Posture Guidance chair

ECE 445 Design Document

Alex Shen, Pablo Corral Vila, and Emre Ulusoy
Team 58
TA: Kexin Hui

3/26/18

1 Introduction

1.1. Objective

According to the American Chiropractic Association, back pain is the leading cause of disability worldwide where it is estimated that 31 million Americans experience back pain [1]. The cost to remedy these back pain issues is reported at about $50 billion [1]. Among the many potential causes of back pain is improper posture that causes harm to the spine and many muscle groups in the body (depending on the pose) [2], [3]. Looking further at the displays of bad posture, it is noted that unhealthy sitting can lead to poor body positioning for prolonged periods of time, which adds stress to the muscles and bones [2]. It was found that people with 'S' shaped spines, possibly due to excessive slouching, tend to have more back problems than people with 'J' shaped spines [3]. Therefore, proper posture is essential to recovery and prevention of back problems, but this is more difficult to achieve once poor posture has been adopted [2].

Our goal is to develop a system of sensors embedded on a chair that can detect poor posture and notify the user of the presence of poor body positioning. We will use pressure sensors on the seat and the back as the main component of recognizing the weight distributions as well as a distance sensor to map out the position of the user. This sensor data will be sent to a computer for processing. We will use the data to develop a supervised learning model that will detect if the posture of the user is good or bad. If bad, the system may notify and guide the user to a better position through haptic feedback with vibration motors. The sensor data can also be used to analyze trends and provide insight on how to correct past and current detected posture issues. This addresses the problem of prolonged poor sitting posture since it notifies and educates the user about the risks of the user's positions.

1.2. Background

As mentioned above, bad posture is a serious problem that could lead to health issues after prolonged use. Unfortunately, it's possible to adopt bad posture subconsciously, which can be effectively countered by maintaining awareness for the user of such an issue and repeatedly correcting it [4]. As technology advances and more jobs require sitting for long periods of time, it's becoming more important to develop safe methods that maintain the health and well-being of users [3]. Thus, a product that can aid in the correction of posture and isn't as physically intrusive as many other proposed solutions would be ideal.

When it comes to classifying whether a posture is good or bad, the distinction can be quite difficult to make. Figure 1 shows the differences between some common postures. Notice the difference in spinal curvature and center of mass. This difference is what we'll be attempting to measure with the pressure sensors measuring the weight distribution and contact, and the distance sensor will be detecting the mid-upper spine curvature. There are many conflicting

opinions about what is the ideal sitting posture. For this project, we consider flat, long lordosis, and short lordosis positions as generally acceptable with preference for short lordosis [7].

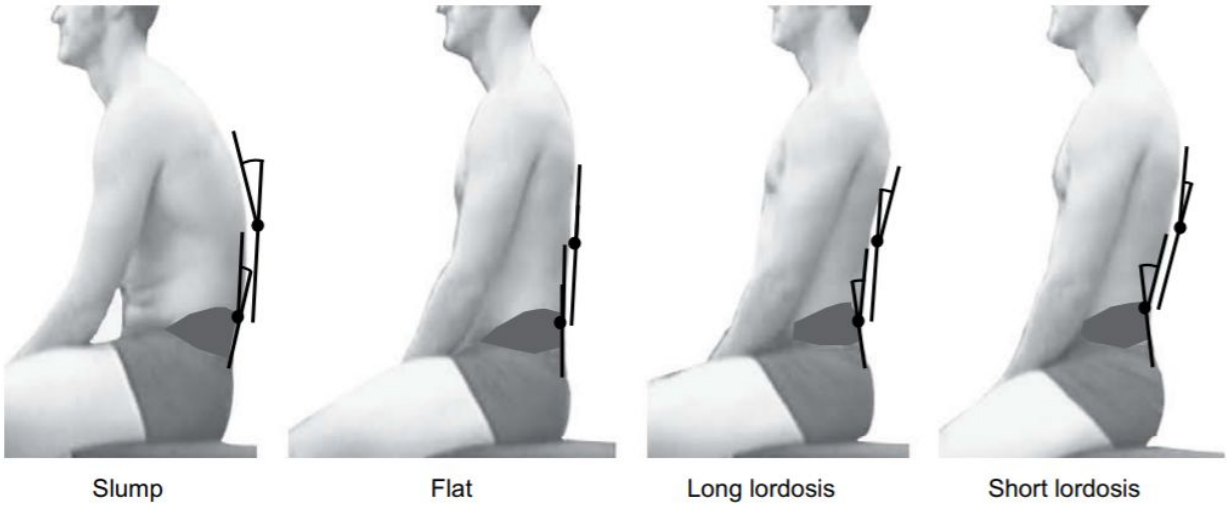

Figure 1: Four common sitting postures. Adapted from [7].

An illustration of some acceptable sitting angles is shown in figure 2. Note that while the back position may be the same, the pressure distribution can be drastically different.

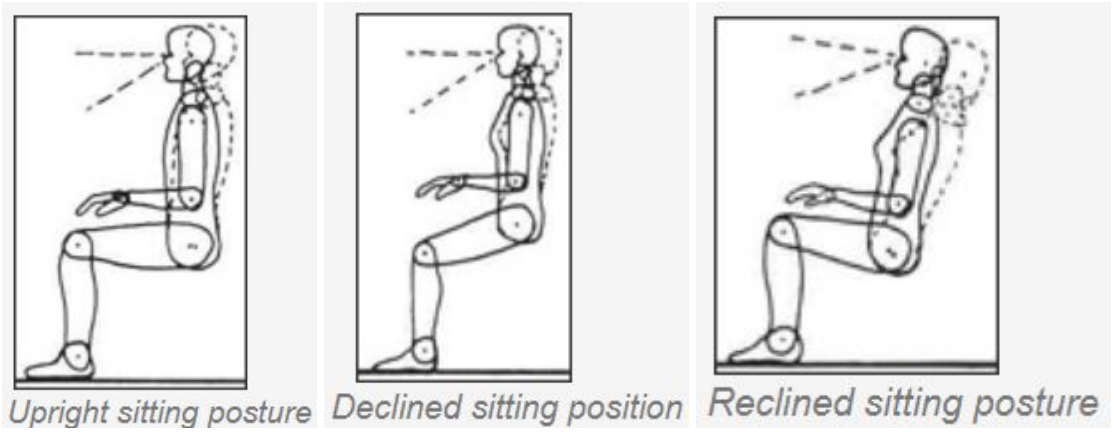

Figure 2: Three good working positions according to OSHA. Adapted from [6].

From our observations in [1], [2], [6], and [7], we have determined four main cases when we consider a posture is bad:

- When the user spends too much time in the same position (lack of movement)
- When the back of the user is too far from the back of the chair (kyphosed)
- When the bottom of the user is too far from the back of the chair (slouching)
- When the user sits with more strength on one of the sides of the chair (imbalanced)

The features that make this project marketable include the many sensors embedded in a commonly used object (chair) that can help provide data to improve the health of the user. Not only does it remind the user to correct their posture, the data can also be processed and displayed on the computer for the user to understand what's going on and how they can improve their positioning. To display the data, we will generate an illustration of the detected posture issues e.g. heat map of pressure imbalance. This project essentially aims to combat the common issue of bad posture with the use of a chair, which is commonly used for sitting for long periods of time.

1.3. High-level requirements

- The sensors must be sensitive enough to detect changes in sitting positions shown in figures 1 and 2 for adults. The microcontroller should distinguish a resolution of approximately 2cm for the ultrasonic sensor and 100g of mass for the pressure sensors to satisfy this sensitivity.
- The posture detection model must be able to correctly classify bad sitting posture at least 85% of the time. Correctness of posture is defined as a slight lordosis of the lumbar and flat or slightly kyphosed thoraco-lumbar angle [7]. OSHA recommended positions in figure 2 must also be considered [6].
- The chair must be power-efficient, ideally able to last at least 10 hours without changing batteries. For ~2500 mAh batteries, this corresponds to approximately no more than 250 mA of current being drawn by the entire system.

2 Design

Figure 3 shows the block diagram of the system. The battery supplies power to the voltage regulators to distribute the appropriate voltages to the rest of the modules. The sensors provide data to the microcontroller that will be sent over Bluetooth to a computer for processing. The data collected by the computer will also be used to display trends and provide recommendations on the user's posture along with the haptic feedback guidance from the vibration motors. These modules will satisfy the high-level requirements because the minimum amount of sensors and power is used to allow the system to work without draining too much power. The voltage regulator is more efficient than a voltage divider in that regard because current is drawn separately for the regulator than it is for the load. Also, the multiple sensors feeding information to the microcontroller attempt to provide enough data for higher classification rates and granular data with the help of the analog multiplexer.

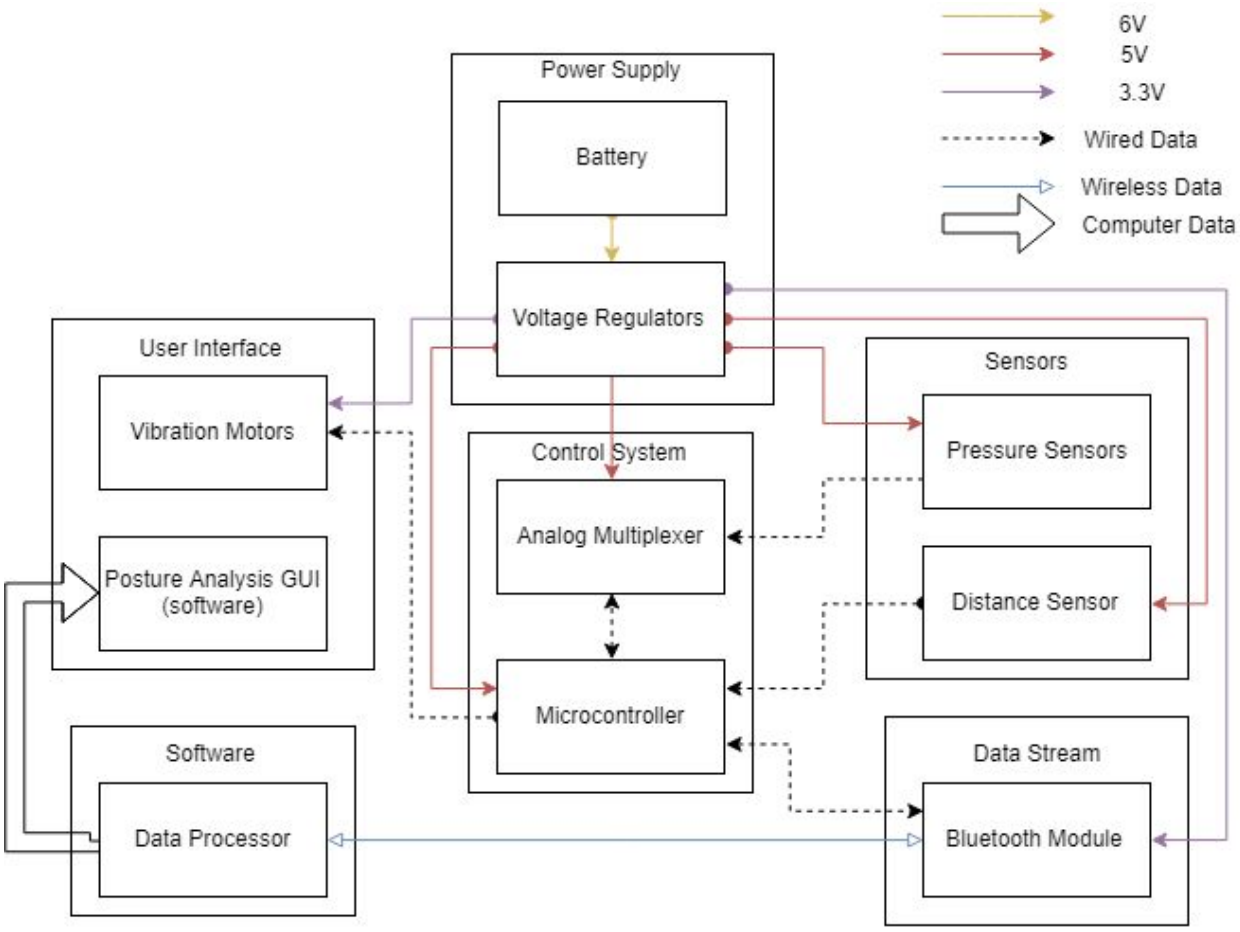

Figure 3: Block diagram

Figure 4 shows the rough physical layout of the system. The pressure sensors on the seat attempt to record pressure changes in the areas near the knee and under the thigh, ischial tuberosities, lumbar, and shoulders. These sensor placements were found to be the most important and distinguishable areas for detecting different postures [8]. The distance sensor is mounted on the back, where it will sense the presence of a user and also detect the distance from the user's spine. The distance detected from the user's spine will help determine if the user is slouching along with the sensors near the knee. A single distance sensor is used because the user is expected to sit directly in front of the sensor. Even if the user is not sitting directly in front of the sensor, the pressure sensors will be able to conclude that based on the weight distribution. The rest of the components will be mounted behind the chair to avoid physical damage. Note that the actual number and placement of sensors will be determined through trial-and-error, but the targeted areas of the body to sense will be the same.

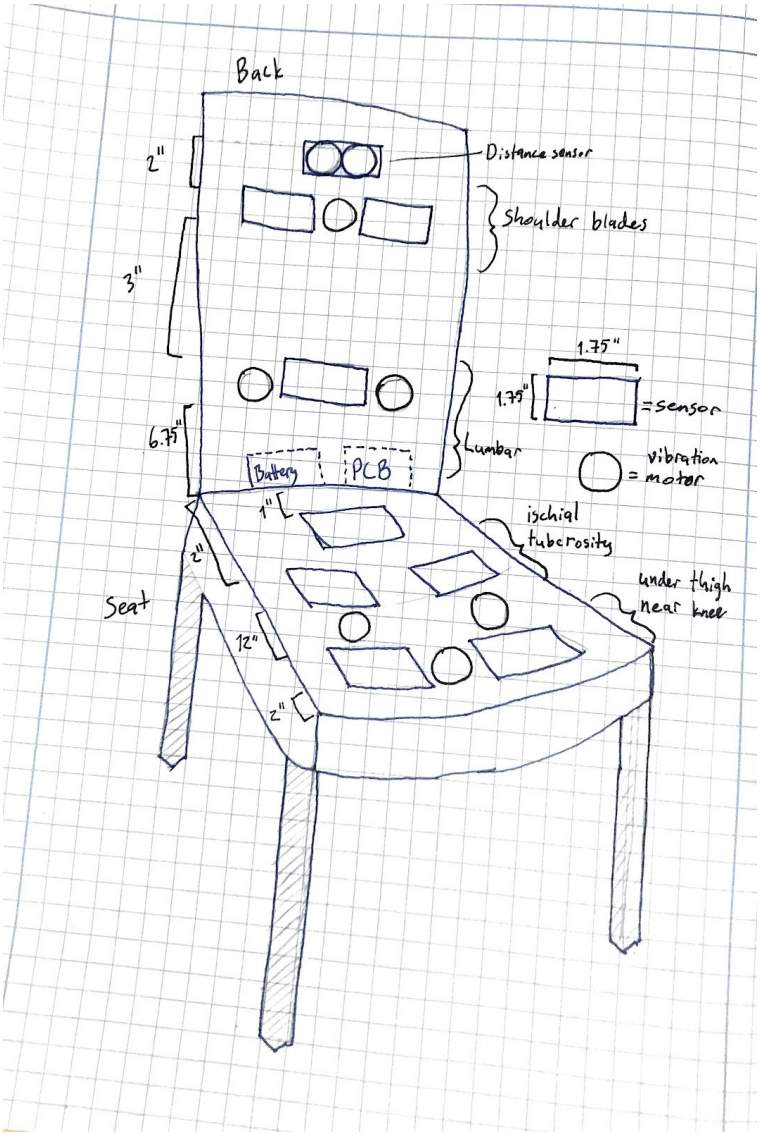

Figure 4: Physical design sketch

To work properly, the chair system will need six modules: power supply, sensors, control system, data stream, user interface and software.

2.1. Power supply

The power supply module is responsible for providing the power used by the other modules/components. The main voltages used are 3.3V and 5V. Adding up the power consumption from all the components below, we obtain the following table:

Module/Component	Approximate Power Usage (mW)
Voltage Regulators	240
Pressure Sensors	18.4
Distance Sensor	75
Microcontroller	26
Bluetooth	26.4
Vibration Motors	240

Table 1: Power Usage by Module/Component

Therefore, the approximate maximum power consumption is ~625.8 mW.

2.1.1. Battery

The battery is the source of the power in the power supply module. We will not be designing it because the off-the-shelf product is sufficient for this project's needs. The total voltage must be above the required 5V for most components. Therefore, the simplest option appears to be 4 Energizer E91 alkaline AA batteries in series. Each battery contains up to 1.5V, so this provides 1.5V*4 = 6V and the batteries have a capacity of approximately 2500mAh at a discharge current of 100mA [17]. Using Eq. 6, we find the current from the 6V batteries required to satisfy the 686 mW power consumption of the system is ~104.3 mA. Therefore, the total time between complete discharges is slightly less than 2500 mAh / 104.3mA = ~24 hours, assuming a constant current and linear discharge. This is sufficient for our power usage high-level requirement, and can be used for several standard work days before requiring a change of batteries. Keep in mind that this power consumption calculation is more of a worst-case estimate since it assumes the vibration motors will always be on and drawing 80 ~ 120 mW of power and that the voltage regulators are drawing 120 mW each (which is not as efficient as off-the-shelf solutions).

No requirements and verification are needed since we will not be designing any aspects of this component.

2.1.2. Voltage regulators

The voltage regulators have the duty of maintaining the desired voltages from the battery to the 3.3V and 5V values. The components that will be using 3.3V are the Bluetooth module and the vibration motors. The 5V components are the microcontroller, analog multiplexer, pressure sensors, and distance sensor will be using 5V.

To add complexity, we decided to make our own voltage regulators. We will need a total of two regulators since we need one to produce ~3.3V and the other to produce ~5V. The design includes a BJT NPN transistor and a Zener diode. It takes advantage of the Zener diode properties, which includes the fact that its voltage is approximately constant when it is in breakdown mode.

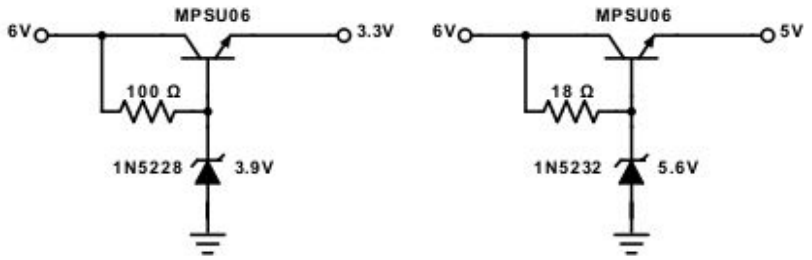

Figure 5: 3.3V (left) and 5V (right) voltage regulator schematics.

To calculate the components, we used the fact that the NPN diode MPSU06, by Motorola, has a emitter-base voltage of 0.6V [19]. Thus, knowing that and the output voltage, we can easily calculate the Zener diode voltage (5.6V for the 5V regulator and 3.9V for the 3.3V regulator). To calculate the resistor value, we first calculated the voltage across it, which is the amount of voltage to dissipate until it reaches the Zener diode breakdown voltage. Then, we followed the formula:

$$R_v = \frac{V_{Rmin}}{I_{Dmin} + \frac{I_{Lmax}}{h_{FE}+1}} \qquad \text{Eq. 1}$$

Where I_{Dmin} is the minimum current in the Zener diode, which was taken from their datasheets: 20 mA for 1N5228 and 1N5232 [18]. I_{Lmax} is the maximum load current, which was taken from making calculations of the elements that are going to be supplied by each linear regulator. We concluded that the 5V regulator, which powers the sensors, the microcontroller, and the analog multiplexer, has a load current of approximately 119.4 mA, and the 3.3V regulator, which powers the Bluetooth module and vibration motors, has a maximum current of ~128 mA. Refer to section 2.4.1 Bluetooth to understand the disparity in power consumption and maximum current of the 3.3V load. Moreover, we know that the parameter h_{fe}, which relates the currents through base and collector, is 60 for this transistor [19]. With equation 1, we are able to calculate an approximate R_v, which is ~18 Ω for the 5V regulator and ~100 Ω for the 3.3V regulator.

Since the voltage needs to be reduced from the 6V batteries, this results in some power dissipated in the voltage regulators. This current is roughly 20 mA using 6V each. Therefore, 240 mW of power will be used up to regulate the voltages of 5V and 3.3V from the 6V source. While it's possible to use a true minimum current through the zener diodes (somewhere around 1mA), the recommended/best practice is to use the test voltage.

We did some simulations for voltage regulators using Simscape (a Matlab extension). We entered the next parameters: a Zener resistance of 0.1ohm for the Zener diodes, input voltages of 5.5-6V (to see what happens if it varies) and we modeled the load as resistances. Knowing the output voltage and current, we used Ohm's law to get the resistance values (80 Ω for the 3.3V regulator and 60 Ω for the 5V regulator).

The results were good: a voltage of ~3.2V on the 3.3V regulator, and a voltage of ~4.8V on 5V one. These values are within the tolerance margin. The results are shown for the 5.5V and 6V inputs for the 5V regulator and similar results on the 3.3V regulator in figures 6 through 8.

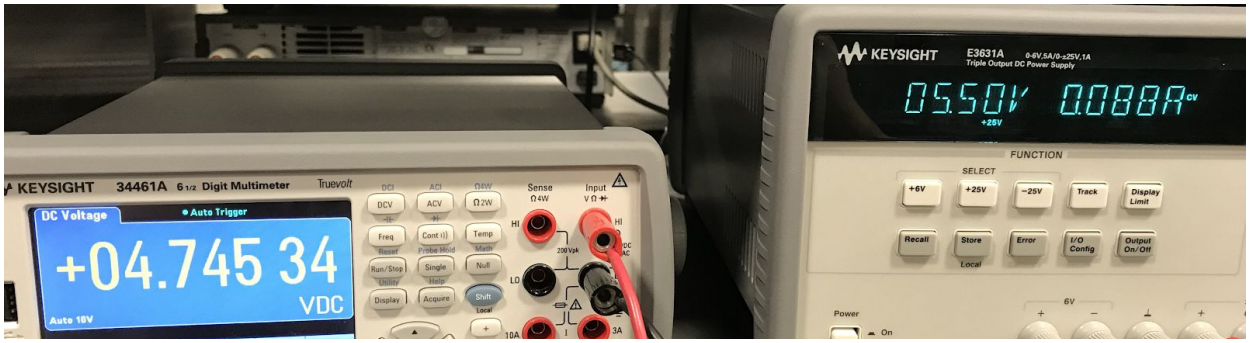

Figure 6: Output of the 5V voltage regulator given 5.50V input

Figure 7: Output of the 5V regulator given a 6.00V input

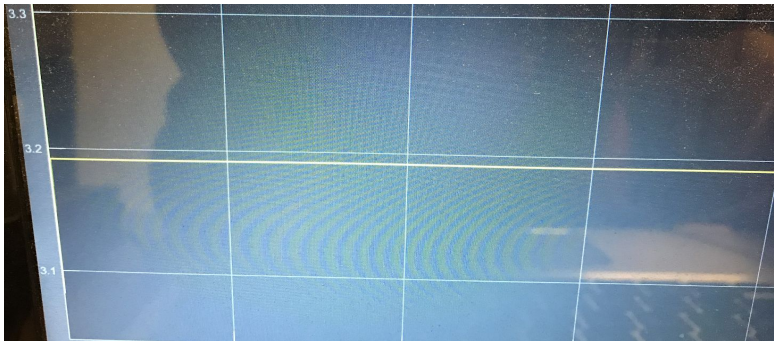

Figure 8: Output of the 3.3V voltage regulator given a 6V input

Requirement	Verification	Points
1. The 3.3V voltage regulator must consistently output voltages between 3.0V - 3.6V given an input of 5.5-6V. 2. The 5V voltage regulator must consistently output voltages between 4.5V - 5.5V. 3. The temperature must not exceed 125℃ while supplying power to avoid injury, destruction, and quicker depletion of the batteries. High temperatures will also affect the current output since V_{BE} and V_Z decrease as temperature increases [18], [19]	1. A. Provide a 5.5-6V input signal into the 3.3V regulator B. Check output voltage to make sure it's within the desired range of 3.0-3.6V 2. A. Provide a ~6V input signal to the 5V regulator B. Check the output voltage to ensure it's within the range of 5V+/-0.5V 3. A. During full operation of the system, measure the maximum temperature using a temperature sensor. B. Also measure the current to ensure	1. 1 2. 1 3. 1

Table 2: Voltage Regulators RV Table

2.2. Sensors

2.2.1. Pressure Sensors

We will use 8 pressure sensors to detect the critical areas of distinguishing postures as illustrated in figure 4. The pressure sensors we will be using are SEN-09376 Force Sensitive Resistors (FSR). These 1.75”x1.75” FSRs are flexible and can support applied force in the range of 100g to 10kg [11]. Load cells are an alternative to FSRs, and they can provide more accurate and sensitive readings. However, load cells are more difficult to work with since many other components are needed to amplify and detect the signal changes. There are also issues with placing them physically on the chair since the center tab needs elevation to allow displacement and the metal bracket will not be comfortable to sit on. The FSR we will be using is convenient in that the thinness and flexibility allow more comfort without being drastically more expensive. The primary concern is that there exists a possibility that the supported force range is not sufficient. We will now perform an analysis on the feasibility of the 0.1-10 kg range.

Considering a 100 kg user and the average sitting area of adults is 179.4 in^2 [10]. To figure out the pressure, we first need to obtain the force. Force is the product of mass and acceleration as shown in Eq. 2 below.

$$F = m * a \qquad \text{Eq. 2}$$

Using Eq. 2 with m = 100 kg and the acceleration due to gravity a = 9.8 m/s^2, the force of the user is calculated as 980 N. Pressure is defined as force per area as shown in Eq. 3.

$$Pressure = \frac{F}{Area} \qquad \text{Eq. 3}$$

Thus, applying the user’s force F = 980 N and the sitting area = 179.4 in^2 = 0.1157 m^2, we get the resulting pressure of 8470 Pa. Using the conversion rate of 1 Pa = 0.000145 psi, the conversion to psi is now 1.228 psi. Now multiplying the pressure with the SEN-09376 area of 1.75”x1.75”, we obtain the number of pounds per sensor as 3.76 lbs. Converting this to kilograms yields a result of 1.706 kg per sensor. Therefore, an even distribution of force of a 100 kg user on the average sitting area is well within the range of the SEN-09376 Force Sensitive Resistor.

The power consumed by the FSR is easy to calculate. We first consider the relationship of resistance and force. As seen in figure 9, the relationship is linear since both scales are logarithmic. Based off of the calculations above, the FSR should be set to approximately 1 kΩ and will be put in series with a 10 kΩ resistor as recommended in [11]. The equation for power in terms of voltage and resistance is shown below.

$$P = \frac{V^2}{R} \qquad \text{Eq. 4}$$

Therefore, applying V = 5V and R = 11 kΩ, the average power consumption for each sensor is ~2.3 mW. The average total power for 8 sensors is 8*2.3= ~18.4 mW.

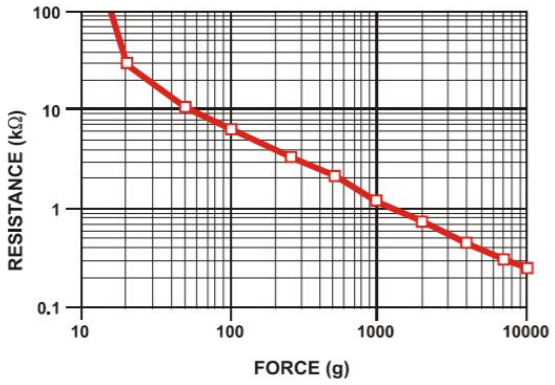

Figure 9: Resistance vs. Force for a Force Sensitive Resistor. Adapted from [11].

As seen in figure 10 below, we will take advantage of the fact that the force sensitive resistor varies in resistance. A constant 5V voltage input with a constant 10 kΩ resistor in series in a resistor divider circuit will allow changes in vout to be directly attributed to the change in resistance of the FSR. This output voltage will used as input to the next component: the analog multiplexer.

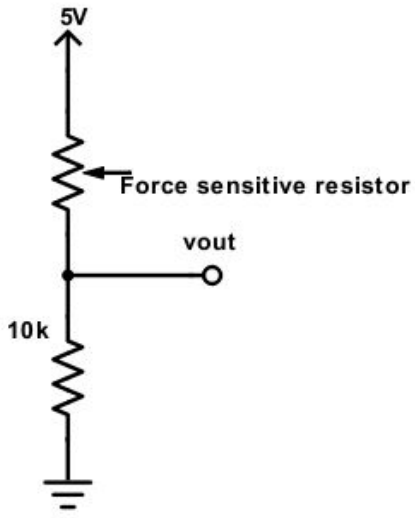

Figure 10: Force sensitive resistor schematic

Requirement	Verification	Points
1. When no pressure is applied, the voltage reading should be approximately 0 and the current should be less than 0.05 mA. 2. A neutral balanced sitting position with two sensors on both sides of the body should output voltages within 0.1V. 3. Varying the weight on each sensor from ~50g to ~1kg should result in a change in resistance from ~10kΩ to ~1kΩ 4. Sensor readings should be consistent within around 0.5V for a user who is sitting still in any position.	1. A. Use a DC generator to supply 5V as the input voltage. B. Measure the voltage and current between the resistor and the FSR using a voltmeter and ammeter. The result should be less than 0.05 mA current and about 0 V. 2. A. Place two sensors directly under the ischial tuberosity of the subject. B. Use a leveler to ensure the user is level and balanced. C. Measure the voltages to make sure the difference is within 0.1V. 3. A. Obtain a set of weights from 50g to 1kg B. Place the 50g weight on the pressure sensor and obtain the output. C. Next, add weights until ~1kg is reached. D. Measure and calculate the resistance using a multimeter and the obtained values should be around 10kΩ and 1kΩ. 4. A. Sit on a pressure sensor. B. Take an initial reading with a voltmeter and have the user continue to remain in the initial pose. C. Continue to take measurements every 20 seconds for 10 minutes, and check to make sure all values are close (within ~0.5V).	1. 0 2. 2 3. 0 4. 0

Table 3: Pressure Sensors RV Table

2.2.2. Distance sensor

Along with the measurements of pressure distribution around the seat and back, we will also want to know the distance between the back of the user and the back of the chair. This information will be useful in distinguishing the postures shown in figure 1. In particular, it helps detect the slump posture from the other acceptable ones. To do this, we will use an HC-SR04

Ultrasonic Sensor. This sensor emits 8 cycle bursts of 40kHz ultrasonic waves in response to a 10 microsecond trigger and outputs a high signal for the duration that it took to receive the reflected ultrasound [12]. As seen in figure 11, the distance sensor has four pins: two of them are the supply pins (supply voltage and ground) and the other ones are the trigger and echo. The trigger essentially toggles the output of ultrasonic waves, and echo outputs a high signal for the round trip duration that it took to receive the ultrasonic waves.

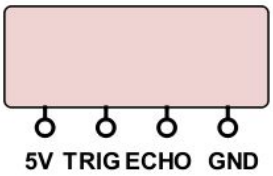

Figure 11: HC-SR04 pin diagram.

Since the waves travel at the speed of sound and they make a round trip, the distance can be measured using the equation below where 340 m/s is the speed of sound:

$$Distance = 340 * \frac{time}{2} \quad \text{Eq. 5}$$

Using the above numbers, we can calculate the time it takes to take a reading. Given how we don't expect to see distances over 30cm, we can calculate the maximum round trip time for the sound to propagate as 2*0.3/340 = 1.765 ms. Since 8 cycles of a 40kHz are emitted, the latency of that is 8 cycles/40,000 cycles/second = 200 microseconds. So adding the latencies together, the maximum latency expected for this sensor in this project is 10 microseconds + 200 microseconds + 1.765ms ≅ 2 ms

The ultrasonic sensor was chosen over alternatives such as infrared because of the ease of use, robustness in lighting conditions, and low cost. The ultrasonic sensor requires 5V as input and operates on 15mA of current [12]. So the power can be calculated using Eq. 6.

$$P = IV \quad \text{Eq. 6}$$

With V=5V and I = 15mA, the average power usage of the ultrasonic sensor is 75 mW.

Requirement	Verification	Points
1. Obtain distance values of between 2-30 cm that are accurate within 2 cm 2. Detect severe slump postures of around 45-60 degrees measured from the seat of the chair from distances 10-30 cm away when mounted on the chair. 3. The plot of time vs slump angle should be linearly proportional (the readings should not register out-of-range objects while slumping). 4. The latency of operation should be no more than 2ms+/-0.2ms	1. A. Hook the sensor up to a microcontroller such as an Arduino and place it at the base of a meter stick. B. Use a large textbook and measure values starting at 2 cm away up to 30 cm away. C. Every measurement should be accurate within 2 cm. 2. A. Mount the ultrasonic sensor on the top of the back of a chair. B. Have the user sit on the chair and measure an angle within 45-60 of the back using a protractor, and have the user sit 10-30 cm away. C. Measure the output value of the ultrasonic sensor from the microcontroller and ensure the values correspond to the selected distance values. 3. A. Using the same setup from requirement 2, the user starts at a neutral 90 degree upright position. B. Record data as the user slowly slumps over until 45 degrees is reached. C. Plot the data and determine a linear relationship between the distance and slump angle. 4. A. Position the ultrasonic sensor at the base of a meter stick facing a textbook 30 cm away. B. In the microcontroller code, insert timer statements to capture the time directly before the trigger and directly after receiving the echo. C. Run the code and make sure the latency is less than 2ms+/-0.2ms D. Repeat from step A 100 times for consistency.	1. 0 2. 3 3. 0 4. 0

Table 4: Distance Sensor RV Table

2.3. Control system

The control system takes the sensor data as input and presents it in a format to send the data to the Bluetooth Module in the data stream module. It will then also accept a response from the data stream to determine actions to output to the vibration motors in the user interface module.

2.3.1. Microcontroller

This is a key part of the project. Think of it like the spinal cord: it connects the information from the brain (software) to the sensors and actuators (sensors, vibration motors, etc.). The microcontroller will take the inputs, which are:

- 1 analog input from 8 pressure sensors pressure sensors
- 1 digital input from the distance sensor
- 2-byte Bluetooth input from the computer through the data stream

The outputs will be:

- 10-bit data for each sensor reading to the computer through Bluetooth
- 6 PWM voltages between 2.3-3.3V each to a vibration motor circuit
- 3 digital outputs to select the next pressure sensor to read from the analog mux

To accomplish the above specifications, an ATmega328P with 16Mhz frequency, 32KB flash memory, 2KB SRAM, 10-bit ADC, and 23 programmable I/O pins will be used [13]. The pins of interest are the analog inputs for the pressure sensors, digital input and output for the ultrasonic sensor, PWM pins to control the vibration motors, RX and TX pins for Bluetooth, and digital output pins for the analog multiplexer selection. It should be noted that the maximum current that a pin can output is 40 mA, so special can needs to be taken when powering the power-intensive vibration motors [13]. This microcontroller was selected over alternatives due to the popularity, resulting in abundant documentation, as well as familiarity and ease of programmability since it is used in an Arduino Uno.

The microcontroller mainly behaves as the central link between components and an interface between hardware and software. Not only should it be able to poll all the pressure sensors from the analog mux using digital output, but it also has to receive ultrasonic sensor readings, send and receive data through Bluetooth, and power the vibration motors. Since the microcontroller has a single core and no multithreading capabilities, these tasks must be quick and/or non-blocking. Figure 12 shows what the control flow of the microcontroller will look like. It will continuously poll data from the sensors and the Bluetooth module to decide what to do. Note that the data can be sent directly after reading it or altogether after all sensors have been read since the microcontroller is synchronous. Generally, being able to go through all the inputs and outputs of the microcontroller in several milliseconds is sufficient for the purpose of the project. Notice

in figure 12 that it's possible to not receive Bluetooth data in the current pass due to a slower data processing speed, so it will just check again in the next loop because the single-threaded nature of the microcontroller cannot afford to block the control flow just to receive data that may not arrive for a relatively long time.

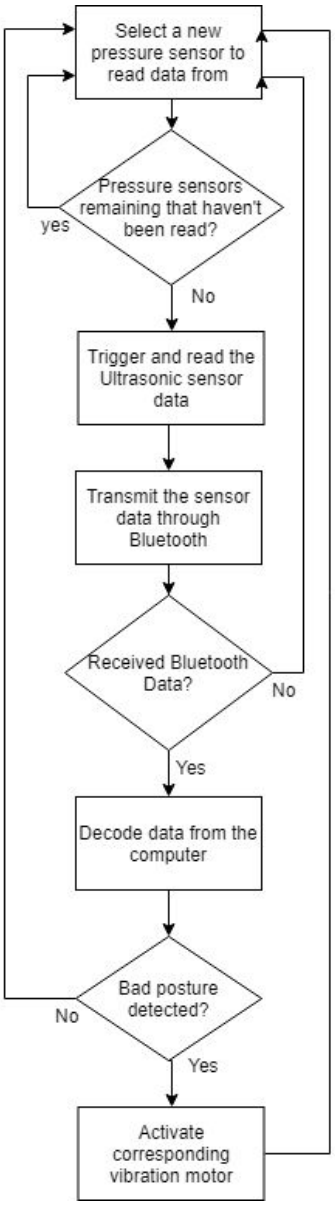

Figure 12: Microcontroller flowchart.

A primary source of concern is the heavy use of one 10-bit ADC within the microcontroller for all 8 of the pressure sensors. The 10-bit ADC takes, on average, 13 clock cycles to perform a conversion and uses a recommended 50kHz to 200kHz input clock frequency [13]. Assuming the input clock frequency is 125kHz, then 13 cycles / 125000 cycles/second = 104 microseconds per

conversion. 8 pressure sensors will result in a total conversion latency of 832 microseconds. This value is still under 1 ms, so it is incredibly fast to any user and not a big source of concern.

The typical current for a 5V input at 8MHz is 5.2mA [13]. So the power consumption of an active microcontroller is approximately 26 mW but this will vary significantly depending on usage.

Requirement	Verification	Points
1. Can provide a linear 2.3-3.3V signal using the PWM output pins while processing input signals and output less than 40 mA. 2. Use the 10 least significant bits of a received Bluetooth message as a mapping onto the 2.3V-3.3V range with a granularity of 1.0mV+/-0.2mV. 3. Send appropriate signals to the corresponding vibration motor if bad posture is detected in that region (extract one-hot coded vibration motor positions from the most significant bit of the received 2-byte computer data) 4. Loop through the sensor, Bluetooth, and vibration motor operations in less than 20 ms.	1. A. Write code to cycle through the motors in order using the one-hot encoded top 6 bits. B. Measure the output voltage and current of the PWM pin at different increments with a multimeter. C. Each step should result in a difference of about 1 mV with the lowest to highest values ranging from 2.3V to 3.3V. 2. A. Collect input data from sensors B. While the data is being collected, program the vibration motor to periodically take in a 2.3-3.3V PWM signal. C. Monitor the inputs to ensure the data has not been corrupted, and make sure the vibration is perceived as continuous as it passes through loops. 3. A. Apply heavy imbalance on the vibration motor designated as 000001. B. Check the received message for 000001 as the 6 most significant bits. C. Physically check that the motor intensity corresponds to the intensity designated in the remaining 10 bits. 4. A. Write code to check timestamps at the beginning and end of each loop. B. Run through the loop 100 and record the timestamps for each loop. C. Verify that the latency of each loop is within 20ms under different sensor inputs (different poses).	1. 0 2. 2 3. 2 4. 1

Table 5: Microcontroller RV Table

2.3.2. Analog Multiplexer

Due to a high number of pressure sensors needed, an analog multiplexer will be used to select the input to process by the 10-bit ADC. This component takes the analog outputs of the pressure sensors as input, then the microcontroller selects one of the sensors to read.

While an alternative is to use a microcontroller with more pins, using the analog multiplexer makes it easier to scale the number of sensors. Also, it's much easier and cheaper to obtain an analog mux than it is to find the appropriate microcontroller. If the need for 16 pressure sensors arises, the current 8-bit multiplexer can be replaced with a 16-bit multiplexer at the cost of an extra digital output pin (4 total to select each input). The number of sensors that can be read scales as $O(2^n)$ where n is the number of select bits/pins. However, keep in mind that the single-threaded microcontroller will require more time to poll a large pool of sensors.

The 74HC4051 will be used as the analog mux. This multiplexer has 8 analog inputs and also uses three bits to select one of the 8 inputs with a typical time of 15ns [14]. It takes in 5V from the supply and also has a current usage of about 2 micro-amps. Therefore, the power usage is negligible.

Requirement	Verification	Points
1. Power dissipation per input must be less than 100 mW. 2. Correct selection of inputs given corresponding selection signals	1. A. Apply the pressure sensor inputs into the mux. B. Measure the product of current and voltage with a multimeter to determine if the power dissipation is under 100 mW. 2. A. For each input of the mux starting from input 0, apply an input as the input number multiplied by 0.625V. (0V in input 0, 0.625V in input 1, 1.25 in input 2, etc.). B. Input each combination of the 3 select signals (000, 001, 010, etc.). C. For each selection, check that the output is the correct voltage value.	1. 1 2. 0

Table 6: Analog Multiplexer RV Table

2.4. Data stream

The Data Stream module is how the microcontroller will be communicating with the computer. The microcontroller will be sending the 10-bit converted sensor data for each sensor. Then, the computer will read it, process it, and it will return a course of action to the microcontroller (no op vs. activate vibration motor *x* with intensity *y*).

2.4.1. Bluetooth

We will be using a Bluetooth connection to transmit data from the chair to the computer to be processed. The Bluetooth module will also be receiving commands from the computer to vibrate as one form of notifying the user. The Bluetooth stack we chose is an embedded system approach, because we are implementing a Bluetooth peripheral device. The Bluetooth module we have picked out is the HC-05. It is a simple TX/RX pipeline, and uses 3.3V with a 3.3V regulator on board so we can test it with an Arduino or a similar 5V board [15].

The amount of current used by the HC-05 varies wildly, depending on the state. Initially, the module will be in pairing mode to attempt to establish a channel of communication with a nearby device. This pairing mode consumes at most 40 mA with an average of 25 mA [15]. After successfully pairing, it will begin communicating, so the usage drops to a stable 8 mA steady-state current [15]. Therefore, the worse case pairing power consumption is 3.3*40 = 132 mW which drops to 26.4 mW after pairing with a device. However, we must consider the fact that if the Bluetooth hasn't been paired, some components will not be able to function. Namely, the vibration motors will not be enabled. Peering ahead, we notice that the vibration motors have a steady-state power consumption of 240 mW with an average 3.0V input, which is significantly higher than the pairing power consumption. Also, the system is not expected to be in pairing mode for long periods of time since the system/chair is not expected to move out of range or pair with something else, so it will only be drawing 8 mA of current for the vast majority of the time. Therefore, the average power consumption will be considered ~26.4 mW.

The range is up to approximately 30 feet. The HC-05 also has a 2.4GHz frequency and 9600bps default transmission rate configurable up to 1,382,400 baud rate [15]. The module is expected to have a 100% accuracy unless the chair drifts too far from the computer (over 30 feet away).

To compute the transmission rate needed, we use the approximately 3 ms latency it takes to collect the sensor data where each sensor reading is sent as 10 bits. Therefore, The minimum transmission rate for a 3 ms loop period is 90/0.003 = 30,000 bps. As we can see, this required rate is over 3 times higher than the default rate. So to improve the Bluetooth transmission efficiency, the default baud rate can be used while limiting the sensor data collection to 10 ms per loop or, better yet, use a baud rate over 30,000 bps.

Requirement	Verification	Points
1. Must be capable of transmitting 90 bits every loop through the sensors where the loop time can be estimated as low as 3 ms. 2. Must be capable of pairing and maintaining a stable connection of sending and receiving data from a computer 6 feet away. 3. Must be able to transmit data with approximately 0% error.	1. A. Create a program that sends 90 bits of data through Bluetooth to a computer and record a timestamp. B. Create a timestamp on the receiving computer for each datum received. C. Compare the latency by finding the difference in timestamps. D. Repeat step A with a 1.3M baud rate and also with a 4800 baud rate. E. The bandwidth is congested if the transmission time is significantly slower (up to 0.5 seconds slower for heavy traffic). 2. A. Use a tape measure to position the Bluetooth module 6 feet from the computer. B. Send 10-bit sensor data to the computer. C. Check that the data was received on the computer and send an acknowledgement to activate a vibration motor. D. Use a multimeter around the output pin check that a signal is created in response to the received message. 3. A. Randomly generate 1000 10-bit values from the microcontroller. B. Transmit the 1000 value sequentially to the computer through the Bluetooth module. C. The computer will then transmit the data back after receiving it. D. When the microcontroller receives data, compare it to the expected value. E. This requirement is satisfied when only 1000 values are received and correct.	1. 2 2. 0 3. 0

Table 7: Bluetooth RV Table

2.5. User interface

The user interface is how the system communicates with the user. The communication media can be through the vibration motors when bad posture is detected or through the visualization and recommendation program from the software. We will attempt to highlight areas that are at risk of injury due to poor posture. This data will be specific to the user since it learns the posture trends of the user. The user interface may eventually be implemented through a website, but this may be replaced with a program that creates a GUI locally. That is where all the necessary calculations can take place and it's easy to access and communicate with the user.

2.5.1. Vibration motors

We will use 6 C1034B018F vibration motors mounted in various regions near their corresponding pressure sensors. These vibration motors operate on 2.7~3.3V with a starting voltage of 2.3V, which scales linearly with intensity [16]. They will be toggled individually by the microcontroller, occupying all 6 of the available PWM pins. We elected to occupy all the PWM pins because the motors have a maximum steady-state current draw of ~80 mA, which is about double the ~40 mA maximum current output of a digital pin of the ATmega328p [16], [13]. Each motor has a rated speed, current, voltage, and noise of ~9000 rpm, ~80mA, 3.0V, and ~50dB, respectively [16]. These values show that a lot of power is used to generate mechanical movement that translates to a noticeable sound, which is ideal for the purpose of notifying the user. The power usage at the rated settings is approximately 240 mW.

Since the vibration motors are DC motors with an inductor element inside, we must be cautious about the properties associated with that. In particular, we should that inductors produce back EMF in response to a change in current. Consider equation 7 below where L is the inductance, dI/dt is the change in current across the inductor with respect to time, and V is the voltage across the inductor:

$$V = L\frac{dI}{dt} \qquad \text{Eq. 7}$$

When a motor is at steady-state, the change in current is small or 0 so the voltage is essentially 0. When the power is switched off, the change in current becomes negative. This will result in a negative voltage which means the inductor is now supplying current proportional to the change in current from the source. This can result in catastrophic voltage levels that may damage components in different parts of the circuit. To remedy this issue, we place the vibration motors in a circuit shown in figure 13 below. The parallel reverse-biased (also known as a "flyback") diode is used to absorb all of the back emf from the motor when the motor is turning off. The

NPN transistor allows the microcontroller to supply a low current digital output that will connect the motor to ground and turn it on.

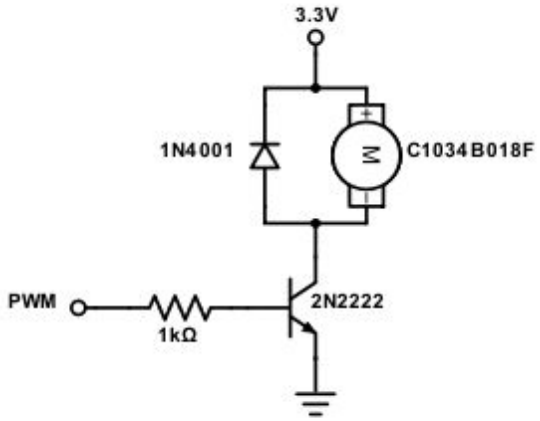

Figure 13: Vibration motor circuit. Adapted from [24].

While we have been able to devise a solution for the back emf with figure 10, there's another problem with DC motors: inrush current. This issue is another consequence of inductors and motors in general where a motor at rest will require a large amount of current because of several properties mentioned above related to equation 7. Basically, the inrush current will be very high but will eventually stabilize. There are solutions to combat this such as current limiters and large capacitors. However, for the sake of simplicity, we will be using the method of ramping up the effective voltage through PWM. With PWM, we can slowly raise the voltage during the inrush current phase until steady state so that the inrush current will not be dangerously high. Starting the PWM ramp from ~2.3V will result in a manageable ~120 mA current draw [16]. Therefore, the startup current power consumption of approximately 276 mW. This power consumption is close to the steady-state value of ~240 mW, so we will say that the average usage is approximately ~240 mW.

Requirement	Verification	Points
1. Must be noticeable under a thin cushion when operating on 3.0V+/-0.3V. 2. The current draw of the motor must not exceed 120 mA when applying a ramped PWM signal. 3. Must be spaced at least 2-3 inches away from the nearest pressure sensor to avoid interference.	1. A. Connect the vibration motor to a 3.3V source as in figure 10 above. B. Place the device under a cushion and determine if the vibrations can be felt. 2. A. Configure the circuit in figure 10. B. Starting with a ~2.3V PWM signal, slowly increase the voltage to ~3.0V. C. Measure current with an ammeter throughout and make sure the current never exceeds 120 mA. 3. A. Obtain a ruler and mark 2-3 inch radiuses around each pressure sensor. B. No motors should be inside the marked areas. C. Move the remaining motors near their designated locations that do not intersect the off-bounds territory.	1. 2 2. 1 3. 1

Table 8: Vibration Motors RV Table

2.5.2. Posture Analysis GUI (software)

The posture analysis is a form of digital feedback for the user to view. This will include a graph of posture trends and movement to track progress. It will also notify the user about reasons the vibration motors were triggered e.g. too much weight was shifted onto the right thigh. Similar to the applications that log activity details, this is designed to allow the user to gain a better understanding of the ubiquity of posture issues and what to do to mitigate those issues. As an additional feature that will be discussed in greater detail in the tolerance analysis section, there will also be a “calibration” button on the GUI for the user to toggle. This button will allow the user to customize the interpretation of sensor readings to conform to a more ideal posture. This can be thought of as a feedback loop in software.

Requirement	Verification	Points
1. Can display messages and pressure maps on the user's computer within 100 ms of the readings being detected. 2. Runs on a separate thread from the main data processing thread and utilizes approximately 0% CPU when no change is detected from the previous readings. 3. Can communicate with the data processing program/process/thread to provide user interactions (calibration button, viewing different timescales, etc.) with little latency (no more than 100+/-10 ms).	1. A. Gather sensor data using the pressure sensors and distance sensor. B. Observe that the plots are updating in real-time. C. Check that the visualization corresponds to the actions that resulted in the data in real-time. 2. A. Take similar, constant data with the same results (good posture) as input. B. Use a visual profiler software to get statistics on the CPU utilization (Visual Studio can do this). C. The thread is sleeping when the CPU utilization of that thread is about 0%. Use a device with a lower hardware thread count to make it easier to determine. 3. A. Create another program that can receive data from the GUI. B. Display the commands received from the GUI and time the latency. C. Compare the received results with the actions and confirm the low latency.	1. 4 2. 2 3. 4

Table 9: Posture Analysis GUI (Software) RV Table

2.6. Software

The software involves programs that collect and analyze the sensor data from Bluetooth. This will be the brains of the operation where good vs bad posture will be classified.

2.6.1. Data Processor

Data from 8 Force Sensitive Resistors and an Ultrasonic sensor must be received in real-time. Good and bad posture will be used to classify the data input. Recommendations will be generated based on the type of bad posture that was detected. Due to the variety of opinions on what is the ideal posture, the best approach is to accept the main ideas behind some positions such as a flat upper back or slight lordosis on the lower back accompanied with lumbar support [7].

This component will use a supervised learning classifier to examine the 9 inputs as features. This model will be responsible for the binary classification of good or bad posture. Since we're not able to find the data we need to train such a classifier, we will have to collect our own data to train the classifier. Therefore, the machine learning algorithm we use must be robust with a relatively small training set.

Our main posture classification algorithm will be implemented as a Support Vector Machine (SVM). We chose to use an SVM because it is reliable for small training sets and works nicely for binary classification (and regression) [20]. Therefore, we can just use each sensor's reading as a feature and, depending on the separability of the data, perform kernel tricks to deduce decision boundaries. Due to the lack of relevant, available posture data, we will be collecting our own posture data for the training set. As with all supervised learning, we must be as thorough in providing data for many cases/postures while being wary of the potential to overfit the classifier. Refer to figures 1 and 2 for an idea of what the training set may consist of.

Figure 14 shows the control flow of the data processor. As we can see, it waits until it receives the 10-bit sensor data from each sensor. Then, it will use the supervised model to classify if the posture is good or bad. While it's tempting to implement the SVM models from scratch, we will instead be tuning the SVM classifier to best fit the data while avoiding overfitting from the relatively training set (50-100 samples). The number and variety of samples will be selected such that a validation set can achieve at least an 85% accuracy as desired from the high-level requirements. The SVM we will be using will be from Sci-kit Learn using Python.

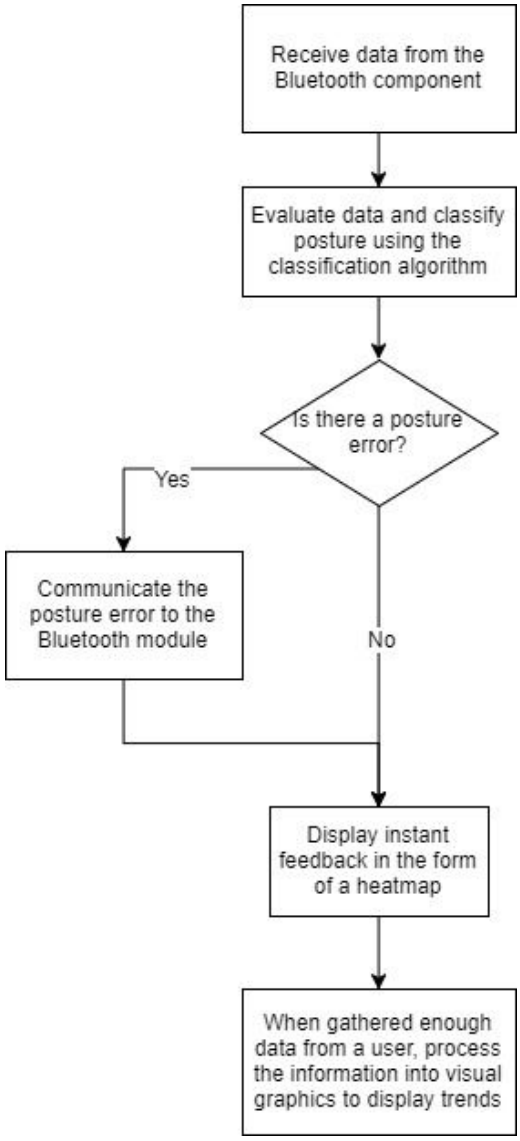

Figure 14: Data processor flowchart.

While SVMs are ideal for our case, it's important to understand the complexity of the algorithm. Since SVM is a quadratic programming problem, the implementation in Scikit Learn takes between $O(n_{features} \times n^2_{samples})$ and $O(n_{features} \times n^3_{samples})$ time [22]. Since each sensor will be used as a feature, $n_{features}$ will be 9 and $n_{samples}$ will be around 50-100, depending on how exhaustive we decide to make our training set. Any more data in the training set will be too time-consuming. The space requirements also scale at a similar rate, but that ultimately depends on the data we're using. Due to the scale invariance of Support Vector Machine algorithms, it is highly recommended to scale/standardize the data with mean 0 and variance 1 [22]. In particular with our dataset, we need to scale the pressure sensor readings to a baseline average of all the current

pressure sensors. This is needed because not all users will exert the same force on the sensors, so using residuals will allow the data to be generalizable to many user types.

Since we are doing binary classification to determine if a posture is good or not, we need to figure out which motor to send the vibration signal to. Given an exhaustive training set covering most cases on each sensor, we can identify which vibration motor to activate by applying the data to a nearest centroid classifier. The centroids are calculated in equation 8 where x is the vector of feature data per sample and C_l is the set of indices belonging to class *l* [23]:

$$\overrightarrow{\mu_l} = \frac{1}{|C_l|}\sum_{i \in C_l} \vec{x}_i \qquad \text{Eq. 8 [23]}$$

After assigning the centroids, the prediction will use equation 9 below to select which centroid is most similar (closest centroid to the current datapoint):

$$\hat{y} = \arg\min_{l \in \mathbf{Y}} \|\vec{\mu}_l - \vec{x}\| \qquad \text{Eq. 9 [23]}$$

The outcome y is then encoded into a two byte packet along with the magnitude of the largest residual to determine which vibration motor and how intense. This packet will be sent to the microcontroller for processing.

One might wonder why it's necessary to use two classifiers instead of combining both into, say, a multiclass SVM. This is a possibility, but SVMs are more natural as binary classifiers and since further action is only taken when a bad posture is detected for a certain amount of time, the nearest centroid classifier is not always used. Therefore, splitting the classifiers up allows for more efficiency and customizability. The nearest centroid classifier can always be scrapped or replaced if deemed too inaccurate, whereas having it built into the binary posture classifier may require major restructuring in sensor data. This design decision was inspired by a modular approach.

This component will also keep track of a vector of weights to apply to the sensor readings. We will expand more on this calibration mechanism in section 2.8 Tolerance Analysis.

Requirement	Verification	Points
1. The supervised learning model should be trained with at least 50 samples 2. The classification accuracy should be at least 85% 3. Should be capable of generating visual feedback of heatmaps of at-risk positions on the body along with a graph of posture trends over time. 4. The classification time should not take more than 500 milliseconds. The preferred classification latency is approximately 0-100ms	1. A. A poorly trained model cannot accurately classify data that are slightly unusual, so provide various, distinct poses. B. Record the posture data with the pressure sensors and distance sensor. C. Monitor the output from the software. D. Observe any discrepancies from the nearest centroid decision and expected motor output. 2. A. Create a substantial test set of sensor data (at least 30 samples). B. Apply the model to the samples and observe the accuracy. It has to correctly classify if a posture is good or bad at least 85% of the time. 3. A. Apply an uneven amount of pressure on one sensor over time (every 10 minutes for an hour). B. Observe the provided trend and hot spots. 4. A. Acquire data in the data processing program. B. Classify the data by using the supervised learning model. C. Time how long it took to receive a classification and compare it to the requirement numbers.	1. 5 2. 5 3. 5 4. 5

Table 10: Software RV Table

2.7. Circuit Schematic

Figure 12 shows the schematic interaction between the sensors, the analog mux, the microcontroller, the Bluetooth module, and the vibration motors. The 5V signals come from the 5V voltage regulator and the 3.3V signal comes from the 3.3V voltage regulator. While the Force Sensitive Resistors and Vibration Motors appear in the schematic, they will actually be mounted around the seat and back of the chair as illustrated in figure 4.

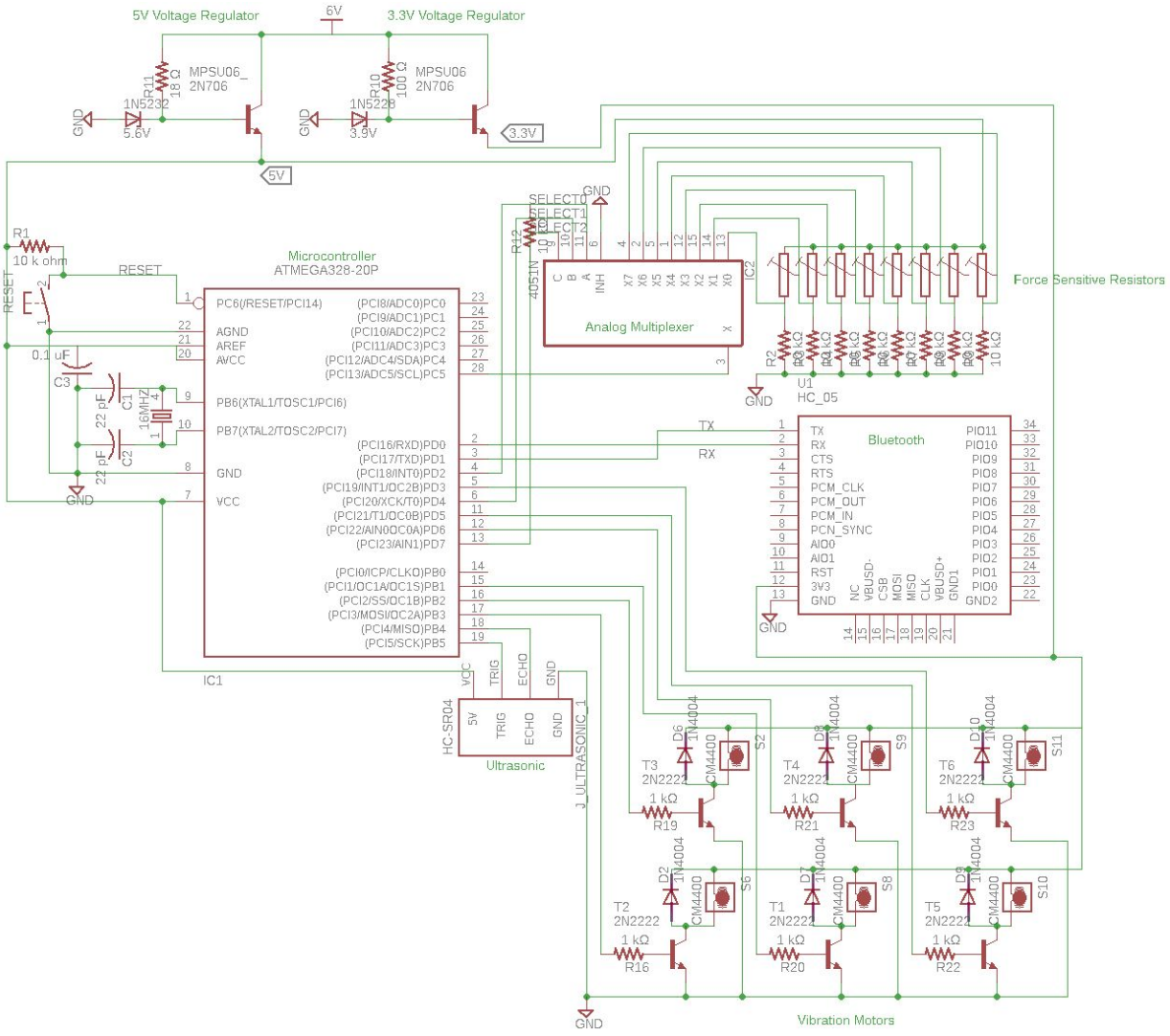

Figure 12: Circuit schematic.

2.8. Tolerance Analysis

The most critical overall features of this project are the sensor readings that provide data to infer posture. These sensors include the pressure data calculated from the pressure sensors as well as the slump distance recorded by the distance sensor. These readings are key to identifying the posture of the user because they will be sent to the microcontroller and then sent to the computer to be processed by the classification algorithms in the software module. The proper classification of the posture depends entirely on the integrity of the data. No matter how good the model is or how efficient the entire system is, inaccurate sensors will compromise the entire system. In the ethics and safety section of this document, we explain the detrimental effects this may have on the user.

2.8.1. Ultrasonic Sensor

We first consider the HC-SR04 ultrasonic sensor. The datasheet of the HC-SR04 states that the sensor has a minimum range of 2 cm and measuring angle of 15° [12]. Because we do not anticipate to detect over a 30 cm range for this project, we will not worry how far it can go distance-wise as long as we leave a 2 cm distance from the back of the chair. We will, however, consider the radius of coverage that is necessary to guarantee accurate readings. Using simple trigonometry, we obtain the following equation for the height:

$$height = distance * tan(angle_{elevation}) \quad \text{Eq. 10}$$

For a distance of 30 cm and an angle of elevation of 15°, we get a maximum radius of about 8.04 cm. This means that the user should be over 8 cm taller than the height of the chair (where the sensor is mounted) when positioned a distance of 30 cm away. Since we are in a three dimensional world, this also applies to the width. The width can be an issue if the user is not properly centered. In this case, the readings are inevitable and rely on the 4 m maximum range. Outside of this range, the values may be non-deterministic. Therefore, the pressure sensors and classification algorithms must be relied on to help detect extremely poor posture when the user is not in view of the ultrasonic sensor.

2.8.2. Force Sensitive Resistors

When considering the pressure sensors, it's important to note that Force Sensitive Resistors are not very accurate. The FSRs can have accuracy ranges with a tolerance of ±5% to ±25% [11]. While these values can be wildly inaccurate, [8] and [9] both utilize them due to their ability to be fitted onto a chair as well as the sufficient readings. While the accuracy is horrible, the FSRs have a redeeming quality in that the force resolution has a tolerance of ±0.5% of the full use force [11]. This means that the readings might not represent the actual force, but the granularity is very high. Therefore, we must counter this inaccuracy through means of calibration.

2.8.3. Calibration

To calibrate the sensors, we introduce the concept of the perceptron weight update rule. Perceptrons use weights **w** to classify data. They can be thought of as simplified neural networks. However, we are only interested in how the perceptron weights are updated instead of classifying things. We observe the perceptron weight update rule in equation 11 below:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\left(y - y'\right)\mathbf{x} \quad \text{Eq. 11 [21]}$$

The weights **w** are updated with a portion of the input **x** with a learning rate $\boldsymbol{\alpha}$. This basically adds a component of the input so that the new weights are more inclined to accept future similar

inputs. In our case, we want to weigh the sensor readings in the software component to accept more similar values. As mentioned in the posture analysis GUI and data processor component sections, this calibration occurs in software. The user will manually indicate that the current posture should be learned. Then, the software will update the weights with the current sensor readings. The learning rate will be determined through trial and error, but should, by convention, be less than 1. This mechanism will introduce an overhead of vector multiplication that scales linearly with the number of sensors (initially all weights of 1).

This tolerance calibration mechanism introduces many new features that make the product more generalizable and customizable. Consider the case of a user with a disproportionate physical figure e.g. amputee. The sensors will initially detect a weight imbalance, but if the user calibrates the weights according to a posture that suits the user, then it will eventually learn the optimal weights for proper posture classification. In addition, the potential accuracy offset can be filtered out in a similar fashion where the user conforms to a good posture and updates the weights until the classification algorithm provides satisfactory results. Since we are not worried about optimal paths or overfitting in this case (remember, we're not using the "perceptron" as a classifier), a decay in learning rate is not necessary since new inputs can be just as relevant as previous inputs.

3 Costs

Assuming each partner makes $40/hour and spends 10 hours a week on the project, we can model the labor cost below:

$$Labor\ cost\ = 3 \cdot \frac{\$40}{hour} \cdot \frac{10\ hours}{week} \cdot 16\ weeks \cdot 2.5 = \$48,000 \qquad \text{Eq. 12}$$

Table 11 below displays the cost of the non-standard components we need to purchase. As seen in the schematic, we will also need capacitors, a crystal, diodes, wires, and resistors. However, these standard components can all be obtained at no cost for our project. Therefore, we will not be including them in this section since they are not relevant at all to the cost.

Description	Manufacturer	Part #	Quantity	Unit Cost ($)	Total Cost ($)
4-pack AA Batteries	Energizer	39800011329	1	3.59	3.59
Battery Holder, 4AA with molex connector	Keystone Electronics	2478CN	1	2.80	2.80
Force Sensitive Resistor	Sparkfun Electronics	SEN-09376	8	9.95	79.60
Ultrasonic Sensor HC-SR04	Sparkfun Electronics	SEN-13959	1	3.95	3.95
ATmega328p	Atmel	DEV-10524	1	9.49	9.49
8-input Analog Multiplexer	ON Semiconductor	74HC4051	1	4.00	4.00
Bluetooth Module	Electronica 60 Norte	HC-05	1	6.60	6.60
Vibration Motors	Jinlong Machinery & Electronics	C1034B018F	6	2.88	17.28

Table 11: Part costs.

Part total: $123.36. The total cost is the labor + part cost which is $48,123.36. This does not account for shipping, which depends on the time sensitivity of the part's arrival and can account for a substantial portion of the actual cost.

4 Schedule

Week	Alex	Pablo	Emre
2/26/18	Order and test parts needed for the project	Get parts needed for analog parts of project	Complete the algorithm designs.
3/5/18	Prototype the microcontroller and write code to interact with the sensors, Bluetooth, and the vibration motors	Build voltage regulators and test them	Help write code for the sensors and Bluetooth.
3/12/18	Design and order the PCB for the microcontroller using updated schematics	Using voltage regulators, test pressure sensors on a chair	Create visual feedback system for user. (Website?)
3/19/18	Plan and create posture training data	Test distance sensor in different circumstances (light, interference…)	Collect many samples for all the postures to be used for classification, and reevaluate algorithms.
3/26/18	Update the final PCB design and write code for the data pipeline between the microcontroller and the computer	Test vibration motors on the chair.	Complete the software components.
4/2/18	Tune the posture classification model and introduce more varied sensor data (different positions or people) to prevent overfitting	Using battery and voltage regulators, connect all parts and test they work altogether	Improve posture classification data and tune the algorithms accordingly
4/9/18	Identify bottlenecks and optimize the flow of data.	Connect analog parts to microcontroller	Optimize the flow of data
4/16/18	Design the physical placement/layout of components on and around the chair	Test complete system (without final assembly)	Test software functionality and get tester feedback
4/23/18	Gather data on the behavior of the system (graphs, error, performance, etc.)	Final assembly	Demonstrate functionality on a test group.
4/30/18	Test all functionality of the system together.	Final test	Final test

Table 12: Project implementation schedule

5 Ethics and safety

There is a certain risk in this project, which must be handled: the risk of having an electrical system at a place in which a human being is going to sit.

The requirement to handle this problem is using enough isolation measures, such as using insulating materials or putting critical components at places which are not too close to the user. There could be around 100 mA of current flowing around the circuit, so protective measures must be taken to avoid potential injury as a result of contact. Also, there are many components spaced out around the chair. Naturally, there will also be a lot of wires and cables to handle properly so that the user doesn't get hurt by accidentally tampering with them. The primary solution for this is to make the design as seamless as possible and keep things organized.

However, it is impossible to move away from the user all the components. For example, the sensors need to be near the user, and in fact, the pressure sensors will be in contact with him or her. These pressure sensors are the most critical part of the system, so an insulating material will be put between the sensors and the user. This material must be strong enough to electrically isolate the sensors and the user, but at the same time it must be soft, so when the user sits on the chair the pressure sensors will be pressed, despite the fact that there is an insulating material between them.

Other risks concern the physical assembly of the chair. However, we do not have the intention of making the chair step by step, but using one to design the electrical system. This means that the assembly of the chair should not be a problem for this project.

This aligns with the IEEE code of ethics. To be more specific, it follows point 1 of IEEE code of ethics: to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment [5]. Our project may be useful to improve the people's health, which is obviously an ethical action.

For instance, if we follow a duty-based ethical theory, such as Kant's deontology theory, it is obviously ethical, because this theory states that an action is good if people would desire that it would be an universal rule. I think that everybody agrees that helping people to improve their health is something that is desirable for everybody, so it is ethical. However, as indicated by IEEE Code of Ethics #3, we must be honest about our claims about what our data represents [5]. This system is intended to improve the posture of the user, but we cannot claim to diagnose or treat potentially serious health issues related to posture. In fact, we'll need to inform the public using disclaimers to see more personalized experts such as their physician or more proven techniques.

Based on our high-level requirement of at least an 85% classification accuracy, it's important to consider that no machine learning algorithm is perfect and fully generalizable to all cases. This could result in odd cases where a bad posture is ignored or, even worse, a good posture is is labeled as bad. This addresses the issue of IEEE Code of Ethics #9, where we must avoid injuring users with false knowledge [5]. We included some measures to consider the user's actions such as through the use of perceptron weights to calibrate the sensor readings to the user's desired specifications. Also, our classifier will be made as accurate as possible with an exhaustive training set.

Due to the collection and analysis of data, it's also important not to use the user's personal data in wrong ways as highlighted in IEEE Code of Ethics #2: "to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist." [5] For example, a conflict of interest could occur if we were to sell posture data to clinics or other companies selling posture tools. Of course, this can be avoided by only using the collected data for the purpose of informing the user. This implication also applies to #5 where bribery should be rejected by not selling user data without their permission.

References

[1] Acatoday.org, "Back Pain Facts and Statistics." [Online]. Available: https://www.acatoday.org/Patients/Health-Wellness-Information/Back-Pain-Facts-and-Statistics. [Accessed: 08-Feb-2018].

[2] Acatoday.org, "Tips to Maintain Good Posture," *Posture*. [Online]. Available: https://acatoday.org/content/posture-power-how-to-correct-your-body-alignment. [Accessed: 08-Feb-2018].

[3] D. Brown, "Experts say posture matters: The good ... and the bad," *GoUpstate*, 10-May-2010. [Online]. Available: http://www.goupstate.com/news/20100511/experts-say-posture-matters-the-good--and-the-bad. [Accessed: 08-Feb-2018].

[4] J. Flynn, "Is it possible to cure a bad posture?," *The Independent*, 26-Oct-2015. [Online]. Available: http://www.independent.co.uk/life-style/health-and-families/features/is-it-possible-to-cure-a-bad-posture-a6709781.html. [Accessed: 08-Feb-2018].

[5] Ieee.org, "IEEE IEEE Code of Ethics", 2018. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 08-Feb-2018].

[6] "UNITED STATES DEPARTMENT OF LABOR," *Occupational Safety and Health Administration*. [Online]. Available: https://www.osha.gov/SLTC/etools/computerworkstations/positions.html. [Accessed: 20-Feb-2018].

[7] A. Claus, J. Hides, G. Moseley, and P. Hodges, "Is 'ideal' sitting posture real?: Measurement of spinal curves in four sitting postures," Manual Therapy, vol. 14, no. 4, pp. 404-408, June 2008. [Online]. Available: www.sciencedirect.com. [Accessed Feb. 20, 2018].

[8] Y. Zheng and J. Morrell, "A vibrotactile feedback approach to posture guidance" in *Haptics Symposium*, *2010 IEEE, 25-26 March 2010* [Online]. Available: IEEE Xplore, www.ieee.org. [Accessed 20-Feb-2018].

[9] L. Martins, R. Lucena, J. Belo, M. Santos, C. Quaresma, A. Jesus, and P. Vieira, "Intelligent Chair Sensor Classification of Sitting Posture," *Communications in Computer and Information* Science, *2013.* [Online]. Available: www.researchgate.com. [20-Feb-2018].

[10] J. J. Swearingen, C. D. Wheelwright, J. D. Garner, "An Analysis of Sitting Areas and Pressures of Man," Civil and Medical Research Institute, Oklahoma City, Oklahoma, 1962. [Online]. Available: www.faa.org. [Accessed 22-Feb-2018].

[11] Interlink Electronics, Inc., "Force Sensing Resistor Integration Guide and Evaluation Parts Catalog." [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/Pressure/fsrguide.pdf. [Accessed 22-Feb-2018].

[12] ElecFreaks, “Ultrasonic Ranging Module HC-SR04.” [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. [Accessed 22-Feb-2018].

[13] Atmel, “8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash,” 2009. [Online]. Available: https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf. [Accessed 22-Feb-2018].

[14] NXP Semiconductors, “74HC4051; 74HCT4051 8-channel analog multiplexer/demultiplexer,” 2016. [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/5/5/3/74HC_HCT4051.pdf. [Accessed 22-Feb-2018].

[15] "HC Serial Bluetooth Products User Instructional Manual", Cdn.makezine.com, 2018. [Online]. Available: https://cdn.makezine.com/uploads/2014/03/hc_hc-05-user-instructions-bluetooth.pdf. [Accessed: 22-Feb-2018].

[16] Jinglong Machinery & Electronics, Inc., “Coin type vibration motor,” 2007. [Online] Available:http://www.vibration-motor.com/products/download/C1034B018F.pdf. [Accessed: 05-Mar-2018].

[17] Energizer Brands, LLC, “Energizer E91 Product Datasheet.” [Online]. Available: http://data.energizer.com/pdfs/e91.pdf. [Accessed: 22- Feb- 2018].

[18] Vishay Semiconductors, “1N5221 - 1N526 Small Signal Zener Diodes,” 2008. [Online]. Available: http://www.vishay.com/docs/85588/1n5221.pdf. [Accessed: 22- Feb-2018].

[19] Motorola, “Bipolar Power Transistor Selector Guide & Cross Reference.” [Online]. Available: http://www.datasheets360.com/pdf/7456842635131963505. [Accessed: 22-Feb- 2018].

[20] S. Lazebnik, “Support Vector Machines,” 2017. [Online]. Available: http://slazebni.cs.illinois.edu/fall17/lec17_svm.pdf. [Accessed: 22- Feb- 2018].

[21] S. Lazebnik, “Neural Networks,” 2017. [Online]. Available: http://slazebni.cs.illinois.edu/fall17/lec18_neural_nets.pdf. [Accessed: 22- Feb- 2018].

[22] “1.4. Support Vector Machines,” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/svm.html. [Accessed: 26-Feb-2018].

[23] “Nearest centroid classifier,” *Wikipedia*, 17-Feb-2018. [Online]. Available: https://en.wikipedia.org/wiki/Nearest_centroid_classifier. [Accessed: 26-Feb-2018].

[24] *How to Build a Vibration Motor Circuit*. [Online]. Available: http://learningaboutelectronics.com/Articles/Vibration-motor-circuit.php. [Accessed: 05-Mar-2018].