Senior Project: Saddle Force Mapping
Cal Poly Physics

Molly Totten

6/10/2015

Abstract

In horseback riding, the size and shape of a saddle can drastically effect the comfort, attitude and performance of the horse as well as the effectiveness of the rider. In an effort to create an accurate means of fitting the saddle to the horse, I have created a pressure sensing device that maps the force differences along critical areas of the horse's back during motion. Using an Arduino microprocessor and appropriate circuits, data was collected about the pressure levels between the saddle and the horse's back and a graphic representation of the findings was created.

1 Introduction

Classical dressage, a type of English horse riding, is defined as the art of riding and training a horse in a manner that develops obedience, flexibility and balance. This training process can take up to 20 years for a horse to master the correct balance and knowledge of movements. In dressage, having a properly fitting saddle is extremely important in allowing full range of motion for and preventing back pain in horses as well as keeping the rider in a symmetrical position. The effect of the rider on the saddle fit is critical to understanding the dynamic nature of saddle fit.

Fitting a saddle to a horse usually involves a series of inaccurate visual tests as well as feeling underneath the saddle to determin pressure points. This method gives us very little information about the correctness of fit while the horse is in motion. Typically, the only information we can gather about the saddle fit while in motion is from sweat patterns left on the horse's back after a strenuous workout. Dry spots in the saddle area indicate pinching while ruffled hairs indicate a loose fit. After following the rules of thumb, I decided to make a tool that can give more accurate dynamical information about the pressure in key areas of the spine and shoulder.

The system used to made to take and interpret information about the pressure between the saddle and the horse's back while in motion included; an Arduino microprocessor, sensors with the appropriate circuitry and a Matlab code to make plots of the data collected. The circuit and Arduino microcontroller sits in a small box that is attached to the back of the saddle. The sensors take multiple pressure readings during a riding session in order to show how well the saddle fits during various types of motion. By getting information while the horse is in motion, we can make a much more comprehensive diagnosis of the saddle fit.

In order to gather information about the changes in force beneath the saddle, sensors were placed in strategic locations on the horse's back and a circuit was created to read the output of the sensors. These sensors collect data while the horse and rider are free to move about. The data is then written to an SD card and read into a Matlab program that compiles and graphs the data. For the circuit configuration, there are two viable options for the sensor configuration. One is to use a variable resistance force sensor and the other is to use a variable voltage Piezo disk. Both can be configured to read pressure data but their setup, output and function vary greatly.

In order to read the forces, a variable resistance force sensor can be used with each sensor placed in a voltage divider. In a voltage divider, a power source is connected to two resistors in series. In this case, the first resistor has a fixed value while the second resistor is a variable resistance force sensitive resistor. Changes to the value of the fixed resistor will increase or decrease the sensitivity of the second resistor. This follows Equation 1, Ohm's Law, and allows the voltage supplied to be split between the resistors. Here, V is the voltage, I is the current and R is the resistance. Rearranging Ohm's Law and solving for current, we get equation 2. Since there are two resistors in series, R can be redifined as $R = r_1 + r_2$. From this, we can determine the current, I, as seen in equation 3. In this system, the force sensor can be thought of as R_2. In order to determine the corresponding voltage, V_2, across the force sensor, we can substitute the expression for current and get the result in equation 4 (Eggleston, 12). This setup works nicely for this application but these sensors typically have a physical limitation of how much weight they can sense. Since the weight of a person is much larger than the typical 10 kg weight limit for standard force sensors, another approach was required.

After discovering the weight limitations of the standard force sensitive resistors, using a Piezo disk instead was a much better option. Using inspiration from the variable resistance force sensors, I set up the Piezo disks in a somewhat similar manner. Since the Piezo disk modulates it's voltage in response to a force differential, the voltage can be read directly from the disk without the need for a voltage divider. There is a fundamental difference in the variable force resistor and the Piezo disk in that the force sensor changes resistance in reaction to force while the Pieazo disk changes it's voltage. This eliminated the need for any signale amplification or complicated circuits.

Piezo disks use a phenomenon called the Piezoelectric Effect, which is the ability of certain materials to generate an electrostatic charge in response to mechanical stress. This creates a transducer effect between the electrical oscillations from the charge of the crystals and the mechanical oscillations of the force or strain itself. Transducers allow the pressure information to be converted into an electrical signal and, in turn, allows the force data to be collected and controled efficiently. This behavior is unique in that the charge crystals do not hold their charge once the stress or force is released. The piezoelectric effect can be seen in common materials such as quartz, sucrose (table sugar) and certain ceramic material. In the Piezo disk, crystals are placed on one side of the disk and become charged when they are compressed or distorted. This charge can then be read as a voltage. (Nave)

Due to the limited number of analog input pins on the Arduino Uno, a 4051 integrated circuit was used in order to reduce the amount of input pins used on the Arduino and take data from multiple sensors and combine it into one string of information. The 4051 is an 8-channel analog multiplexer and demultiplexer which means that it can turn eight analog inputs into one digital output when configured for multiplexing or vice versa for demultiplexing. Though there are other multiplexers available, the 4051 is most compatible with the Arduino's and accepts a high number of analog inputs. With other multiplexers, such as the 4066 which was initially tried, their voltage demand is higher than the Arduino's 5V supply voltage and an NPN transistor must be used to supply enough potential. After the Piezo disks and the 4051 multiplexer were selected and confirmed to be the most effective for this application, I was able to successfully integrate the circuit to record and then interperet the data gathered from the horse and rider.

$$V = IR \tag{1}$$

$$I = \frac{V}{R} \tag{2}$$

$$I = \frac{V}{r_1 + r_2} \tag{3}$$

$$V_2 = R_1 \frac{V}{r_1 + r_2} \tag{4}$$

2 Apparatus

The saddle sensor system consists of four parts; the saddle pad with sensors, a circuit that is connected to an Arduino, an Arduino with a code to record and organize the data and a Matlab program to map and display the data in an easily understood manner. As previously mentioned, the sadde pad has eight sensors placed in strategic locations with sensors placed along the back and down the scapula on both sides. The pad sits between the saddle and the horse's back and is used to keep the horse comfortable while sweating and to keep the saddle clean. The sensors are sewn into a pad so that the system can be used with different saddle, horse and rider combinations. The sensors consist of a Piezo disk that changes voltage with increased or decreased force. Each Piezo disk is placed in a strategic location along the horse's back in order to line up with the latisimus dorsi, the long back muscles that run alone either side of the spine and carry the weight of the saddle and rider. In order to keep the circuit protected from dust,

the breadboard and Arduino are placed into a small cloth bag. The ends of the bag are then tucked underneath the edge of the saddle to ensure the bag stays in place but the circuit stays accessible. A picture of the saddle pad with sensor placement indications can be seen in figure 1. Pictures of the saddle pad sensor on the horse can be seen in figures 2 and 3. The bag remained untucked in the rear view photo to show the wires extending from the back of the saddle.

2.1 Circuit Configuration

The Piezo disks are arranged on a breadboard and are connected to their appropriate analog pins on the 4051 (y0 through y7). Since the disks change their voltage when compressed or distorted, they can be connected directly to the 4051. The 4051 then interfaces with the Arduino via the Arduino code. A photo of the sensor can be seen in figure 4 and a Fritzing diagram of the circuit can be seen in figure 5. In the diagram, the green wires connect from pins 9, 10 and 11 on the 4051 to the three digital output pins, S0, S1 and S2. These pins are responsible for cycling through the analog inputs and creating one output for each of the 8 inputs.

The 4051 uses pins y0 through y7 for each of the analog inputs from the Piezo disks along with a common analog, z, outpus that connects to the Arduino. The 4051 also needs three pins connected to ground and one pin connected to the 5V pin, both on the Arduino. Finally, the 4051 uses three select pins (S0, S1 and S2), each of which connect to one digital out pin on the Arduino. S0 is assigned the value 1 while S1 is 2 and S2 is 4. When one of the select pins is set to high by the Arduino, the value of that select pin is transmitted to the 4051. The Arduino code is then designed to turn each of the select pins to high in various combinations in order to count through the analog inputs. For example, is S0, S1 and S2 are all set to high, the 4051 recieves (S0+S1+S2) or (1+2+4) = 7 and will then read the value from the 7th analog input value, y6. A table for the counting logic can be seen in figure 6, where INH is the inhibitor or common analog and A, B and C are S0, S1, S2, respectively. The pin configuration for the 4051 can be seen in figure 7, where E, Vee and gnd all correspond to the three ground pins and Vcc connects to the 5V pin.

2.2 Arduino Sketch

From the Arduino, the Arduino sketch or code takes in the data from the 4051. There are two parts to the code; one part to initialize and write the data to the SD card, one to communicate with the 4051. In order to make data collection physically possible and eliminate the need for computer interface during data collection, the data must be written to an SD card so that the sensor system can be used while riding. Within the Arduino code, there are two parts; one for implimenting the counting logic and controlling the 4051 and one for reading and writing the data to the SD card. An SD card is a serial peripheral interface (SPI) device. These devices can be SD cards, microprocessors or other devices and are controlled by a master microprocessor which, in this case, is an Arduino Uno. The SPI type and Arduino type will then determine what to include into the Arduino sketch.

In order to read from the SD card, SPI.h and SD.h must be included in the initialization section above the setup. Next, for the 4051 initialization, the select pins must be initialized. In the code, these are defined as r0, r1 and r2 for S0, S1 and S2, respectively, and have an initial value of zero. Next, the count variable for the 4051 must be initialized and also set to an initial value of 0. Next, a the common analog pin must be set. In this case, the pin is A0. A variable, called PiezoValue, also must be initialized in order to store the sensor value. After initializing the SD card and variables, the setup portion of the sketch follows in which the pins are activated and the SD card is opened.

In the setup section of the Arduino code, digital pins 2, 3 and 4 are used for the select pins S0, S1 and S2. These pins must be set to be outputs using the pinMode command. Digital pin 10 must also be set to output using pinMode since this pin is the default hardware SS pin for the Arduino Uno. Certain Arduinos have various SS pins but 10 is most common. The serial communication for the SD card must also be opened in the setup using the Serial.begin command. Here, the Serial.begin argument is 9600 which means that the communication rate is 9600 bits per second or 9600 baud. The baud rate must match between the Arduino serial monitor rate and the rate at which the computer recieves the data. After setting the digital pins to the proper mode and starting the serial communication, the loop must be executed.

In the loop, the counting logic for the 4051 is implemented by using a for loop. The count variable will cycle through values 0 through 7 which corresponds to each Piezo disk. Next, r0, r1 and r2 are read in order using the bitRead command. The values of the select pins S0, S1 and S2 are then assigned values of r0, r1 and r2

using the digitalWrite command. After that, the 4051 reads the analog values of each Piezo disk using the analogRead command and assigns these readings to the Piezovalue variable. The Piezovalue readings are then converted into string values and placed into an array so that they can be sent to the SD card. Each count value (0 through 7) will correspond to one Piezovalue entry and will be placed into the 1x8 array.

After the analog sensor values are placed into the array, the SD card will initialize and open a file called "datalog.txt" using the File and SD.open commands. If the file is available, the SD card will write each string onto the SD card and close the file. The string of data is then printed to the serial port so that it can be monitored while connected to the computer. The serial monitor will then read each individual Piezo disk value followed by the string for that data point. If the datalog file cannot open, then the Adruino will print an error message. In order to keep the size of the data file from getting very large, a delay was added to the end of the code to decrease the collection rate. The delay function argument is set to 500 which represents a 500 millisecond pause between each reading of al 8 sensors. It should be noted that the code research was done using the Arduino Playground, an open source platform for Arduino programming information. A copy of the Arduino code can be seen in Appendix A of this document.

2.3 Matlab Script

After the Arduino code takes in the data, it can be put into the Matlab code for plotting. The data file is first imported into the workspace using the import tool. In the import tool, the data from the text file is recognized as a column since each data point is delimited by a space. Each column represents sensors 1 through 8 and can be renamed and imported into the workspace as single column arrays. Interestingly enough, the Arduino requires an array that must be converted to strings for the SD card which once again must be transformed into arrays in Matlab. Once the data has been imported, a script can be opened. In the script, there are commands for three separate plots; a calibration plot, a plot of the data while the horse and rider stood still and a plot of the data while the horse and rider were in motion.

To make the separate plots, the average of the imported data was calculated. This is to get an overall reading and balance out any outlying high or low data points. For the plots of the horse and rider standing still and in motion, the calibration averages were subtracted from the readings in order to ignore thhis

force. Then, arrays for x and y values are created in order to set the location of the plot points. Then an array for the color values is created to set the color values of the data points. The maximum and minimum values of color are also set up along with the size and scale of the color scale. A for loop is then implimented in order to systematically assign color values to each data point. The results are plotted as a color value at the set location of the x adn y arrays. A copy of the Matlab code can be seen in the appendix of this document.

3 Results and Analysis

3.1 Calibration

The first plot created by the Matlab script is a calibration plot. The data for this plot was taken while the saddle was on the horse while standing still with no rider. This data should not be taken into consideration when analyzing the effects of riding the horse on the performance and fit of the saddle. This pressure is from the saddle being secured on the horse and does not affect saddle fit. The plot is oriented as a bird's eye view of the saddle; the top right point corresponds to the sensor near the shoulder on the right side and the lower left point corresponds to the sensor at the rear of the saddle of the left side. In order to generate this plot, the average of each sensor value was determined. This is to account for the baseline pressure the saddle exerts while cinched up. Upon examining the data, the mean values of the sensors on the right side were close to those on the left. This shows the Piezo disks are uniformly accurate and that the saddle is sitting squarely on the horse. The plot also showed higher values at the front and in the rear for sensors 1, 3,4, 5, 7 and 8. Sensors 2 and 6 are located just behind the shoulder. These values are lower than the others becuase the saddle is intentionally fit loosely in this area so that the scapula is allowed to move freely. The calibration plot can be seen in figure 8. A table of the average values can be seen below.

Sensor Number	Sensor Value (mV)
1	212
2	146
3	202
4	213
5	236
6	126
7	193
8	201

3.2 Results

After the sensor system was calibrated to ignore the base force from the saddle being tightly secured, a plot was created to represent the data collected while the rider sat in the saddle and stood still. Again, this plot is shown as a bird's eye view of the saddle while on the horse's back. These values were calculated by subtracting the calibration values with no rider from the average of each sensor values when the horse and rider stood still. Here, we can see that the average pressure values are higher with the additional weight of the rider. Once again we see also see a decrease in force for the sensors 2 and 6 located just behind the horse's shoulder. This shows that even with the addition of the rider also sitting just behind the shoulder, the sensor value is still low and the horse's shoulder is still free from pinching or interference of movement. Sensor 6 has a much lower value than sensor 2, indicating a tighter fit on the right. This is also consistent with the horse's muscle structure- his right back muscle is more developed than the left. It is possible that the saddle may need to be readjusted to fit looser on the right, especially if the horse develops more muscle tone along the back muscles. The plot of the data collected while the horse was not in motion can be seen in figure 9. A table of the calibrated sensor readings can be seen below.

Sensor Number	Sensor Value (mV)
1	391
2	195
3	367
4	356
5	269
6	80
7	295
8	225

After examining the behavior of the saddle while horse and rider were standing, data was taken while the horse and rider were in motion. A plot of this data was created by subtracting the calibration values from the average values of the motion readings. As expected with a well-fitting saddle, the pressures evened out for all eight sensors during exercise. The slight variations in the force values also follow logic in that the rider does not distribute their weight ovenly across the entire sufrace of the saddle. An experienced rider sits lighter on the back edge of the sitting surface (location of sensors 3, 4, 7 and 8) while in motion than standing still. Also, sensors 1 and 8, located low on the horses shoulder, experienced a drop in force reading which is to be expected since the knees should only mildly touch the saddle while in motion. Sensors 2 and 6, the area behind the shoulder, should increase since the inner thigh is used strongly to control, direct and balance the horse which accounts for the increased readings in this area. A plot of the average values of the data collected while the horse was in motion can be seen in figure 10. A table of the average values recorded while the horse was in motion can be seen below.

Sensor Number	Sensor Value (mV)
1	250
2	230
3	240
4	235
5	245
6	238
7	245
8	234

4 Conclusion

In English riding, proper saddle fit is vital to the comfort and performance of the horse and rider. Conventionally, saddles are fit using innacurate visual tests and dust or sweat patterns left on the horse's skin after exercise. In order to create a more accurate and dynamic saddle fitting tool, I have created a pressure sensor system and mapping routine. After attempting to use a variable force resistor, a more accurate and intuitive measure, the physical limitations of the sensors required a different approach. In order to compensate for these limitations, Piezo disks were used instead to record the differences in voltage when pressure was applied to the disks. It was almost astonishing how well the Piezo disks worked and how logical the data is.

In order to create the sensor system, a 4051 multiplexer and Piezo disks were configured and connected to an Arduino Uno microprocessor. The information collected from the sensor was then read by the Arduino and stored onto an SD card. The SD card data was then interpereted by a Matlab script and plots of the data were created. The data points were set in the location of the corresponding sensor and a color value was assigned to the data point.

After careful calibration, the data showed patterns that reflected the expectations of the system. The data also showed that the horse and rider combination have a proper fitting saddle and that the rider appropriately uses their body while the horse is in motion. In order to further develop this system, more calibration is necessary to determine the exact force applied to each area as well as a pad that allows the sensors to be easily rearranged. The exact location of the sensors can also be determined with more calibration and measurement. Though there can be a lot of improvement to the system to make it more reliable and easier to use, the current configuration shows accurate readings that are easily repeated.

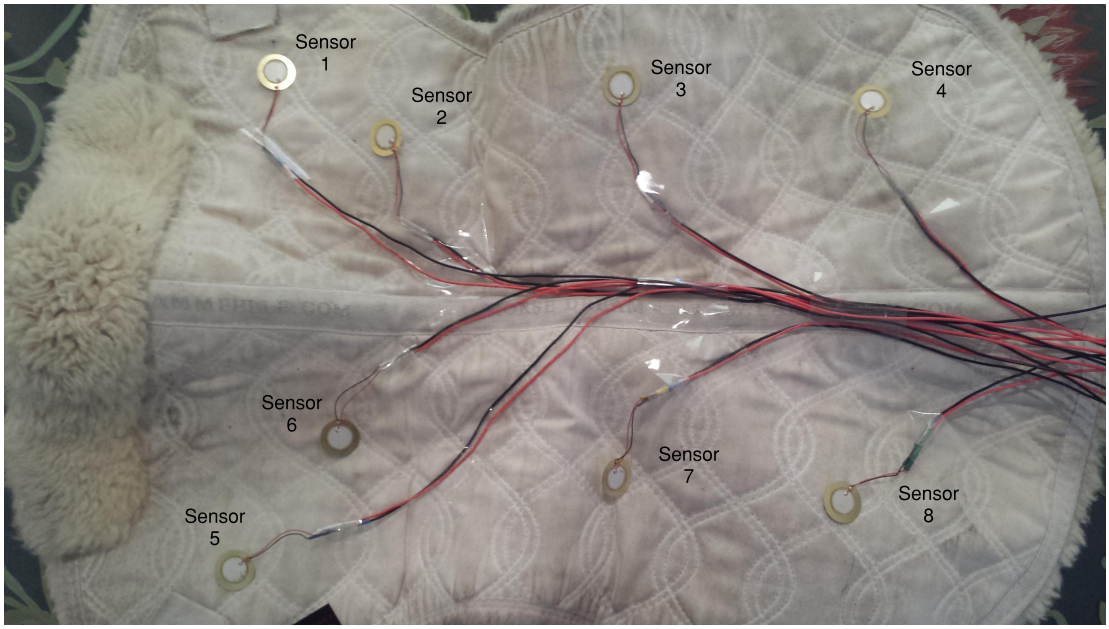


Figure 1: View of the Pad with Sensors

Figure 2: Side View of the Sensor System

Figure 3: Rear View of Sensor System

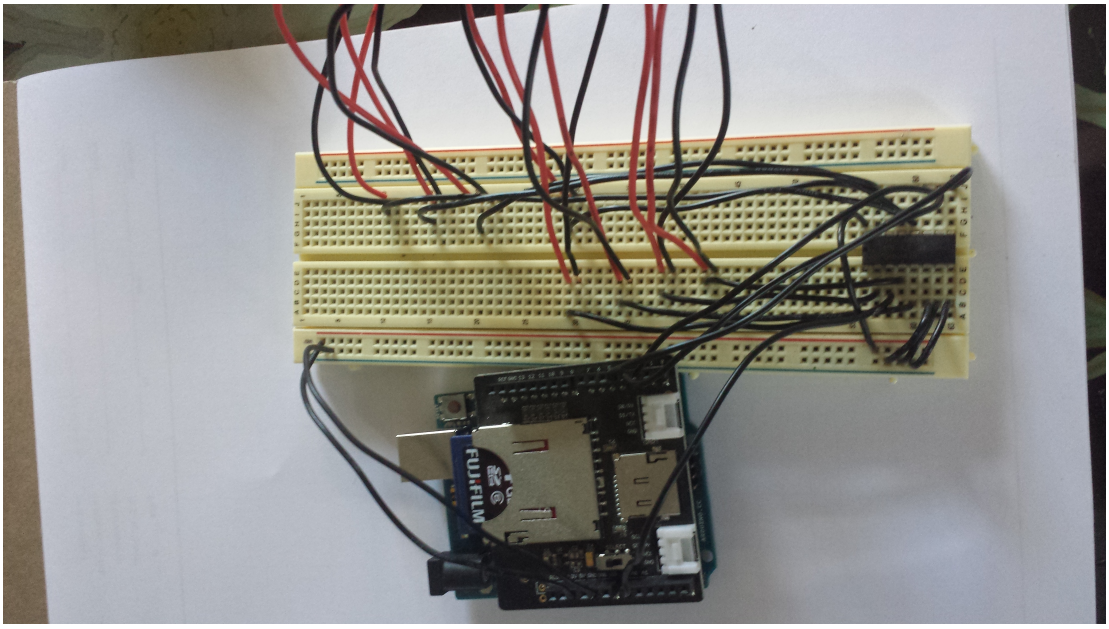
Picture.jpg 

Figure 4: Circuit Layout

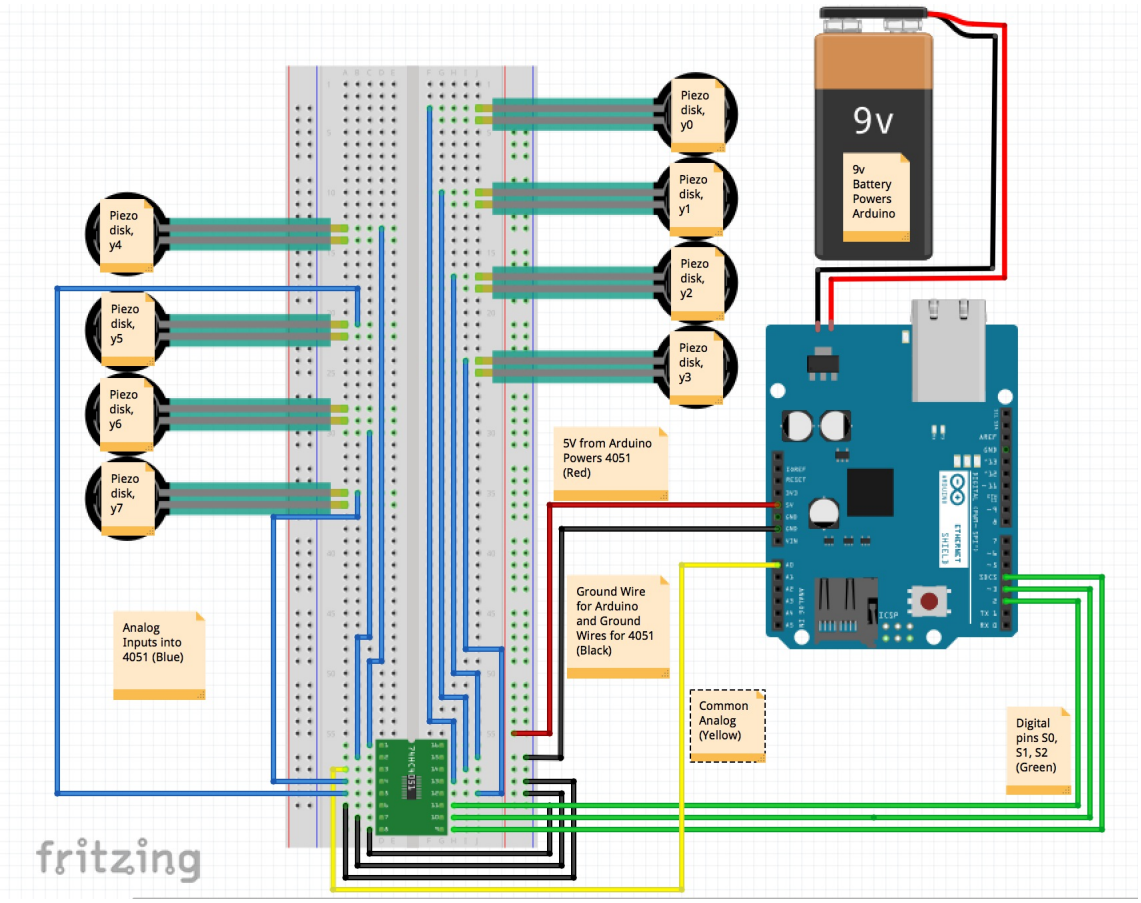


Figure 5: Fritzing Circuit Diagram for Saddle Sensor System

CD4051

INPUTS				ON CHANNEL
INH	C	B	A	
L	L	L	L	0
L	L	L	H	1
L	L	H	L	2
L	L	H	H	3
L	H	L	L	4
L	H	L	H	5
L	H	H	L	6
L	H	H	H	7
H	X	X	X	None

X = don't care

Figure 6: Counting Logic for Select Pins A, B, C (S0,S1,S2)

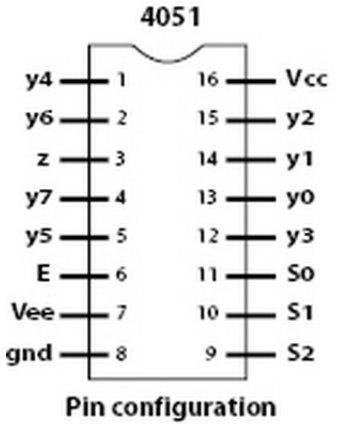
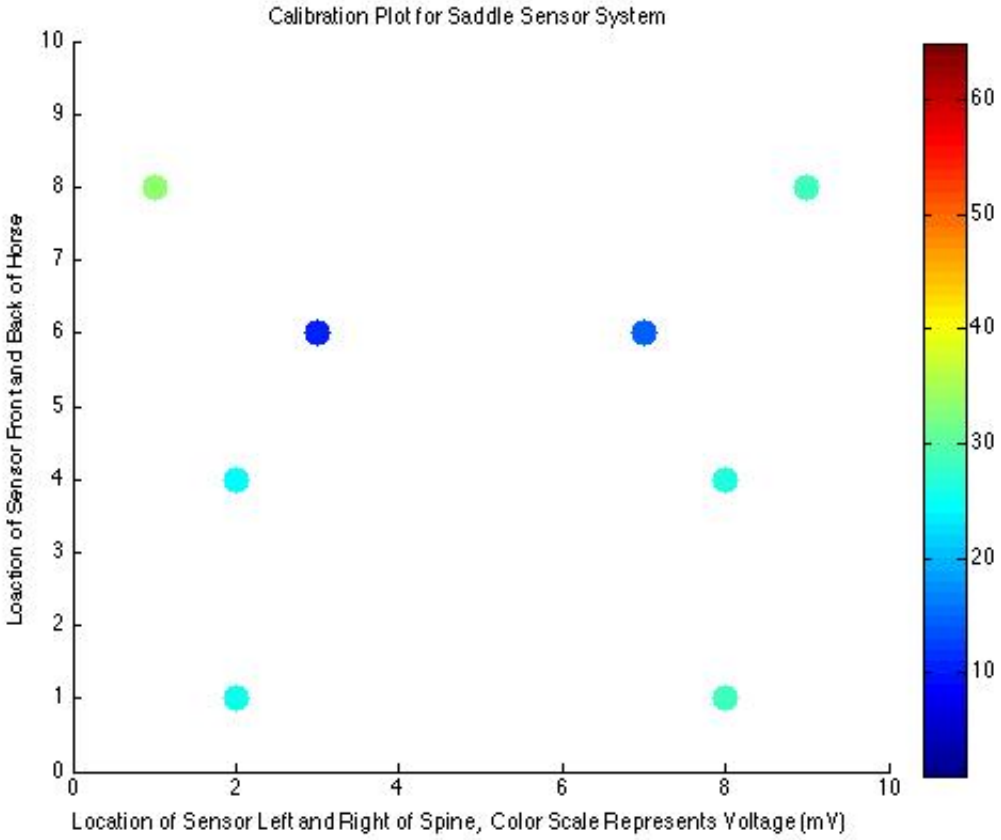
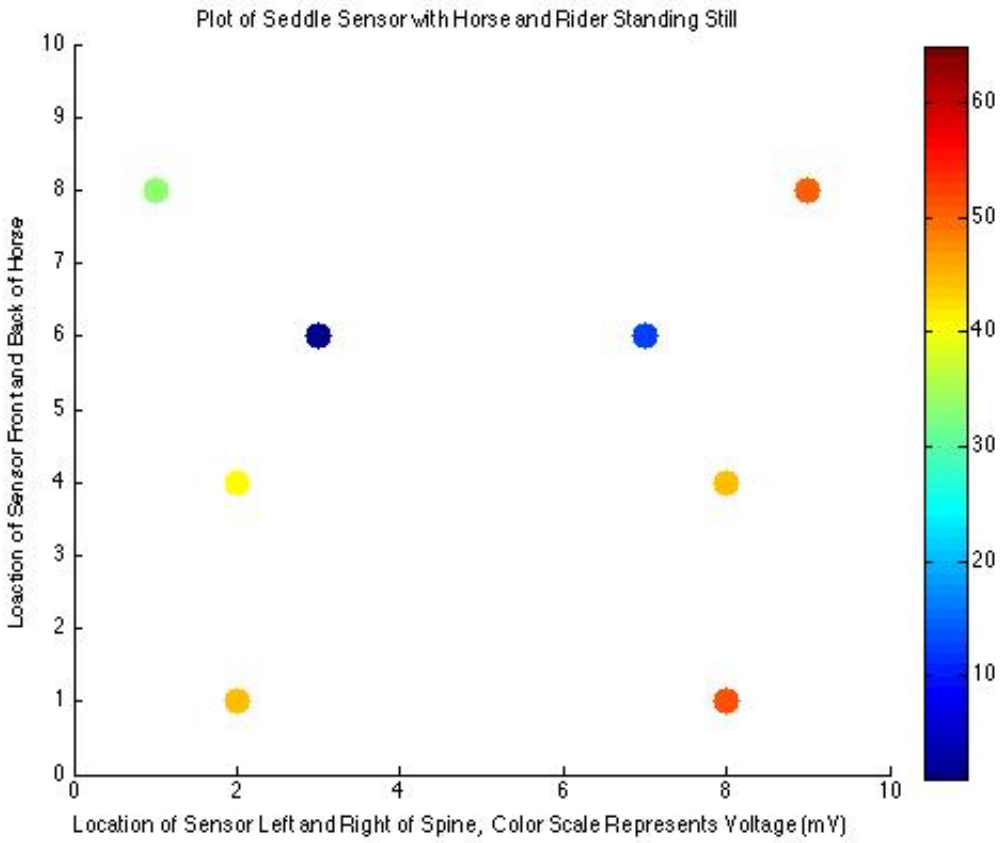


Figure 7: 4051 Pin Configuration



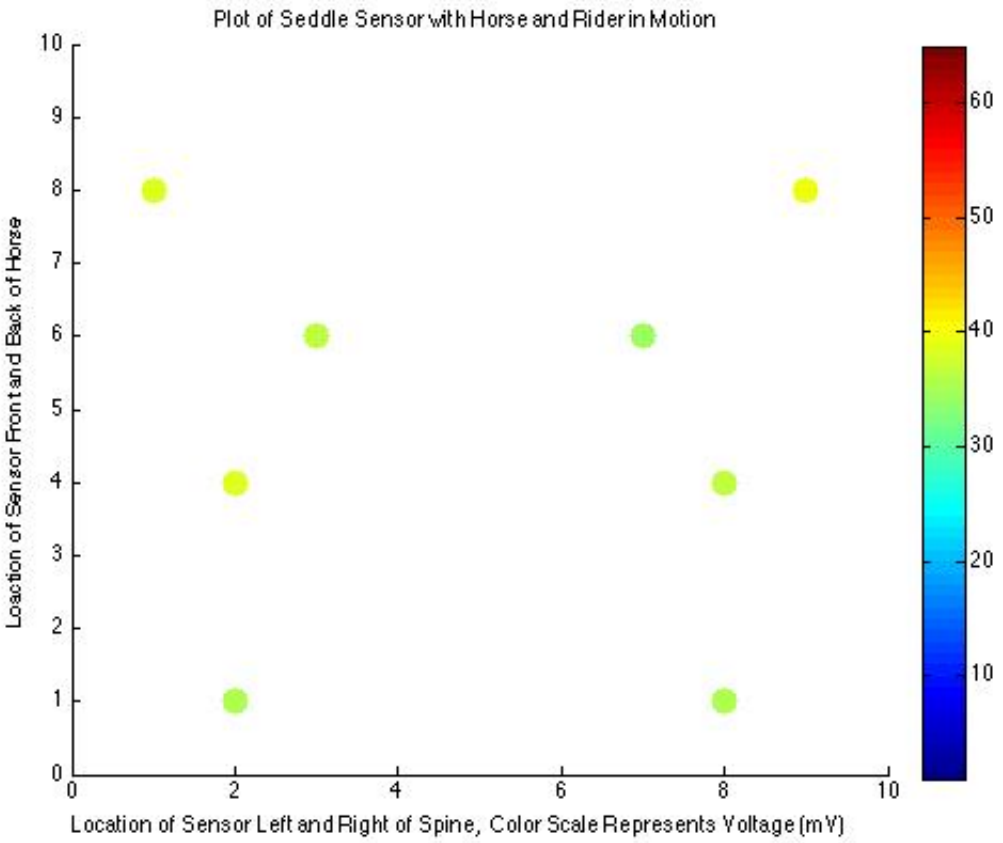
Plot.jpg

Figure 8: Calibration Plot of Saddle Sensor



While Still.jpg

Figure 9: Plot of Saddle Sensor with Horse and Rider Standing Still



In Motion.jpg

Figure 10: Plot of Saddle Sensor with Horse and Rider in Motion

Appendix

A Arduino Sketch

```
/*
Code for Saddle Force Sensor by Molly Totten
/
#include <SPI.h> //initialize for SD card
#include <SD.h> //initialize for SD card
const int chipSelect = 4; //for initializing SD card, set CS default pin
```

```
int mysensorvalue[8]=0,0,0,0,0,0,0,0; //sets up an array for the data to go on SD card
int r0 = 0; // set select pin value, s0, for 4051 set up
int r1 = 0; // set select pin value, S1, for 4051 set up
int r2 = 0; // set select pin value, S2, for 4051 set up
int count = 0; // set up count to select analog input pins y0 through y7
int sensorPin = A0; // select the common analog input pin for the Piezo disk
int PiezoValue = 0; // variable to store the value coming from the sensor


    void setup(){
pinMode(2, OUTPUT); // set S0 to be a digital output (digital pins 0,1 not used)
pinMode(3, OUTPUT); // set S1 to be a digital output (digital pins 0,1 not used)
pinMode(4, OUTPUT); // set S2 to be a digital output (digital pins 0,1 not used)
pinMode(10, OUTPUT); //set default chipSelect pin to output mode. Here, hardware SS pin is 10 by default
Serial.begin(9600); //open serial communications for SD card
Serial.print("Initializing SD card..."); //print SD initialization
// see if the card is present and can be initialized:
if (!SD.begin(chipSelect)) {
Serial.println("Card failed, or not present"); //stop reading if no SD card present
return;
}
Serial.println("card initialized."); //confirm that card has initialized and opened
}


    void loop () {
String dataString= " ";
for (count=0; count¡=7; count++) { //use counting logic to select digital pins S0,S1,S2
r0 = bitRead(count,0); // read bit S0 (read 1st number from count, get value of S0)
r1 = bitRead(count,1); // read bit S1 (read 2nd number from count, get value of S1)
r2 = bitRead(count,2); // read bit S2 (read 3rd number from count, get value of S2)
digitalWrite(2, r0); //set S0 to the value of r0
digitalWrite(3, r1); //set S1 to the value of r1
digitalWrite(4, r2); //set S2 to the value of r2
```

```
PiezoValue = analogRead(sensorPin); //read the analog value of the Piezo disk
mysensorvalue[count]=PiezoValue; //use values of each PiezoValue in count for
the array mysensorvalue
dataString += String(PiezoValue); //convert data string from array to string
dataString += " "; //add space in string
Serial.print(count); //print "count"
Serial.print(" "); //add space
Serial.println(PiezoValue); //print "PiezoValue"
}
File dataFile = SD.open("datalog.txt", FILE_WRITE); //open the SD file
if (dataFile) { //put the datafile strings onto the SD card file
dataFile.println(dataString); //put the datafile strings onto the SD card file
dataFile.close(); //close ater putting onto SD card
Serial.println(dataString); //print to serial port
}
else {
Serial.println("error opening datalog.txt"); //print error message if datalog not
available
}
delay(500); //add a delay to decrease data collection rate
}
```

B Matlab Script

```
%First, the data must be imported into the workspace via the import tool.
%The data is extracted from the text file created by the SD card. The data
%values are recorded as separate columns in the text file and import as
%nx1 arrays where n is the number of data points.
%Each sensor value column was renamed according to their wiring order and
%their data type (ex. Sensor1Rest)


    Rest1 = mean(Disk1Rest); %The Rest1-Rest8 variables are averages of the
Rest2 = mean(Disk2Rest); %ensor readings while the saddle sits on the
Rest3 = mean(Disk3Rest); %horse without a rider
Rest4 = mean(Disk4Rest);
Rest5 = mean(Disk5Rest);
Rest6 = mean(Disk6Rest);
```

```
Rest7 = mean(Disk7Rest);
Rest8 = mean(Disk8Rest);
disp(Rest1); % display the values of the data points in the command window
disp(Rest2);
disp(Rest3);
disp(Rest4);
disp(Rest5);
disp(Rest6);
disp(Rest7);
disp(Rest8);
x = [9,7,8,8,1,3,2,2]; %set up location of points for x and y
y = [8,6,4,1,8,6,4,1];
colorweight = [Rest1,Rest2,Rest3,Rest4,Rest5,Rest6,Rest7,Rest8];
%set color weight values according to the sensor values calculated in
%Restl1-Rest8
zmin=0; %set minimum color value
zmax=max(colorweight); %set maximum color value
map=colormap; %create new varable called map
colorsteps=size(map,1); %set size/scale of the color scale hold on
for i=1:colorsteps %go through color scale
ind=find(colorweight¡zmin+i*(zmax-zmin)/colorsteps & colorweight¿=zmin+(i-1)*(zmax-
zmin)/colorsteps); %assign x and y pairs to the colorweight and colorsteps to assign
and
%scale colors
plot(x(ind),y(ind),'.','Color',map(i,:),'MarkerSize',40); %plot the data
%adjust plot settings
xlabel('Location of Sensor Left and Right of Spine, Color Scale Represents Voltage
(mV)')
ylabel('Loaction of Sensor Front and Back of Horse')
title('Calibration Plot for Saddle Sensor')
axis([0 10 0 10]);
end


    % Plot data taken while horse and rider are still
Still1 = mean(Disk1Still)-Rest1; %The Still1-Still8 variables are calibrated
Still2 = mean(Disk2Still)-Rest2;%values (rest1-rest8) subtracted from
Still3 = mean(Disk3Still)-Rest3;%the averages of the sensor readings
Still4 = mean(Disk4Still)-Rest4; %while the horse and rider were in motion
Still5 = mean(Disk5Still)-Rest5;
```

```
Still6 = mean(Disk6Still)-Rest6;
Still7 = mean(Disk7Still)-Rest7;
Still8 = mean(Disk8Still)-Rest8;
disp(Still1); %display the values of the data points in the command window
disp(Still2);
disp(Still3);
disp(Still4);
disp(Still5);
disp(Still6);
disp(Still7);
disp(Still8);
x2 = [9,7,8,8,1,3,2,2]; %set up the locations of the plot points for x and y
y2 = [8,6,4,1,8,6,4,1];
colorweight2 = [Still1,Still2,Still3,Still4,Still5,Still6,Still7,Still8];
%set color weight values according to the sensor values calculated in%Still1-Still8
zmin=0; %set minimum color value
zmax=max(colorweight2); %set maximum color value
map=colormap; %reate new varable called map
colorsteps=size(map,1)%set size/scale of the color scale
hold on
for i=1:colorsteps %go through color scale
ind=find(colorweight¡zmin+i*(zmax-zmin)/colorsteps & colorweight¿=zmin+(i-1)*(zmax-
zmin)/colorsteps);
%assign x and y pairs to the coloreight and colorsteps to assign and
%scale colors
plot(x2(ind),y2(ind),'.','Color',map(i,:),'MarkerSize',40);%plot the data
%adjust plot settings
xlabel('Location of Sensor Left and Right of Spine, Color Scale Represents Voltage
(mV)')
ylabel('Loaction of Sensor Front and Back of Horse')
title('Plot of Seddle Sensor with Horse and Rider Standing Still')
axis([0 10 0 10]);
end


    %Plot data taken while horse and rider were in motion
Motion1 = mean(Disk1Motion)-Rest1; %The Motion1-Motion8 variables are cali-
brated
Motion2 = mean(Disk2Motion)-Rest2; %values (rest1-rest8) subtracted from
Motion4 = mean(Disk4Motion)-Rest4; %the averages of the sensor readings
```

```
Motion5 = mean(Disk5Motion)-Rest5;%while the horse and rider were in motion
Motion6 = mean(Disk6Motion)-Rest6;
Motion7 = mean(Disk7Motion)-Rest7;
Motion8 = mean(Disk8Motion)-Rest8;
disp(Motion1); %display the values of the data points in the command window
disp(Motion2);
disp(Motion3);
disp(Motion4);
disp(Motion5);
disp(Motion6);
disp(Motion7);
disp(Motion8);
x3 = [9,7,8,8,1,3,2,2]; %set up the locations of the plot points for x and y
y3 = [8,6,4,1,8,6,4,1];
colorweight3 = [Motion1,Motion2,Motion3,Motion4,Motion5,Motion6,Motion7,Motion8];
%set color weight values according to the sensor values calculated in
%Motion1-Motion8
zmin=0; %set minimum color value
zmax=max(colorweight3); %set maximum color value
map=colormap; %create new varable called map
colorsteps=size(map,1); %set size/scale of the color scale
hold on
for i=1:colorsteps %go through color scale
%assign x and y pairs to the coloreight and colorsteps to assign and
%scale colors
ind=find(colorweight3¡zmin+i*(zmax-zmin)/colorsteps & colorweight3¿=zmin+(i-
1)*(zmax-zmin)/colorsteps);
plot(x3(ind),y3(ind),'.','Color',map(i,:),'MarkerSize',40); %plot the data
%adjust plot settings
xlabel('Location of Sensor Left and Right of Spine, Color Scale Represents Voltage
(mV)')
ylabel('Loaction of Sensor Front and Back of Horse')
title('Plot of Seddle Sensor with Horse and Rider in Motion')
axis([0 10 0 10]);
end
```

References

[1] Dennis L. Eggleston. *TBasic Electronics*. Cambridge Press, Cambridge, United Kingdom, 1953.

[2] Carl Nave. *Piezoelectricity.*, 2005
`http://hyperphysics.phy-astr.gsu.edu/hbase/solids/piezo.html`