Western Michigan University

Map Proper Footwork with the use of Piezoelectric Sensors

Daniel Kolosa
Justin Rittenhouse
Smart Material – ME 6500
Dr. Ari-Gur
December 7, 2016

Abstract

The key to success in sports is often proper technique. Technique is often hard to master, and practicing with incorrect technique can potentially do more harm than good. An important technique that is often overlooked in football is not to be caught flat footed in open space. Once flat footed, it is too hard to change direction at a rate as fast as the imposing player. This can often lead to being put on the wrong end of a highlight reel. This is one of numerous examples where self-analyzing shoes could come into play. Piezoelectric sensors can be placed into shoes to read the pressure distribution being applied to different parts of the foot. The signals generated from the piezoelectric sensors will be conditioned and converted to a digital signal via microcontroller. The microcontroller will transmit the data wirelessly to either a computer or mobile device via a Bluetooth module. The information can be stored and analyzed or broadcast in real-time allowing the user to correct improper footwork.

Table of Contents

Table of Figures

Table of Tables

Introduction

The modern athlete trains for countless hours to develop their skills. This is because the level of competition seems to be ever increasing. Records are constantly being broken. This is partly due to the advancement in sports technology. Modern technology has allowed today's athlete to hone their skills to new levels. This is because technology today can help improve

proper technique. Technique is what can separate the legends from the greats. Little things such as foot or hand placement can save precious milliseconds.

There is little out there in the world of foot mapping. Foot placement and pressure distribution of the foot are key aspects to many sports such as: golf, baseball and football just to name a few. In golf, the golfer's weight should be put more towards the front of the foot and off the heels. This allows for a proper swing. This is also true for baseball. In football if the defender is caught flat footed in open space, odds are, he is going to get juked. The defender should be on the balls of their feet, and their feet should never be static. An additional sport where foot mapping would be beneficial is running. Long distance running is probably one of the toughest sports on the feet. It is critical to have proper pressure distribution. Those are just some of the many examples to make proper foot mapping technology more reachable for the modern athlete.

The use of microcontrollers could be used to solve this problem. A microcontroller is an integrated circuit with memory and a processor core that has the ability to be programmed to take inputs or sent outputs through either digital or analog signals. The microcontroller is the heart of the Arduino board. All prototyping was done on an Arduino Uno board. The Arduino is the base model to what will be created. Eventually a printed circuit board (PCB) could be made. This is due to the Arduino Uno's size and its lack of analog inputs. It is clearly too large to fit into a shoe. In addition to the size disadvantage, the Arduino Uno only has 6 analog inputs. It is possible to use digital inputs as analog inputs. Though, this requires more parts. More parts requires more space, time and money. All three of which are limiting factors in today's competitive marketplace. A PCB will not only give us more flexibility with the size but with features as well. The Arduino is a base design and is meant for many purposes. This flexibility is what limits its appeal for mass production for anything other than prototyping. A PCB can be designed with only one goal in mind. In this case, it will be design to read at least 10 analog inputs from the piezoelectric sensors and send data via Bluetooth. The Bluetooth will be communicating with an application available on Android products.

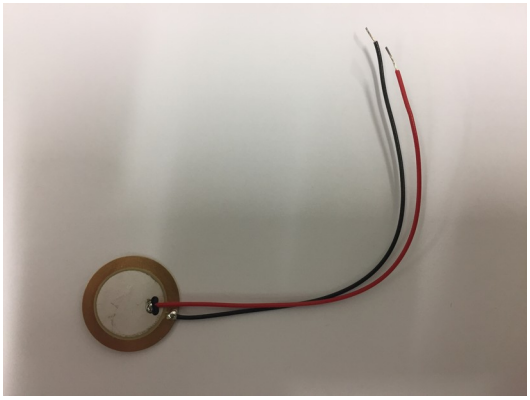
Android is the most widely used phone operating system in the world. It runs on millions of devices from phones, tvs, watches, and even cars. Android is currently being developed by Google, and to create widespread adoption, they have made it very accessible to develop apps.

Applications on Android are developed in Java on top of a java virtual machine. Android development can be done on Linux, macOS, or Windows and it is free to publish apps to the Google play store. Also android has few restrictions on third party hardware so it is easier to interact with custom build hardware.

iOS is the second biggest phone operating system developed by Apple. A larger amount of app revenue is generated by iOS app than Android making it a very tempting development platform. But iOS development has its drawbacks including all development must be done on Apple computers, a fee to begin developing iOs apps, developers must use either the Objective-C or Swift programming languages, and restrictions on third party hardware support like Bluetooth modules. Considering all the pro and cons we have decided it would be easier to develop an interface for our Arduino using Android.

Piezoelectric

Piezo-electrics is a material that creates an electric charge due to a stress being applied. In this case, the stress will be applied through force. The force applied by stepping on them is more than enough to create a readable charge. The charge is relatively constant. This means, when the same force is applied, it creates roughly the same charge. This allows the ability for the piezoelectric sensors to be calibrated. It should be noted that the charge does dissipates over time until the force is removed and replied. This dissipation has little effects on this project due to the fact athletic events are normally dynamic movements. The sensors can be seen in Figure ## below. The specifications of the piezoelectric sensors chosen can be seen in Table ## below [1].



Piezoelectric Sensors Specifications	
Resonant Frequency	6.8 KHz
Resonant Resistance	300 ohm max
Capacitance	15,000 pF +/- 30%
Weight	1.6 oz
Manufacturer	Audiowell
Diameter	20 mm
Leads Length	100 mm

Arduino Program

The Arduino's program can be seen in Appendix A. The program is designed to read the input from the piezoelectric sensors and send the information out via a Bluetooth module to an Android phone. The sensors are not read together. They are read independently, and information is sent out one at a time. This is possible due to the rapid rate at which the force is read. The rate to read and send pressure information is only 400 milliseconds. The application seems instantaneous at this rate.

The Arduino can do a great deal more than what is required here. The reason for keeping the Arduino program as basic as possible is the difference in power between the Arduino and an average Android phone. The main specifications between the Arduino and Samsung Galaxy S6 are shown in Table 1 below [2, 5]. It is clear the phone has significantly more power and the ability to do tasks at a quicker rate. This is the reason the phone will be doing the "heavy lifting" as far as programming is concerned.

Table 1: Specs of Arduino versus Galaxy S6.

Specifications	Arduino	Samsung Galaxy S6
Flash Memory	32 KB	3 GB
Memory	2 KB	32+ GB
Clock Speed	16 MHz	4x2.1 GHz – Cortex-A57 & 4x1.5 GHz Cortex-A53

Android Program

When creating an android app, various different files are created. These files contain configuration data, build information, application logic, user interface layout, and icons, photos, and outer resource files. For the purposes of this project the main logic of the app will be included. The main logic located in the MainActivity.java. This file first starts by calling the OnCreate method. OnCreate displays the user interface and loads the widgets. Within the onCreate method, the Bluetooth connection is established in the bluetoothstate method. In bluetoothstate, Bluetooth support is checked and permission is requested to the user to activate Bluetooth if it is not already turned on. For the two devices to successfully connect to each other, a Bluetooth socket must be opened and a client-server relationship must be established. The Bluetooth socket creates a secure connection between two devices , the phone and the arduino Bluetooth module. When the connection is established, a thread must be created to transfer data between the two devices.,

Back in the OnCreate activity, the getuserweight is called to display a message for the user to enter in their weight. The user then enters and numeric value that will be used to calibrate the sensors for the user's weight. Then to transfer data between the android and arduino a thread must be created to handle the data. This thread of data is then passed to a data handler.

A class is created called ConnectedThread in line 240. The connected thread creates an input stream that will only allow the arduino device to send data. The connected thread class can extend to write data to the arduino but at this time it is not used due to software security practices. The connected thread reads data from the arduino as byte data, continuously until the connection is closed.

In the OnCreate method the handler checked if data is received by the thread. If there is data then the bytes are converted into a string. This string is the human readable sensor data and can be parsed and interpreted. The String that is obtained contained a # signifying the beginning of a set of data and a ~ signifying the end of a data stream. One data stream represents the outputs of the sensor data from all of the piezoelectrics sensors that are connected to the arduino. When the end of a data stream is determined, The number are then split up based on a delimiter value, in this case the + (plus) sign. Each individual set of data is interpreted for its corresponding text box. The order of the sensors are read from left to right, top to bottom. The sensor data is read as an integer and is then the value goes into four color categories based on highest value: blue, green, red, and magenta, Blue represents the lowest value or applying little to no pressure on the piezoelectric and magenta represents the largest amount of pressure that can be applied. The color values are then set as the background color for each respective textbox. This process will continue until the application is closed or the connection is interrupted.

Calibration Process

The calibration process is one of the key parts to this project. This is due to the flexibility wanted with different users. The weight of athletes various so dramatically, the program must be able to account for it and accurately read the force applied to the sensor. The first round of calibration was done through Matlab. An Arduino program was written to read a force being applied to a piezoelectric sensor and send that information to a Matlab program. The Matlab program was written to read that incoming information. After the programming was complete, five different forces were applied to the sensor. The Matlab program would then read all five forces, one at a time. Once all the forces were applied and read, the polyfit function within the Matlab program would create a curve fitting function. This function would then be able to approximate the weight being applied. Table 2 below shows the percent error between the created function and the actual weight applied to the sensor. This step was done for two reasons. The first to ensure that it is possible to create a curve fitting function; that could accurately approximate the force being applied to the sensor. The second was for testing. The calibration program will actually be written within the Android application created. This Matlab program allowed further progress to be gained on the project while the application was still being created. The Arduino and Matlab programs can both be seen in Appendix C below.

Table 2: Weight applied versus approximated weight.

Weight Applied (Mass) [N (g)]	Approximated Weight [N]	Percent Error
1.47 (150)	1.40	4.8%
2.45 (250)	2.39	2.4%
3.43 (350)	3.33	2.9%
4.41 (450)	4.25	3.6%
5.39 (550)	5.27	2.2%

Future Improvements

A future improvement that would greatly increase the marketability of this product is making it Apple compatible. Apple holds about 40% of the smartphone market [3] and has about 13% of the second quarter market shares of 2016 [4]. This is the largest share for a single company. Apple also holds the second highest percent of the tablet market for the second quarter of 2016 at 26% [6]. A future Apple application would potentially double the market present.

Another future improvement would be to design a PCB board. The PCB could be designed in KiCAD. The board could be designed specifically for this application. The board would include a microcontroller, multiplexer, 10 analog inputs and a Bluetooth module. The ideal microcontroller would be Atmel Attiny85 and the specifications can be seen in Table 3 below. The microcontroller is the brains of the operation. It is the item that read the program and tell the Bluetooth what to send. The multiplexer is there to increase the analog inputs. The ideal multiplexer would be a Switch ICs 8-Channel from Texas Instruments. This multiplexer can receive 8 inputs and reduce the signal down to one for the microcontroller to read. This allows for a total of 10+ analog inputs to use between the microcontroller and the multiplexer. The Bluetooth module that would be used is the BLE 4.1 Xmit by TDK. This Bluetooth module was picked for its cost of only $9.00 and more importantly, its size. The Bluetooth is only 3.5x3.5x1 mms.

Table 3: Specifications for the Atmel Attiny85.

Microcontroller	Specifications
Max Clock Frequency	20 MHz
Program Memory	8 kB
Supply Voltage	2.7-5.5 V
Weight	.3 oz
Dimensions	3.1 x 3.1 x .5 inches

Conclusion

The idea to have piezoelectric sensors be a useful tool for foot mapping was a success. There were three main parts of this project. The first being able to read piezoelectric sensors with an Arduino. This part was done successfully through the use of the analog pins on the Arduino. The next task was to transmit this data wirelessly. This was done effectively through the use of a Bluetooth module. The third and probably hardest part of this project was building a phone application that could read the incoming data from the Bluetooth and display it in a pleasant manner to the user. This application was successfully made through Android Studio. In the end, this project proved that small scaled foot mapping could be implemented in the future. This technology could potentially improve athletic performance through the improvement of technique. Resetting records along the way.

References

1. "15 Pieces - 20mm Piezo Disc Elements with 4" Leads - Acoustic Pickup - Cigar Box Guitar CBG - Touch Sensor." *Amazon*. N.p., n.d. Web. 05 Dec. 2016.
2. "Arduino - ArduinoBoardUno." *Arduino - ArduinoBoardUno*. N.p., n.d. Web. 01 Dec. 2016.
3. Elmer-DeWitt, Philip. "About Apple's 40% Share of the U.S. Smartphone Market." *Fortune*. N.p., 10 Feb. 2016. Web. 07 Dec. 2016.
4. "Gartner Says Five of Top 10 Worldwide Mobile Phone Vendors Increased Sales in Second Quarter of 2016." *Gartner Says Five of Top 10 Worldwide Mobile Phone Vendors Increased Sales in Second Quarter of 2016*. N.p., 19 Aug. 2016. Web. 05 Dec. 2016.
5. "Samsung Galaxy S6." *Samsung Galaxy S6 - Full Phone Specifications*. N.p., n.d. Web. 03 Dec. 2016.
6. Spence, Ewan. "Apple's Continued Domination Of A Shrinking Tablet Market." *Forbes*. Forbes Magazine, 2 Aug. 2016. Web. 05 Dec. 2016.

Appendices

Appendix A

```
const int ANAPINS[] = {0,1,2};
int incoming;
int npins = 3; // number of pins plus 1
int value;

void setup() {
    for (int pin = 0; pin < npins; pin++) {
        pinMode(ANAPINS[pin], OUTPUT);
    }
    Serial.begin(9600);
}

void loop() {
  if(Serial.available() > 0);{
    for (int analogpin = 0; analogpin < npins; analogpin++)
    {
        analogRead(analogpin);
        delay(100);
        value = analogRead(analogpin);
        Serial.print(value);
    }
     Serial.print("\n");
    delay(300);
  }
}
```

Figure 1: Arduino Code

Appendix B

Appendix C

```
int mode = -1;
int value;

void setup() {
  Serial.begin(9600);
  Serial.println('a');
  char a = 'b';
  while (a !='a')
  {
    a=Serial.read();
  }
}

void loop() {
  if (Serial.available()> 0)
  {
    mode=Serial.read();
    switch (mode)
    {
      case 'F':
      value= analogRead(A0);
      Serial.println(value);
      break;
          }
  }

}
```

Figure 2: Arduino code for calibration.

```
function [force] = readFSR(out)
    fprintf(out.s,'F');
    force = fscanf(out.s,'%d');
end
```

Figure 3: The function file that reads the voltages.

```
function [s,flag] = setupSerial(comPort)
    flag = 1;
    s = serial(comPort);
    set(s,'DataBits', 8);
    set(s,'StopBits', 1);
    set(s,'BaudRate', 9600);
    set(s,'Parity', 'none');
    fopen(s);
    a='b';
while (a ~='a')
    a=fread(s,1,'uchar')
end
if (a=='a')
    disp('serial read');
end
    fprintf(s,'%c','a');
    mbox = msgbox('Serial Communication setup.'); uiwait(mbox);
    fscanf(s,'%u');
end
```

Figure 4: The Matlab function file that sets up the serial input.

```
clear all
if ~isempty(instrfind)
    fclose(instrfind);
    delete(instrfind);
end
    close all
disp('Serial Port Closed')
```

Figure 5: The Matlab function file that closes the serial port.

```
clc
clear all
close all

%% Specifies the COM port that the arduino board is connected to
comPort = '/dev/tty.usbmodem1a12131'; % this can be found out using the device manager
% connect octave to the accelerometer
if (~exist('serialFlag','var'))
    [fsr.s,serialFlag] = setupSerial(comPort);
end

%% initalize the figure and buttons that we will plot in if it does not eist
if(~exist('h','var') || ~ishandle(h))
    h = figure(1);
end

if(~exist('text1','var'))
    text1 = uicontrol('Style','text','String','X: 0 degrees',...
                      'pos',[450 100 100 25],'parent', h)
end

if(~exist('text2','var'))
    text2 = uicontrol('Style','text','String','Y: 0 degrees',...
                      'pos',[450 75 100 25],'parent', h)
end

if(~exist('button','var'))
    text2 = uicontrol('Style','togglebutton','String','Stop & Close Serial Port',...
                      'pos',[0 0 200 25],'parent', h)
end

%% Calibrate the Sensor
weights = [0 20 50 70 100 120 150 170 200 220 250];
m1=zeros(length(weights),1)';

% Read the values for each weight and assign it
```

```
for i=2:length(weights)
    mbox = msgbox(['Place ' num2str(weights(i)) ' grams on the FSR']); uiwait(mbox);
    m1(i) = readFSR(fsr)

    while (m1(i) < m1(i-1)) || m1(i) ==0;
    m1(i) = readFSR(fsr)
    end
end
m=m1

P1 = ployfit(m,weights,2); % Setup the curve fitting

% Setup the graph
myaxes = axes('xlim',[-20 20], 'ylim',[-20 20], 'zlim',[0 250]);
view(3);
grid on;
axis equal;
hold on;

% Draw the 3D sphere
[xsphere ysphere zsphere] = sphere();
h(1) = surface(xsphere,ysphere,zsphere);

%Add the 3D sphere to the figure
combinedobject = hgtransform('parent',myaxes);
set(h,'parent',combinedobject)
drawnow

%% Acquire data and plot

while (get(button, 'Value') ==0)
    [voltage] = readFSR(fsr) % Read data from the FSR
    mass = polyval(p1, voltage) % in grams

    if(mass > 0)
        force=mass*9.81; % [N]
```

```
        % Update the readouts on the figure
        set(text1,'String',['Mass : ' num2str(round(mass)) ' grams'])
        set(text2,'String',['Force : ' num2str(round(mass)) ' N'])

        % Move sphere along z axis
        translation = makehgtform('translate',[0 0 mass]);
        set(combinedobject,'matrix',translation);

        % Scale sphere t 1/10th mass
        scaling = makehgtform('scale', mass/10);
        set(combinedobject,'matrix', scaling);

        % Set the translation and scaling to the sphere
        set(combinedobject,'matrix',translation*scaling);

    end
    %pause
    pause(0.1);
end

% when graph closed, close serial
closeSerial;
```

Figure 6: The Matlab file that does the weight estimates.