Handreichung: Physical Computing – Projekt Binäruhr

1. Kurzvorstellung

In diesem Physical Computing Projekt entwickeln die Schülerinnen und Schüler in Gruppenarbeit eine Binäruhr, welche auf Wunsch auch im Klassenraum platziert werden kann. Diese Uhr zeigt die Zeit nicht in herkömmlicher Form an, sondern verwendet das binäre Zahlensystem. Jede Ziffer der Uhrzeit wird durch eine Kombination von LEDs dargestellt, die entsprechend der binären Werte aufleuchten. Durch das Projekt kann das Verständnis des Binärsystems vertieft werden und die SuS werden zur Teamarbeit angeregt. Der Problemlöseprozess wird durch Arbeitsblätter unterteilt und begleitet, wodurch Interdependenzen zwischen den SuS geschaffen werden. Das Projekt kombiniert Informatik, Mathematik und Elektrotechnik und kann durch komplexe Zusatzaufgaben (z.B. Blinkanzeige am Stundenbeginn oder Bau eines Gehäuses) beliebig erweitert und perfektioniert werden.

2. Einordnung in die Lehrpläne

Die LBs beziehen sich stets auf die neuste Überarbeitung des Lehrplans. Das Projekt kann an verschiedenen Schulformen durchgeführt werden, jedoch empfehle ich die Verwendung in einem gymnasialen Grund- oder Leistungskurs.

Klasse	OS	Gy	BS
7			
8			
9	LB2		
10	LB2		
11/12		GK LB7 oder LK LB11	GK LB 4b

Für die reine Bearbeitung der Arbeitsblätter, den Bau und die Programmierung der Uhr ist mit ca. vier bis fünf Unterrichtsstunden zu rechnen. Davon wird eine Stunde für die Einführung ins Projekt benötigt, 1-2 für die EA und die Teilaufgaben und zwei Stunden für die Synthese der Teilaufgaben. Durch die Zusatzaufgaben und ggf. eine Reflexion oder Dokumentation kann der Umfang des Projektes erweitert werden.

3. Lernziele

Kognitive Ziele:

- Die SuS übertragen eine Uhrzeit in ihre Binärdarstellung.
- Die SuS entwickeln und konstruieren einen Schaltkreis für eine Binäruhr mit dem Arduino Uno.
- Die SuS entwickeln Teilfunktionen für den Programmcode (z.B. clock() oder displayHours()) in EA.
- Die SuS fügen die Teilergebnisse zu einer funktionierenden Arduino-Binäruhr in GA zusammen.

Affektive Ziele:

- Die SuS beachten den Unterschied zwischen Binär- und Dezimalzahlen.
- Die SuS beteiligen sich an der Gruppenarbeit, indem sie einen Schaltplan / Codebaustein beitragen und evaluieren die Beiträge der anderen Gruppenmitglieder.

Psychomotrische Ziele:

- Die SuS gestalten ihre Binäruhr optisch ansprechend.
- Die SuS verbessern die Bedienbarkeit der Binäruhr.

4. Voraussetzungen

Fachliche Voraussetzungen / Vorkenntnisse

Um das Projekt realisieren zu können benötigen die SuS grundlegende elektrotechnische Grundkenntnisse zu elektrischen Stromkreisen. Sie sollten in der Lage sein mit einem Breadboard, Jumperkabeln, Widerständen und LEDs zu arbeiten und mit dem Ohm'schen Gesetz vertraut sein. Die Grundlagen dafür werden im Physikunterricht in folgenden Lernbereichen gelegt:

Physik – Oberschule:

- Kl 6 – LB 4: Elektrische Stromkreise
- Kl 8 – LB 1: Leitungsvorgänge in Metallen

Physik - Gymnasium:

- Kl 6 – LB 4 Elektrische Stromkreise
- Kl 8 - LB 3: Eigenschaften elektrischer Bauelemente

Die Binäruhr ist in erster Linie ein Programmierprojekt. Die SuS müssen mit Sequenzen, Wiederholstrukturen, Verzweigungen, Datentypen, logischen Operatoren, Unterprogrammen und Variablen vertraut sein, um das Projekt realisieren zu können.

Informatik – Oberschule:

- Kl 7 – LB 1 Informatik im Alltag
- Kl 8 – LB 1 Algorithmen und Programme

Informatik – Gymnasium:

- Kl 7 – LB 1 Informationen und Daten
- Kl 7 – LB 3 Algorithmen
- Kl 8 – LB 1 Algorithmen
- Kl 10 – LB 1 Algorithmen

Die SuS sollten außerdem mit dem modulo-Operator vertraut sein und bereits ein simples Arduino-Programm realisiert haben, damit der Umgang mit dem Minicomputer und seinen Pins sie nicht vor nicht bewältigbare Herausforderungen stellt.

Technische und materielle Voraussetzungen

Pro Gruppe (3 Personen):

- Computer mit Internetverbindung (um Schaltkreis-Simulator von tinkercad.com nutzen zu können)
- Arduino Uno mit PC-Verbindung
- Breadboard
- mind. 11 LEDs (sechs für Sekundenanzeige, fünf für Minuten) + Widerstände und Kabel

  Für Zusatzaufgaben ggf.:
  - 7-Segment Uhranzeige
  - Real Time Clock Module
  - Powerbank

5. Kurzdarstellung

Arbeitsblätter

Die Arbeitsblätter sollten in Gruppen mit je drei Mitgliedern bearbeitet werden. AB 1 ist in EA zu bearbeiten. Die Teilaufgaben werden auf die einzelnen Mitglieder verteilt und in der Synthese zu einer funktionierenden Binäruhr zusammengeführt.

Name: Projekt Binäruhr Datum: 03.07.2024

Physical Computing-Projekt: Binäruhr

In Klasse 7 habt ihr gelernt, wie man Daten als Binärzahlen darstellt. Konzipiert und programmiert nun mit diesem Wissen eine Uhr, welche die Uhrzeit im Binärsystem anzeigt und mit dem Arduino und LEDs arbeitet.

Bsp.:

a) Erläutere, warum man für die Darstellung der Minuten sechs LEDs braucht, während man für die Stunden nur fünf benötigt.

b) Skizziere die Darstellung der folgenden Uhrzeiten, indem du die leuchtenden LEDs markierst.

c) Um mit dem Arduino eine Uhr zu realisieren, nutzen wir die Funktion millis(). Diese gibt die Anzahl von Millisekunden zurück, seit das Arduino-Board das aktuelle Programm gestartet hat. Gib den Code für eine Umrechnung in Sekunden an. Hinweis: Du benötigst dafür die modulo-Operation, welche mit % codiert wird.

int seconds =

d) Über die int-Variablen hours, seconds und minutes kann die Startzeit des Programmes eingegeben werden. Ändere die Codezeile für seconds so, dass auch der Startwert von seconds berücksichtigt wird.

int seconds =

Zusatz: Notiere zwei zusätzliche Features, die die Uhr für den Klassenraum noch nützlicher machen würden.

Begib dich nach dem Vergleich dieses Arbeitsblattes in die von der Lehrkraft zugteilten Gruppen.

1

Teilaufgabe 1: Der Schaltplan

Nutze den Schaltkreis-Designer von tinkercad.com, um eine funktionierende Schaltung für eine Binäruhr mit elf, auf einem Breadboard platzierten LEDs und dem Arduino Uno zu entwerfen.

a) Platziere den Arduino Uno und die kleine Steckplatine im Arbeitsbereich.
b) Bringe 11 LEDs (5 für die Stunden, 6 für die Minuten) auf dem Breadboard an und sorge dafür, dass Stunden und Minuten voneinander unterscheidbar sind.
c) Um den Stromfluss zu begrenzen, benötigen wir Widerstände. Die LEDs haben einen Betriebsstrom von 20mA, eine Durchlassspannung von 2V und wir schließen sie an die digitalen Arduino-Pins 2-12 an. Diese haben eine Ausgangsspannung von 5V. Berechne mit dem Ohmschen Gesetz den benötigten Widerstand.

Widerstand R =

d) Ergänze die Widerstände und benötigten Kabel auf dem Schaltplan.
e) Schreibe die Funktion setup, in welcher der Betriebsmodus der benötigten Pins festgelegt wird. Konsultiere zur Unterstützung die Arduino-Dokumentation: https://www.arduino.cc/reference/de/language/functions/digital-io/pinmode/

```
1   void setup() {
5
10
15  }
```

Teilaufgabe 2: Die Uhr

Wir nutzen die loop-Schleife und die millis()-Funktion für die Umsetzung der Uhr, welche hours, minutes und seconds in gleichnamigen, global deklarierten int-Variablen speichert. Um festzustellen, ob sich der Sekundenwert verändert hat, wird die int-Variable secondsOld genutzt.

a) Schreibe die loop-Funktion unter Nutzung der Zeile aus Aufgabe 1d:

```
1   void loop() {
                                    // Zeile aus 1d
                                    // falls sich der Wert von seconds verändert hat, dann…
                                    // rufe die Funktion clock auf
5                                   // rufe die Funktion displayMinutes auf
                                    // rufe die Funktion displayHours auf
    }
    }
```

b) Schreibe die Funktion clock. Sie soll die Variable minutes um 1 erhöhen, wenn 59 Sekunden erreicht sind. Wenn 60 Minuten erreicht sind, sollen die Minuten zurück auf 0 gesetzt werden und hours wird um 1 erhöht. Bei 24 Stunden wird hours zurück auf 0 gesetzt.

```
1   void clock() {
5
10  }
```

Teilaufgabe 3: Die Anzeige

Um die Zeit darzustellen, nutzen wir die von Gruppenmitglied 1 erstellte Schaltung und die von Gruppenmitglied 2 erstellte Uhr-Funktion. Den Funktionen displayHours und displayMinutes wird jeweils der darzustellende Wert als int übergeben. Dieser soll dann in ein char-Array zur Binärdarstellung umgewandelt werden („0" = aus, „1" = an) und anschließend sollen die entsprechenden LEDs zum Leuchten gebracht werden.

a) Schreibe eine for-Schleife, welche den Eingabeparameter int displayMinutes in das char-Array minutesArray speichert.

```
1   for(
5
10      }
```

b) Schreibe eine for-Schleife, welche das Array ausliest und die zu Einsen gehörenden LEDs zum Leuchten bringt. Nutze dafür digitalWrite(led, HIGH) für leuchtende LEDs und digitalWrite(led, LOW) für LEDs, die nicht leuchten.

```
1   for(
5
10      }
```

c) Übertrage dieses Prinzip auf die Funktion displayHours und notiere sie auf der Rückseite. Achte auf die veränderte Anzahl und Nummerierung der LEDs.

Synthese: Binäruhr mit dem Arduino

a) Wertet gemeinsam die Ergebnisse der Teilaufgaben aus. Erstellt eine Liste mit allen für das Projekt benötigten Bauteilen.
b) Wertet gemeinsam die Ergebnisse der Programmieraufgaben aus. Beginnt euer Arduino-Programm im Tinkercad-Schaltkreis-Simulator, indem ihr die globalen Variablen (hours, minutes, seconds etc.) mit ihrem Datentyp deklariert.
c) Überprüft gemeinsam die isolierte Funktionalität der von euch geschriebenen Funktionen, indem ihr z.B. displayMinutes mit einem selbst gewählten int-Parameter und einem print-Befehl testet.
d) Nutzt dann die setup-Funktion von Mitglied 1 und die loop-Funktion von Mitglied 2 und stellt eine Verbindung zu clock und displayMinutes / displayHours her.
e) Testet euer Programm ausführlich mit dem Simulator in Tinkercad und korrigiert ggf. Bugs.
f) Lasst euch dann von eurer Lehrkraft einen Arduino Uno und alle benötigten Bauteile aushändigen. Baut den Schaltkreis auf.
g) Übernehmt das Programm aus der Tinkercad-Simulation in die Arduino IDE und testet die Funktionalität der Uhr ausführlich.

Zusatzaufgaben Binäruhr

a) Erweitert das Programm so, dass der Stundenbeginn im Programm in Arrays oder als Tupel (Stunde | Minute) eingegeben werden kann. Am Anfang jeder Stunde sollen alle LEDs gleichzeitig für 5 Sekunden blinken.
b) Ergänzt die Möglichkeit der Benutzereingabe über den Serien-Monitor. Die BenutzerInnen sollen die aktuelle Uhrzeit und die Stundenbeginne beim Programmstart eingeben. Die Eingaben sollen dabei auf ihre syntaktische und semantische Korrektheit überprüft werden, sodass „21:43“ akzeptiert wird, „Hallo“ oder „26:67“ jedoch nicht.
c) Ergänzt die Binäruhr um eine Sekundenanzeige. Diskutiert wie man das Problem des Mangels an Pins lösen kann.
d) Lasst euch von eurer Lehrkraft eine 7-Segment-Uhrenanzeige aushändigen und ändert den Schaltplan und das Programm so, dass die Uhr autark vom Computer mit einer Powerbank betrieben werden kann. Die Uhrzeit soll mit Knöpfen und der digitalen Anzeige beim Programmstart eingegeben werden.

Quellecode (mit Zusatzaufgabe zum Blinken)

```
int hours=11;
int minutes=19;
int seconds=0;
int secondsOld=0;
int secondsLoop=0;
char minutesArray[7]; // Make an Array to hold the 6 Bits.
char hoursArray[6];
int arrayPosition=0; // Keep track of the position of each Bit in MinutesArray
int arrayPositionHours=0;
int pauseHoursArray[]={11,11,10,25};//beide Arrays müssen gleich groß sein!
int pauseMinutesArray[]={16,20,30,61};

void setup() {
  Serial.begin(9600);
  for (int i = 2; i <= 7; i++){
      pinMode(i, OUTPUT);
  }
}

void loop(){
  seconds = ((millis() + 50000)/1000%60);//Achtung: startet bei 50s
  if (seconds != secondsOld){
      secondsOld = seconds;
      clock();
      displayMinutes(minutes);
      displayHours(hours);
  }
}
```

```
void clock(){
    if (seconds == 59){
      minutes++;
      if(minutes > 59){
        minutes=0;
        hours++;
        if(hours>23){
          hours=0;
        }
      }
      for (int i=0; i<sizeof(pauseHoursArray); i++){
            if ((pauseHoursArray[i]==hours)&&(pauseMinutesArray[i]==minutes)){
                  blink();
            }
      }
    }
}
```

```
void displayMinutes(int displayMinutes){
      for(int a=32; a>=1; a=a/2){      // 32, dann 16, dann 8 usw.
            if((displayMinutes-a)>=0){         // int groß genug für 1?
                  minutesArray[arrayPosition]='1';  // 1 an array-Position
                  displayMinutes-=a;  // Subtraktion vom Integer
            }
            else{
                  minutesArray[arrayPosition]='0';
            } // int war nicht groß genug → 0 an array-Position
            arrayPosition++; // eine Array-Position weiter gehen
      }
      arrayPosition=0;

      for (int i=5; i>=0; i=i-1){
            int led=7-i;
            if (minutesArray[i]=='1'){
                  digitalWrite(led, HIGH);
            } else {
                  digitalWrite(led, LOW);
            }
      }
}

void displayHours (int displayHours){
      for(int a=16; a>=1; a=a/2){      // 16, 8, 4 usw.
            if((displayHours-a)>=0){         // int groß genug für 1?
                  hoursArray[arrayPositionHours]='1';  // '1' an arrayPosition
                  displayHours-=a;  // Subtraktion vom int
            } else {
                  hoursArray[arrayPositionHours]='0';
            } // int zu klein → '0'
            arrayPositionHours++; // ein Schritt weiter im Array gehen
      }
      arrayPositionHours=0;
```

```
      for (int i=4; i>=0; i=i-1){
            int led=12-i; //Pin-Nummer der „höchsten“ LED = 12
            if (hoursArray[i]=='1'){
                  digitalWrite(led, HIGH);
            } else {
                  digitalWrite(led, LOW);
            }
      }
}

void blink(){
      for (int i=0; i<5; i++){
            for (int j=2; j<13; j++){
                  digitalWrite(j, HIGH);
            }
            delay(800);
            for (int k=2; k<13; k++){
                  digitalWrite(k, LOW);
            }
            delay(200);
      }
}
```

Schaltplan

Link zum Musterprojekt bei Tinkercad:
https://www.tinkercad.com/things/gdivU8xaQ1B-copy-of-binaryschoolclock/editel?sharecode=F4ebredX7avjYYHLLn6cO7kBsrm9ZhrFxka9wWzW4SI