```
/*
  Fertiges Tischkicker-Zähler-Programm von Leo Pieren und Reto Morgenthaler 2016-17. Version vom 28. April 17 beim Einbau in den Tisch.
  ---------------------------------------------------------------------------------------------------------------------------
  Programm-Kommentare kann man irgendwo schreiben; am Anfang und am Ende gekennzeichnet mit dem Schrägstrich und dem Stern. Dem System
  wird so mitgeteilt, dass es sich nicht im Programm-Code handelt.
*/
                                                  // In einem solchen Programm kann man sein Programm in jeder Zeile nach einem Doppelstrich kommentieren.
                                                  // Wir machen das, weil wir dieses Programm ausdrucken und aufhängen wollen.


#include <Wire.h>                                 // Diese 3 Zeilen lesen Programm-Code ein vom Hersteller "Adafruit" des Displays;
#include <Adafruit_GFX.h>                         // man nennt sie englisch Libraries, also Bibliotheken.
#include "Adafruit_LEDBackpack.h"

Adafruit_7segment matrix = Adafruit_7segment();   // Hier wird das Anzeige-Element "matrix" aus einer eingelesenen Bibliothek angesprochen.

int Rot;                                          // Hier werden die Variablen fuer die Torststaende definiert, die wir weiter unten verwenden.
int Blau;

#define LEDPIN 13                                 // Hier wird die im "Mini-Computer" unter dem Tisch eingebaute LED an Pin 13 definiert.
                                                  // (Jetzt, wo alles funktioniert und in ein Gehäuse eingebaut ist, ist das nutzlos
                                                  // und koennte weggelassen werden.)

#define SENSORPIN 4                               // Und hier werden die beiden Sensorpins definiert, an denen die Schranken eingesteckt sind..
#define SENSORPIN2 2                              // So "weiss" das Programm, an welchem Anschluss eine Strom-Schwankung zu erwarten ist,
                                                  // wenn die Lichtschranke unterbrochen wird.
#define SENSORPIN3 7                              // An diesen Pin wird der rote Nullsteller-Knopf angeschlossen.


int sensorState = 0, lastState=0;                 // Hier wird der Zustand der einen Lichtschranke als Variable "sensorState" definiert und nullgestellt.
int sensorState2 = 0, lastState2=0;               // Dasselbe fuer die andere Lichtschranke
int sensorState3 = 0, lastState3=0;               // Dasselbe für den Nullsteller-Knopf

void setup() {                                    // Ein Arduino-Programm ist in der Programmier-Sprache C++ geschrieben und enthält immer einen Programmteil,
                                                  // der mit "void setup" beginnt. Dieser Teil wird 1x abgearbeitet. Er enthält die noetigen Startangaben.

  pinMode(LEDPIN, OUTPUT);                        // Der eingebaute Ledpin wird als Output-Pin definiert. So kann der Mini-Computer dort Spannung anlegen,
                                                  // damit die LED leuchtet.
                                                  // (Die LED ist im Gehäuse drin. Sie dient wie oben erwähnt nur noch zur Kontrolle bei Fehlersuche.)

  pinMode(SENSORPIN, INPUT);                      // Der eine Lichtschranken-Anschluss wird als Input definiert. Hier warten wir sozusagen auf ein Signal.
  digitalWrite(SENSORPIN, HIGH);                  // Mit "HIGH" wird eingebauter elektrischer Widerstand fuer diesen Pin aktiviert,
                                                  // denn nur mit diesem Widerstand wird das Signal korrekt erkannt,
                                                  // falls die Lichtschranke vom Ball unterbrochen wird.

  pinMode(SENSORPIN2, INPUT);                     // Dasselbe für den anderen Lichtschranken-Anschluss
  digitalWrite(SENSORPIN2, HIGH);

  pinMode(SENSORPIN3, INPUT);                     // Dasselbe für den Anschluss des Nullsteller-Knopfs
```

```
  digitalWrite(SENSORPIN3, HIGH);

  Rot = 0;                                  //Hier erhalten die Variablen fuer die Torstaende einen Startwert.
  Blau = 0;

  matrix.begin(0x70);                       //Hier wird Anschluss-Adresse vom Display im "Mini-Computer" definiert.
}



void loop(){                                // Ein Arduino-Programm ist in der Programmier-Sprache C++ und enthält immer einen Programmteil,
                                            // der "void loop" enthaelt. Dieser Teil ab hier wird in schnellem Tempo endlos durchlaufen.

  sensorState = digitalRead(SENSORPIN);     // Auf diesen zwei Zeilen werden die Zustaende der beiden Lichtrschranken ausgelesen und gespeichert.
  sensorState2 = digitalRead(SENSORPIN2);

  sensorState3 = digitalRead(SENSORPIN3);   // Und hier dasselbe für den Zustand des Nullstellers


                                           // Die folgenden Zeilen bewriken, dass die die Torstaende Rot und Blau am Display angezeigt werden.
  matrix.writeDigitNum(0, (Rot / 10) );     // Definiert, dass an Display-Stelle 0 die erste Stelle von Variable Rot kommt.
  matrix.writeDigitNum(1, Rot % 10 );       // Definiert, dass an Display-Stelle 1 die zweite Stelle von Variable Rot kommt.
  matrix.drawColon(true);                  // Definiert, dass der Doppelpunkt in der Mitte des Displays angezeigt wird.
  matrix.writeDigitNum(3, (Blau / 10) );    // Definiert, dass an Display-Stelle 3 die erste Stelle von Variable Blau kommt.
  matrix.writeDigitNum(4, Blau % 10 );     // Definiert, dass an Display-Stelle 4 die zweite Stelle von Variable Blau kommt.
  matrix.writeDisplay();                   // Zeigt alle diese Werte auf dem Display an.


                                            // Ab hier folgen "Falls-Sonst-Abfragen" (englisch "if" und "else") an den Lichtschranken und am Nullsteller.
                                            // Je nach Zustand werden die Torstaende erhoeht oder auf Null gestellt.


if (sensorState == LOW) {                   // Hier wird die Spannung an der einen Lichtschranke abgefragt.
                                            // Ist der Lichtstrahl unterbrochen, ist der Status LOW
  digitalWrite(LEDPIN, HIGH);               // Und die Kontroll-LED wird eingeschaltet (im Gehaeuse, von aussen nicht sichtbar)
}
else {

  digitalWrite(LEDPIN, LOW);              // ansonsten wird die LED nicht mit Strom versorgt und leuchtet nicht
}

if (sensorState && !lastState) {                // Abfrage der einen Lichtschranke. Das ! kehrt den Wert um,
                                                // ohne Ausrufezeichen wuerde staendig nach oben gezaehlt, wenn kein Ball ins Tor fliegt.
  Serial.println("Unbroken");                   // Im einen Fall ist sie nicht unterbrochen. Bei einem angeschlossenen Computerdisplay
                                                // (auch zur Kontrolle) wuerde stehen "Unbroken".
                                                // Sonst soll nichts geschehen. Es wird also kein Torstand erhoeht.

}
if (!sensorState && lastState) {
  Serial.println("Broken");                     // Im anderen Fall die Lichtschranfke sie unterbrochen
  Serial.println(Rot);                          // An einem angeschlossenen Display wuerde stehen "Broken"
```

```
    Rot = Rot + 1;                          // Und die eine Torvariable wird um 1 erhoeht
  }

  if (sensorState2 && !lastState2) {        // Auf den folgenden Zeilen geschieht dieselbe Abfrage an der anderen Lichtschranke
    Serial.println("2 Unbroken");
  }
  if (!sensorState2 && lastState2) {
    Serial.println("2 Broken");
    Blau = Blau + 1;                        // Und bei unterbrochenem Lichtstrahl wird die andere Torvariable um 1 erhoeht
  }

  if (sensorState3 && !lastState3) {        // Und schliesslich wird der Zustand des Nullstellers abgefragt
  }
  if (!sensorState3 && lastState3) {
    Serial.println("reset");                // Wenn er gedrueckt ist, wuerde auf einem angeschlossenen Display "reset" stehen
    Blau = 0;                               // Und die beiden Torvariablen werden auf Null gestellt
    Rot = 0;
  }
  lastState = sensorState;                  // Auf diesen 3 Zeilen werden die zuletzt erkannten Zustaende
  lastState2 = sensorState2;                // an den Lichtschranken und dem Nullsteller gespeichert
  lastState3 = sensorState3;
}                                           // Hier endet der void loop. Der Arduino bearbeitet ihn nun erneut ab; springt
                                            // also hoch zur Stelle "void loop" und arbeitet sich dann erneut bis hierher.
```