ARTIFICIAL SKIN TACTILE SENSOR
FOR PROSTHETIC AND ROBOTIC
APPLICATIONS

A Thesis
presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Mechanical Engineering
by
Ross James Miller
December 2010
COMMITTEE MEMBERSHIP

TITLE: ARTIFICIAL SKIN TACTILE SENSOR FOR PROSTHETIC AND ROBOTIC APPLICATIONS

AUTHOR: ROSS JAMES MILLER

DATE SUBMITTED: DECEMBER 2010

COMMITTEE CHAIR: DR. SAEED NIKU, PROFESSOR OF MECHANICAL ENGINEERING

COMMITTEE MEMBER: DR. WILLIAM R. MURRAY, PROFESSOR OF MECHANICAL ENGINEERING

COMMITTEE MEMBER: DR. JOHN RIDGELY, PROFESSOR OF MECHANICAL ENGINEERING
ABSTRACT

Artificial Skin Tactile Sensor for Prosthetic and Robotic Applications

Ross James Miller

To solve the problem of limited tactile sensing in humanoid robotics as well as provide for future planned mechanical prostheses, an innovative tactile sensor system was created and embedded into two realistic-looking artificial skin gloves. These artificial skin tactile sensors used small piezoelectric ceramic disks to measure applied force at multiple points on each glove. The gloves were created using silicone rubber to simulate both the texture and look of human skin, while maintaining both flexibility and durability. The sensor outputs were buffered by high-impedance voltage-following operational amplifiers, and then read sequentially using a multiplexing scheme by a microcontroller. Sensor data were sent via USB to a computer, where a graphical user display was created to show the tactile information in real time. These prototypes successfully demonstrated the viability of small piezoelectric elements embedded in silicone rubber for use in creating flexible and elastic tactile sensors.

Keywords: Tactile, Sensor, Robot, Prosthesis, Piezoelectric, Piezo
Acknowledgments and Thanks

Special thanks to:

Dr. Saeed Niku, my thesis advisor, whose guidance and direction made this project possible

Dr. John Ridgely and Dr. William R. Murray, members of my thesis committee, whose knowledge and aid helped in overcoming several technical hurdles

QL+ and its members, for funding this project and freely lending their knowledge of the application of realistic-looking silicone rubber, which gave this project its visual elegance

This project was also supported by the Cal Poly Mechanical Engineering Department’s Bentley Center for Innovative Research
# Table of Contents

List of Tables ........................................................................................................... viii  
List of Figures ......................................................................................................... ix  

1. INTRODUCTION ................................................................................................. 1  
   1.1 Statement of Problem .................................................................................. 1  
   1.2 Background .................................................................................................. 1  
      1.2.1 Damaged Human Sense of Touch ....................................................... 2  
      1.2.2 Prostheses ......................................................................................... 3  
      1.2.3 Interfacing Techniques ..................................................................... 11  
      1.2.4 Humanoid Robotics .......................................................................... 16  
      1.2.5 Force Sensors .................................................................................... 20  
      1.2.6 Op-Amps ............................................................................................ 23  
      1.2.7 Multiplexers ....................................................................................... 25  

2. Literature Review ............................................................................................... 27  
   2.1 Current Tactile Sensor Technologies ........................................................... 27  
   2.2 Research on Tactile Sensor Technologies .................................................. 29  
      2.2.1 Understanding the Human Sense of Touch ....................................... 29  
      2.2.2 MEMS ................................................................................................ 31  
      2.2.3 Photo-reflective ............................................................................... 34  
      2.2.4 Takao Someya Group Research ......................................................... 34  
      2.2.5 Revolutionizing Prosthetics ................................................................. 35  
      2.2.6 Japanese Rubber Tactile Sheet ......................................................... 36  
      2.2.7 Shadow Hand ................................................................................... 37  
      2.2.8 Capacitive Flexible PCB ................................................................. 38  

3. Need for Further Research .................................................................................. 40  
   3.1 Design ........................................................................................................ 40  
      3.1.1 Sensor Design and Choice ................................................................. 40  
      3.1.2 Creating Realistic Skin Substrate ....................................................... 45  
      3.1.4 Multiplexer Sampling ....................................................................... 45  
      3.1.5 Arduino Microcontroller ................................................................. 46  
      3.1.6 Sampling Design ............................................................................... 46  
      3.1.7 Preliminary Testing and Design Modifications ............................... 47  
      3.1.8 Complete Design ............................................................................. 53  
   3.2 Manufacturing .............................................................................................. 54  
      3.2.1 Sensor Manufacturing and Quality Testing ..................................... 54
3.2.2 Silicone Rubber Molds .................................................................................. 58
3.2.3 Embedding Sensors ....................................................................................... 59
3.2.4 Electronics ..................................................................................................... 62
3.2.5 Programming Arduino Microcontroller ....................................................... 63
3.2.6 Programming Computer Graphical Display .................................................. 64
3.3 Testing and Results .......................................................................................... 66
3.4 Further Testing and Research Required .......................................................... 81
3.5 Conclusion ......................................................................................................... 82
Bibliography ............................................................................................................. 84
Appendices .................................................................................................................. 88
A. Additional Information about Piezoelectricity ................................................... 88
B. Arduino Code ....................................................................................................... 93
  1. Thumb_Prototype ............................................................................................... 93
  2. Hand_Prototype ................................................................................................. 96
C. Processing Code .................................................................................................. 103
  1. Thumb_Prototype ............................................................................................ 103
  2. Hand_Prototype ............................................................................................... 110
List of Tables

Table 1. Truth Table for a 2-to-1 Multiplexer ................................................................. 26
Table 2. Pair-wise Comparison of Designs ................................................................. 43
Table 3. Results of Pair-wise Comparison................................................................. 43
Table 4. Decision Matrix ......................................................................................... 44
Table 5. Sensor Sweep Rate .................................................................................... 70
List of Figures

Figure 1. Drawing of Human Skin and its Parts [35] ................................................................. 2
Figure 2. Iron Prosthetic Hand Worn circa 1508 [12] ................................................................. 3
Figure 3. Cosmetic Finger Prosthesis [15] .................................................................................. 4
Figure 4. Ear Prosthesis [15] ....................................................................................................... 5
Figure 5. Cosmetic Facial Prosthesis [15] ................................................................................... 5
Figure 6. Silicone Thumb Prosthesis ......................................................................................... 6
Figure 7. Hook Prostheses [58], [44] .......................................................................................... 7
Figure 8. Oscar Pistorius and His Prosthetic Legs [47] ............................................................... 7
Figure 9. Luke Arm [34] .............................................................................................................. 9
Figure 10. Johns Hopkins Proto 1 [22], [65] ............................................................................... 10
Figure 11. Johns Hopkins Proto 2 [44] ....................................................................................... 10
Figure 12. Non-Invasive Surface Electrodes [1] .......................................................................... 11
Figure 13. Surface Electrode Cap [5] ........................................................................................ 12
Figure 14. Injectable Myoelectric Sensor [2] ............................................................................ 13
Figure 15. Sensory Mapping for TMR by RIC [3] ........................................................................ 14
Figure 16. Utah Slant Electrode Array [2] .................................................................................. 15
Figure 17. Monkey-Controlled Robotic Arm [31] ................................................................. 16
Figure 18. Asimo Humanoid Robot Made by Honda [13] .......................................................... 17
Figure 19. Partner Robot Made by Toyota [16] ......................................................................... 17
Figure 20. Robonaut Made by NASA and GM [17] ................................................................. 18
Figure 21. Hiroshi Ishiguro and his Android [30] ...................................................................... 19
Figure 22. Capacitive Force Sensor [23] ................................................................................ 20
Figure 23. FlexiForce Piezoresistive Force Sensor [62] ......................................................... 21
Figure 24. Piezoelectric Sensor Being Compressed from Top and Bottom [53] ......................... 22
Figure 25. Equivalent Circuit Diagram of a Piezoelectric Transducer [25] .............................. 22
Figure 26. Optical Sensor Design ............................................................................................ 23
Figure 27. The Ideal Op-amp [42] ............................................................................................ 24
Figure 28. Voltage Follower [42] ............................................................................................ 24
Figure 29. Noninverting Charge Amplifier [32] ...................................................................... 25
Figure 30. A 2-to-1 Multiplexer ............................................................................................. 25
Figure 31. A 16 to 1 Multiplexer ............................................................................................. 26
Figure 32. TekScan Matrix-Based Tactile Sensor [62] .......................................................... 28
Figure 33. Gasketed Interface Pressure Measurement using Pressure Sensitive Paper [59] .... 28
Figure 34. Tekscan Grip System [62] .................................................................................... 29
Figure 35. ConTacts Conformable Tactile Pad [27] ................................................................. 29
Figure 36. Comparison of Optical Microscope and New MEMS Tactile Sensor Image [20] ....... 31
Figure 37. Fabrication Process and Finished Tactile Sensor Array [39] ................................. 32
Figure 38. Micromachined Tactile Sensor [36] ....................................................................... 33
Figure 39. Capacitive MEMS Sensor [18] .............................................................................. 33
Figure 40. Photo-reflective Tactile Sensor [38] ...................................................................... 34
Figure 41. Synthetic E-Skin by Takao Someya Group [63], [37] .............................................. 35
Figure 42. Stretchable Skin by Takao Someya Group [37] ..................................................... 35
Figure 43. Makota Sewn Wire Design and Implemented Tactile Sensors [41] ....................... 37
Figure 44. Shadow Hand with Tactile Sensor [60] ................................................................. 38
Figure 45. Flexible PCB Tactile Sensor [29] ......................................................................... 39
Figure 46. Flexible PCB Tactile Sensor with Silicone Covering [29] ....................................... 39
Figure 47. Designs A and B ..................................................................................................... 41
Figure 48. Piezoceramic Disks [53] ......................................................................................... 42
Figure 100. Force Applied to a Piezo in the Polarization Direction [53] ................................. 90
1. INTRODUCTION

1.1 Statement of Problem

With technology making ever more advanced mechanical manipulators possible, both in the fields of robotics and prosthetics, a need has arisen for tactile sensors superior to those of today. Current tactile sensors consist mainly of rigid or semi-rigid flat sheets that must be conformed to flat grippers. Recent research has found that superior handling can be achieved by using deformable grippers in order to maintain more contact area and contact friction with the object, much as human skin is able to deform around a hard object [43]. To make a tactile sensor that is deformable, even stretchable, while maintaining the ability for accurate measurement would be advantageous to many different fields of science and engineering.

1.2 Background

Since their inception, sensors have provided knowledge about the environment to an external user or source. Merriam-Webster’s Dictionary defines a sensor as “a device that responds to a physical stimulus (as heat, light, sound, pressure, magnetism, or a particular motion) and transmits a resulting impulse (as for measurement or operating a control)” [45]. Sensors have a multitude of uses in modern society. Often times sensors are used to quantify information about the environment, such as telling temperature or force. Sensors are sometimes used to relay information from an inaccessible location, such as beyond a wall or inside a chamber. Many times they provide information beyond the human scope of sensing, such as extremely high speeds. Still other times, sensors are used to provide information otherwise unattainable to humans, such as electric fields or the presence of carbon monoxide. Often these sensors are binary in nature, that is, a simple on or off, present/not present. In other cases, the sensor can provide a range of data about the measurement, such as the magnitude of a force value relative to a known measurement.
Perhaps the most basic sense, but also the most useful type of sensor is one that provides input about pressure or force. In humans especially, the sense of touch is vital to performing many routine functions. Being able to pick up or interact with almost any object requires the sense of touch in order to determine when contact has been made, and with how much force contact must be maintained. A great deal of knowledge can be attained through the sense of touch. Texture, temperature, shape, and mass are all common inferences through touching an object. Sensing the magnitude of the force via touch is crucial when dealing with fragile objects, such as picking up an egg or screwing in a light bulb. Too hard a force exerted on a fragile object and it is likely to break. Counter to that, when dealing with heavy objects, a significant force must be applied to the object. The sense of touch, rudimentary though it may be, is very crucial to interacting in the world.

1.2.1 Damaged Human Sense of Touch

Even with the most modern medical knowledge, nerve endings and the central nervous system are mysterious and not completely understood. Because the human sense of touch is provided through the skin, it is also easily damaged. As shown in Figure 1, the nerve endings are located in the epidermis, very close to the surface of the skin.

![The Skin](image)

**Figure 1. Drawing of Human Skin and its Parts [35]**
When nerve endings are severely damaged, the sense of touch is usually unable to be repaired. Severe trauma, cuts, and burns often damage nerves and nerve endings. In the case of severe burns for example, medical treatment often includes a skin graft. Skin grafts involve the transfer of skin from an undamaged donor to replace damaged skin. This can be done for both cosmetic reasons, as to repair the appearance of a burned area, as well as to aide in recovery time. A skin graft is performed by taking skin from an area such as the back or posterior of the leg and adhering it to the damaged area. While the overall healing process is usually expedited and appearance improved, skin graft procedures most often result in very limited sense of touch, if any, depending on the depth and severity of the original injury.

1.2.2 Prostheses

To overcome the limitations produced by injury, people have created and used prosthetic devices. Prosthetic devices have been around for centuries, with the oldest specimen discovered archaeologically, known as the Roman Capua Leg, dating back to 300 BC. Found in a tomb in Italy, it was made of copper and wood. Even older than that, artificial toes have been found on Egyptian mummies dating to 1295 BC. In the 15th and 16th centuries, blacksmiths made artificial limbs out of iron for soldiers who had lost limbs, as you can see in Figure 2 [12].

Figure 2. Iron Prosthetic Hand Worn circa 1508 [12]
Prostheses can be separated into several categories, such as cosmetic, functional, and with emerging technology, robotic prostheses.

**Cosmetic Prosthesis**

Cosmetic prostheses are used to cosmetically repair a user’s appearance. Perhaps the most common cosmetic prosthesis would be the artificial (glass) eye. Other cosmetic prostheses can be used to cover scarring from burns, crashes, cuts, or surgery. The cosmetic damage produced by many injuries can make it hard for victims to go about their normal business without attracting unwanted stares, questions, or even prejudice. Similarly, some birth defects also result in an altered physical appearance which draws unwanted attention. For these people, the use of a cosmetic prosthesis (sometimes called a cosmesis) can be of great help.

In situations where a body part has been amputated or else never formed, the use of prosthesis can aid in appearance and sometimes functionality. A finger prosthesis, as shown in Figure 3, can help repair the natural appearance to a person’s hand, fixing uncomfortable handshakes and improving overall confidence. It also serves to protect the sensitive tip of a finger from trauma and temperatures, as well as return functionality for using a computer keyboard correctly and without discomfort [15].

![Figure 3. Cosmetic Finger Prosthesis [15]](image-url)
An ear prosthesis, as shown in Figure 4, can restore an ear which has been lost to cancer surgery, amputation, burn, or congenital defects. These ears can aid in channeling sound waves into the auditory canal, improving hearing by as much as 20%. The prosthetic ear also helps keep the canal clean from dust and debris, retain eyeglasses and a hearing aid if needed, as well as acts as a great psychological benefit in the rehabilitation of the patient [15].

Figure 4. Ear Prosthesis [15]

A full face prosthetic device, as shown in Figure 5, provides valuable protection for the skin grafted face and head. It is able to restore normal symmetry to the face, eyebrows, and even a beard or mustache if wished. Mostly however, a full face prosthesis allows the patient to socially interact in public and avoid embarrassing stares and unwanted attention produced by their differences.

Figure 5. Cosmetic Facial Prosthesis [15]
In most cases, these cosmetic prostheses are made out of a highly durable yet soft silicone rubber which is designed for prolonged use under normal conditions. The prosthesis is capable of simulating both the texture and look of human skin. It can feature both major and minor wrinkles (including fingerprints), are hand painted and tinted to differing shades of skin pigmentation in order to blend with the surrounding anatomy, and can even be implanted with hair [15]. A very detailed silicone rubber finger prosthesis made by Dr. Robert Erb of Pennsylvania can be seen in Figure 6.

![Silicone Thumb Prosthesis](image)

**Figure 6. Silicone Thumb Prosthesis**

While cosmetic prostheses can return partial functionality, appearance, and have great psychological benefits, they do have one serious problem; they cannot provide any sensory information to the user.

**Functional Prostheses**

In addition to cosmetic prostheses, there are also prostheses which admittedly do not try to resemble the original human appendage, but instead aim to return as much functionality to the user as possible. Historically, the most common functional prostheses have been wooden peg legs. Like all other aspects of human life though, prostheses have now begun to take advantage of advances in science and engineering. Introduced in 1912, the “hook prosthesis” uses a series
of aircraft cables and rubber bands to allow users to open and close a hook at the end of a prosthetic arm by a shrug of their shoulder, as seen in Figure 7 [9].

![Figure 7. Hook Prostheses [58], [44]](image)

The wooden peg leg has now been replaced with titanium shafts, ankle joints, and plastic “feet” to enable the wearing of shoes. For some applications, such as sprinting, prosthetic legs have even come under controversy for benefitting the runner more than a biological leg would. In being able to reposition his legs 15.7% faster than the fastest man in the world (Usain Bolt), and losing only 9% of the ground force per step compared to a biological leg, South African sprinter Oscar Pistorius, seen in Figure 8, nearly managed to make the 2008 Olympic team [47].

![Figure 8. Oscar Pistorius and His Prosthetic Legs [47]](image)
Robotic Prostheses

Now in the midst of the computer age, recent advances have allowed for movement of these prostheses via motors or actuators. Soon, even more advanced prosthetics will be developed which allow not only simple movement, but human-level movement. Many different groups are working on devices and techniques for controlling prosthetic arms that would make them capable of matching or exceeding the human range of motion.

The Revolutionizing Prosthetics Program, begun by the Defense Advanced Research Projects Agency (DARPA) in 2005, has sponsored research at over 30 different organizations including 10 universities. The goal of the project is to fund the development of two mechanical prosthetic arms. The first of these projects, aimed at developing the most sophisticated mechanical arm possible with currently available technologies, was contracted to Deka Research and Development Corporation in New Hampshire. The second arm, coordinated by the Applied Physics Laboratory at Johns Hopkins University, seeks a fully functioning neurally controlled prosthetic arm using experimental technologies. So far, DARPA has funded at least $100 million into the two projects [21].

The Luke Arm by Deka

The goal for the first arm, made by Deka and called the Luke arm, is to give amputees an advanced prosthesis that could be available immediately “for people who want to literally strap it on and go.” The team, led by Segway inventor Dean Kamen, designed the Luke arm to be controlled with noninvasive measures using a foot-operated joystick, although it can be adapted to work with other technologies [4].

The Luke arm, seen in Figure 9, features 18 degrees of freedom (compared to the 23 of the human hand and arm), and can be modularized depending on the degree of amputation: the hand and wrist as one piece, the forearm as another, the elbow/upper arm as a third, and finally the shoulder, each containing separate electronics. The Luke arm also provides force feedback to
the user without surgery. A tactor – a small vibrating motor – is secured against the user’s skin. A sensor on the prosthetic sends a signal to the tactor which changes with grip strength. When a user grips something lightly, the tactor vibrates slightly. As the user’s grip tightens, the frequency of the vibration increases. This limited feedback can enable users like Hildreth to pick up a flimsy paper cup without crushing it, or peel a banana without squishing it [4]. The Luke Arm is now undergoing widespread testing among veterans in a three year study by the U.S. Department of Veterans Affairs to provide engineering feedback, as well as FDA approval [34].

![Figure 9. Luke Arm](image)

*Figure 9. Luke Arm* Error! Reference source not found.

**Johns Hopkins Hand (Proto 1 and Proto 2)**

The Modular Prosthetic Limb (MPL) project, a four-year effort coordinated by the Applied Physics Laboratory at Johns Hopkins University, has produced two arms, the Proto 1 and the Proto 2, both to be controlled with a user’s thoughts. The Proto 1 arm features eight degrees of freedom, thus providing a level of natural control far beyond the current state of the art for prosthetic limbs, and even incorporates sensory feedback. Both the control and integrated sensory feedback demonstrated with the Proto 1 are enabled by Targeted Muscle Reinnervation (TMR), a technique pioneered by Dr. T. Kuiken at the Rehabilitation Institute of Chicago. The
Proto 1, seen in Figure 10, was such a success that it has been transitioned to industry and sold commercially by one of the project partners, Otto Bock [22].

The Proto 2, seen in Figure 11, has 27 degrees of freedom, including independent movement of each finger. Meant for a transhumeral, or a complete shoulder disarticulation, it is capable of reproducing the strength and dexterity of a human arm, capable of curling 50 pounds [21]. The complete arm weighs about nine pounds, with a final engineering goal of weighing close to 7 pounds (the weight of an average female arm). It also has more than 80 individual sensory elements for feedback of touch, temperature, and limb position [56]. It has been tested using both Targeted Muscle Reinnervation as well as Injectable MyoElectric Sensors (IMES) [6].
1.2.3 Interfacing Techniques

Several different ways of interfacing to these mechanical prosthetic arms are being developed and tested, with each enabling the user to control the prosthesis using only his or her mind. As it turns out, the degree of control, as well as level of sensory feedback possible, is directly proportional to the invasiveness of the method.

*Surface Electrodes*

For the most basic activities, such as grasping a ball, surgery is not required. Completely non-invasive electrodes, called Myoelectric devices, can be placed on the skin surface of the residual limb, as seen in Figure 12. These devices are capable of detecting EMG signals (electrical activity produced by muscle activity) still being transmitted to the nonexistent hand from those residual muscles. Using signal processing and pattern-recognition algorithms, those electrical impulses can be translated into instructions for the motors and microprocessors of the prosthetic arm. However, while these electrodes can amplify the signal, they cannot clean it up, and by the time the signal travels from the originating muscle through layers of flesh and skin, a lot of noise has been introduced. Thus it is difficult to differentiate between separate or differing degrees of muscle movement [1].
A different method of control using surface electrodes is also being tested at Johns Hopkins, involving 64 electrodes inside a tight-fitting cap to be worn on the head, as seen in Figure 13. The electrodes pick up electric fields in the mu band range, caused by neurons firing inside the motor cortex area of the brain. These electric fields provide only a broad reflection of what the user is thinking about moving, but do not require any actual movement, only thought about movement. However, manipulating these fields can take years of training to get reliable results, and have been deemed impractical for the aims of DARPA [5].

![Figure 13. Surface Electrode Cap [5]](image)

**Injectable MyoElectric Sensors (IMES)**

The next level of invasiveness and control uses small wireless devices called Injectable MyoElectric Sensors (IMES), developed by Chicago-based Sigenics Inc. [9]. These tiny, rice grain-like devices, as seen in Figure 14, are injected or surgically implanted into the muscle tissue of the residual arm and work just like the surface electrodes to tap into the muscles signals right at the source. By picking up the signals at the source, more information about the specific muscle groups sending the signal can be determined, and finer control can be had. Individual movements of the fingers are capable of being differentiated using IMES devices, and relayed to the prosthetic limb. These IMES are perpetually powered by a coil in the prosthetic limb, so they never need batteries. To use IMES, however, there must be some residual arm in which to inject the devices [1].
Targeted Muscle Reinnervation (TMR)

For more severe amputations, for example having both arms removed at the shoulder, there may not be much arm or muscle left for IMES or surface electrodes with which to work. The next level of interface bypasses the residual muscles to tap directly into the peripheral nerve which directs the muscles. Using Targeted Muscle Reinnervation, pioneered by Todd Kuiken at the Rehabilitation Institute of Chicago (RIC), some of the 70,000 nerves that once traveled from the brain into the arm are rerouted to the pectoral muscles. Then, when a person thinks about moving their (non-existent) arm, a nerve signal travels from the brain down the nerve as a result of the intention, and that spike causes twitches in the pectoral muscle. By mapping how the pectoral muscles twitch under different intentions, those signals can be translated into corresponding actions for the prosthetic arm [1]. With Kuiken’s surgery, a user can naturally control a prosthetic arm with his or her thoughts as if it were an extension of the person’s flesh [4].
Incredibly, this technique has also allowed some pressure feedback applied to the limb. When pressure was applied to the prosthetic thumb, the user was able to feel it on their nonexistent thumb, not on their chest [1]. Now, Kuiken and his team have found that patients can actually feel touch on the skin of the chest as if it were on the skin of the missing hand. After mapping the sensitive spots on the chest to specific parts of the missing fingers and hand, researchers found that patients can feel heat, cold, and pain, as shown in Figure 15 [64]. Targeted Muscle Reinnervation has had such success that it is now undergoing FDA approval [9].

**Utah Slant Electrode Array (USEA)**

If, for whatever reason, these unused areas of muscle are unavailable or damaged, access to peripheral nerves is also available using penetrating electrodes that intersect the nerves with tiny needles. Researchers at the University of Utah have developed an implantable device called the Utah Slant Electrode Array (USEA), a 5-millimeter-square grid of 100 needlelike electrodes, seen in Figure 16. These electrodes hold hundreds of different mechanisms, among them signal...
amplifiers, storage registers, and a multiplexing scheme to transmit to a receiver on the skin. Like IMES, these devices can be powered wirelessly through the skin and extract a signal in real time, however, unlike IMES, they access the nerves directly instead of the muscles obeying the nerves. In theory, that subtracts another layer of signal interference. The electrode arrays are still in experimental stages however [1][1].

In the most extreme cases, where the body no longer offers any means of interfacing to the artificial limb, for whom even nerve-rerouting surgery may not be an option, Utah Electrode Arrays are relocated to the source of all neural signals – the motor cortex of the brain. The electrodes intercept the motor neurons firing their instructions, which are interpreted using complex algorithms and translated into direction for the mechanical prosthesis [1][1]. Currently, these electrodes have only been implanted into the brain of rhesus monkeys [2]. In the MotorLab at the University of Pittsburgh, scientists have taught a monkey to control a robotic manipulator with 7 degrees of freedom, shown in Figure 17. The monkey received two brain implants, one in the hand area and another in the arm area of its motor cortex. On thought alone, the monkey has been taught to maneuver the robotic arm, grasp a knob, and twist [31].
The largest problem with current brain penetrating electrodes is that after a year, the defensive mechanisms of the brain kick into gear. Protective astrocytes and glial cells then seal off the foreign object inside a thick white capsule, effectively blocking access to the neuronal spikes that could control a prosthetic limb [5].

The goal for all these methods is to intuitively be able to control a prosthetic limb, with nearly no learning curve. The user will think about moving, and the prosthetic device will move. The addition of sensory feedback from some of these methods provides the possibility for even more functionality from these robotic prostheses.

1.2.4 Humanoid Robotics

Similar to mechanical prostheses, many robotic manipulators have the need for tactile sensors to allow them to properly handle objects. These robots can range from industrial manipulators used in manufacturing to sophisticated human-like autonomous robots. In order to be useful, these robots need to be able to manipulate objects, and that means knowing when and with how much force they are contacting it.
ASIMO, an advanced humanoid robot shown in Figure 18, is not only capable of walking, running, and making hand gestures, it can also move objects, distinguish sounds, and recognize faces. Over 100 ASIMO units are now in existence, with each one costing just under $1 million to manufacture. With the aim of Honda being to make ASIMO a household companion capable of performing a variety of useful tasks, the complexity to which it will be expected to manipulate objects will only increase [19].
In 2004, Toyota announced plans to build “Toyota Partner Robots”, seen in Figure 19, and begin selling them as early as 2010. Toyota envisions these robots assisting in many different capacities, the first of which might be in hospitals. Plans are under way to field test these Partner Robots extensively at the hospital of Toyota in Toyota City, Japan as nurses and aides to the elderly. Eventually, they might be capable of aiding elderly at home or even work in factories [16]. Another humanoid robot, made in a joint effort by the National Aeronautics and Space Administration (NASA) and General Motors (GM), is intended to help humans work and explore in space. Called Robonaut and seen in Figure 20, it is currently in its 2nd iteration (thus Robonaut 2, or R2). The goal of Robonaut is to one day assist astronauts with repairs on the space shuttle, the International Space Station (ISS), or even explore other planets, most likely by remote. NASA plans to launch R2 into space on the Space Shuttle Discovery in November 2010 where it will be deployed on the ISS. Though the robot currently does not have a lower half and will be deployed on a fixed pedestal while inside the ISS, research is underway on adding either legs or wheels to the robot to traverse across Lunar or Martian terrain [17].

Figure 20. Robonaut Made by NASA and GM [17]

Other humanoid robots have been made with their primary focus not on performing useful tasks, but in the aim of simulating human looks as closely as possible. Hiroshi Ishiguro of Japan has done just that, constructing a robotic copy of himself out of silicone rubber, pneumatic
actuators, powerful electronics, and hair from his own scalp, as shown in Figure 21. While the robot is permanently in a sitting posture, it can be controlled remotely by Ishiguro via computer, with a microphone used to capture his voice and a camera to track his facial and head movements. When Ishiguro speaks, the android speaks, when Ishiguro tilts his head, the android does the same. The mechanical android can blink, twitch, and even appears to breathe while being used for remote conversations. Ishiguro has used it to hold a trans-oceanic conversation from Japan to Austria [30].

Soon, robots may do household chores, care for the elderly, assist with physical therapy, monitor the sick, teach classes, and serve cappuccinos at Starbucks. While not all scientists agree, to function in human surroundings and use human tools, robots are best suited if they function like humans (that is, two legs, two arms, and about our height). But in order for them to be accepted in these roles, robots have to not only look like but act like humans. In response to this next challenge, social robots are coming to life in labs around the world, including MIT, Carnegie Mellon, universities in Japan, China, Korea, and elsewhere [30]. And to interact with human environment as humans do, humanoid robots need a comparable sense of touch to humans.
1.2.5 Force Sensors

A variety of different methods exist for sensing pressure and force. In most cases these sensors work using the capacitive, piezoresistive, or piezoelectric effects of a material, with a small number of force sensors actually using optics. Piezoresistive force sensors are by far the most common [52].

**Capacitive**

Capacitive sensors use the electrical property of capacitance in order to determine the amount of force applied. One method of measuring capacitance involves applying an alternating voltage between two conductive plates, which causes the charges to continually reverse their positions. The capacitance between the two plates is proportional to the area of the plates and inversely proportional to the distance between them, as shown in Figure 22. Thus larger and closer objects cause greater capacitance than smaller and more distant objects. The moving of the charges between the two plates creates an alternating electric current which is measured by the sensor, with more capacitance causing more current flow. Changes in the distance between two conductive surfaces result in an affected capacitance, which the sensor can measure to determine the force applied [23].

![Figure 22. Capacitive Force Sensor [23]](image-url)
Piezoresistive

Piezoresistive sensors, seen in Figure 23, operate based on the principle that as a conductive metal is stressed, its resistance changes. When stretched, the metal becomes skinnier and longer, both of which result in an increase in its electrical resistance. Conversely, when placed under a compressive load (without buckling), it will broaden and shorten, resulting in a decrease in the electrical resistance. When contained so as to prevent permanent deformation of the metal, the strip can be used to measure an applied force. This technology is used to make most load cells as well as fluid pressure sensors. A variety of load cell designs exist, which orient this conductive material differently to measure applied forces. Likewise, pressure sensors most often use a strain gauge between the two sides of a diaphragm to measure the difference in pressure caused by an applied force [52].

Figure 23. FlexiForce Piezoresistive Force Sensor [62]

Piezoelectric

Piezoelectric sensors use the material property of piezoelectricity in order to measure applied force or deformation. A piezoelectric element is simply an element which changes dimensions when stressed by an electric voltage, or when stressed mechanically by a force generates an electric charge. Because of this dual nature, a piezoelectric is capable of acting as either a sensing or actuating element.
When a mechanical stress is applied to a single sheet of piezoceramic, a voltage is generated which tries to return the piezo to its original thickness, as shown in Figure 24. By measuring the voltage difference between the two sides of the piezoceramic, a measurement can be made of the amount of force applied to it [53].

When used as a transducer, a piezoelectric has a very high DC output impedance, and can be modeled as a proportional voltage source and filter network. A detailed model, shown in Figure 25, includes the effects of the mechanical construction of the sensor and other non-idealities. The inductance, shown as $L_1$, is due to the seismic mass and inertia of the sensor itself. In most cases, the inductance can be ignored as the sensor is not used near its resonance frequency. A capacitor $C_1$ is inversely proportional to the mechanical elasticity of the sensor. The capacitor $C_0$ represents the static capacitance of the transducer, resulting from an inertial mass of infinite size. The resistor $R_0$ is the insulation leakage resistance of the transducer element. When the sensor is connected to a load resistance, this acts in parallel with the insulation resistance [25].
Due to the nature of piezoelectric elements, when used as sensors or force gauges by themselves they are excellent for handling dynamic and transient inputs, but poor at measuring static inputs. This is due to charge leakage (current escaping) from the element. On their own, piezoelectric elements can be used effectively for transient force measurements. In order to better measure long-lasting forces though, additional electronics and computer coding are usually added to the system [53]. These added electronics typically consist of a voltage follower or charge amplifier. Additional information about piezoelectricity can be found in Appendix A.

**Optical**

A small number of force sensors use optics as their underlying technology. By arranging both an optical source (usually an LED) and a photodetector in such a manner that any an applied force would alter the path or intensity of the light from source to detector, a functional force sensor can be created, as shown in Figure 26. There are a variety of orientations possible that are capable of accurately measuring the change in light intensity, and thus the amount of force applied. With the advancement of electronics, both photo emitters and detectors are becoming ever smaller, allowing for smaller overall sensor design.

![Figure 26. Optical Sensor Design](image)

1.2.6 Op-Amps

Operational amplifiers, often called op-amps, are an important part of many electronic circuits. High-gain electronic amplifiers feature a differential input and usually a single-ended output, capable of amplifying a voltage hundreds to thousands of times larger than the voltage difference between the inputs. Ideally, an op-amp has an input offset voltage of zero (V_E in
Figure 27), as well as infinite input resistance \((Z_i)\). While ideal op-amps do not exist in real life, present day op-amps have come so close to ideal that they can usually be assumed as ideal for analysis purposes [42].

![Figure 27. The Ideal Op-amp](image)

A multitude of circuits can be created using op-amps, including noninverting amplifiers, inverting amplifiers, differential amplifiers, adding circuits, and many others, each serving a different purpose. When operating at unity gain (no amplification), the noninverting amplifier reduces to a voltage follower, where the output voltage is identical to the input voltage, as seen in Figure 28 [42]. By using an op-amp to create a voltage follower circuit, a high impedance input source can be buffered to a low impedance output. When used with a piezoelectric sensor, it allows for measuring of the voltage difference, while greatly reducing charge leakage.

![Figure 28. Voltage Follower](image)

Charge amplifiers, as seen in Figure 29 and sometimes called current integrators, are capable of integrating weak charge pulses and converting them into voltage pulses for amplification. Because of this, they are often used in applications using piezoelectric sensors or
photodiodes, where the charge is output can be converted to a voltage [32]. In practice, a charge amplifier requires additional circuitry in order to allow for a circuit reset, as there is no method of discharging the capacitor, hence any leakage current will eventually charge the capacitor until the circuit becomes saturated [42].

![Noninverting Charge Amplifier](image)

**Figure 29. Noninverting Charge Amplifier [32]**

1.2.7 Multiplexers

A multiplexer, or mux, is another common component of many circuit designs. A multiplexer is an electronic device capable of selecting one out of a number of analog or digital input signals and forwarding that signal along its output line. As a general rule, multiplexers come with $2^n$ input lines, requiring $n$ select lines which are used to choose which input line to forward to the output. A multiplexer thus makes it possible to send several signals worth of data over a single communication line, rather than having one line per device. A multiplexer can be used in reverse (called a demultiplexer) in order to convert one time-dependent data line into several output signals [48]. Often times, a multiplexer and demultiplexer will be used in connection in order to reduce the number of wires required to send several channels worth of data, thus saving money.

![2-to-1 Multiplexer](image)

**Figure 30. A 2-to-1 Multiplexer**
How a multiplexer is used can be expressed by a truth table. For instance, the truth table for a 2 to 1 multiplexer as shown in Figure 30 can be seen in the accompanying Table 1. When the signal line ($S_0$) is 0, the signal from input A is forwarded to the output, and the signal on input B does not matter (represented by an X in the truth table). Conversely, when the signal line is a 1, the signal on input B is forwarded to output, and the signal on input A does not matter.

**Table 1. Truth Table for a 2-to-1 Multiplexer**

<table>
<thead>
<tr>
<th>$S_0$</th>
<th>A (0)</th>
<th>B (1)</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Similarly, a 16 to 1 multiplexer is able to forward any of 16 input signals to one output signal, as shown in Figure 31. To choose between 16 input lines, 4 signal lines are required. This is because it takes 4 binary bits in order to indicate the numbers 0 to 15.

**Figure 31. A 16 to 1 Multiplexer**
2. Literature Review

With the great advances in electronics and ever increasing presence of robotics in everyday life, the need for tactile sensing to go along with other forms of sensory input has become apparent. While interacting with a static or known environment can easily be done with limited sensory input, additional information is required when dealing with an unknown or changing environment. For instance, a pre-programmed robot can be made to perform tasks based on precise timing and position, granted that the environment fits with the programming of the robot. However, when interacting with unknown objects of unknown size, weight, distance, etc., getting sensory input becomes vital. Cameras and range sensors only go so far when interacting with an environment, at which point the need for tactile sensing becomes evident.

2.1 Current Tactile Sensor Technologies

Tactile technologies using piezoresistivity are now nearly common-place, with a large variety of companies offering tactile sensors for use in a wide assortment of fields. In most cases, these piezoresistive sensors are created by containing a pressure-sensitive (resistive) ink between two conductive surfaces, encased in flexible sheets of polyester as a backing. For high temperature sensors, the backing can be made from polyimide in place of polyester. They can be made in both single load-cell configurations as well as grid (matrix)-based sensors with multiple sensing areas, as seen in Figure 32. In grid-based sensors, the conductive surface of one sheet forms a row pattern while the conductive surface of the other employs a column pattern. The intersection of each of these rows and columns creates a sensing cell, called a sensel. The spacing between the rows and columns can be varied according to the sensor application, sometimes as small as 0.2 mm [62].
Alternatively, a one-time-use pressure-sensitive paper can also be used as a tactile sensor. By coating paper in tiny dyed microcapsules that rupture under pressure, an approximation of the pressure endured can be made. With more intense pressure the density of ruptured capsules increases, and by examining the intensity of the color, a reasonable pressure estimate can be made, as shown in Figure 33 [59].

These commercially available tactile sensors have an assortment of intended uses, from body mapping for shoes, mattresses, and body armor, to the testing of mechanical seals. Additionally, tactile sensors can be used for product testing, as with the grip system shown in Figure 34, or in medicine for fitting prostheses, analyzing joints, analyzing foot pressure, gait, and posture, or in dentistry for measuring occlusal pressure [62]. These tactile sensors are also commonly used in conjunction with robotic grippers to provide knowledge of how much pressure is being applied to the object.
A small number of commercially available tactile sensors are capacitive based. Also built in a number of different configurations for different applications, these pads are capable of providing sensory feedback for both object centered (applied to an object) and user centered (attached to a user, as on the fingertips) applications [27].

2.2 Research on Tactile Sensor Technologies

With the use of tactile sensors increasing in recent years, shortcomings in current tactile sensors have been made more apparent, and these limitations have prevented them from being further implemented into robotic design. A significant amount of research has been conducted into improving upon these tactile sensor shortcomings; among them their inflexibility, rigidity, resolution, size, and sensing range.

2.2.1 Understanding the Human Sense of Touch

In order to better design a mechanical or electronic tactile sensor though, a significant amount of research has gone into better understanding the human sense of touch. Dr. Mandayam A. Srinivasan, founder of the MIT Touch Lab, says the ideal sensor “essentially has to be what...
[human] skin is: flexible, with the ability to sense dynamically and with high spatial resolution, and physically robust – it shouldn’t break, it shouldn’t wear out” [20].

With about 17,000 mechanoreceptors in the grasping surfaces of the human hand, and a center-to-center spacing ranging from .7 mm in the fingertip to 2 mm in the palm, it is quite a challenge for engineers to duplicate. This spatial resolution means that the human fingertip is able to recognize the difference between one indenter and two indenters when separated by at least 1 mm. When sliding a hand over smooth surface, it is possible to detect a feature raised by as little as 0.85 microns [33].

Also, recent robotics research has found that superior handling can be achieved by using deformable grippers, much as human skin is able to deform around a hard object. Human fingers conform to the shape of a grasped object, giving rise to larger contact areas and the ability to apply larger frictional forces. This deformation is a key factor in the ability of the human hand to create stable, encompassing grasps with subsets of fingers. Typical robot hands use stiff fingers that do not deform, and this often leads to difficulty in grasping. To make a tactile sensor that is deformable, even stretchable, while maintaining the ability for accurate measurement would be advantageous to many different fields of science and engineering [43].

However, in some ways, electronic sensors already outperform biological sensors. Nerve conduction velocities are only 60 m/s, very slow compared to robotics. For involuntary responses, such as clutching a glass that is felt slipping out of the hand, latencies of 20-30 ms are common, with much longer response times for other reflexes that require voluntary reaction. It is also worth noting that biological tactile sensors are hysteretic, nonlinear, and time varying; all properties generally looked down upon when attempting to create an electronic sensor. Even with these deficiencies though, the large number of sensors available in biology allows for the collection of good data by sorting through redundant information [33].
2.2.2 MEMS

Much research has been done involving the creation of microelectromechanical systems (MEMS), which are an obvious method for increasing sensor resolution. By creating structures on the microscale level, extremely small sensors can be created, thus allowing for very high sensor resolution and tight spacing.

Among the leading MEMS research, a nanoparticle film by Ravi F. Saraf and Vivek Maheshwari at the University of Nebraska has been created which is only 100 nm thick. The thin film does not have the same robustness, flexibility, or ability to sense temperature as human skin, but is a big step forward in spatial resolution. The sensor has a high enough resolution (40 micrometers horizontally and 5 micrometers vertically) to “feel” single cells. This one day could be used by surgeons to help find the perimeter of a tumor during surgical procedures. The actual sensing can be done either based on the principle of piezoresistivity by examining the current flow through the different layers of the film or by using a small camera on the backside of the film, which picks up the glow from cadmium sulfite nanoparticles as electricity travels through the layers of the film, as shown in Figure 36. Both the light and the electrical current are proportional to the pressure on the sensor, with the camera able to take readings about 5-10 times per second and the electric current able to take readings between 20 and 50 times per second. The film is presently placed on an electrode-coated glass backing, but could soon be put on a flexible polymer instead of glass, thus allowing the film to wrap around the finger of a robot [20].

![Figure 36. Comparison of Optical Microscope and New MEMS Tactile Sensor Image [20]](image-url)
Another MEMS-based tactile sensor has been created by Korean scientists and packaged on a flexible PCB intended for human-robot companion applications. By creating a diaphragm of 1.2 mm x 1.2 mm, as shown in Figure 37, and creating a series of piezoresistors using phosphorous ion implantation, then placing one at each of the 4 edges of the diaphragm, any deformation of the diaphragm results in a load detected by the piezoresistors. Each completed sensor is 1.5 mm x 1.5 mm, then arranged onto a 7.7 mm x 7.7 mm array of 16 sensors [39].

![Fabrication Process and Finished Tactile Sensor Array](image)

**Figure 37. Fabrication Process and Finished Tactile Sensor Array [39]**

A similar approach has been used by researchers at the University of Illinois at Urbana-Champaign to create a diaphragm MEMS tactile sensor. To increase the robustness of the sensor, a polymer substrate was used that is mechanically rigid and brittle, thus prohibitive to use over contoured surfaces. Micromachined thin-film metal strain gauges were positioned on the edges of polymer diaphragms (called Tactile Bumps by the researchers, as shown in Figure 38) where again the change in resistance was used to measure the force applied to the diaphragm. Also to increase the robustness, the entire sensor was machined in one piece using an inverted fabrication process [36].
Another type of MEMS tactile sensor was designed by researchers at the University of California at Berkeley using capacitance sensors rather than piezoresistive sensor. An eight-by-eight array of sensors was micromachined where the entire sensor array was smaller than normal human spatial resolution of 1mm. Each square tactel was less than 100 micrometers on a side, with similar spacing between elements. It was created by using doped polysilicon with an air gap dielectric of .5 micrometers, and a thin protective layer of silicone rubber, as shown in Figure 39. This silicone rubber adhered to the polysilicon surface to provide interpolation of normal loads between elements. Although the sensor had severe hysteresis problems, it successfully showed a very high resolution capacitive MEMS tactile sensor [18].

Figure 38. Micromachined Tactile Sensor [36]

Figure 39. Capacitive MEMS Sensor [18]
2.2.3 Photo-reflective

Research has been done at the University of Tokyo to create a highly modular and scalable tactile sensor capable of covering a full body. Their tactile sensing design consisted of a photo-reflector covered by urethane foam, as shown in Figure 40. The sensors work by measuring the light scattered by the urethane foam upon deformation from a force. In order to detect forces between sensing elements, important for reducing the total number of required sensing elements, the force located between two sensors could be calculated as the ratio of the sensor outputs. These photo-reflectors were put on a flexible ribbon that is bendable and foldable for altering the length and allowing for correct placement of the sensing element. By using this design, the size of this tactile sensor was greatly reduced compared to other optical sensor, at only 3.2 mm x 1.7 mm x 1.1 mm each. Unfortunately, each sensor was found to use 50 mA, so a time-sharing technique was implemented to reduce overall power consumption [38].

![Photo-reflective Tactile Sensor](image)

Figure 40. Photo-reflective Tactile Sensor [38]

2.2.4 Takao Someya Group Research

Introduced in 2005, an electronic skin composed of pressure-sensitive rubber and organic transistors was developed by Japanese researcher Takao Someya and his team. A mesh of organic diodes acted as thermal sensors, and a mesh of organic transistors read data from the pressure sensors. These sensors are then embedded in a thin plastic film to create a net-like matrix, as shown in Figure 41. The e-skin could also be used for prosthetics, but the sensors still need a factor of 10 improvement to be able to pick up tiny pressures that human skin can sense [51].
More recently, the Takao Someya Group has been conducting research to create a truly elastic conductive material. By combining a salty liquid with malleable single-walled carbon nanotubes, a material has been made that can stretch up to 134% of its original size while improving conductivity by 570%, as shown in Figure 42. Further development is required however to sustain softness and elasticity in the rubber over time without compromising the conductivity [37].

2.2.5 Revolutionizing Prosthetics

Similar to the work done by the Takao Someya Group, researchers at Oak Ridge National Laboratory in Tennessee in conjunction with NASA and the National Institute of Aerospace (NIA) are embedding carbon nanotubes into a polyimide base to create a material that looks, feels, and functions like human skin. Part of the Revolutionizing Prosthetics Program, their initial plan is to create a 6-square-centimeter patch of the flexible, stretchable, lightweight, and
durable skin. The carbon nanotubes not only make the polymer stronger, but they also enhance the piezoelectricity of the polyimide, which is what is used to detect pressure. Next, researchers plan to embed temperature sensors under the polyimide layer, with the carbon nanotubes helping to transfer heat as quickly as possible from the polymer surface to the temperature sensors below [51].

The DARPA goal is to have artificial skin that can measure a force as small as 0.1 Newton, though their nanotube composite is not yet that sensitive. The nanotube composite is close to being able to achieving the spatial resolution of human nerve cells though, with a resolution of 5 mm [51].

### 2.2.6 Japanese Rubber Tactile Sheet

Another innovative idea to allow tactile sensors to cover three-dimensional bodies was implemented by a group of researchers also at the University of Tokyo in Japan. Using pressure-conductive rubber with stitched electrical wires, they were able to make a thin, flexible, single-layer composite structure that is very durable to external forces. The rubber consists of carbon particles, which act as electroconductive material and are dispersed uniformly in a silicone rubber matrix. The electrically-conductive rubber works on the principle that when under no pressure (steady state), the carbon particles are far enough apart that they do not conduct electricity well and thus have an extremely high resistance. However, when pressure is applied, the thickness of the rubber decreases, bringing the carbon particles closer together, and thus decreasing the resistance of the material [41].
A small diameter wire was then sewn into the semi-conductive rubber to create a row of electrodes, with another wire sewn in as a column, as shown in Figure 43. At each intersection of wires is a pressure-sensitive area. Thin films of the material were then mounted on the tips of 4 fingers of a robotic hand for testing. Though the sensors had high levels of hysteresis, they were found to be very durable and provide good conformability around curved surfaces [41].

2.2.7 Shadow Hand

The Shadow Robot Company in the United Kingdom has also implemented a pressure sensitive rubber on their Shadow Hand, as shown in Figure 44. The standard Shadow Tactile Sensor is available in two shapes, one suitable for the thumb and the other for the fingers, with either 34 or 22 individual tactile elements (tactels) on each. The 3.3mm diameter tactels are evenly distributed over the surface of the sensor for uniform coverage and to ensure no insensitive regions. Each sensor contains an on-board Programmable System-on-Chip (PSoC), a small integrated circuit, which allows the user to actively control the range and sensitivity of the sensor, making it capable of detecting loads ranging from 0.1 N to 25 N [60].
The pressure sensors, created by Peratech, use Quantum Tunnelling Composite (QTC) as the sensing medium. QTCs are a composite material made from metal filler particles combined with an elastomeric binder, typically silicone rubber. The unique method of combining these raw materials results in a composite which exhibits significantly different electrical properties when compared with any other electrically conductive material; it changes from an electrical insulator to a metal-like conductor when placed under pressure, thus acting as a pressure sensitive resistor. QTC has an unstressed resistance of $10^{12}$ ohms, but under pressure, the resistance can decrease to less than 1 ohm, and requires much less deformation to create the same resistance change compared to other carbon composites (such as strain gauges) [55].

2.2.8 Capacitive Flexible PCB

Another tactile sensor system has been researched by Korean scientists which is based on a mesh of capacitive sensors placed on top of a deformable flexible substrate. Each sensor has 12 capacitive taxels formed in a triangular shape placed on a two-sided flex PCB, and is able to be interconnected with up to 15 other sensors, as shown in Figure 45. A commercially available capacitance to digital converter integrated circuit (CDC) was used as the sensing element. One on side of the PCB are etched 12 circular pads, forming taxels and acting as armatures of capacitors. On the other side the CDC chip is mounted. A protective silicone rubber covering was placed over the taxels side of the PCB, as shown in Figure 46, also insulating the capacitive
armatures from natural electric signals created by human skin. Quantitative experiments are ongoing [29].

Figure 45. Flexible PCB Tactile Sensor [29]

Figure 46. Flexible PCB Tactile Sensor with Silicone Covering [29]
3. Need for Further Research

A significant amount of research has gone into tactile sensors, with improvement mainly concentrating on sensor resolution, size, and ensuring flexibility. After reviewing the literature on previous research, especially designs allowing for both flexible bending and elastic stretching, an area of need for further research was seen. Also, though multiple sources identified human prosthetics as a likely use for their tactile sensor, none implemented it in their research by covering an entire hand, rather than just fingertips, or by attempting to make the sensor covering look realistic to human skin (see Figure 46). With these deficiencies in mind, the design and creation of a new tactile sensor for use with humanoid robotics or advanced prosthetics was started, ensuring that the sensor would be deformable, elastic, and durable, and to make the sensor look as real to human skin as possible.

3.1 Design

3.1.1 Sensor Design and Choice

The first component of creating the artificial tactile sensor was choosing the method of sensing. As discussed in Chapter 1, there are several different types of pressure sensors. Using these different sensor types, four main designs were considered for this project, consisting of a piezoresistive, a piezoelectric, a capacitive MEMS, and a microfluidic MEMS (using piezoresistive fluid pressure) sensors.

Design A and B

Designs A and B would use piezoresistive sensors, either as a large tactile array, or as a number of smaller load cells for detecting the pressure at each point, as shown in Figure 47. Making a large piezoresistive sensor or multiple smaller ones from scratch would be extremely difficult and time consuming, thus commercially available piezoresistive sensors were researched. A large piezoresistive tactile sensor, as shown in Design A, would be extremely expensive, on the
order of $15,000 from Tekscan, but would provide excellent resolution and sensing capabilities. For Design B, the smallest piezoresistive sensors found were Flexiforce sensors, also by Tekscan, with a sensing diameter of 0.375” inside a 0.55” wide sensor and only 0.008” thick. Flexiforce sensors are priced at roughly $15 each. Tekscan also offers a variety of computer programs and electronic systems for use with their sensors, ranging from $600 to $1200 [62]. While the Flexiforce and tactile sensor are made on a flexible polymer, they would be able to bend, but provide almost no elasticity. Having a large number of 0.55” flat sensing surfaces would limit the conformability of the design too.

**Figure 47. Designs A and B**

**Design C**

Design C would use a number of very small piezoelectric sensors, which are not flexible at all, but can be found much smaller than piezoresistive sensors, allowing the overall system to still be flexible. Piezo Systems Inc. provides a wide assortment of piezoceramic disks, shown in Figure 48, which work using piezoelectricity. Their smallest disk, measuring only 1/8” in diameter and 0.0075” thick, needs only a wire attached to each side of the ceramic in order to measure the voltage difference across the sides and function as a piezoelectric sensor. These
small piezoelectric sensors could then be used similarly to Design B, and cost only $4 a piece [53].

![Figure 48. Piezoceramic Disks [53]](image)

**Design D**

Design D involved the creation of a capacitive MEMS sensor, using a method similar to one of the researched tactile sensors. Through micromachining, an array of capacitance sensors would be created on a flexible substrate using air as the dielectric gap. While the design would have excellent resolution, micromachining a large enough sensor or enough smaller sensors to cover an entire hand would be a very arduous manufacturing process.

**Design E**

Design E would also involve micromachining, in this case to etch tiny fluid channels out of a flexible substrate, with microfluidic pressure sensors attached to the channels to detect pressure changes due to loads. While the design would be highly flexible as well as elastic, differentiating the pressure changes due to elastic stretching or bending of the sensor as opposed to applied forces would be very difficult. Also, micromachining a large enough sensor, or else micromachining enough smaller sensors with which to cover an entire hand, would be very difficult.
**Final Design Selection**

To select the best design for this thesis, a pair-wise comparison of different criteria was performed, as shown in Table 2. In this comparison, each design criterion was compared against every other design criterion, and in each case a decision was made about which was more important for the design. In doing this, a total importance factor was determined for each aspect of the design, allowing for a meaningful weight to be applied to each criterion when creating a decision matrix.

Table 2. Pair-wise Comparison of Designs

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Resolution</th>
<th>Individual Sensor Size</th>
<th>Ability to Cover Hand</th>
<th>Flexibility</th>
<th>Elasticity</th>
<th>Difficulty of Manufacturing</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>x</td>
<td>I</td>
<td>A</td>
<td>F</td>
<td>E</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>Individual Sensor Size</td>
<td>x</td>
<td>x</td>
<td>A</td>
<td>F</td>
<td>E</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>Ability to Cover Hand</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>F</td>
<td>E</td>
<td>D</td>
<td>-</td>
</tr>
<tr>
<td>Flexibility</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>E</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Elasticity</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>Difficulty of Manufacturing</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

\[
11 + 2A + 5F + 6E + 3D + 2C = 19x
\]

\[
19x = 100, \text{ therefore } x = 5.26
\]

Because pair-wise comparisons always result in the lowest ranked criterion ending up with zero importance and often end with a total percentage not equal to 100%, these values were used as a baseline for which to round the values, as shown in Table 3.

Table 3. Results of Pair-wise Comparison

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Calculated</th>
<th>Rounded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elasticity: 6x</td>
<td>32%</td>
<td>30%</td>
</tr>
<tr>
<td>Flexibility: 5x</td>
<td>26%</td>
<td>25%</td>
</tr>
<tr>
<td>Difficulty of Manufacturing: 3x</td>
<td>16%</td>
<td>15%</td>
</tr>
<tr>
<td>Ability to Cover Hand: 2x</td>
<td>11%</td>
<td>10%</td>
</tr>
<tr>
<td>Cost: 2x</td>
<td>11%</td>
<td>10%</td>
</tr>
<tr>
<td>Individual Sensor Size: 1x</td>
<td>5%</td>
<td>5%</td>
</tr>
<tr>
<td>Resolution: 0x</td>
<td>0%</td>
<td>5%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>101%</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>
Using the weights shown in Table 3, each design was examined with respect to the design criteria as shown in the decision matrix in Table 4. First a weight was given to each of the design criteria, ranking their importance to the overall project. Next, each of the designs was given a raw score from 1-10, with 1 meaning poor and 10 meaning great. These raw scores were multiplied by the criteria weight, and all the weighted scores were totaled to give the design a value, with the highest value being the best overall design.

Table 4. Decision Matrix

<table>
<thead>
<tr>
<th>Design</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resolution</td>
<td>5%</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Individual Sensor Size</td>
<td>5%</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Ability to Cover Hand</td>
<td>10%</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Flexibility</td>
<td>25%</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Elasticity</td>
<td>30%</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Difficulty of Manufacturing</td>
<td>15%</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Cost</td>
<td>10%</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Raw Total</td>
<td>34</td>
<td>35</td>
<td>44</td>
<td>34</td>
<td>38</td>
</tr>
<tr>
<td>Weighted Total</td>
<td>450</td>
<td>495</td>
<td>575</td>
<td>370</td>
<td>485</td>
</tr>
</tbody>
</table>

After completing the decision matrix, Design C was deemed to be the best overall design both using raw values and using the weighted values from the pair-wise comparison. The small size of the piezoceramics would allow for the overall design to remain elastic, and their ease of manufacturing and low cost compared to other designs made them a clear favorite. Also, the use of piezoceramic disks in building a tactile sensor was not found in the literature review, thus it was deemed worthwhile research to conduct and test this type of tactile sensor. While it was initially planned to make a flat “skin” tactile sensor, working with Quality of Life Plus, Inc. (QL+) provided the challenge of giving the skin an application. QL+ is a nonprofit organization created to foster innovations that improve the quality of life for those injured in the line of duty. In order to ensure this research met that goal, the creation of a tactile sensor following the above design specifications for use as a prosthetic hand glove was begun.
3.1.2 Creating Realistic Skin Substrate

The next step in designing the tactile sensor was determining the method for creating a realistic covering. After talking with cosmetic prostheses experts as well as examining the state of the art in special effects, a material called Dragon Skin® which is a high-performance platinum cure silicone rubber was chosen, shown in Figure 49. By mixing an equal ratio by volume of Part A and Part B, it created a material with a Shore A Hardness of 10, tear strength of 102 pounds per linear inch and a possible elongation of up to 1000% [61].

![Figure 49. Dragon Skin High Performance Silicone Rubber [61]](image)

By adding a third equal part of Slacker® Tactile Mutator, a silicone rubber can be made which is softer and more “flesh-like,” with rebound properties even more human-like. Lastly, by adding a small quantity of Fleshtone Silc Pig® Silicone Pigment, a very realistic flesh-colored covering can be made [61].

3.1.4 Multiplexer Sampling

While a number of different methods for sampling the sensors were considered, a multiplexing technique was ultimately chosen. Using a set of 16:1 analog multiplexers (with breakout boards for ease of use, shown in Figure 50), up to 16 sensor signals could be read over a single analog data line using 4 digital addressing lines. At only $5 a piece, with a 10µs delay required between channel switches, multiplexers are a very affordable method of quickly reading a large number of sensors [10].
3.1.5 Arduino Microcontroller

To read the sensor signals from the multiplexer, a microcontroller had to be chosen. After researching available microcontrollers, the affordable Arduino Mega microcontroller board was chosen, shown in Figure 51, at roughly $60 each. Based on the ATmega1280 microchip, it runs on a 16 MHz crystal oscillator and has 54 digital input/outputs along with 16 analog inputs [11].

3.1.6 Sampling Design

Combining the 16 analog inputs, 54 digital inputs and a number of multiplexers, this design could in theory sample up to 208 sensors using 13 multiplexers, or an enormous number \(16^{13}\) of sensors using a cascading multiplexer scheme. It is important to note though, that this scheme would also require a huge number of multiplexers \(16^{12}\) and the sample rate per sensor would go down with each sensor or multiplexer added to the system.
As the analog read of the Arduino is limited in speed by the analog-to-digital-converter (ADC) clock (16 MHz) divided by a defined prescale factor (factory set to 128), the ADC runs at 125 KHz. Sampling takes 13 ADC clocks, thus the sample rate is 125 KHz/13, or roughly 9600 Hz (100µs per read). By setting the prescale factor to 16 instead of 128, the ADC could be made to run at 1 MHz and thus sample at roughly 77 KHz (13 µs per read). Though Atmel does not recommend running the ADC at frequencies above 200 KHz for optimum performance, they claim speeds up to 1 MHz do not reduce the ADC resolution or accuracy significantly [14].

3.1.7 Preliminary Testing and Design Modifications

Upon purchase of each of the above components, preliminary testing was performed to understand the behaviors of the components and characterize their performance, especially with the piezoceramics. After soldering wires to the piezoceramic disks as further illustrated in the Manufacturing section, initial tests for understanding their piezoelectricity were conducted first using a multimeter. By connecting one wire to the positive lead of the multimeter and one wire to the negative lead of the multimeter, then applying a load to the disk, a noticeable voltage difference was detected. In this arrangement, an applied load would result in a positive spike, and release of the load would result in a negative spike, as shown in Figure 52.

![Figure 52. Simplified Piezoceramic Behavior When Read by Multimeter](image)

Because of this, it was originally expected that the piezo signal would have to be integrated to determine the actual load being applied, most likely using a charge amplifier. In order to ensure that the piezo signal was proportional to the amount of load applied, a calibration
test was performed, the results of which are shown in Figure 53. In this figure, “Max” represents the maximum voltage from the positive spike, while “Min” represents the minimum voltage from the negative spike. As you can see, over a range of 0 to 20 Newton, the piezo behavior can be considered roughly linear.

![Figure 53. Piezoceramic Calibration](image)

In the next stage of testing, the piezo was connected to the analog input of a microcontroller, as shown in Figure 54. With one lead from the piezoceramic connected to an analog input and the second wire connected to the ground of the microcontroller, a very simple computer code was written to read the analog input and print that value to the serial port for examination.

![Figure 54. Piezo Connected to A/D Input Circuitry](image)
In this arrangement, the sensor behavior did not spike as in Figure 52, but rather was able to hold a fairly steady signal. In the previous test, the multimeter had read a relative voltage difference between the two sides of the piezo, resulting in both positive and negative signals being output. When read using the A/D of the microcontroller, and with one of the piezo electrodes grounded, only positive sensor signals were obtained as shown in Figure 55.

![Piezo Behavior When Read by Microcontroller](image)

**Figure 55. Piezo Behavior When Read by Microcontroller**

To verify this, experiments were run in which the two leads of the piezo were connected to different analog inputs, as opposed to one of them going to ground, and a difference was calculated between the analog inputs. By doing this, both positive and negative signals could again be obtained, but at the expense of using an additional analog input. With no obvious benefit in having both positive and negative as opposed to only positive signals, the setup shown in Figure 54 resulting in positive outputs only was chosen.

When read in this arrangement, the equivalent circuit diagram of the piezo and A/D circuitry of the microcontroller is shown in Figure 56. Using the simplified equivalent RC circuit, a time constant, $\tau$, (in seconds) can be found by multiplying the equivalent resistance (in ohms) by the capacitance (in farads), as shown in Equation 1. Using that time constant, a value for the 10% to 90% rise time, $\tau_r$, was found, shown in Equation 2. A rise time of roughly 3 microseconds was deemed short enough that it would not present a problem for accurate reading the sensors.
Figure 56. Equivalent Sensor Circuit Diagram

\[
\tau = RC = (100 \, \text{k}\Omega)(13.7 \, \text{pF}) = 1.37 \, \mu\text{s}
\]

Equation 1. Time Constant in an RC Circuit

\[
\tau_r = 2.2\tau = 2.2(1.37 \, \mu\text{s}) = 3.01 \, \mu\text{s}
\]

Equation 2. Rise Time (10% to 90%) in an RC Circuit

It was also determined that the change from a spiking behavior, which had been thought to be the natural behavior of the piezo as shown in Figure 52, to a behavior capable of holding a high DC voltage as shown in Figure 55, was due to differences in the amount of charge leakage between different sensors, and not a difference in the method of reading them. Charge leakage will continue to be discussed throughout this paper.

Next, a number of molds were made out of the silicone-rubber. A variety of simple molds were made by pouring the silicone rubber out on flat surface which produced flat-puddle like molds. It was examined that small bubbles formed in the mold when poured, but by mixing more smoothly and for longer, as well as poking the bubbles during the first few minutes of drying, they could be popped and filled naturally by the rubber. Initial tests were also made as to the necessary amount of flesh-toned silicone pigment to add in order to get a good-looking skin color, which turned out to be a very small amount. A number of samples were also poured over raised objects to see how the material would run due to gravity, the amount of detail produced, and the thickness of the dried mold, as shown in Figure 57.
A number of piezo sensors were then embedded in silicone rubber molds to test their behavior in that medium, as shown in Figure 58. It was at this time, in comparing a large number of sensors, that the extent of the differing behaviors were observed between the sensors. Roughly half of the sensors made were able to hold a high signal for 10+ seconds under a steady applied load, while the other half of sensors could only maintain a high signal for less than 1 second under a steady applied load, at which point the signal returned to zero.

As previously identified, this behavior is indicative of “charge leakage.” Several piezoceramic experts at Piezo Systems Inc. were then consulted, and two main solutions were suggested; 1) using a high-impedance operational amplifier to further reduce leakage to the electronics, and 2) to “clean, clean, clean!” Schematically, charge leakage is caused by charge dissipating through the internal insulation resistance of the piezo, shown in Figure 56. While values are typically on the order of $10^{12}$ Ω, small amounts of sweat, oil, flux, solder, or impurities
on the piezoceramic disk effectively decrease the internal insulation resistance. This decrease in resistance allows for charge to more easily “leak” from one side of the disk to the other, thus dissipating the voltage difference. Several steps were taken to maximize the cleanliness of the piezoceramics for forthcoming sensors, as talked about in the manufacturing section [26].

In order to make use of high-impedance operational amplifiers as suggested by the experts, a number of voltage followers were added to the design. By using a high-impedance op-amp, such as the seventy-seven cent TL032 shown in Figure 59, a simple voltage follower circuit could be used to detect the sensor signal while greatly reducing charge leakage to the electronics. With a current-input bias of 2pA and a sensor output voltage of roughly 5V, the TL032 resulted in an impedance of nearly $2.5 \times 10^{12}$ ohms.

Figure 59. TLE032 Op-amp

Because each sensor needed only one wire to be read and the other wire grounded to the microcontroller, it was thought that a single ground wire could be shared between multiple sensors, as shown in Figure 60. Several attempts were made to create sensors wired in this fashion and embedded into molds for testing; however it was found that the design worked better in theory than in actual practice. Soldering the sensors in such a fashion was found to be extremely difficult, both physically and with respect to the amount of charge leakage found in such sensors. In the end, soldering each sensor with its own separate ground wire was chosen as the most effective method, which would also reduce signal noise from the sensors.
3.1.8 Complete Design

With the components of the tactile sensor decided upon, the overall system design was set as shown in Figure 61. The next step was to begin manufacturing of the prototype tactile sensors.

---

Figure 60. Shared Ground Sensor Design

Figure 61. Complete System Design
3.2 Manufacturing

3.2.1 Sensor Manufacturing and Quality Testing

As mentioned above, the first step of manufacturing was to create the piezoelectric sensors from the purchased piezoceramic disks. To detect the voltage difference between the two sides of the piezoceramic, small flexible wires would have to be soldered to the disk. As directed by the piezoceramic dealer, a number of steps were taken to solder wires to each piezoceramic disk. First, the disks themselves were cleaned of any debris or oils using an ultrasonic cleaner. Next, a very small drop of #67 liquid flux from Superior Flux Mfg. & Co. was placed on the disk. Then, a small drop of solder was melted onto a soldering iron, which was lightly touched to the flux on the disk in order to transfer the solder to the disk, as shown in Figure 62.

![Figure 62. Adding Liquid Flux and Solder to One Side of Piezo Disk](image)

Next, a pre-tinned wire was dipped in the liquid flux, then placed on the soldered portion on the disk. By lightly touching the soldering iron to the top of the wire, the solder momentarily melted and joined with the pre-tinned solder of the wire, as shown in Figure 63.
The same steps were then taken on the reverse side of the piezo disk in order to attach a second wire to that side, as shown in Figure 64. Special care was taken to apply the soldering iron only lightly and for short durations of time in order to prevent the solder on the reverse side of the disk from melting, and allowing the first wire to disconnect. Also, after conversing with the piezo experts, it became clear that it would be necessary not to ever touch the piezoceramic disk or the connection portion of the wires with anything but clean plastic tweezers, and to always work on a flat surface cleaned with rubbing alcohol. Once soldering was finished, the sensors were then placed in an ultrasonic cleaner for 10 minutes.

In order to allow for the wire to easily deform as well as stretch, a simple method of zig-zagging the wire was performed, as shown in Figure 65. Lastly, an electronics header was attached to the end of each wire in order to allow for easy interface of the sensor to other electronics.
Even with the greatest efforts being made to ensure clean sensor manufacturing, each piezo sensor was tested for charge leakage after production without being connected to an op-amp. A testing rig was made which would allow a 1 kg mass to be easily placed on top of and removed from the sensor. Each sensor was tested by applying the 1 kg mass (9.8 Newton force) on the sensor for 4 seconds, removing the sensor, and then reapplying the mass for 0.5 seconds, shown with a dotted green line. Great disparity was found between sensors, as shown in Figure 66, with Figure 66(a) showing the expected and desired behavior. The sensor shown in Figure 66(a) was able to maintain a high voltage signal under a sustained force and returned to a resting voltage quickly upon removal of the force. The behaviors shown in Figure 66(b) and Figure 66(c) were not ideal, as those sensors were unable to maintain a high voltage signal under a sustained force, indicative of charge leakage. Calculations were performed to determine the discharge time constant (in seconds) of each of the sensors shown in Figure 66, each showing a different level of charge leakage. The discharge time constant is similarly calculated to the charge-up time constant, as shown in Equation 3, but disregards the capacitance of the piezoceramic and features the internal insulation resistance of the piezo in series with the A/D resistance, rather than in parallel.

\[ \tau = RC = (10^{12} \Omega)(14 \text{ pF}) = 14s \]

**Equation 3. Discharge Time Constant in an RC Circuit**

\[ \tau_r = 2.2 \tau = 2.2(14s) = 30.8s \]

**Equation 4. Fall Time (90% to 10%) in an RC Circuit**
With a typical internal insulation resistance of $10^{12} \, \Omega$, a discharge time constant of 14 seconds and a fall time of roughly 30 seconds can be expected. The sensor shown in Figure 66(b), with a fall time of roughly 4 seconds, was estimated to have an internal insulation resistance of roughly $1.3 \times 10^{11}$. The sensor shown in Figure 66(c), with a fall time of roughly 1 second, was estimated to have an internal insulation resistance of roughly $3.2 \times 10^{10}$.

![Figure 66. Testing and Comparison of 3 Piezo Sensors](image)
Even after taking the piezoceramic experts’ advice, a large number of the piezos made featured significant charge leakage, and as such, over 100 sensors were made in order to obtain 38 sensors with the desired behavior.

3.2.2 Silicone Rubber Molds

The next step was to create a realistic looking glove in which to embed the sensors. First, a mold of a hand was made using Alja-Safe molding gel. Then, liquid casting plaster was poured into the negative mold which resulted in a plaster-cast hand positive. By pouring silicone rubber out over the plaster-cast hand and allowing gravity to let the material run down, a hand-shaped glove was formed. By pouring several successive layers out over the casting and allowing ample time to fully dry, a realistic glove was made with enough thickness and strength for ensuing steps, as shown in Figure 67.

![Figure 67. Silicone Rubber Glove](image)

Through the association of this thesis with the QL+ Lab at Cal Poly, cosmetic prosthesis experts Dr. Bob Barron and Dr. Robert Erb were consulted in making the glove shown in Figure 67. An even more realistic thumb glove, shown in Figure 68, was made by Dr. Robert Erb, a cosmetic prosthesis expert from Pennsylvania, to fully show the possibilities of realistic coverings for this thesis. This sophisticated thumb was made from vacuum-forming silicone foam, which required seven coats of silicone with very detailed artistic painting taking place between coats.
3.2.3 Embedding Sensors

Using both the whole-hand silicone rubber glove as well as the thumb glove, two different prototypes were to be made serving as proofs-of-concept for the tactile sensor design. For the hand prototype, the plan was to use 30 sensors, as shown in Figure 69. Though many more sensors could theoretically be implemented using the electronics design chosen, 30 sensing points on the hand were deemed an acceptable level of resolution without overcomplicating the manufacturing process.

First, the silicone rubber glove was placed over the plaster casting of the hand to serve as a working platform. Then, one at a time sensors were placed onto their correct location and held in place using a small amount of rubber cement, or if more hold were required, silicone glue.
Once a number of sensors were in position, a coat of silicone rubber was poured onto the glove and allowed to dry. In subsequent steps, additional sensors were placed in their correct position, and again covered with a coat of silicone rubber. Lastly, an ample amount of baby powder was applied to the completed hand to coat the silicone and provide a smooth surface for a user’s hand, as shown in Figure 70.

![Figure 70. Embedding Sensors in Glove](image)

While it was originally planned to flip the glove right-side-out after manufacturing, the resulting appearance was not cosmetically satisfactory, thus it was decided to leave the glove inside-out, as will be further discussed in the Results section. For the thumb prototype, the plan was to embed 8 sensors as shown in Figure 71. Eight sensors were deemed to be an adequate proof-of-concept resolution for a single finger, and again, minimize difficulties in manufacturing.
As with the hand, sensors were placed in their correct position and held in place using a small amount of silicone glue. Once all 8 sensors were in position, a coat of silicone rubber was poured and allowed to dry. Several subsequent layers of silicone rubber were poured over the thumb to ensure the sensors and their wires would remain in place. An ample amount of baby powder was applied to the completed thumb, shown in Figure 72, to provide a smooth surface for the user. At this point, the thumb was turned right-side out. While the appearance was very satisfactory, several sensors came slightly loose during the flipping process. Therefore the thumb was flipped inside-out and another two layers of silicone rubber were applied. Then, with the sensors firmly secured, the thumb was again flipped right-side-out.

Figure 71. Planned Sensor Location (Thumb)

Figure 72. Embedding Sensors in Thumb Glove
3.2.4 Electronics

With the sensors embedded in the two silicone rubber gloves, the next step was wiring the electronics. For the hand prototype with its 30 sensors, 15 ICs (2 op-amps were IC) were required (each op-amp features 2 channels). Each op-amp requires a power line and a ground line from the microcontroller, the correct feedback line to create the voltage follower circuit, and a line going from each output of the op-amp. These 30 output lines traveled to two 16:1 Multiplexers (15 each), with the 16th input on the multiplexer being grounded to ensure the data was cleared from the ADC. The multiplexer also required a power line and a ground line from the microcontroller, 4 digital data lines (for choosing the right channel to read), and a signal line to one of the analog inputs of the microcontroller. A simplified wiring diagram for two sensors can be seen in Figure 73, with the complete electronics wiring for the hand-glove seen in Figure 74. For the thumb wiring, the exact same configuration was used, with only 4 op-amps and one multiplexer required.

![Figure 73. Simplified Wiring Layout](image-url)
3.2.5 Programming Arduino Microcontroller

To use the Arduino microcontroller, the Arduino open source software was used to write a computer program which was uploaded to the microcontroller. The Arduino software is a very simple and easy to understand language, yet offering many flexible libraries for advanced users. It is also capable of running on Windows, Macintosh OSX, and Linux operating systems. The Arduino development environment contains a text editor for writing code, a message area for examining errors, and a toolbar with buttons for common functions such as “Create New,” “Open,” “Save,” “Compile,” “Upload to I/O Board,” and “Serial Monitor” shown in Figure 75.

![Figure 74. Finished Electronics for Hand Prototype](image1)

![Figure 75. Arduino Programming Environment](image2)
A very simple program was then written for both prototypes to scan through each channel of the multiplexer/s, read that channel, and then print the sensor data to the serial port. In between each sensor read, a grounded analog input was read to clear the Arduino ADC. The Arduino Mega microcontroller was connected via USB to a computer, where a separate graphical display was programmed. A program flowchart is shown in Figure 76, with the full program for both the Thumb and Hand Tactile Sensors shown in their entirety in Appendix B.

![Figure 76. Simplified Arduino Program Flowchart](image)

3.2.6 Programming Computer Graphical Display

*Processing* is an open source programming language and environment meant to create images, animations, and interactions. Because the Arduino programming environment is based on the *Processing* programming environment, *Processing* was chosen to create a graphical user display. This graphical user display would enable viewing the data from each sensor in real-time using a computer screen. Several sample programs included in the Arduino programming environment provided instruction for how to read data from the serial port using Processing, and a multitude of tutorials on the *Processing* website provided instruction for drawing simple shapes for a graphical display. Among these simple commands, “point( ),” “line( ),” “rectangle( ),” and
“ellipse( ),” were used, as well as the commands needed for specifying size, draw color, fill color, and printing text.

Writing a very simple yet long computer program, a graphical user display was created for both the thumb prototype and the hand prototype, as shown in Figure 77 and Figure 78.

Figure 77. Thumb Prototype Graphical User Display
Figure 78. Hand Prototype Graphical User Display

As you can see, the raw value coming from each sensor is provided on the left in both displays. For the thumb display, that data is also shown in the form of a bar graph in the lower middle of the screen. Then, an image of the tactile sensor is provided with small ellipses drawn to show the location and value of each sensor. Each ellipse is one of 4 colors (red, green, blue, or black) with that ellipse corresponding to the sensor shown on the drawing of the tactile sensor. The ellipses then change transparency from clear to dark to show the magnitude of the currently applied load at that location. The Processing computer graphical display code can be seen in its entirety in Appendix C.

3.3 Testing and Results

As mentioned previously, the original plan was to flip the hand prototype right-side out following the completion of embedding sensors, shown in Figure 79. However, when flipped right-side out the glove showed several areas that were cosmetically unsatisfactory, as shown in
Figure 80. The majority of these areas consisted of wrinkles and folds in the glove caused by the method of manufacturing. When pouring successive layers of silicone rubber on the inside-out glove, small connections of silicone rubber were formed between the fingers. When turned right-side-out, these “bridges” caused areas of excess in the outer layer of the glove, and resulted in flaps and wrinkles.

Figure 79. Hand Prototype Right-Side-Out

Figure 80. Folds and Excess in Right-Side-Out Hand Prototype

When it came to turning the thumb prototype right-side out, this was not an issue. Because it was only one finger, there were no gaps between fingers across which a silicone rubber bridge could be formed. The thumb prototype thus looked exactly as planned once turned right-side out, as shown in Figure 81.
With the thumb prototype looking so realistic, it was debated whether it was truly necessary to turn the hand prototype right-side out, or if it could be left inside-out and thus show the wiring and sensors in more detail. Other possible solutions would be to add additional layers of flesh-toned silicone rubber to the inside-out hand prototype, thus changing the inside to the outside and attempting to recreate a new life-like exterior, or to cover the prototype with an outer glove (latex perhaps). Though all these are still possibilities, at this time, the hand prototype was left inside out, as shown with the electronics attached in Figure 82. Likewise, the thumb prototype with its attached electronics is shown in Figure 83.
It was discovered that when putting the thumb prototype on an actual thumb, shown in Figure 84, the inner volume of the thumb had decreased significantly. This decrease was enough to cause a very tight fit, which had not been the case prior to embedding sensors. The increased snugness of the fit resulted in problems when attempting to read the sensors while wearing the prototype, as the applied force on the inside of the glove from the user was picked up by the sensors without any external force being applied. This problem could easily be overcome by creating an oversized initial thumb, or by fitting the finished glove over a smaller user’s thumb.
With both prototypes completed and connected to their electronics, testing was conducted to verify the sensors were being scanned at a high rate (enough to be considered real-time), as shown in Table 5. By connecting the tactile sensors to an oscilloscope, it was found that only 2.2 ms were taken to read each sensor. With 8 sensors on the thumb prototype, this resulted in each sensor being read once every 17.6 ms, or roughly 57 times per second. With 30 sensors on the hand prototype, each sensor was read once every 66 ms, or roughly 15 times per second. For use in a tactile sensor, values in this range can safely be considered real-time, that is, sufficiently fast to rapidly be able to respond to external forces.

<table>
<thead>
<tr>
<th># of Sensors</th>
<th>Time to Sweep all Sensors (ms)</th>
<th>Number of Reads per Sensor per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.2</td>
<td>~ 455</td>
</tr>
<tr>
<td>8</td>
<td>17.6</td>
<td>~ 57</td>
</tr>
<tr>
<td>30</td>
<td>66.0</td>
<td>~ 15</td>
</tr>
</tbody>
</table>

Perhaps most importantly, both prototypes proved to be very flexible as well as elastic. The silicone rubber and zig-zagged wiring scheme for the tactile sensors allowed the prototypes to easily conform to a variety of different objects while maintaining their original shape and realistic look. The elasticity of the prototypes was proven by performing a stretch test, shown in Figure 85, in which the prototypes were found capable of stretching more than 75% with no harm being done to them. This sort of flexibility and elasticity allows the sensor design to be used in a wide variety of applications previously not possible with other tactile sensors.

Figure 85. Elastic Stretch Test
With the scanning method via multiplexers verified, as well as the flexibility and elasticity of the prototypes, the sensing behavior of the piezoceramics was then tested. Several peculiar behaviors were found. The first was that the sensor values in the prototypes exhibited a tendency to slowly rise over time under no applied force. It was observed however that when the prototypes were lifted up off display boards, the sensor values would drop significantly. This led to the belief that the cast acrylic display boards might accumulate static electricity. More expensive static-dissipative cast acrylic display boards were then purchased and implemented, which effectively solved the problem. Unfortunately, this showed that the piezoceramic sensors could be significantly affected by stray charge.

Next, in testing the accuracy and repeatability of the sensors, it was found that the sensor signal from the piezos would progressively rise under successive applications of the same force as generalized in Figure 86. In testing this behavior, a force of 10 Newton was applied to one of the sensors on the prototype for duration of roughly 0.2 seconds and then removed (shown in red). One second later, the same force was reapplied for the same duration, removed, and repeated multiple times. The sensor signal (blue), which had given repeatable results for the first few forces, became noticeably larger by the 4th application of the force, and by the 8th application, the sensor was effectively unusable. By around the 10th application of force, the sensor signal would no longer drop under removal of the force, remaining at its maximum value. 60+ seconds of rest were then required to get the sensor signal back to zero.
Even more unexpectedly, this signal behavior was seen in other sensor signals, sensors that had not experienced an applied force. For example, if a force were to be repeatedly applied to sensor 1, as the sensor 1 signal started to max-out, other sensor signals (from sensors that had not experienced an applied force) also started to rise and eventually max-out. Because the sensors were individually wired, ran through their own op-amps, and both the multiplexer and Arduino Mega board read a ground signal in between each sensor reading, the sensor signals should have been completely isolated from each other. It was hypothesized that the cause was “free charge,” created by the piezoceramics themselves, traveling across the prototype and being picked up by other sensors.

Additional testing was performed to attempt to eliminate this unwanted behavior. In doing so, an alternative to waiting 60+ seconds for the sensor signals to return to zero was found. By physically grounding the signal line of each sensor, the signal of that sensor would return to zero, effectively resetting the sensor. After being reset, the sensor could again accurately measure applied forces (for a limited number of times). Initially, because the electronics design featured no method for grounding each signal, it had to be done by hand using a small piece of copper wire to ground each sensor. In order to better implement this method of grounding, several designs were considered which would add the electronic components necessary for
grounding the sensors quickly and electronically. Each of these designs would work in a fashion similar to the one shown in Figure 87. In these designs, the relay or transistor would regulate the flow of charge away from the sensors. When open, the system would keep the signal from each sensor isolated (via diodes) and not allow the charge to escape. When closed, it would allow the charge from each of the sensors to be grounded, using only a single control line to operate the additional electronics.

**Figure 87. Sensor Grounding Electronics Design**

It was hoped that in applying this design and quickly resetting the sensors at a given interval, the issue of progressive signal increase would be eliminated, as each sensor would quickly be reset in between applied forces, or quickly return to the correct signal under a previously applied force. Once implemented however, it was found that once a sensor was reset, it took several seconds before the sensor could quickly respond to any applied forces. The same behavior was seen with a sensor under load prior to being reset; it would take several seconds before the signal was able to climb back to its previous value, as shown in Figure 88.
This slow response from the sensor after being reset is thought to be due to the physical nature of the piezo material itself. Piezoelectric materials consist of a number of polarized molecules, as simplified in the model shown in Figure 89. When a piezo material is deformed under an applied load, its polarized molecules rearrange and create a voltage difference across the two sides of the material (which tries to return the material to its original shape). When an electric field is applied across the material, the polarized molecules rearrange to match that electric field (thus deforming the material).

One possible explanation for the delayed-response behavior seen after grounding involves the inability of the piezo to quickly turn from a transducer to a generator. When the piezo is physically grounded, an electric potential difference of 0 volts is applied across the material and the molecules rearrange (if needed) to accommodate that field. When that field is removed, the molecules of the piezo are unable to instantaneously rearrange to accommodate
applied forces. Another possible explanation is that charge from the piezo actually escapes when grounded. This would result in the sensor having to re-absorb charge from the surroundings before returning to neutral, and then being able to create a voltage difference under applied loads.

One other significant difficulty was found while testing the prototypes dealing with their behavior while being worn. As mentioned previously, the finished thumb prototype resulted in quite a snug fit, causing all sensor readings to be influenced by the pressure on the inside of the glove rather than forces applied to the outside. In wearing the hand prototype, an over-snug fit was not an issue, yet, the sensor signals still rose until maxed-out under no applied forces at all, as shown in Figure 90. When not worn or under applied forces, the sensor signals were able to remain at zero as was expected, shown in Figure 78.

![Hand Prototype Graphical User Display](image)

**Figure 90. Hand Prototype Behavior While Worn**

The theorized cause of this behavior is once again “free charge,” this time from the person’s hand. The human body naturally collects static electricity, or “free charge,” from a variety of sources, as well as producing a small electric field itself. It appears that the piezoelectric elements are able to detect this free charge, thus skewing the sensor signals. When
the sensors were grounded, the signals would return to zero as expected, however, would quickly rise again until maxed-out within a few short seconds.

An attempt was therefore made to counter the static electricity build-up through programming. By grounding a sensor and then measuring the slope of the static electricity rise over time, and then subtracting that value in the programming from the sensor reading each sweep, it was hoped that an accurate sensor reading could be attained. An exaggerated model of this theory can be seen in Figure 91.

![Figure 91. Method for Countering Static Electricity via Programming](image)

However, in testing this method, it became apparent that because the sensor itself was gaining charge from the static electricity, it would not work once maxed out, regardless of what the programming did with the value. Once maxed-out, the sensor would be unable to detect applied forces, even if the programming (falsely) displayed that the sensor was at rest. This was confirmed as shown in Figure 92, where the display showed several sensors at rest, and the majority of sensors not maxed-out. Even though the display showed these sensors as not maxed-out, the sensors in actuality were, and were thus unable to change signal when a force was applied. None of the sensors were able to respond to external forces with this method applied to the prototypes while being worn.

76
Though these problems did limit the number and complexity of tests that could reliably be run, several important results were still obtained. By grounding the sensors on the thumb prototype, waiting 5 seconds, then holding the thumb inside an electronics static shield bag and pressing it against a surface, the ability of the prototype to show different types of contact was still possible. As shown in Figure 93, the thumb prototype was pressed on its tip into a hard table with roughly 12 Newton of force, resulting in only the two sensors on the tip of the thumb producing a high sensor signal (about 400 A/D counts, or 1.95 volts).
When pressed with a 16 Newton force into the same hard table, this time with the entire thumb-print area, significantly higher signal values (about 600 A/D counts) were obtained by the 4 sensors on that area of the thumb, as shown in Figure 94.
When not pressed into a table, but rather held freely and then gripped from behind, only the sensor on the back of the thumb showed a high signal, as shown in Figure 95. Though each of these tests had to be set up correctly beforehand, they show that the thumb prototype is very capable of differentiating between different magnitudes and locations of applied force.

Likewise, by grounding each of the sensors on the hand prototype, waiting 5 seconds, then applying different types of force, usable data could be obtained. For these tests, the hand was not worn, but was instead gripped around the outside with an insulating piece of static-shield bag between the user’s hand and the glove. As shown in Figure 96, very clear data can be obtained by touching the tip thumb to the tip of the pinky. Several other sensors, such as the middle of the pinky and the base of the thumb, experienced small forces, likely due to bending in the silicone rubber glove. When the prototype was gripped around the thumb, as shown in Figure 97, nearly all of the sensors embedded around the thumb showed a strong force applied.
Perhaps most importantly, when the prototype was gripped around a cup, as shown in Figure 98, the graphical user display shows that an applied force is being read by many of the interior sensors on the fingers. Not only does this allow for detailed information regarding the
type of object being held, but this information could also be used to adjust the grip. For example, as shown in Figure 98, both the tip and base of the ring finger are experiencing no force. Thus, the ring finger could be closed more tightly in order to provide a more even grip on the object.

![Figure 98. Hand Prototype Grabbing a Cup](image)

### 3.4 Further Testing and Research Required

As mentioned in the Testing and Results section, the prototypes were hampered by the piezoelectric sensors’ propensity to pick up stray charge. This remains the most significant area where research is still required, as eliminating stray charge from the sensor readings must be done before the sensors can function in real-world applications. Several work-arounds were attempted, but none proved to be viable solutions. Possible fixes might include coating the piezoelectric sensors in a static-free case prior to embedding them in the silicone rubber, or purchasing commercial piezoelectric sensors which have overcome this problem.
Assuming the method of creating piezoelectric sensors as performed in this thesis is continued, the next most important area where further research is required is to fully understand charge leakage, and ensure consistent behavior between sensors during manufacturing. It is believed that in a more structured manufacturing environment, such as a cleanroom, discrepancies in sensor behavior could be eliminated, but this is yet to be verified. The ability to ensure consistent behavior between manufactured sensors is a crucial step in piezoelectrics’ use as small force sensors.

As discussed earlier, the sensor behavior was calibrated over a range of force values and found to act linearly from 0 to 20 Newton. However, when read by a microcontroller, significant applied forces caused the sensors to reach 5V, thus maxing out the ADC of the microcontroller. Under an applied force of sufficient magnitude, it was seen that some sensors would max-out and remain above 5V for tens of seconds, even after removal of the force. For future implementation of this design, it should be examined whether using an ADC with greater voltage range is beneficial to observing the sensor behavior, or if a voltage of 5V (relating to 20 Newton) is a sufficient maximum value.

Lastly, additional testing needs to be done on the endurance of the tactile sensors under repeated use, as well as the durability of the sensor against dirt, sweat, and impact forces. As the expected application of these tactile sensors would result in substantial use, the sensor must show equally impressive durability.

3.5 Conclusion

With robots becoming ever more common in society, the need for cost-effective tactile sensors is greatly increasing. By combining tactile sensors with other sensors commonly used by robots to structure their environment, the variety and quality of tasks capable by robots will greatly increase, enabling them to act as capable human-companions or aides. In order for these robots to be accepted into society, as well as be able to perform the types of tasks and use the
same tools that humans do, they will need to look and perform similarly to humans; specifically their sense of touch.

Also benefiting from the advances made in electronics, current research is developing the next generation of electromechanical prostheses. These prostheses are returning the use of limbs to people who have lost them, and soon the biological-technological connection required to convert electronic sensor data to nerve impulses will be perfected. Advanced tactile sensors might one day return the sense of touch to those who have lost it.

In this thesis, two prototype artificial skin tactile sensors were created. First, a large number of small piezoelectric sensors were manufactured. With their wires zig-zagged, the sensors were then embedded into a flexible and stretchable silicone rubber, allowing for the overall tactile sensor design to also stretch and bend. These sensors were buffered through a number of operational amplifiers, and sampled using a multiplexing scheme by a microcontroller. From there, the sensor readings were used to create a graphical user display which showed the sensor data in real time.

While additional research is required to perfect these designs, the prototypes successfully serve as proofs-of-concept for a method of reading a large number of small piezoelectric sensors embedded in a realistic-looking silicone rubber glove. With controlled production and the elimination of stray charge affecting the sensors, it is predicted that the behavior of the tactile sensors could be greatly improved upon, ensuring identical, repeatable, and accurate behavior from all sensors.
Bibliography


http://www.technologyreview.com/biomedicine/16970/?a=f


Capacitive Sensors. (2010). Retrieved 2010 June, from Lion Precision:


Appendices

A. Additional Information about Piezoelectricity

Because the field of piezoelectricity is so broad yet rarely studied in relation to electronics and robotics, additional information is provided here about the subject. This background section focuses on the aspects and features of piezoelectrics as they were used in this thesis, which will be described below.

Piezoelectricity was first discovered by Pierre and Jacques Curie in 1880 when they found electric surface charges appearing on specially prepared crystals (tourmaline, quartz, topaz, cane sugar, and Rochelle salt among them) when subjected to a mechanical stress. Over the next 40 years, research involving piezoelectricity continued, and during World War 1, an ultrasonic submarine detector (sonar) was invented by P. Langevin by gluing a mosaic of thin quartz crystals between two steel plates. By emitting a high frequency “chirp” and measuring the time until the return echo, they were able to determine the depth of enemy submarines. Since that first application, piezoelectrics have been used for countless other purposes including microphones, accelerometers, and phonograph pick-ups [53].

Scientifically, the relationship between applied forces and the resultant responses in piezoelectric materials is caused by the atomic lattice structures of the material. In piezoelectric materials, this lattice structure has an essential unit or “cell” of a cubic or rhomboid cage made of atoms. Inside this cell is a single semi-mobile ion which has several stable quantum states. This post-ion state of the ion can be caused to shift by either deforming the cage through applied strain or by applying an electric field [53].

The relationship between applied force and the resultant response depends on the piezoelectric properties of the ceramic, the size and shape of the piece, and the direction of the electrical or mechanical excitation. As shown in Figure 99, the polar axis (3) is defined as being parallel to the direction of polarization within the ceramic. It was excitation in this direction that
was used in this thesis, as opposed to excitation along the 1 or 2 axes which produce different responses. In manufactured piezoceramics, the polarization direction is established by applying a high DC voltage between a pair of electrode faces to activate the material [53].

![Piezoelectric Polarization Direction](image)

**Figure 99. Piezoelectric Polarization Direction [53]**

In order to characterize a piezo, a number of property coefficients are defined, many of which feature single or double subscripts that link electrical and mechanical quantities. The first of these subscripts indicates the direction (as defined in Figure 99) of the electrical field associated with the voltage applied, or charge produced. The second subscript gives the direction of the mechanical stress or strain [53]. In relation to this thesis, the most important property constant to discuss is the voltage coefficient, “G,” which is used to relate the electric field produced by the mechanical stress, having units of volts/meter per Newton/square meter. The expected output voltage can be calculated by multiplying the electric field by the thickness of the ceramic between electrodes. Conversely, the voltage coefficient can also be seen as the ratio of strain developed to the applied charge density with units of meter per meter over coulombs per square meter. A 33 subscript indicates that the electric field and the mechanical stress are both along the polarization axis, as shown in Figure 100. For sensors, high G constants are sought after as they indicate a large voltage output per unit of applied stress [53].
Also important, the charge coefficient, “D,” relates the mechanical strain produced by the applied electric field, or conversely, the charge collected on the electrodes due to an applied mechanical stress. D_{33} applies when the force is in the 3 direction (along the polarization axis) and is impressed on the same surface which the charge is collected, as shown in Figure 100. The units for the D_{ij} coefficients are commonly expressed as coulombs/square meter per Newton/square meter. When the force that is applied is distributed over an area which is fully covered by electrodes, the units of area cancel, and the units become coulombs per Newton [53].

Some piezoelectric material constants are also written with superscripts, which specify either a mechanical or electrical boundary condition. These can be T, signifying constant stress/mechanically free, E, signifying constant field/short circuit, D, signifying constant electrical displacement/open circuit, and S, signifying constant strain/mechanically clamped. For example, K_T^3 represents the dielectric constant (K) measured in the polar direction (3) with no mechanical clamping applied (T). The dielectric constant is defined as the ratio of the permittivity of the material, \( \varepsilon \), to the permittivity of free space, \( \varepsilon_0 \), in the unconstrained condition (well below the mechanical resonance of the part) [53].

Several other constants are used when talking about piezoelectrics. The capacitance, which unlike the dielectric constant depends on the quantity, type and dimensions of the material, is calculated as the product of the dielectric constant, K, the permittivity of free space, \( \varepsilon_0 \) (8.9 \times 10^{-12} \text{ farads/meter}), and the electrode surface area, A, divided by the thickness separating the electrodes, t, as shown in Equation 5 [53].
\[ C = \frac{K \varepsilon_0 A}{t} \]

**Equation 5. Piezoelectric Capacitance [53]**

At frequencies far below resonance, piezoelectric ceramic transducers are fundamentally capacitors. Because of this, the voltage coefficient \( G \) is related to the charge coefficient \( D \) by the dielectric constant \( K \) as shown in Equation 6 [53].

\[ d_{33} = K^r \varepsilon_0 \varepsilon_{33} \]

**Equation 6. Piezoelectric Capacitance Relationship [53]**

Using the piezoelectric coefficients as described above, the final charge (\( Q \)) and voltage (\( V \)) can be found using Equation 7 and Equation 8 [53].

\[ Q = FD_{33} \]

**Equation 7. Piezo Charge Equation [53]**

\[ V = \frac{FTG_{33}}{LW} \]

**Equation 8. Piezo Voltage Equation [53]**

Though it was not greatly studied for this thesis, piezoelectric materials are also pyroelectric. Pyroelectricity is the characteristic of some materials to produce electric charge as they undergo a temperature change. As their temperature increases, a voltage develops having the same orientation as the polarization voltage. This electric field can be calculated as shown in Equation 9, where \( E \) is in volts/meter, \( \alpha \) is the pyroelectric coefficient in coulomb/°C·meter\(^2\), \( \Delta T \) is the temperature different in °C, \( K_3 \) is the dielectric constant, and \( \varepsilon_0 \) is the dielectric permittivity of free space. Because the temperature change in this project was negligible, the pyroelectric effects were ignored [53].

\[ E(\text{pyro}) = \frac{\alpha(\Delta T)}{K_3 \varepsilon_0} \]

**Equation 9. Pyroelectric Relationship [53]**

Lastly, a few general notes about piezos. When working with or using piezos, it should be kept in mind that piezo elements are very fragile and can be shattered by sudden forceful
impacts. The small size of piezo elements used for this thesis reduced the chance of them being broken, but did not remove the possibility. Also, piezos provide very repeatable results, and once calibrated, can be used as accurate force sensors for years of service. The ceramics used in this thesis used a thin layer of nickel on the sides of the ceramics as electrodes, which required the use of a special liquid flux to ensure uniform soldering during manufacturing [53].
B. Arduino Code

1. Thumb_Prototype

```c
/*
 * Thumb Prototype Arduino Code
 * - Reads the 8 sensors embedded in the artificial skin
 * thumb prototype using a single analog input on the
 * Arduino MEGA board which comes from a 16:1 analog
 * multiplexer. Digital pins 50-53 are used to send the
 * correct signal to the Mux, choosing which of the eight
 * sensors will be read. Each sensor attached to the mux
 * is read sequentially, also reading pin 15 on the mux
 * (attached to ground) to clear any old signals out of the
 * A/D. After each value is read, analog input 15 on
 * the Arduino MEGA board is also read (attached to ground)
 * in order to clear any old signals out of the A/D. Prints
 * the analog values to the serial port, separated by a
 * comma and ending each sensor sweep with a newline.

Written by Ross Miller
Mechanical Engineering Master's Thesis
California Polytechnic State University
11/3/2010
*/

// 8 sensors read one at a time
int numSensors = 8;
int zero = 0;

int muxOutput = 8;   // Arduino analog input pin for Mux
int groundPin = 15;  // Arduino grounded analog input pin

// Note: Verify mux signal wires go to correct digital pins
int signalPin0 = 53;
int signalPin1 = 51;
int signalPin2 = 52;
int signalPin3 = 50;

void setup()
{
    Serial.begin(9600);

    // Set signal pins as outputs
    pinMode(signalPin0, OUTPUT);
    pinMode(signalPin1, OUTPUT);
    pinMode(signalPin2, OUTPUT);
    pinMode(signalPin3, OUTPUT);
}

void loop()
{
    int sensors[numSensors];

    for (int i = 0; i < numSensors; i++)
    {
```
// clear Arduino A/D
zero = analogRead(15);

// Sensor 0
if(i == 0)
{
  // Set digital signal to 0000 = signal 0
digitalWrite(signalPin0, LOW);
digitalWrite(signalPin1, LOW);
digitalWrite(signalPin2, LOW);
digitalWrite(signalPin3, LOW);
}

// Sensor 1
else if(i == 1)
{
  // Set digital signal to 0001 = signal 1
digitalWrite(signalPin0, HIGH);
digitalWrite(signalPin1, LOW);
digitalWrite(signalPin2, LOW);
digitalWrite(signalPin3, LOW);
}

// Sensor 2
else if(i == 2)
{
  // Set digital signal to 0010 = signal 2
digitalWrite(signalPin0, LOW);
digitalWrite(signalPin1, HIGH);
digitalWrite(signalPin2, LOW);
digitalWrite(signalPin3, LOW);
}

// Sensor 3
else if(i == 3)
{
  // Set digital signal to 0011 = signal 3
digitalWrite(signalPin0, HIGH);
digitalWrite(signalPin1, HIGH);
digitalWrite(signalPin2, LOW);
digitalWrite(signalPin3, LOW);
}

// Sensor 4
if(i == 4)
{
  // Set digital signal to 0100 = signal 4
digitalWrite(signalPin0, LOW);
digitalWrite(signalPin1, LOW);
digitalWrite(signalPin2, HIGH);
digitalWrite(signalPin3, LOW);
}

// Sensor 5
if(i == 5)
{
  // Set digital signal to 0101 = signal 5
digitalWrite(signalPin0, HIGH);
digitalWrite(signalPin1, LOW);
digitalWrite(signalPin2, HIGH);
digitalWrite(signalPin3, LOW);
}

// Sensor 6
if (i == 6) {
    // Set digital signal to 0110 = signal 6
    digitalWrite(signalPin0, LOW);
digitalWrite(signalPin1, HIGH);
digitalWrite(signalPin2, HIGH);
digitalWrite(signalPin3, LOW);
}

// Sensor 7
if (i == 7) {
    // Set digital signal to 0111 = signal 8
    digitalWrite(signalPin0, HIGH);
digitalWrite(signalPin1, HIGH);
digitalWrite(signalPin2, HIGH);
digitalWrite(signalPin3, LOW);
}

// Steps Common to Every Mux Reading
// Read mux and print sensor value to serial
sensors[i] = analogRead(muxOutput);
Serial.print(sensors[i]);

// Set mux to 1111 = signal 15
digitalWrite(signalPin0, HIGH);
digitalWrite(signalPin1, HIGH);
digitalWrite(signalPin2, HIGH);
digitalWrite(signalPin3, HIGH);

zero = analogRead(muxOutput);  // clear mux A/D
zero = analogRead(groundPin);  // clear Arduino A/D

if (i < (numSensors-1)) {
    Serial.print(",");
}
Serial.println();}
2. Hand Prototype

/*
Hand Prototype Arduino Code
- Reads the 30 sensors embedded in the artificial skin
hand prototype using two analog inputs on the Arduino
MEGA board which come from a pair of 16:1 analog
multiplexers. Digital pins 46 to 49 are used to select
the mux 1 signal, and digital pins 50-53 are used to
select the mux 2 signal, choosing which of the fifteen
sensors will be read. The sensors attached to
the muxes are read sequentially, also reading pin 15 on
the mux (attached to ground) to clear any old signals out
of the mux A/D. After each value is read, analog input 15 on
the Arduino MEGA board is also read (attached to ground) in
order to clear any old signals out of the Arduino A/D. Prints
the analog values to the serial port separated by a comma and
ending each sensor sweep with a new line.

Written by Ross Miller
Mechanical Engineering Master's Thesis
California Polytechnic State University
11/3/2010
*/

// 30 sensors read two at a time
int numMux1Sensors = 15;  // 0 to 14
int numMux2Sensors = 15;  // 0 to 14
int numReads = 15;
int zero = 0;

int mux1Output = 0;  // Arduino analog input pin for Mux 1
int mux2Output = 8;  // Arduino analog input pin for Mux 2
int groundPin = 15;  // Arduino grounded analog input pin

// Note: Verify mux signal wires go to correct digital pins
// Mux 1, sensors 0-14 + 1 ground
int mux1s0 = 43;
int mux1s1 = 41;
int mux1s2 = 42;
int mux1s3 = 40;

// Mux 2, sensors 0-14 + 1 ground
int mux2s0 = 53;
int mux2s1 = 51;
int mux2s2 = 52;
int mux2s3 = 50;

void setup()
{
    Serial.begin(9600);

    // Set Mux 1 signal pins as outputs
    pinMode(mux1s0,OUTPUT);
    pinMode(mux1s1,OUTPUT);
    pinMode(mux1s2,OUTPUT);
pinMode(mux1s3, OUTPUT);

// Set Mux 2 signal pins as outputs
pinMode(mux2s0, OUTPUT);
pinMode(mux2s1, OUTPUT);
pinMode(mux2s2, OUTPUT);
pinMode(mux2s3, OUTPUT);
}

void loop()
{
    int mux1sensors[numMux1Sensors];
    int mux2sensors[numMux2Sensors];

    for (int i = 0; i < numReads; i++)
    {
        // clear Arduino A/D
        zero = analogRead(groundPin);

        // Sensors 0 and 0
        if (i == 0)
        {
            // Set Mux 1 to 0000 = signal 0
            digitalWrite(mux1s0, LOW);
            digitalWrite(mux1s1, LOW);
            digitalWrite(mux1s2, LOW);
            digitalWrite(mux1s3, LOW);

            // Set Mux 2 to 0000 = signal 0
            digitalWrite(mux2s0, LOW);
            digitalWrite(mux2s1, LOW);
            digitalWrite(mux2s2, LOW);
            digitalWrite(mux2s3, LOW);
        }

        // Sensors 1 and 1
        if (i == 1)
        {
            // Set Mux 1 to 0001 = signal 1
            digitalWrite(mux1s0, HIGH);
            digitalWrite(mux1s1, LOW);
            digitalWrite(mux1s2, LOW);
            digitalWrite(mux1s3, LOW);

            // Set Mux 2 to 0001 = signal 1
            digitalWrite(mux2s0, LOW);
            digitalWrite(mux2s1, HIGH);
            digitalWrite(mux2s2, LOW);
            digitalWrite(mux2s3, LOW);
        }

        // Sensors 2 and 2
        if (i == 2)
        {
            // Set Mux 1 to 0010 = signal 2
            digitalWrite(mux1s0, LOW);
            digitalWrite(mux1s1, HIGH);
            digitalWrite(mux1s2, LOW);
            digitalWrite(mux1s3, LOW);

            // Set Mux 2 to 0010 = signal 2
            digitalWrite(mux2s0, LOW);
            digitalWrite(mux2s1, LOW);
            digitalWrite(mux2s2, HIGH);
            digitalWrite(mux2s3, LOW);
        }
    }
}
digitalWrite(mux1s2, LOW);
digitalWrite(mux1s3, LOW);

// Set Mux 2 to 0010 = signal 2
digitalWrite(mux2s0, LOW);
digitalWrite(mux2s1, HIGH);
digitalWrite(mux2s2, LOW);
digitalWrite(mux2s3, LOW);
}

// Sensors 3 and 3
if (i == 3) {

    // Set Mux 1 to 0011 = signal 3
digitalWrite(mux1s0, HIGH);
digitalWrite(mux1s1, HIGH);
digitalWrite(mux1s2, LOW);
digitalWrite(mux1s3, LOW);

    // Set Mux 2 to 0011 = signal 3
digitalWrite(mux2s0, HIGH);
digitalWrite(mux2s1, HIGH);
digitalWrite(mux2s2, LOW);
digitalWrite(mux2s3, LOW);
}

// Sensors 4 and 4
if (i == 4) {

    // Set Mux 1 to 0100 = signal 4
digitalWrite(mux1s0, LOW);
digitalWrite(mux1s1, LOW);
digitalWrite(mux1s2, HIGH);
digitalWrite(mux1s3, LOW);

    // Set Mux 2 to 0100 = signal 4
digitalWrite(mux2s0, LOW);
digitalWrite(mux2s1, LOW);
digitalWrite(mux2s2, HIGH);
digitalWrite(mux2s3, LOW);
}

// Sensors 5 and 5
if (i == 5) {

    // Set Mux 1 to 0101 = signal 5
digitalWrite(mux1s0, HIGH);
digitalWrite(mux1s1, LOW);
digitalWrite(mux1s2, HIGH);
digitalWrite(mux1s3, LOW);

    // Set Mux 2 to 0101 = signal 5
digitalWrite(mux2s0, HIGH);
digitalWrite(mux2s1, LOW);
digitalWrite(mux2s2, HIGH);
digitalWrite(mux2s3, LOW);
}
// Sensors 6 and 6
if (i == 6) {
    // Set Mux 1 to 0110 = signal 6
    digitalWrite(mux1s0, LOW);
    digitalWrite(mux1s1, HIGH);
    digitalWrite(mux1s2, HIGH);
    digitalWrite(mux1s3, LOW);

    // Set Mux 2 to 0110 = signal 6
    digitalWrite(mux2s0, LOW);
    digitalWrite(mux2s1, HIGH);
    digitalWrite(mux2s2, HIGH);
    digitalWrite(mux2s3, LOW);
}

// Sensors 7 and 7
if (i == 7) {
    // Set Mux 1 to 0111 = signal 7
    digitalWrite(mux1s0, HIGH);
    digitalWrite(mux1s1, HIGH);
    digitalWrite(mux1s2, HIGH);
    digitalWrite(mux1s3, LOW);

    // Set Mux 2 to 0111 = signal 7
    digitalWrite(mux2s0, LOW);
    digitalWrite(mux2s1, HIGH);
    digitalWrite(mux2s2, HIGH);
    digitalWrite(mux2s3, LOW);
}

// Sensors 8 and 8
if (i == 8) {
    // Set Mux 1 to 1000 = signal 8
    digitalWrite(mux1s0, LOW);
    digitalWrite(mux1s1, LOW);
    digitalWrite(mux1s2, LOW);
    digitalWrite(mux1s3, HIGH);

    // Set Mux 2 to 1000 = signal 8
    digitalWrite(mux2s0, HIGH);
    digitalWrite(mux2s1, LOW);
    digitalWrite(mux2s2, LOW);
    digitalWrite(mux2s3, HIGH);
}

// Sensors 9 and 9
if (i == 9) {
    // Set Mux 1 to 1001 = signal 9
    digitalWrite(mux1s0, HIGH);
    digitalWrite(mux1s1, LOW);
    digitalWrite(mux1s2, LOW);
    digitalWrite(mux1s3, HIGH);
// Set Mux 2 to 1001 = signal 9
digitalWrite(mux2s0, HIGH);
digitalWrite(mux2s1, LOW);
digitalWrite(mux2s2, LOW);
digitalWrite(mux2s3, HIGH);
}

// Sensors 10 and 10
if (i == 10)
{
    // Set Mux 1 to 1010 = signal 10
    digitalWrite(mux1s0, LOW);
digitalWrite(mux1s1, HIGH);
digitalWrite(mux1s2, LOW);
digitalWrite(mux1s3, HIGH);
    // Set Mux 2 to 1010 = signal 10
    digitalWrite(mux2s0, LOW);
digitalWrite(mux2s1, HIGH);
digitalWrite(mux2s2, LOW);
digitalWrite(mux2s3, HIGH);
}

// Sensors 11 and 11
if (i == 11)
{
    // Set Mux 1 to 1011 = signal 11
    digitalWrite(mux1s0, HIGH);
digitalWrite(mux1s1, HIGH);
digitalWrite(mux1s2, LOW);
digitalWrite(mux1s3, HIGH);
    // Set Mux 2 to 1011 = signal 11
    digitalWrite(mux2s0, HIGH);
digitalWrite(mux2s1, HIGH);
digitalWrite(mux2s2, LOW);
digitalWrite(mux2s3, HIGH);
}

// Sensors 12 and 12
if (i == 12)
{
    // Set Mux 1 to 1100 = signal 12
    digitalWrite(mux1s0, LOW);
digitalWrite(mux1s1, LOW);
digitalWrite(mux1s2, HIGH);
digitalWrite(mux1s3, HIGH);
    // Set Mux 2 to 1100 = signal 12
    digitalWrite(mux2s0, LOW);
digitalWrite(mux2s1, LOW);
digitalWrite(mux2s2, HIGH);
digitalWrite(mux2s3, HIGH);
}

// Sensors 13 and 13
if(i == 13)
{
    // Set Mux 1 to 1101 = signal 13
    digitalWrite(mux1s0, HIGH);
    digitalWrite(mux1s1, LOW);
    digitalWrite(mux1s2, HIGH);
    digitalWrite(mux1s3, HIGH);

    // Set Mux 2 to 1101 = signal 13
    digitalWrite(mux2s0, HIGH);
    digitalWrite(mux2s1, LOW);
    digitalWrite(mux2s2, HIGH);
    digitalWrite(mux2s3, HIGH);
}

// Sensors 14 and 14
if(i == 14)
{
    // Set Mux 1 to 0110 = signal 6
    digitalWrite(mux1s0, LOW);
    digitalWrite(mux1s1, HIGH);
    digitalWrite(mux1s2, HIGH);
    digitalWrite(mux1s3, HIGH);

    // Set Mux 2 to 0110 = signal 14
    digitalWrite(mux2s0, LOW);
    digitalWrite(mux2s1, HIGH);
    digitalWrite(mux2s2, HIGH);
    digitalWrite(mux2s3, HIGH);
}

// Steps Common to Every Mux Reading
// Read mux 1 and print sensor value to serial
mux1sensors[i] = analogRead(mux1Output);
Serial.print(mux1sensors[i]);
Serial.print(",");

// clear Arduino A/D
zero = analogRead(groundPin);

// Read mux 2 and print sensor value to serial
mux2sensors[i] = analogRead(mux2Output);
Serial.print(mux2sensors[i]);

// Set mux 1 to 1111 = signal 15
digitalWrite(mux1s0, HIGH);
digitalWrite(mux1s1, HIGH);
digitalWrite(mux1s2, HIGH);
digitalWrite(mux1s3, HIGH);

// Set mux 2 to 1111 = signal 15
digitalWrite(mux2s0, HIGH);
digitalWrite(mux2s1, HIGH);
digitalWrite(mux2s2, HIGH);
digitalWrite(mux2s3, HIGH);

zero = analogRead(mux1Output); // clear mux 1 A/D
zero = analogRead(mux2Output); // clear mux 2 A/D
zero = analogRead(groundPin); // clear Arduino A/D

if (i < (numReads-1))
{
    Serial.print(',');
}

Serial.println();
C. Processing Code

1. Thumb_Prototype

/*
Thumb Prototype Graphical User Display
- Reads the 8 sensor signals from the thumb prototype via serial port, displaying the data as a real-time bar graph as well as changing the transparency of colored sensor dots on a drawn model of the thumb prototype. Raw sensor values are displayed next to each corresponding sensor dot.

Written by Ross Miller
Mechanical Engineering Master's Thesis
California Polytechnic State University
11/3/2010
*/

import processing.serial.*;

Serial myPort;
final int linefeed = 10;

// maximum number of sensors to display
final int maxSensors = 8;

// raw analog input values from controller
int raw[];
int rawMin[];
int rawMax[];

// values scaled to fit screen
float scaledVal[];
float scaledMin[];
float scaledMax[];
float prevScaledVal[];

// min/max values of analog input from controller
final int minAnalogVal = 0;
final int maxAnalogVal = 1024;

// colors used to draw sensor graphs
color colors[];

int xCursor = 0;

// length of each line segment in graph, 1=1 pixel
final int plotLineLength = 1;

PFont myFont;
final int fontSize = 24;

final int drawDelay = 10;

boolean madeContact = false;
void setup()
{
    myPort = new Serial(this, Serial.list()[1], 9600);
    myPort.bufferUntil(linefeed);

    // initialize raw vars
    raw = new int[maxSensors];
    rawMin = new int[maxSensors];
    for (int i = 0; i<rawMin.length; i++)
    {
        rawMin[i] = 1023;
    }
    rawMax = new int[maxSensors];

    // initialize scaled vars
    scaledVal = new float[maxSensors];
    scaledMin = new float[maxSensors];
    for (int i = 0; i<scaledMin.length; i++)
    {
        scaledMin[i] = 1023;
    }
    scaledMax = new float[maxSensors];

    prevScaledVal = new float[maxSensors];

    // set colors used for each sensor display
    colors = new color[maxSensors];
    colors[0] = color(255, 0, 0);    // red
    colors[1] = color(0, 132, 0);    // green
    colors[2] = color(0, 0, 255);    // blue
    colors[3] = color(0, 0, 0);      // black
    colors[4] = color(255, 0, 0);    // red
    colors[5] = color(0, 132, 0);    // green
    colors[6] = color(0, 0, 255);    // blue
    colors[7] = color(0, 0, 0);      // black

    // println(PFont.list());
    PFont myFont = createFont(PFont.list()[4], fontSize);
    textFont(myFont);

    size(1050, 800);
    background(255);
}

void draw()
{
    stroke(255);
    fill(255);

    // erases palette
    rect(0,0,1050,800);

    if(madeContact==false)
    {
        // start handshake w/controller
        myPort.write('\r');
    }
}   

else   
{   
// ==============================================================
// TITLE
// ==============================================================
fill(0);   

text("Thumb Prototype Graphical User Display", 10, 25);    
text("Created by Ross Miller", 30, 50);   

// ==============================================================
// BAR GRAPH
// ==============================================================
int startingX = 20;    
int barWidth = 50;   

for (int i = 0; i < raw.length; i++)   
{   
    stroke(0);   
    fill(colors[i]);   
    
    rect(startingX+i*50, 800-raw[i]/2, 50, raw[i]);   
    text(raw[i], startingX+7+i*50, 800-raw[i]/2-12);   
}   

// ==============================================================
// DRAWS THUMB SHAPE
// ==============================================================
stroke(0);

// 3D Thumb
int initX = 550;    
int initY = 100;   

line(initX+441, initY+584, initX+367, initY+418);    
line(initX+367, initY+418, initX+308, initY+369);    
line(initX+308, initY+369, initX+255, initY+269);    
line(initX+255, initY+269, initX+239, initY+198);    
line(initX+239, initY+198, initX+198, initY+122);    
line(initX+198, initY+122, initX+176, initY+29);    
line(initX+176, initY+29, initX+167, initY+88);    // thumbnail   
line(initX+167, initY+88, initX+198, initY+122);    // thumbnail   
line(initX+176, initY+29, initX+130, initY+7);    
line(initX+130, initY+7, initX+85, initY+17);    
line(initX+85, initY+17, initX+39, initY+72);    
line(initX+39, initY+72, initX+24, initY+165);    
line(initX+24, initY+165, initX+55, initY+326);    
line(initX+55, initY+326, initX+117, initY+446);    
line(initX+117, initY+446, initX+77, initY+543);   

// ==============================================================
// CREATES SENSOR DOTS
// ==============================================================
for (int i = 0; i < raw.length; i++)
{ stroke(0);

    // Red
    if(i == 0 || i == 4) {
        // 255,197,197 = transparent red
        // 255,0,0 = red
        if(raw[i] < 100) {
            fill(255,197,197);
        }
        else if(raw[i] < 200) {
            fill(255,150,150);
        }
        else if(raw[i] < 300) {
            fill(255,100,100);
        }
        else if(raw[i] < 400) {
            fill(255,50,50);
        }
        else {
            fill(255,0,0);
        }
    }

    // Green
    if(i == 1 || i == 5) {
        // 197,255,197 = transparent green
        // 0,255,0 = green
        if(raw[i] < 100) {
            fill(197,255,197);
        }
        else if(raw[i] < 200) {
            fill(100,255,100);
        }
        else if(raw[i] < 300) {
            fill(0,255,0);
        }
        else if(raw[i] < 400) {
            fill(0,200,0);
        }
        else {
            fill(0,150,0);
        }
    }
}
if (i == 2 || i == 6) {
    // 197,197,255 = transparent blue
    // 0,0,255 = dark blue
    if (raw[i] < 100)
        fill(197,197,255);
    else if (raw[i] < 200)
        fill(150,150,255);
    else if (raw[i] < 300)
        fill(100,100,255);
    else if (raw[i] < 400)
        fill(50,50,255);
    else
        fill(0,0,255);
}

if (i == 3 || i == 7) {
    // 197,197,197 = light grey
    // 50,50,50 = Black
    if (raw[i] < 100)
        fill(197,197,197);
    else if (raw[i] < 200)
        fill(150,150,150);
    else if (raw[i] < 300)
        fill(100,100,100);
    else if (raw[i] < 400)
        fill(50,50,50);
    else
        fill(0,0,0);
}
// Inner-Upper Distal
if(i == 0)
{
    ellipse(initX+46,initY+79,12,30); // x,y,width,height
    fill(255,0,0);
    text(raw[i],initX-10,initY+82);
}

// Outer-Upper Distal
if(i == 1)
{
    ellipse(initX+90,initY+75,29,30); // x,y,width,height
    fill(0,200,0);
    text(raw[i],initX+120,initY+84);
}

// Inner-Lower Distal
if(i == 2)
{
    ellipse(initX+39,initY+184,15,30); // x,y,width,height
    fill(0,0,255);
    text(raw[i],initX-20,initY+190);
}

// Outer-Lower Distal
if(i == 3)
{
    ellipse(initX+89,initY+184,29,30); // x,y,width,height
    fill(0,0,0);
    text(raw[i],initX+125,initY+190);
}

// Proximal
if(i == 4)
{
    ellipse(initX+101,initY+342,21,30); // x,y,width,height
    fill(255,0,0);
    text(raw[i],initX+10,initY+346);
}

// Backside Knuckle
if(i == 5)
{
    ellipse(initX+232,initY+226,12,30); // x,y,width,height
    fill(0,150,0);
    text(raw[i],initX+260,initY+236);
}

// Inner Metacarpal
if(i == 6)
{
    ellipse(initX+112,initY+508,18,30); // x,y,width,height
    fill(0,0,255);
    text(raw[i],initX+50,initY+515);
}
// Outer Metacarpal
if (i == 7) {
    ellipse(initX+354,initY+518,30,30); // x,y,width,height
    fill(0,0,0);
    text(raw[i],initX+420,initY+525);
}
}

// =============================================================
// READING THE VALUES FROM THE SERIAL
// =============================================================

void serialEvent(Serial myPort) {
    madeContact = true;

    String rawInput = myPort.readStringUntil(linefeed);
    if (rawInput != null) {
        rawInput = trim(rawInput);

        int sensors[] = int(split(rawInput, ','));

        //read in raw sensor values
        for (int i=0; i<sensors.length; i++) {
            raw[i] = sensors[i]-126;
            rawMin[i] = min(rawMin[i], raw[i]);
            rawMax[i] = max(rawMax[i], raw[i]);
            //print(i + " : " + raw[i] + "\t" + rawMin[i] + "|" +
            rawMax[i] + "\t"");
        }
        println();

        //scale raw sensor values
        for (int i=0; i<sensors.length; i++) {
            scaledVal[i] = height * (raw[i] - minAnalogVal) / maxAnalogVal;
            scaledMin[i] = height * (rawMin[i] - minAnalogVal) / maxAnalogVal;
            scaledMax[i] = height * (rawMax[i] - minAnalogVal) / maxAnalogVal;
        }
    }
}

//request more data from controller
myPort.write(\r\n);
2. Hand_Prototype

/*
Hand Prototype Graphical User Display
- Reads the 30 sensor signals from the hand prototype via
serial port, displaying the data as a real-time bar graph
as well as changing the transparency of colored sensor dots
on a drawn model of the hand prototype. Raw sensor values
are displayed next to each corresponding sensor dot.

Written by Ross Miller
Mechanical Engineering Master's Thesis
California Polytechnic State University
11/3/2010
*/

import processing.serial.*;

Serial myPort;
final int linefeed = 10;

// maximum number of sensors to display
final int maxSensors = 30;

// raw analog input values from controller
int raw[];
int rawMin[];
int rawMax[];

// values scaled to fit screen
float scaledVal[];
float scaledMin[];
float scaledMax[];
float prevScaledVal[];

// min/max values of analog input from controller
final int minAnalogVal = 0;
final int maxAnalogVal = 1024;

// colors used to draw sensor graphs
color colors[];
int xCursor = 0;

// length of each line segment in graph, 1=1 pixel
final int plotLineLength = 1;

PFont font;
final int drawDelay = 10;
boolean madeContact = false;

void setup()
{
    myPort = new Serial(this, Serial.list()[1], 9600);
myPort.bufferUntil(linefeed);

// initialize raw vars
raw = new int[maxSensors];
rawMin = new int[maxSensors];
for (int i = 0; i < rawMin.length; i++)
{
    rawMin[i] = 1023;
}
rawMax = new int[maxSensors];

// initialize scaled vars
scaledVal = new float[maxSensors];
scaledMin = new float[maxSensors];
for (int i = 0; i < scaledMin.length; i++)
{
    scaledMin[i] = 1023;
}
scaledMax = new float[maxSensors];

prevScaledVal = new float[maxSensors];

// set colors used for each sensor display
colors = new color[maxSensors];
colors[0] = color(255, 0, 0); // red
colors[1] = color(0, 132, 0); // green
colors[2] = color(0, 0, 255); // blue
colors[3] = color(0, 0, 0); // black
colors[4] = color(255, 0, 0); // red
colors[5] = color(0, 132, 0); // green
colors[6] = color(0, 0, 255); // blue
colors[7] = color(0, 0, 0); // black
colors[8] = color(255, 0, 0); // red
colors[9] = color(0, 132, 0); // green
colors[10] = color(0, 0, 255); // blue
colors[11] = color(0, 0, 0); // black
colors[12] = color(255, 0, 0); // red
colors[13] = color(0, 132, 0); // green
colors[14] = color(0, 0, 255); // blue
colors[15] = color(0, 0, 0); // black
colors[16] = color(255, 0, 0); // red
colors[17] = color(0, 132, 0); // green
colors[18] = color(0, 0, 255); // blue
colors[19] = color(0, 0, 0); // black
colors[20] = color(255, 0, 0); // red
colors[21] = color(0, 132, 0); // green
colors[22] = color(0, 0, 255); // blue
colors[23] = color(0, 0, 0); // black
colors[24] = color(255, 0, 0); // red
colors[25] = color(0, 132, 0); // green
colors[26] = color(0, 0, 255); // blue
colors[27] = color(0, 0, 0); // black
colors[28] = color(255, 0, 0); // red
colors[29] = color(0, 132, 0); // green

PFont font = createFont(PFont.list()[4], 24);
textFont(font);
size(1200, 800);
background(255);
}

void draw()
{
    stroke(255);
    fill(255);

    // erases palette area
    rect(0,0,1200,800);  // x,y,width,height

    if(madeContact==false)
    {
        //start handshake w/controller
        myPort.write('\r');
    }

    else
    {
        // ==============================================================
        // TITLE
        // ==============================================================
        fill(0);

        textSize(24);
        text("Hand Prototype Graphical User Display", 10, 24);
        text("Created by Ross Miller", 30, 50);

        textSize(20);
        // ==============================================================
        // BAR GRAPHS
        // ==============================================================
        stroke(255);
        fill(255);

        int startingX = 0;
        int barWidth = 38;

        for (int i = 0; i < raw.length; i++)
        {
            stroke(0);
            fill(colors[i]);

            rect(startingX+i*40,800-raw[i]/3,barWidth,raw[i]);
            text(raw[i],startingX+i*40,800-raw[i]/3-12);
        }

        // ==============================================================
        // Left Hand Inside
        // ==============================================================

        int insideInitX = 220;
        int insideInitY = 5;

        // Thumb
line(insideInitX+174,insideInitY+540,insideInitX+174,insideInitY+504);
line(insideInitX+174,insideInitY+504,insideInitX+90,insideInitY+389);
line(insideInitX+90,insideInitY+389,insideInitX+65,insideInitY+315);
line(insideInitX+65,insideInitY+315,insideInitX+24,insideInitY+254);
line(insideInitX+24,insideInitY+254,insideInitX+27,insideInitY+243);
line(insideInitX+27,insideInitY+243,insideInitX+42,insideInitY+234);
line(insideInitX+42,insideInitY+234,insideInitX+66,insideInitY+236);
line(insideInitX+66,insideInitY+236,insideInitX+105,insideInitY+273);
line(insideInitX+105,insideInitY+273,insideInitX+136,insideInitY+321);
// Pointer
line(insideInitX+136,insideInitY+321,insideInitX+156,insideInitY+324);
line(insideInitX+156,insideInitY+324,insideInitX+169,insideInitY+295);
line(insideInitX+169,insideInitY+295,insideInitX+170,insideInitY+242);
line(insideInitX+170,insideInitY+242,insideInitX+156,insideInitY+172);
line(insideInitX+156,insideInitY+172,insideInitX+148,insideInitY+113);
line(insideInitX+148,insideInitY+113,insideInitX+144,insideInitY+58);
line(insideInitX+144,insideInitY+58,insideInitX+151,insideInitY+44);
line(insideInitX+151,insideInitY+44,insideInitX+166,insideInitY+39);
line(insideInitX+166,insideInitY+39,insideInitX+171,insideInitY+40);
line(insideInitX+171,insideInitY+40,insideInitX+182,insideInitY+48);
line(insideInitX+182,insideInitY+48,insideInitX+195,insideInitY+102);
line(insideInitX+195,insideInitY+102,insideInitX+209,insideInitY+158);
line(insideInitX+209,insideInitY+158,insideInitX+224,insideInitY+219);
line(insideInitX+224,insideInitY+219,insideInitX+229,insideInitY+227);

// Middle
line(insideInitX+229,insideInitY+227,insideInitX+245,insideInitY+152);
line(insideInitX+245,insideInitY+152,insideInitX+242,insideInitY+86);
line(insideInitX+242,insideInitY+86,insideInitX+243,insideInitY+33);
line(insideInitX+243,insideInitY+33,insideInitX+250,insideInitY+19);
line(insideInitX+250,insideInitY+19,insideInitX+262,insideInitY+11);
line(insideInitX+262,insideInitY+11,insideInitX+275,insideInitY+12);
line(insideInitX+275,insideInitY+12,insideInitX+287,insideInitY+22);
line(insideInitX+287,insideInitY+22,insideInitX+292,insideInitY+85);
line(insideInitX+292,insideInitY+85,insideInitX+296,insideInitY+147);
line(insideInitX+296,insideInitY+147,insideInitX+298,insideInitY+231);
line(insideInitX+298,insideInitY+231,insideInitX+307,insideInitY+236);

// Ring
line(insideInitX+307,insideInitY+236,insideInitX+310,insideInitY+181);
line(insideInitX+310,insideInitY+181,insideInitX+320,insideInitY+127);
line(insideInitX+320,insideInitY+127,insideInitX+325,insideInitY+70);
line(insideInitX+325,insideInitY+70,insideInitX+334,insideInitY+57);
line(insideInitX+334,insideInitY+57,insideInitX+347,insideInitY+52);
line(insideInitX+347,insideInitY+52,insideInitX+361,insideInitY+59);
line(insideInitX+361,insideInitY+59,insideInitX+367,insideInitY+69);
line(insideInitX+367,insideInitY+69,insideInitX+363,insideInitY+131);
line(insideInitX+363,insideInitY+131,insideInitX+359,insideInitY+186);
line(insideInitX+359,insideInitY+186,insideInitX+357,insideInitY+262);
line(insideInitX+357,insideInitY+262,insideInitX+376,insideInitY+276);

    // Pinkie

line(insideInitX+376,insideInitY+276,insideInitX+397,insideInitY+238);
line(insideInitX+397,insideInitY+238,insideInitX+416,insideInitY+199);
line(insideInitX+416,insideInitY+199,insideInitX+433,insideInitY+155);
line(insideInitX+433,insideInitY+155,insideInitX+440,insideInitY+146);
line(insideInitX+440,insideInitY+146,insideInitX+453,insideInitY+144);
line(insideInitX+453,insideInitY+144,insideInitX+465,insideInitY+150);
line(insideInitX+465,insideInitY+150,insideInitX+472,insideInitY+166);
line(insideInitX+472,insideInitY+166,insideInitX+456,insideInitY+217);

line(insideInitX+456,insideInitY+217,insideInitX+440,insideInitY+255);

line(insideInitX+440,insideInitY+255,insideInitX+425,insideInitY+307);

line(insideInitX+425,insideInitY+307,insideInitX+424,insideInitY+347);

line(insideInitX+424,insideInitY+347,insideInitX+419,insideInitY+374);

line(insideInitX+419,insideInitY+374,insideInitX+404,insideInitY+443);

line(insideInitX+404,insideInitY+443,insideInitX+360,insideInitY+528);

line(insideInitX+360,insideInitY+528,insideInitX+357,insideInitY+540);

// =============================================
// Left Hand Outside
// =============================================

int outsideInitX = 690;
int outsideInitY = 5;

// Pinkie

line(outsideInitX+137,outsideInitY+533,outsideInitX+138,outsideInitY+517);

line(outsideInitX+138,outsideInitY+517,outsideInitX+80,outsideInitY+317);

line(outsideInitX+80,outsideInitY+317,outsideInitX+57,outsideInitY+265);

line(outsideInitX+57,outsideInitY+265,outsideInitX+38,outsideInitY+225);

line(outsideInitX+38,outsideInitY+225,outsideInitX+25,outsideInitY+185);

line(outsideInitX+25,outsideInitY+185,outsideInitX+27,outsideInitY+177);

line(outsideInitX+27,outsideInitY+177,insideInitX+33,outsideInitY+161);

line(outsideInitX+33,outsideInitY+161,insideInitX+43,outsideInitY+160);
line(outsideInitX+43, outsideInitY+160, outsideInitX+60, outsideInitY+174);
line(outsideInitX+60, outsideInitY+174, outsideInitX+75, outsideInitY+206);
line(outsideInitX+75, outsideInitY+206, outsideInitX+97, outsideInitY+242);
line(outsideInitX+97, outsideInitY+242, outsideInitX+128, outsideInitY+277);
line(outsideInitX+128, outsideInitY+277, outsideInitX+137, outsideInitY+276);

// fingernail
line(outsideInitX+26, outsideInitY+180, outsideInitX+33, outsideInitY+197);
line(outsideInitX+33, outsideInitY+197, outsideInitX+57, outsideInitY+182);
line(outsideInitX+57, outsideInitY+182, outsideInitX+50, outsideInitY+166);

// Ring

line(outsideInitX+137, outsideInitY+276, outsideInitX+127, outsideInitY+190);
line(outsideInitX+127, outsideInitY+190, outsideInitX+117, outsideInitY+129);
line(outsideInitX+117, outsideInitY+129, outsideInitX+108, outsideInitY+79);
line(outsideInitX+108, outsideInitY+79, outsideInitX+112, outsideInitY+69);
line(outsideInitX+112, outsideInitY+69, outsideInitX+122, outsideInitY+62);
line(outsideInitX+122, outsideInitY+62, outsideInitX+136, outsideInitY+62);
line(outsideInitX+136, outsideInitY+62, outsideInitX+144, outsideInitY+58);
line(outsideInitX+144, outsideInitY+58, outsideInitX+161, outsideInitY+115);
line(outsideInitX+161, outsideInitY+115, outsideInitX+173, outsideInitY+167);
line(outsideInitX+173, outsideInitY+167, outsideInitX+193, outsideInitY+230);
// fingernail
line(outsideInitX+112, outsideInitY+70, outsideInitX+118, outsideInitY+94);
line(outsideInitX+118, outsideInitY+94, outsideInitX+141, outsideInitY+86);
line(outsideInitX+141, outsideInitY+86, outsideInitX+137, outsideInitY+63);

// Middle
line(outsideInitX+193, outsideInitY+230, outsideInitX+195, outsideInitY+143);
line(outsideInitX+195, outsideInitY+143, outsideInitX+199, outsideInitY+75);
line(outsideInitX+199, outsideInitY+75, outsideInitX+200, outsideInitY+29);
line(outsideInitX+200, outsideInitY+29, outsideInitX+206, outsideInitY+20);
line(outsideInitX+206, outsideInitY+20, outsideInitX+218, outsideInitY+14);
line(outsideInitX+218, outsideInitY+14, outsideInitX+235, outsideInitY+19);
line(outsideInitX+235, outsideInitY+19, outsideInitX+245, outsideInitY+30);
line(outsideInitX+245, outsideInitY+30, outsideInitX+249, outsideInitY+73);
line(outsideInitX+249, outsideInitY+73, outsideInitX+256, outsideInitY+139);
line(outsideInitX+256, outsideInitY+139, outsideInitX+261, outsideInitY+220);
line(outsideInitX+261, outsideInitY+220, outsideInitX+275, outsideInitY+217);

// fingernail
line(outsideInitX+204, outsideInitY+22, outsideInitX+205, outsideInitY+44);
line(outsideInitX+205, outsideInitY+44, outsideInitX+236, outsideInitY+44);
line(outsideInitX+236, outsideInitY+44, outsideInitX+236, outsideInitY+21);

// Pointer
line(outsideInitX+275, outsideInitY+217, outsideInitX+295, outsideInitY+138);
line(outsideInitX+295, outsideInitY+138, outsideInitX+307, outsideInitY+81);
line(outsideInitX+307, outsideInitY+81, outsideInitX+314, outsideInitY+53);
line(outsideInitX+314, outsideInitY+53, outsideInitX+321, outsideInitY+44);
line(outsideInitX+321, outsideInitY+44, outsideInitX+333, outsideInitY+40);
line(outsideInitX+333, outsideInitY+40, outsideInitX+350, outsideInitY+47);
line(outsideInitX+350, outsideInitY+47, outsideInitX+358, outsideInitY+63);
line(outsideInitX+358, outsideInitY+63, outsideInitX+357, outsideInitY+88);
line(outsideInitX+357, outsideInitY+88, outsideInitX+355, outsideInitY+146);
line(outsideInitX+355, outsideInitY+146, outsideInitX+346, outsideInitY+227);
line(outsideInitX+346, outsideInitY+227, outsideInitX+364, outsideInitY+307);
  // fingernail
line(outsideInitX+322, outsideInitY+43, outsideInitX+319, outsideInitY+64);
line(outsideInitX+319, outsideInitY+64, outsideInitX+350, outsideInitY+70);
line(outsideInitX+350, outsideInitY+70, outsideInitX+353, outsideInitY+52);
  // Thumb
line(outsideInitX+364, outsideInitY+307, outsideInitX+383, outsideInitY+303);
line(outsideInitX+383, outsideInitY+303, outsideInitX+421, outsideInitY+260);
line(outsideInitX+421, outsideInitY+260, outsideInitX+451, outsideInitY+225);
line(outsideInitX+451, outsideInitY+225, outsideInitX+475, outsideInitY+214);
line(outsideInitX+475, outsideInitY+214, outsideInitX+494, outsideInitY+218);
line(outsideInitX+494, outsideInitY+218, outsideInitX+504, outsideInitY+229);
line(outsideInitX+504, outsideInitY+229, outsideInitX+504, outsideInitY+238);
line(outsideInitX+504, outsideInitY+238, outsideInitX+487, outsideInitY+261);
line(outsideInitX+487, outsideInitY+261, outsideInitX+476, outsideInitY+290);
line(outsideInitX+476, outsideInitY+290, outsideInitX+433, outsideInitY+365);
line(outsideInitX+433, outsideInitY+365, outsideInitX+373, outsideInitY+487);
line(outsideInitX+373, outsideInitY+487, outsideInitX+356, outsideInitY+533);
    // fingernail
line(outsideInitX+487, outsideInitY+217, outsideInitX+472, outsideInitY+238);
line(outsideInitX+472, outsideInitY+238, outsideInitX+486, outsideInitY+257);
line(outsideInitX+486, outsideInitY+257, outsideInitX+503, outsideInitY+235);
    // ==============================================================
    // CREATES SENSOR DOTS
    // ==============================================================

    for (int i = 0; i < raw.length; i++)
    {
        stroke(0);

        // Red
        if(i == 0 || i == 4 || i == 8 || i == 12 || i == 16
            || i == 20 || i == 24 || i == 28)
        {
            // 255,0,0 = red
            // 255,235,235 = transparent red
            if(raw[i] < 100)
            {
                fill(255,235,235);
            }
            else if(raw[i] < 200)
            {
                fill(255,150,150);
            }
        }
    }
else if(raw[i]<300)
{
    fill(255,100,100);
}
else if(raw[i]<400)
{
    fill(255,50,50);
}
else
{
    fill(255,0,0);
}

// Green
if(i == 1 || i == 5 || i == 9 || i == 13 || i == 17 || i == 21 || i == 25 || i == 29)
{
    // 235,255,235 = transparent green
    // 0,150,0 = green

    if(raw[i] <100)
    {
        fill(235,255,235);
    }
    else if(raw[i] <200)
    {
        fill(100,255,100);
    }
    else if(raw[i]<300)
    {
        fill(0,255,0);
    }
    else if(raw[i]<400)
    {
        fill(0,200,0);
    }
    else
    {
        fill(0,150,0);
    }
}

// Blue
if(i == 2 || i == 6 || i == 10 || i == 14 || i == 18 || i == 22 || i == 26)
{
    // 235,235,255 = transparent blue
    // 0,0,255 = dark blue

    if(raw[i] <100)
    {
        fill(235,235,255);
    }
    else if(raw[i] <200)
    {
        fill(150,150,255);
    }
}  
else if (raw[i] < 300)  
{  
  fill(100, 100, 255);  
}  
else if (raw[i] < 400)  
{  
  fill(50, 50, 255);  
}  
else  
{  
  fill(0, 0, 255);  
}  
}  

// Black  
if (i == 3 || i == 7 || i == 11 || i == 15 || i == 19  
|| i == 23 || i == 27)  
{  
  // 50, 50, 50 = Black  
  // 197, 197, 197 = light grey  
  if (raw[i] < 100)  
  {  
    fill(235, 235, 235);  
  }  
  else if (raw[i] < 200)  
  {  
    fill(150, 150, 150);  
  }  
  else if (raw[i] < 300)  
  {  
    fill(100, 100, 100);  
  }  
  else if (raw[i] < 400)  
  {  
    fill(50, 50, 50);  
  }  
  else  
  {  
    fill(0, 0, 0);  
  }  
}  

// INSIDE  
if (i == 25)  
{  
  // Thumb 0  
  ellipse(insideInitX+55, insideInitY+250, 10, 10);  
  // x, y, width, height  
  fill(0, 0, 0);  
  text(raw[i], insideInitX+40, insideInitY+230);  
}  
else if (i == 3)  
{  
  // Thumb 1  
}
ellipse(insideInitX+90, insideInitY+300, 10, 10);  // x, y, width, height
  fill(0, 0, 0);
  text(raw[i], insideInitX+60, insideInitY+290);
}

else if (i == 18)
{
  // Thumb 2
  ellipse(insideInitX+120, insideInitY+370, 10, 10);  // x, y, width, height
  fill(0, 0, 0);
  text(raw[i], insideInitX+100, insideInitY+360);
}

else if (i == 19)
{
  // Thumb 3
  ellipse(insideInitX+170, insideInitY+440, 10, 10);  // x, y, width, height
  fill(0, 0, 0);
  text(raw[i], insideInitX+150, insideInitY+430);
}

else if (i == 4)
{
  // Pointer 0
  ellipse(insideInitX+165, insideInitY+60, 10, 10);  // x, y, width, height
  fill(0, 0, 0);
  text(raw[i], insideInitX+145, insideInitY+30);
}

else if (i == 22)
{
  // Pointer 1
  ellipse(insideInitX+175, insideInitY+125, 10, 10);  // x, y, width, height
  fill(0, 0, 0);
  text(raw[i], insideInitX+155, insideInitY+115);
}

else if (i == 27)
{
  // Pointer 2
  ellipse(insideInitX+185, insideInitY+190, 10, 10);  // x, y, width, height
  fill(0, 0, 0);
  text(raw[i], insideInitX+165, insideInitY+180);
}

else if (i == 20)
{
  // Pointer 3
  ellipse(insideInitX+200, insideInitY+270, 10, 10);  // x, y, width, height
  fill(0, 0, 0);
text(raw[i],insideInitX+180,insideInitY+260);
}

else if (i == 10)
{
    // Middle 0
ellipse(insideInitX+265,insideInitY+35,10,10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+212,insideInitY+38);
}

else if (i == 8)
{
    // Middle 1
    ellipse(insideInitX+268,insideInitY+110,10,10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+248,insideInitY+100);
}

else if (i == 6)
{
    // Middle 2
    ellipse(insideInitX+266,insideInitY+180,10,10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+251,insideInitY+170);
}

else if (i == 26)
{
    // Middle 3
    ellipse(insideInitX+270,insideInitY+270,10,10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+255,insideInitY+260);
}

else if (i == 24)
{
    // Middle 4
    ellipse(insideInitX+270,insideInitY+350,10,10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+255,insideInitY+340);
}

else if (i == 21)
{
    // Ring 0
    ellipse(insideInitX+345,insideInitY+75,10,10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+330,insideInitY+45);
}
```cpp
else if (i == 0)
{
    // Ring 1
    ellipse(insideInitX+340, insideInitY+150, 10, 10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+325,insideInitY+140);
}
else if (i == 14)
{
    // Ring 2
    ellipse(insideInitX+335, insideInitY+207, 10, 10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+320,insideInitY+197);
}
else if (i == 12)
{
    // Ring 3
    ellipse(insideInitX+325, insideInitY+280, 10, 10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+313,insideInitY+270);
}
else if (i == 29)
{
    // Pinky 0
    ellipse(insideInitX+448, insideInitY+170, 10, 10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+398,insideInitY+165);
}
else if (i == 23)
{
    // Pinky 1
    ellipse(insideInitX+428, insideInitY+220, 10, 10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+422,insideInitY+210);
}
else if (i == 16)
{
    // Pinky 2
    ellipse(insideInitX+410, insideInitY+270, 10, 10); // x,y,width,height
    fill(0,0,0);
    text(raw[i],insideInitX+395,insideInitY+260);
}
else if (i == 28)
{
    // Pinky 3
    //...
```
ellipse(insideInitX+385,insideInitY+322,10,10);  //
x,y,width,height
fill(0,0,0);
text(raw[i],insideInitX+370,insideInitY+312);
}

else if (i == 17)
{
  // Pinky 4
  ellipse(insideInitX+370,insideInitY+385,10,10);  //
x,y,width,height
  fill(0,0,0);
text(raw[i],insideInitX+355,insideInitY+375);
}

// BACKSIDE
else if (i == 15)
{
  // Thumb Back
  ellipse(outsideInitX+465,outsideInitY+280,10,10);  //
x,y,width,height
  fill(0,0,0);
text(raw[i],outsideInitX+435,outsideInitY+270);
}

else if (i == 1)
{
  // Pointer Back
  ellipse(outsideInitX+325,outsideInitY+140,10,10);  //
x,y,width,height
  fill(0,0,0);
text(raw[i],outsideInitX+312,outsideInitY+130);
}

else if (i == 5)
{
  // Middle Back
  ellipse(outsideInitX+225,outsideInitY+130,10,10);  //
x,y,width,height
  fill(0,0,0);
text(raw[i],outsideInitX+210,outsideInitY+120);
}

else if (i == 2)
{
  // Ring Back
  ellipse(outsideInitX+145,outsideInitY+155,10,10);  //
x,y,width,height
  fill(0,0,0);
text(raw[i],outsideInitX+128,outsideInitY+145);
}

else if (i == 11)
{
  // Pinky Back
  ellipse(outsideInitX+70,outsideInitY+240,10,10);  //
x,y,width,height
```cpp
fill(0,0,0);
text(raw[i],outsideInitX+45,outsideInitY+230);
}

else if (i == 9)
{
    // Hand Back
    ellipse(outsideInitX+235,outsideInitY+350,10,10); // x,y,width,height
    fill(0,0,0);
text(raw[i],outsideInitX+215,outsideInitY+330);
}

else if (i == 13)
{
    // Thumb Back Base
    ellipse(outsideInitX+410,outsideInitY+365,10,10); // x,y,width,height
    fill(0,0,0);
text(raw[i],outsideInitX+370,outsideInitY+375);
}

else if (i == 7)
{
    // Thumb Crease
    ellipse(outsideInitX+385,outsideInitY+315,10,10); // x,y,width,height
    fill(0,0,0);
text(raw[i],outsideInitX+340,outsideInitY+325);
}

}

// ==============================================================
// READING THE VALUES FROM THE SERIAL
// ==============================================================

void serialEvent(Serial myPort) {

    madeContact = true;

    String rawInput = myPort.readStringUntil(linefeed);

    if (rawInput != null) {
        rawInput = trim(rawInput);

        int sensors[] = int(split(rawInput, ', '));

        //read in raw sensor values
        for (int i=0; i< sensors.length; i++) {
            raw[i] = sensors[i]-133;
            rawMin[i] = min(rawMin[i], raw[i]);
            rawMax[i] = max(rawMax[i], raw[i]);
            //print(i + ":" + raw[i] + "|" + rawMin[i] + "|" + rawMax[i] + "\t" + rawMin[i] + "|" + rawMax[i] + "\t")
        }
    }
```
// Sensor 15 is not working so cut it out of display
raw[14] = 0;
println();

// scale raw sensor values
for (int i=0; i<sensors.length; i++) {
  scaledVal[i] = height * (raw[i] - minAnalogVal) / maxAnalogVal;
  scaledMin[i] = height * (rawMin[i] - minAnalogVal) / maxAnalogVal;
  scaledMax[i] = height * (rawMax[i] - minAnalogVal) / maxAnalogVal;
}

// request more data from controller
myPort.write('\r');
}