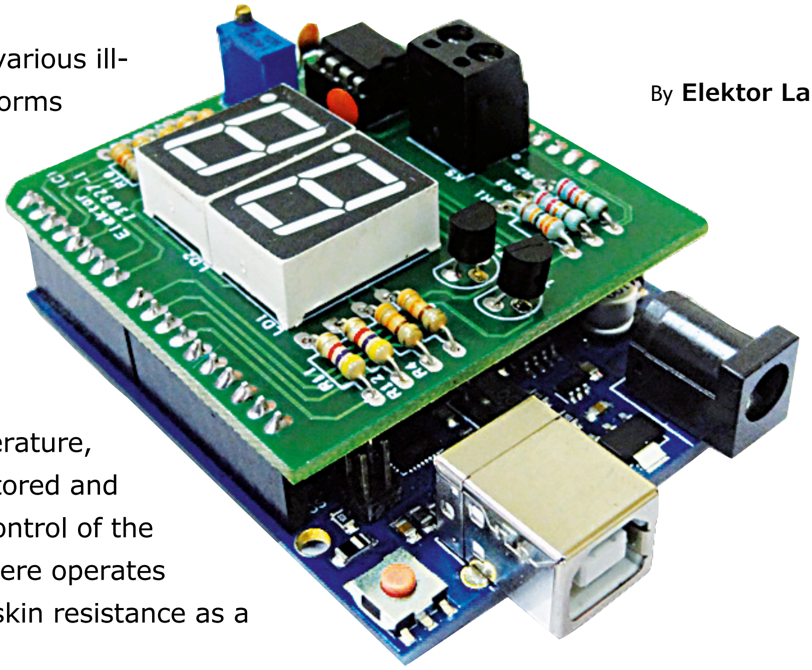
Bio-control Stress Tester

Dr. Arduino measures skin resistance

By **Elektor Labs** India

With stress a proven cause of various illnesses and afflictions various forms of 'autogenic' training as a means of relaxation have hit the consumer market. Different types of 'Bio-feedback' circuits have also become popular—the idea being that certain physiological functions such as heartbeat, body temperature, and brain activity can be monitored and brought under the conscious control of the patient. The circuit described here operates on the principle of monitoring skin resistance as a measure of how tense you are.



The hardware is an electronic circuit made up of two main parts. The first part is an Arduino shield comprising a type 555 timer IC to generate the input signal for the microcontroller. The second block is built around an Arduino Uno R3 board. It accepts the signal generated by the 555, manipulates it and displays the result on two 7-segment displays interfaced with the microcontroller.

How it works

Referring to the schematic in **Figure 1**, the LM555 in position IC1 is configured as an astable multivibrator. When the person to be examined touches the sense wires connected to K5 the frequency at the output of the 555 goes up. A variation in the skin resistance between wires E1 and E2 causes the frequency of the oscillator built around IC1 to vary.
The output of the oscillator is fed to pin 5 of the Arduino Uno R3 board. The software executed by the microcontroller calculates the amount of stress as a percentage (0–99) on the basis of the input signal calculated using the freqCounter library of Arduino. This stress percentage is then displayed as a 2-digit number under control of the Arduino Uno R3 board. The 2-digit display LD1/LD2 is multiplexed with the help of transistors T1 and T2 switching the CC (common cathode) terminals to ground.

Software

The firmware for the project was developed using the Arduino IDE 1.0.5 software release and coded entirely using the C language. It also uses the "freqCounter" library provided by Arduino, for calculation and scaling of the input signal. The library can be downloaded from [1], and the Arduino sketch from [2]. The complete sketch is also appended to this article.
The main function blocks in the code are described below.

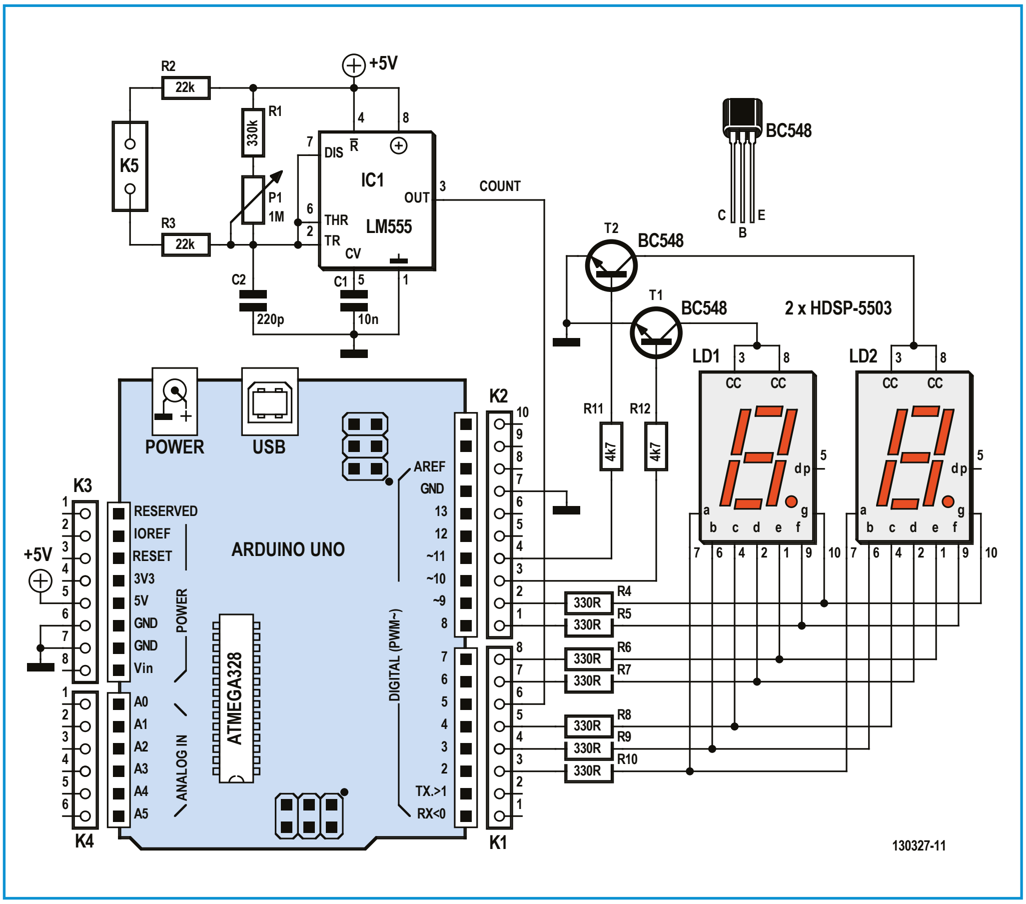


Figure 1.
From the schematic it's blatantly obvious this project is an Arduino shield.

Setup
This function defines the configuration of each pin to be used as output or input.

- Pins 2, 3, 4, 6, 7, 8, 9 are configured as outputs and are connected to the pins a, b, c, d, e, f, and g representing the 7 segments of the display.
- Pins 10 and 11 are also configured as outputs and are connected as display select pins for the two 7-segment displays.
- Pin 5 is configured as an input and is connected to pin 3 the 555.

Loop
Here we employ the functions of the "freqcounter" library to get a steady supply of pulses to count.
Each pulse count value is manipulated as follows. When the wires remain untouched the pulse count is 500 (adjusted using trimpot P1) and this value is taken as the 'zero' level, meaning any value less than or equal to this value is considered as 0% stress.
When the wires are touched the count increases. The value cannot exceed 11,000, which is the maximum value obtained when the wires are shorted. Anything above 11,000 causes 99% to be displayed indicating that the two wires are shorted.
The percentage of stress is calculated with respect to the maximum value obtained after shorting the two wires.

Display_seg(unsigned long stress_per)
Any argument passed to this function is the stress percentage value. This function is used to get the digits of the number sent from the loop() function and display on to the seven segment display.

pickNumber(int x)
Any argument passed to this function is the digit to be displayed. This function selects the number to be displayed.

Construction
The circuit board to build the stress meter shield is shown in **Figure 2**. Only through hole

Component List

Resistors
R1 = 330kΩ 5% 5% .25W
R2,R3 = 22kΩ 5% .25W
R4-R10 = 330Ω 5% .25W
R11,R12 = 4.7kΩ 5% .25W
P1 = 1MΩ preset

Capacitors
C1 = 10nF
C2 = 220pF

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

Miscellaneous
K1,K3 = 8-pin pinheader
K2 = 10-pin pinheader
K4 = 6-pin pinheader
K5 = 2-way PCB screw terminal block
Arduino Uno R3
PCB ref. 130237

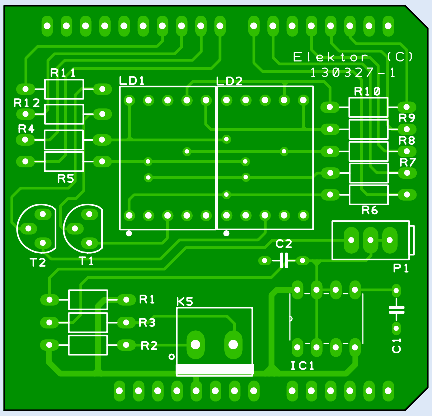


Figure 2.
Circuit board designed for the stress meter shield.

components are used so assembly should not be a problem. After the board has been populated and inspected, plug it onto the Arduino. Program the Arduino with the firmware.

Practical use

Switch on power to the Arduino. Short the sense wires and check the value on the display—it should read 99. If not, adjust trim-

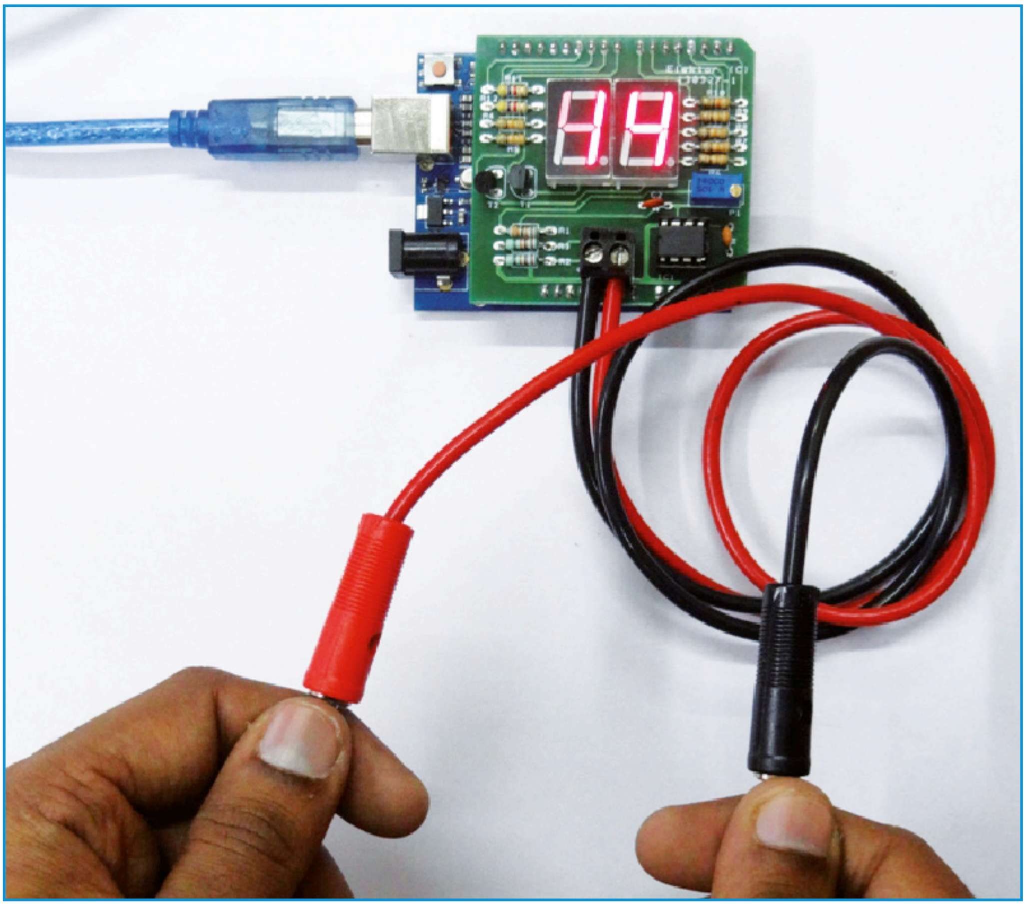


Figure 3.
The Arduino based stress meter in use.

pot P1 on the shield board such until 99 is displayed when the sense wires are shorted. The circuit is then ready for use.
To perform a stress level test, hold a wire in each hand with normal pressure, the percentage of stress will be displayed on the 7-segment display like in **Figure 3**.

From practical tests it was deduced that values below 15% indicate normal stress levels; anything higher than 15% should be taken as an indication of the patient being stressed to a degree.

(130327)

Alternative calibration method using USB

Connect the circuit to the computer via USB and open the serial monitor or HyperTerminal.
Check the value of frq (frequency input) on the terminal—it should read between 450 and 500. If the value is not in range, adjust the trimpot such that the value comes in the above range when the wires are untouched.

Web Links

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

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

Program Listing

Stress_Tester.ino

Download from: 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 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);
}
//************************************************************************************
```