Paper-based, Capacitive Touch Pads

Aaron D. Mazzeo[1], Will B. Kalb[1], Lawrence Chan[1], Matthew G. Killian[1],

Jean-Francis Bloch[2], Brian A. Mazzeo[3], and George M. Whitesides[1,4,*]

[1]Department of Chemistry and Chemical Biology, Harvard University

[2]Institut National Polytechnique de Grenoble

[3]Department of Electrical and Computer Engineering, Brigham Young University

[4]Wyss Institute for Biologically Inspired Engineering at Harvard University

*corresponding author: gwhitesides@gmwgroup.harvard.edu

Supporting Information:

Characterization of Metallized Paper

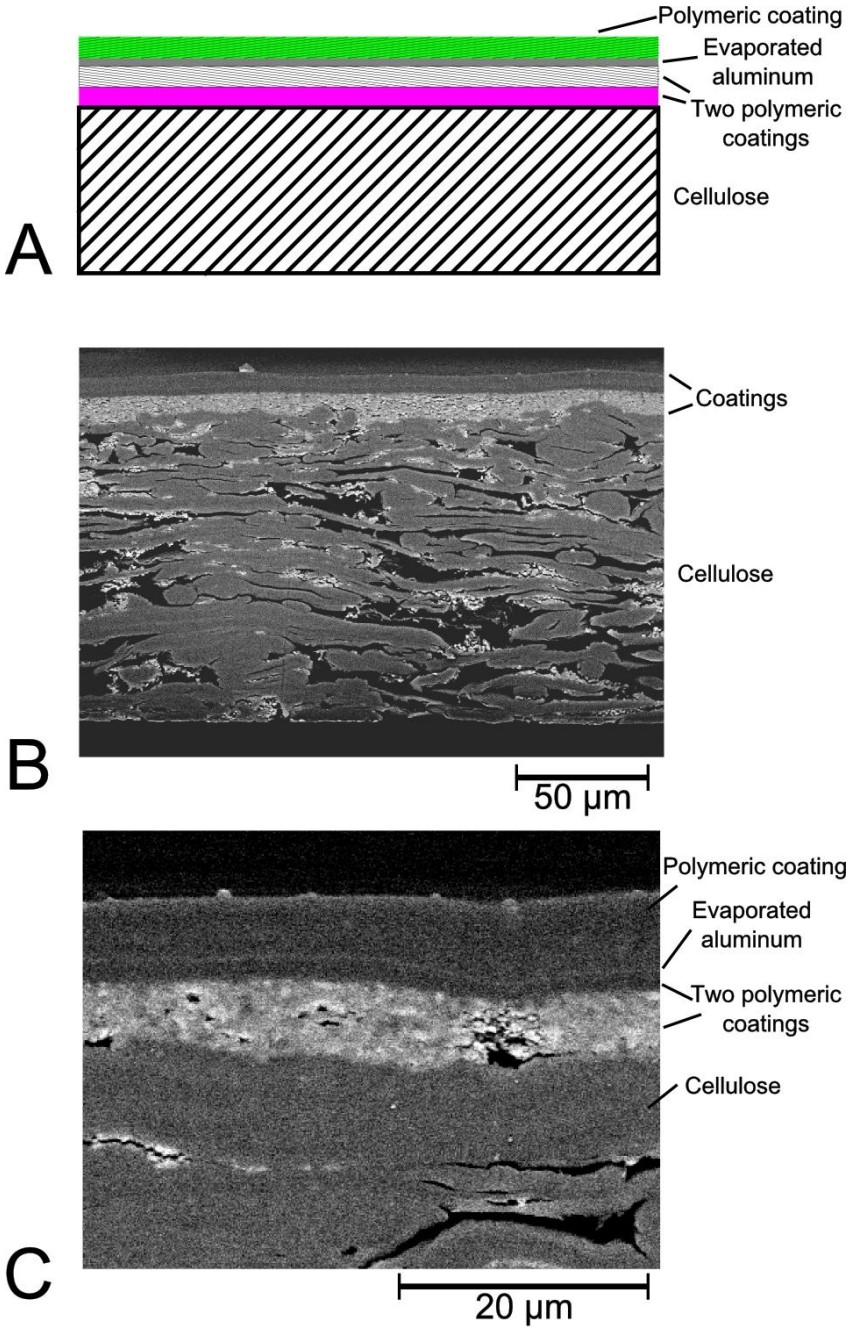
For the metallized/aluminized paper used in this work, we measured resistances along pieces with a length of 50 mm and widths ranging from 2 to 25 mm. For the widest set (width of 25 mm) of 7 separate pieces, we measured a resistance of 5±0.6 Ohms (mean ± 1 standard deviation). For the thinnest set of 7 separate pieces, we measured a resistance of 58±7 Ohms (see SI Figure 2). Using a resistivity of 26 nOhms/m for aluminum, the average calculated thickness across all 49 samples was 13 nm±2 nm.

Etching and Cutting Components

The process of assembling a touch pad consisted of cutting the individual layers from metallized paper, double-sided tape, and if desired, additional paper for structural support. For this work, we used a VLS3.50 laser cutter (50-watt laser) from Universal Laser Systems with the standard 2.0-in. lens. To array buttons on a single sheet of metallized paper, the laser cutter ablated lines in the thicker Vacumet A-550 metallized paper without cutting completely through the paper to form conductive traces or separate regions of conductivity on the paper. With the thin metallized paper (Vacumet A-238) and non-uniformities in the cutting power of the laser cutter, it was difficult to avoid cutting through the entire piece of paper in certain regions while still removing the aluminum. In some cases, ablation at low power left perforated lines through the paper. Cutting through the thickness of the thin paper was not usually an issue for fabricating buttons when using double-sided tape to adhere the patterned metallized paper to another surface.

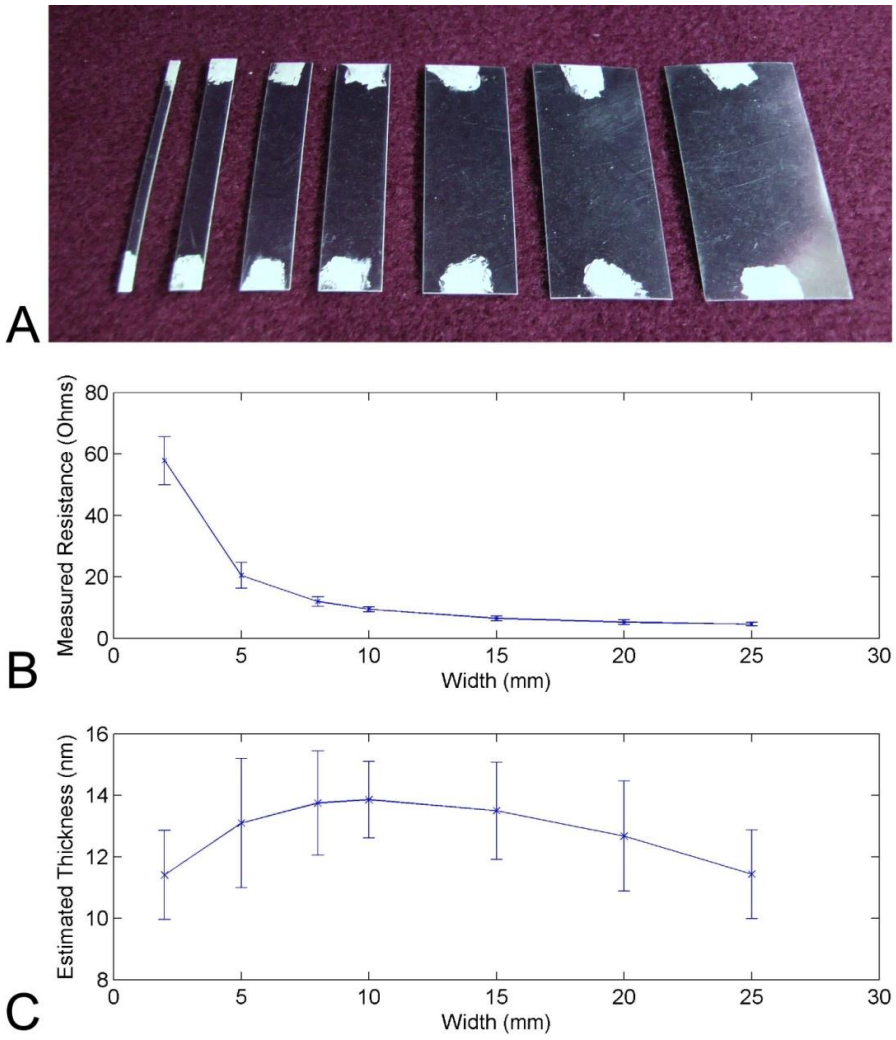
We cut through Vacumet A-238 (aluminized paper with a thickness of 56 microns) with the settings of 12% power, 80% speed, and 500 pulses per inch. To ablate through

SI Figure 1



SI Figure 1: Morphology of metallized paper from Vacumet Corpation. (A) The metallized paper consists of cellulose with 3 polymeric coatings and 1 layer of evaporated aluminum. (B) SEM image showing the entire thickness of a cross-sectional slice of metallized paper (Vacumet A-550). (C) SEM image showing a cross-sectional portion of Vacumet A-550. The layer of evaporated aluminum is much thinner than the polymeric layers (see SI Figure 2).

SI Figure 2



SI Figure 2: Measurements of the resistances and estimated thicknesses of cut samples of metallized paper. (A) Image of one set of samples with a length of 50 mm and varied widths of 2, 5, 8, 10, 15, 20, and 25 mm. The spots at each end of the samples are regions of silver-based conductive paste. (B) Averaged resistances (BK Precision 5370) of seven different samples for each specified width with the error bars showing ±1 standard deviation. (C) Estimated thicknesses for the evaporated aluminum, where the thickness is $h = \rho L/(wR)$, ρ is the resistivity of the aluminum (26 nOhms/m), L is the length of the samples (50 mm), w is the width of the samples, and R is the measured resistance. The number of samples is 7 (n=7) at each point shown on the graph. The averaged, estimated thickness of all the samples (n=49) is 13 nm with a standard deviation of 2 nm.

the conductive layer without cutting out parts completely, we used 3.5%, 80% speed, and 500 pulses per inch on the laser cutter or 3% power, 70% speed, and 500 pulses per inch. To cut through the Vacumet A-550 (thickness of 136 microns) with settings of 15% power, 80% speed, and 500 pulses per inch. We found settings of 6% power, 80% speed, and 500 pulses per inch adequate for cutting traces in the paper without cutting completely through the paper.

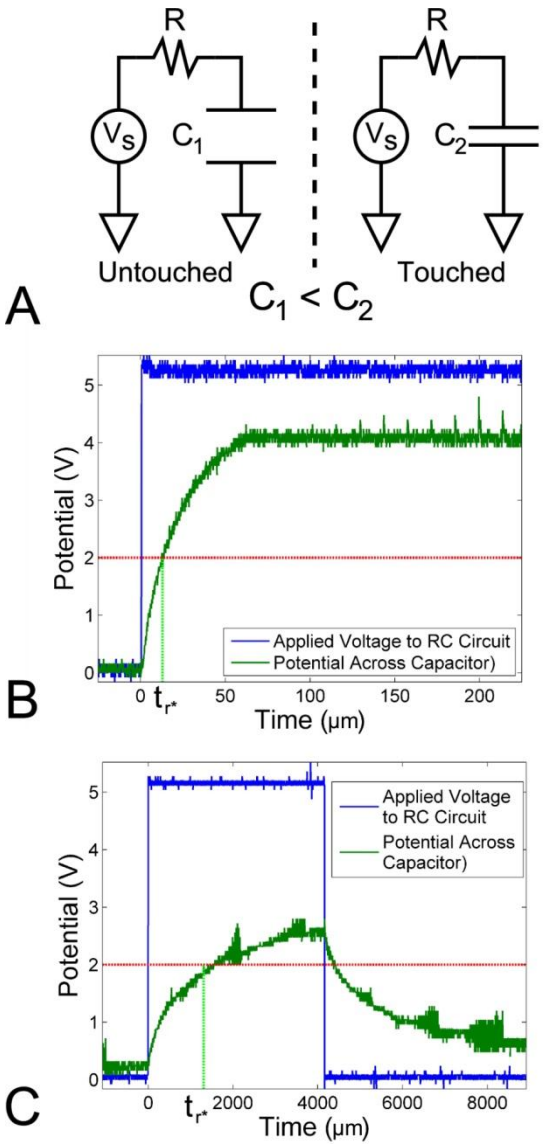
Interfacing to external electronics

We brushed on conductive silver paste (Silver Conductive Adhesive 503 from Electron Microscopy Sciences) to connect traces or tabs on the metallized paper electrically with external electronics through conductive pads or wired leads. Initially, we scraped away or dissolved a portion of the polymeric coating above the evaporated aluminum with a razor blade or 2 µL of acetone. Nevertheless, we found that the solvent in the conductive adhesive dissolved through the insulating coating sufficiently to form an electrically conductive bond.

Thresholds for registering touch

The Arduino-based systems calculated delays or elapsed times and compared them with temporal thresholds to determine changes in effective capacitance. To measure capacitance in terms of standard units (Farads), we estimated an effective time constant t_{r*}, which corresponded to the time required for the voltage to reach 2 V out of the applied 5 V. For an ideal RC circuit (SI Figure 3A), $t_{r*}=\ln(5/3)RC$ or $0.51RC$, where R and C represent the resistance and capacitance, respectively. Notably, the measured potential with an oscilloscope – not used to acquire any data shown in the main text – were not ideal (SI Figures 3B and C) possibly because of the complex nature of the impedance of the contact between the finger and the button, the impedance through or along the finger itself, or the finger-body-ground impedance. For Figure 4 C, the thresholds were 95 µs for key 0, 97 µs for key 1, 95 µs for key 2, 102 µs for key 3, 106 µs for key 4, 100 µs for key 6, 100 µs for key 6, 106 µs for key 7, 106 µs for key 8, and 100 µs for

SI Figure 3



SI Figure 3: (A) The capacitance of a button (type shown in Figure 2E) increases when touched. To measure the change in capacitance of a button, we used a resistor in series with the capacitive button. (B) While applying a step in potential across the resistor-capacitor (RC) circuit, we observed the potential across the capacitive button increase with a time constant equivalent to the product of the resistance and capacitance in the circuit (data shown is from the button in Figure 2 E). The required amount of time for the capacitor to charge to 2 V (threshold for Arduino's input to go from 0 to 1) required time t_{r^*}. For the shown measurement of the untouched button, t_{r^*} =13 μs. (C) When touched, the capacitance across the button increased and t_{r^*} =1300 μs.

key 9. In other versions of the programmed Arduino (e.g., Figure 4 B, Figure 5, and Figure 6), we implemented an averaging routine (see http://www.arduino.cc/en/Tutorial/Smoothing) to adjust thresholds in real-time based on measurements of effective capacitance of the untouched keys.

Measurements on single buttons

We constructed the buttons shown in Figure 2; one button consisted of parallel plates (Figure 2 A and B), while the other button consisted of a single layer of metallized paper with etched traces (Figure 2 D and E). The material for the top layer of Figure 2 B and the entire device shown in Figure 2 E was the thin metallized paper (Vacumet A-238), while the bottom layer of Figure 2 B was the thicker metallized paper (Vacumet A-550). To test the devices, we mounted them to a manila folder with double-sided tape.

To measure the changing capacitance of the devices shown in Figure 2 C and F, we placed a resistor of 1.01 MOhm (measured with R.S.R 308B Multimeter at 21°C, 50% RH) in series with the capacitive button. Using Arduino, it was possible to calculate the required amounts of time for the potential across the capacitor to reach 2 V. Each point in the graphs is the mean of the effective capacitance of the button calculated over five seconds of sampling.

For measurements on two-layered devices (22 °C, 50 % RH), seven measurements without being touched had a mean capacitance of 206 pF and a standard deviation of 15 pF for n=4746 (n is the number of samples taken during the five seconds of sampling). Seven measurements while touched with a bare finger had a mean capacitance of 331 pF and a standard deviation of 33 pF for n=4634.

For measurements on a button with a single layer of metallized paper (22 °C, 49% RH), seven measurements without being touched had a mean capacitance of 43.8 pF and a standard

deviation of 4.1 pF for n=5564. Seven measurements while touched with a bare finger had a mean capacitance of 2120 pF and a standard deviation of 520 pF for n=3837.

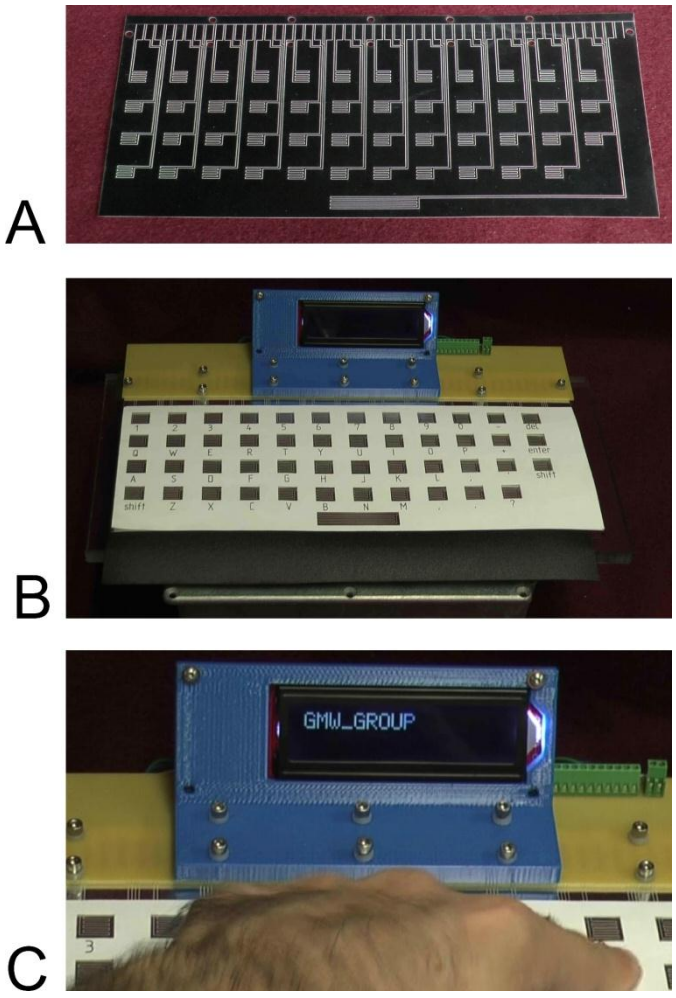
To make an approximate comparison for the sensitivities between the two-layered button and the button made from a single layer of metallized paper, we calculated an upper limit for the capacitance of an untouched button (mean capacitance plus 3X the standard deviation of the measurements) with a lower limit for the capacitance of a touched button (mean capacitance minus 3X the standard deviation of the measurements). For the two-layered button, we found the upper limit of 251 pF (206 pF+ 3×15 pF) for the untouched button to be greater than the lower limit of 238 pF (331-3×17pF) for the touched button. For the button fabricated with inter-digitated electrodes on a single piece of metallized paper, we found upper limit of 56.1 pF (43.8 pF + 3×4.1) for the untouched button to be less than the lower limit of 560 pF (2120pF-3×520 pF) for the touched button. Thus, this comparison suggested that the touch pads with a single layer of metallized paper and inter-digitated electrodes/buttons were more sensitive than the buttons fabricated using two layers of metallized paper.

Cost

Thin metallized paper costs less than US $0.25 per square meter, and thicker metallized paper costs less than US $0.75 per square meter. The 10-button keypad requires between 2-3 minutes of time on the laser cutter. An estimate for the mass of conductive paste per contact is 10 mg, and the conductive paste costs US $39 per 15 g. Thus, the conductive paste cost about US $0.03 per button. A cheaper alternative might be carbon ink. An estimated cost of the thicker metallized paper for the 10-button keypad (0.01 m^2) is US $0.08, and an estimate cost for the conductive paste is $0.33 (ten buttons but 11 contacts including electrical ground). Thus, the cost of materials for the 10-button keypad is less than $0.50.

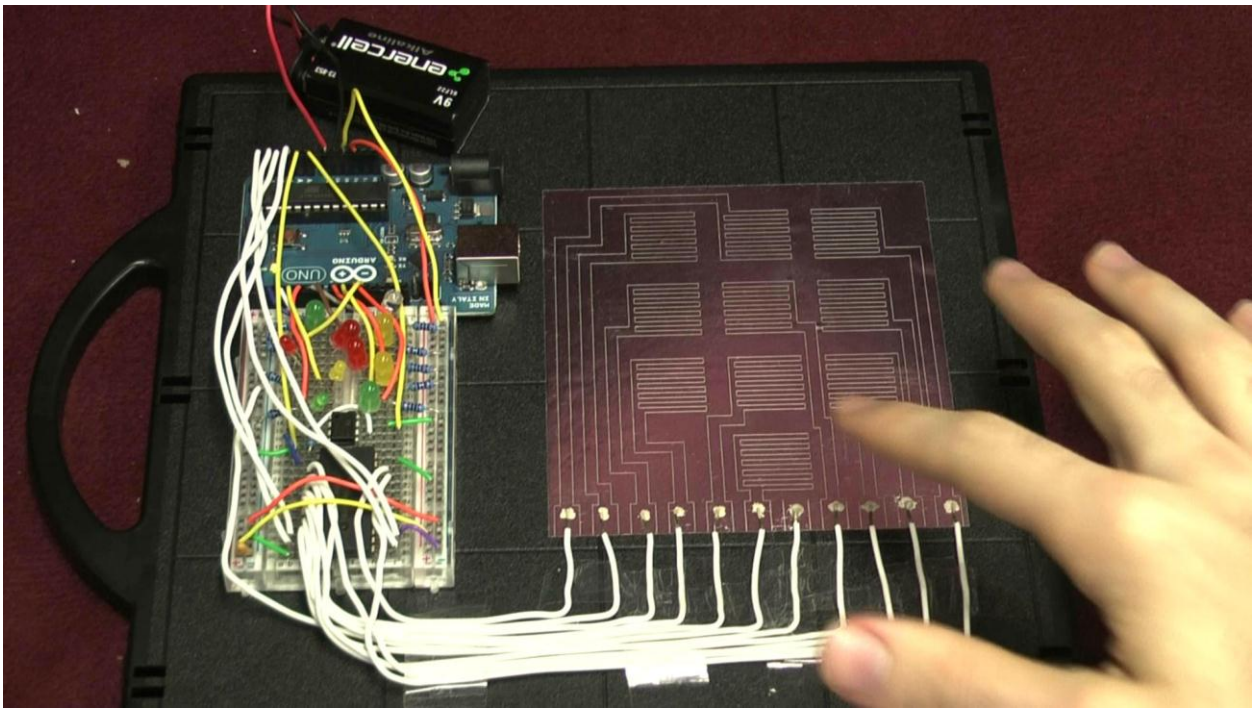
The Arduino costs about US $25 in its pre-made form, but less user-friendly controllers could cost less. The breadboard costs about US $7. The operational amplifier costs less than US $3 each in small quantities. The software for programming the Arduino is open-source (free). The buzzer for the alarmed box costs less than US $4. The total cost of the reusable electronics is less than US $50, and the electronic components could cost much less with appropriate design for manufacturing.

SI Figure 4



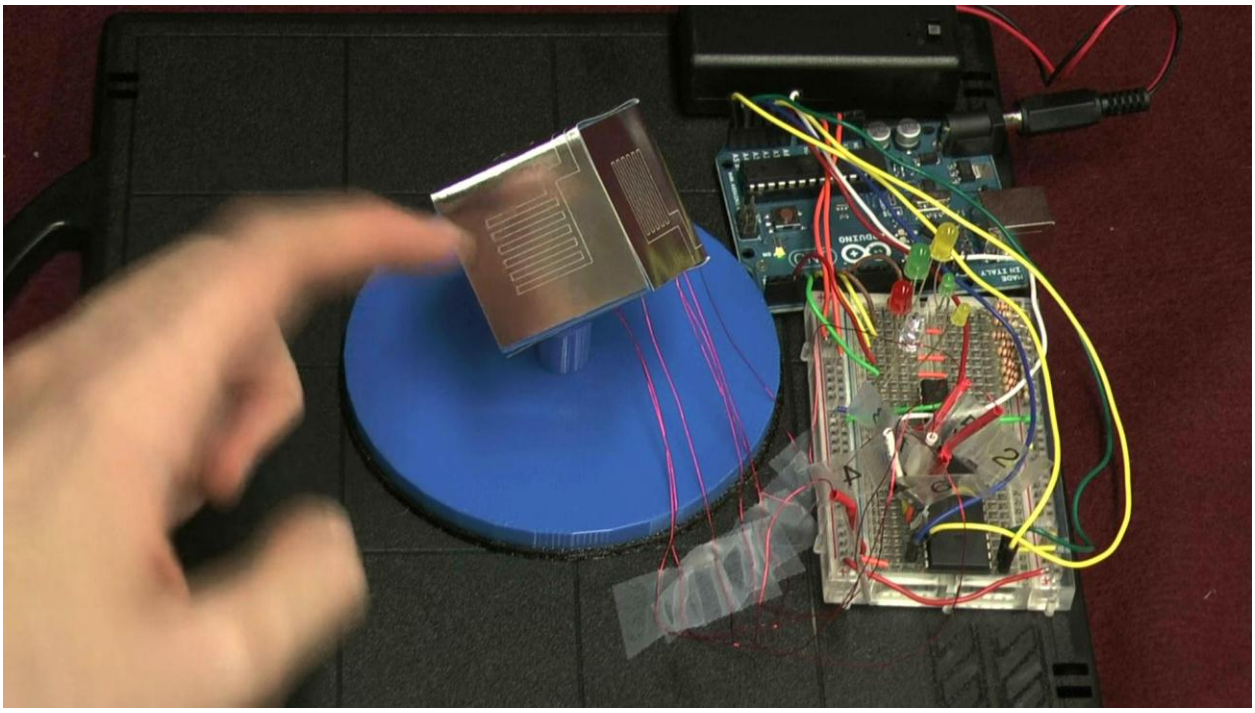
SI Figure 4: A functional QWERTY-based keyboard with 48 keys. (A) Metallized paper with etched traces and buttons of the QWERTY-based keyboard before its assembly. Individual traces (etched with the laser cutter) lead to a button for each key. The width of the keyboard was approximately 25 cm wide and 10 cm tall (not including the extra printed circuit board). (B) The assembled keyboard with a patterned piece of paper designating the location of each key and also allowing contact between the metallized finger and a bare finger. Below the metallized paper is foam, acrylic, and an additional printed circuit board with an Arduino board, multiplexers, and other appropriate electronics. (C) The user after just having finished typing "GMW Group" (see SI Video 4).

SI Video 1



SI Video 1: This video shows four different clips showing a functional 10-button keypad. The first two clips show bare fingers touching keys, while the last two clips show gloved fingers touching the keys. Each key addresses an individual LED. Simultaneous touching of multiple keys lights multiple LEDS. The "ThresholdFactor" (variable in the code included at the end of the Supporting Information) had a value of 1.12 for all four clips.

SI Video 2

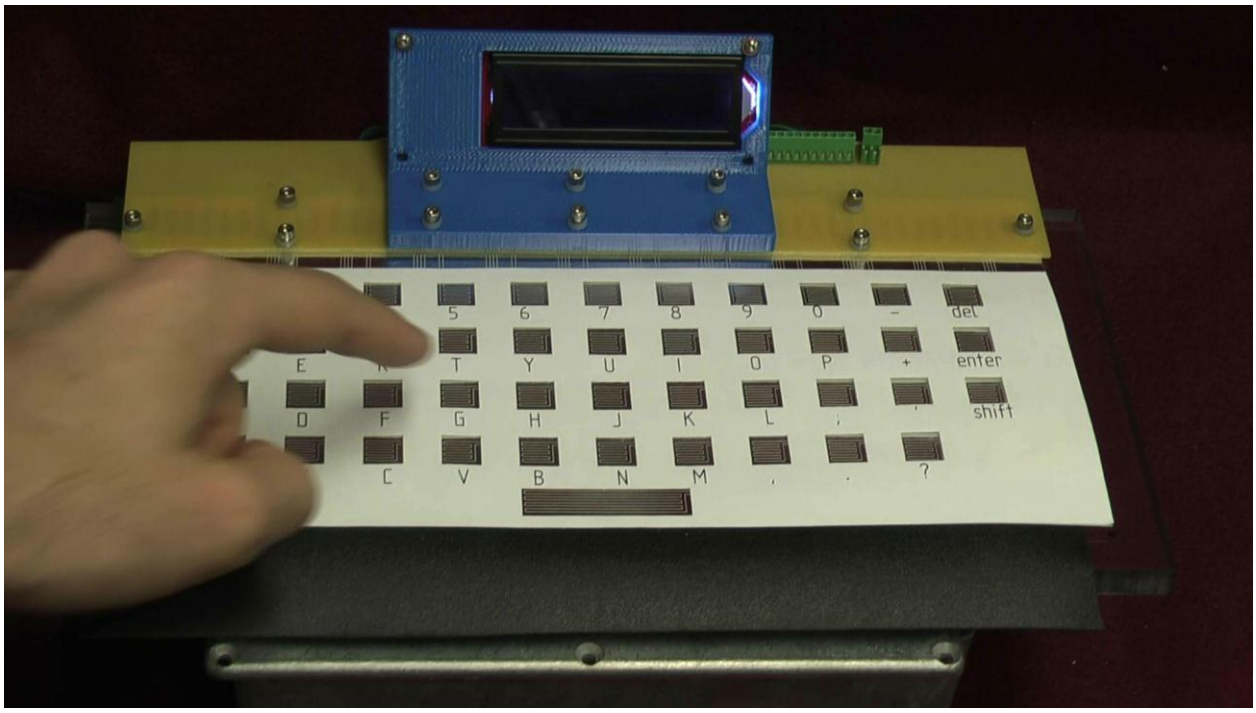


SI Video 2: This video shows four different clips showing a functional 6-button keypad folded around a cube. The first two clips show bare fingers touching keys, while the last two clips show gloved fingers touching the keys. Each key/face addresses an individual LED. Simultaneous touching of multiple keys lights multiple LEDS. Within the same code used for SI Video 1, the "ThresholdFactor" (variable in the code included at the end of the Supporting Information) had a value of 1.13 for all four clips.

SI Video 3

SI Video 3: Video of the alarmed box described in Figure 6. The box is in an unarmed state initially, so that opening it does not set off an alarm. Touching the keypad then arms the box, and the alarm goes off with the opening of the lids. By typing the password into the keypad, the box returns to its unarmed state, and it is possible to open the lids again without setting off the alarm.

SI Video 4:



SI Video 4: Video of the functioning keyboard described in SI Figure 4. The user types in the phrase “GMW Group” and then proceeds to demonstrate the functionality of every key.

Arduino Code for SI Video 1

```
/*Adapted from
http://arduino.cc/it/Tutorial/CapacitanceMeter and
http://www.arduino.cc/en/Tutorial/Smoothing*/
#define Output A1
#define Input1 A0
#define MultiplexA A2
#define MultiplexB A3
#define MultiplexC A4
#define MultiplexD A5
#define LED0 2
#define LED1 3
#define LED2 4
#define LED3 5
#define LED4 6
#define LED5 7
#define LED6 8
#define LED7 9
#define LED8 10
#define LED9 11

unsigned long StartTime; //Time when step output goes
high
unsigned long ElapsedTime0; //Length of time
(microseconds)
//required for RC circuit to reach 2 V.
unsigned long ElapsedTime1;
unsigned long ElapsedTime2;
unsigned long ElapsedTime3;
unsigned long ElapsedTime4;
unsigned long ElapsedTime5;
unsigned long ElapsedTime6;
unsigned long ElapsedTime7;
unsigned long ElapsedTime8;
unsigned long ElapsedTime9;

unsigned long TotalStartTime; //Time when measurments
begin

const int NumReadings=30;
const int SkipCountsForSmoothing=5;
int AverageNow=0;
long int Counter=0;

const float ThresholdFactor=1.12;
int Index0=0;
int Index1=0;
int Index2=0;
int Index3=0;
int Index4=0;
int Index5=0;
int Index6=0;
int Index7=0;
int Index8=0;
int Index9=0;

int Threshold0=95;  //Starting Values for Moving Averaged
//Thresholds in microseconds
int Threshold1=97;
int Threshold2=95;
int Threshold3=102;
int Threshold4=106;
int Threshold5=100;
int Threshold6=100;
int Threshold7=106;
int Threshold8=106;
int Threshold9=100;

int Thresholds0[NumReadings];  //Array of the Moving
Averages
int Thresholds1[NumReadings];  //Array of the Moving
Averages
int Thresholds2[NumReadings];  //Array of the Moving
Averages
int Thresholds3[NumReadings];  //Array of the Moving
Averages
int Thresholds4[NumReadings];  //Array of the Moving
Averages
int Thresholds5[NumReadings];  //Array of the Moving
Averages
int Thresholds6[NumReadings];  //Array of the Moving
Averages
int Thresholds7[NumReadings];  //Array of the Moving
Averages
int Thresholds8[NumReadings];  //Array of the Moving
Averages
int Thresholds9[NumReadings];  //Array of the Moving
Averages

int ThresholdsTotal0=NumReadings*Threshold0;
//Running Total for Moving Averages
int ThresholdsTotal1=NumReadings*Threshold1;
int ThresholdsTotal2=NumReadings*Threshold2;
int ThresholdsTotal3=NumReadings*Threshold3;
int ThresholdsTotal4=NumReadings*Threshold4;
int ThresholdsTotal5=NumReadings*Threshold5;
int ThresholdsTotal6=NumReadings*Threshold6;
int ThresholdsTotal7=NumReadings*Threshold7;
int ThresholdsTotal8=NumReadings*Threshold8;
int ThresholdsTotal9=NumReadings*Threshold9;

void setup()
{
  pinMode(Output, OUTPUT);
  pinMode(Input1, INPUT);
  pinMode(MultiplexA, OUTPUT);
  pinMode(MultiplexB, OUTPUT);
  pinMode(MultiplexC, OUTPUT);
  pinMode(MultiplexD, OUTPUT);
  pinMode(LED0,OUTPUT);
  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);
  pinMode(LED3,OUTPUT);
  pinMode(LED4,OUTPUT);
  pinMode(LED5,OUTPUT);
  pinMode(LED6,OUTPUT);
  pinMode(LED7,OUTPUT);
  pinMode(LED8,OUTPUT);
  pinMode(LED9,OUTPUT);
  digitalWrite(MultiplexA,LOW);
  digitalWrite(MultiplexB,LOW);
  digitalWrite(MultiplexC,LOW);
  digitalWrite(MultiplexD,LOW);
  digitalWrite(LED0,LOW);
```

```
  digitalWrite(LED1,LOW);
  digitalWrite(LED2,LOW);
  digitalWrite(LED3,LOW);
  digitalWrite(LED4,LOW);
  digitalWrite(LED5,LOW);
  digitalWrite(LED6,LOW);
  digitalWrite(LED7,LOW);
  digitalWrite(LED8,LOW);
  digitalWrite(LED9,LOW);
  for (int ThisReading = 0; ThisReading < NumReadings;
ThisReading++){
    Thresholds0[ThisReading] = Threshold0;
    Thresholds1[ThisReading] = Threshold1;
    Thresholds2[ThisReading] = Threshold2;
    Thresholds3[ThisReading] = Threshold3;
    Thresholds4[ThisReading] = Threshold4;
    Thresholds5[ThisReading] = Threshold5;
    Thresholds6[ThisReading] = Threshold6;
    Thresholds7[ThisReading] = Threshold7;
    Thresholds8[ThisReading] = Threshold8;
    Thresholds9[ThisReading] = Threshold9;
  }
  Serial.begin(9600);
  Serial.println("System for Measuring Capacitance ");
  Serial.println("Time to reach 2 V in microseconds");
  TotalStartTime=millis();
}

void loop()
{
  //For Channel 0 on multiplexer (Button 0)
  digitalWrite(MultiplexA,LOW);
  digitalWrite(MultiplexB,LOW);
  digitalWrite(MultiplexC,LOW);
  digitalWrite(MultiplexD,LOW);
  delay(1);
  digitalWrite(Output,HIGH);
  StartTime=micros();
  while(digitalRead(Input1)<1){
  }
  ElapsedTime0=micros()-StartTime;
  digitalWrite(Output,LOW);
  delay(2);
  //For Channel 1 on multiplexer (Button 1)
  digitalWrite(MultiplexA,HIGH);
  digitalWrite(MultiplexB,LOW);
  digitalWrite(MultiplexC,LOW);
  digitalWrite(MultiplexD,LOW);
  delay(1);
  digitalWrite(Output,HIGH);
  StartTime=micros();
  while(digitalRead(Input1)<1){
  }
  ElapsedTime1=micros()-StartTime;
  digitalWrite(Output,LOW);
  delay(2);
  //For Channel 2 on multiplexer (Button 2)
  digitalWrite(MultiplexA,LOW);
  digitalWrite(MultiplexB,HIGH);
  digitalWrite(MultiplexC,LOW);
  digitalWrite(MultiplexD,LOW);
  delay(1);
  digitalWrite(Output,HIGH);
  StartTime=micros();
  while(digitalRead(Input1)<1){
  }
  ElapsedTime2=micros()-StartTime;
  digitalWrite(Output,LOW);
  delay(2);
  //For Channel 3 on multiplexer (Button 3)
  digitalWrite(MultiplexA,HIGH);
  digitalWrite(MultiplexB,HIGH);
  digitalWrite(MultiplexC,LOW);
  digitalWrite(MultiplexD,LOW);
  delay(1);
  digitalWrite(Output,HIGH);
  StartTime=micros();
  while(digitalRead(Input1)<1){
  }
  ElapsedTime3=micros()-StartTime;
  digitalWrite(Output,LOW);
  delay(2);
  //For Channel 4 on multiplexer (Button 4)
  digitalWrite(MultiplexA,LOW);
  digitalWrite(MultiplexB,LOW);
  digitalWrite(MultiplexC,HIGH);
  digitalWrite(MultiplexD,LOW);
  delay(1);
  digitalWrite(Output,HIGH);
  StartTime=micros();
  while(digitalRead(Input1)<1){
  }
  ElapsedTime4=micros()-StartTime;
  digitalWrite(Output,LOW);
  delay(2);
  //For Channel 5 on multiplexer (Button 5)
  digitalWrite(MultiplexA,HIGH);
  digitalWrite(MultiplexB,LOW);
  digitalWrite(MultiplexC,HIGH);
  digitalWrite(MultiplexD,LOW);
  delay(1);
  digitalWrite(Output,HIGH);
  StartTime=micros();
  while(digitalRead(Input1)<1){
  }
  ElapsedTime5=micros()-StartTime;
  digitalWrite(Output,LOW);
  delay(2);
  //For Channel 6 on multiplexer (Button 6)
  digitalWrite(MultiplexA,LOW);
  digitalWrite(MultiplexB,HIGH);
  digitalWrite(MultiplexC,HIGH);
  digitalWrite(MultiplexD,LOW);
  delay(1);
  digitalWrite(Output,HIGH);
  StartTime=micros();
  while(digitalRead(Input1)<1){
  }
  ElapsedTime6=micros()-StartTime;
  digitalWrite(Output,LOW);
  delay(2);
  //For Channel 7 on multiplexer (Button 7)
  digitalWrite(MultiplexA,HIGH);
  digitalWrite(MultiplexB,HIGH);
```

```
digitalWrite(MultiplexC,HIGH);
digitalWrite(MultiplexD,LOW);
delay(1);
digitalWrite(Output,HIGH);
StartTime=micros();
while(digitalRead(Input1)<1){
}
ElapsedTime7=micros()-StartTime;
digitalWrite(Output,LOW);
delay(2);
//For Channel 8 on multiplexer (Button 8)
digitalWrite(MultiplexA,LOW);
digitalWrite(MultiplexB,LOW);
digitalWrite(MultiplexC,LOW);
digitalWrite(MultiplexD,HIGH);
delay(1);
digitalWrite(Output,HIGH);
StartTime=micros();
while(digitalRead(Input1)<1){
}
ElapsedTime8=micros()-StartTime;
digitalWrite(Output,LOW);
delay(2);
//For Channel 9 on multiplexer (Button 9)
digitalWrite(MultiplexA,HIGH);
digitalWrite(MultiplexB,LOW);
digitalWrite(MultiplexC,LOW);
digitalWrite(MultiplexD,HIGH);
delay(1);
digitalWrite(Output,HIGH);
StartTime=micros();
while(digitalRead(Input1)<1){
}
ElapsedTime9=micros()-StartTime;
digitalWrite(Output,LOW);
delay(2);
//Print Results
/*Serial.print(millis()-TotalStartTime);
 Serial.print("   ");
 Serial.print(ElapsedTime0,DEC);
 Serial.print("   ");
 Serial.print(ElapsedTime1,DEC);
 Serial.print("   ");
 Serial.print(ElapsedTime2,DEC);
 Serial.print("   ");
 Serial.print(ElapsedTime3,DEC);
 Serial.print("   ");
 Serial.print(ElapsedTime4,DEC);
 Serial.print("   ");
 Serial.print(ElapsedTime5,DEC);
 Serial.print("   ");
 Serial.print(ElapsedTime6,DEC);
 Serial.print("   ");
 Serial.print(ElapsedTime7,DEC);
 Serial.print("   ");
 Serial.print(ElapsedTime8,DEC);
 Serial.print("   ");
 Serial.print(ElapsedTime9,DEC);
 Serial.print("   ");*/
//Turn on appropriate LED(s)
AverageNow=Counter%SkipCountsForSmoothing;
if(ElapsedTime0>Threshold0*ThresholdFactor){
  digitalWrite(LED0,HIGH);
  //Serial.print("   ");
  //Serial.print("1");
 }
 else{
  digitalWrite(LED0,LOW);
  //Serial.print("   ");
  //Serial.print("0");
  if(AverageNow==0){
   ThresholdsTotal0=ThresholdsTotal0-
Thresholds0[Index0];
   Thresholds0[Index0]=ElapsedTime0;
   ThresholdsTotal0=ThresholdsTotal0+ElapsedTime0;
   Threshold0=ThresholdsTotal0/NumReadings;
   Index0=Index0+1;
   if(Index0>=NumReadings){
    Index0=0;
   }
  }
 }
 if(ElapsedTime1>Threshold1*ThresholdFactor){
  digitalWrite(LED1,HIGH);
  //Serial.print("   ");
  //Serial.print("1");
 }
 else{
  digitalWrite(LED1,LOW);
  //Serial.print("   ");
  //Serial.print("0");
  if(AverageNow==0){
   ThresholdsTotal1=ThresholdsTotal1-
Thresholds1[Index1];
   Thresholds1[Index1]=ElapsedTime1;
   ThresholdsTotal1=ThresholdsTotal1+ElapsedTime1;
   Threshold1=ThresholdsTotal1/NumReadings;
   Index1=Index1+1;
   if(Index1>=NumReadings){
    Index1=0;
   }
  }
 }
 if(ElapsedTime2>Threshold2*ThresholdFactor){
  digitalWrite(LED2,HIGH);
  //Serial.print("   ");
  //Serial.print("1");
 }
 else{
  digitalWrite(LED2,LOW);
  //Serial.print("   ");
  //Serial.print("0");
  if(AverageNow==0){
   ThresholdsTotal2=ThresholdsTotal2-
Thresholds2[Index2];
   Thresholds2[Index2]=ElapsedTime2;
   ThresholdsTotal2=ThresholdsTotal2+ElapsedTime2;
   Threshold2=ThresholdsTotal2/NumReadings;
   Index2=Index2+1;
   if(Index2>=NumReadings){
    Index2=0;
   }
  }
 }
```

```
  if(ElapsedTime3>Threshold3*ThresholdFactor){
    digitalWrite(LED3,HIGH);
    //Serial.print("   ");
    //Serial.print("1");
  }
  else{
    digitalWrite(LED3,LOW);
    //Serial.print("   ");
    //Serial.print("0");
       if(AverageNow==0){
      ThresholdsTotal3=ThresholdsTotal3-
Thresholds3[Index3];
      Thresholds3[Index3]=ElapsedTime3;
      ThresholdsTotal3=ThresholdsTotal3+ElapsedTime3;
      Threshold3=ThresholdsTotal3/NumReadings;
      Index3=Index3+1;
      if(Index3>=NumReadings){
        Index3=0;
      }
    }
  }
  if(ElapsedTime4>Threshold4*ThresholdFactor){
    digitalWrite(LED4,HIGH);
    //Serial.print("   ");
    //Serial.print("1");
  }
  else{
    digitalWrite(LED4,LOW);
    //Serial.print("   ");
    //Serial.print("0");
    if(AverageNow==0){
      ThresholdsTotal4=ThresholdsTotal4-
Thresholds4[Index4];
      Thresholds4[Index4]=ElapsedTime4;
      ThresholdsTotal4=ThresholdsTotal4+ElapsedTime4;
      Threshold4=ThresholdsTotal4/NumReadings;
      Index4=Index4+1;
      if(Index4>=NumReadings){
        Index4=0;
      }
    }
  }
  if(ElapsedTime5>Threshold5*ThresholdFactor){
    digitalWrite(LED5,HIGH);
    //Serial.print("   ");
    //Serial.print("1");
  }
  else{
    digitalWrite(LED5,LOW);
    //Serial.print("   ");
    //Serial.print("0");
    if(AverageNow==0){
      ThresholdsTotal5=ThresholdsTotal5-
Thresholds5[Index5];
      Thresholds5[Index5]=ElapsedTime5;
      ThresholdsTotal5=ThresholdsTotal5+ElapsedTime5;
      Threshold5=ThresholdsTotal5/NumReadings;
      Index5=Index5+1;
      if(Index5>=NumReadings){
        Index5=0;
      }
    }
  }
  if(ElapsedTime6>Threshold6*ThresholdFactor){
    digitalWrite(LED6,HIGH);
    //Serial.print("   ");
    //Serial.print("1");
  }
  else{
    digitalWrite(LED6,LOW);
    //Serial.print("   ");
    //Serial.print("0");
    if(AverageNow==0){
      ThresholdsTotal6=ThresholdsTotal6-
Thresholds6[Index6];
      Thresholds6[Index6]=ElapsedTime6;
      ThresholdsTotal6=ThresholdsTotal6+ElapsedTime6;
      Threshold6=ThresholdsTotal6/NumReadings;
      Index6=Index6+1;
      if(Index6>=NumReadings){
        Index6=0;
      }
    }
  }
  if(ElapsedTime7>Threshold7*ThresholdFactor){
    digitalWrite(LED7,HIGH);
    //Serial.print("   ");
    //Serial.print("1");
  }
  else{
    digitalWrite(LED7,LOW);
    //Serial.print("   ");
    //Serial.print("0");
    if(AverageNow==0){
      ThresholdsTotal7=ThresholdsTotal7-
Thresholds7[Index7];
      Thresholds7[Index7]=ElapsedTime7;
      ThresholdsTotal7=ThresholdsTotal7+ElapsedTime7;
      Threshold7=ThresholdsTotal7/NumReadings;
      Index7=Index7+1;
      if(Index7>=NumReadings){
        Index7=0;
      }
    }
  }
  if(ElapsedTime8>Threshold8*ThresholdFactor){
    digitalWrite(LED8,HIGH);
    //Serial.print("   ");
    //Serial.print("1");
  }
  else{
    digitalWrite(LED8,LOW);
    //Serial.print("   ");
    //Serial.print("0");
    if(AverageNow==0){
      ThresholdsTotal8=ThresholdsTotal8-
Thresholds8[Index8];
      Thresholds8[Index8]=ElapsedTime8;
      ThresholdsTotal8=ThresholdsTotal8+ElapsedTime8;
      Threshold8=ThresholdsTotal8/NumReadings;
      Index8=Index8+1;
      if(Index8>=NumReadings){
        Index8=0;
      }
```

```
   }
  }
  if(ElapsedTime9>Threshold9*ThresholdFactor){
    digitalWrite(LED9,HIGH);
    //Serial.print("    ");
    //Serial.println("1");
  }
  else{
    digitalWrite(LED9,LOW);
    //Serial.print("    ");
    //Serial.println("0");
    if(AverageNow==0){
      ThresholdsTotal9=ThresholdsTotal9-
Thresholds9[Index9];
      Thresholds9[Index9]=ElapsedTime9;
      ThresholdsTotal9=ThresholdsTotal9+ElapsedTime9;
      Threshold9=ThresholdsTotal9/NumReadings;
      Index9=Index9+1;
      if(Index9>=NumReadings){
        Index9=0;
      }
    }
  }
  Counter=Counter+1;
```