**Video about what this is:** [**https://youtu.be/uA70TDW96uE**](https://youtu.be/uA70TDW96uE)

Basically, we needed to stress test some audio cables for a prototyping project and we had to get going with this before our industrial grade machine arrived. Our engineering staff knocked this tester together in the course of a day. The Servo motor came from either an RC plane or car.

**“Trumeter 7000” Digital Counter - 28€**

****

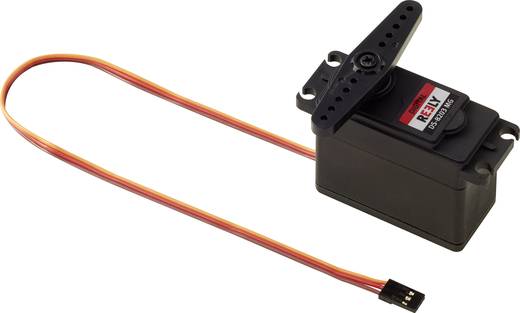
https://www.conrad.at/de/trumeter-7000-elektronischer-miniaturimpulszaehler-einbaumasse-294-x-22-mm-127419.html

**Arduino Uno 22-25€**



<https://at.rs-online.com/web/p/products/7154081/?grossPrice=Y&cm_mmc=AT-PLA-DS3A-_-google-_-PLA_AT_DE_Halbleiter-_-Entwicklungskits%7CEntwicklungskits_Prozessor_And_Mikrocontroller-_-PRODUCT+GROUP&matchtype=&gclid=EAIaIQobChMI38nLreGb2wIVC54bCh0npg4iEAQYASABEgLsw_D_BwE&gclsrc=aw.ds>

**DS-8203 MG Digital-Servo**



https://www.conrad.at/de/reely-spezial-servo-ds-8203-mg-digital-servo-getriebe-material-metall-stecksystem-jr-1365553.html

Austrian Audio Engineering Staff – (Priceless They wanted me to write that!)

**Oh, yeah: Source Code**

/\* Sweep

  by RIRO & DAWO & ALKA

  V6

  ADC Schwellwert = Kabelbruch

  Zeigt ADC Wert bei 0°, 180°

  Meldung bei Kabelbruch

  Läuft weiter bei Kabel = OK

  Pause bei 90° (kein Kabelknick)

  Zeigt Pausenzeit

\*/

#include <Servo.h>

#include <SPI.h>

#include <SD.h>

Servo myservo;  // create servo object to control a servo

enum e\_State {

  IDLE = 0,  PAUSE,  RUN, BROKEN

} myState;

const int chipSelect = 4;

int ledPin = 13;             // LED connected to digital pin 13

int countPin = 4;            // count output is pin 4

int buttonPin = 2;           // button connected to pin 2

int buttonval = 0;           // variable to store the read value

int pos = 0;                 // variable to store the servo position

int sensorPin = A0;

int sensorValue = 1023;

int threshold = 10;          // ADC Values 0-1023 (1023 = 5VDC) => with 1k pull up => threshold 10 = 10R

int timecount = 0;           // Pause time

void cableBroken() {

  Serial.println("");

  Serial.print(" - !!!Kabel gebrochen!!! -  A0 ADC Wert: ");

  Serial.println(sensorValue);

  Serial.println("");

}

void countUp() {

  static unsigned long counter = 1;  // 32bit only positive

  digitalWrite(countPin, HIGH);

  delay (5);                         // 5ms High signal

  digitalWrite(countPin, LOW);

  Serial.print("Counter: ");

  Serial.println(counter++);

}

void ADCValue() {

  Serial.print("  A0 ADC Wert: ");

  Serial.println(sensorValue);        // ADC value

}

void servopos(int endpos) {

  bool changed = false;

  if (pos < endpos) {

    for (; pos < endpos; pos += 3) {    // counts up in 3° steps (speed = 180°/3°\*20ms = 1.2sec)

      myservo.write(pos);               // tell servo to go to position in variable 'pos'

      delay(20);                        // 20ms delay for every 3° step (20ms according to Servo Spec)

    }

    changed = true;

  }

  if (pos > endpos) {

    for (; pos > endpos; pos -= 3) {

      myservo.write(pos);

      delay(20);

    }

    changed = true;

  }

  if (changed == true) {

    Serial.print(pos);

    Serial.print("°");

    if (pos == 90) {

      Serial.print(" - Pause - ");

    }

  }

}

void setup() {

 pinMode(ledPin, OUTPUT);     // sets the digital ledPin as output

  pinMode(countPin, OUTPUT);   // sets the countPin as output

  pinMode(buttonPin, INPUT);   // button pin set as input

  digitalWrite(countPin, LOW);

  myservo.attach(9, 650, 2400); // attaches the servo signal on pin 9, min, max = shortest, longest pulse with (µs) = physical end position for 0°, 180°

  Serial.begin(9600) ;

  Serial.println("Hello World");

  myState = e\_State::RUN;

}

void loop() {

  switch (myState) {

    case e\_State::IDLE:

      sensorValue = analogRead(sensorPin);// A0 einlesen

      if (sensorValue <= threshold) {     // Wenn Kabel ok => Servobewegung

        myState = e\_State::RUN;

      }

      break;

    case e\_State::BROKEN:

      cableBroken();

      myState = e\_State::IDLE;

      break;

    case e\_State::RUN:

      buttonval = digitalRead(buttonPin);

      if (buttonval == HIGH) {

        myState = e\_State::PAUSE;

        break;

      }

      servopos(180);

      sensorValue = analogRead(sensorPin);// A0 einlesen

      ADCValue();                         // Wert ausgeben

      if (sensorValue > threshold) {      // Wenn Kabel ok => Servobewegung

        myState = e\_State::BROKEN;

        break;

      }

      servopos(0);

      sensorValue = analogRead(sensorPin);

      ADCValue();

      if (sensorValue > threshold) {

        myState = e\_State::BROKEN;

        break;

      }

      countUp();

      break;

    case e\_State::PAUSE:

      servopos(90);

      delay(1000);

      timecount++;

      buttonval = digitalRead(buttonPin);

      if (buttonval == LOW) {

        Serial.print("Dauer: ");

        Serial.print(timecount);

        Serial.println("s");

        Serial.println("");

        timecount = 0;

        myState = e\_State::RUN;

      }

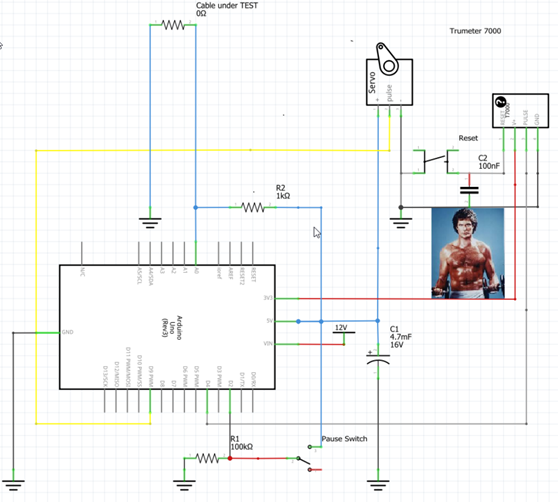
      break;

  }

}

**Schematics: I dunno. The Engineers sent me two sets… :D**

**Hoff Equipped**



**The following one actually works:**

