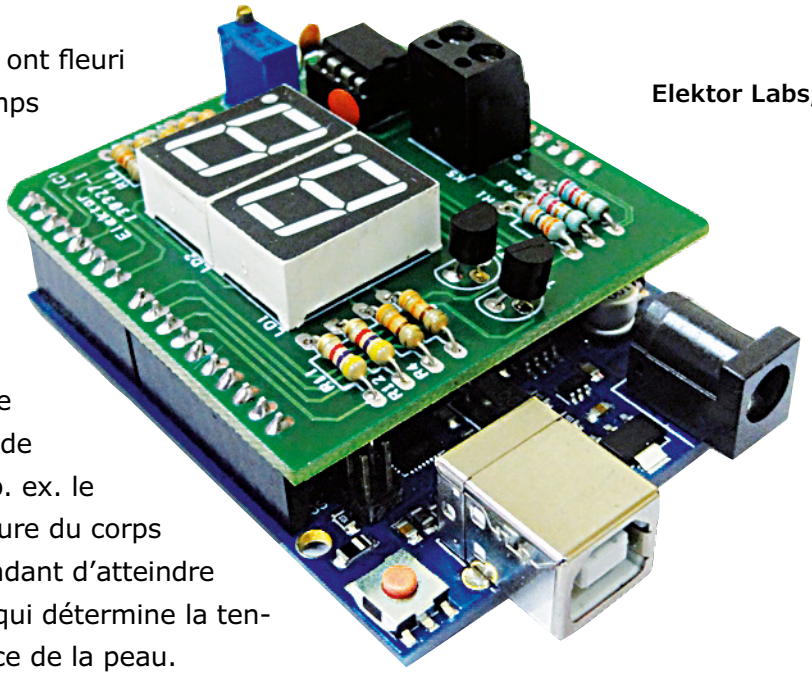
le stressthoscope du Dr Arduino

mesure la résistance de la peau

Elektor Labs, Inde



Les méthodes d'autorelaxation ont fleuri comme pâquerettes au printemps depuis que le stress a été reconnu comme source de divers maux et maladies. Les circuits de « rétroaction biologique » sont une autre voie pour le combattre. Ils nous permettraient d'agir de façon consciente sur certaines de nos fonctions physiologiques, p. ex. le rythme cardiaque, la température du corps ou l'activité cérébrale. En attendant d'atteindre le niveau yogi, voici un circuit qui détermine la tension par mesure de la résistance de la peau.

Le matériel est composé de deux blocs principaux. Le premier est une carte d'extension Arduino (un *shield*) équipée d'un LM555 chargé de produire le signal d'entrée pour le microcontrôleur. Le second est construit autour d'une carte Arduino Uno R3. Il reçoit le signal produit par le 555, le traite de façon idoine et affiche le résultat sur un afficheur à 7 segments interfacé avec le microcontrôleur.

Fonctionnement

Le temporisateur LM555 (IC1, **fig. 1**) est configuré en multivibrateur astable. Sa fréquence de sortie augmente lorsque le sujet touche les fils E1 et E2 reliés à K5. Toute variation de la résistance de la peau mesurée entre les fils entraîne une variation de la fréquence de l'oscillateur construit autour d'IC1. La sortie de l'oscillateur est injectée dans la broche 5 de la carte Arduino Uno R3. Le programme exécuté par le microcontrôleur calcule le « pourcentage de stress » (sur une échelle allant de 0 à 99 %) sur la base du signal d'entrée. Ce signal est mesuré, traité et mis à l'échelle à l'aide de la bibliothèque Arduino *FreqCounter*. Le pourcentage est ensuite affiché par la carte Uno R3 sous la forme d'un nombre à deux chiffres. L'afficheur à deux unités LD1/LD2 est multiplexé grâce aux transistors T1 et T2 qui commutent les broches *CC* (cathodes communes) à la masse.

Programme

Le code du micrologiciel a été écrit en C avec la version 1.0.5 de l'EDI Arduino. La bibliothèque Arduino *FreqCounter* peut être téléchargée [1]. Le croquis Arduino complet est joint à l'article, mais vous pouvez également le télécharger depuis [2]. Les fonctions principales du croquis sont décrites ci-dessous.

Fonction setup()

La fonction *setup()* définit la configuration de chaque broche utilisée, entrée ou sortie.

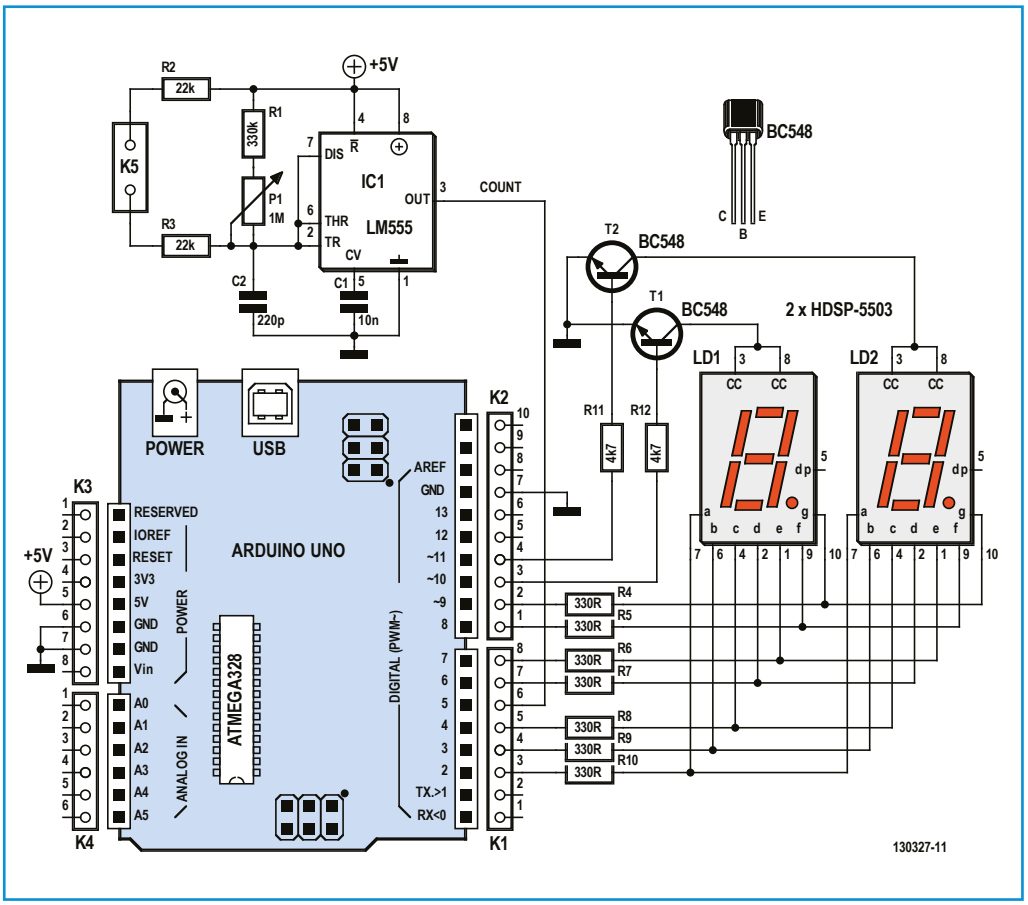


Figure 1.
Schéma de principe du projet. Incontestablement un *shield* Arduino.

Les broches 2, 3, 4, 6, 7, 8, 9 sont définies comme sorties et reliées aux broches a, b, c, d, e, f, et g de l'afficheur à 7 segments.
Définies elles aussi comme sorties, les broches 10 et 11 sont reliées aux broches de sélection de l'afficheur à deux unités.
La broche 5 est configurée comme entrée et reliée à la broche 3 du 555.

Fonction loop()

La fonction *loop()* utilise les fonctions de la bibliothèque *FreqCounter* pour obtenir un nombre constant d'impulsions à compter.
Le nombre d'impulsions vaut 500 lorsque le sujet ne touche pas les fils. Cette valeur se règle avec le potentiomètre P1 et sert de « niveau zéro » : toute valeur inférieure ou égale à ce niveau est considérée comme correspondant à 0 % de stress.
Le nombre d'impulsions augmente lorsque le sujet touche les fils. Ce nombre ne peut pas dépasser 11 000 (voir commentaires du listage) ; il correspond à la valeur maximale obtenue lorsque les fils sont mis en court-circuit. Le code affiche un pourcentage de stress de 99 % pour toute mesure supérieure à la valeur de 11 000 indiquant que les fils sont court-circuités.
Le pourcentage de stress est calculé en fonction de la valeur maximale obtenue après la mise en court-circuit des deux fils.

Fonction Display_seg(unsigned long stress_per)

L'argument passé à la fonction *Display_seg()* est la valeur du pourcentage de stress. Cette fonction récupère les chiffres du nombre envoyé par la fonction *loop()* et les affiche sur l'afficheur à 7 segments.

Fonction pickNumber(int x)

L'argument passé à la fonction *pickNumber()* est le chiffre à afficher. Elle sélectionne le nombre à afficher.

Construction

La carte de circuit imprimé sur laquelle repose le « stressthoscope » est reproduite sur la **figure 2**. L'assemblage ne devrait poser aucun problème puisque seuls des composants tra-

Liste des composants

Résistances
R1 = 330 kΩ 5 % 5 % 0,25 W
R2, R3 = 22 kΩ 5 % 0,25 W
R4 à R10 = 330 Ω 5 % 0,25 W
R11, R12 = 4,7 kΩ 5 % 0,25 W
P1 = ajus. 1 MΩ

Condensateurs
C1 = 10 nF
C2 = 220 pF

Semi-conducteurs
IC1 = LM555CN/NOPB
T1, T2 = BC548
LD1, LD2 = afficheur 7 segments à LED, cathode commune, SBC56-21EGWA (Kingbright)

Divers
K1, K3 = embase mâle à 8 broches
K2 = embase mâle à 10 broches
K4 = embase mâle à 6 broches
K5 = bornier à vis 2 voies pour CI
Arduino Uno R3
circuit imprimé 130237

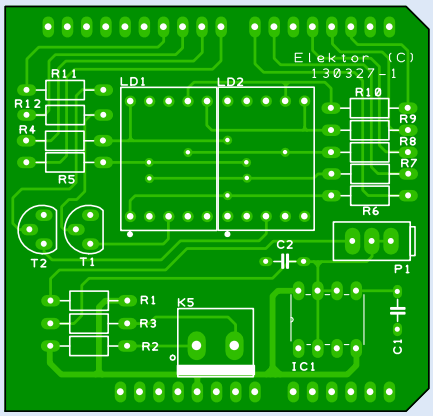


Figure 2.
Dessin du circuit imprimé de l'analyseur de stress.

versants sont utilisés. Vérifiez votre montage, enfichez la carte sur l'Arduino, puis programmez le microcontrôleur avec le croquis.

Mode d'emploi

Mettez l'Arduino sous tension. Court-circuitez les fils et vérifiez que la valeur affichée est bien 99. Si elle est différente, gardez les fils

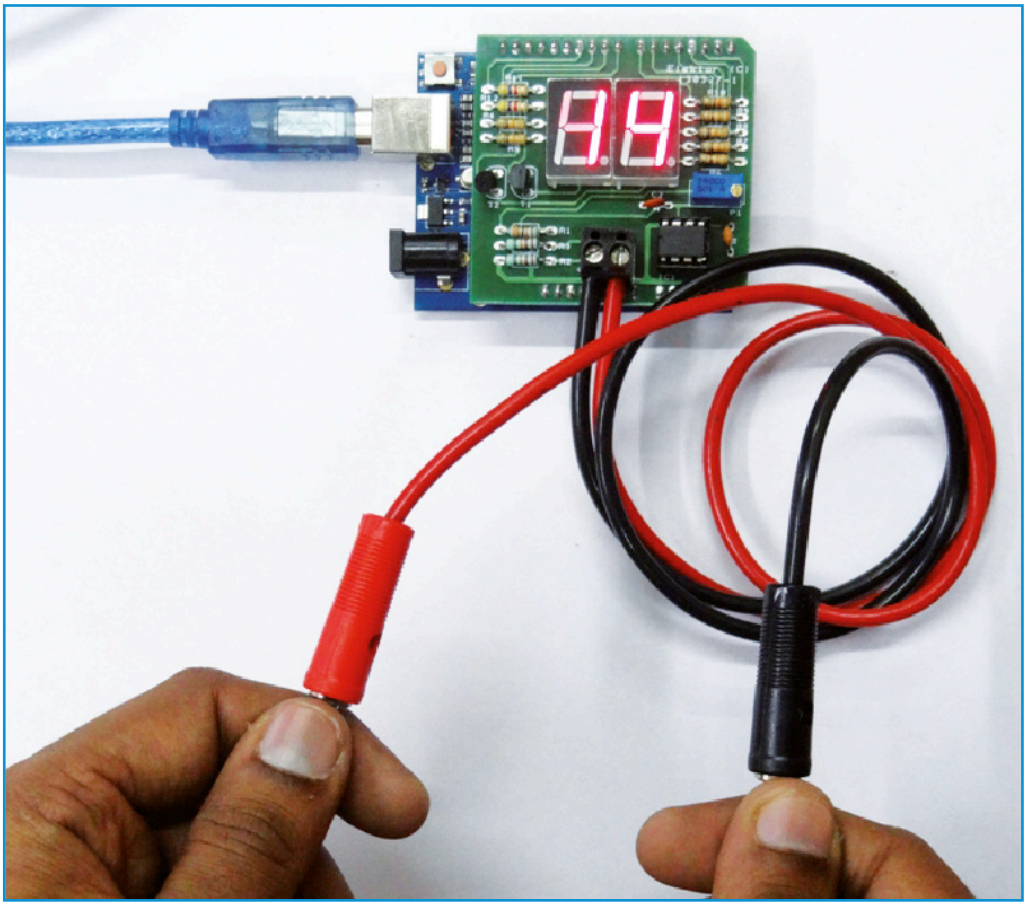


Figure 3.
Le « stressthoscope » du Dr Arduino en pleine mesure.

en court-circuit et réglez l'ajustable P1 du *shield* jusqu'à ce que la valeur affichée soit 99. Le circuit est alors prêt.
Pour mesurer votre niveau de stress, tenez simplement un fil dans chaque main en le serrant normalement. Votre pourcentage de stress s'affiche (**fig. 3**).

Il a été déduit de plusieurs essais que des valeurs inférieures à 15 % indiquent un niveau de stress normal. Toute valeur supérieure à 15 % indique *a contrario* un certain niveau de stress.

(130327 – version française : Hervé Moreau)

Autre méthode de calibrage : **par USB**

Reliez par USB le circuit à votre ordinateur, puis ouvrez le *Moniteur série* de l'EDI Arduino ou le programme HyperTerminal.
Le terminal devrait afficher pour la variable *frq* (fréquence d'entrée) une valeur comprise entre 450 et 500. Si ce n'est pas le cas, réglez l'ajustable de façon à ce que la valeur de *freq* tombe dans cet intervalle lorsque vous ne touchez pas les fils.

Liens

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

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

Croquis

Stress_Tester.ino

Téléchargement : www.elektor-magazine.com/post

```
//*************************************************************************************************
//Project Name: Stress Tester
//Microcontroller:ATmega328p(Arduino Uno R3)
//This project is used to measure the percentage of stress of a human body on the basis of skin
//resistance. In order to achieve this we have used a circuit to generate the input signal;
//it 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 units connected to
//pin 2,3,4,6,7,8,9,10,11 of Arduino Uno board. Human body stress cannot exceed a frequency of
//32 kHz 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 from 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 is 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 be more
                            //than 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);
}
//********************************************************************************
```