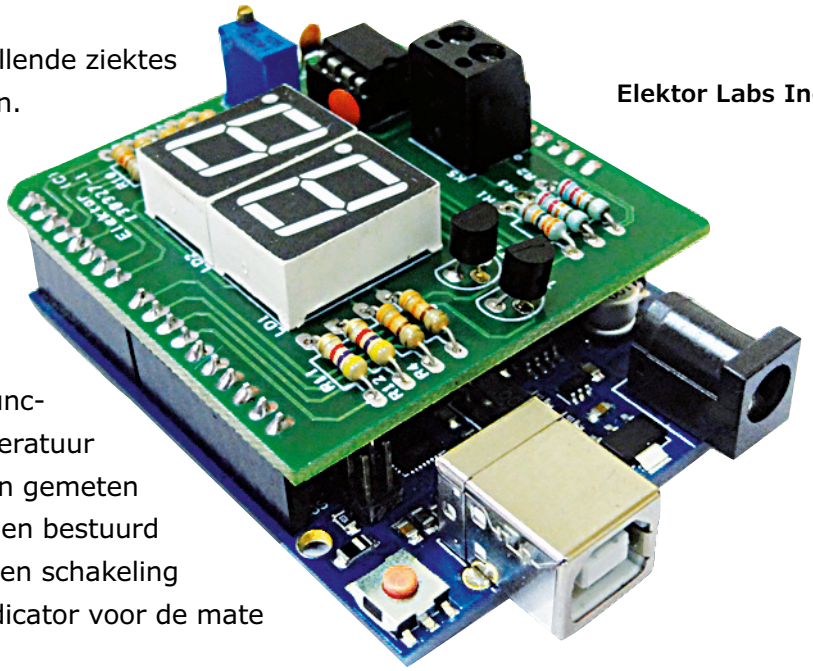
Bio-control Stresstester

Dr. Arduino meet de huidweerstand

Elektor Labs India



Het is bewezen dat stress verschillende ziektes en aandoeningen kan veroorzaken. Vandaar dat er vele vormen van 'autogene' ontspanningstraining op de markt zijn verschenen. Ook allerlei 'bio-feedback'-schakelingen zijn populair geworden. De gedachte hierachter is dat bepaalde lichamelijke functies zoals hartslag, lichaamstemperatuur en hersenactiviteit kunnen worden gemeten en daardoor bewust kunnen worden bestuurd door de patiënt. De hier beschreven schakeling gebruikt de huidweerstand als indicator voor de mate waarin iemand gespannen is.

De hardware is een elektronische schakeling die bestaat uit twee delen. Het eerste deel is een Arduino-shield met een 555 timer-IC om een inputsignaal voor de microcontroller te genereren. Het tweede blok is opgebouwd rondom een Arduino Uno R3-board. Dit verwerkt het door de 555 gegenereerde signaal en geeft het resultaat weer op twee aan de microcontroller gekoppelde 7-segment-displays.

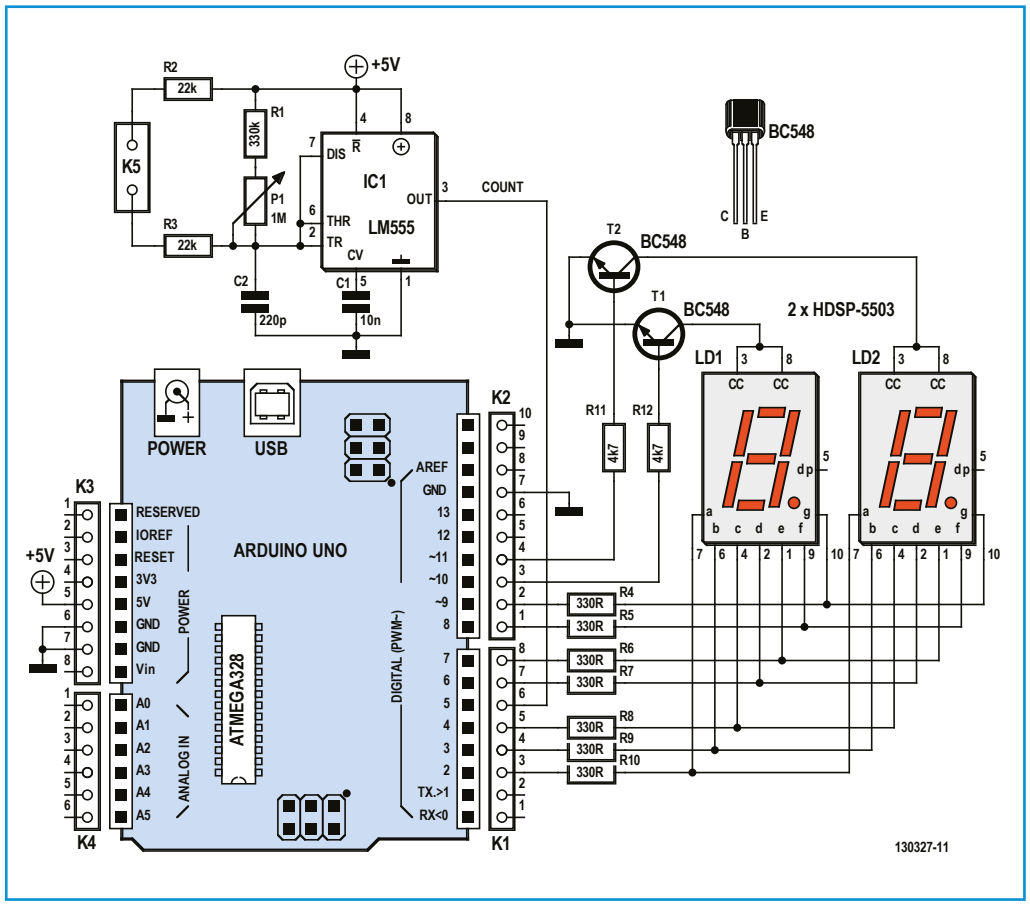
De werking

Zoals te zien is in het schema in **figuur 1**, is de LM555 (IC1) geconfigureerd als een astabiele multivibrator. Wanneer de proefpersoon de sensordraden die zijn verbonden met K5 aanraakt, neemt de frequentie op de output van de 555 toe. Elke verandering van de huidweerstand tussen de draden E1 en E2 zorgt ervoor dat ook de frequentie van de oscillator IC1 verandert.

Het uitgangssignaal van de oscillator gaat naar pen 5 van de Arduino Uno R3-kaart. De software in de microcontroller berekent de stress als een percentage (0–99) op basis van het met behulp van de Arduino freqCounter-bibliotheek berekende ingangssignaal. Dit stresspercentage wordt dan onder besturing van het Arduino Uno R3-board weergegeven als een getal van twee cijfers. Het tweecijferige display LD1/LD2 wordt gemultiplext met behulp van de transistors T1 en T2 die de gemeenschappelijke kathode-aansluitingen naar massa schakelen.

Software

De firmware voor het project is ontwikkeld met de Arduino IDE 1.0.5-software release en is helemaal geschreven in de taal C. Er is ook gebruik gemaakt van de "freqCounter"-bibliotheek voor Arduino voor de berekening en schaling van het ingangssignaal. Deze library is te downloaden van [1] en de Arduino-sketch van [2]. De complete sketch is ook aangehecht aan dit artikel.



Figuur 1.
Uit het schema blijkt al meteen dat dit project een Arduino-shield is.

We zullen nu de belangrijkste functieblokken in de code bespreken.

Setup

Deze functie definieert de configuratie van alle pennen die als output of input worden gebruikt.

- De pennen 2, 3, 4, 6, 7, 8 en 9 zijn geconfigureerd als outputs en zijn verbonden met de pennen a, b, c, d, e, f en g voor de 7 segmenten van het display.
- De pennen 10 en 11 zijn ook geconfigureerd als outputs en zorgen voor de keuze tussen de twee 7-segment-displays.
- Pen 5 is geconfigureerd als input en is verbonden met pen 3 van de 555.

Loop

Hier gebruiken we de functies uit de “freqcounter”-bibliotheek om een constante stroom van pulsen op te wekken die we kunnen tellen.
De getelde waarden worden als volgt geïnterpreteerd: Als de draden niet worden aangeraakt, worden er 500 pulsen geteld (dit is in te stellen met instelpotmeter P1). Deze waarde is het ‘nul’-niveau; alle waarden kleiner of gelijk aan 500 worden beschouwd als 0% stress.
Als de draden worden aangeraakt, neemt de tellerwaarde toe. De waarde kan niet groter worden dan 11.000, dat is de maximumwaarde die we krijgen als de draden worden kortgesloten. Bij alle waarden boven 11.000 wordt 99% weergegeven, wat betekent dat de draden zijn kortgesloten.
Het stress-percentage wordt berekend in verhouding tot deze maximale waarde.

Display_seg(unsigned long stress_per)

Deze functie krijgt de huidige stress-waarde als parameter mee. Hij zet de cijfers die berekend zijn in de functie loop() op de 7-segment-displays.

pickNumber(int x)

Deze functie wordt aangeroepen met een weer te geven cijfer als parameter. Hij kiest dan de subroutine die het cijfer weergeeft.

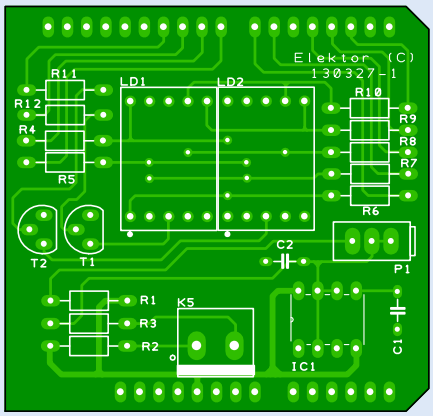
Onderdelenlijst

Weerstanden
R1 = 330kΩ 5% 0,25W
R2,R3 = 22kΩ 5% 0,25W
R4-R10 = 330Ω 5% 0,25W
R11,R12 = 4,7kΩ 5% 0,25W
P1 = 1MΩ instelpotmeter

Condensatoren
C1 = 10nF
C2 = 220pF

Halfgeleiders
IC1 = LM555CN/NOPB
T1,T2 = BC548
LD1,LD2 = SBC56-21EGWA (Kingbright) 7-segment LED-display, CC

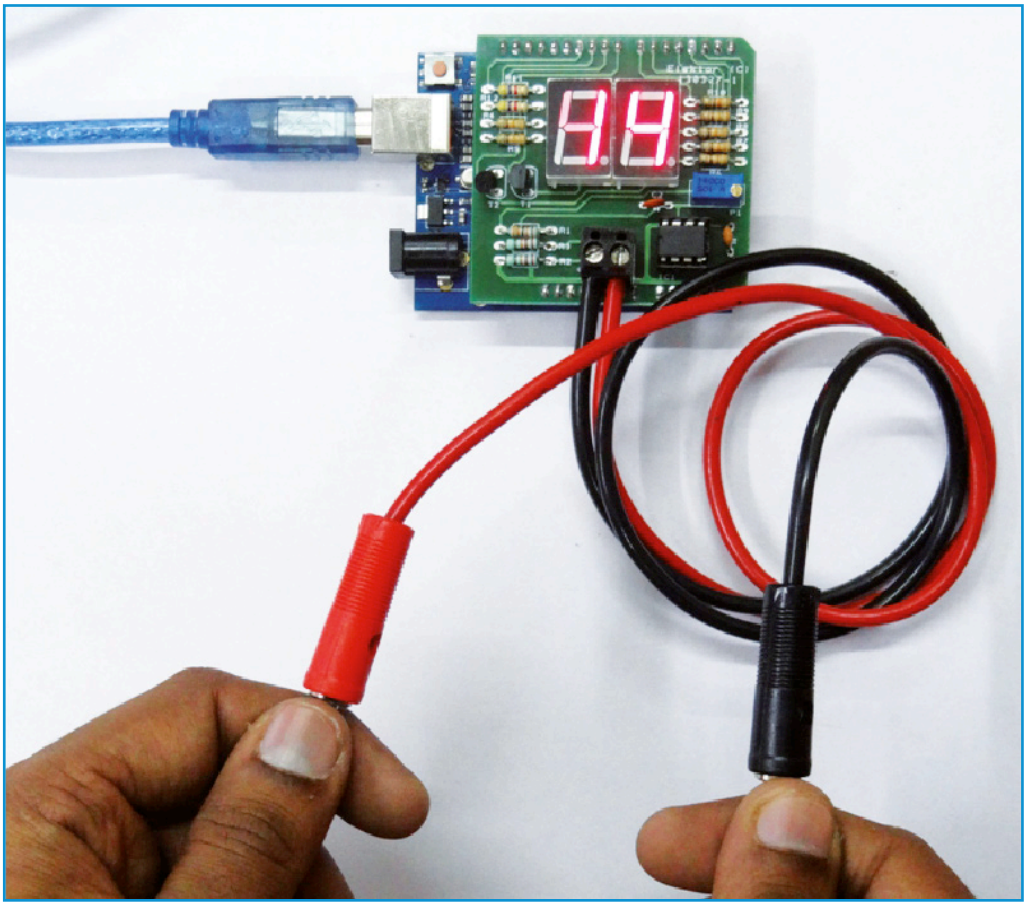
Diversen
K1,K3 = 8-polige pinheader
K2 = 10-polige pinheader
K4 = 6-polige pinheader
K5 = 2-polige schroefconnector voor printmontage
Arduino Uno R3
Print 130237



Figuur 2.
De voor het stressmeter-shield ontworpen print.

Opbouw

De schakeling wordt opgebouwd op de print die te zien is in **figuur 2**. Er worden alleen normale, bedrade componenten gebruikt dus de opbouw zou geen probleem moeten zijn. Als de kaart is bestukt en gecontroleerd, kunnen we hem op de Arduino aansluiten. Programmeer de Arduino met de firmware.



Figuur 3.
De Arduino-stressmeter in gebruik.

Het gebruik

Schakel de Arduino in. Sluit de sensordraden kort en controleer de waarde op het display: die zou 99 moeten zijn. Als dat niet het geval is, stel dan instelpotmeter P1 bij tot er 99 op het display staat bij kortgesloten draden. De schakeling is dan klaar voor gebruik.

We kunnen nu een stressniveau meten door met elke hand één draad vast te houden. Het is niet nodig hard in de draden te knijpen. Het meetresultaat verschijnt op het 7-segment-display zoals in figuur 3.

Praktijktests hebben aangetoond dat waarden onder 15% een normaal stressniveau aangeven; waarden groter dan 15% wijzen er op dat de proefpersoon min of meer gespannen is.

(130327)

Een alternatieve kalibratiemethode met behulp van USB

Verbind de schakeling via USB met de computer en start HyperTerminal.
Controleer de waarde van frq (frequentie-input) in het terminalvenster: die zou tussen 450 en 500 moeten zijn. Als dat niet het geval is, verdraai dan de instelpotmeter tot het klopt.

Weblinks

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

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

Programma listing

Stress_Tester.ino

Download van: www.elektor-magazine.nl/post

```
//*****************************************************************************
//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);
}
//*****************************************************************************
```