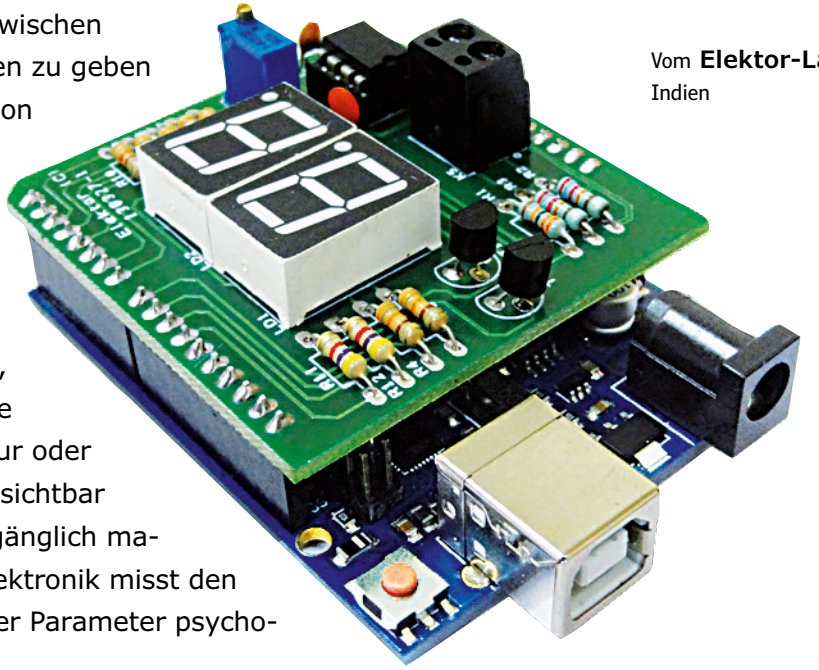
# Stresstester

## Dr. Arduino misst Hautleitfähigkeit

Vom **Elektor-Labor**
Indien

Da es viele Zusammenhänge zwischen Stress und diversen Krankheiten zu geben scheint, ist die Beeinflussung von nicht willkürlich steuerbaren Körperfunktionen per Autogenem Training ein naheliegendes Antistress-Verfahren. Hierzu gibt es viele verschiedene Biofeedback-Schaltungen, die physiologische Maße wie die Herzfrequenz, Körpertemperatur oder gar elektrische Gehirnaktivität sichtbar und so einer Beeinflussung zugänglich machen. Die hier beschriebene Elektronik misst den Hautleitwert – ein sehr sensibler Parameter psychophysiologischer Erregung.



Die nötige Hardware ist eigentlich ganz einfach: Ein Interface in Form des Timer-ICs auf einem Arduino-Shield generiert ein von einem Mikrocontroller auswertbares Signal. Dieses wird dann vom Controller auf einem Arduino-Board des Typs Uno R3 verarbeitet. Das Ergebnis der Berechnungen wird schließlich auf zwei Sieben-Segment-Displays angezeigt.

### Funktionsweise

Bei IC1 in der Schaltung von **Bild 1** handelt es sich um einen 555-Timer, der als astabiler Multivibrator geschaltet ist. Wenn man die beiden an K5 angeschlossenen Elektroden berührt, steigt die Ausgangsfrequenz des 555. Änderungen der Hautleitfähigkeit (in der Medizin spricht man nicht vom Widerstand, wie in der Technik üblich) schlagen sich daher in Änderungen der Frequenz des Oszillators um IC1 nieder.
Das Ausgangssignal von IC1 liegt an Pin 5 des Arduino-Boards. Die Firmware des Mikrocontrollers sorgt dafür, dass aus dieser Frequenz ein Maß für den Stress in Prozentform (mit dem Zahlenbereich 0...99) berechnet wird. Dieser Stresspegel wird dann als zweistellige Zahl angezeigt. Hierzu werden die beiden Dezimalstellen des Displays LD1 und LD2 mit Hilfe der beiden Transistoren durch entsprechend abwechselnde Verbindung der gemeinsamen Kathoden mit Masse gemultiplext.

### Software

Die Firmware für das Projekt wurde mit der Arduino-IDE Version 1.0.5 in C geschrieben. Dabei kam für Berechnungen und die Skalierung des Eingangssignals die Library „freqCounter“ zum Einsatz. Die Library kann man von [1] downloaden und den Arduino-Sketch von [2]. Außerdem ist der komplette Sketch-Code am Ende dieses Artikels abgedruckt.
Die wichtigsten Funktionsblöcke werden nachfolgend erläutert:

### Setup

Diese Funktion definiert die Konfiguration der einzelnen Pins als Ein-bzw. Ausgang.

**Gefahr: Nie am Körper messen, wenn der Stresstester per USB mit einem PC verbunden ist!**

- Die Pins 2, 3, 4, 6, 7, 8 und 9 sind als Ausgänge mit den Segment-Anschlüssen b, c, d, e, f und g des Displays verbunden.
- Die Pins 10 und 11 sind ebenfalls Ausgänge und dienen als Display-Select-Pins für LD1 und LD2.
- Pin 5 ist ein Eingang und mit dem 555-Ausgang Pin 3 verbunden.

**Loop**

Hier werden die Funktionen der Library „freqcounter“ zum Zählen der Impulse benutzt. Die sich ergebende Impulsanzahl wird wie folgt verarbeitet: Wenn die Elektroden nicht berührt werden, ergibt sich ein Impulswert von 500 (einstellbar mit Trimmpoti P1). Dieser und geringere Werte werden als 0 % Stress interpretiert, was dem Nullpunkt des Hautleitwerts entspricht.

Berührt man die Elektroden, steigt der Impulswert. Der Maximalwert bei kurzgeschlossenen Elektroden liegt bei 11.000, weshalb alle Werte ≥11.000 als 99 % Stress angezeigt werden.
Der Stress-Prozentsatz wird also relativ zum Maximalwert mit kurzgeschlossenen Elektroden berechnet.

**Display_seg(unsigned long stress_per)**

Dieser Funktion wird in der Hauptschleife der Stress als Prozentwert übergeben. Die Funktion ermittelt zunächst die Werte für die beiden Stellen und gibt sie dann auf dem Sieben-Segment-Display aus.

**pickNumber(int x)**

Der dieser Funktion übergebene Wert (<10) wird auf dem zuvor ausgewählten Display als jeweilige Stelle des zweistelligen Resultats angezeigt.

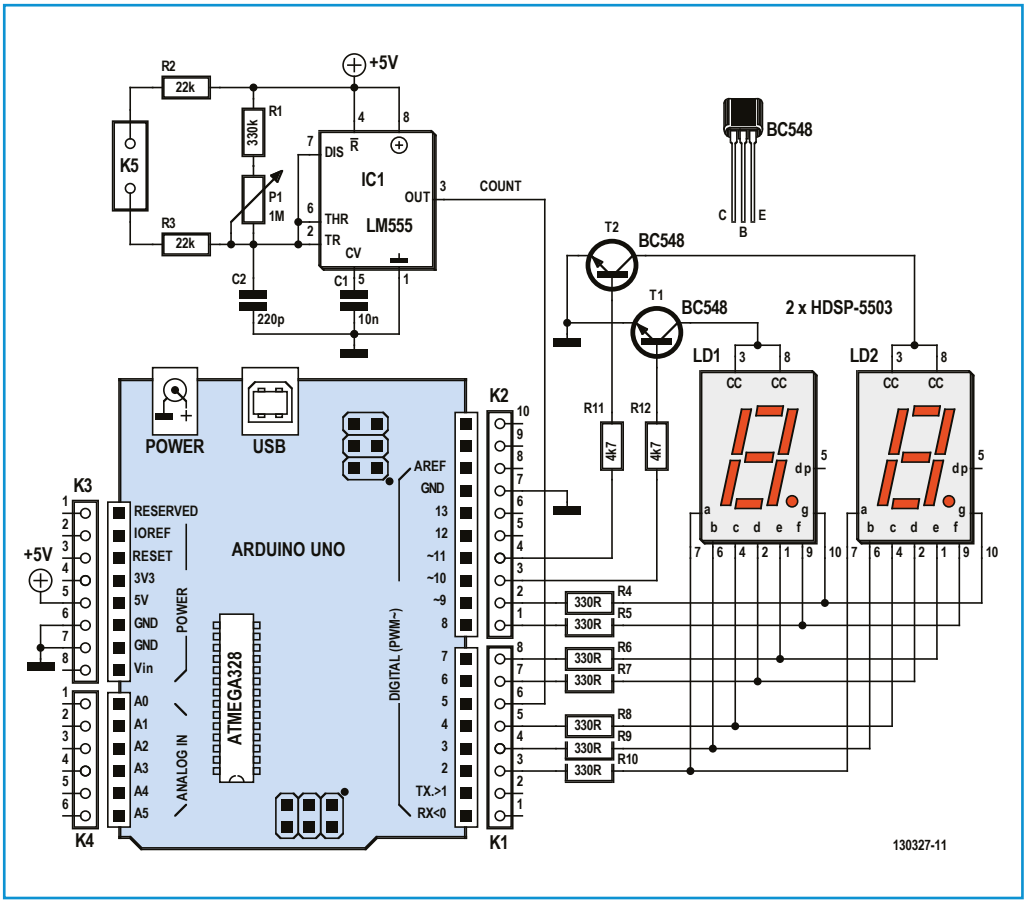



Bild 1.
Die Schaltung verrät es: Dies ist ein Arduino-Shield.

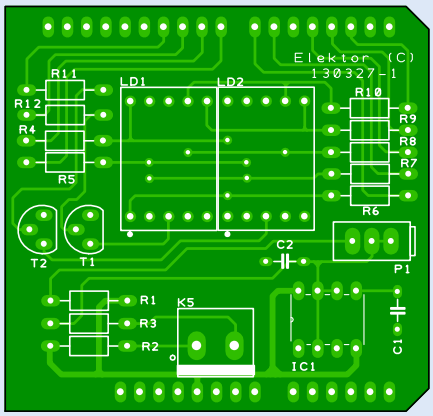
## Stückliste

**Widerstände:**
(alle 5 % und 0,25 W)
R1 = 330 k
R2,R3 = 22 k
R4..R10 = 330 Ω
R11,R12 = 4,7 k
P1 = 1 M, Mehrgang-Trimmpoti, stehend

**Kondensatoren:**
C1 = 10 n
C2 = 220 p

**Halbleiter:**
IC1 = LM555CN/NOPB
T1,T2 = BC548
LD1,LD2 = SBC56-21EGWA (Kingbright),
7-Segment-LED-Display, gemeinsame Kathode

**Außerdem:**
K1,K3 = 8x1-Stiftleiste, RM 1/10“
K2 = 10x1-Stiftleiste, RM 1/10“




K4 = 6x1-Stiftleiste, RM 1/10“
K5 = 2-pol. Schraubklemme, RM 5 mm
Arduino Uno R3
Platine 130237-1

Bild 2.
Platine zum Arduino-Shield.

### Aufbau

Die Platine des Stresstesters ist in **Bild 2** zu sehen. Da lediglich konventionelle bedrahtete Bauteile vorgesehen sind, dürfte die Bestückung keine Probleme bereiten. Nach Bestückung und Inspektion der Platine kann man diese auf das Arduino-Board stecken.

Wenn noch nicht geschehen, sollte spätestens jetzt die Firmware aufgespielt werden.

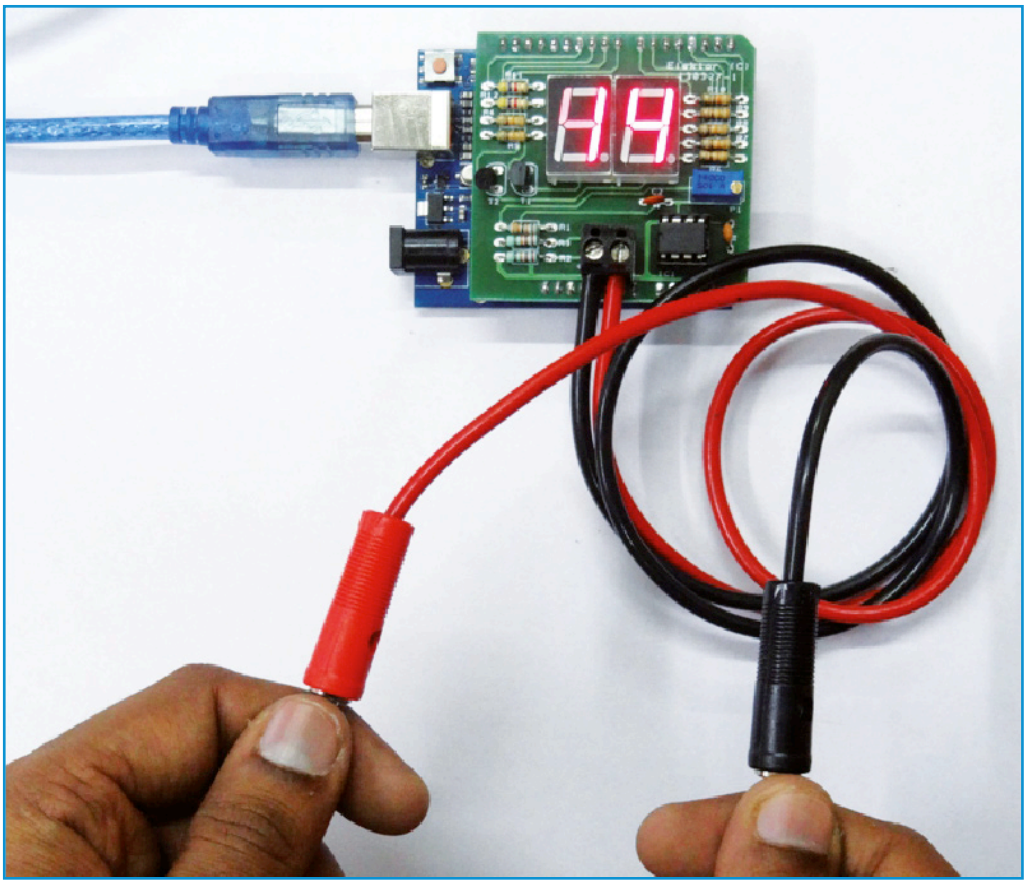


Bild 3.
Stresstester auf Arduino-Basis im Einsatz.

**Praktische Anwendung**

Nach dem Einschalten der Stromversorgung sollte man die Elektroden kurzschließen und die angezeigten Werte ablesen. Es sollte die Zahl „99“ zu lesen sein. Wenn nicht, stellt man P1 so ein, dass gerade „99“ angezeigt wird. Das wars auch schon – jetzt ist der Stresstester bereit für den praktischen Einsatz.

Um den eigenen Stresspegel zu messen, nimmt man in jede Hand eine Elektrode (nicht zu fest drücken), und schon zeigt sich der aktuelle Stresspegel auf dem Display (siehe **Bild 3**).

In der Praxis bedeuten Werte von unter 15 eher geringen Stress. Bei allen größeren Werten ist irgendetwas los...

***Hinweis:*** Zur Sicherheit den Stresstester **nur mit Batterien oder Akkus betreiben!** Spannungen zwischen 7,5 und 9 V genügen.

(130327)

## Alternative Kalibrierung via USB

Man verbindet den Tester zunächst per USB mit einem PC, öffnet ein Terminal-Programm (z.B. HyperTerminal) und stellt 9.600 Bd ein.

Nun inspiziert man bei offenen Elektroden mit dem Terminal die angezeigten Werte für „frq“ (Eingangsfrequenz). Die Werte sollten zwischen 450 und 500 liegen. Sind die Werte außerhalb dieses Bereichs, korrigiert man dies durch Verstellung von P1.

### Weblinks

[1] Arduino Frequenzzähler-Library:
http://interface.khm.de/index.php/lab/experiments/arduino-frequency-counter-library/

[2] Arduino-Sketch: www.elektor-magazine.com/post

## Sketch

**Datei „Stress_Tester.ino“**

```
//*****************************************************************************
//Project Name: Stress Tester
//Microcontroller:ATmega328p(Arduino Uno R3)
//This project is used to measure the percentage of stress of a humans body on the basis of skin
//resistance. In order to achieve this we have used a circuit
//to generate the input signal, this uses the LM555 IC to generate the pulse when the electrodes
//are touched the frequency of the signal increases and this frequency is fed as input to pin 5
//of MCU. Percentage of this pulse count with respect to 32000 is displayed onto two seven segment
//displayed connected to pin 2,3,4,6,7,8,9,10,11 of arduino uno board. Human body stress cannot
//exceed a frequency of 32kHz and hence the maximum value of count is considered as 32000.
//*****************************************************************************

#include <FreqCounter.h>


unsigned long frq ,f,init_freq;
int cnt;
int aPin = 2;
int bPin = 3;
```

```
int cPin = 4;
int dPin = 6;
int ePin = 7;
int fPin = 8;
int gPin = 9;
int SEG1 = 10;
int SEG2 = 11;
int num,count,i;
int dig1 = 0;
int dig2 = 0;
int dig3 = 0;
int dig4 = 0;
int DTime = 1;
int j;

void setup() {

  Serial.begin(9600);
  pinMode(aPin, OUTPUT);
  pinMode(bPin, OUTPUT);
  pinMode(cPin, OUTPUT);
  pinMode(dPin, OUTPUT);
  pinMode(ePin, OUTPUT);
  pinMode(fPin, OUTPUT);
  pinMode(gPin, OUTPUT);
  pinMode(SEG1, OUTPUT);
  pinMode(SEG2, OUTPUT);
  count = 1;
}
//********************************************************************************
void Display_seg(unsigned long init_freq)
{
   num = init_freq;
  dig3 = num / 10;
  dig4 = num - (dig3 *10);

  digitalWrite( SEG2, HIGH);   //digit 2
  pickNumber(dig4);
 delay(DTime);
  digitalWrite( SEG2, LOW);

  digitalWrite( SEG1,HIGH);   //digit 1
  pickNumber(dig3);
  delay(DTime);
  digitalWrite( SEG1, LOW);
}
//********************************************************************************
void loop() {

  // wait if any serial is going on
```

```
 Display_seg(init_freq);
  FreqCounter::f_comp=10;   // Cal Value / Calibrate with professional Freq Counter
  FreqCounter::start(100);  // 100 ms Gate Time

  while (FreqCounter::f_ready == 0)  // until pulse occurs on input pin5
  {
    Display_seg(init_freq); //display the value of frequency onto the seven segment display
  }

  frq=FreqCounter::f_freq;  //put the calculated frequency value in frq variable,
                            //this value is calculated and passed rom the "freqCounter" file
   Display_seg(init_freq);  //Display the value of frequency onto the seven segment display

   //calibration of input frequency value to calculate percentage of stress
   if (frq < 450)           //when electrodes remain untouched the frequency value is <=3000,
                            //hence this value is considered as zero value of stress
   {
     init_freq = 0;
     Display_seg(init_freq);//Display the value of frequency onto the seven segment display
   }
   else
   {
     f = frq - 450;         // if value greater than 450 then the value if first brought to its
                            //reference zero value by subtracting 450 from it
     if (f > 11000)         //check if value is less than 450 as the human stress cannot me more
                            //then 32kHz
     {
       if(count == 1)
       {
         init_freq = 0;
         count = 0;
       }
       else
         init_freq = 99;    // if value is above 30000 then stress percentage is 99
     }
     else
     {
       //percentage is calculated of value obtained from input signal
       init_freq = ((f * 100) / 11000);
     }
     Display_seg(init_freq);
   }
    Display_seg(init_freq);
    Serial.print("frq");
    Serial.println(frq);
    Serial.println(f);
    Serial.print("Stress");
    Serial.println(init_freq);

}
```

```
//********************** pick the digit to be displayed onto the seven segment *****
void pickNumber(int x){
   switch(x){
     case 1: one(); break;
     case 2: two(); break;
     case 3: three(); break;
     case 4: four(); break;
     case 5: five(); break;
     case 6: six(); break;
     case 7: seven(); break;
     case 8: eight(); break;
     case 9: nine(); break;
     default: zero(); break;
   }
}
//*************************************************************************************
//************************** clear all segments of the display **********************
void clearLEDs()
{
  digitalWrite(  2, LOW); // A
  digitalWrite(  3, LOW); // B
  digitalWrite(  4, LOW); // C
  digitalWrite(  6, LOW); // D
  digitalWrite(  7, LOW); // E
  digitalWrite(  8, LOW); // F
  digitalWrite(  9, LOW); // G
}
//*************************************************************************************
//************************** Display digit one(1) on seven segment ******************
void one()
{
  digitalWrite( aPin, LOW);
  digitalWrite( bPin, HIGH);
  digitalWrite( cPin, HIGH);
  digitalWrite( dPin, LOW);
  digitalWrite( ePin, LOW);
  digitalWrite( fPin, LOW);
  digitalWrite( gPin, LOW);
}
//*************************************************************************************
//************************** Display digit two(2) on seven segment ******************
void two()
{
  digitalWrite( aPin, HIGH);
  digitalWrite( bPin, HIGH);
  digitalWrite( cPin, LOW);
  digitalWrite( dPin, HIGH);
  digitalWrite( ePin, HIGH);
  digitalWrite( fPin, LOW);
  digitalWrite( gPin, HIGH);
}
```

```
//*****************************************************************************
//************************** Display digit three(3) on seven segment *****************
void three()
{
  digitalWrite( aPin, HIGH);
  digitalWrite( bPin, HIGH);
  digitalWrite( cPin, HIGH);
  digitalWrite( dPin, HIGH);
  digitalWrite( ePin, LOW);
  digitalWrite( fPin, LOW);
  digitalWrite( gPin, HIGH);
}
//*****************************************************************************
//************************ Display digit four(4) on seven segment ******************
void four()
{
  digitalWrite( aPin, LOW);
  digitalWrite( bPin, HIGH);
  digitalWrite( cPin, HIGH);
  digitalWrite( dPin, LOW);
  digitalWrite( ePin, LOW);
  digitalWrite( fPin, HIGH);
  digitalWrite( gPin, HIGH);
}
//*****************************************************************************
//************************ Display digit five(5) on seven segment ******************
void five()
{
  digitalWrite( aPin, HIGH);
  digitalWrite( bPin, LOW);
  digitalWrite( cPin, HIGH);
  digitalWrite( dPin, HIGH);
  digitalWrite( ePin, LOW);
  digitalWrite( fPin, HIGH);
  digitalWrite( gPin, HIGH);
}
//*****************************************************************************
//************************** Display digit six(6) on seven segment ******************
void six()
{
  digitalWrite( aPin, HIGH);
  digitalWrite( bPin, LOW);
  digitalWrite( cPin, HIGH);
  digitalWrite( dPin, HIGH);
  digitalWrite( ePin, HIGH);
  digitalWrite( fPin, HIGH);
  digitalWrite( gPin, HIGH);
}
//*****************************************************************************
//************************** Display digit seven(7) on seven segment ****************
void seven()
```

```
{
  digitalWrite( aPin, HIGH);
  digitalWrite( bPin, HIGH);
  digitalWrite( cPin, HIGH);
  digitalWrite( dPin, LOW);
  digitalWrite( ePin, LOW);
  digitalWrite( fPin, LOW);
  digitalWrite( gPin, LOW);
}
//*****************************************************************************
//************************* Display digit eight(8) on seven segment ***************
void eight()
{
  digitalWrite( aPin, HIGH);
  digitalWrite( bPin, HIGH);
  digitalWrite( cPin, HIGH);
  digitalWrite( dPin, HIGH);
  digitalWrite( ePin, HIGH);
  digitalWrite( fPin, HIGH);
  digitalWrite( gPin, HIGH);
}
//*****************************************************************************
//************************* Display digit nine(9) on seven segment ****************
void nine()
{
  digitalWrite( aPin, HIGH);
  digitalWrite( bPin, HIGH);
  digitalWrite( cPin, HIGH);
  digitalWrite( dPin, HIGH);
  digitalWrite( ePin, LOW);
  digitalWrite( fPin, HIGH);
  digitalWrite( gPin, HIGH);
}
//*****************************************************************************
//************************** Display digit zero(0) on seven segment ***************
void zero()
{
  digitalWrite( aPin, HIGH);
  digitalWrite( bPin, HIGH);
  digitalWrite( cPin, HIGH);
  digitalWrite( dPin, HIGH);
  digitalWrite( ePin, HIGH);
  digitalWrite( fPin, HIGH);
  digitalWrite( gPin, LOW);
}
//*****************************************************************************
```