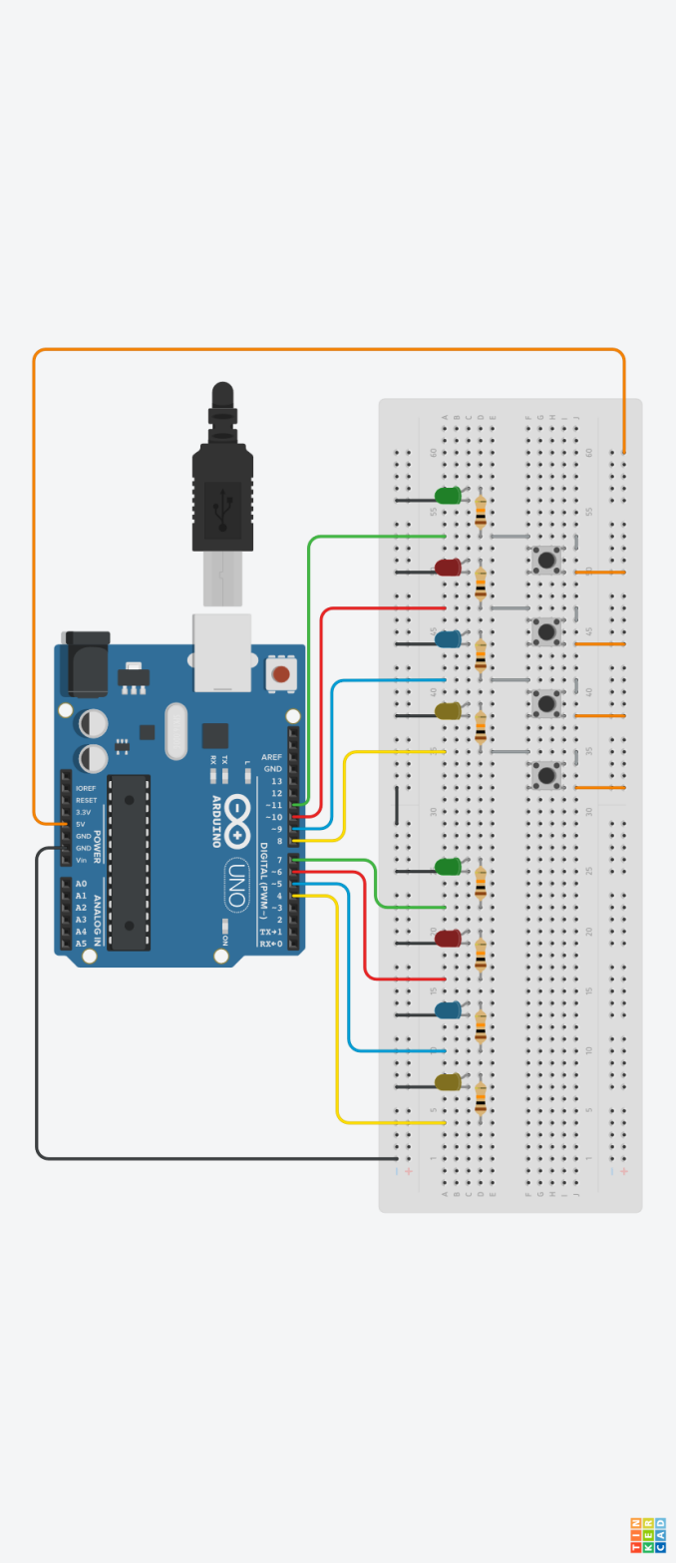
4x Drucktaster

5. Kurzdarstellung

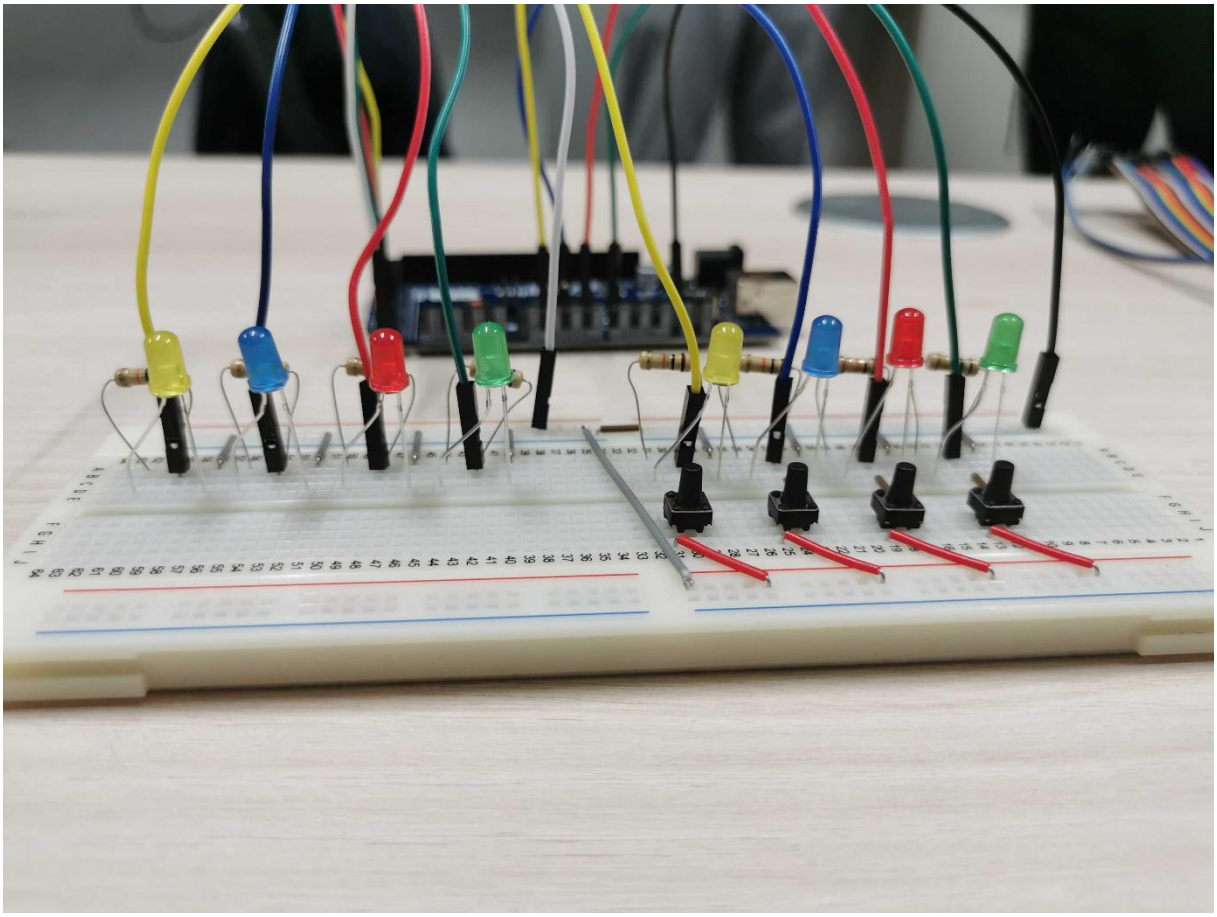
Aufbau

Hinweis: die vier linken digitalen Pins (4, 5, 6, 7) des Schaltplans sind auf *OUTPUT* geschaltet, die vier rechten (8,9,10,11) auf *INPUT* (siehe Quelltext). Natürlich können die Pinnummern je nach Arduino-Board frei gewählt werden.

Schaltplan



Umsetzung (Beispiel)



Quelltext

Hinweis: Zur Programmierung des Arduino empfiehlt sich der Einsatz der Arduino IDE (https://www.arduino.cc/en/software). Eine Referenz für die Programmiersprache findet sich unter https://www.arduino.cc/reference/en/.

```
//Belegung der Pins
int in_y = 8;
int in_b = 9;
int in_r = 10;
int in_g = 11;

int out_y = 4;
int out_b = 5;
int out_r = 6;
int out_g = 7;

//In- und Outputsequenz als Feld
int sequence_o[] = {0,0,0,0,0,0,0,0,0,0};
int sequence_i[] = {0,0,0,0,0,0,0,0,0,0};
```

```
//Hilfsvariablen
bool next = false;
bool failed = false;

void setup() {
  //Schalten der Pins auf Input und Output
  pinMode(in_y, INPUT);
  pinMode(in_b, INPUT);
  pinMode(in_r, INPUT);
  pinMode(in_g, INPUT);

  pinMode(out_y, OUTPUT);
  pinMode(out_b, OUTPUT);
  pinMode(out_r, OUTPUT);
  pinMode(out_g, OUTPUT);

  //Initialisierung der Konsole für Debug
  Serial.begin(9600);

  //Erzeugung eines Seeds für eine zufällige Blinksequenz
  randomSeed(analogRead(0));
}

void loop() {
  failed = false;

  //Initialisierung der zufälligen Blinksequenz
  for (int i = 0; i < 10; i++){
    sequence_o[i] = random(1,5);

    //Ausgabe auf Konsole für Debug
    Serial.print(sequence_o[i]);
  }

  Serial.print(" ");

  //Start des Spiels
  for (int i = 0; i < 10; i++){
    if(!failed){

      //Ausgabe der aktuellen Sequenz
      for (int j = 0; j <= i; j++){
        if(sequence_o[j] == 1){
          digitalWrite(out_y, HIGH);
          delay(300);
          digitalWrite(out_y, LOW);
          delay(200);
        }
        if(sequence_o[j] == 2){
          digitalWrite(out_b, HIGH);
          delay(300);
          digitalWrite(out_b, LOW);
```

```
    delay(200);
  }
  if(sequence_o[j] == 3){
    digitalWrite(out_r, HIGH);
    delay(300);
    digitalWrite(out_r, LOW);
    delay(200);
  }
  if(sequence_o[j] == 4){
    digitalWrite(out_g, HIGH);
    delay(300);
    digitalWrite(out_g, LOW);
    delay(200);
  }
}

//Eingabe der aktuellen Sequenz
for (int j = 0; j <= i; j++) {
  next = false;
  while(!next) {
    if(!next && digitalRead(in_y)){
      sequence_i[j] = 1;
      next = true;
      delay(500);
    }
    if(!next && digitalRead(in_b)){
      sequence_i[j] = 2;
      next = true;
      delay(500);
    }
    if(!next && digitalRead(in_r)){
      sequence_i[j] = 3;
      next = true;
      delay(500);
    }
    if(!next && digitalRead(in_g)){
      sequence_i[j] = 4;
      next = true;
      delay(500);
    }
  }
  Serial.print(sequence_i[j]);
}
Serial.print(" ");

//Abgleich der Eingabe- und Ausgabesequenz
for (int j = 0; j <= i; j++){
  if(sequence_o[j] != sequence_i[j]){
    failed = true;
  }
}
```

```
      //Signal, dass Fehler gemacht wurde und Spiel neu startet
      if(failed){
        for (int j = 0; j <= 3; j++){
          digitalWrite(out_y, HIGH);
          digitalWrite(out_b, HIGH);
          digitalWrite(out_r, HIGH);
          digitalWrite(out_g, HIGH);
          delay(300);
          digitalWrite(out_y, LOW);
          digitalWrite(out_b, LOW);
          digitalWrite(out_r, LOW);
          digitalWrite(out_g, LOW);
          delay(300);
        }
        delay(1000);
      }
    }

  }

  //Signal, dass längste Sequenz (10) erfolgreich wiederholt wurde
  //und das Spiel neu startet
  if(!failed){
    for (int j = 0; j <= 3; j++){
      digitalWrite(out_y, HIGH);
      delay(150);
      digitalWrite(out_y, LOW);
      digitalWrite(out_b, HIGH);
      delay(150);
      digitalWrite(out_b, LOW);
      digitalWrite(out_r, HIGH);
      delay(150);
      digitalWrite(out_r, LOW);
      digitalWrite(out_g, HIGH);
      delay(150);
      digitalWrite(out_g, LOW);
    }
    delay(1000);
  }

}
```

6. Einsatz- und Modifikationsmöglichkeiten

Das Merkspiel kann auf verschiedene Arten im Unterricht zum Einsatz kommen. Einige ausgewählte werden im Nachfolgenden dargestellt.

Motivation

Das Spiel kann zu Demonstrationszwecken eingesetzt werden, um den Schüler*innen ein motivierendes Beispiel für den Einsatz von höheren Programmiersprachen und Mikrocontrollern zu bieten. Dabei kann das Spiel herumgegeben oder vor der gesamten Klasse demonstriert werden, wobei ausgewählte Schüler*innen die aktuelle Sequenz ansagen oder eingeben.

Lehren von algorithmischen Grundstrukturen, Unterprogrammen, etc.

Durch Veränderungen am Quelltext können bei gleichem physischen Aufbau gezielt Inhalte wie algorithmische Grundstrukturen, Unterprogramme und strukturierte Datentypen von der Lehrkraft am Beispiel demonstriert werden. Dabei sollte der Quelltext so aufbereitet werden, dass er keine erst später zu vermittelnden Inhalte enthält. Beispielsweise enthält der unter **5. Kurzdarstellung** gegebene Quelltext Arrays, die im Gymnasium erst in der Klassenstufe 11/12 behandelt werden. Für eine Verwendung in Klassenstufe 10 sollte er also angepasst und vereinfacht werden. Beispielsweise könnte das Merkspiel statt 10 Runden nur 4 Runden umfassen und die Eingabe- und Ausgabesequenz in einzelnen Variablen gespeichert werden.

Schrittweise Erarbeitung eines komplexen Projekts

Um die Fähigkeit zur Implementierung von Programmen zu fördern und den Schüler*innen Erfolgserlebnisse zu bieten, ist es denkbar, nach der Behandlung von Inhalten der Algorithmierung und Programmierung das Merkspiel in Kleingruppen schrittweise nachzubauen. Dabei sollte nicht nur der Quelltext, sondern auch der physische Aufbau von Stunde zu Stunde erweitert werden.

Zum Beispiel könnten im Unterricht in der Klassenstufe 10 nacheinander die algorithmischen Grundstrukturen wiederholt und deren Umsetzung in der Arduino-Programmiersprache (angelehnt an C/C++) eingeführt werden, und das Projekt nach jeder neuen Grundstruktur erweitert werden. Später können dann bestimmte Programmteile in Unterprogramme ausgelagert werden, wie etwa die LED-Signale. In Klassenstufe 11/12 kann dann das Projekt durch die Verwendung von Arrays finalisiert werden.

Auch die Hardware sollte in einem solchen Szenario schrittweise aufgebaut werden. Ist beispielsweise erst die Implementierung einer Sequenz auf dem Arduino bekannt, kann zuerst das Blinken einer LED realisiert werden. In diesem Fall wird dann zuerst nur die linke gelbe LED angeschlossen. Eingaben können dann zu einem späteren passenden Zeitpunkt realisiert werden, indem zuerst nur die zugehörige gelbe LED auf der rechten Seite mit einem Drucktaster angeschlossen wird.

Auf diese Weise können die Hardware und Software schrittweise basierend auf den Kenntnissen der Schüler*innen über mehrere Stunden (und eventuell Schuljahre) erarbeitet werden. Die Schüler*innen haben so ein Ziel vor Augen und können erworbene Kenntnisse in einem komplexen Zusammenhang direkt praktisch umsetzen.

7. Erklärung der Freigabe zur Nachnutzung der Handreichung

Hiermit erkläre ich, Ferdinand Probst, diese Handreichung unter Wahrung des Urheberrechts erstellt zu haben.

Ich stelle diese Handreichung zur Nachnutzung nach Lizenz CC BY-NC-SA (Namensnennung, Bearbeitung, nicht kommerziell, Weitergabe unter gleichen Bedingungen) zur Verfügung.

Unterschrift: F. Probst