The Huggable: A Socially Assistive Robot for Pediatric Care

By

Kristopher B. Dos Santos

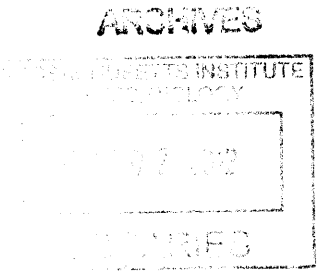
BS Mechanical Engineering, Massachusetts Institute of Technology (2010)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Media Arts and Sciences

At The

MASSACHUSETTS INSTITUTE OF TECHNOLOGY



September 2012

Author _____
Program in Media Arts and Sciences
August 24, 2012

Certified by _____
Cynthia Breazeal, Ph.D.
Associate Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by _____
Prof. Patricia Maes
Associate Academic Head
Program in Media Arts and Sciences

The Huggable: A Socially Assistive Robot for Pediatric Care

By

Kristopher B. Dos Santos

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
On August 24, 2012, in Partial Fulfillment of the
Requirements for the Degree of Master Science in Media Arts and Sciences
At The Massachusetts Institute of Technology

Abstract
The purpose of this thesis is to present the design and evaluation of a new type of socially assistive robot, one that can interact with people and collect various types of sensory input while being small enough to hold in one's arms. This project is a completely new revision of the Huggable project created by Dan Stiehl and Cynthia Breazeal, which features a new mechanical design, a revamped electronics structure, and a polished control system based off of its sister project, DragonBot (developed by Adam Setapen). This thesis describes the process of how this new design came to be, and provides extensive content on how it was designed, along with all major components that were included. An evaluation is also presented as a test run for the new Huggable, in the form of an online survey. The results, along with much of the work done with the initial prototype, showed that there is still much work to be done to be convincing as a robust research robot. Improvements are listed, as well as its future work with Boston Children's Hospital. This new design hopes to finally bring the Huggable project out into the field for actual use with people.

Thesis Supervisor: Dr. Cynthia Breazeal
Title: Associate Professor of Media Arts and Sciences, Program in Media Arts and Sciences

The Huggable: A Socially Assistive Robot for Pediatric Care

By

Kristopher B. Dos Santos

The following people served as readers for this thesis:

Thesis Reader _____

Peter Weinstock, M.D., Ph.D.
Director, Simulator Program
Children's Hospital Boston

Thesis Reader _____

Rosalind Picard, Ph.D.
Associate Professor of Media Arts and Sciences
Program in Media Arts and Sciences

Acknowledgements

Without the assistance of many, this new iteration of the Huggable would not have come to fruition. Thanks go to the following people:

To Cynthia Breazeal, who has provided me with this incredible opportunity for six years. I am so grateful to have such a supportive advisor like yourself; you have given me so much. If it was not for you and your lab, I would not be where I am today, and I thank you wholeheartedly.

To Peter Weinstock, as a reader and a sponsor of this project coming to Boston Children's Hospital. Thank you for welcoming the Huggable into your program, and providing me with great feedback.

To Rosalind Picard, for reading and providing feedback on writing this thesis. I am very glad you agreed to help out all of the Robots this year!

To Steve Ohler, S.K. Wong, Lewie Leung, Spen Yang, Liu Liang, and the rest of the R&D Team at Jetta for giving me one of the greatest crash courses in product design and manufacturing that I have ever had. I have learned so much while working with you all, and I will never forget my experience at Jetta. Thank you for lending your resources and your wisdom in developing this product, and I look forward to working with you all in the future.

To Adam Setapen, whose work with the DragonBot platform laid down the groundwork for building the Huggable electronics and software system. If it was not for you my friend, the Huggable would not live. Thank you for everything, you are awesome, and I sincerely hope we get to work together in the future.

To Sooyeon Jeong, whose tireless efforts coded the Huggable's new AndroidController and Tele-Op Interface. You have helped me a great deal with software, especially coming off work on the DragonBot platform. Sooyeon, you have truly made this robot come to life!

To Rachel Lewis, for spending a gracious amount of time re-editing the fur patterns of the Huggable. Thank you for all of your hard work!

To Siggi Orn, who developed a robust motor controller that works so well with R1D1 and is compact enough for actual use with these new portable platforms. Also, thank you for general software/electrical debugging assistance.

To Fardad Faridi, who crafted such an elegant new design for the Huggable. It looks so much cuter now Fardad, awesome job!

To Dan Stiehl, for giving me the opportunity to work on this project as a young freshman at MIT. Who knew I would bring it to this state Dan? Thank you for all of your guidance; you were a terrific mentor.

To all of the other members of the Personal Robots Group, both current and graduated (Jin Joo, Natalie, Nick, David, Nancy, Kenton, Angela, Jesse, Matt, Adam W., Peter, Philipp, Jason), for being supportive throughout this project. Whether it was critical design advice, or just saying "Keep at it!" you all are wonderful colleagues and friends.

To my friends near and far, wherever they may be. Thank you for showing support for my work, and thinking that a robot teddy bear was cool.

To Sam, for always believing in me, even at times when I did not believe in myself. I will never forget your love and support; thank you.

To my parents, who have always had my back for these six years that I have been at MIT. You both have been through the best and worst with me, even if you could not be here physically. I love you both so much.

Table of Contents (will have page numbers after edits)

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

1.1 Introduction & Motivation

Robotic applications for health care have skyrocketed over the past decade. Not only have surgical robots seen great improvements, but robotic solutions are being applied to a great number of other scenarios, such as human rehabilitation, patient care, assisted living, and social therapy [1]. One such area of budding research is the idea of a socially assistive robot, or SAR. As defined by Mataric and Feil-Seifer, socially assistive robotics is based around the idea that social interaction can assist people young and old in a variety of areas. SARs have the potential to help various groups, such as the elderly, those with cognitive disorders, and even students [2].

Figure 1-1: Charles the robot

One such example of a SAR was Charles, a robot developed by the University of Auckland to take blood pressure (pictured above in Figure 1-1). Charles was designed off of the Peoplebot platform from Mobile Robotics, and equipped with an Omron blood pressure monitor. It was also programmed to have a digital face and voice that would instruct the user on how to take their blood pressure [3]. Another big area of research for SARs is children with ASD, or autism spectrum disorder. Many groups are already exploring the use of robots for treatment of ASD, including Mataric et al and their work with the Behavior-Based Behavior Intervention Architecture (a project used for the evaluation of HRI intervention studies). This architecture allows for certain robots to

engage the child in social interaction with themselves and with other human beings [4]. As well, Nikolopolous et al used the Lego NXT platform (shown below in Figure 1-2) to construct robots for social interaction studies with those that had ASD. The fascination with a seemingly inanimate, nonthreatening object allows for the children to learn from the object when it then starts to interact with them, thereby encouraging the establishment of social behavior [5].

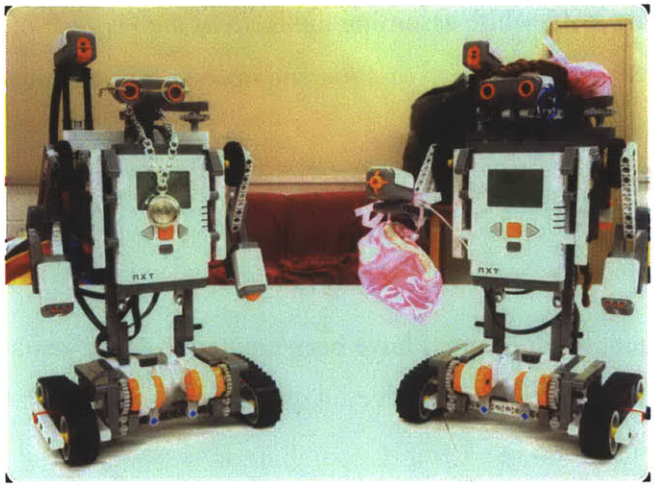


Figure 1-2: "Johnny" and "Julie", two NXT Lego robots

The value of SARs in treatment and therapy is certainly nontrivial, but it seems that there is not a perfect platform for social and physical interaction in a mobile platform. The various projects in the field right now are too big, too small, too complex, too simple, etc. There needs to be a balance of features and form in a SAR that allows it to be placed in multiple scenarios for research. Specifically, the design of a robot complex enough to receive multiple sources of sensor data while being compact enough to be handled by even the smallest hands is necessary.

1.2 Contributions & Scope

Enter the Huggable Project. The purpose of this thesis is to present the design and evaluation of a new type of SAR, one that can interact with people and collect various types of sensory input while being small enough to hold in one's arms. Its name is the Huggable, a semi-autonomous robotic avatar in the form of a teddy bear. This new version of the project started by Dan Stiehl and Cynthia Breazeal back in 2005 completely updates the old system with a robust and stable

mechanical structure, an updated electronic system, and a wireless computation system incorporating the Android software and R1D1 codebase.

The hopeful contribution of this project is to provide a mobile platform for HRI that easy to use, in a small and cuddly form. While the future uses of the Huggable are seemingly unlimited, the scope of this thesis is to merely prepare the Huggable for future use with young children at the hospital. With a smaller evaluation, the Huggable can be tweaked and edited for deployment. The hope is that the nursing staff, while assuming the form of the Huggable, can successfully calm a child in a situation of anxiety or anticipation of pain, and become a consistent character for children in the hospital setting.

1.3 Related Work

Several different robot projects have been created over the years to find some of the same results that this project is looking to achieve. Arguably one of the most famous examples is PARO, the robotic seal.

Figure 1-3: PARO, the robotic seal

PARO is a therapeutic robot developed by the National Institute of Advanced Industrial Science and Technology (Japan) to simulate companion animal therapy. It is a robot in the form of a baby seal that can respond to different types of touch in the way that a pet would respond. In a study done by Yamamoto and Kimura, PARO, and two other brands of famous entertainment robots (AIBO and ifbot) were introduced into a playtime environment with children ages 4-6. After observing their play, they noticed that the children accepted that they had a form of consciousness, but were not actual living things. They did observe that common interactions involved petting and

holding the robots tenderly, while some even imitated the motions of the robots. However, the flaw is that these robots have no way in recording some of this tactile feedback. Also, these robots are not designed for social assistance, but rather, social interaction [6].

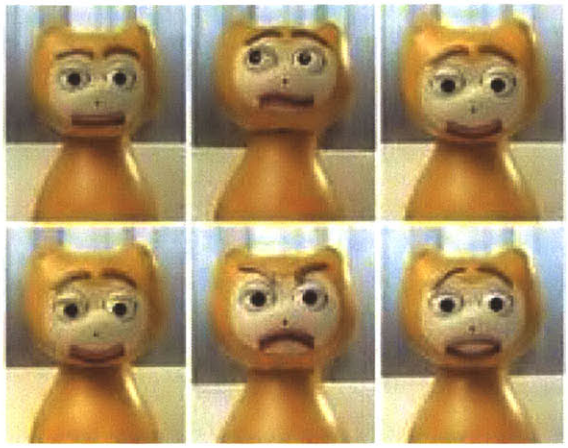


Figure 1-4: The iCat robot from Philips

The iCat robot from Philips (shown above in Figure 1-4) was used in a study by Looije et al to determine if a physical agent, rather than a virtual one or simply a text-based guide, was better at being a motivator/educator/buddy. The results showed that children preferred the character as opposed to the text interface for the support roles that they enjoyed most: being a buddy. They also noted that the physical robot was the most fun as far as interaction. This can highly support the socially assistive attributes of the iCat platform, but it is not mobile (to clarify, it only sits in one spot) and the only expressiveness comes from noisy head/eye movements, and eyebrow/mouth movements. Also, there seems to be no tactile feedback with this platform, something desirable when working with young children [7].

A most similar and interesting project developed recently is that of Probo, a "huggable" robotic platform designed for robot-assisted therapy, or RAT. Probo is a semi-autonomous robot in the form of a half-zoomorphic, half-cartoonish form based on woolly mammoths and elephants.

Figure 1-5: Probo, the "huggable" robot

As show above in Figure 1-5, it has an expressive face and actuated trunk, with a soft green exterior and digital screen in its belly. It's described to use a human operator, and is capable of utilizing actual speech and nonsense speech that can transmit emotion. Many of the features and abilities that Probo is described to have are features that are desirable of the new Huggable prototype, but Probo is 66cm tall and 32cm wide. Quite frankly, it's huge. Children may be able to "hug" this robot, but it misses the mobility that some of the smaller robots have. That is, it cannot be held and cuddled [8].

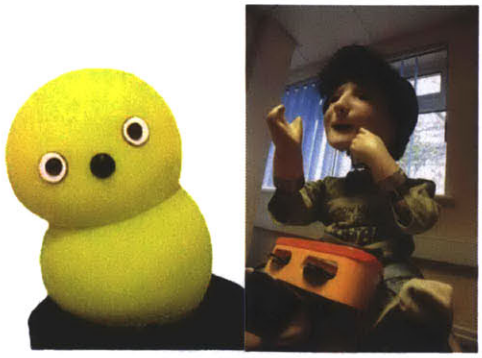


Figure 1-6: Keepon (left) developed by BeatBots, and KASPAR (right) developed by University of Hetfordshire

There are many more research robots in use that are trying to address interaction with children and, specifically, autism treatment. Robots such as Keepon [9], and KASPAR[10], along with more commercially built robots like Nao [11] and Zeno [12], utilize their ability to create

expressive movement in non-threatening form factors. Keepon's simplistic design allows for a relaxed interaction with children, resulting in more natural play and communication. More humanoid robots like KASPAR, Nao, and Zeno are equipped with more degrees of freedom, allowing for a more articulated range of motion and expression in a compact package. KASPAR and Zeno also leverage the use of an expressive face to add to the interaction. What this new version of the Huggable wishes to achieve is the expressivity of these robots in a more "huggable", furry form that will allow children to truly achieve physical interaction with a robot like this.

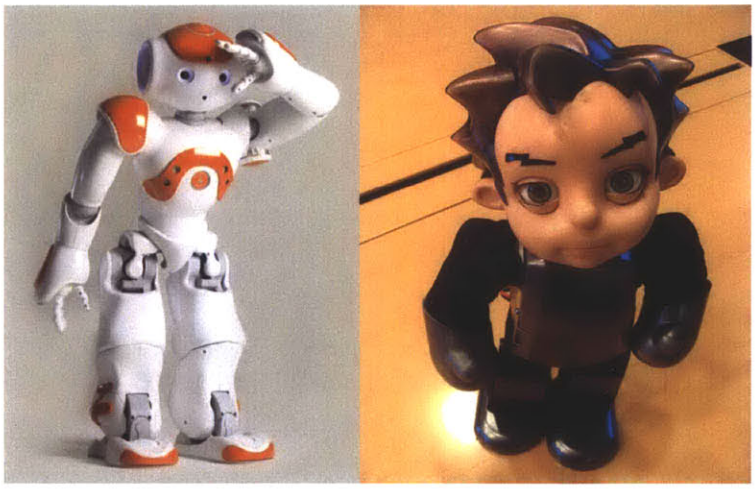


Figure 1-7: Nao (left) developed by Aldebaran, and Zeno (right), developed by Hanson

Lastly, this new Huggable project is based on the work of Dan Stiehl et al on the original Huggable project (pictured below in Figure 1-6). This robot was designed, like PARO, to be a therapeutic robotic pet surrogate for companion animal therapy.

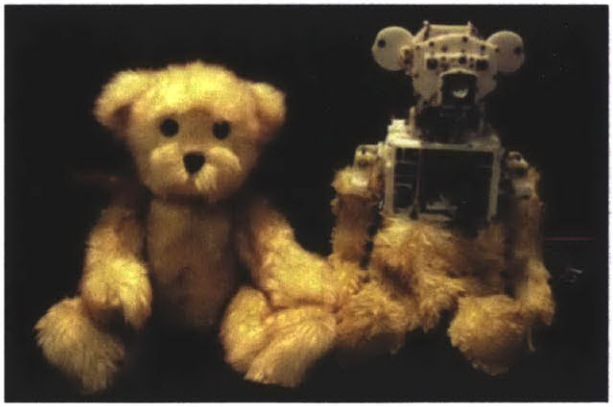


Figure 1-8: The Huggable Project by Dan Stiehl

Most of its initial work was based on developing a new system of touch feedback that could distinguish between different types of physical interactions [13]. Its potential applications have since expanded to include uses in education and long distance family interaction [14]. However, the most current prototype has not been able to be applied in a research situation for a number of reasons. Since its creation, its technology is outdated, its structure is rigid and heavy, and its system is way too complex for use.

Chapter 2

Design Process

2.1 From the Inside Out: Redesigning the Huggable

In order to create this SAR that would satisfy the requirements that the Personal Robots Group had in such a robot, the design of the Huggable needed to be completely changed. As Salter et al described, robots used in child-robot interaction should be sturdy enough to survive in the studies [11]. The old prototype was obsolete, rigid, and, above all, non-portable. With this in mind, one of the first design decisions that were made was to use a more portable computation system.

2.1.1 The Switch to Android

The Personal Robots Group, at the time of this particular redesign, was already perfecting a system that would operate the low level processes that a robot needed to function. One of the first of these platforms to utilize this system was Project LilGuy, which was collaboration between Samsung SAIT and the Personal Robots Group.

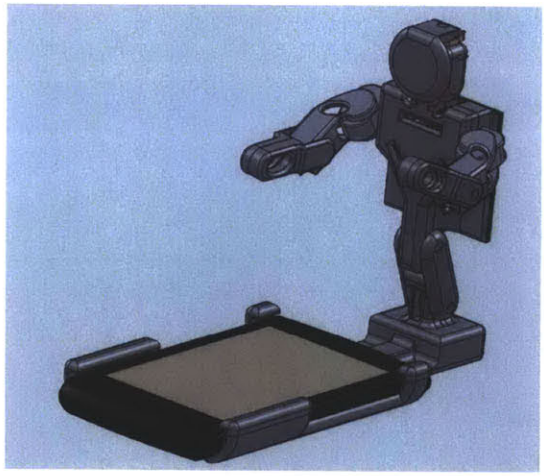


Figure 2-1: The LilGuy project, in solid model form

This project's purpose (shown above in Figure 2-1) was to provide a device that could control virtual characters wirelessly. The objective was achieved by successfully creating an Android application that could run our R1D1 codebase. Using this, the controller could translate the movements of the figurine and send them to a computer running one the virtual models of the robots owned by the Personal Robots Group [12]. The other platform in development is the Dragonbot (shown on the next page in Figure 2-2).

Figure 2-2: The Dragonbot Platform

The Dragonbot is a robotic platform whose operation relies on the computational power of an Android smartphone [13]. The phone generates its digital eyes and face, its microphone and speaker provide a pipeline for communication, its front-facing camera allows teleoperators of the robot to see what it sees, and it transmits/receives information wirelessly to an Android tablet or computer. The beauty of this system lies in its portability, and it was something that the Personal Robots Group wanted in the new version of the Huggable. The plan was to modify the Dragonbot architecture to suit the Huggable's needs. The phone would be placed in the head of the robot, and the screen would provide a pair of digital eyes that would shine through a removable mask. The phone would also process not only motor controller data, but sensor input as well. The resulting challenge was to design a head and body that could support it plus the weight of the phone.

2.1.2 New Motor Boards

Previously, the Huggable used a complex motor board that was difficult to sync with the R1D1 codebase. Also, the motor board itself was huge. This simply would not work in this new version of the Huggable. The next design consideration was to find a motor controller solution that would interface better. Luckily, Sigurdor Orn developed such a motor controller board for the Dragonbot, and they proved to be a robust solution. Sigurdor had been developing the MCB (pictured on the next page in Figure 2-3) for his own projects, and later improved upon it just in time for the new Dragonbot platform.

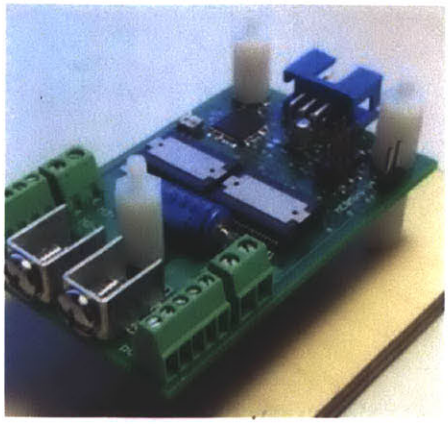


Figure 2-3: The MCBMini motor board

The Dragonbot platform uses a stack of three motor boards with a COMM board, and this stack works very well with the encoder-based Dragonbot motors. These boards are also equipped to handle potentiometer data, which is exactly what the Huggable would need. The desire was to use both encoder and potentiometer feedback, like the original Huggable system, in order to provide a smooth animation.

2.1.3 Additional Degrees of Freedom

In this new iteration of the Huggable, the Personal Robots Group wanted to increase the expressivity of the robot. It already had eight degrees of freedom: one for the ears, three for the head/neck and two for each arm. But with the recent development of "squash-and-stretch" robotic platforms, the group had discovered the richness of expression in the simple movement of leaning in with interest. This small motion gives a more lifelike attribute to the robot, and allows for a more engaging interaction. Thus, the group included this type of motion as an additional degree of freedom. What was also noticed was that the original prototype could not indicate physically when it was speaking. This lack of movement in the mouth emits a feeling of entrapment within the robot. Hence, the desire was to add a degree of freedom that would alleviate this, which would be a muzzle movement. The idea would be to synchronize the muzzle's "wiggle" with speech, so as to give the appearance of speaking. The last addition to the degrees of freedom was to give the Huggable the true quality of being "huggable". The older prototype had no way of being able to hug back. The lack of an elbow joint made it difficult for the Huggable to perform many different motions: pointing to itself, covering its face, etc. The group made sure that this degree of freedom was a priority to include, since this would give more depth to the arm movements.

2.2 A New Form: The Jetta Collaboration

In order to create this new iteration of the Huggable, the Personal Robots Group collaborated with Jetta Company LLC to do a complete redesign of its mechanical structure. With this instance, the group wanted to have the ability to "mass produce" the robot, so that many research groups could use the platform for their own studies. Thus, this iteration was to be designed for manufacturing, and Jetta had both the facilities and the experience to assist with the DFM process. The group was able to work very closely with the R&D sector of Jetta, and saw firsthand how a robot such as this goes from concept to product.

2.2.1 Initial Specifications

When a product such as this starts out as a concept, the first requirement is what is referred to as a "wish list". Jetta required initial specifications for what the robot will contain and how it will operate. The very first detail needed was the dimension. The Personal Robots Group wanted to reduce the size of the current prototype to something that a child could hold and manage. According to Jetta's experience, a robot of this nature would work best at a height between 250mm and 300mm. Jetta, a manufacturer for many children's toys, explained that a moving robot designed for children ages 5-8 cannot be too big so that they cannot hold it, but not too small that parts of it could potentially be a choking hazard or easy to break. Therefore, their height suggestion for a teddy bear robot was 280mm. The other dimensions (width and length) were roughly determined using a previous model of the Huggable in full bear form. The width was estimated to be in the range of 150-200mm, and the length was to be between 200-250mm. After specifying the overall dimensions for the robot, Jetta then inquired about the individual components of the bear, or features. Each part of the bear was to be described and listed with what it was supposed to do and contain. This process was part of their formula for creating an initial design. Specifying its dimensions and placing its features in the right areas, plus knowing what the final product was going to look like, would allow the designer to correctly proportion the body of the robot to properly house the components, and ensure feasibility.

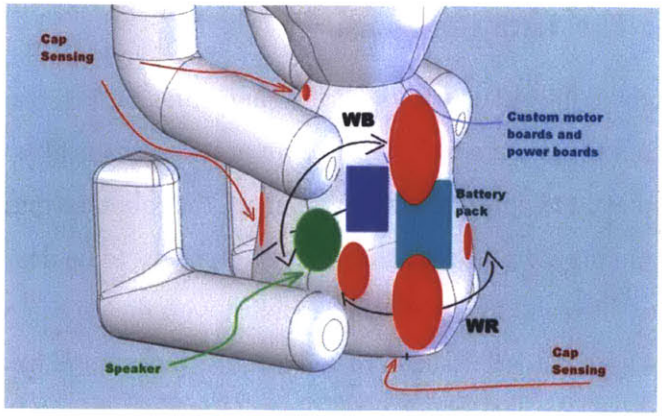


Figure 2-4: Example diagram from "wishlist"

The original feature list included in Appendix A uses the old version of the Huggable model to showcase what belongs where (an example is shown in Figure 2-4). Each section also lists the desired degrees of freedom, and their desired range of motion. Along with a list, there was to be a "diagram" that illustrated how all of the components were to be fitted inside the section. It also demonstrated the path of motion for the DOFs to show how exactly they would move.

Also, the diagrams showed the possible location of sensors for use in the Huggable. In the previous prototype, arrays of QTC (quantum tunneling composite) sensors were responsible for touch and pressure sensing. However, QTC is very expensive, and the goal of this new iteration was to lower the cost. Jetta recommended the use of capacitive sensors and pressure bladders. Capacitive ICs are very inexpensive, and the electrodes are easily customizable, being that they are as simple as pieces of copper tape. Pressure bladders attached to sensors are also inexpensive, but they would have to remain in specific areas. With that in mind, the locations of capacitive electrodes and possible areas of including pressure bladders were indicated. Once each section was outlined, the list goes on to describe the behavior of the motor and feedback system that was desired. Attributes such as voltage rating, maximum torque, and maximum RPM were essential in browsing for new components to use. Also, the type of feedback was important to list, in order to find a proper variable resistor (potentiometer) and/or encoder for the job. In the case of the Huggable, the group wanted to have the flexibility in using both for feedback purposes.

Lastly, the "wish list" needed a schedule. The schedule was important in assessing how much time was needed for each part. Jetta had a specific pipeline for doing a project like this, but this was a special case. Things needed to be done a bit faster, and so the schedule needed to reflect a faster, but feasible, timeline. Once this was completed, Jetta approved the requirements, and with the author, moved forward in the design process.

2.2.2 Brand New Outlook

Designing a prototype like this also needed the final physical appearance, or product outlook. The group had a model plush already made for the Huggable, but advisor Cynthia Breazeal and resident artist Fardad Faridi had a new plan in mind for how this new bear would appear. The original outlook of the Huggable resembled more of a "real" teddy bear, but this time, the design went for a more cartoonish look. Fardad wanted to stress in this iteration of the Huggable that the shape needed to look more, well, "huggable". The rigid shapes that dominated the old prototype did not give off any appeal for comfort, nor was it easier to wrap arms around it. According to Fardad (personal communication, July 28. 2012), humans are more comfortable wrapping their arms around a bean-shaped object similar to the body of a young child than anything else. The contours of the shape are more inviting for a hug than hard edges and lines. This idea of a neonate form factor was also expressed by Gould [18] in describing the change in form of Mickey Mouse over the years. The juvenile appearance of the form is naturally appealing to humans, and they feel more affection towards them than other forms.

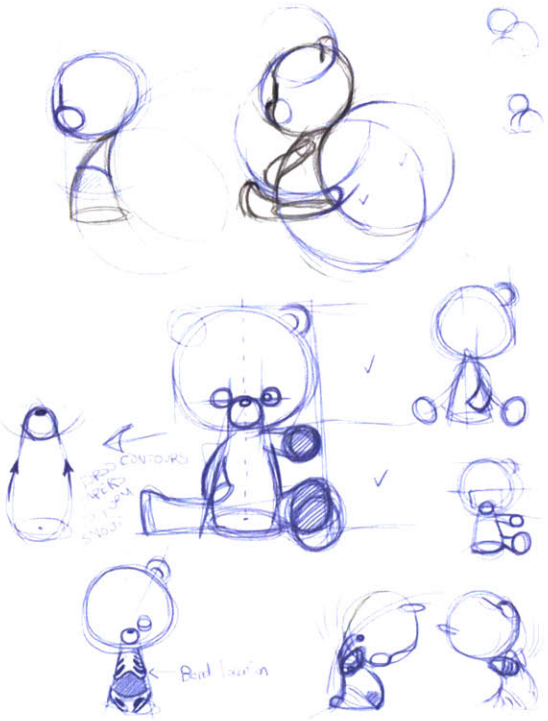


Figure 2-5: Sketches of new outlook by Fardad Faridi

Once this new outlook was created, the author worked closely with Fardad to ensure the physical possibility of this new concept (as shown in Figure 2-6). Certain aesthetic qualities, such as the size of the head in comparison to the body, the shape of the arms, and the thickness of body, had to be compromised to adhere to physical capacity and ability.

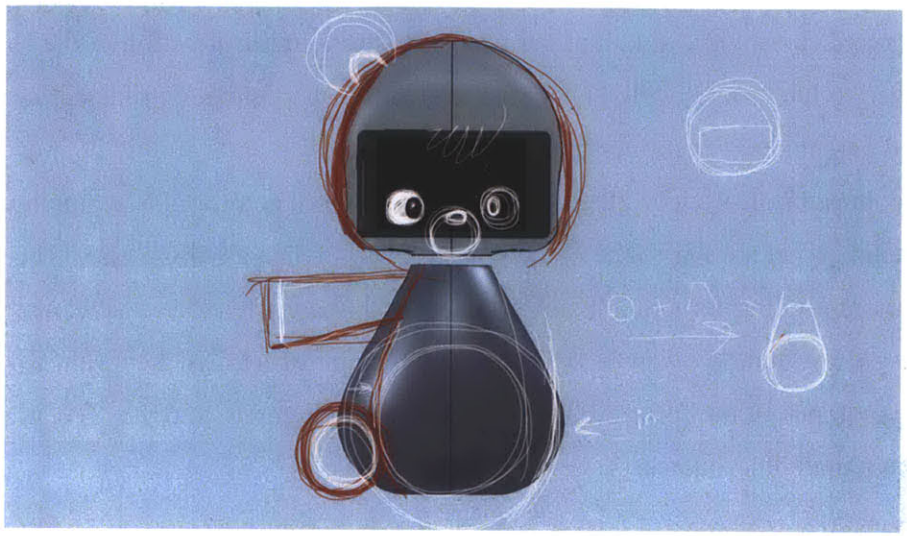


Figure 2-6: Example of graphic exchange between the author and Fardad

With a design this elegant, many components would have not normally fit correctly. Certain curves had to be widened to give room for motors and gears, especially near the back of the body. Also, areas near the neck and shoulders needed to be widened to avoid bunching of components. The right proportions had to be established with the head to ensure that the robot would not become too top heavy. With the phone already comprising most of the weight of the head, it was important that the Huggable was designed to support its own head, and have the ability to move it freely without tipping. With the addition of the elbow joint, the arms also needed to be lengthened and widened to ensure proper casing for components such as sensor PCBs and motors. Through frequent exchanges, a final outlook was determined, and the DFM process began.

2.2.3 Developing a Mechanical Layout

Once the features, dimensions, and appearance of the product were established, the next process was to begin defining the internals. Component placement, basic mechanism design, and to-scale depiction were all comprised in the process of what was referred to as the mechanical layout. The mechanical layout was a simplistic scale drawing of how everything "fit" in an outline of the

final outlook of the bear. The author and the director of R&D at Jetta, S.K. Wong, exchanged multiple versions of this layout before approving it for detailed design. At first, there was some confusion as to how this mechanical layout was to be structured. As shown below in Figure 2-7, the first layout done by the author is very different than the one provided by S.K.

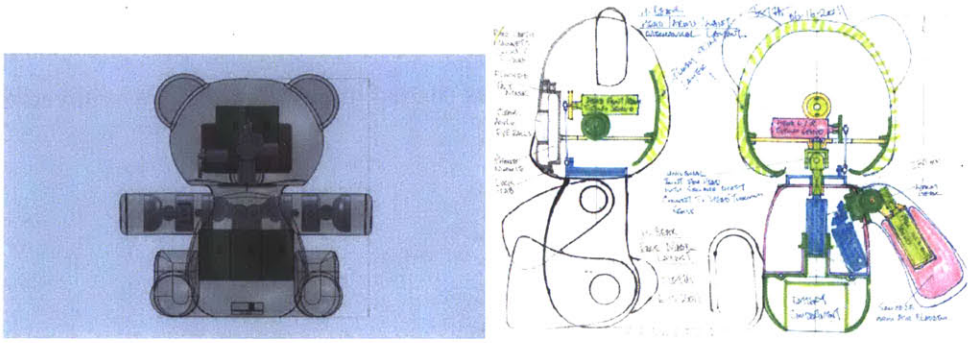


Figure 2-7: Side-by-side comparison of mechanical layouts. Left is done by the author, right is done by S.K. Wong

However, with example layouts provided by S.K., the idea became clearer, and the layout became more and more detailed over time. Instead of sketching by hand, the author used the 3D to 2D sketch ability of Solidworks to gain a 3D perspective on how parts were fitting within the outline, and communicate a 2D front and side view that is easier to read for Jetta's design team. With such a complex series of curves that comprise the outline, it was difficult to see from just the 2D views if parts actually fit within all of the boundaries. Using Solidworks to generate these mechanical layouts guaranteed that the placement of these to-scale components would fit in all of the boundaries.

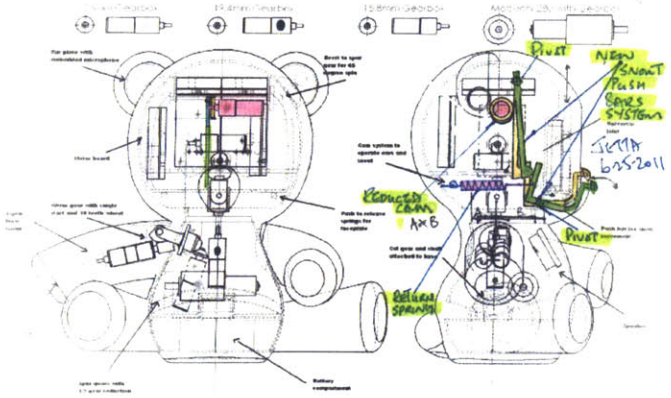


Figure 2-8: Combined editing of the author and S.K. Wong on mechanical layout

2.2.4 Workflow: From Solidworks to ProE

Once the layout was finalized and approved by both Jetta and the Personal Robots Group, it was passed down to Jetta's R&D Team to begin designing detailed CAD models of the prototype. The author had a firsthand look at how these mechanical layouts become complex 3D models. A team of three people divided the robot into modules, and began generating the models using ProE Wildfire. The author worked alongside them using Solidworks, guiding the shell designs with the contours specified by Fardad's final outlook. Once the shell was shaped in Solidworks, it was saved as a .STP file and given to the team to modify.

Figure 2-9: Example CAD model from ProE

After sending a rough shell of both the head and the body, the author oversaw and observed the design process that the R&D team used, such as regulating wall thicknesses, using fins to support cylindrical structures, and check screw patterns. JPEGs of the work in progress were forwarded to the Personal Robots Group for approval and editing. Once the final edits were made, Jetta began production on a prototype to be shipped to the Personal Robots Group, which would be populated with the necessary components by them.

2.3 Post-Machining and Prototype Modification

Figure 2-10: The Huggable prototype, before post-machining

After receiving the prototype, many modifications had to be made in order to get the Huggable prototype operational. The prototype's shells were milled out of ABS plastic, using a CNC machine, and bonded together. This material made it easy to make physical edits that were greatly needed. One major modification was the removal of the ear/muzzle mechanism and adding housing for a small motor, making the ears move independently of the muzzle. As this DOF was being tested, there seemed to be a lot of friction through the timing pulley stages going up to the gear that spun the ears forward/backward. Strangely enough, the whole structure inside of the head would warp when the head would close, preventing smooth movement; this was even after reaming all of the nylon bushings and smoothening the half-socket in which the ear shaft was rotating. Shell warping due to fastening seemed to affect the operation of many of the joints, so applying the right amount of torque to the machine screws became a sensitive matter. Also, friction in the shoulders and elbows, as well as other places, had to be smoothened out with a rotary tool due to incorrect tolerances with the ABS parts. Additions needed to be made to certain joints to make them operate smoother, such as the neck. The inclusion of a ball bearing in the neck rotation was highly necessary

for operation, and greatly improved the smoothness of the DOF. But above all, an extensive amount of drilling, cutting, and widening needed to take place to account for wiring. There were many wires extending from all parts of the robot, and most had to branch into the head (where, for example, the motor controller boards were placed). There was either not enough room for wires to pass through the robot, or no room at all. Unfortunately, these designs errors were not caught earlier; examining screenshots of the models were not conducive to scrutinous observation. Hence, these (and other) modifications had to be done by hand, and were noted along the way to include in a follow-up report for future iterations.

Chapter 3

The Hardware and Software of the Huggable

3.1 The Huggable: Current Prototype

This thesis will now present the current design of the Huggable Project. It first describes the mechanical design of the robot. This will involve all design decisions pre- and post-modifications. Then, it will list and explain all of the electrical components contained in the Huggable. This includes components purchased and MCBs designed by the Personal Robots Group. Finally, it will walk through the software implemented in the robot. This will showcase the pipeline that goes from R1D1 to Android to controlling interface, and vice-versa.

3.2 Mechanical Design

Table 3-1: Degree of Freedom List with Specifications

DOF	Range (Degrees)	Motor	Gear Ratio	Max Speed (RPM)	Continuous Torque (mNm)
Ear Wiggle	+20, -20	Faulhaber 1016	48:1	343.75	45
Muzzle Wiggle	0, -15	Faulhaber 1516	28:1	460.71	100
Head Tilt	+15, -15	Faulhaber 1516	152:1	84.87	150
Head Nod	+15, -30	Faulhaber 1516	152:1	84.87	150
Head Rotate	+60, -60	Faulhaber 1516	112:1	115.18	300
Shoulder Rotate	+30, -90	Faulhaber 1516	415:1	31.08	300
Shoulder Lift	+65, -15	Faulhaber 1516	280:1	46.07	200
Elbow Joint	0, -80	Faulhaber 1516	52:1	248.08	100
Waist Bend	+15, -10	Faulhaber 1724	989:1	7.99	300

3.2.1 Motor Selection

The Personal Robots Group sought the expertise of Jetta in finding a cost-efficient motor solution for the robot. For the designated DOFs, the group needed motors that would provide an angular velocity of about 15-20 RPM. This speed was fast enough to make a significant movement, but slow enough to not surprise or scare children. Certain DOFs (ears, waist) were designated to have different RPM requirements due to the weight (or lack thereof) of the part moving. The group also stipulated that the motors were rated for 12V. The MCBMini that was being implemented required the motors that were being controlled to have a minimum voltage of 9V, but for optimal operation, the motors should be rated at 12V. This was a surprise to Jetta, since a voltage this high is uncommon for a product such as this. In the toy industry, manufacturers tend to design in the range of 3.3-5V, with a maximum rating of 7-9V. The size of these desired motors, along with the rated voltage, was rare to find, being that smaller motors (especially for this purpose) are more common in

the 6V range. However, Jetta was able to find a motor company that could provide 12V motors at the scale that we desired. Two motors sizes were selected: one for the waist (being the heaviest DOF), and one for use throughout the body.

Figure 3-1: Original Gear Motor for use in Huggable, from Twirl Co.

These were planetary gear motors, and the provided data sheets gave potential speed/torque outputs with the configured gear ratio. Unfortunately, the Huggable could not utilize the selected motors because they did not support encoder feedback. The only other motors that could perform at the size and voltage desired were from Faulhaber, a company very familiar to the Huggable project. The group was able to obtain motors of the same voltage with a slightly smaller package size and a bit more torque than the previous selections [19] [20] [21] [22]. Thus, the robot was equipped with both potentiometers and encoders for feedback, but an unforeseen problem came about when using the encoders. Since many of the motors had very high gear ratios, the MCBMini could not actually keep track of the sheer number of ticks being produced, which rendered the encoders useless. As of now, the current feedback system of the Huggable is purely position-based using potentiometers. In the sections that follow, all DOFs will have each motor and gearhead listed, along with the mechanism they power.

3.2.2 The Head

Probably the most nontrivial part of the robot, the head was extremely tricky to design. Due to the nature of the outlook, this robot was already going to be a bit top heavy; the head was pictured to be nearly the same proportion as its body. Also, this vessel was tasked with containing the Android phone that gives function to the robot, and the phone is easily the heaviest object (6.0

ounces or about 170 grams) in the robot by comparison [23]. With that in mind, there were a couple of goals to achieve with this part of the design:

- Keep as many motors out of the head as possible
- Center all motors to avoid them being tangential loads
- Allow for plenty of room for wiring

The head had four distinct DOFs which required the motors to be within the head: ears, muzzle, head-nod, and head-tilt. The ears and the muzzle were very lightweight DOFs that demanded the least mechanical strength, so a plan was formed to combine these DOFs. A very common practice in the toy industry is to make a motor operate two different DOFs in the same range of motion. For example, a certain range may move a pair of eyelids, but past that, the motor will move an ear or a mouthpiece. This combination seemed like a very practical idea for reducing the number of motors and increasing the efficiency of the robot. The motor used in this mechanism was a Faulhaber 1516 DC-Micromotor paired with a 15A plastic gearhead with a ratio of 28:1. The mechanism developed to do this was a cut-gear operated by a motor placed under the chin of the robot (which kept its center of mass low). The motor pushed a cam that pressed on a spring-loaded lever that made the muzzle wiggle. The shaft on which this system rotated contained a pulley that transferred motion (through an intermediate stage of pulleys) up to a pulley that spun a cut gear. When the cam operated the muzzle, the cut gear did not engage the shaft that controlled the ears. Past that limit, the ears would engage forward or backward, depending on which way the cam moved last. Fortunately, the range of motion combined fit in the range of the potentiometer, so the motion never hit a "dead zone" in the device.

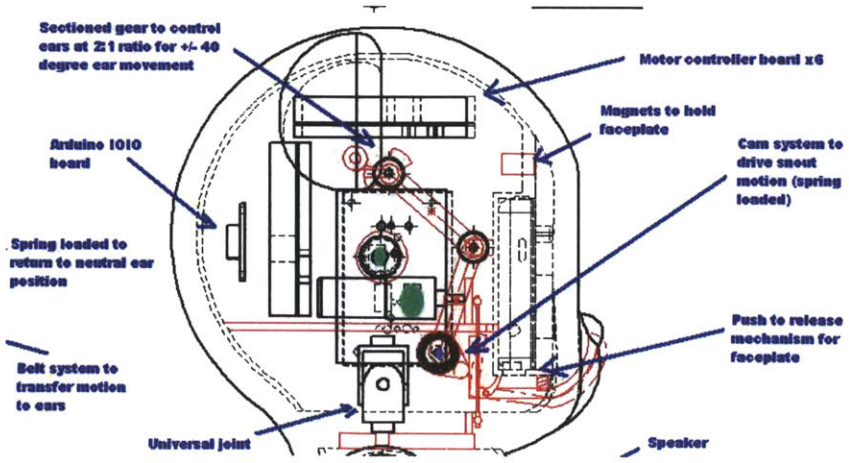


Figure 3-2: Clipped image from mechanical layout showcasing the cut gear system

Unfortunately, there were two downsides to this mechanism. One, the ears were susceptible to dislodging if the mechanism was operating the muzzle, since the ear shaft would be spinning somewhat freely in the cut-gear zone. Two, there would be a delay in moving the ears past their zero point since the muzzle had to engage first before going to either extreme on the ears. As previously mentioned, the mechanism itself was having trouble running smoothly due to warping, so all of it was scrapped. Instead, the muzzle was left operating independently of the ears, and the ear mechanism was changed to a direct spur gear drive, using a 3:1 gear reduction leading to a Faulhaber 1016 DC-Micromotor equipped with a 10/1 metal gearhead with a ratio of 16:1 [24][25]. This setup is depicted below in Figure 3-3. The motor was contained in a custom-machined housing built into the core of the head motor casings. This way, the ears could be back-driven without becoming dislodged.

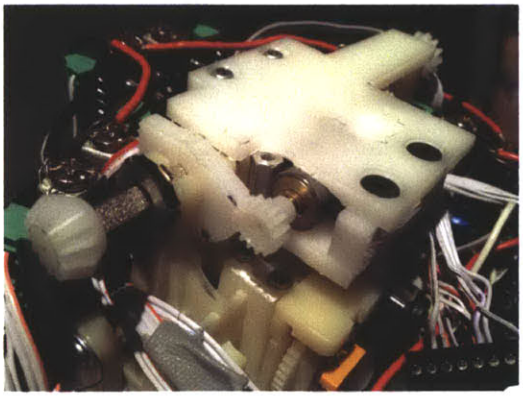


Figure 3-3: New independent ear mechanism with custom housing

Another note about the ears: the head shells were designed to come together and form a series of shaft holes for the ears about which to rotate. Again, the warping of the plastic caused irregularity to the movement, so custom-machined bearing blocks were made for the ears that screw into the front head shell, shown in Figure 3-4 (next page). This way, taking apart the head does not allow the ears to fall out, but stay in one place. This also allowed anyone operating the DOF to ensure its smoothness without having to constantly take apart and put back together the whole head; only the front head shell needed to be in place.

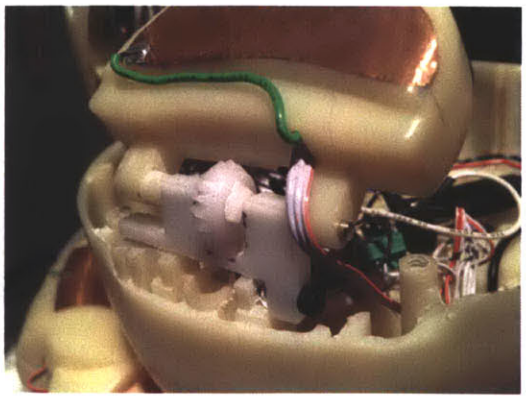


Figure 3-4: Ear bearing block attached to front head shell

To design the two essential DOFs of the head (head-tilt and head-nod), Jetta recommended another common practice in toy design. Previously, the Huggable's head was built within these two mechanisms. This made the head very rugged and sturdy, but very stocky and bulky.

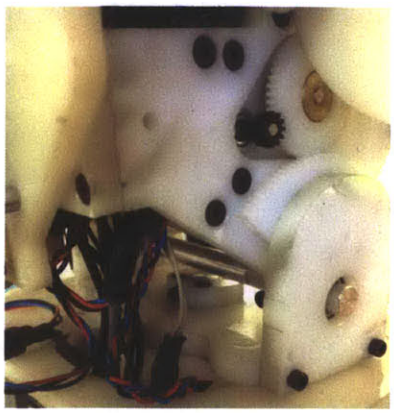


Figure 3-5: The old Huggable's head-nod mechanism

Instead, the recommended (and later approved) design was to use a cam system using ball-jointed rods and plastic arms, as shown in Figure 3-6. At the end of each motor was a plastic arm whose length controls the range of motion of the DOF. This arm is connected to a ball-jointed rod whose other, ball-jointed end was pinned to a base (which would effectively be the base of the neck). As the motor rotated, the plastic arm pushes or pulls the rod, which pivots the whole head about an axis. To achieve two axes of motion in the neck, a metal universal joint was implemented, allowing the perpendicular axes to be close to one another and save space. The movements were approximated to have similar loads to move, so each DOF was fitted with a Faulhaber 1516 DC-Micromotor and 15A plastic gearhead at a 69:1 gear ratio. (As further testing proceeded, these motors were switched with ones containing 15A gearheads using a 152:1 reduction.) The motors

were, like the muzzle motor, placed as close to the rotation axes as possible, and centered to reduce the effects on the head's center of mass. One flaw in this mechanism is the tendency to overextend the rotation, thereby inducing a singularity at the top/bottom of rotation. The robot often cannot free itself in these positions, so both hard and soft stops were included as part of the system.

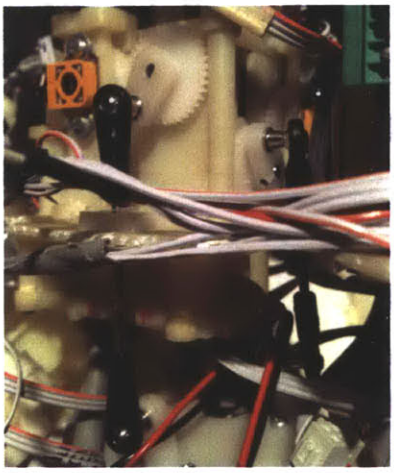


Figure 3-6: The ball-jointed cam system

Once these mechanisms were placed in the head, it was noted that there was still considerable space. As space was quickly running out of the rest of the body, it became logical to include the IOIO board and the MCBMini boards (discussed later) in the head. The phone was already in the head, so it made sense to place the IOIO near it (see Figure 3-7).

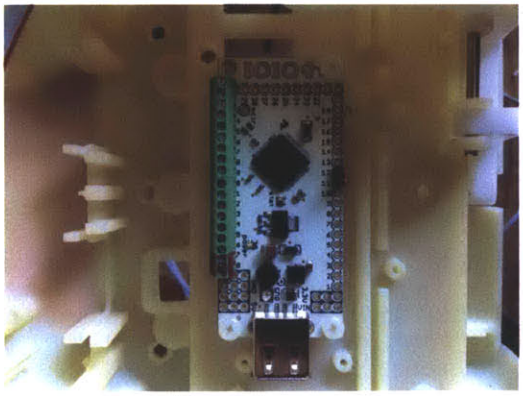


Figure 3-7: The IOIO as positioned onto the back of the front shell

The MCBMini controls two motors at a time, which meant that the Huggable demanded six MCBMinis to cover all the DOFs. A stack of six boards would not fit anywhere else in the robot, but

they did fan out nicely in the empty space inside the head. Organized in the back of the head, the MCBMinis provided a bit of counter weight for the phone, which helped in the head-nod DOF.

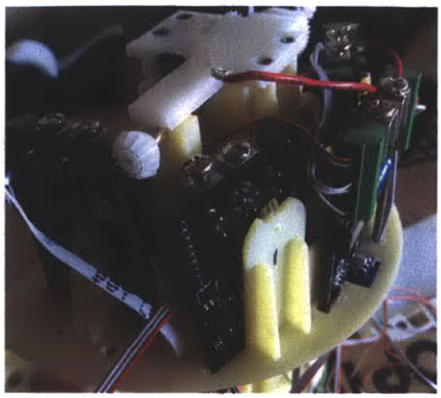


Figure 3-8: Four of the six MCBMini board oriented in the head

Originally, there was a plan for an external camera, which would have been placed somewhere in the nose. However, it was impossible to wirelessly stream the quality of video that the Personal Robots Group desired, and the necessary hardware for external video processing would have not fit anywhere in the robot. Thus, the forward-facing camera on the phone is used for video streaming, similar to the Dragonbot platform. And, similar to the Dragonbot, a mask was designed to cover the phone face and allow the digital eyes and camera to shine through. The mask used precious metal magnets to snap to the face, guided by two slanted pegs on the bottom that fit into two respective slots. The camera looks through a specially designed "lens" that presses into the mask. It's an insert that pushes away the fur covering on the mask, and opens wide so that the camera's field of vision is unobstructed. Both the mask and the insert were 3D printed at the Media Lab after the Huggable's delivery due to edits in the design and outlook. Using the generated surface provided by Fardad, a new mask was printed using the specifications from the original mask (magnet holes, camera and eye holes, etc.) which showcased a more contoured face. This gave the Huggable a final cosmetic look of childlike innocence, as shown on the next page in Figure 3-9.

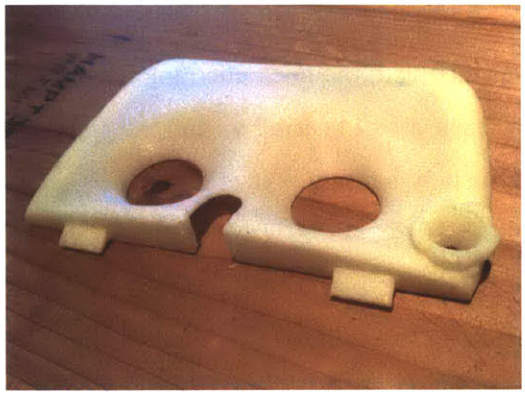


Figure 3-9: Final 3D Printed Mask

3.2.3 The Body

The last DOF that was to control the head was actually placed down in the body. The head-rotate (neck rotation) DOF was a rather simple task to complete. The plan was to simply use a geared motor to directly drive the whole head on an axis that went straight through the U-joint. A plastic axle was created that pinned through the base of the U-joint and the neck base, passed through a plastic bushing, and fitted to the motor shaft. The fitting used a D-shaped hole, and the motor shaft was flattened to match this D-shape. This is a common practice to ensure a nonslip rotation and torque transfer; it is used throughout the robot with all plastic fittings. The plastic bushing is secured at the top of the body when the two body shells come together, and the motor is encased onto the back body shell. The motor is a Faulhaber 1516 DC-Micromotor equipped with a 16/7 gearhead at 112:1.

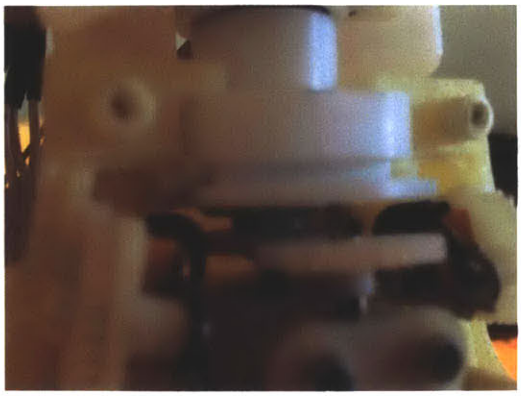


Figure 3-10: The head rotate mechanism, in the back

When the robot was received, the neck encountered a lot of friction trying to move around. The solution to this was to insert a ball bearing within the plastic bushing, and machine a new plastic axle the would fit through the new bearing. Needless to say, smoothness of movement was drastically increased (Figure 3-10).

Three other DOFs were to be placed within the body as well (hence the reason why some of the original component locations had to be moved elsewhere). The waist bend DOF was originally designed to be two distinct motions. The Personal Robot Group first desired that there would be a small waist-turn as well as a waist-bend. Unfortunately, there could only be room for one DOF, and the decision was to go with the waist-bend. Since this robot was going to be top-heavy, the waist had to have a very restricted range of motion. Finding a motor that could support a bigger range of motion would not be able to fit inside the robot, and would add much more weight. Thus, the DOF was restricted to about 15 degrees on both sides (front/back), and a high gear ratio / slow RPM geared motor was sought. These requirements would ensure that the motor's size would fit inside the robot. The motor was fixed to the top half of the body, and rode along a cut spur gear that was fixed to the base of the body, as shown in Figure 3-11 (next page).

Figure 3-11: Waist bend mechanism and shaft clips for waist

The motor that was selected for this task was a Faulhaber 1724 DC-Micromotor with a 16/7 gearhead using a 989:1 ratio. The body shells are guided along circular edges of the base shell, and secure in the middle to a metal shaft. Each shell "clips" onto the shaft with a press fit, which allows each shell to be easily removed from the base.

With these two DOFs positioned in the top half of the body, space was running low. However, there was just enough space on the sides and in the very front for the arm-rotation DOFs and the speaker. In the old prototype, the arm-rotation DOF was placed parallel to the axis of

rotation, which ensured motor efficiency. However, this took up a lot of room in the chest cavity, and another solution was necessary. Due to the delicate, tapered shape of the Huggable, the parallel configuration was no longer an option, especially with the elongated shape that the geared motors had. The solution: use bevel gears at the last stage of motion.

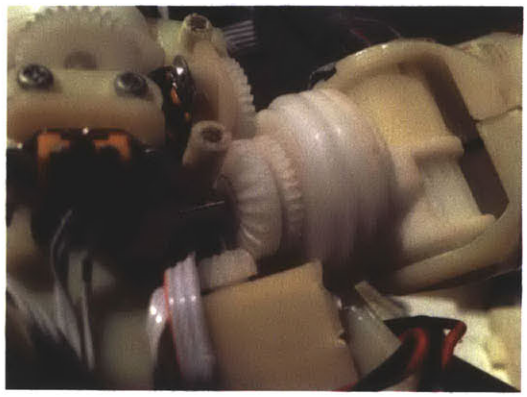


Figure 3-12: The arm rotation mechanism

The motors could be oriented along the side this way, perfectly fitting under the curves. At the end of the geared motor, a bevel gear was placed to interact with its respective gear belonging to the arm, as shown above in Figure 3-12. The shaft on the arm that contained this bevel gear would be supported on either side: one side rotated within a nylon sleeve that was press fit into the core of the robot, and the other rotates in a special bushing block secured by the two body shells. This DOF is powered with a Faulhaber 1516 DC-Micromotor equipped with a 16/7 gearhead at a ratio of 415:1. The high gear ratio ensures a strong and slow movement that will avoid the appearance of "wild arm swinging" that could scare off a child. This movement is also ratcheted to prevent damage to the gears and motor when the arms are over-torqued.

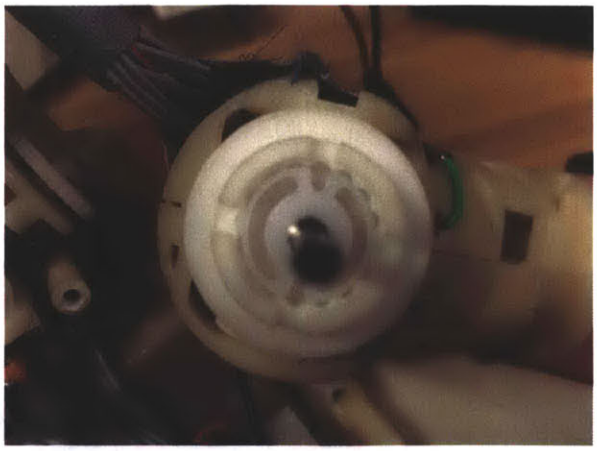


Figure 3-13: Ratchet mechanism to prevent gear damage from over-torque

This ratchet mechanism (shown above in Figure 3-13) is also used in the shoulder lift and elbow joint DOFs (discussed later). The last component to be placed in the top of the body was the speaker. Previously, the speaker was located in the face, so that the voice appeared to originate from the bear's mouth. However, the phone now resides in the front of the face, and a new location was needed. The only other location that made sense was in the front of the chest, and so a separate cavity was formed in the front body shell to house it, as shown in Figure 3-14 (next page). The cavity conforms to the shape of the speaker, and resounds through a perforated plate that attaches to the shell, providing continuity to the body shape. A small, rectangular hole in the back of the cavity allows for the speaker wires to reach the speaker amp circuit (discussed later).

Figure 3-14: Speaker cavity in chest plate with speaker

3.2.4 The Arms

The arms were also tricky modules to design, because of the demands for a double jointed movement and slim profile. The plan was to encase two motors in each arm, with a dependent joint that bent inward. That would make three movements in the arm: shoulder, elbow, and wrist. However, due to the size of the motors in consideration, the arms would have had to be almost as long as the body is tall, and that would destroy the outlook of the robot. The final design plan was to just have two DOFS in each arm, along with pressure bladders in the "paws". These pressure bladders were originally supposed to be in the legs, but (as discussed in a later section) a better plan was to have these house the batteries. Thus, the "paws" were designated to be pressure bladders, which made practical sense. The Personal Robots Group imagined a scenario in which the Huggable reached out to a child and asked for them to squeeze its paws in order to communicate pain. This would be an alternative way of having a quantitative measure of the child's "pain level" if the child could not verbally express it.

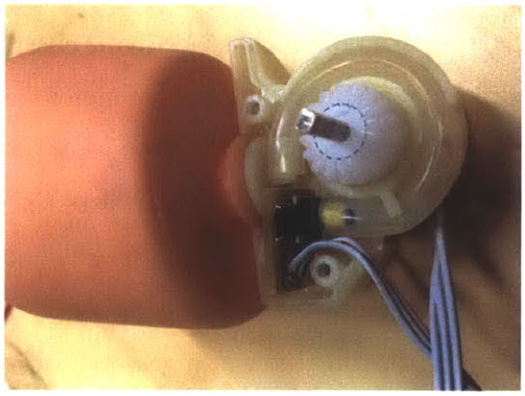


Figure 3-15: Pressure bladder "paw" in forearm piece

The pressure bladder is secured to the arm via press fit (as shown above in Figure 3-15), and a hose extends from it that feeds into a pressure sensor (discussed later). The two DOFs (shoulder and elbow) were designed using two separate mechanisms, but both utilized the ratchet mechanism described previously. This was especially important for the shoulder, because the proposed mechanism would normally not support back-driving. To keep a slim profile, Jetta recommended a worm gear configuration. Although the downside of this was a noisy mechanism, the worm gear allowed for a very high gear ratio in a very small space, which is exactly what the Huggable needed for its shoulders.

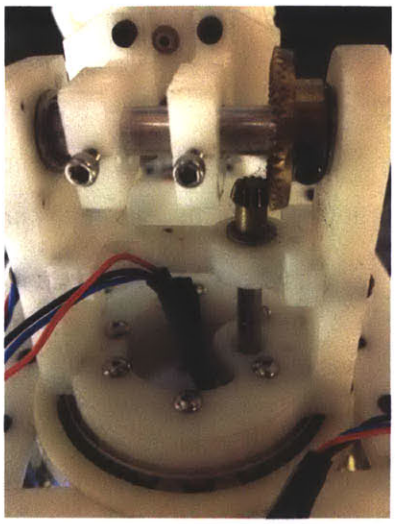


Figure 3-16: Bevel gear mechanism to operate shoulder lifting in old Huggable

Previously, the shoulder mechanism was a very complex gearbox that ended in a bevel gear. This is shown above in Figure 3-16. This, similar to the arm rotation mechanism, gave the Huggable very broad shoulders, and cut into the horizontal space in the chest. The most logical plan was to move this DOF into the arm itself. The orientation of the motor that made the most sense was parallel to the arm, which lent to the idea of using a worm gear. The worm gear would travel along a thick spur gear attached perpendicular to the arm rotation shaft (mentioned earlier); whose interior contains the ratchet mechanism. A worm gear cannot normally back-drive, and when made of plastic and put in a children's toy, it would be susceptible to breaking. The ratchet lets the spur gear click into place if pushed past its limit, allowing the worm gear to "back-drive" in a way. The worm gear is at a 10:1 gear ratio, which is powered by a Faulhaber 1516 DC-Micromotor using a 15A gearhead at 28:1, making the entire gear reduction 280:1. This high gear ratio provides a strong movement for lifting the entire arm without lifting too fast or too slow. The placement of this DOF left just enough room for the elbow DOF, which would use a similar motor orientation, but a different mechanism. This DOF was fairly simple to design; all that was needed was a bevel gear attached to a motor. The other bevel gear would be fixed to the "forearm" shell, with a ratchet tucked underneath to prevent damage. This movement was not torque-heavy, so a Faulhaber 1516 DC-Micromotor that had a 15A gearhead at 28:1 (later switched to 52:1) was used to power the DOF. The increased gear ratio helped slow the movement a bit and give it a bit more torque to push through the joint. Unfortunately, this joint was particularly susceptible to sensitivity in over-tightening of the screws. Too much, and the gears would be pressed together so tightly, the elbow would not budge. Too little,

and the gears would slip, causing the elbow to get off track. Also, this joint needed a lot of post-machining to make smooth and clear a way for wires to pass through to the head. For example, the potentiometer in this joint, when received, had no foreseeable access to its terminals before post-machining. Also, the arm pieces that served as bearing blocks to the forearms had too tight of a tolerance, preventing smooth rotation. With a little bit of post-machining, the arms became functional enough for operation.

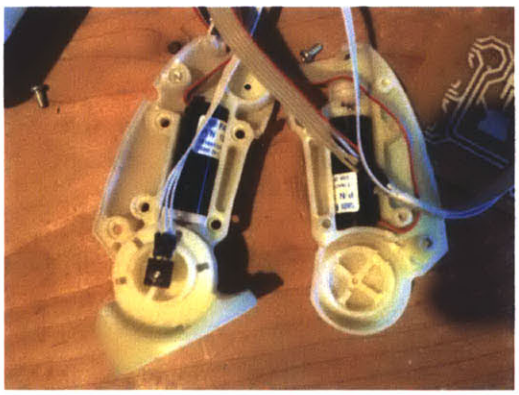


Figure 3-17: The upper arm, showcasing each motor for their respective DOFs

3.2.5 The Base/Legs

After the body had been divided by the waist DOF, the bottom of the body was designated to be the base, in which the batteries and the SEED power management board would be held. The legs originally were going to have passive ankles for simple play, and pressure bladders for sensing. But with space in short supply, the legs would be better designed to carry the batteries. This proved to be ideal counterweight for the combined lean of the head and the waist. The legs also provided ample space for long-lasting batteries, which are discussed in a later section. Originally, the legs were designed to be rigid, so that the bear was always in a seated position, and provided a sturdy base for movement. However, picking up the bear was a problem, as the legs would get in the way. There was no comfortable way to hug the bear, so the legs were modified to swing down when pushed. Luckily, the battery wires pass through the center of rotation, and do not interfere with this motion.

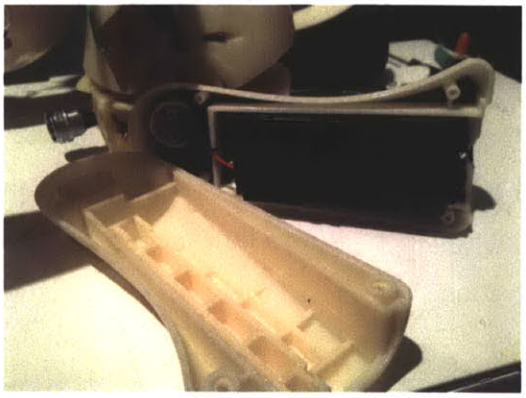


Figure 3-18: Newly printed legs with enclosed battery

The wires pass through a hole in each leg and connect to the SEED power management board. The board (also discussed in a later section) was used in the Dragonbot architecture to provide power to both the logic and the motors. This also served well for the Huggable, but the current board did not fit well in the base. Fortunately, the board was able to be clipped on one end due to inactive connectors, and secured to the underside of the top plate. The charge connector for this board was rather big, and needed access from the back, which was unavailable in the present design. The "tail" of the shell was therefore cut off, and the connector was left exposed for easy plugging as shown on the next page in Figure 3-19.

Figure 3-19: Back view of Huggable to show power port

The fur suit (discussed later) has a flap that covers this hole when not in use. And the most important component, the on/off switch, was added post-machining. A simple dual-pole, single-throw switch was added to the bottom of the bear, to make it very similar to most toys with this kind of switch. Adding it flush to the bottom made sure the switch was unobtrusive yet accessible.

3.2.6 The Fur

Figure 3-20: Black fur suit for the Huggable, in separate pieces

The Huggable robot would not be complete without a cute, cuddly exterior. Jetta generated a pattern for the Huggable to cover three main parts: the head, the body, and the mask. These patterns were also equipped with stretch fabric under the arms to allow for unhindered movement. However, further testing showed that stretch fabric would be needed in other locations, such as the ears and the elbows. There was a significant amount of bunching that would prevent motion of the joint; the stretch fabric would aim to prevent this. The body was later divided into two halves that joined together with Velcro to simplify the robing process. The fur needed to be easily removable for quick changes at Children's Hospital and other research sites. Velcro was added to the back of the head as well, along with the covering for the face mask. The hole in the face mask was widened more to leave room for the "lens" insert to fit. Different types of material (such as easily cleanable / machine-wash surfaces) are also being researched for later use.

3.3 Electrical Hardware

Table 3-2: Component List for the Huggable

Type	Component	Count
Computation Device	HTC Evo 3D from Sprint	1
Motor Controller	MCBMini Motor Controller	6
Android Interpreter	Sparkfun Android IOIO	1
Position Feedback	Panasonic EVW-AE4001B14 Variable Resistor	1
	Potentiometer Variant #1 (Brand Unknown)	7
	Potentiometer Variant #2 (Brand Unknown)	4
Touch Sensing	Atmel AT42QT1011 Capacitive IC	12
	Custom PCB	12
Pressure Sensing	Honeywell SXSMT100 Ceramic Pressure Sensor	2
	Analog Devices AD8223 Instrumentation Amp	2
	Custom PCB	2
Power Source	14.8V Custom LiPo Pack w/ Protection PCB	2
Power Management	SEEDPower Management Board	1
Audio Output	Anpec SA4871 Audio Amplifier	1
	Custom PCB	1
	2W 4Ohm Speaker	1

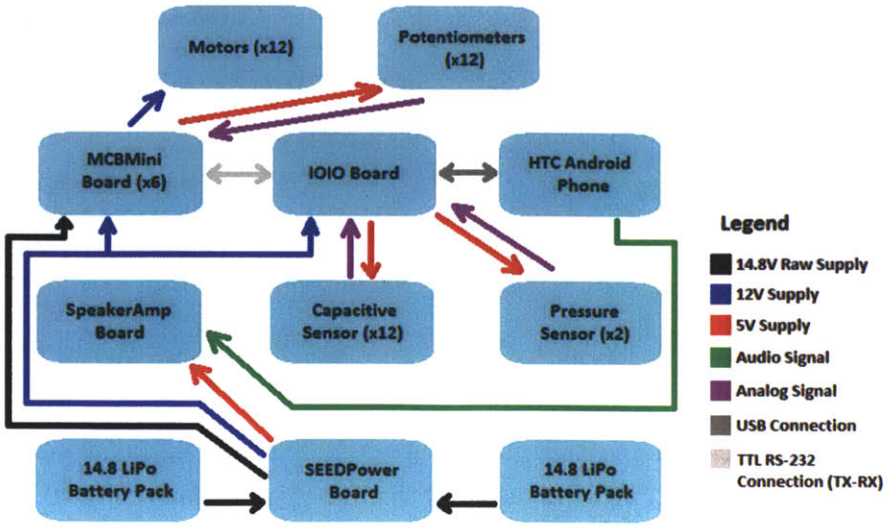


Figure 3-21: Electrical diagram showcasing component connections

3.3.1 SEED Power Management Board

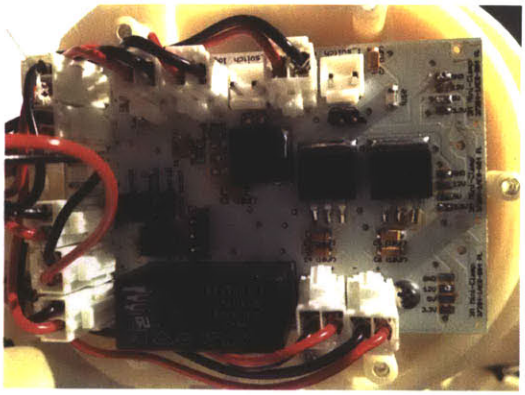


Figure 3-22: The SEED Power Management Board, as fixed in the Huggable's base cap

The Personal Robots Group decided to use a customized power management solution for use in the Huggable (Figure 3-22). Adam Setapen, the creator of the Dragonbot platform, developed a power board that catered to the needs of the Huggable's electrical system, which was designed to be very similar to the Dragonbot system. The board can provide power at 12V, 5V, and 3.3V, support battery charging alongside external power operation, and separate motor and logic power supply for protection to either source. The board can take up to 3 batteries for power input, and can provide 4 different power outputs (the 3 voltage ratings listed above, and a raw voltage supply for motor power). The board supports a DPST (dual pole, single throw) switch for powering the robot on/off, and uses an on-board relay to switch between battery operation and external power operation. Since the board at its current dimensions would not have fit in the base of the bear (and since the Huggable did not have use for 3.3V power) the connectors at the end of the board were trimmed, which provided an even more compact power solution for the Huggable. The robot uses the raw power line to operate the motors, the 5V line to power the speaker board, and the 12V line to support the IOIO board [26].

3.3.2 Batteries

Battery development was very crucial in the design of the Huggable. In order to create a more portable Huggable, the Personal Robots group needed to have a robust battery solution that would last a decent amount of time (1-2 hours of continuous operation max). Since a similar situation faced the Dragonbot, the author sought Adam's advice on what the best battery pack would be. The idea was to find a battery pack that could charge through its discharge wires, and was rated

at 14.8V. Adam's battery charging station was already equipped with chargers that would handle batteries at this voltage, so the smart plan was to find batteries that could work with this charging station. After scouring many different companies for a pack that could fit within the shape of the legs without any great result, the plan changed to create a custom battery pack.

Figure 3-23: LiPo battery cells and Protection Circuit Module from Powerizer

A series of four 3.7V, 1100 mAh LiPo battery cells connected to a protection PCB (for safe charge/discharge) was dimensioned to be the perfect fit inside each leg of the Huggable. The PCB is a Protection Circuit Module for use with 14.8V Li-Ion batteries at 10A, equipped with an equilibrium charge function (which is very critical for multiple-cell LiPo packs). The batteries are encased into the legs; the only way to remove them is by removing the legs themselves. Originally, plates at the base of the feet removed to reveal accessible batteries, but that was when the original batteries were much smaller. These plates can still be removed, but the batteries cannot.

3.3.3 MCBMini

As was previously mentioned, the Personal Robots Group was able to use a new piece of hardware for motor control. Sigurdur Orn developed a robust motor controller solution called the MCBMini. The MCBMini motor board measures 68mm by 43mm (which is a fairly small size) and is ideal for use inside the Huggable robot. The board uses an ATMega328 microprocessor with a pair of Cirrus Logic SA57AHU-FH H-bridge chips to control up to 2 DC motors at a current limit of 8A These motors can range from 9V to 24V, and the channels support both potentiometer and encoder feedback. With these types of feedback, both position and velocity can be used to control the motors. The software that operates the MCBMini uses a PID control system that refreshes at 200Hz. This system has a number of ways that it ensures the accuracy and safety of the motor in use. It first

averages all analog data over five measurements before it is sent to the feedback system. Next, it prevents motor oscillations about the target position by utilizing a target dead-band with hysteresis. A maximum step can also be set to prevent the motor from moving too far to reach its target position.

Also, if for whatever reason, there is a timeout error longer than 1 second (for example, if the smartphone crashes unexpectedly), the channels shut down to prevent unwanted operation of the motor. However, once communication is reestablished, the motor boards rebroadcast and resume operation without the need to reset. The MCBMini system also comes with a COMM board that uses RS-485 to talk to the MCBMini stack, but was not used in the Huggable due to lack of space. However, the MCBMini boards have special solder jumpers that (once the RS-485 chips are removed) allow the stack to communicate with the IOIO board (mentioned below) directly using TTL RS-232, which simplifies the connection to a simple TX-RX line. This makes the MCBMini stack more susceptible to electrical noise, but so far, the Huggable has run normally with this configuration [27].

3.3.4 The Android IOIO and the HTC Evo 3D

The two Android powered components in the Huggable are the smartphone itself, and the board that allows it to communicate with the MCBMini stack. The smartphone is a HTC Evo 3D, pictured below in Figure 3-24.

Figure 3-24: The HTC Evo 3D for Sprint

It packs a Qualcomm MSM8660 Chipset with a Dual-Core 1.2GHz CPU [23]. At the time of purchasing, this was the best phone on the market for handling this amount of processing required for the robot. The phone is responsible for many tasks, such as motor rendering and data handling

between the MCBMini and the tablet/computer interface. The phone also provides a set of digital eyes, which are animated to showcase a wider variety of emotion that simple mechanical eyes. In order to utilize the power of the Evo 3D, the Personal Robots Group found a board that could serve as a USB host to the phone and other peripherals – the IOIO.

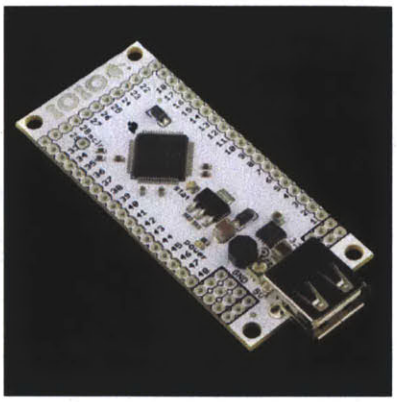


Figure 3-25: The IOIO for Android

Developed by Ytai Ben-Tsvi and Sparkfun Technologies, this board works with a Java API to be fully controllable with Android applications, which makes this very useful for utilizing our Java-based R1D1 library. It not only interprets from Android, but interacts with analog inputs and other peripheral devices. These features also make it the ideal piece of equipment to have, since it can transmit to / receive from the MCBMini stack, and it can take in all of the sensor input coming from the capacitive sensors and the pressure sensors (mentioned below) [28].

3.3.5 Capacitive Sensing

As previously mentioned, the Personal Robots Group consulted with Jetta in using similar methods that they use in many of their products to do touch sensing. The way that they did it was using a capacitive IC attached to a copper tape electrode. Unfortunately, the IC that they sent over was not in stock anymore. Thus, the plan changed to finding a more up-to-date IC, and it was found in Atmel's QTouch sensors.

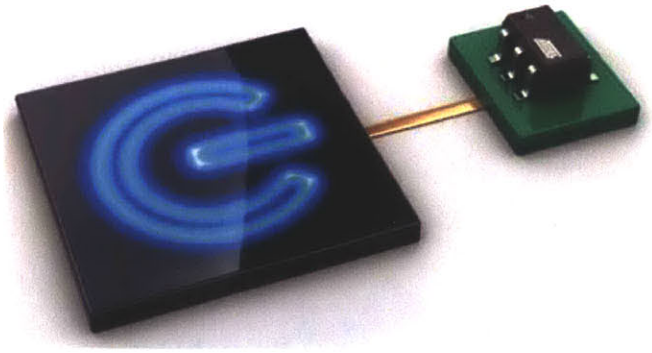


Figure 3-26: Graphic representation of the QTouch Integrated Circuit

The QTouch IC was perfect for the Huggable's touch-sensing needs. The AT42QT1011 sensor was selected for use as a simple close proximity sensor. What was important to detect is if the bear has had one of its regions touched. Therefore, all that was needed was a simple IC that would pull high (registering a max voltage signal) if the electrode was touched. It was important for the sensor to have a bit of range for detection, since it would need to detect beyond a furry surface. This particular IC only needed a few components in order to create a proximity field strong enough to penetrate the fuzzy exterior of the Huggable fur, making it an excellent candidate. The best part: its extremely small size made it ideal for individual PCB creation.

Rather than design a board populated with these little ICs (which would never fit anywhere in the robot), the better plan was to design an individual board that was small enough to fit in the tiniest of spaces close to the electrode it was governing. This also would prevent the capacitance of one IC to interfere with another. The board designed was very simple, and was organized like a potentiometer. It measures just over 0.5" long and 0.4" wide. Three wires extend from the board, with either side being power and ground for the IC, and the middle being the analog signal being read. One wire extends off of the other side of the board, and is soldered to an external piece of copper tape (the electrode). As shown in the documentation, the IC can be adjusted for sensitivity by changing the values of Cs, Cx, and Rs (two capacitors and a resistor, respectively). After some initial tests involving the covering of a sample electrode with a fur sample, the values that created just the right sensing field were 10nF, 10pF, and 10KOhms, respectively [29].

3.3.6 Pressure Sensing

It seems the same story was the case when trying to request a pressure sensing IC. The original IC sent to the Personal Robots Group was unattainable in the US, and Jetta could not send

any samples of it. Once again, the plan was changed to finding a more accessible pressure sensor, and it was found in the Honeywell collection of pressure sensing ICs.

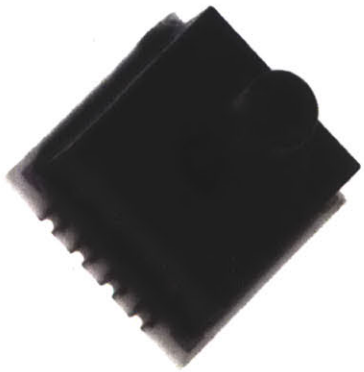


Figure 3-27: The Honeywell SX SMT series pressure sensor

This particular sensor is a Honeywell SX SMT series ceramic pressure sensor, with SMD pads for PCB attachment [30]. The particular model purchased is rated for 0 to 100 PSI sensing, which is a great range for the purposes of squeeze bladders in the Huggable. The IC is a Wheatstone bridge, which is a variable resistor circuit designed to detect small changes in resistance caused by things like strain, stress, or in this case, pressure due to air influx. However, the output signal is generally very small (on the order of mV), so an amplification circuit was necessary to get significant data. This called for an instrumentation amplifier, and one with an easily adjustable resistance. An instrumentation amplifier is a type of differential amplifiers that amplifies the difference of two voltage signals while buffering each individual signal to prevent noise and DC offset. Amplifiers like these often come in an IC package that may or may not have terminals to adjust the output gain. The desired IC was one that had the adjustable option, just in case that two pressure sensors needed different amplifications. The IC that was used was an Analog Devices AD8223 Single-Supply Instrumentation Amplifier with 5 to 1000 as adjustable gain [31]. The gain that this pressure sensor would need (about 300) could be achieved by using a 274 Ohm resistor, according to the list provided in the datasheet. This simple pressure sensor combined with the AD8223 fit in a very small area (~0.4225in^2), which made it perfectly fit in a nook in the forearm casing. The hose from the pressure bladder looped around the elbow joint and tucked into the nook, where it connected to the pressure sensor. The wires from the whole PCB (also set up like a potentiometer), ran up the arm and into the head.

3.3.7 Speaker Amp

In the previous iteration of the Huggable, there was no way to generate external sound from any of the components already in the robot unless an external speaker was used. Now, with the Android smartphone in use, the external speaker was seemingly obsolete. However, the external speaker used generated an unparalleled level of volume and quality of sound for its size, even compared to the phone's external speaker. The speaker was an iHome IH77 portable speaker, rated at 2W and 4Ohms. This particular model is no longer in production, but the Personal Robots Group was fortunate enough to have a spare model for breakdown and analysis. The speaker itself was nothing special, but the audio amplifier seemed to work exceptionally well for the size and operating voltage. Thanks to the expertise of fellow grad student Brian Mayton, the audio amplifier was isolated an identified as an Anpec SA4871 [32].

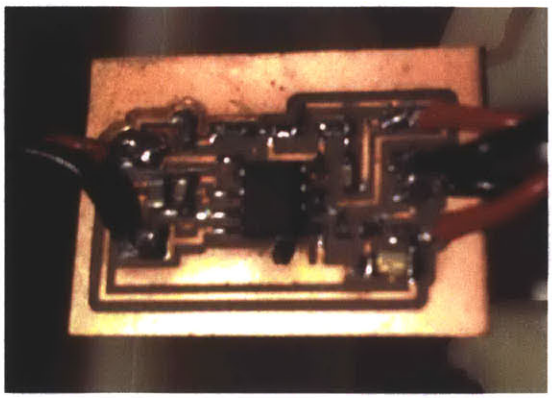


Figure 3-28: The speaker amplification board

The datasheet provided a simple circuit to easily connect a speaker, and a LED was added to indicate that the circuit was receiving current. This became useful as an indicator light for the Huggable when it was activated. The LED is strong enough to shine through the ABS shell, and gives off a warm, red glow when active. The output is routed through the core of the Huggable, going up the back, and exiting through the side of the face, where it plugs into the headphone jack of the smartphone.

3.4 Software System

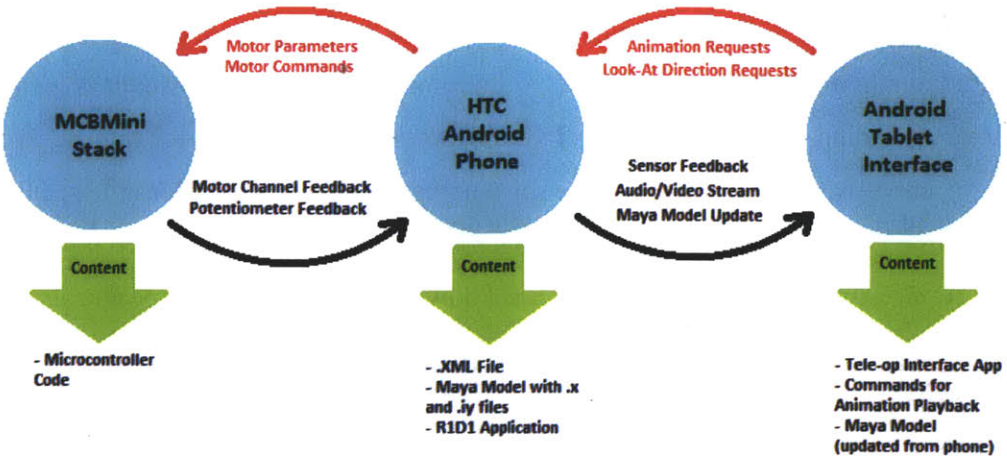


Figure 3-29: Software diagram for the Huggable platform

The software used in the Huggable stems from the architecture currently used in the Dragonbot platform. Save for a few modifications, the two systems are identical in function. The next two sections will be brief summaries on how the software works. For a more in-depth explanation, please refer to Adam Setapen's work on the Dragonbot platform [26].

3.4.1 Summary of the System Pipeline

The software architecture of the Huggable, like many of the robots in the Personal Robots Group, began with a virtual avatar crafted in Maya by Fardad. In this particular instance, the solid model from Solidworks was imported into Maya, along with joint indicators (flat cylindrical solids that indicated axes of rotation) for Fardad to manipulate. Once this model was fitted with moveable joints, it was sent to Jesse Gray, cofounder of ifRobots and an alumnus of the Personal Robots group who has programmed many of the features utilized in the R1D1 codebase. Jesse took the model and added attributes to the degrees of freedom that allowed them to interface with R1D1. This basically involves adding a motor channel within the code of the Maya model (the .mb file) that is accessible to the codebase.

Once this was complete, the .mb file was sent through xExporter (another creation of Jesse's that links the newly rigged model to R1D1) to produce the Huggable's .x file and .iy file. The .x file contains information about joint positions, rotations, and relationships with other joints, along with

mesh information (the model's "skins") and how they link with joints. The .iy file contains information about the DOFs' axes of rotation, limits, and the nature of each linked joint. The .x file needed to be tested in the R1D1 environment to make sure it was not broken; once that was verified, Fardad was able to go ahead and create new animations with the Huggable model in Maya. This process was repeated with the new animations to make sure they could operate within R1D1. The new piece to this pipeline was then exporting these files to the `HuggableAndroidController` so that both R1D1 and the model could run on the smartphone.

The last piece to this pipeline helps the MCBMini system to talk to the Huggable model. Sigurdor included a .xml file to store on the phone that lists a collection of parameters unique to each channel of each motor board in the system. (The code can be found in the Appendix section.) The first set of parameters relates to the PID control of the motors (gains, deadband, etc.). The next set sets up modes for the channel, such as feedback modes and motor polarities. After this, the motor channel is given name and description parameters that point to the joints in the .x file with the same name, thus connecting the MCBMini channels with the appropriate joints. There are also parameters that specify limits on the joints, and a conversion factor for these limits that translates them into the limits set in the .iy file. The last two parameters are safeguards for the channels so as to not damage the hardware on the physical board.

The .xml file (shown in Appendix D) was designed to govern the MCBMini boards independently from the `HuggableAndroidController`, so that the PID gains and joint limits could be initially set. This was actually the first task that was completed in operating the Huggable. Using a GUI that allows for direct access to the motor channels' parameters, the Huggable was tested in each degree of freedom to ensure that the gains stabilized the movements, and the limits prevented unwanted movement. Once these parameters were set, they were written into the .xml file, which then governed the motors' movements as the `HuggableAndroidController` was implemented. These values were later adjusted to complement the animations that were created.

3.4.2 The Teleoperator Interface

Controlling the Huggable is very similar to controlling the Dragonbot platform. A special teleoperator interface was developed by Sooyeon Jeong to control the Huggable's movements and receive data from it, such as the video stream, sensor data, and joint information. Figure 3-28 depicts this interface with all of its features.

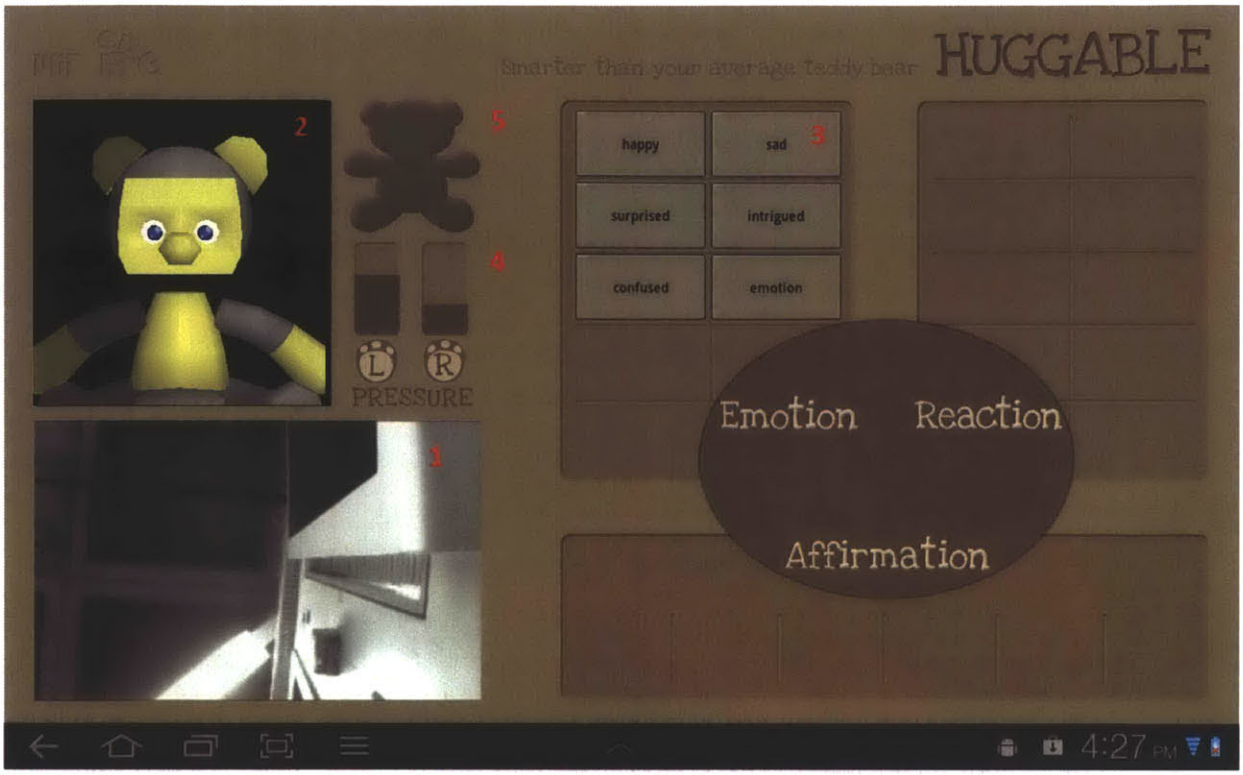


Figure 3-30: The Huggable Teleoperator Interface

1. Video/Audio Stream

The audio stream is a direct stream from the smartphone's microphone that is sent over IRCP and played through the tablet's speakers. The video is streamed from the smartphone's front-facing camera over IRCP as well, and filtered to a gray scale image to conserve the data being transferred. This window also supports a look-at behavior that, when touched, will allow the Huggable to center its view to where the touched area is specified. This gives the user control over where the Huggable can focus its gaze.

2. Virtual Character Visualization

To give the user an idea of how the Huggable is oriented in the physical space, the interface presents a virtual visualization of the bear. This model, created from Maya and exported into R1D1, receives joint information from the smartphone via IRCP. The Android controller on the smartphone calls on the IOIO board, which then calls on each MCBMini board and asks for each joint's position data. This data is then sent to the tablet for a bit of mathematical conversion based on data contained

in the motor controller's .xml file. The converted data is then utilized by the model to represent the current state of the bear.

3. Animation Playback

Figure 3-31: Animations stills of intrigue (top) and sadness (bottom)

A list of animation buttons governs the movement of the Huggable robot. Each button is associated with a certain animation created in Maya by Fardad. When the button is pressed, a string of data is sent to the Android controller to trigger the desired animation. This animation will also be represented on the virtual model visualization through joint feedback.

4. Pressure Sensor Meters

Two visual meters are displayed on the tablet to indicate the amount of pressure applied to each "paw" of the Huggable. Instead of using a raw number, the meter allows the operator to gain a sense of intensity, which visually registers better than a raw number. The hope is that this intensity reading can be related to a question asked of the child interacting with the Huggable, and will provide a visual response for the operator. The data sent to the tablet is actually a high value at its normal state, which is then brought down when pressure is applied. This data stream is reversed to visually show the meter increasing with increased pressure.

5. Touch Sensor Map

Another visual to give insight to the operator is the touch sensor map. This map includes all of the capacitive touch areas on the body. Rather than complicate the Maya model with changing

color meshes, this map became separate from the visualization as a simple image of the bear with designated zones. Once a touch zone has been touched, that area will light up on the map, and remain lit as long as that area is being touched. This information allows the operator to infer as to how the child is interacting with the bear. For example, if the two side zones are alit, it can be inferred that the child is picking up the bear. Or, if the ears light up, the child could be tickling the bear's ears. With this information, the operator can trigger a correct emotional response, and provide a more natural interaction.

3.4.3 Analog Input Handler

The main difference between Dragonbot and Huggable is that the current prototype of Dragonbot does not leverage the analog input channels of the IOIO. Since the Huggable has 14 different analog signals that need to be read, it was necessary to include handlers in the code to receive this data and interpret it. The IOIO library features specific interfaces to deal with analog input. Once the pins being used were labeled, it was a matter of including the interfaces within the instance of the IOIO class in `AndroidIOIOPSerial`. Each instance of the `AnalogInput` interface corresponds to one of the analog pins on the IOIO, so each sensor had its own instance of `AnalogInput` associated with it, labeled like so:

```
AnalogInput pressLeft = ioio.openAnalogInput(38);
```

To read the value, a simple `AnalogInput.read()` command is used that returns a value between 0 and 1 [33]. The exact voltage signal is not necessary, because these signals are being represented graphically. Once the values are collected, they are sent over IRCP, and the teleoperator interface receives them. Since the motor controller and the analog input handler are naturally in concurrency (that is, they both struggle against each other to utilize the IOIO class in `AndroidIOIOPSerial`), a "lock" was created to alleviate a potential concurrent error. This "lock" (called `ioio_lock` in the code) allows for only one of the controllers to have access to the IOIO class at a time, thus acting as a software relay. As of now, this system is still being developed, but is showing promising signs of operation.

Chapter 4
Evaluation

4.1 Technical Evaluation

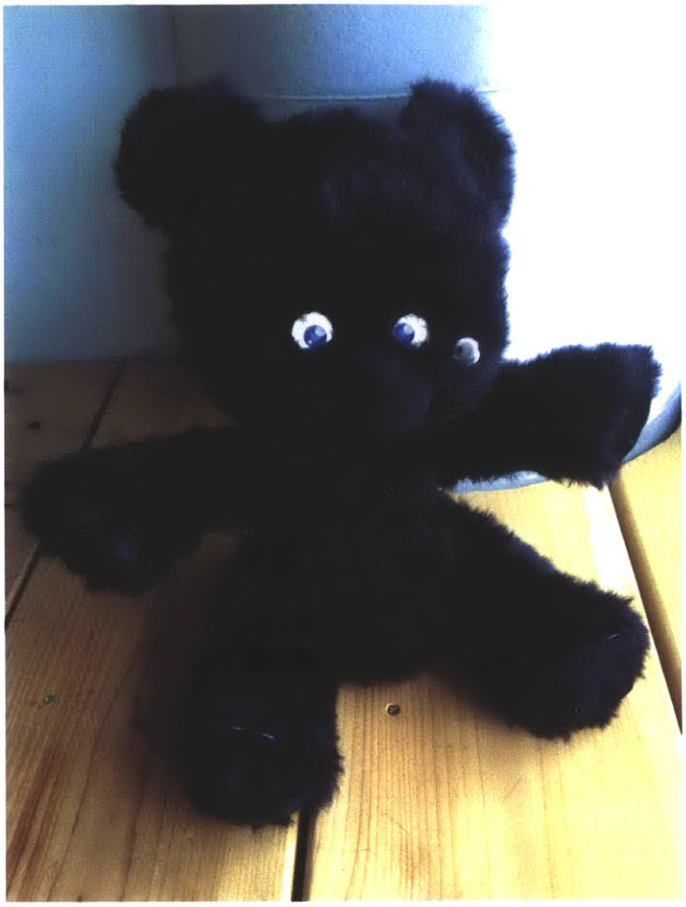


Figure 4-1: The Huggable, fully furred and with eyes operational

In the performance of animations, the Huggable was not exactly perfect in emulating its virtual self. Fardad's intended animations had to be slowed down to 8 times the normal speed so that the motors could actually perform the intended movements. Also, the backlash in many of the joints (e.g. the shoulders' up and down movement, the head tilt) caused some of the movements looked limp, as if the motor was struggling to keep the joints in place. This prevented the Huggable to replicate the smoothness and flow of Fardad's animations. Another interesting case is that the fur considerably limited the range of motion of certain DOFs, such as the arms and the elbows. It seemed that the fur was a bit too thick, and got in the way of the arms whilst they moved around in its own space. However, most of the other joints were able to move with enough speed to match the intended animation, and the digital eyes were able to portray the suggested emotion that went along with the animation.

Latency in the user interface was not able to be tested properly, because it was discovered that the phone had issues with hosting all of the animations. In order to run animations for testing, each one had to be loaded manually on the phone to ensure that the phone would not crash. Due to this memory issue, video streaming and audio streaming were also not tested properly.

4.2 User Survey

Before sending the Huggable to Boston Children's Hospital for initial research and evaluation, a "pre-study" was conducted to first evaluate some simple behavior animations. Twenty MIT students and alumni, ranging in the ages of 19 to 24, participated in this user evaluation. The academic backgrounds were mixed and the results were collected anonymously to eliminate the proctor's knowledge of gender, age, race, or background of those participating. The evaluation was done through an online survey, using small video clips of the Huggable for rating. Only the gratitude of the researcher was offered as compensation for the participants.

4.2.1 Survey Design

The online survey was very simply structured using a Google form. Each participant would "turn the page" to find an unlisted Youtube link that showcased the Huggable conveying an emotion. The video clip was only labeled by a number, and the user had to guess the emotion being conveyed. The user wrote down their guess and rated how confident he or she was in the response on a 7-point Likert scale. Then, after "turning the page", the emotion was revealed, and the user rated how correct their answer was. Both this question and the questions that followed were rated on 7-point Likert scales as well. The endpoints on the following Likert scales were levels of agreement, with 1 being strong disagreement and 7 being strong agreement. The user was first asked if he or she thought the Huggable conveyed this emotion with ease (i.e. did the robot look like it was struggling to move or did It look unhindered), and then was asked if the animation looked natural and not mechanical. Then, the user was asked, in three separate questions for clarity, if the animation was too slow, too fast, or the correct speed, respectively. The survey contained five videos of five emotions: surprised, intrigued, happy, sad, and confused, in that order. At the end, the user was asked three general opinion questions about the robot, asking if the robot was ready for research and ready to interact with children (once again rated on a 7-point Likert scale of agreement). From the observations in the technical evaluation, it was hypothesized that the users would not be able to guess

the emotions very well (or not be confident in their answer), and rate the ease/nature/speed to be poor. It was also hypothesized that the user would think that this robot is not research-ready at all, even with the consideration that this is still a prototype.

4.3 Results

Table 4-1: Average ratings for online user survey

Video #1 - Surprised	Average Rating	Standard Dev.
How confident is your answer?	4.4	1.1
How close was your answer?	4.8	2.7
The robot conveyed this emotion with ease.	4.1	1.7
The emotion conveyed seemed natural and not mechanical.	4.0	1.7
The animation was too slow.	4.6	1.8
The animation was too fast.	2.2	1.1
The animation was the correct speed.	3.9	1.9
Video #2 - Intrigued	Average Rating	Standard Dev.
How confident is your answer?	4.0	1.6
How close was your answer?	4.4	2.1
The robot conveyed this emotion with ease.	2.5	1.4
The emotion conveyed seemed natural and not mechanical.	3.5	1.7
The animation was too slow.	3.4	1.9
The animation was too fast.	2.0	1.1
The animation was the correct speed.	4.3	2.1
Video #3 - Happy	Average Rating	Standard Dev.
How confident is your answer?	4.2	1.6
How close was your answer?	4.1	1.7
The robot conveyed this emotion with ease.	2.0	1.9
The emotion conveyed seemed natural and not mechanical.	3.2	1.6
The animation was too slow.	4.8	1.8
The animation was too fast.	2.0	1.1
The animation was the correct speed.	3.2	1.5
Video #4 - Sad	Average Rating	Standard Dev.
How confident is your answer?	5.0	1.9

Table 4-1(Cont.): Average ratings for online user survey

How close was your answer?	6.1	1.4
The robot conveyed this emotion with ease.	5.1	1.7
The emotion conveyed seemed natural and not mechanical.	4.6	1.8
The animation was too slow.	3.0	Standard Dev.
The animation was too fast.	1.8	1.1
The animation was the correct speed.	5.0	2.7
Video #5 - Confused	Average Rating	1.7
How confident is your answer?	4.4	1.7
How close was your answer?	4.5	1.8
The robot conveyed this emotion with ease.	4.5	1.1
The emotion conveyed seemed natural and not mechanical.	5.0	1.9
The animation was too slow.	3.1	Standard Dev.
The animation was too fast.	2.3	1.6
The animation was the correct speed.	4.7	2.1
General Statements	Average Rating	1.4
The robot looks "huggable" and nonthreatening.	5.2	1.7
The robot looks robust enough for research in the field.	5.5	1.9
I would be comfortable allowing a child to interact with this robot.	5.7	1.1

The Likert data were analyzed by averaging across the participants and plotting the relevant averages against each other, with standard deviations shown as calculated in Excel. The differences were analyzed using a t-test, and alpha was set to 0.05 as the significance tolerance. The t-test uses the value determined by ratio of the difference of averages to the standard error of deviation (1), as shown below [34].

$$t := \frac{X_1 - X_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \qquad (1)$$

Here, X represents the mean of the group, σ represents the standard deviation of the group, and n represents the number of participants for the group of data. This value is compared to an alpha value based on the degrees of freedom (here, calculated as n-1) found in a student t-chart, and was labeled as a significant difference if the corresponding alpha value was less than 0.05 [35].

The first comparison chart, shown in Figure 4-2 pitted confidence against accuracy in guessing the emotion that the robot tried to convey. In this case, confidence refers to the question, "How confident is your answer?" and accuracy refers to, "How close was your answer?" People were more confident and more accurate in determining sadness than any of the other emotions ($p<0.05$). The only insignificant difference in this case was the confidence between "sad" and "surprised" ($p>0.05$). The differences between confidence and accuracy were insignificant except in the case of sadness ($p<0.05$). It is interesting to note that, although insignificant ($p>0.05$), it appeared that people were more confident than accurate in determining "happy".

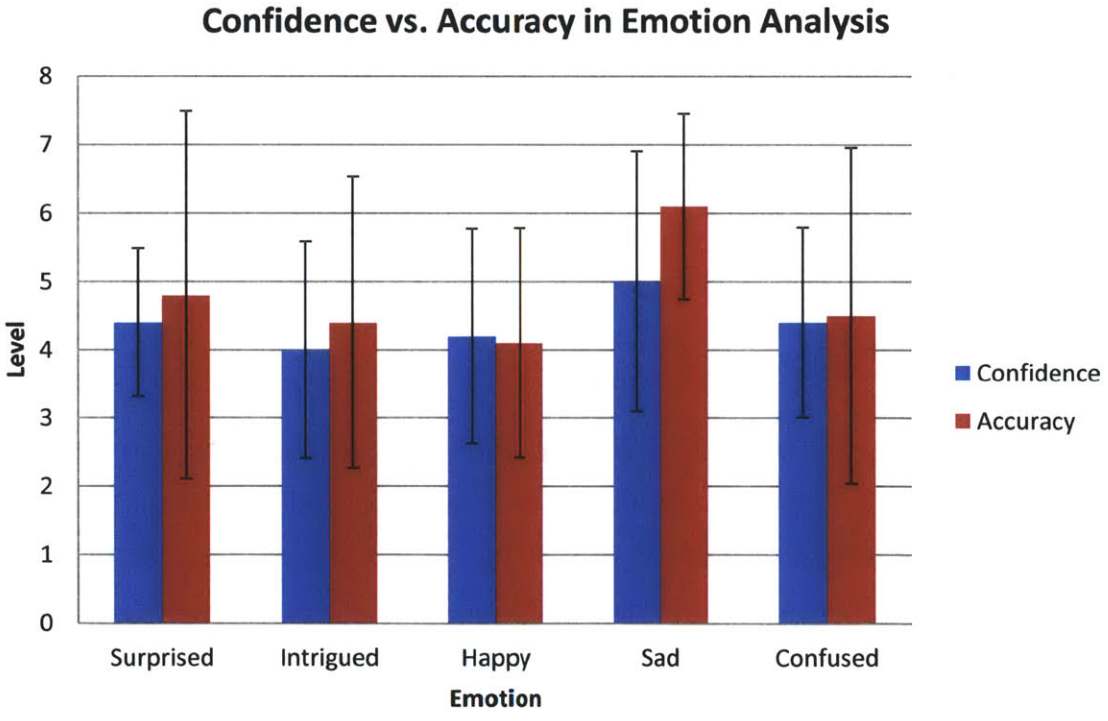


Figure 4-2: Comparison of confidence versus accuracy by rating

The second comparison chart, shown below in Figure 4-3, compared ease versus naturalness of movement for the animations. Ease refers to the statement "The robot conveyed this emotion with ease." and naturalness refers to "The emotion conveyed seemed natural and not mechanical." There

was not a very strong agreement in either category for the Huggable's movements. Both "intrigued" and "happy" were rated as the most struggled of animations ($p<0.05$), while "sad" has the least amount of struggle compared to all the others ($p<0.05$), except "confused" ($p>0.05$). However, the naturalness of "intrigued" and "happy" were rated higher than their ease ($p<0.05$). No other differences between ease and naturalness were significant ($p>0.05$).

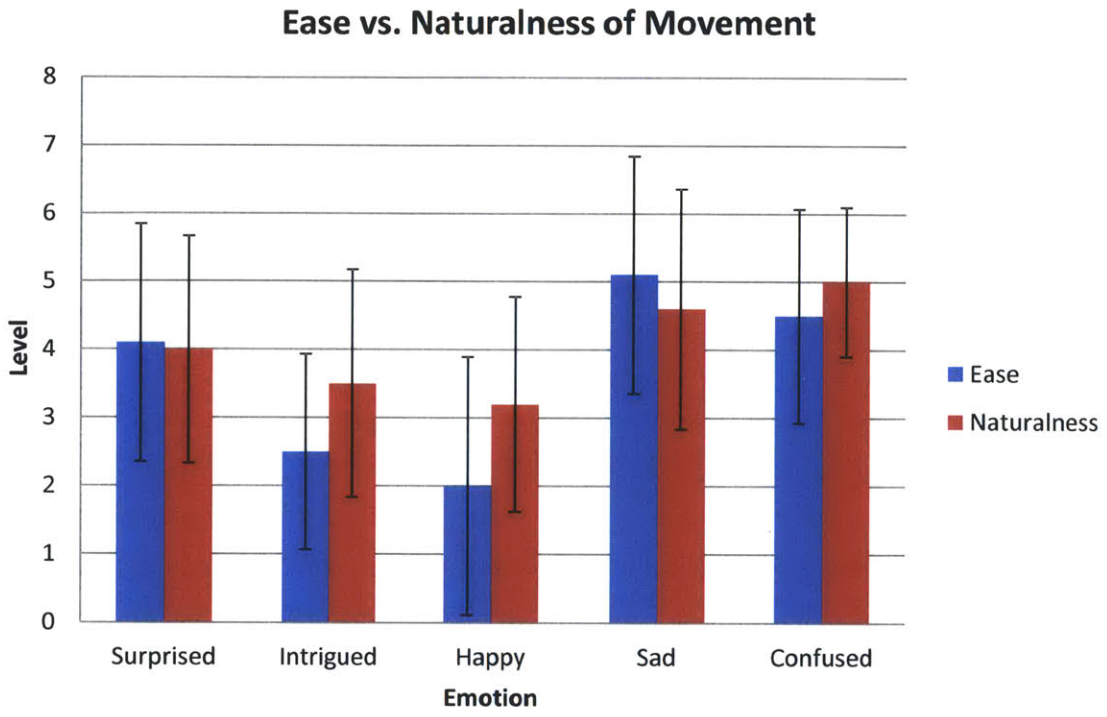


Figure 4-3: Comparison of ease versus naturalness of movement by rating

The third comparison chart, shown below in Figure 4-4, looked a side-by-side comparison of the three speed statements. Rather than make this one rating of speed, the researcher was curious as to how the users would respond with these being separate statements on which to agree or disagree. One thing is certain; across the board, none of these animations were too fast ($p<0.05$), except in the case of "confused" in which there was an insignificant difference between too fast and too slow. The "surprised" and "intrigued" animation had insignificant differences between too slow and correct speed ($p>0.05$), "happy" was rated as too slow versus correct ($p<0.05$), and both "sad" and "confused" were rated as the correct speed above the other choices ($p<0.05$).

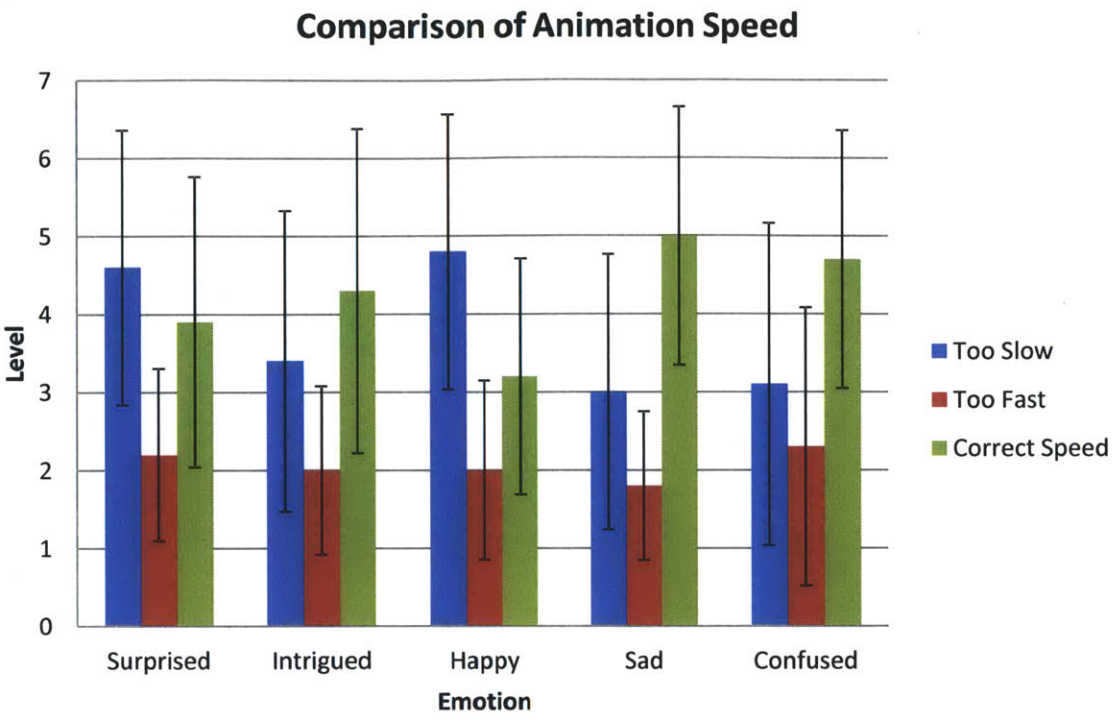


Figure 4-4: Comparison of animation speed by rating

The last chart, shown below in Figure 4-5, compared the general statements against each other. As shown, the statements are not significantly ranked higher or lower than one another ($p>0.05$). However, all three statements are significantly weak in their agreement, but in agreement nonetheless ($p<0.05$). The results of the preliminary study were surprising, but a bigger sample size with a better constructed prototype should give a better idea of its abilities, along with the capacity for richer research to be done.

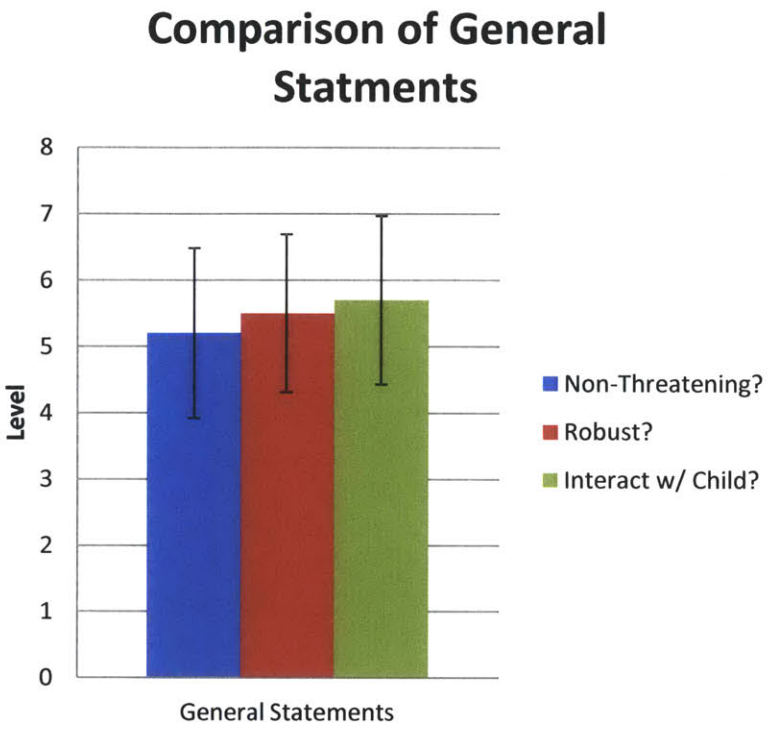


Figure 4-5: Comparison of general statements by rating

Chapter 5

Future Work and Conclusion

5.1 Future Work

5.1.1 Children's Hospital Boston

The Huggable Project was designed for future deployment into the hospital setting. The Personal Robots Group is working closely with the Boston Children's Hospital to put forth a research initiative, using the Huggable Project as the testing apparatus. The research is aimed to evaluate the Huggable's abilities to quantify and socially mitigate stress and anxiety in child patients, around the ages of 3 to 10. The Huggable would be placed in the Pediatric Intensive Care Unit (PICU) and the Oncology units to interact with children and help them throughout the healing process. The robot will hopefully improve the quality of experience in the hospital for children and their families by being a consistent and nonthreatening character that can provide social assistance.

The Child Life Group will use this robot in a series of sessions that can either be play therapy, simple social interaction, or preparedness sessions. This group is comprised of child specialists that use techniques like developmental interventions and play to help patients and families get through their time at the hospital, and to educate them about their situation [36]. The Child Life staff may use this robot to help instruct children in medical procedures. Meanwhile, the staff can provide feedback to further improve the Huggable. Children's Hospital and the Personal Robots group will continuously work on the Huggable to reduce the cost and provide a more autonomous system that can ultimately travel home with the child for post-care.

5.1.2 Redesign and Second Iterations

As hinted in some of the sections of this thesis, this iteration of the Huggable could use many improvements for better functionality. Future work for the Huggable will include a second iteration of the design, to fix many of the interior problems that were found. Below lists all of the necessary details that need to be fixed in the next iteration:

- General
    - Improved spacing for wires to be placed
    - Better wire/mechanical connectors for easy module removal
    - Better access to fasteners
    - Better structure design to prevent warping/misalignment
    - Hex screws instead of Philips head screws used to fasten all shells

    - Press fit slots for capacitive sensor PCBs
    - Reduce cost by using less expensive motor configurations without encoders
    - Use motors rated for lower voltage and slightly higher torque/RPM
    - New MCBMini for lower-voltage motors
    - Dial down analog voltage signals to prevent overloading IOIO pins
- The Base/Legs
    - Change electrical hardware / batteries for operation at a lower voltage
    - New design for battery pack and charging port – meet toy standards
    - Allow legs to swing down but create hard stops to prevent legs from swinging too far back or too far forward
    - Better on/off indication and power switch
    - Latched base for easy access to electrical hardware for troubleshooting
- The Arms
    - Stronger casing for pressure tube
    - Include heartbeat sensor system
    - Press fit slot for pressure sensor PCB
    - Improved bearing block for elbow joint
    - Designated path for elbow potentiometer wire to travel
    - Improved shoulder gear train to prevent spur gear backlash
    - Better potentiometer casing to include room for terminals and wires
    - Position wires to travel through body and closer to arm rotation axis for less tangle and more concealment (if possible)
    - More modularity for easy removal and troubleshooting
- The Body
    - Better clearance for motor casing fasteners
    - Press fit slot for speaker amp board
    - Actual slot to press fit onto waist potentiometer shaft
    - Remake neck bushing to include actual ball bearing
    - Wider channel for wires to pass
    - Better bearing for arm rotation
    - Better track for waist bend
- The Head

- Better stabilization of the head
- Redesign head tilt / head nod mechanisms for stability
- Redesign head structure to include new ear mechanism
- Redesign muzzle mechanism with new spring
- Perforate the head shell for air ventilation
- Better accessibility to electronic hardware of troubleshooting
- Redesign shells for easy removal
- Redesign ear mechanism for modularity
- Extend out the muzzle bar to avoid mask and fur collision

With these listed changes, the Huggable should operate much better.

5.2 Conclusion

The purpose of this thesis was to present the Huggable, a new socially assistive robot designed to interact with children in hospitals and collect sensory information that can help doctors and nurses tend to their patients better. The DFM process used to reconstruct this project was one containing many valuable lessons of time management, design considerations, product requirements, and communication. The prototype that resulted was both a dramatic step in a new direction and a huge learning experience in both mechanical and electrical hardware. The software developed was a chance to further improve and generalize the R1D1-Android pipeline for future robots of the Personal Robots Group. With this new robot taking an active role in new research with Children's Hospital Boston, the future of the Huggable Project looks promising.

References

[1] Okamura, A.; Mataric, M.J.; Christensen, H.; , "Medical and Health-care Robotics," *IEEE Robotics & Automation Magazine*, September 2010, pp. 26-37.

[2] Feil-Seifer, D.; Mataric, M.J.; , "Defining socially assistive robotics," *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on* , vol., no., pp. 465- 468, 28 June-1 July 2005

[3] Kuo, I.H.; Rabindran, J.M.; Broadbent, E.; Lee, Y.I.; Kerse, N.; Stafford, R.M.Q.; MacDonald, B.A.; , "Age and gender factors in user acceptance of healthcare robots," *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on* , vol., no., pp.214-219, Sept. 27 2009-Oct. 2 2009

[4] Feil-Seifer, D.; Mataric, M.J.; , "Toward Socially Assistive Robotics For Augmenting Interventions For Children With Autism Spectrum Disorders," Available at http://cres.usc.edu/pubdb_html/files_upload/589.pdf

[5] Nikolopoulos, C.; Kuester, D.; Sheehan, M.; Ramteke, S.; Karmarkar, A.; Thota, S.; Kearney, J.; Boirum, C.; Bojedla, S.; Lee, A.; , "Robotic agents used to help teach social skills to children with Autism: The third generation," *RO-MAN, 2011 IEEE* , vol., no., pp.253-258, July 31 2011-Aug. 3 2011

[6] Yamamoto, S.; Kimura, R.; , "Investigation of playing with entertainment robotic pet of pre-school aged child in nursery school by video observation," *SICE, 2007 Annual Conference* , vol., no., pp.654-659, 17-20 Sept. 2007

[7] Looije, R.; Neerincx, M.A.; de Lange, V.; , "Children's responses and opinion on three bots that motivate, educate and play," *Journal of Physical Agents*, vol. 2, no. 2, pp. 13-20, June 2008

[8] Saldien, J.; Goris, K.; Yilmazyildiz, S.; Verhelst, W.; Lefeber, D.; , "On the Design of the Huggable Robot Probo," *Journal of Physical Agents*, vol. 2, no. 2, pp. 3-11, June 2008

[9] Kozima, H.; Nakagawa, C.; Yasuda, Y.; , "Interactive robots for communication-care: a case-study in autism therapy," *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on* , vol., no., pp. 341- 346, 13-15 Aug. 2005

[10] Robins, B.; Dautenhahn, K.; Dickerson, P.; , "From Isolation to Communication: A Case Study Evaluation of Robot Assisted Play for Children with Autism with a Minimally Expressive Humanoid Robot," *Advances in Computer-Human Interactions, 2009. ACHI '09. Second International Conferences on* , vol., no., pp.205-211, 1-7 Feb. 2009

[11] Villano, M.; Crowell, C.; Wier, K; Tang, K.; Thomas, B; Shea, N.; Schmitt, L.; Diehl, J.; ,"DOMER: A Wizard of Oz Interface for Using Interactive Robots to Scaffold Social Skills for

Children with Autism Spectrum Disorders," *Proceedings of the 6th international conference on Human-robot interaction. HRI '11.* pp.279-280, 2011.

[12] Hanson, D; Baurmann, S.; Riccio, T; Margolin, R.; Dockins, T; Tavares. M; Carpenter, K; ,"Zeno: a Cognitive Character," *The 2008 Association for the Advancement of Artificial Intelligence Workshop. AAAI'08.* pp. 9-11, 17 Jul. 2008.

[13] Stiehl, W.D.; Lieberman, J.; Breazeal, C.; Basel, L.; Lalla, L.; Wolf, M.; , "Design of a therapeutic robotic companion for relational, affective touch," *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on* , vol., no., pp. 408- 415, 13-15 Aug. 2005

[14] Jun Ki Lee; Toscano, R.L.; Stiehl, W.D.; Breazeal, C.; , "The design of a semi-autonomous robot avatar for family communication and education," *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on* , vol., no., pp.166-173, 1-3 Aug. 2008

[15] Salter, T.; Werry, I.; Michaud, F.; , "Going into the wild in child–robot interaction studies: issues in social robotic development," *Intel Serv Robotics*, pp. 93-108, 2008

[16] Dos Santos, K.; Breazeal, C.; , "Project "LilGuy": A Robotic User Interface for Controlling Virtual Characters". Available at http://web.media.mit.edu/~kdos/sait_prg_lilguy_finalreport.pdf, Feb. 2011

[17] Setapen, A.; Breazeal, C.; , "DragonBot: A Platform for Longitudinal Cloud-HRI," *Human-Robot Interaction*, Boston, MA. 2012

[18] Gould, S.; "A Biological Homage to Mickey Mouse," *Natural History.* pp. 30-36, May 1988

[19] Faulhaber 1516 DC Micromotor Datasheet. Available at http://www.micromo.com/Micromo/DCMicroMotors/1516_SR_DFF.pdf

[20] Faulhaber 1724 DC Micromotor Datasheet. Available at http://www.micromo.com/Micromo/DCMicroMotors/1724_SR_DFF.pdf

[21] Faulhaber 15A Planetary Gearhead. Available at http://www.micromo.com/Micromo/DriveLinearServices/15_A_DFF.pdf

[22] Faulhaber 16/7 Planetary Gearhead. Available at http://www.micromo.com/Micromo/DriveLinearServices/16_7_MIN.pdf

[23] "HTC Mobile Phones – Evo 3D Sprint – Specifications". HTC.com. Available at http://www.htc.com/us/smartphones/htc-evo-3d-sprint/

[24] Faulhaber 1016 DC Micromotor Datasheet. Available at http://www.micromo.com/Micromo/DCMicroMotors/1016_G_MIN.pdf

[25] Faulhaber 10/1 Planetary Gearhead. Available at http://www.micromo.com/Micromo/DriveLinearServices/10_1_MIN.pdf

[26] Setapen, A.; "Creating Robotic Characters for Long-Term Interaction". *MIT MAS Master's Thesis* (2012)

[27] Orn, S.; "MCBMini V4.2 Manual". Available at http://siggiorn.com/wp-content/uploads/mcbmini/MCBMiniV4Manual.pdf, Mar. 2011

[28] "IOIO for Android". Sparkfun.com. Available at https://www.sparkfun.com/products/10748

[29] Atmel AT42QT1011 One-channel Touch Sensor IC Datasheet. Available at http://www.atmel.com/Images/doc9542.pdf

[30] Honeywell SX SMT Series Microstructure Pressure Sensors Datasheet. Available at http://sensing.honeywell.com/index.php/ci_id/53120/la_id/1/document/1/re_id/0

[31] Analog Devices Single-Supply, Low Cost Instrumentation Amplifier Datasheet. Available at http://www.analog.com/static/imported-files/data_sheets/AD8223.pdf

[32] Anpec SA4871 3W Mono Low-Voltage Audio Power Amplifier Datasheet. Available at http://www.pstrong.com/webcenter/download/200832814265179445.pdf

[33] Ben-Tsvi, Y.; "Analog Input". Available at https://github.com/ytai/ioio/wiki/Analog-Input

[34] The T-Test. Research Methods Knowledge Base. Available at http://www.socialresearchmethods.net/kb/stat_t.php

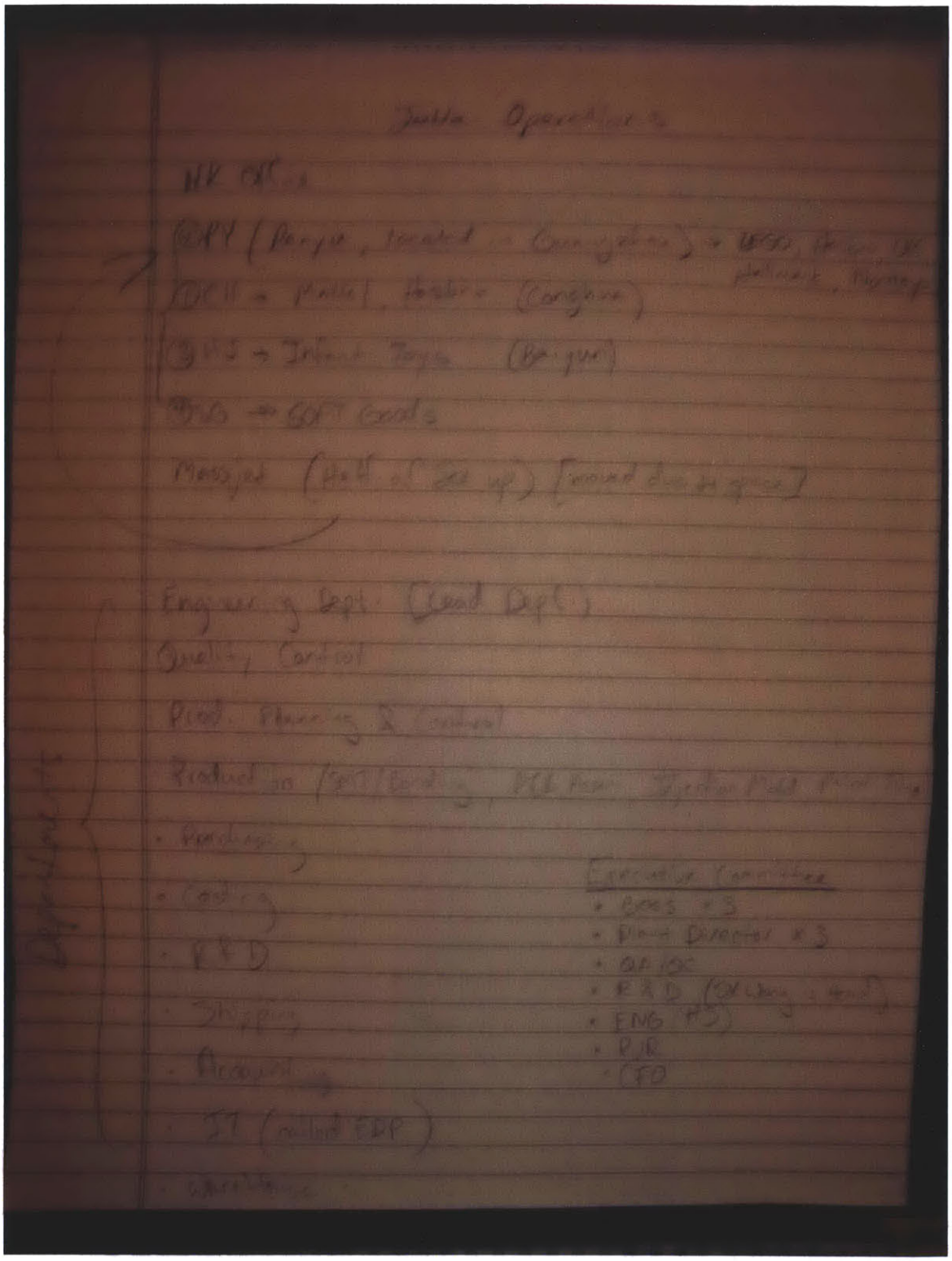
[35] T-Statistics Reference Table. Available at http://www.pindling.org/Math/Statistics/Textbook/Tables_references/student_t_upper_tail_alpha.htm

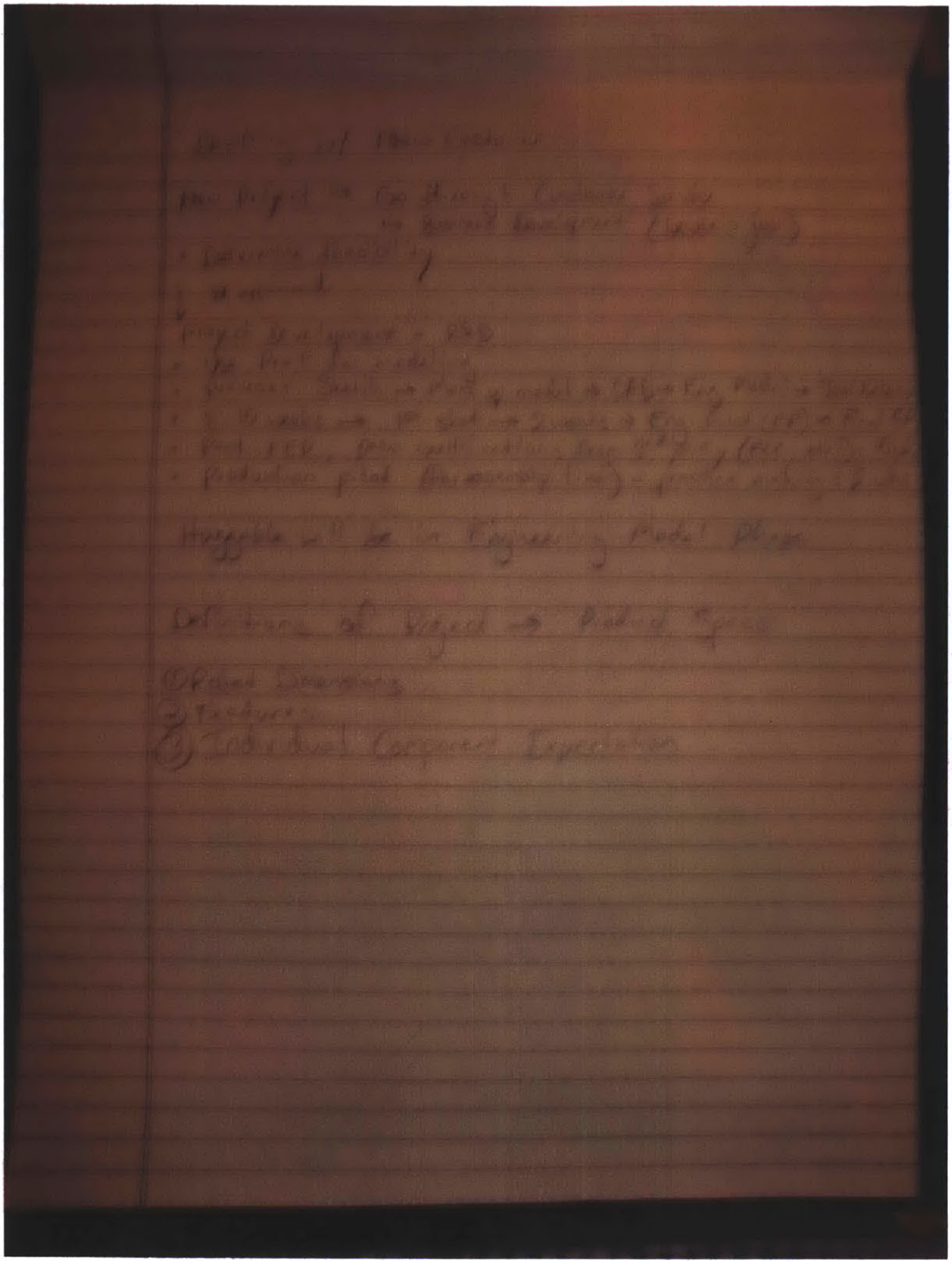
[36] "Child Life Specialists". Boston Children's Hospital Website. Available at http://childrenshospital.org/patientsfamilies/Site1393/mainpageS1393P48.html

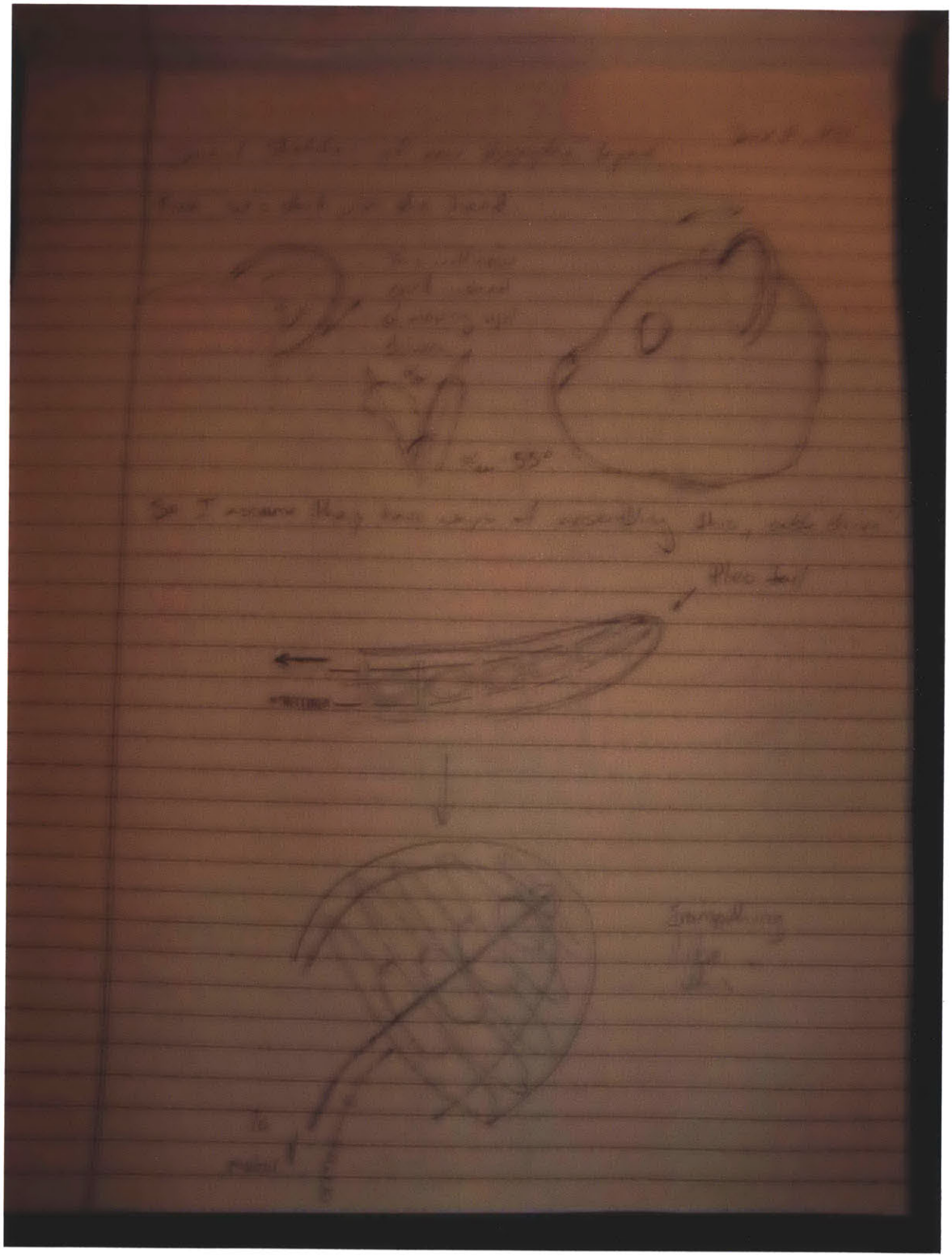
Appendix A: Huggable Project Logbooks

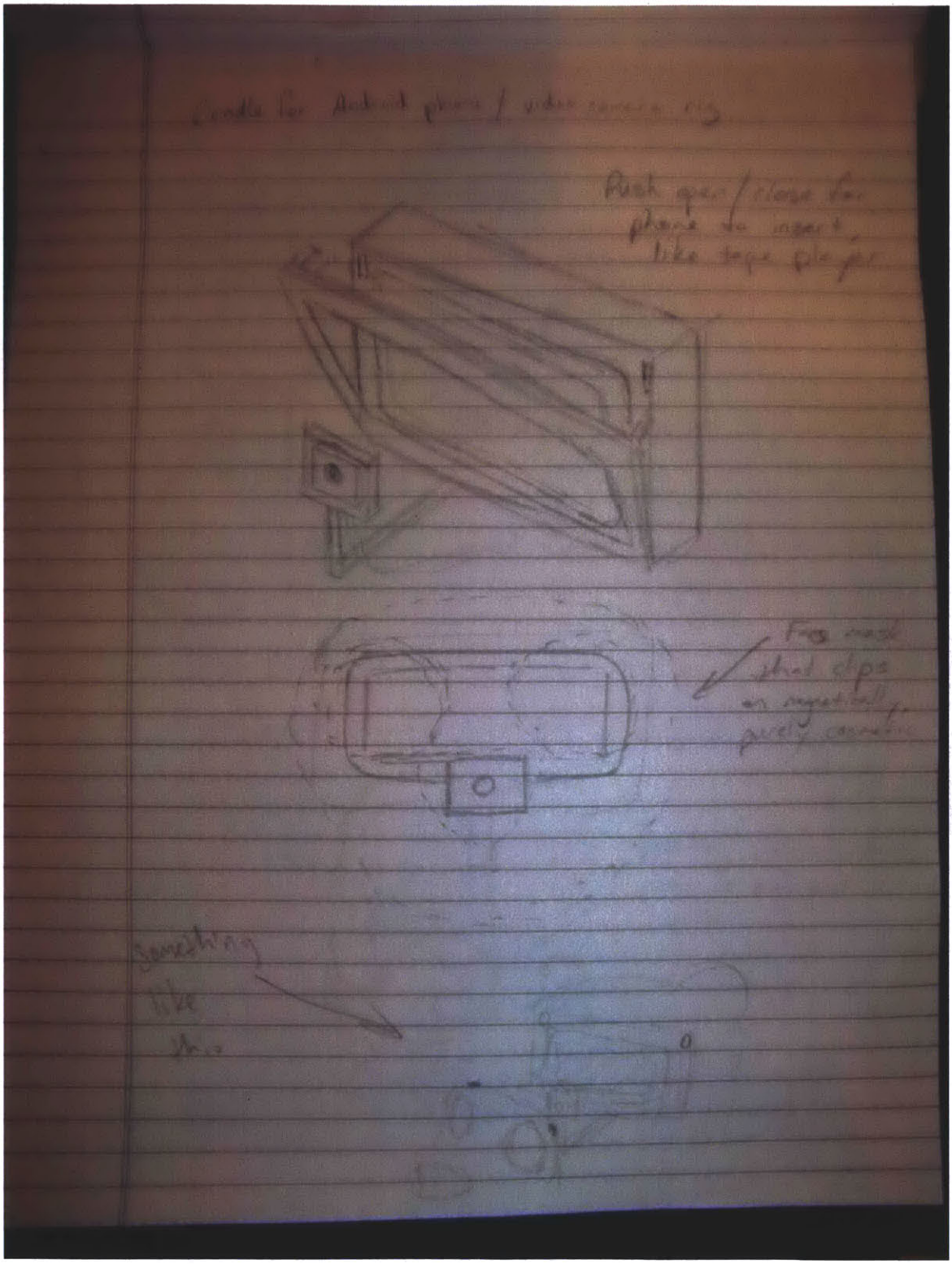
Week 1 and 2 Log Book

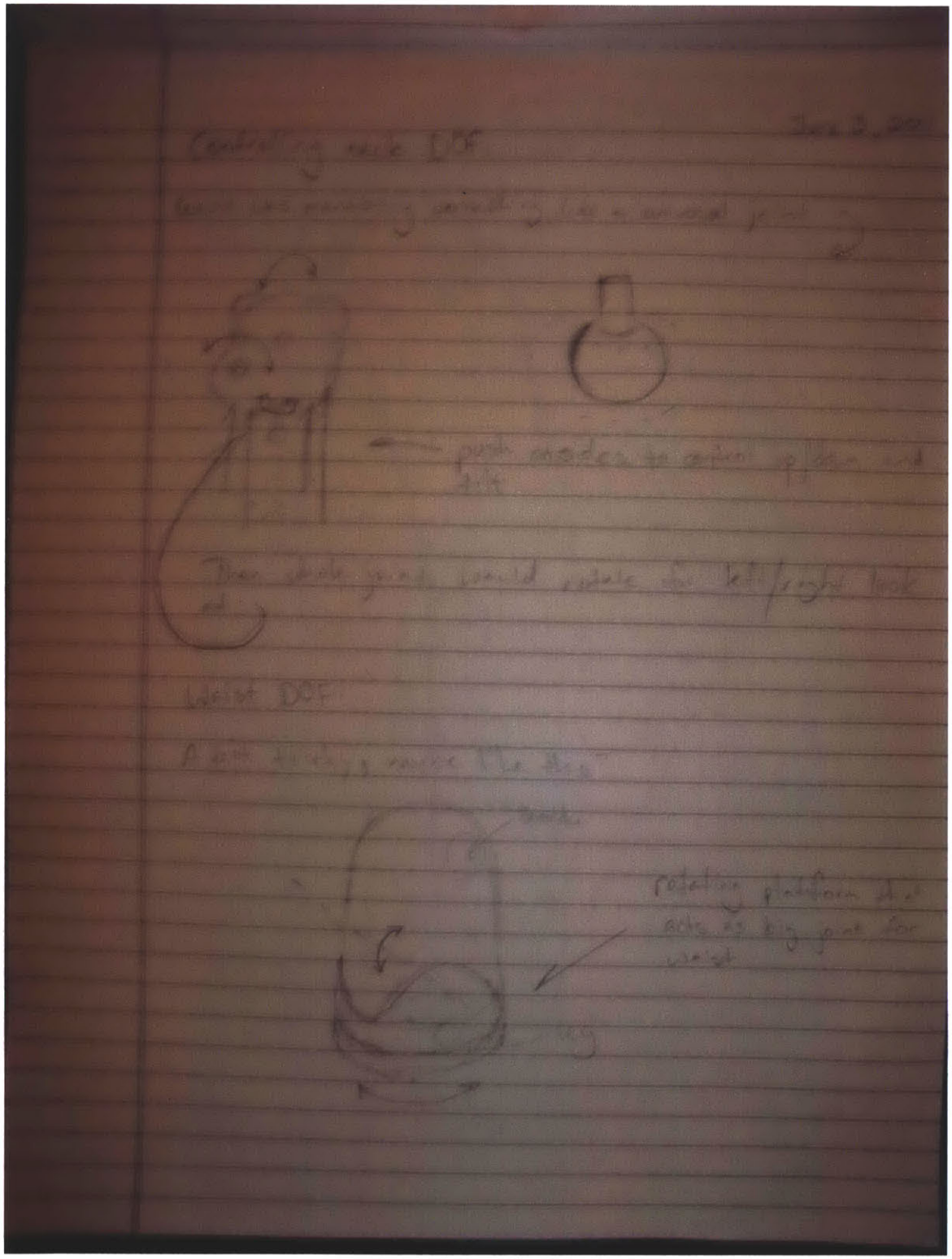
Contents of this Book:

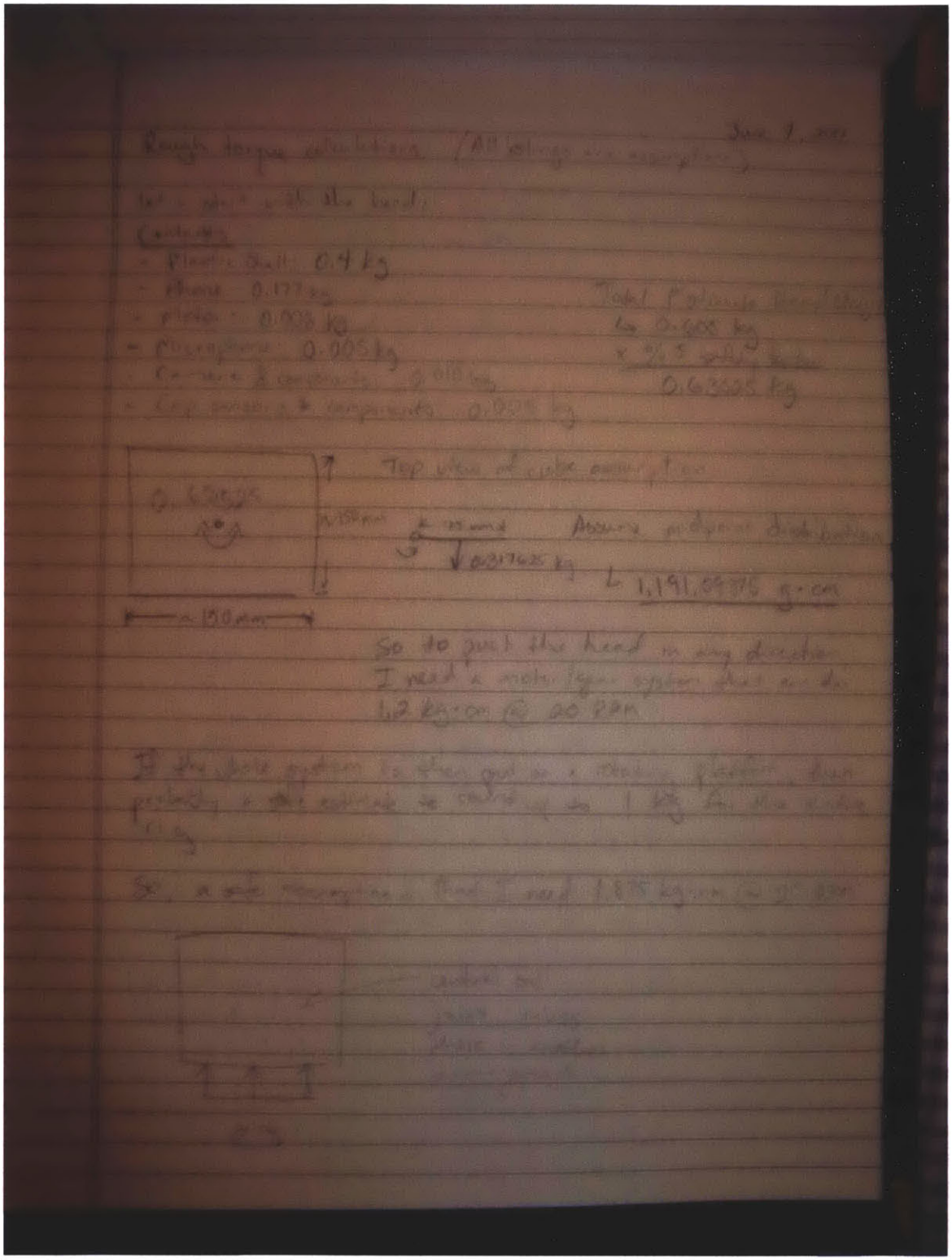


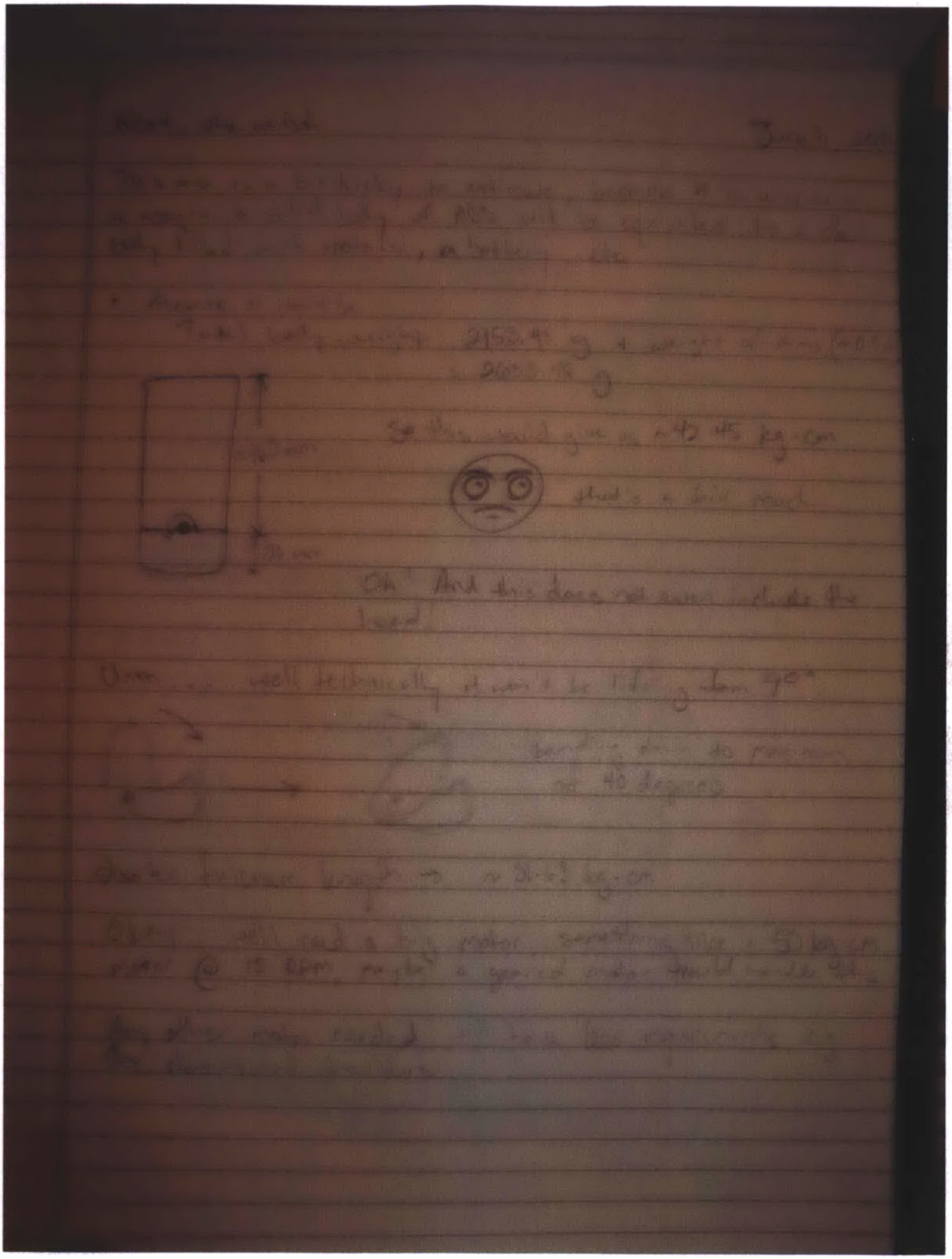


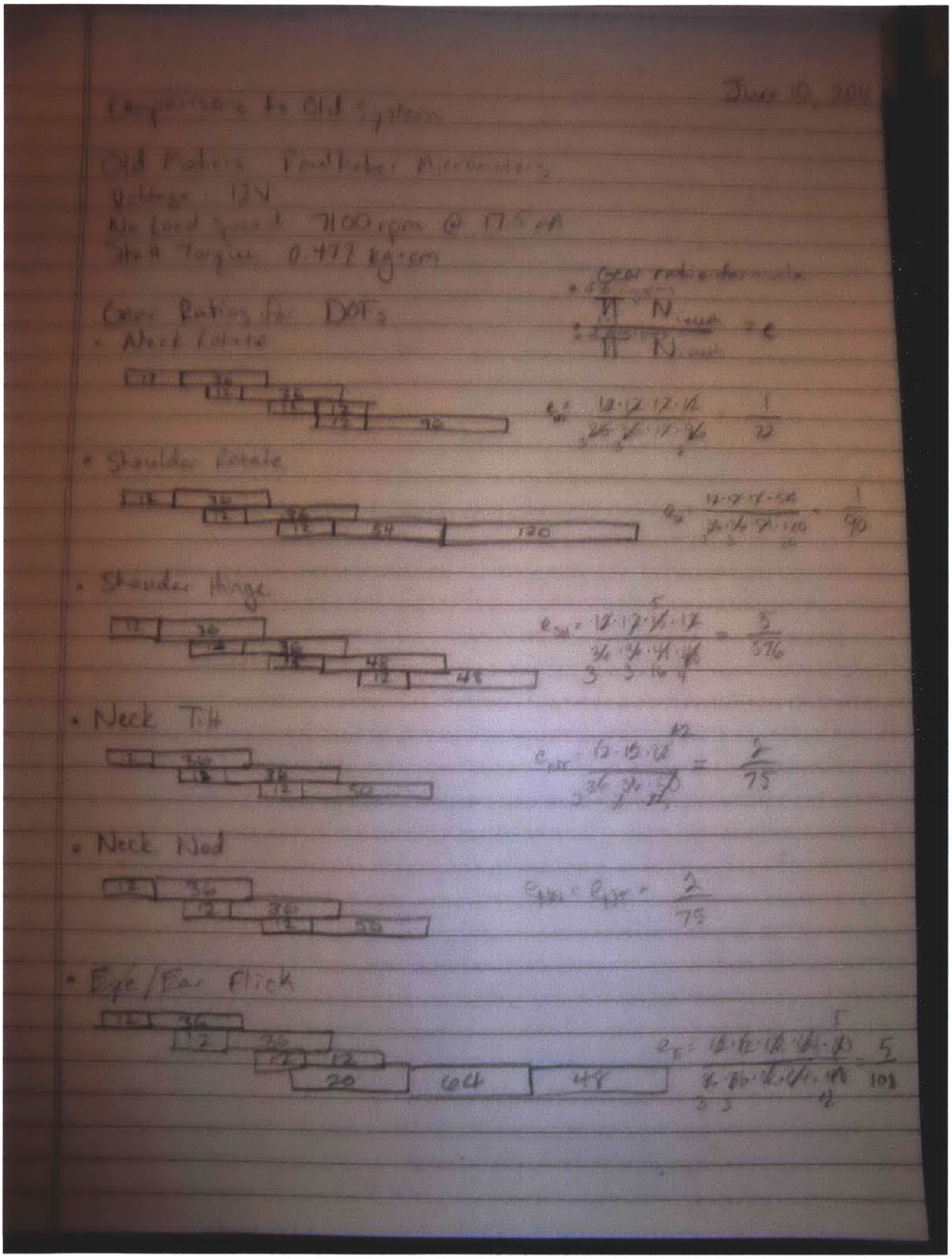



Push gun / close to phone to insert like tape player
Something like


Wrist DOF
rotation platform 2 DOF
axle as big gear for
wrist



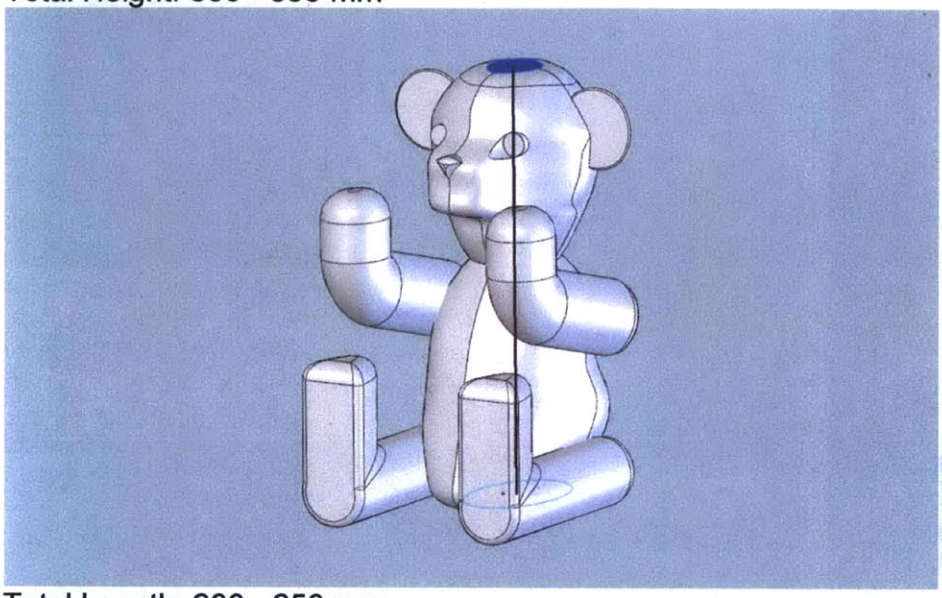



June 10, 2011
Comparison to Old System
Old Robot: Feedback Mechanisms
Voltage: 12V
No Load Speed: 7100 rpm @ 175 mA
Stall Torque: 0.472 kg·cm
Gear Ratios for DOF's
• Neck Rotate
• Shoulder Rotate
• Shoulder Hinge
• Neck Tilt
• Neck Nod
• Eye/Ear Flick

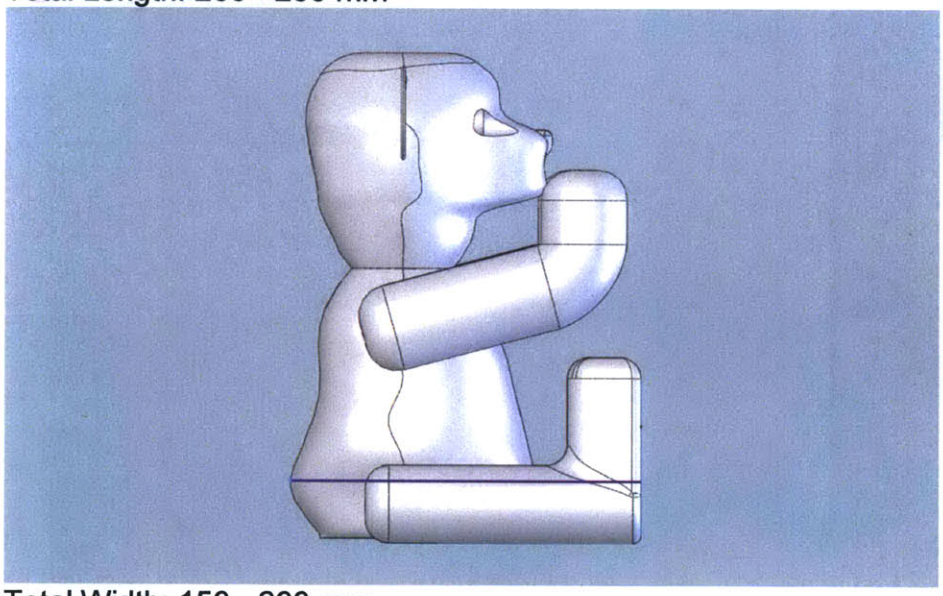
Huggable Project - Desired Specifications:

Project Dimensions:

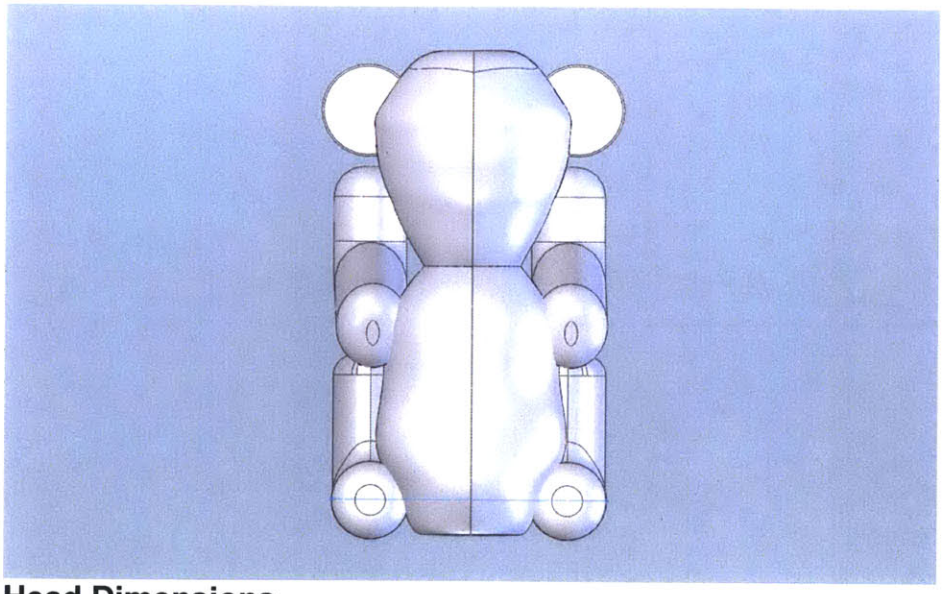
Major Dimensions -
Total Height: 300 - 350 mm



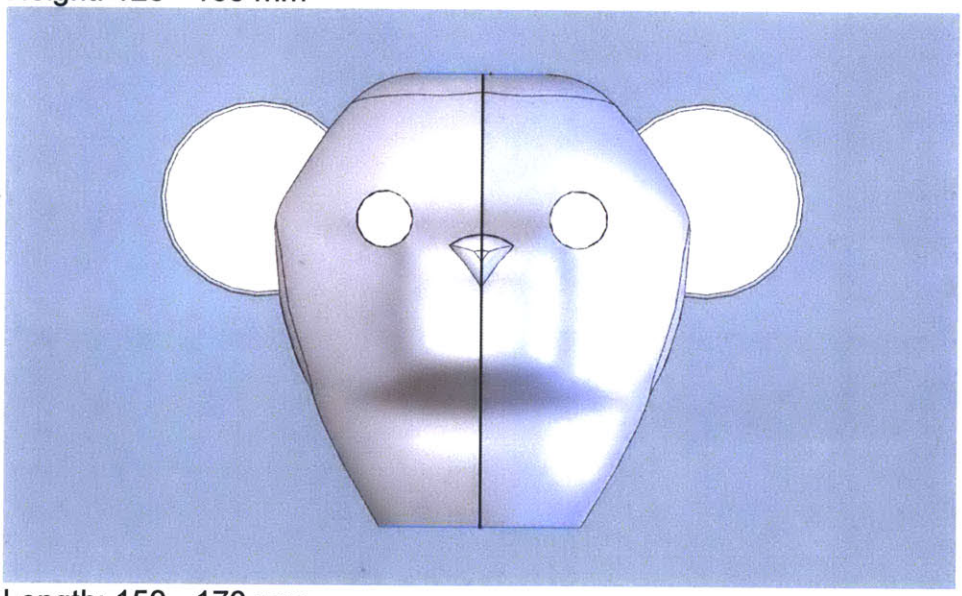
Total Length: 200 - 250 mm



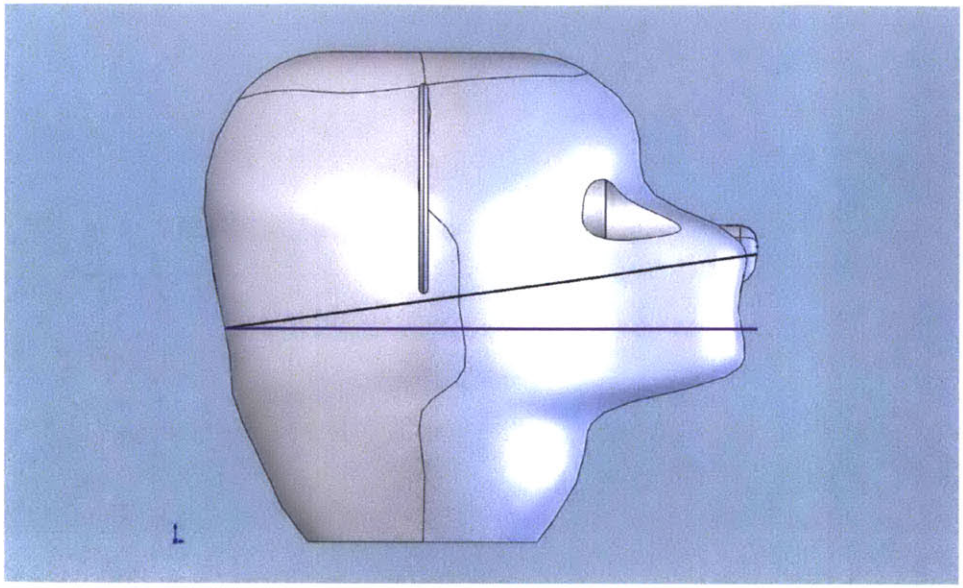
Total Width: 150 - 200 mm



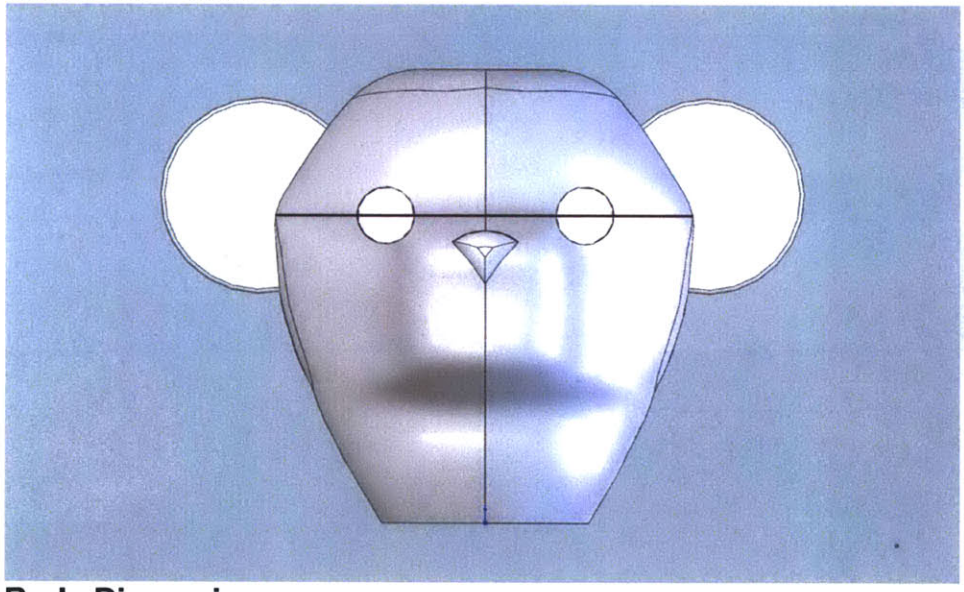
Head Dimensions -
Height: 120 - 160 mm



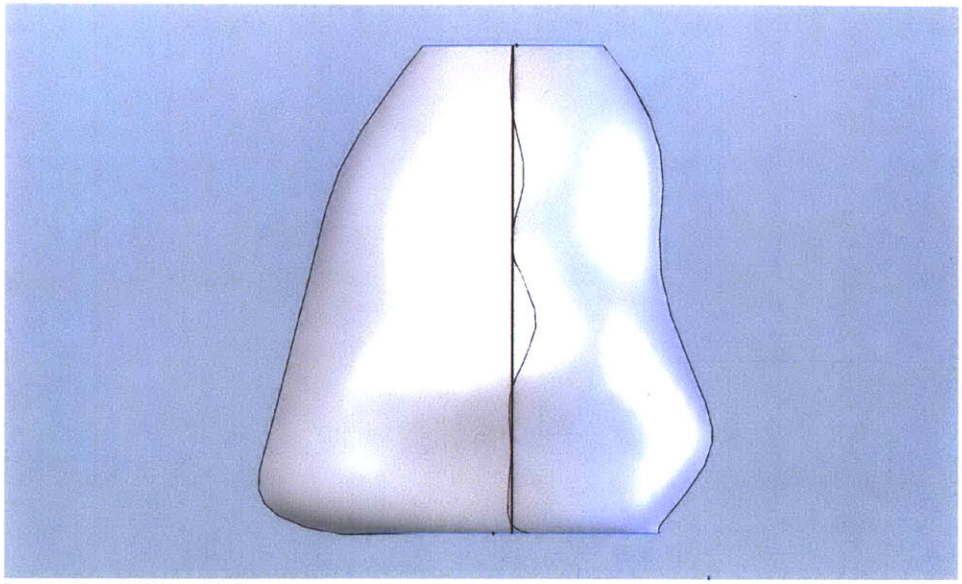
Length: 150 - 170 mm



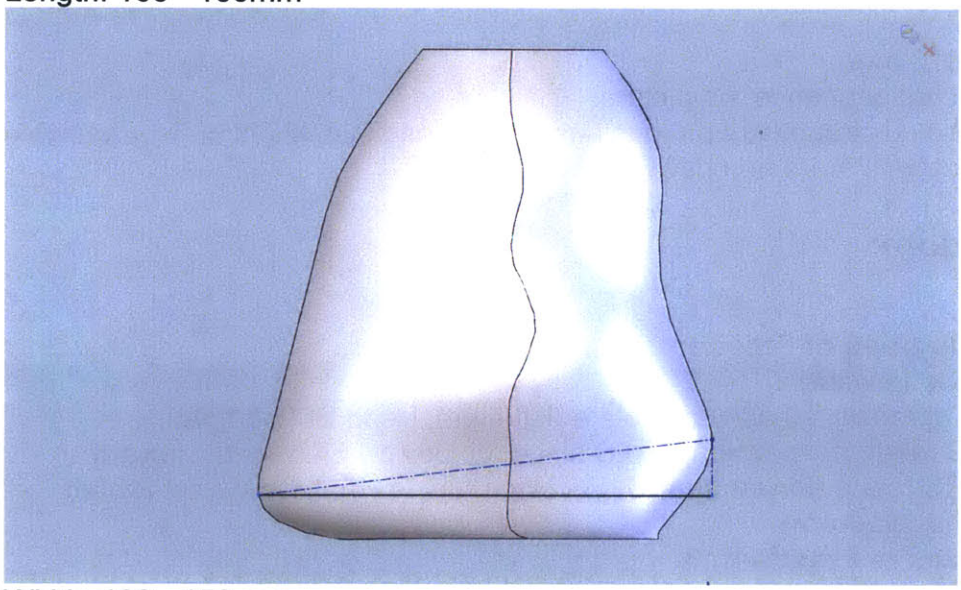
Width: 150 - 180 mm



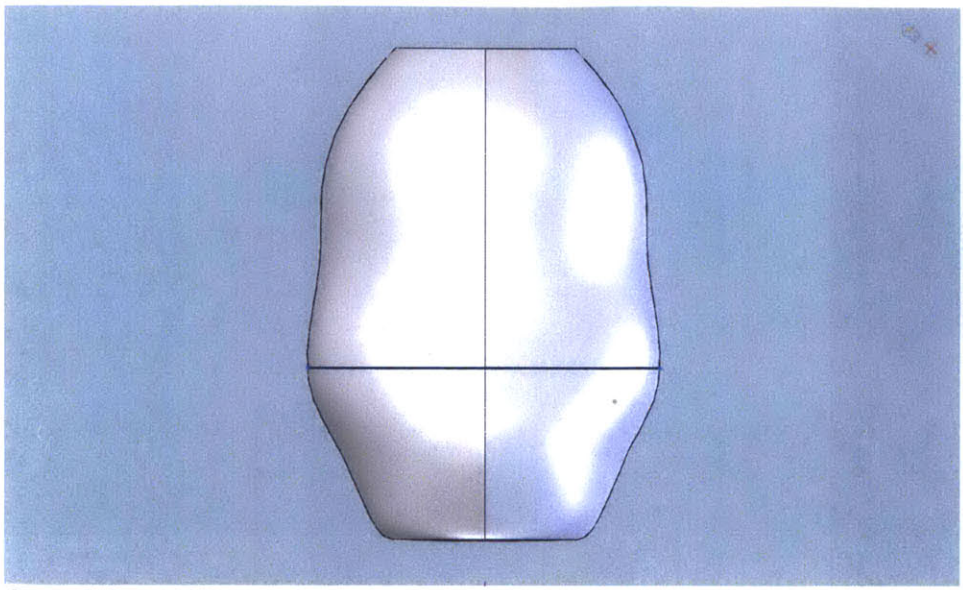
Body Dimensions -
Height: 170 - 190 mm



Length: 160 - 180mm



Width: 130 - 150 mm



Arm & Leg Dimensions -
Length: 170 - 190 mm
Circumference: 160 - 170 mm
*Note: arm shape will change, hence, no picture.
YES. I think the arm should change shape. don't need that bend in it. Might be worth having fardad provide as shape with more of a paw.

Degrees of Freedom:

Total Independent Degrees of Freedom: 11
- 1 for the ears: Ears curl (inward)
- 3 for the head: Head up/down, Head/waist rotate (left/right), Head tilt (left/right).
- 3 for left shoulder: LS rotate (forward/back), LS hinge (up/down), Elbow/wrist inward
- 3 for right shoulder: RS rotate (forward/back), RS hinge (up/down), Elbow/wrist inward
- 1 for waist: Waist bend (up/down)

Total Coupled Degrees of Freedom: 3
- 2 for wrists: Left wrist inward (coupled with left elbow), Right elbow (coupled with right elbow)
- 1 for waist: Waist rotate (left/right) (coupled with neck rotation)

Range of DOFs
*Note: Each ear is not independent; ears would move together
*Note: We would like for these motions to be quiet and back-drivable
WHERE ARE THE MIDPOINTS ON THESE?
DOF maximum speed: 10 rpm (needs to be faster for high energy expressions... we want smooth movement, most the time movement will be warm and friendly.)
DOF maximum acceleration: 3 rpm^2
- Ears curl: 20 degrees (deflection) -- probably want more... fold over more...get fardad's opinion
- Head up/down: 50 degrees (more important to look further up than down... will be looking up at kids a lot). Waist bend will help it to look down.
- Head rotate (left/right): 70 degrees (can you do more... closer to 120?)

- Head tilt (left/right): 30 degrees
- Shoulder rotate (up/down): 150 degrees
- Shoulder hinge (up/down): 60 degrees (can we do more to go above horizontal? Say 120?)
- Elbow: 60 degrees (can we go 90?)
- Wrist: 20 degrees (get fardad to weigh in on this)
- Waist rotate (left/right): 40 degrees

Features and Components:

Vision
- Wide-angle view camera in nose
- Range: 100 degrees -- can you find this?
- Resolution: 640x480
- Need wireless transmission package. Phillip will tell you how to do this.

Pressure / Capacitive Sensing
- Detect levels of contact on surface under the fur
- Pleo's capacitive sensing tech would be acceptable
- FSR or air bladder to detect pressure. Adam makes good point .Good for children in hospital to indicate how much pain they feel by squeezing its arm. Air bladder is probably way to go like in squeeze doll. Dan may be able to answer some questions.
- Capacitive sensing all around body (major zones): Top of head, Back of head, Nose/Snout, Front of Body, Back of Body, Arms, Forearms, Hands, Legs, Feet,
- Pressure sensing focused to just limbs (whole body optional).

Motor Feedback
- Analog potentiometers for joint data
- Currently using P12426CT-ND (POT 10K OHM 10MM 347 DEGREE SMD)
- We like this component, easy to use and slim profile
- Would like to place passive feedback in ankles
- Motor encoders: 78-100 counts per 1 degree resolution at motor shaft

Audio Input/Output
- Microphone to be placed in head, speaker to be placed in body
- We would like to use similar mic/speaker that Pleo and Autom uses.

Processing
- Using Android-based smartphone to handle higher-level component processing
- Will need to make mount in the head to place general smartphone (adjustable)
- Utilize screen for virtual eyelids/eyes/pupil/iris [NOT eyebrows] for expressive eyes (with clear domes for eyeballs). May want these to be clear but solid so touch screen-- may add depth to eyes.
- Magnetic muzzle faceplate to cover phone and complete head

Vitals Recording System
- Implement simplified EKG system through contact in Huggable's hands
- Conductive contacts in hands (copper thin plate)
- Output from circuit is voltage for processing/analyzing

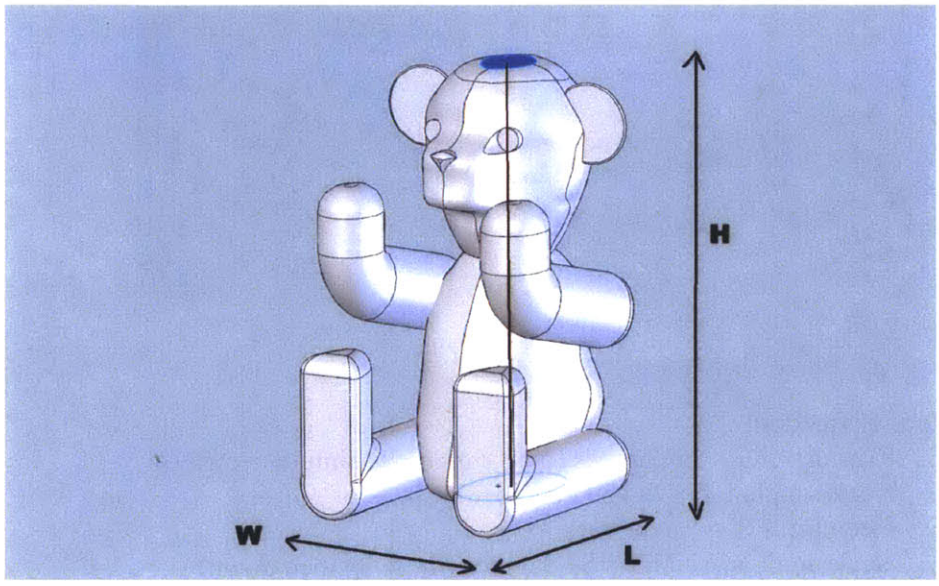
Other Things
- We want Huggable to survive at least 1m drop
- Desired battery life: 3 hours [More if possible----- how fast can we recharge. Ask Adam S.]

- Desired duty cycle: 60% (I think)
- Desired number of models: 10

Huggable Project - Desired Specifications:

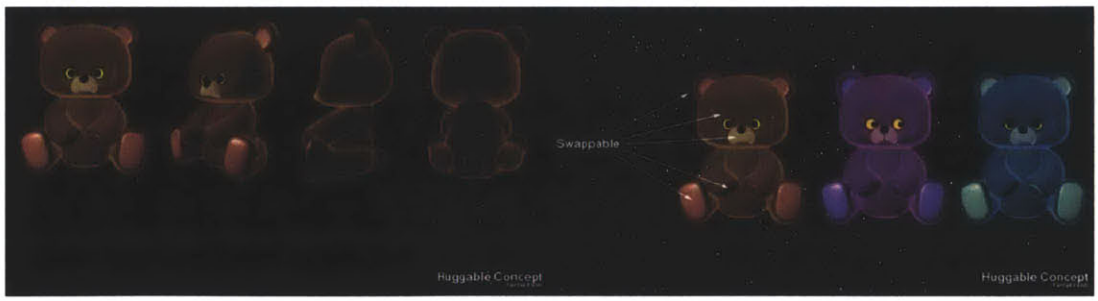
Project Dimensions:

Major Dimensions -



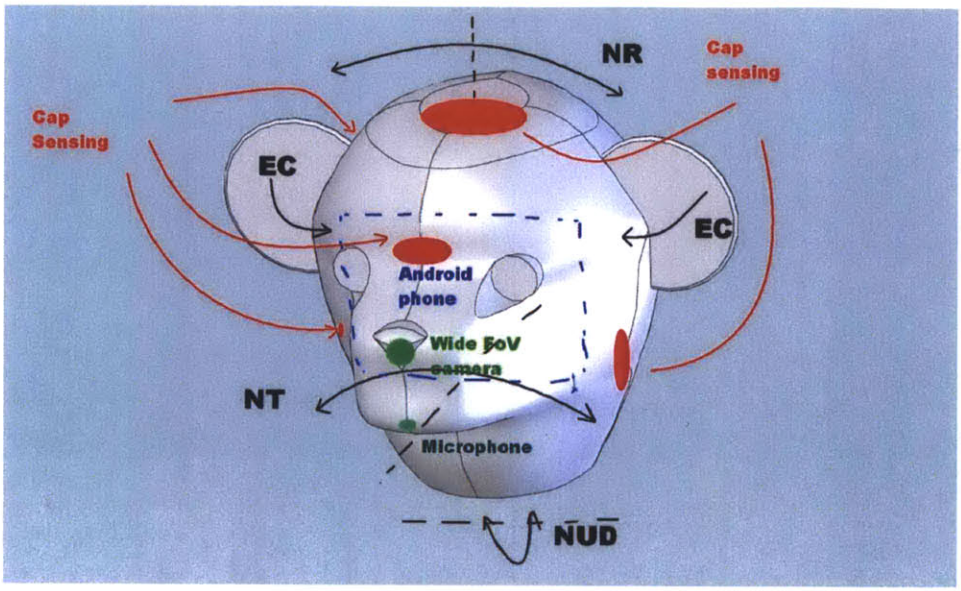
- Total Height: H = 300 - 350 mm
- Total Length: L = 200 - 250 mm
- Total Width: W = 150 - 200 mm

Final Appearance



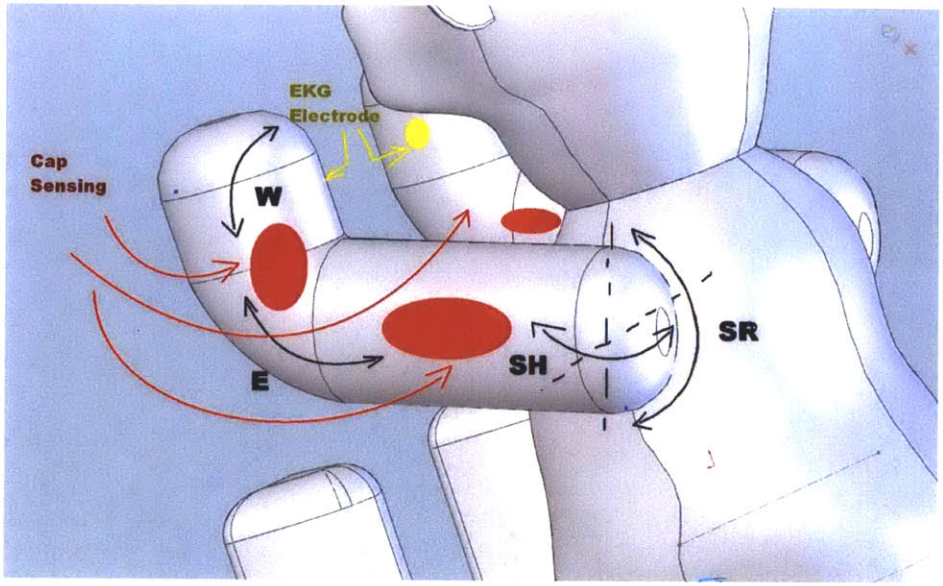
Features by Module

Head



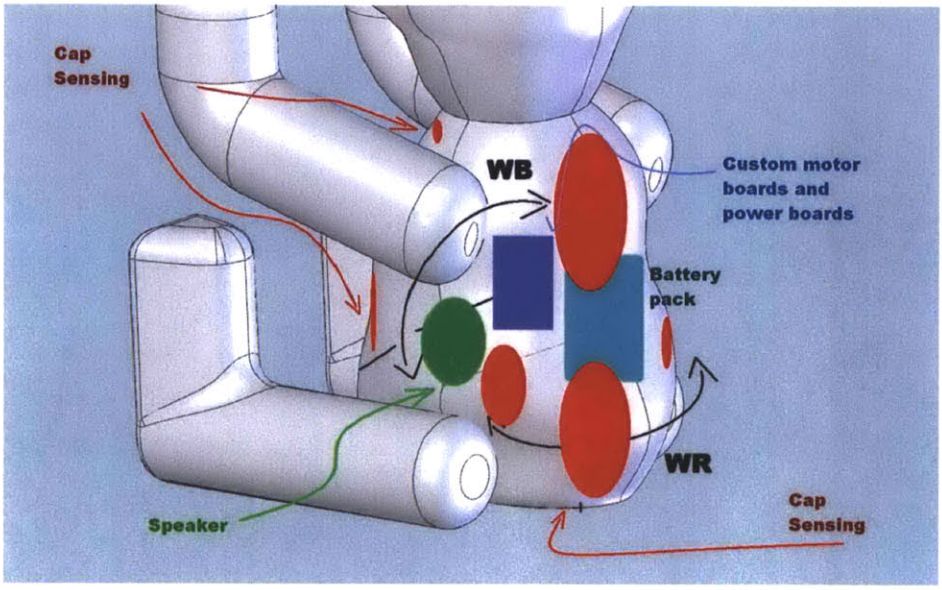
- Degrees of freedom
  - Ear curl: EC = 55 degrees (both ears move simultaneously)
  - Neck rotation: NR = 120 degrees
  - Neck tilt: NT = 30 degrees
  - Neck up/down: NUD = 50 degrees (more up then down)
- Wide FoV camera
  - Range: 100 degrees (desired), 80 degrees (will settle for)
  - Resolution: 640 x 480
  - Need wireless transmission package
- Android phone in head
  - Will need to make cradle mount in the head to place inserts for different smartphones (we would 3D print molds for phones, just need space to put mold)
  - Utilize screen for virtual eyelids/eyes/pupil/iris for expressive eyes (with clear domes for eyeballs)
  - Magnetic muzzle faceplate to cover phone and camera, and complete head
- Microphone in head
  - Would like mic specs similar to Pleo and Autom
  - Need omnidirectional microphone, detect left/right channel (at least), front/back/left/right (best)
- Capacitive sensing
  - Would like Pleo technology implementation (2mm detection)

Arms



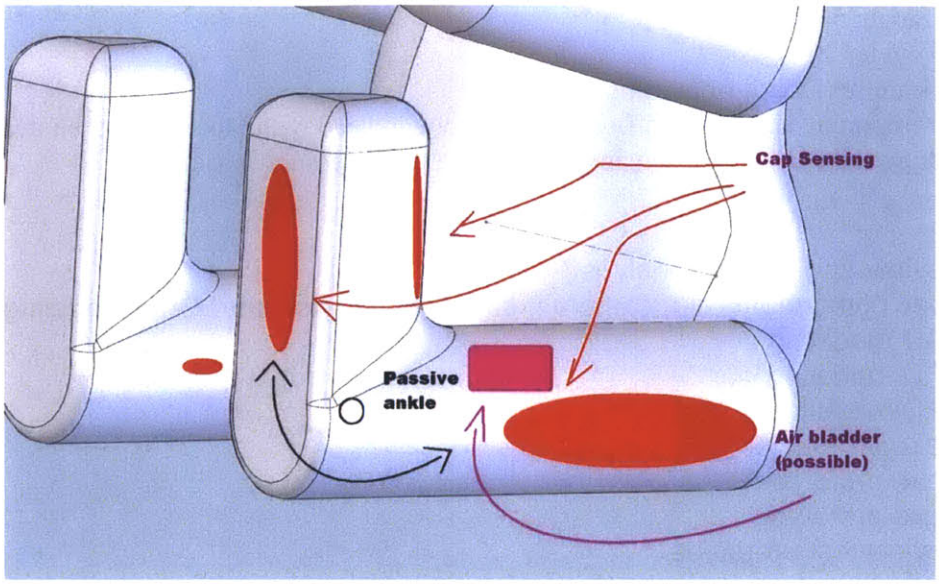
- Degrees of freedom
  - Shoulder rotate: SR = 150 degrees
  - Shoulder Hinge: SH = 110 degrees
  - Elbow: E = 90 degrees
  - Wrist: W = 40 degrees (coupled with elbow)
  - Arms operate independently from each other
  - Shoulder hinge operates independently from elbow
- EKG Electrode
  - Exposed through fur
  - Conductive contacts in hands (copper thin plate)
  - Routed to custom circuit board to output heartbeat signal as analog voltage
- Capacitive sensing
  - Would like Pleo technology implementation
- (Possible) Pressure sensing
  - Done with big FSR pads (perhaps)

Body



- Degrees of freedom
  - Waist bend: WB = 40 degrees
  - Waist rotate: WR = 40 degrees
- Battery pack
  - Desired battery life: 3-6 hours
- Capacitive sensing
  - Would like Pleo technology implementation
- Speaker
  - iHome iH77 speaker (CMS0401KL-1X)
  - CMS series, 20-270Hz, 8 Ohms, 4 Watts, Magnetic, 40mm L x 40mm W x 14.5mm H, 86dB pressure level
  - Better speaker if possible
- Custom motor boards and power management boards
  - My group will design these; will be great if they can be made here
  - Need to include space to house them

Legs



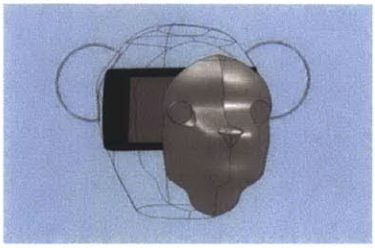
- Passive degree of freedom in ankle
  - Just for sensing if someone plays with foot
- Capacitive sensing
  - Would like Pleo technology implementation
- Pressure sensing
  - Maybe implement air bladder or FSR here for sensing

Motor / Feedback

- Maximum velocity: 15-20 RPM (we want smooth movement)
- Analog potentiometers for joint data
- Currently using P12426CT-ND (POT 10K OHM 10MM 347 DEGREE SMD)
- We like this component, easy to use and slim profile
- Would like to place passive feedback in ankles using these pots
- Motor encoders: 78-100 counts per 1 degree resolution at motor shaft

Processing

- Using Android-based smartphone to handle higher-level component processing.
- We would like a cradle in head to push open/push close to insert smartphone
- Different phones, but at least view screen of 4.3"



Other Things

- Age range: 5+
- Duty cycle: 20%
- Replicate fur length from sample bear
- Most important feature is to fit Android phone in head with face cover, then degrees of freedom, then capacitive sensing, then EKG, then pressure sensing.

Schedule

Week 1: Review and finalize "wish list"
Week 2: Spec components like cameras, cap sensing, etc., start CAD modeling
Week 3: CAD modeling
Week 4: CAD modeling
Week 5: CAD modeling
Week 6: Make prototype
Week 7: Make prototype
Week 8: Make prototype
Week 9: Review/test prototype
Week 10: Modify prototype / more testing
Week 11: Modify prototype / more testing
Week 12: Finish engineering models
Week 13: Finish engineering models

Huggable Project - Desired Specifications (unfinished yet)

Old Huggable v3.0 Current Components

Head

- 4 Microphones in head
  - 2 Microphones in Ears (Supercircuits PA3)
  - 1 Microphone in Back of Head (Supercircuits PA3)
  - 1 Microphone in Front of Head between cameras (Supercircuits PA3)
- Sensitive Skin System
  - Head
- Speaker in Mouth
  - Modified iHome iHM7
- Cameras in the Eyes (Electronics)
  http://www.supercircuits.com/search?keywords=snake+video
  - Color Supercircuits PC229XP
  - B/W Supercircuits PC224XP (note newer model PC229HRXP)
- 3 DOF
  - Neck Nod
    - Potentiometer: Digikey
    - Motor: MicroMo 2232U012SRIE2-512
    - Mechanisms: SDP-SI
  - Neck Tilt
    - Potentiometer: Digikey
    - Motor: MicroMo 2232U012SRIE2-512
    - Mechanisms: SDP-SI
  - Ears
    - Potentiometer: Digikey
    - Motor: MicroMo 2232U012SRIE2-512
    - Mechanisms: SDP-SI

Body

- Video Digitizers (2)
  - Kworld DVD Maker 2 http://us.kworld-global.com/main/prod_in.aspx?mnuid=1306&modid=10&pcid=73&ifid=17&prodid=102
- 5 DOF
  - Left Shoulder Rotate:
    - Potentiometer: Digikey
    - Motor: MicroMo 2232U012SRIE2-512
    - Mechanisms: SDP-SI

    - Right Shoulder Rotate:
        - Potentiometer: Digikey
        - Motor: MicroMo 2232U012SRIE2-512
        - Mechanisms: SDP-SI
    - Left Shoulder Up/Down:
        - Potentiometer: Digikey
        - Motor: MicroMo 2232U012SRIE2-512
        - Mechanisms: SDP-SI
    - Right Shoulder Up/Down:
        - Potentiometer: Digikey
        - Motor: MicroMo 2232U012SRIE2-512
        - Mechanisms: SDP-SI
    - Neck Rotate:
        - Potentiometer: Digikey
        - Motor: MicroMo 2232U012SRIE2-512
        - Mechanisms: SDP-SI
- Pot/Temp Board (Electronics)
    - Left Ankle
    - Left Hip In/Out
    - Left Hip Up/Down
    - Right Ankle
    - Right Hip In/Out
    - Right Hip Up/Down
- IMU
- Sensitive Skin System
    - Left Leg
    - Right Leg
    - Left Arm
    - Right Arm
    - Body
- 8-Channel Motor Driver Board
- Hard Drive
- Embedded PC
- 802.11n WiFi USB Adapter
- Voltage Regulator Board

Components of Interest in Old Version

Motor: Micromotors Faulhaber 2232U012SR
Nominal voltage: 12 V
Terminal resistance: 4,09 Ω
Output power: 8,7 W
Efficiency, max.: 86%
No-load speed: 7 100 rpm
No-load current (with shaft ø 2 mm): 0,0175 A
Stall torque: 0,477 kg-cm
Friction torque: 2.855 g-cm
Speed constant: 595 rpm/V
Back-EMF constant: 1,68 mV/rpm

Torque constant: 16 mNm/A
Current constant: 0,062 A/mNm
Slope of n-M curve: 152 rpm/mNm
Weight: 62 g

Recommended settings

Speed up to 8 000 rpm
Torque up to 10 mNm
Current up to (thermal limits) 0,94 A

Microphone: Supercircuits PA3

Frequency: 20 - 16 000 Hz
S/N ratio: > 58 db
Power: 12 V DC @ 20 mA (max)

Speaker: iH77 Mini Speaker

CMS series
Range: 20-270Hz
Impedance: 8 Ohms
Power: 4 Watts
Type: Magnetic
Dims: 40mm L x 40mm W x 14.5mm H
Pressure: 86dB

Camera: Supercircuits Snake Camera

B/W Camera

Horizontal Resolution: 768
Imager Manufacturer: Sony
Imager Type: Ex-View CCD
Lines: 570
Lux: 0.005
Vertical Resolution: 494
Max Focal Length (mm): 3.7
Ultra Pinhole: yes
Physical Characteristics
Depth (in): .45
Height (in): .75
Mounting Bracket Included: no
Width (in): .45
Amps DC (mA): 70
Volts DC: 12

Color Camera

Imager Manufacturer: Sony
Imager Type: Ex-View CCD 510 x 492 pixels
Lines: 540
Lux: 0.05
Max Focal Length (mm): 3.7
Ultra Pinhole: yes
Depth (in): 0.45
Height (in): 0.75
Width (in): 0.45
Amps DC (mA): 70

Volts DC: 12

Components for New Version

Motor: TBD
Camera: TBD
Microphone: Billion Elite BE6027G
http://www.billionelite.cn/eshop_show.asp?id=121
RESONANT IMPEDANCE(KΩ): ≤ 2.2
STANDARD OPERATION VOLTAGE (V): 3
SENSITIVITY(dB): 60±2
FREQUENCY RANGE(Hz): 20-----20000
DIRECTIVITY: OMNIDIRECTIONAL
Speaker: TBD

Weeks 3 and 4

Contents of this Book:

1. Motor / gearbox selection for the Huggable
2. Fur selection for new Huggable concept
3. v4 Mechanical Layout Pictures
    a. v4.1 Layout (First attempt at layout)
    b. v4.2 Layout (Second attempt with rudimentary mechanisms)
    c. v4.3 Layout (New Concept Outline)
    d. v4.4 Layout (Final Concept Outline)
    e. v4.5 Layout (Proper Mechanical Outlines)
4. Waist test pictures
5. Neck test pictures

Motor / Gearbox Selection for Motions

DC Carbon-brush motors

PG16M050 Geared Motor Series

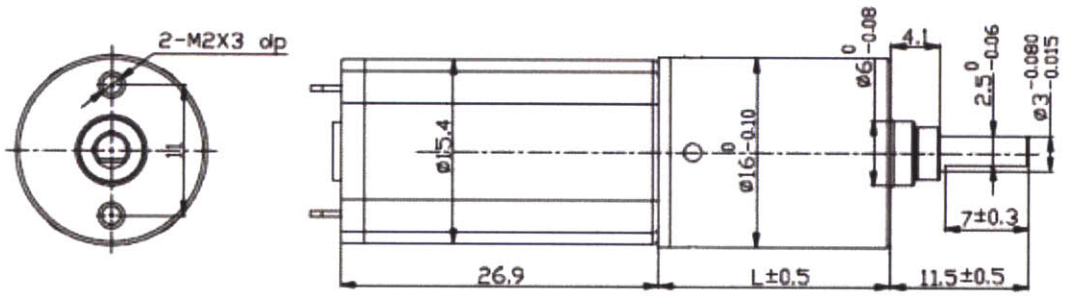
PG16M050

Typical applications:

Optical equipment,Montitoring cameras,Kind of figer-electroni locks,Automatic energy savingbath,Water IC card,Toys and gifts,Office equipment,Household appliances,Automatic actuator.

- **Ear/snout motion, elbow motion, shoulder hinge motion**
- **Neck up/down motion, neck left right motion**
- **Neck twist motion**
- **Shoulder twist motion**

Appearance size



Geard Motor Torque/Speed

Reduction ratio		1/4	1/4.75	1/16	1/19	1/22.5	1/64	1/76	1/90	1/107	1/256	1/304	1/361	1/428	1/509	1/1024	1/1216	1/1444	1/1714	1/2036	1/2418
6V	Rated torque (gf.cm)	27	32	97	115	137	350	416	493	586	1250	1480	1760	2080	2480	4000	4000	4000	4000	4000	4000
	Rated speed (r/min)	1625	1369	407	343	289	102	85	72	61	25	21	18	15.2	12.8	6.3	5.4	4.5	3.8	3.2	2.7
9V	Rated torque (gf.cm)	25	30	90	108	127	326	388	460	545	1150	1380	1640	1950	2310	4000	4000	4000	4000	4000	4000
	Rated speed (r/min)	1625	1369	407	343	289	102	85	72	61	25	21	18	15.2	12.8	6.3	5.4	4.5	3.8	3.2	2.7
Gearbox "L" mm		15.8		15.8			19.4				23					26.6					

Motor Data

Rated volt (V)	Rated torque (g-cm)	Rated speed (rpm)	Rated current (mA)	No load speed (rpm)	No load current (mA)	Rated output (w)	Weight (g)
6	7.5	6500	≤160	8000	≤65	0.488	18
9	7.0	6500	≤110	8000	≤50	0.455	18

Motors for all motions except waist motion

MABUCHI MOTOR

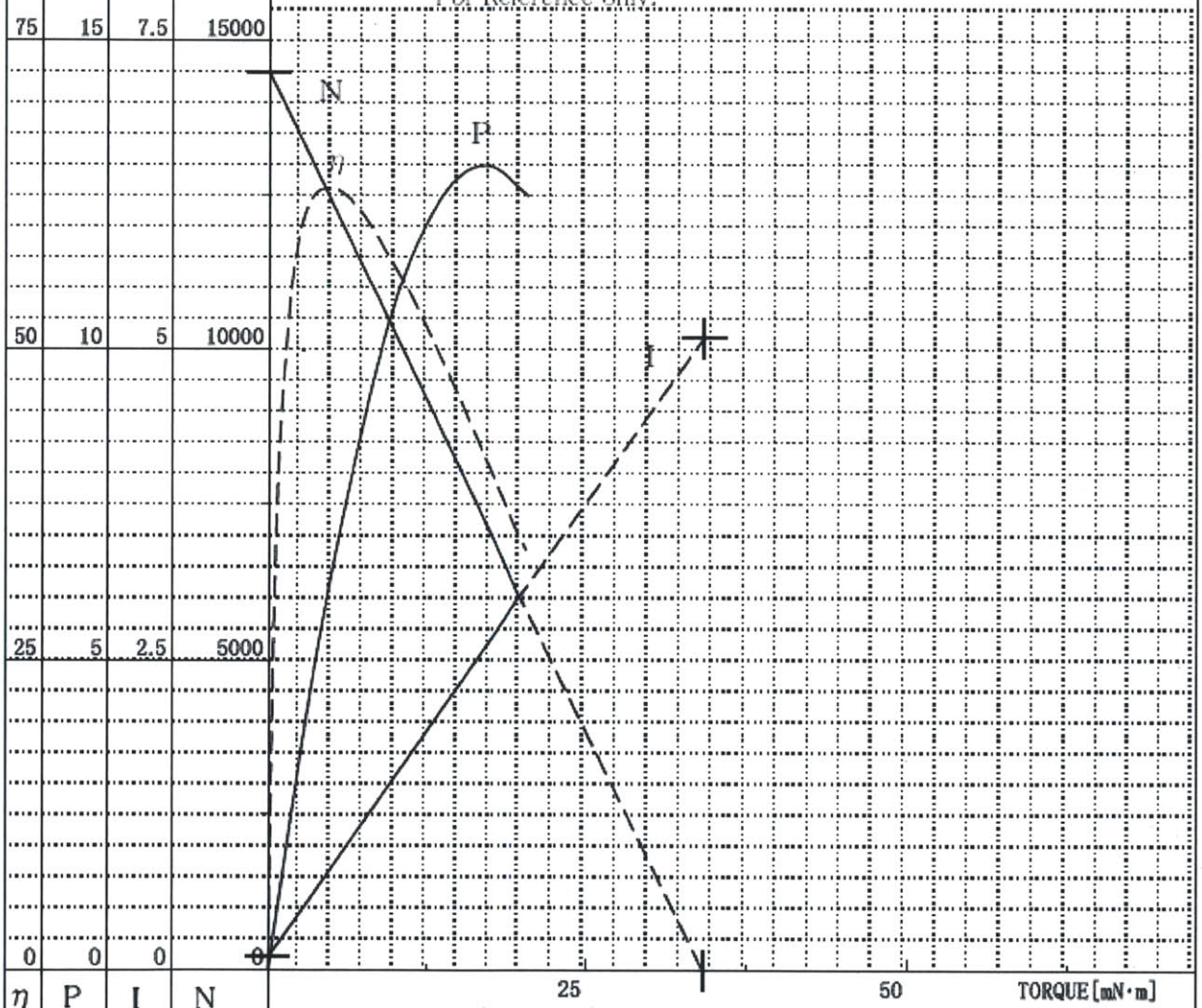
DATE 3.2.01
approx.25℃

MODEL FC-280PT (22125)

VOLTAGE 12 V

CCW(0)/CW(0) Constant

For Reference only.

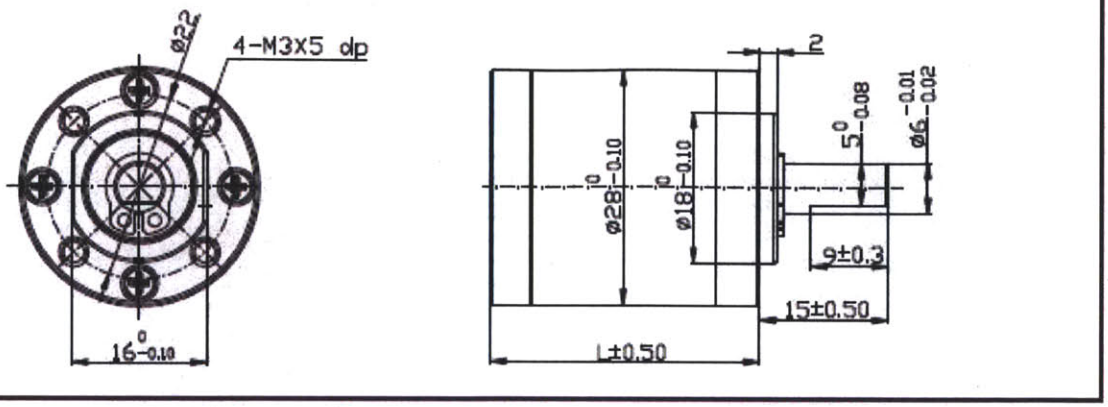


N = 40

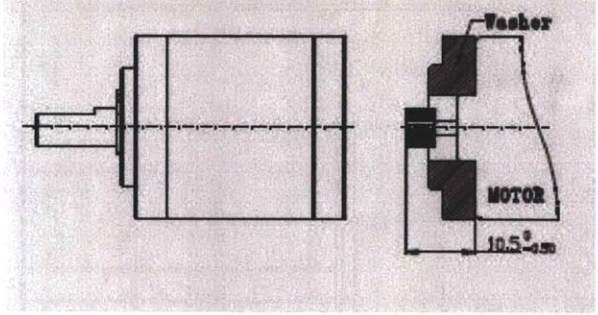
No. FC280-0232

At Maximum Efficiency		At Maximum Power		Motor Characteristics			
Speed	12470 r/min	Torque	17.16 mNm	No	14500 r/min	Ec	2.39 V
Current	0.826 A	Speed	7250 r/min	Io	0.13 A	α	29.47
Power	6.275 W	Current	2.615 A	Is	5.1 A	V	12 V
Torque	4.805 mNm	Power	13.03 W	Ts	34.32 mNm	Kt	6.906 mNm/A
Eff.	63.32 %	Eff.	41.52 %	R	2.353 Ω	Km	4.502 mNm/√W

Appearance size



Motor Installation



Pinion Specifications

Module	0.4	0.4
NO.of teeth	12	15
Pressure angle	20°	
Hole diameter	Φ1.98, Φ2.28, Φ2.98	
Reduction ratio	4, 19, 22.5, 90, 107, 304, 361, 428, 509, 2036, 2418	4.8, 16, 64, 76, 256, 1024, 1216, 1444, 1715

Operating temperature range:−10℃ ~ +60℃

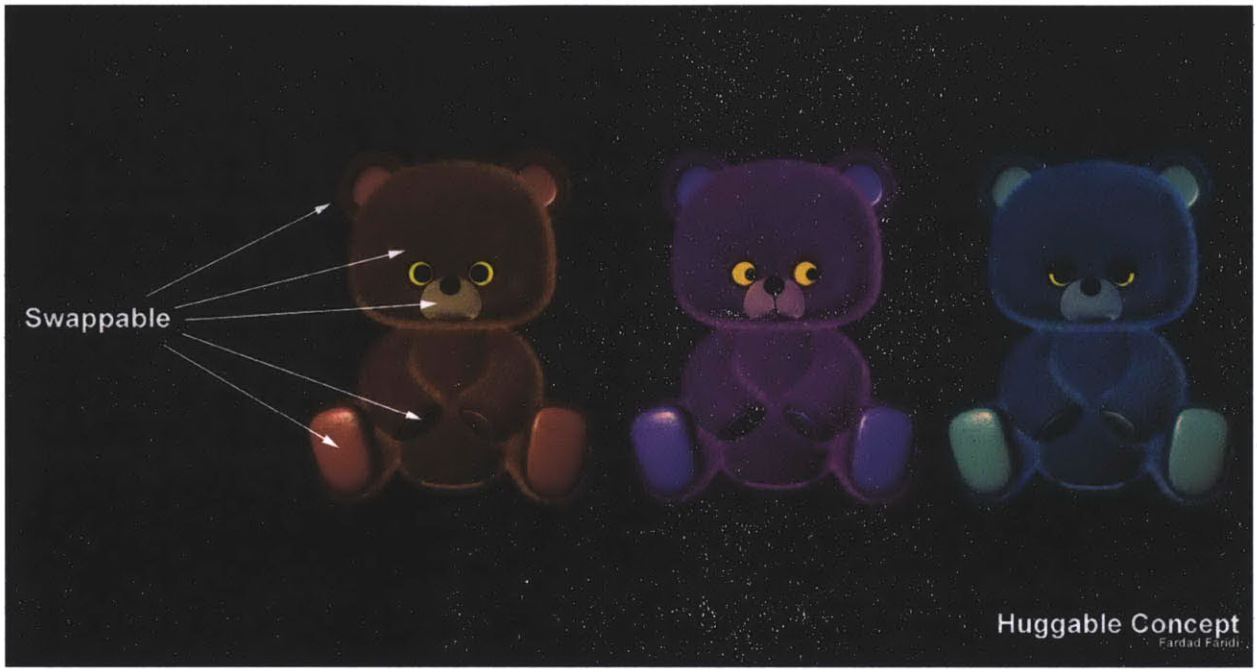
Operating relative humidity:20%~85%RH

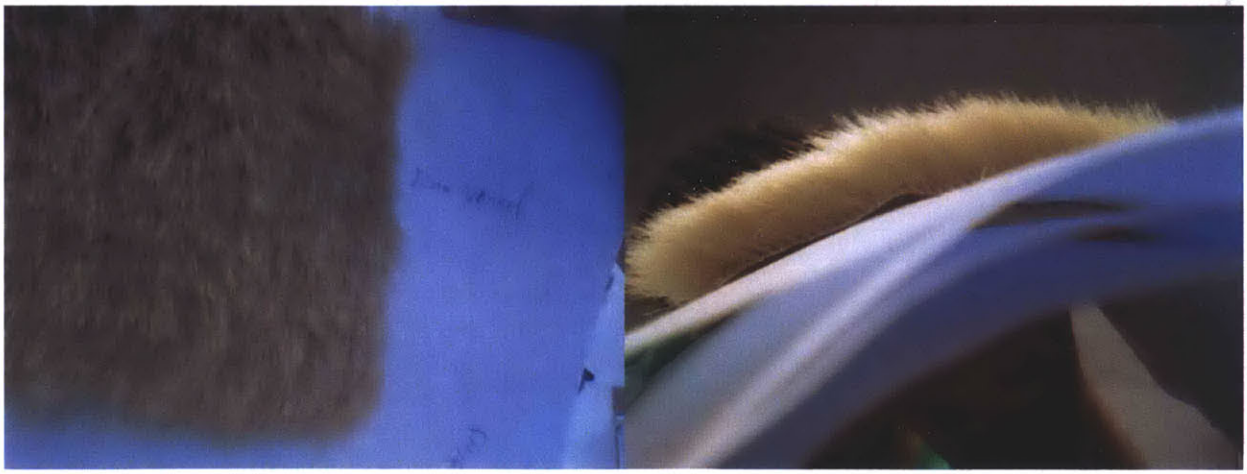
Gearboxes Specifi cations

Raduction ratio	Reduction absolute	Rated tolerance torque	Max breaking torque	Efficiency	Number of stages	Gearbox length "L" mm	Direction (Relationship with motor)
4, 4.75	4, 4 3/4	2.0kgf • cm	6.0kgf • cm	90%	1	22.0	Different
16, 19, 22.5	16, 19, 22 9/16	3.0kgf • cm	9.0kgf • cm	81%	2	27.1	Same
64, 76, 90	64, 76, 90 1/4	4.0kgf • cm	12kgf • cm	73%	3	32.2	Different
107	107 11/64	4.0kgf • cm	12kgf • cm	73%	3	32.2	Different
256, 304	256, 304	6.0kgf • cm	18kgf • cm	65%	4	37.3	Same
361, 428	361, 428 11/16	6.0kgf • cm	18kgf • cm	65%	4	37.3	Same
509	509 17/256	6.0kgf • cm	18kgf • cm	65%	4	37.3	Same
1024, 1216	1024, 1216	10kgf • cm	30kgf • cm	59%	5	42.4	Different
1444, 1715	1444, 1714 3/4	10kgf • cm	30kgf • cm	59%	5	42.4	Different
2036, 2418	2036 17/64, 2418 57/1024	10kgf • cm	30kgf • cm	59%	5	42.4	Different

This gearbox will allow the Mabuchi 280 motor to handle the load of operating the waist.

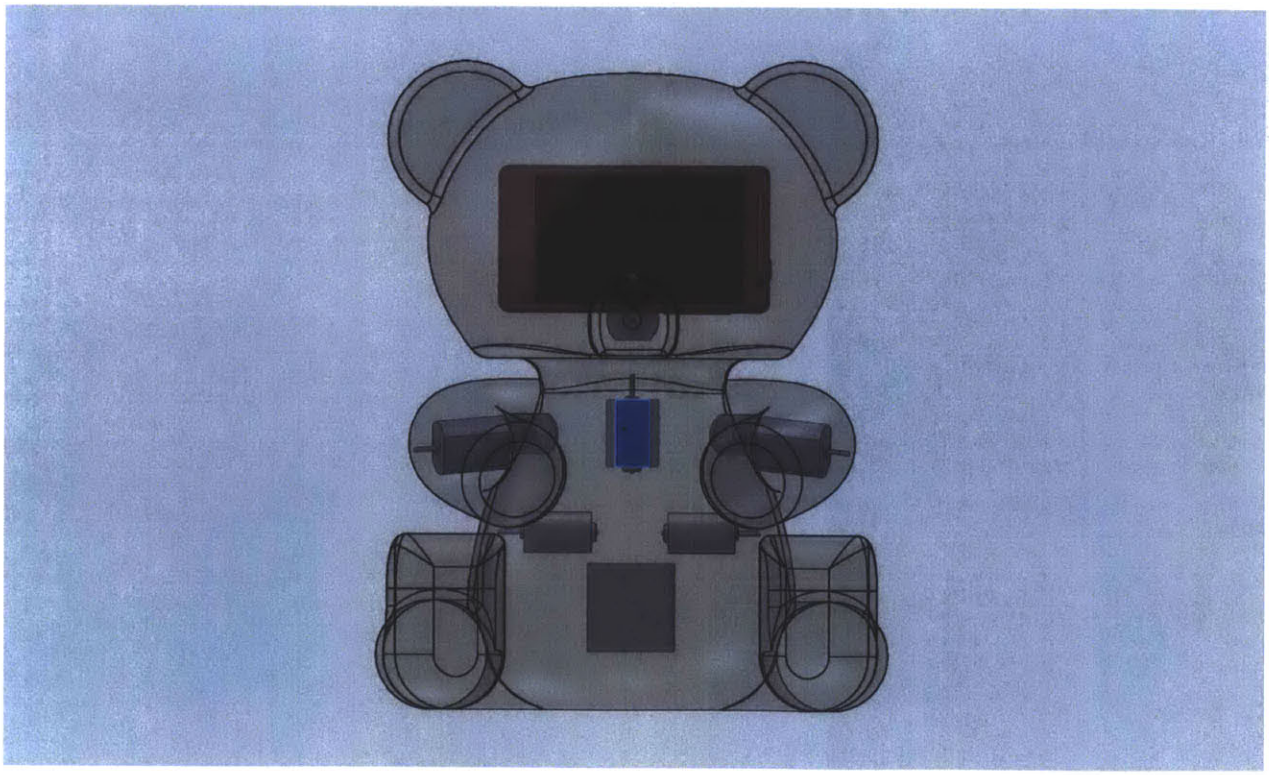
Fur Selection for Huggable

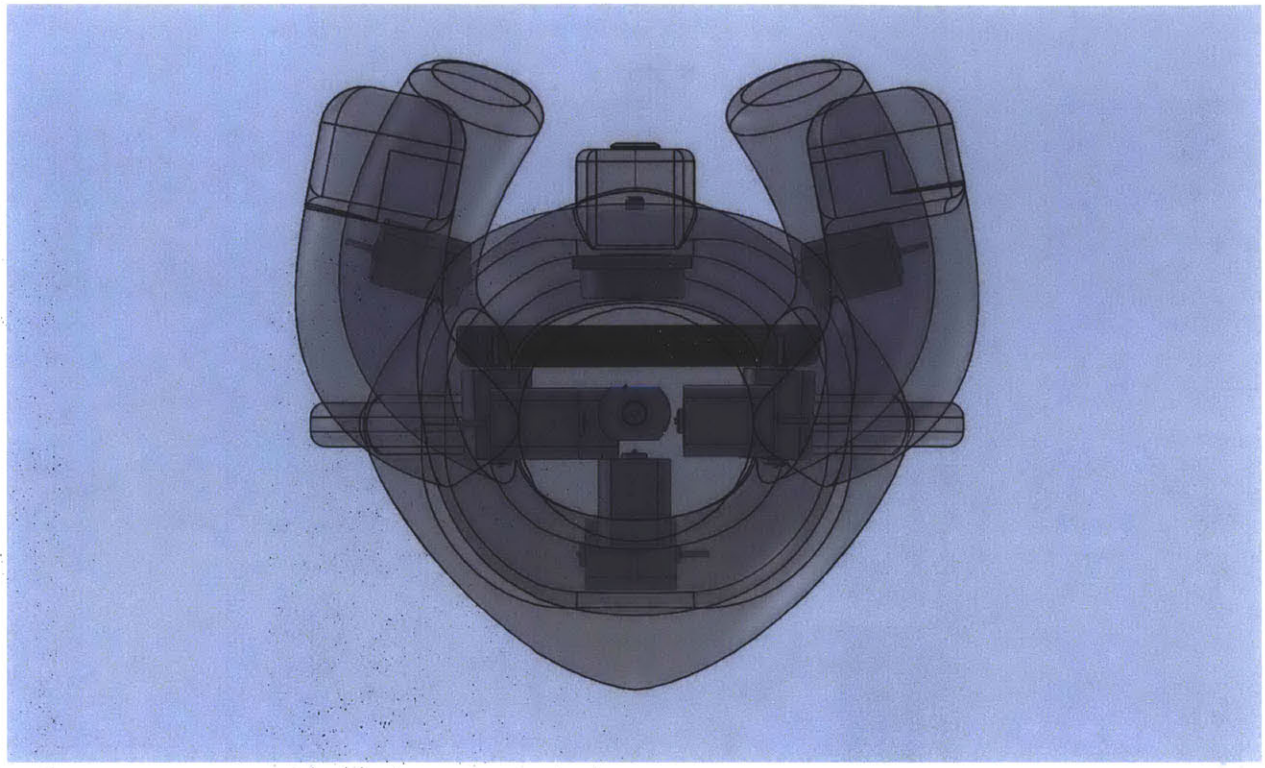


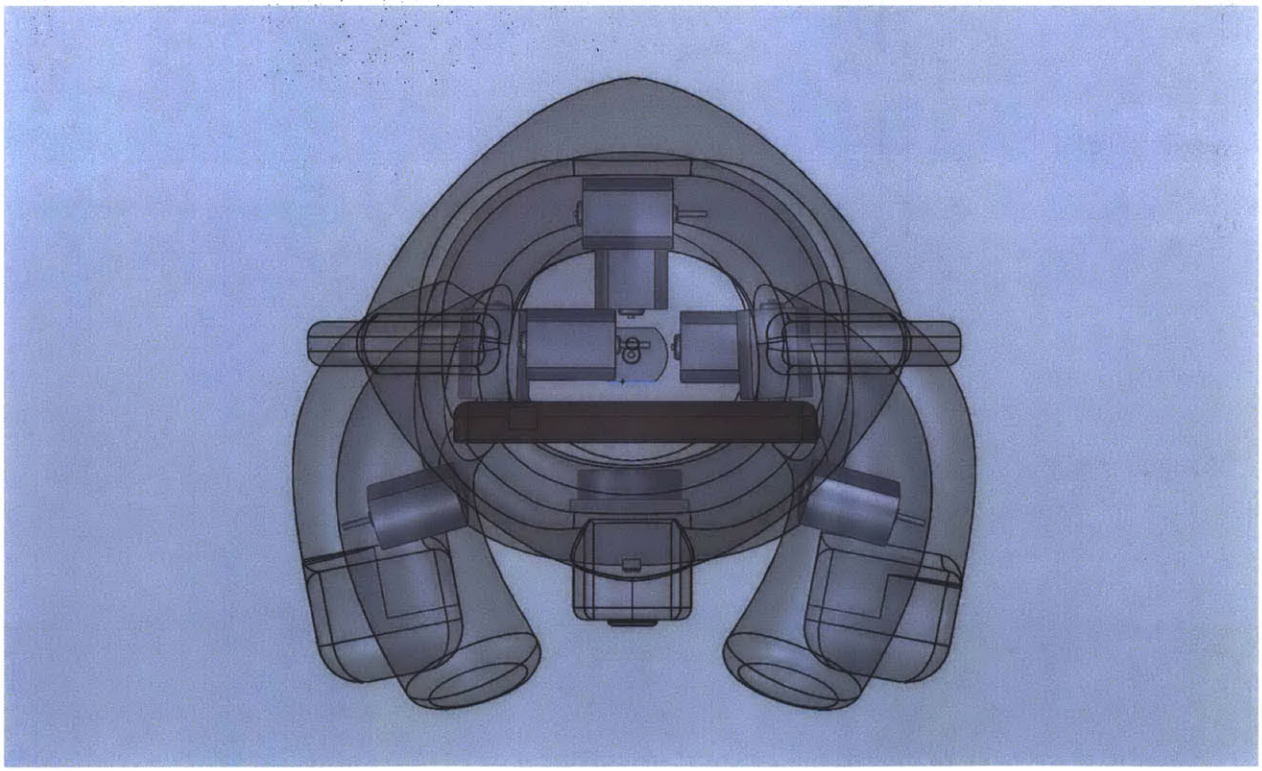


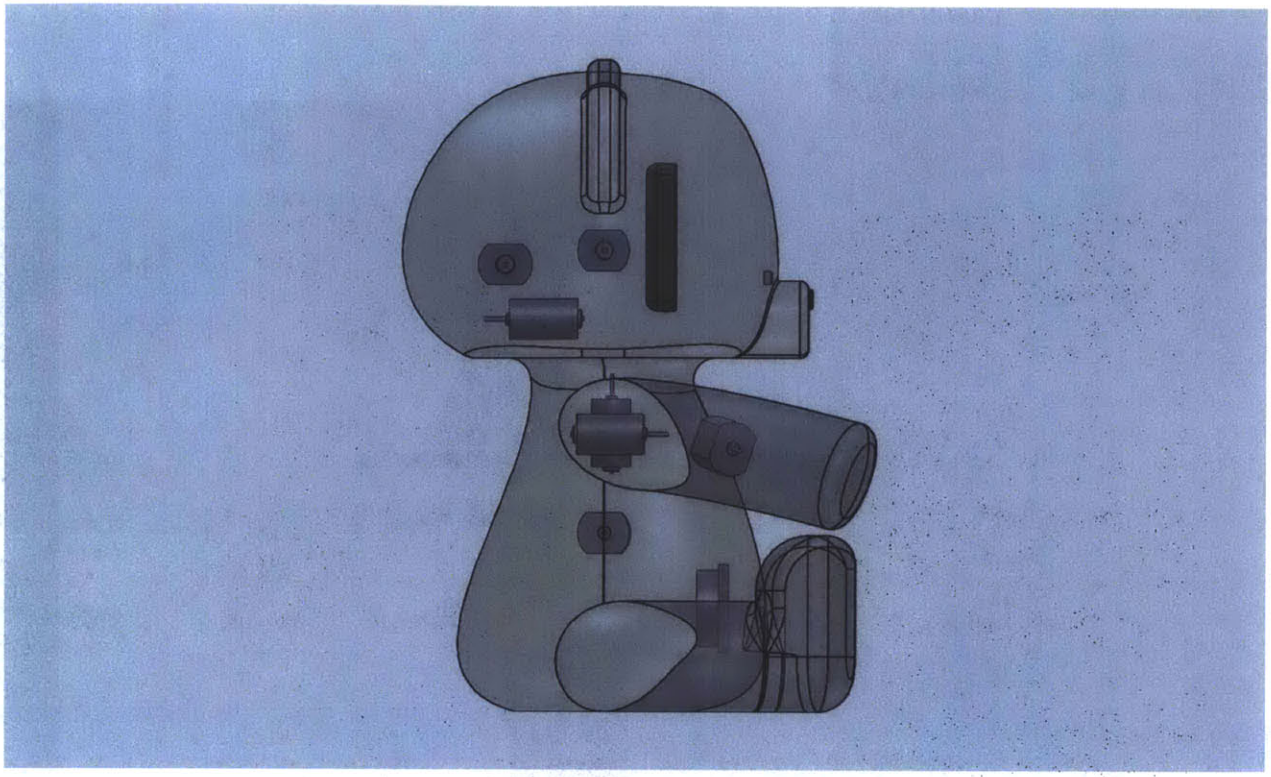
The fur selected here is nice and soft, ideal for Huggable application.

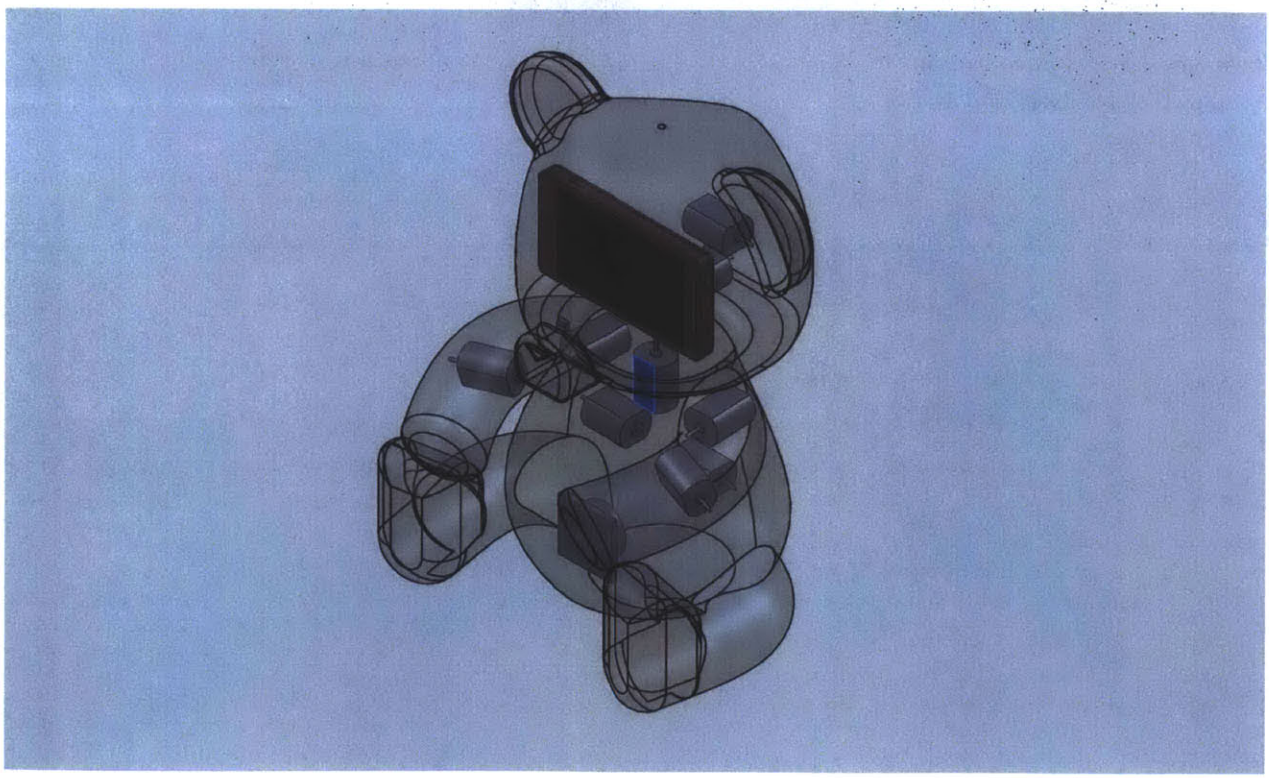
v4.1 Layout Pictures



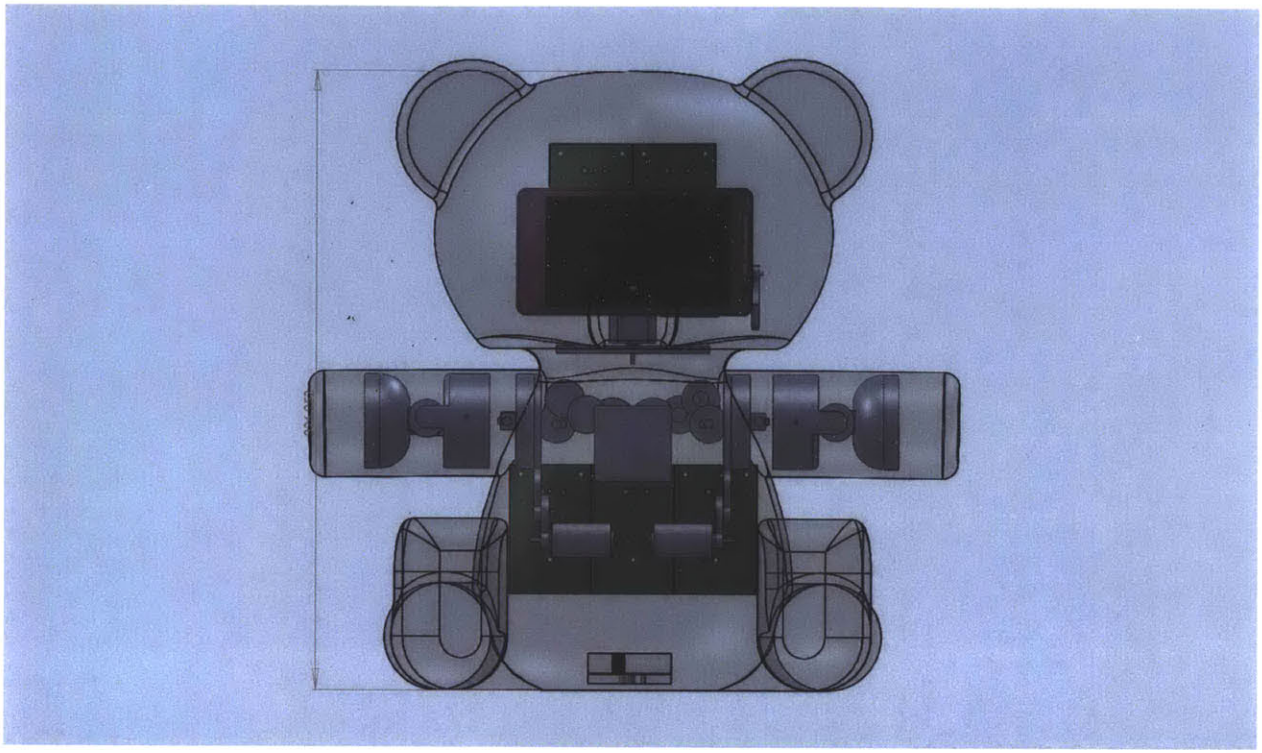


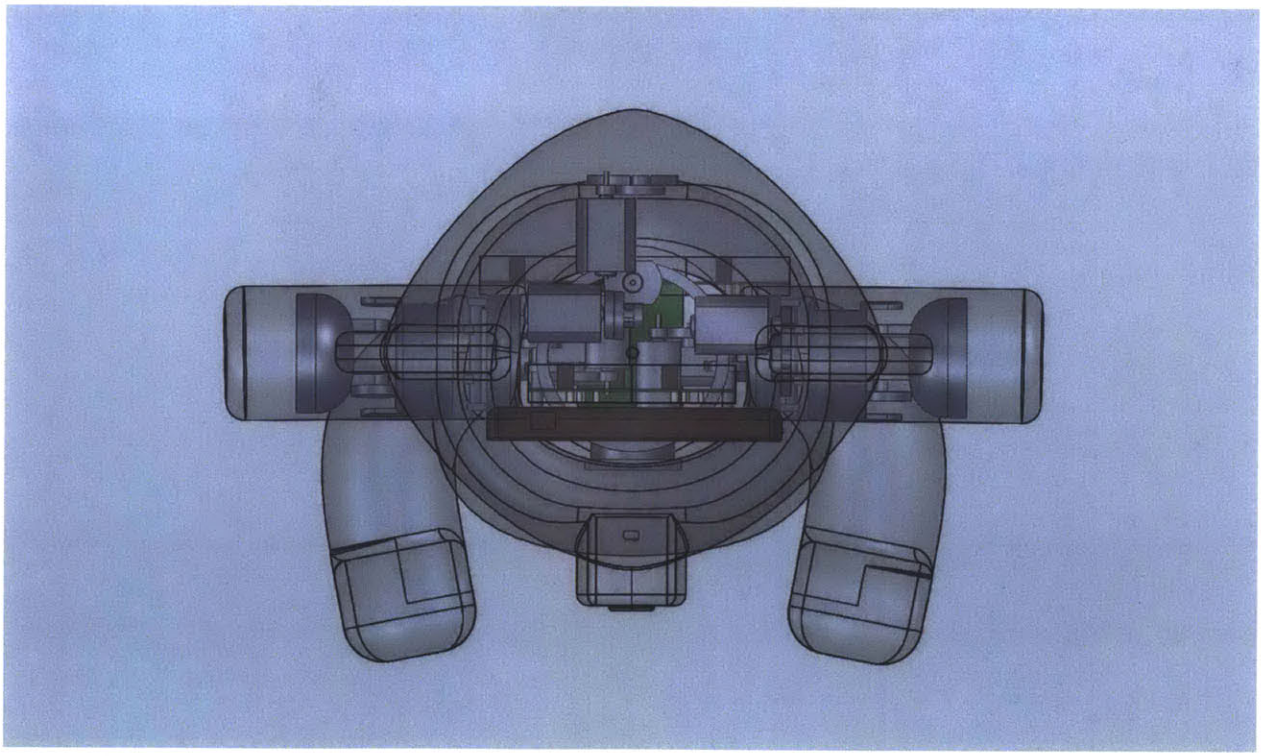


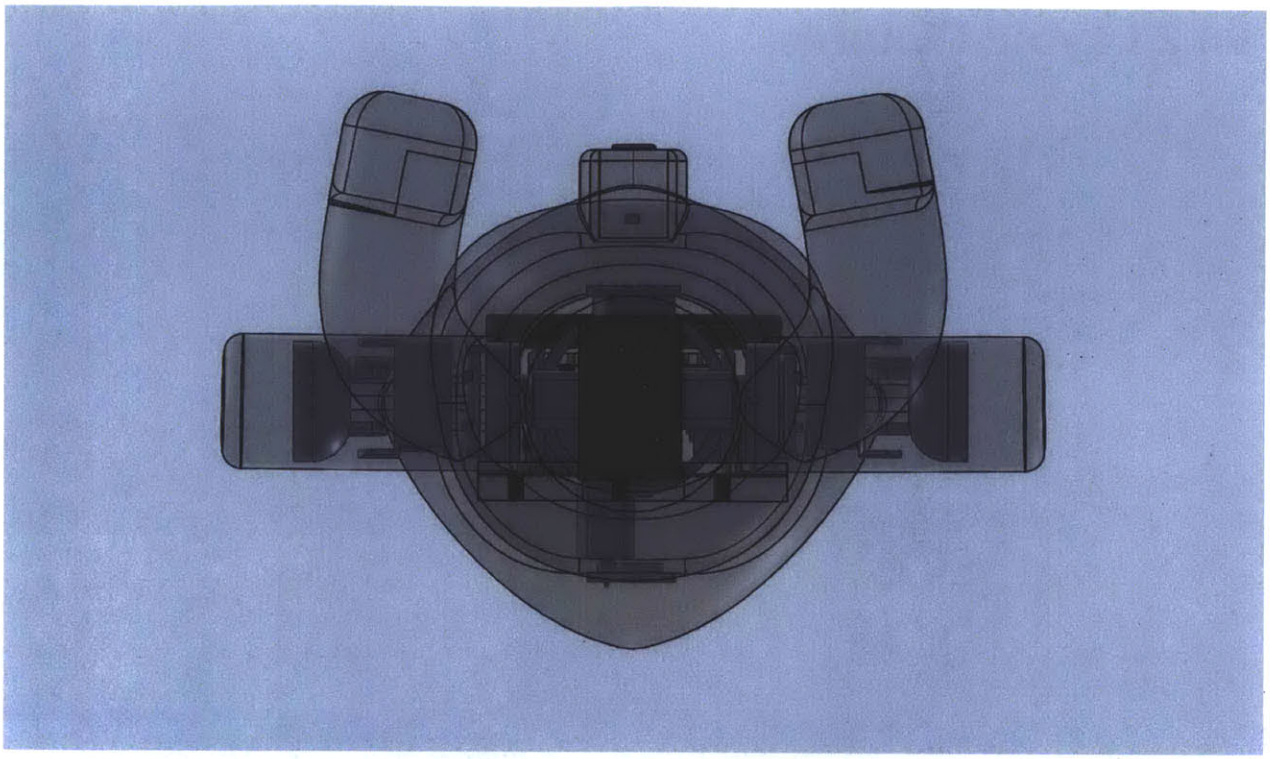


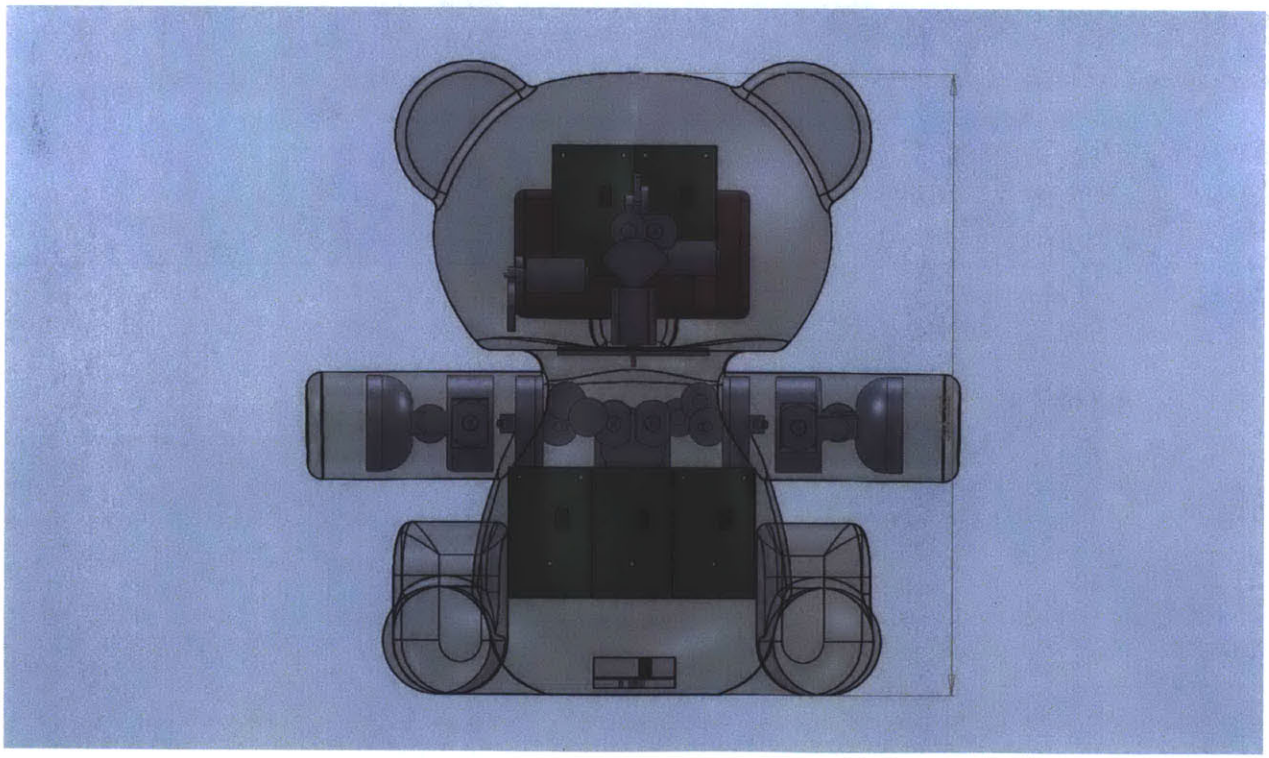


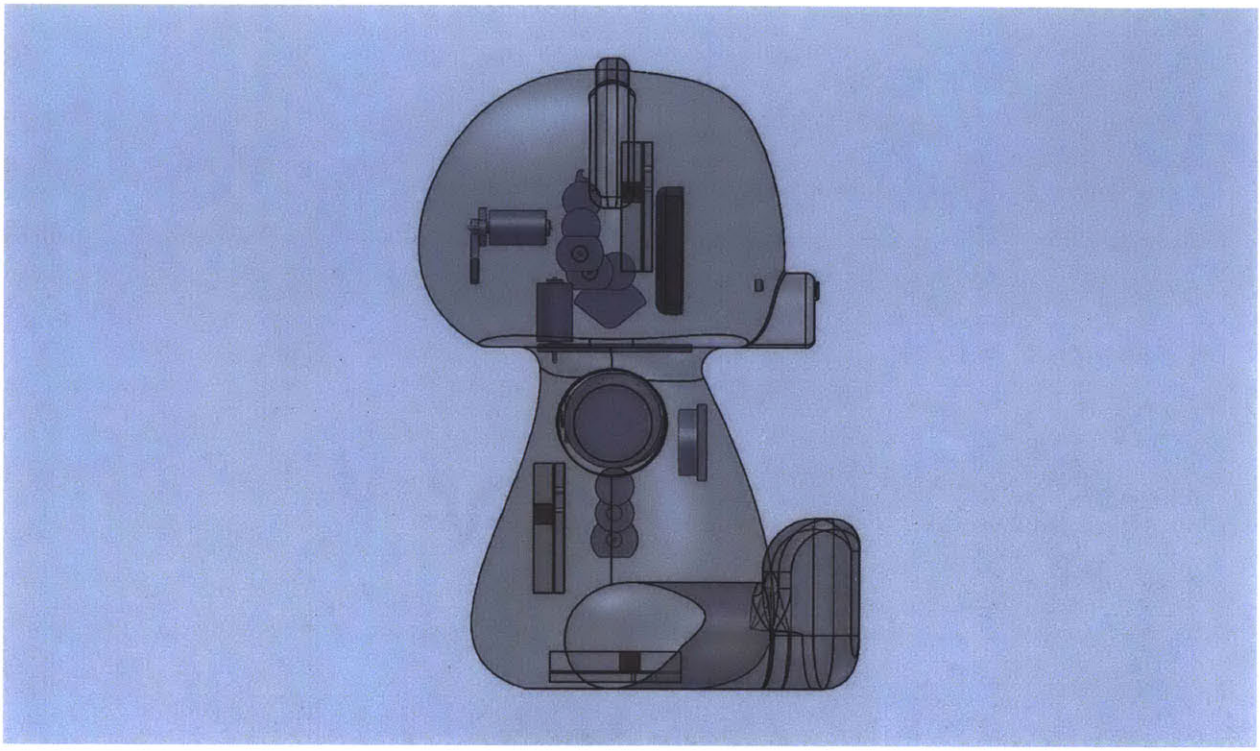
v4.2 Layout Pictures

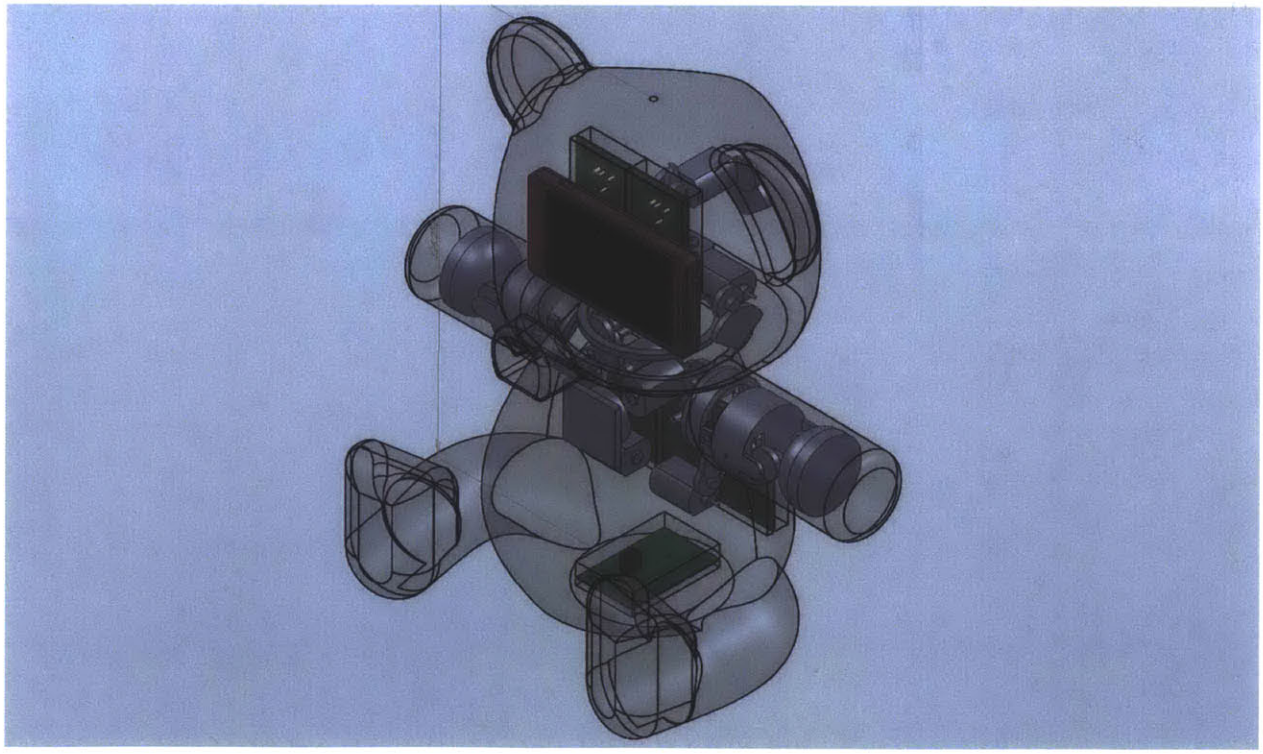


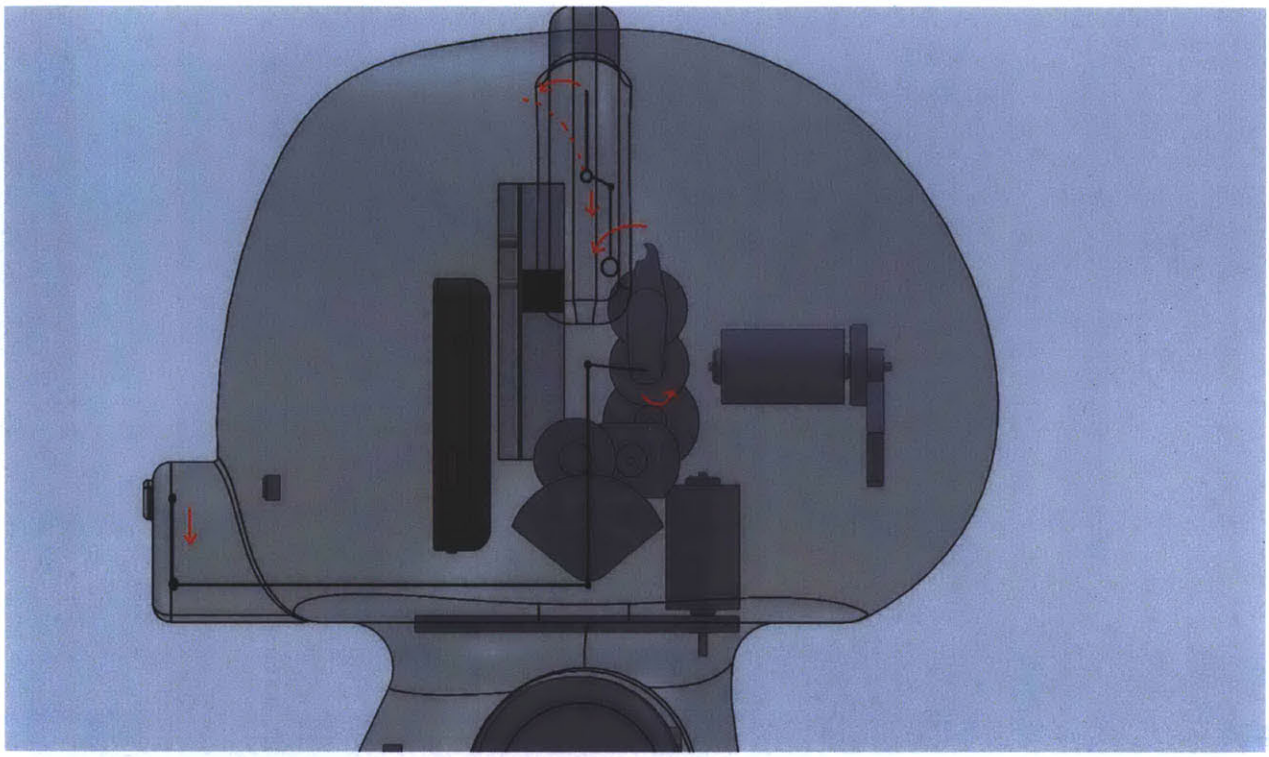


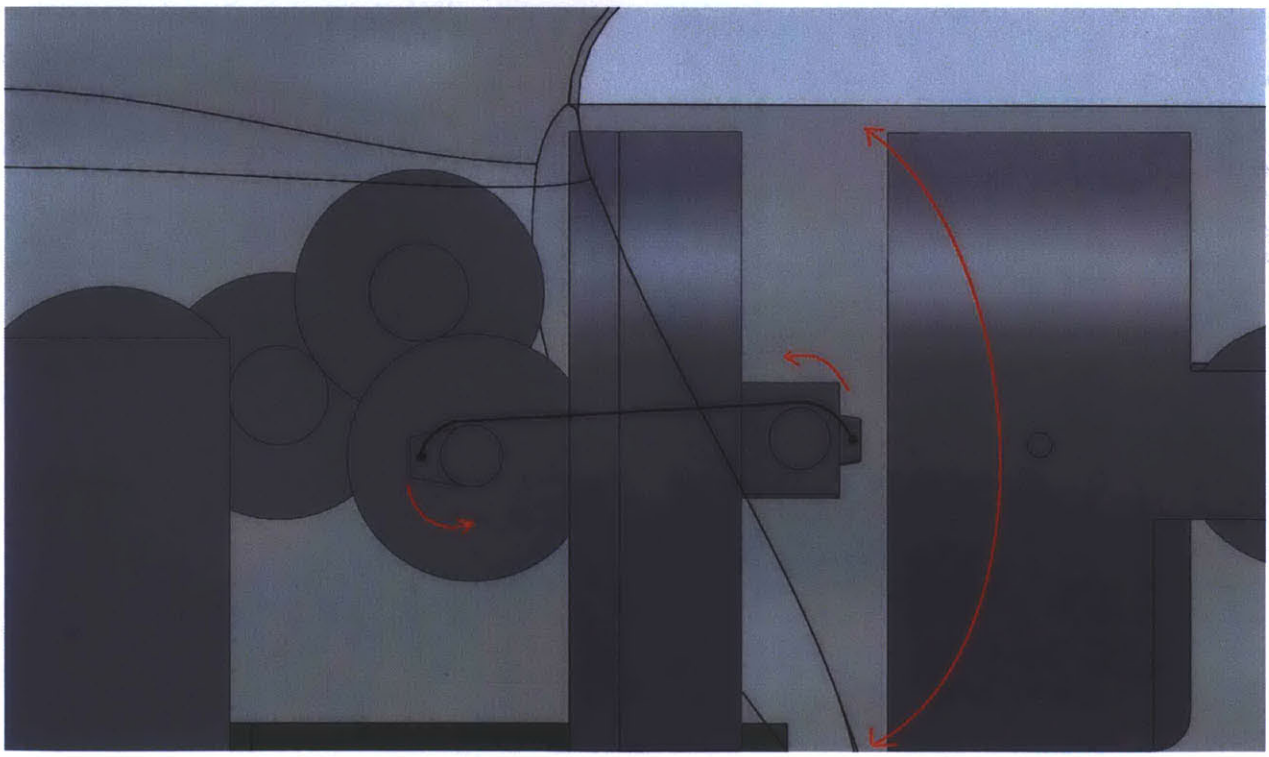




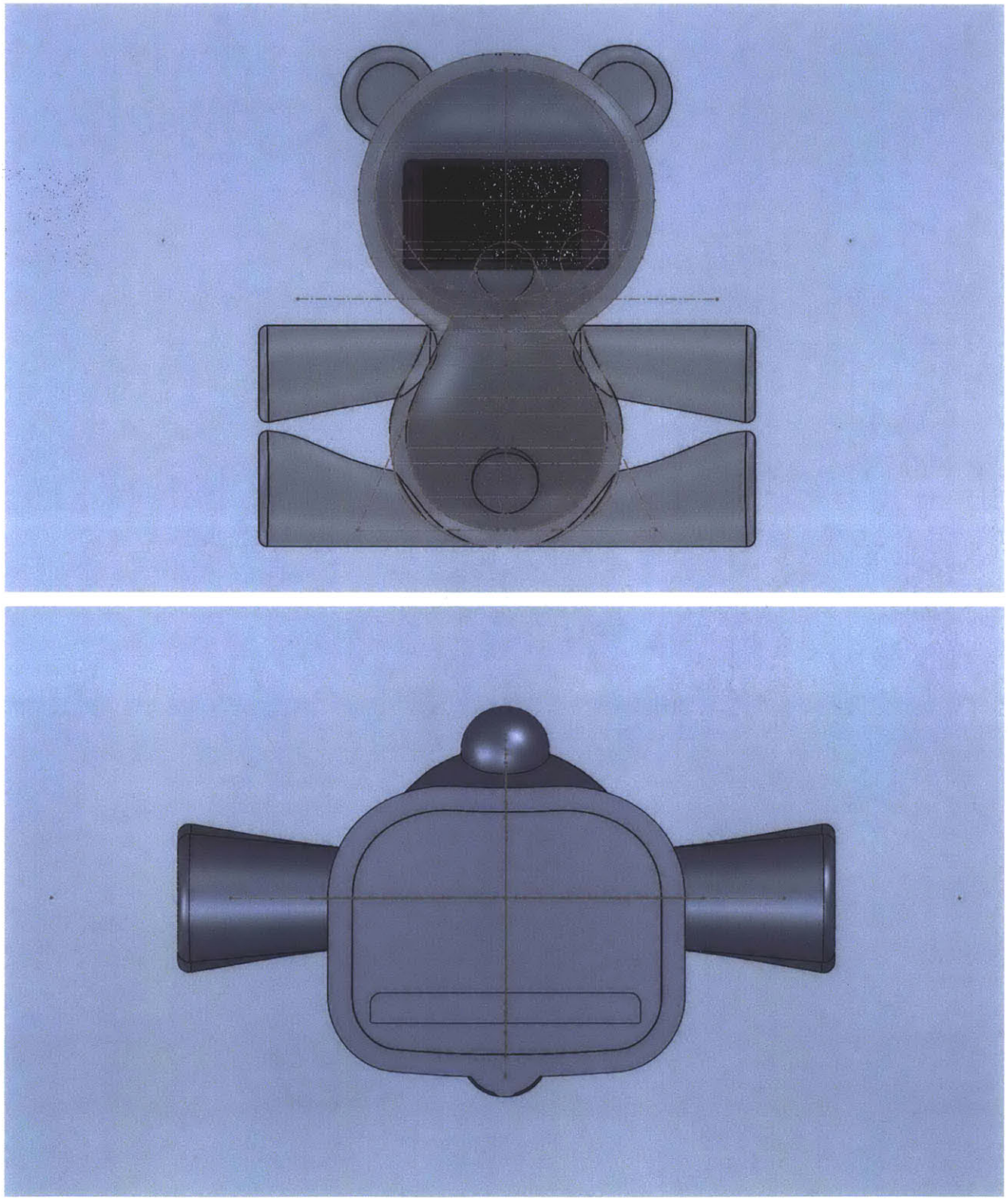


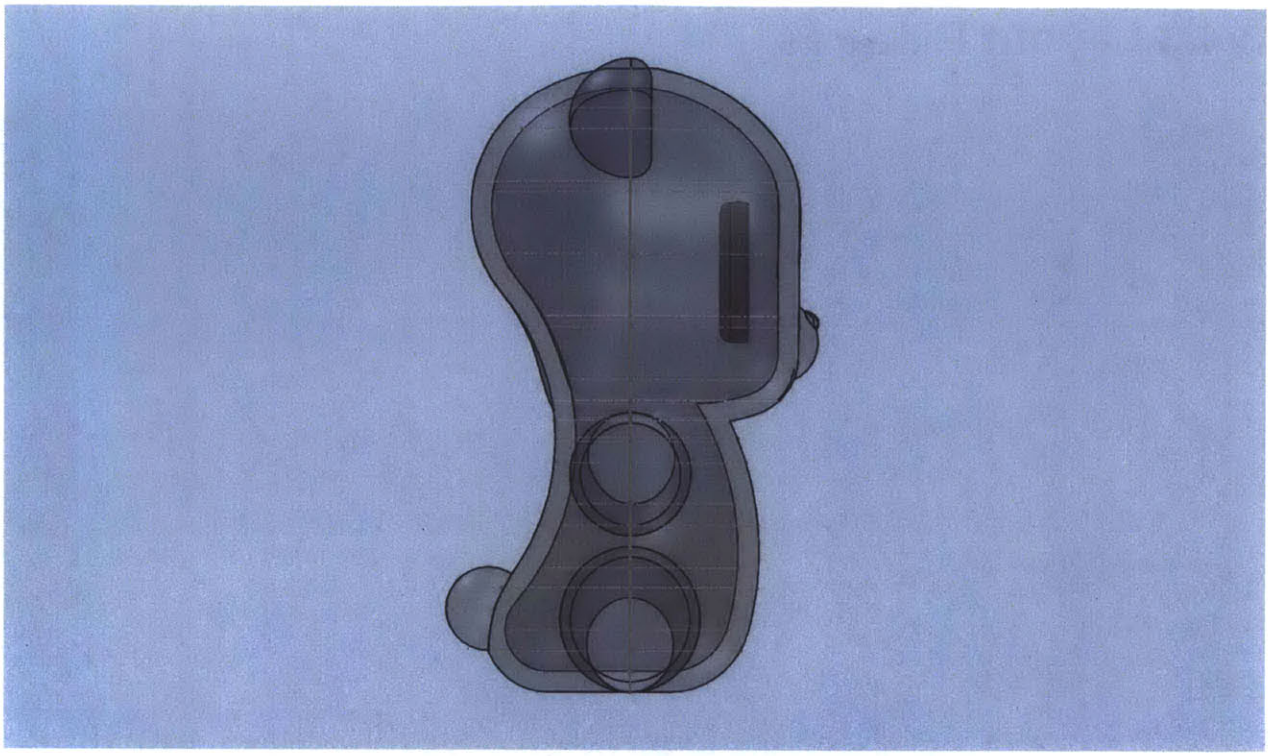


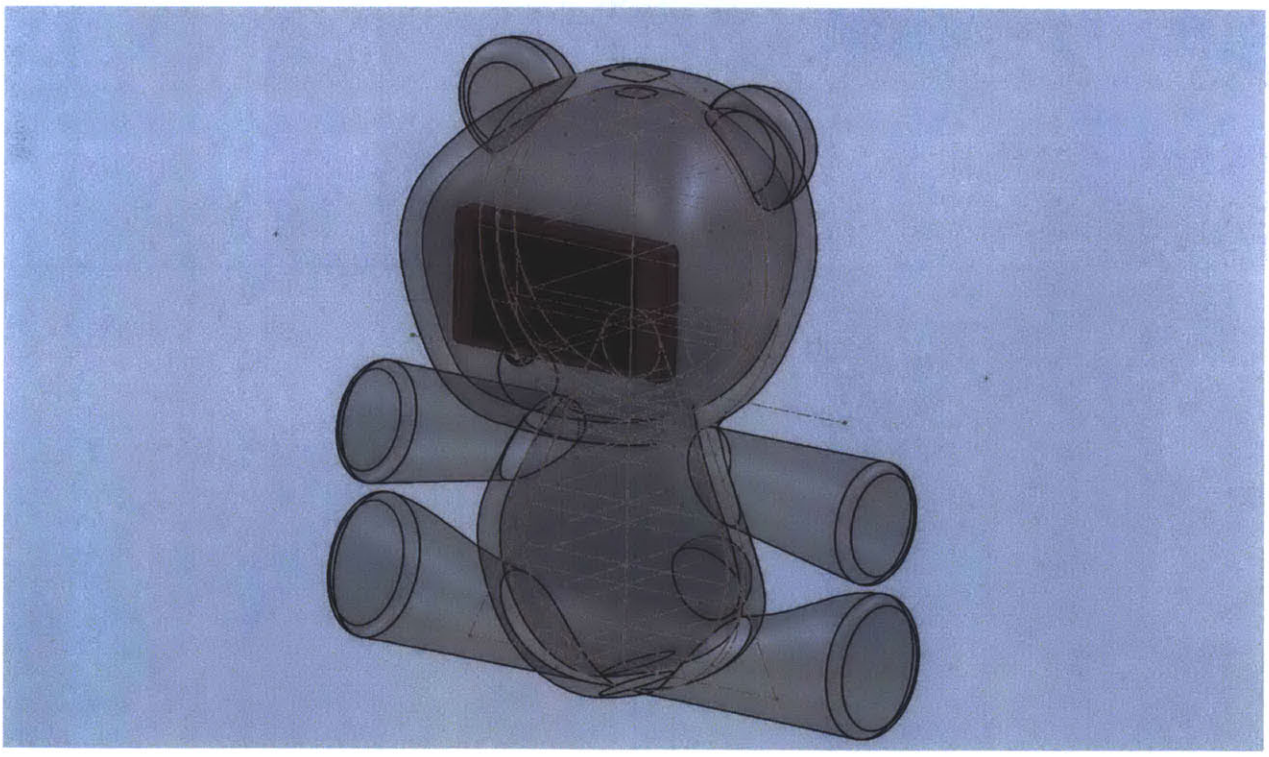


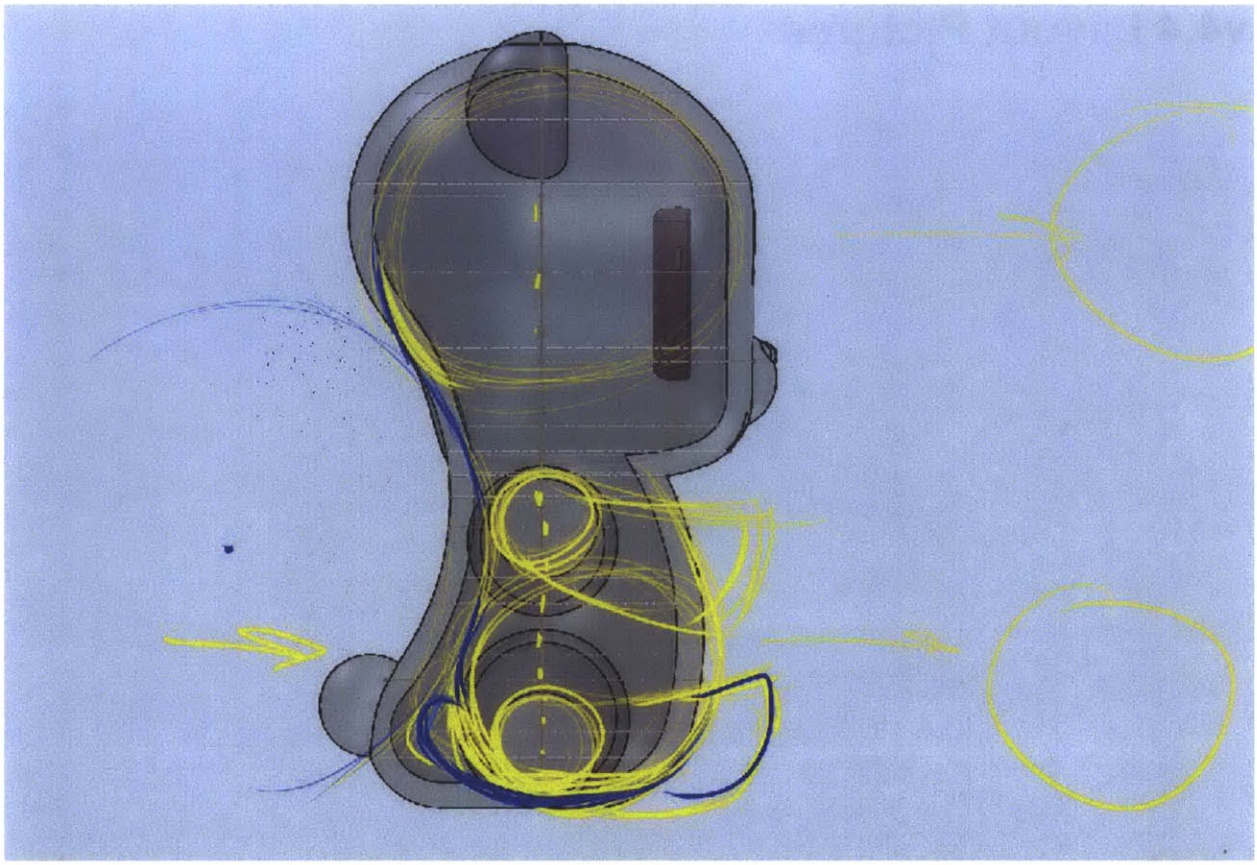


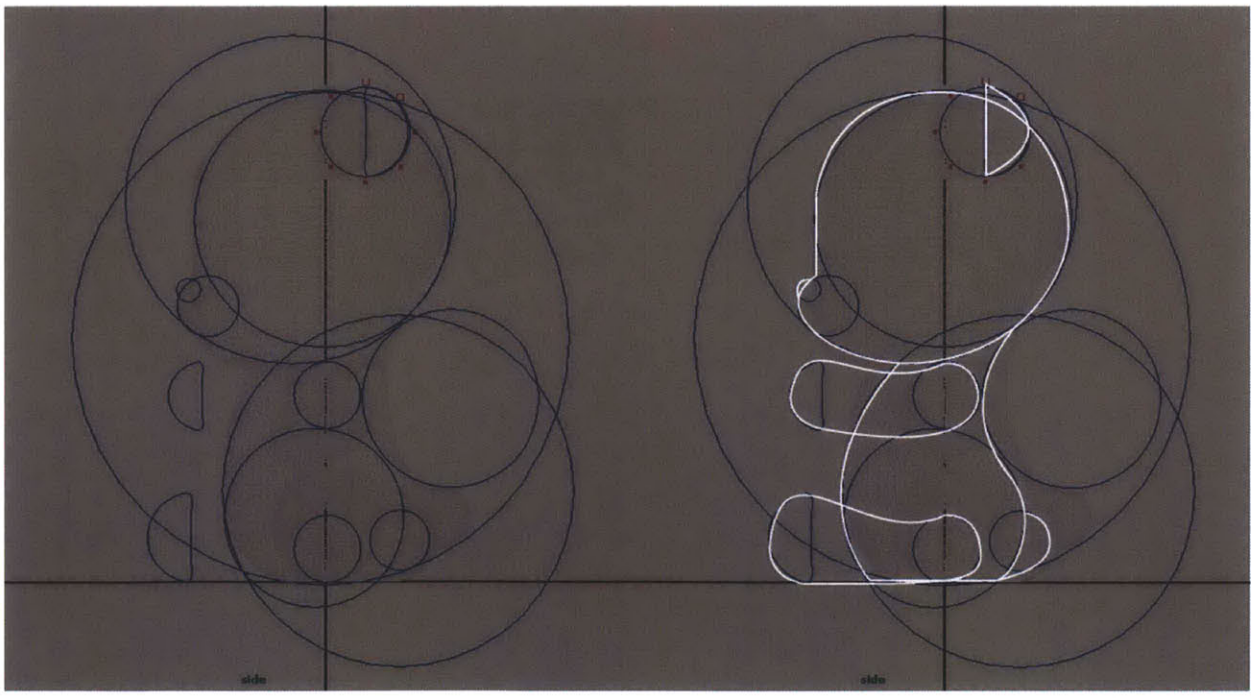
v4.3 Layout Pictures



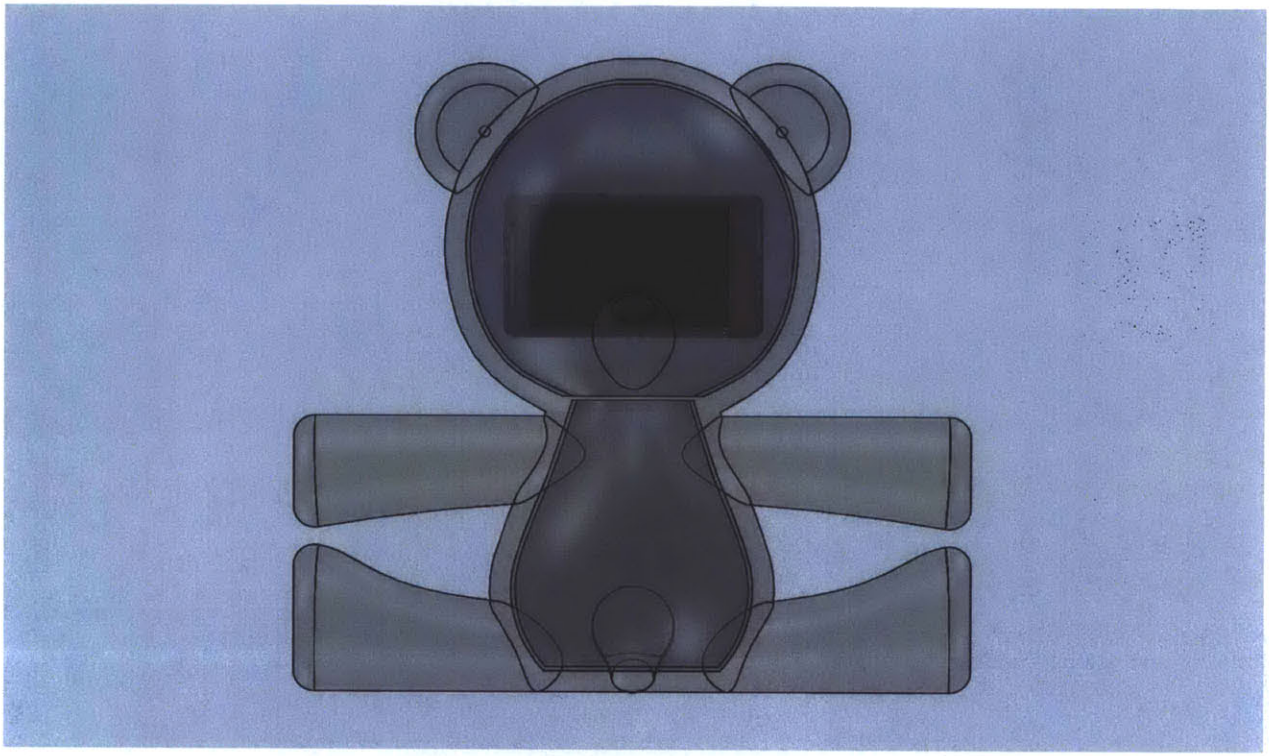


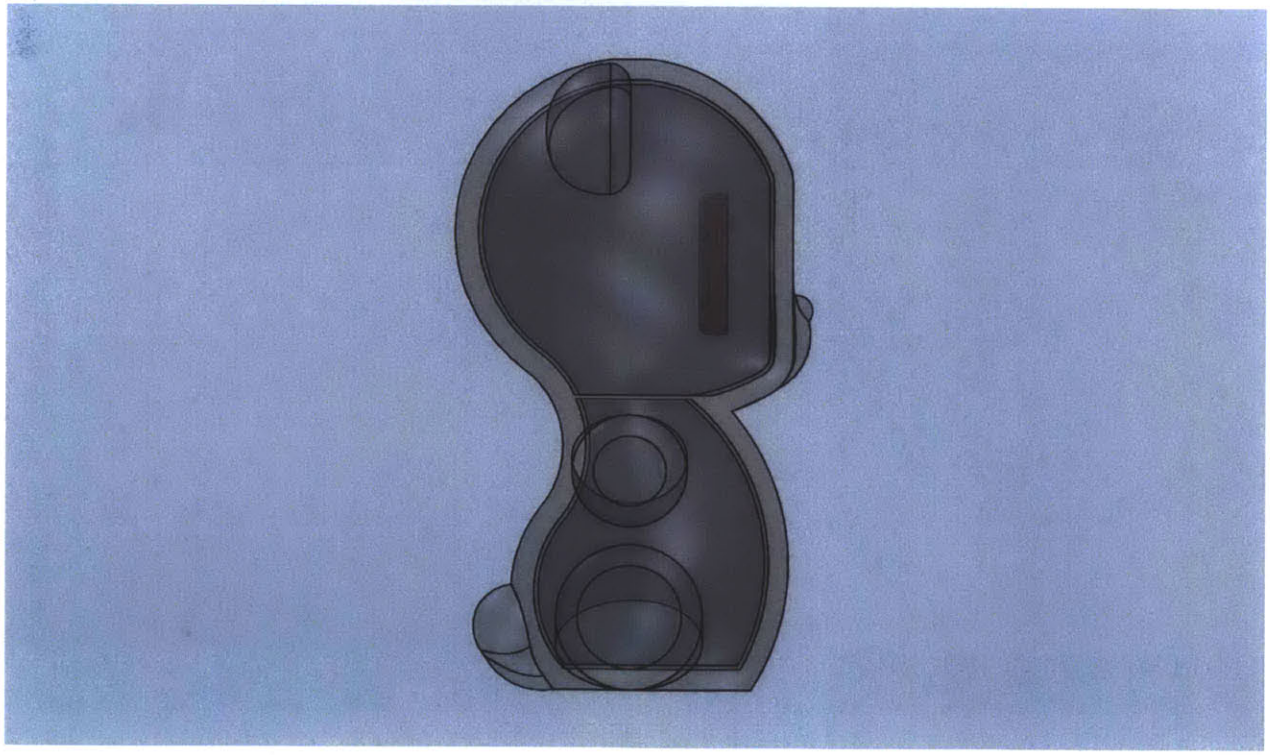


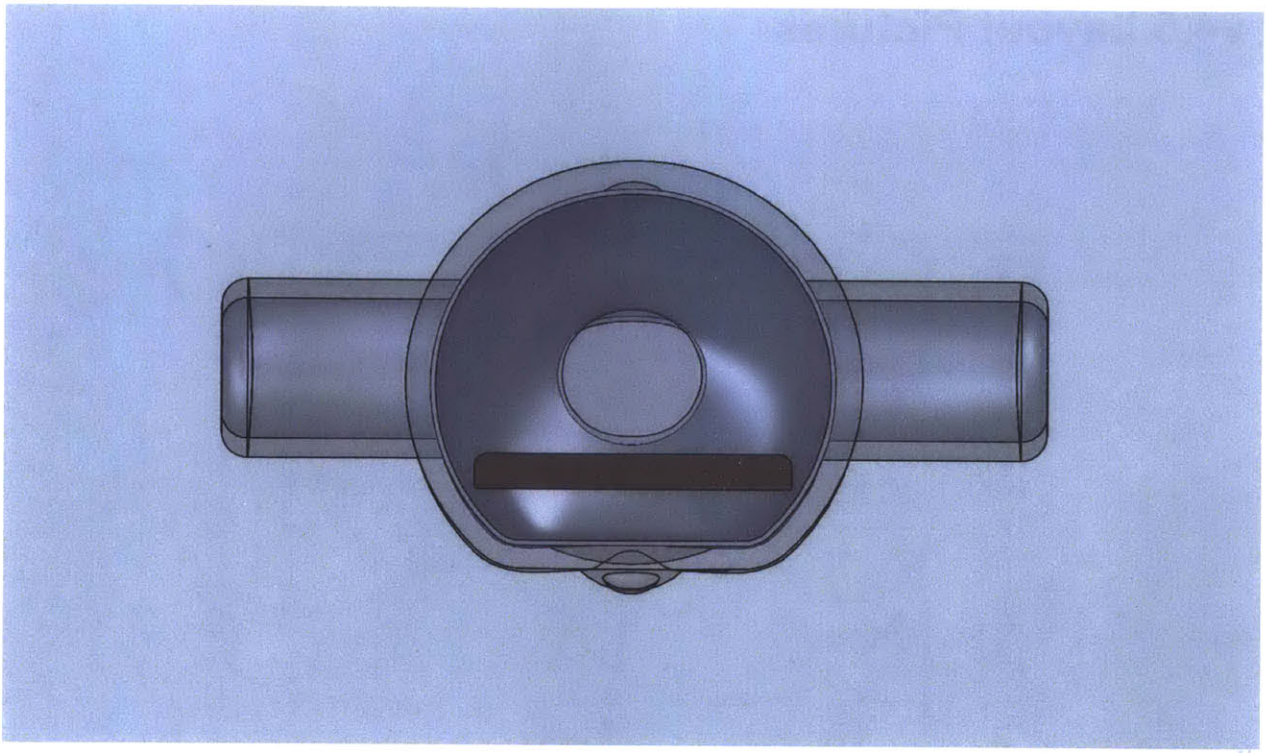


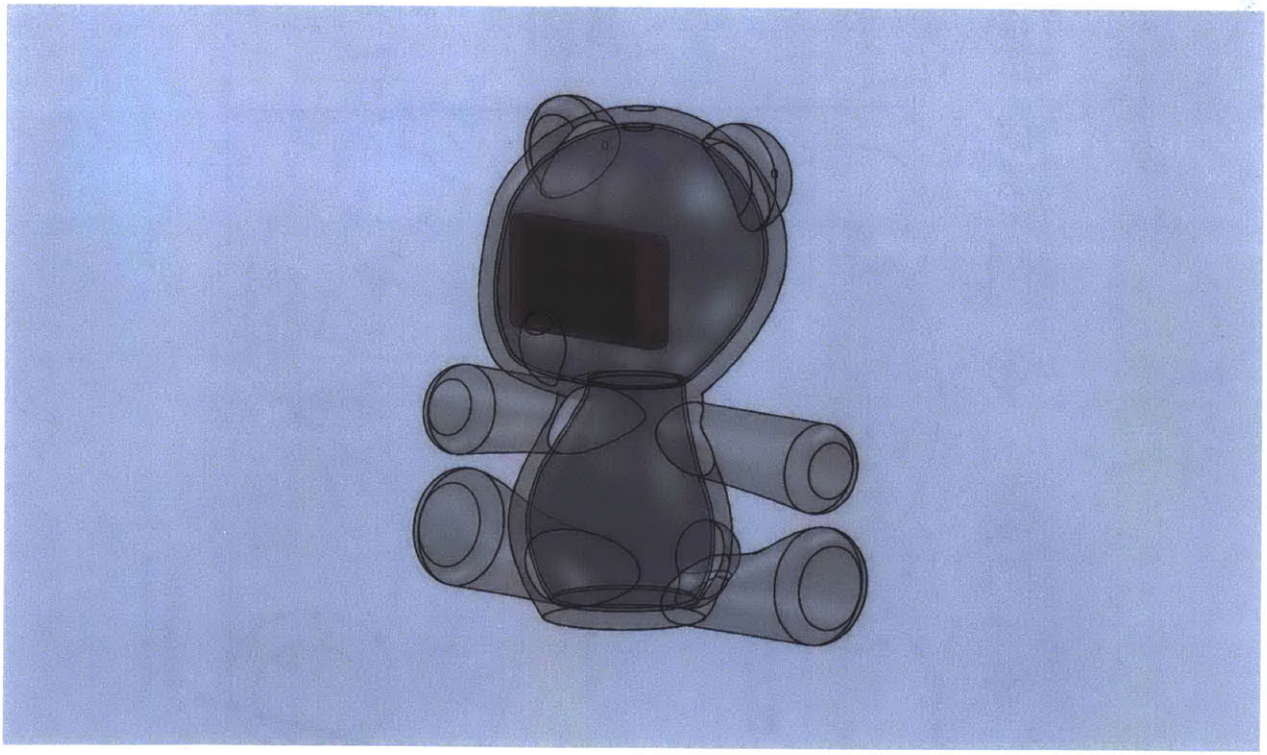


v4.4 Layout Pictures

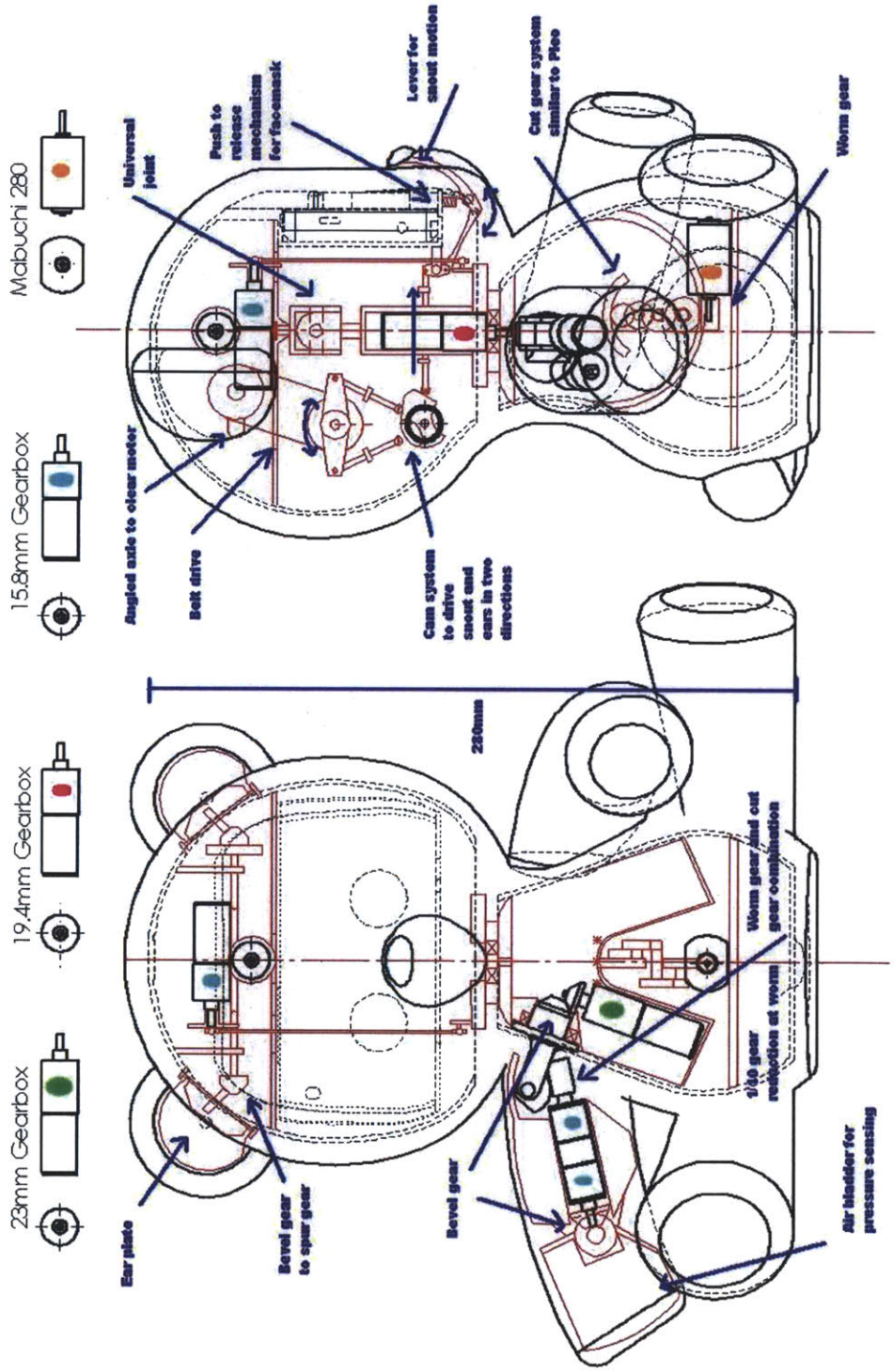


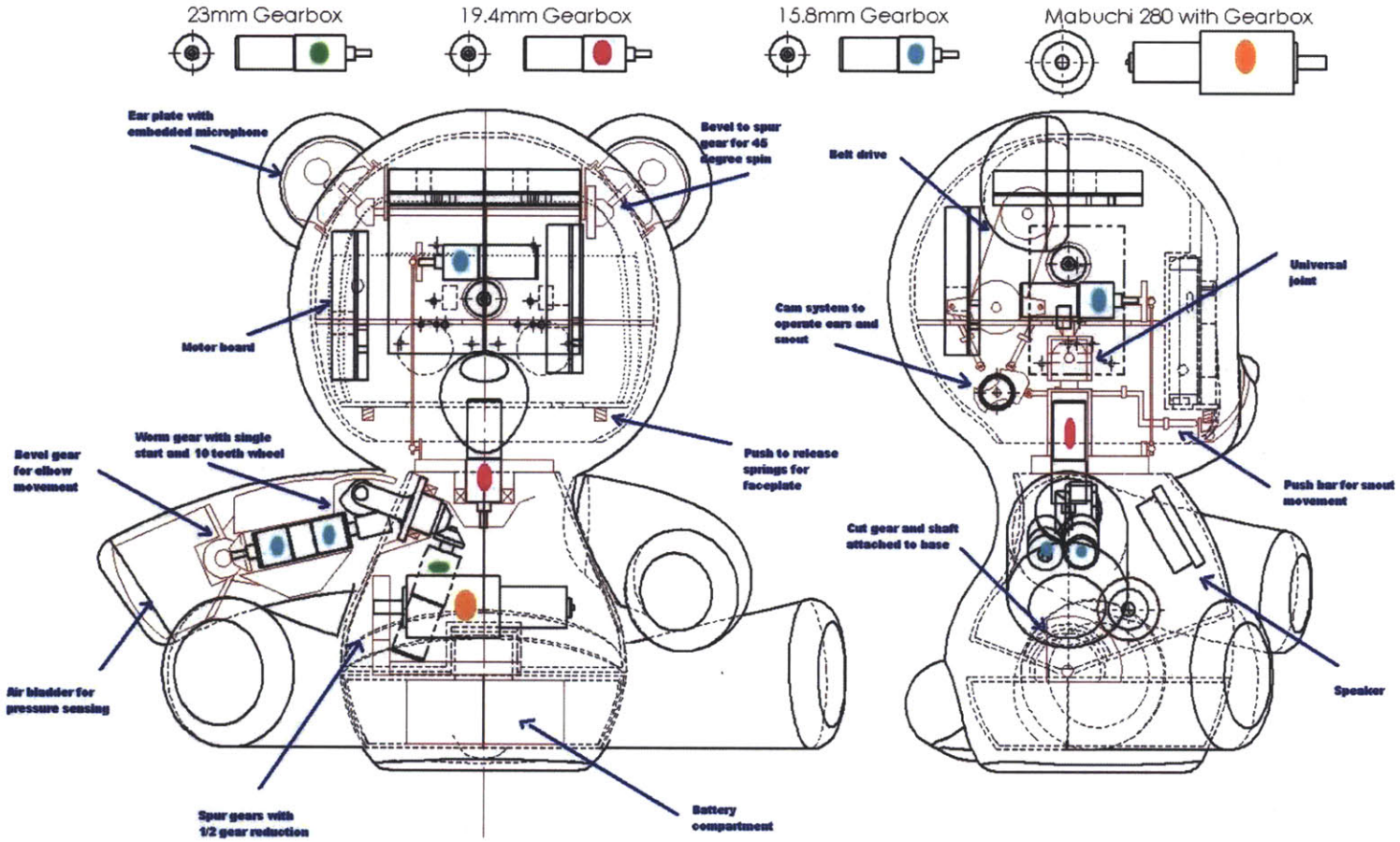


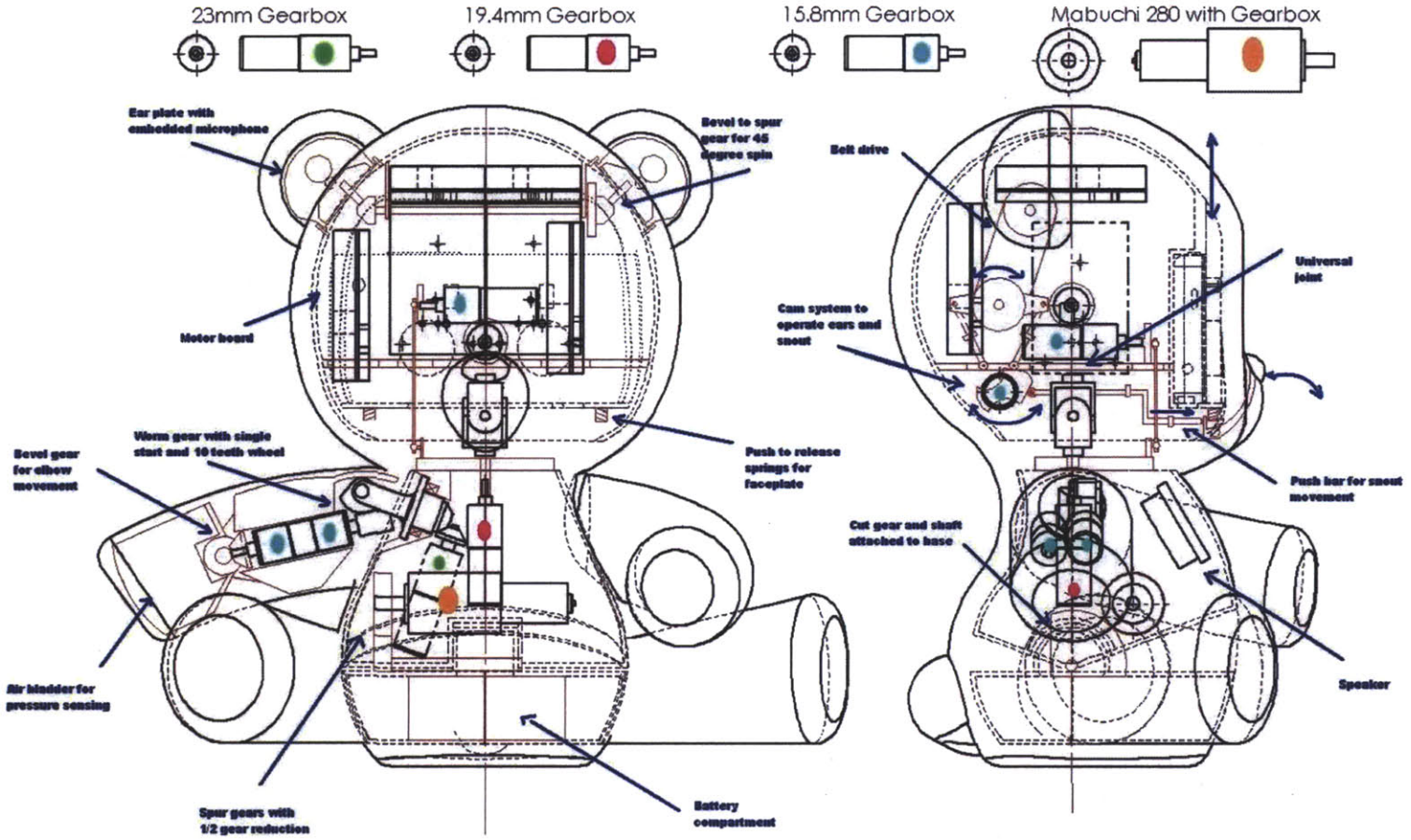


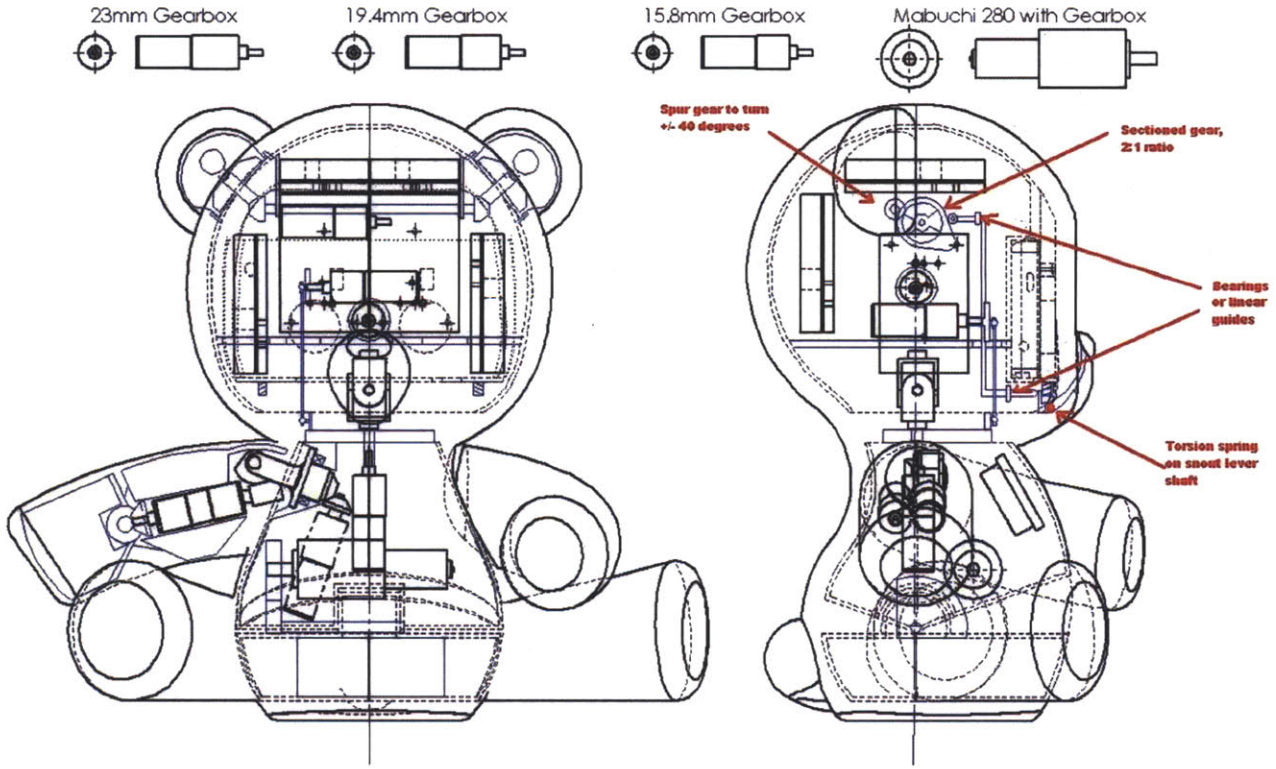


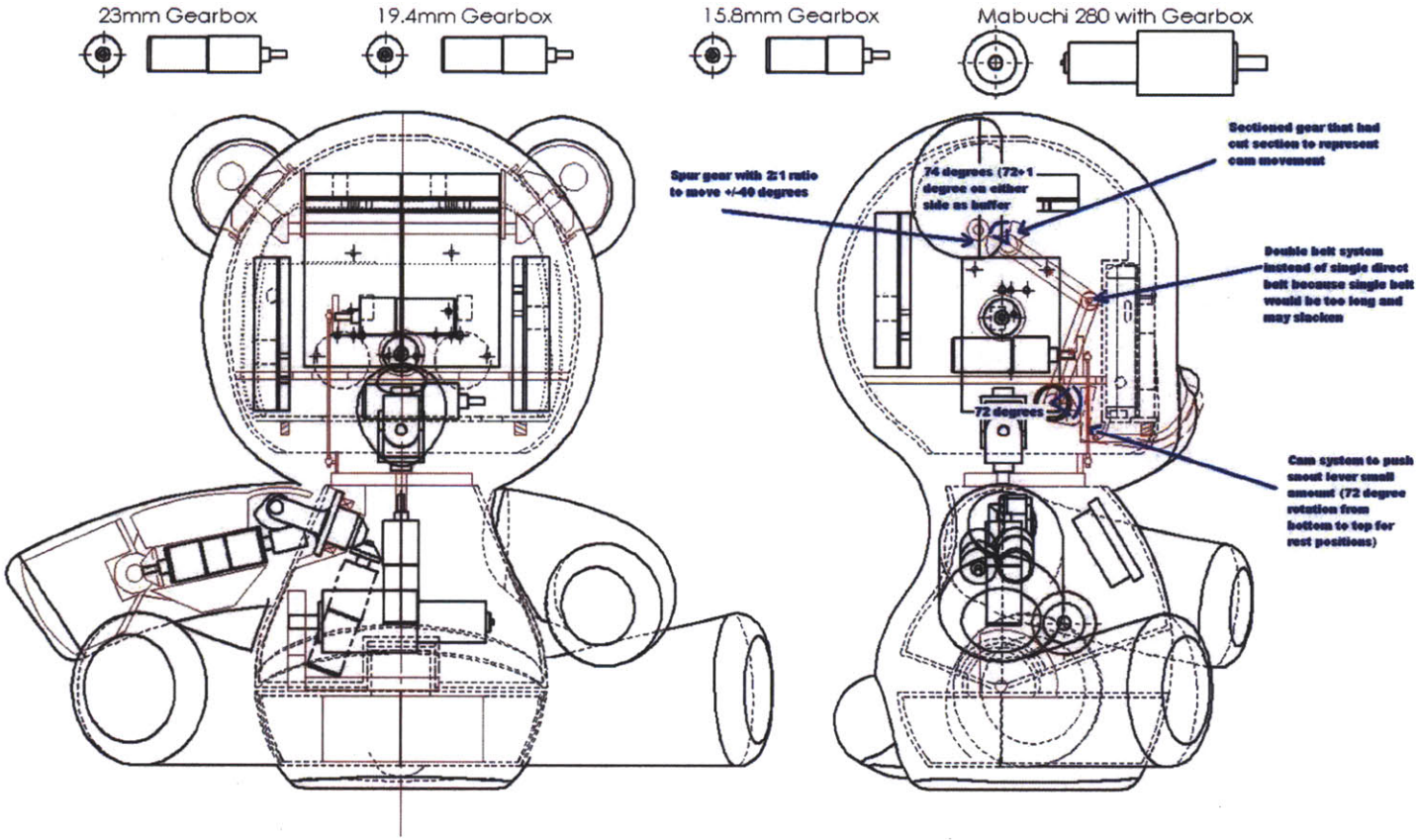
v4.5 Layout Pictures

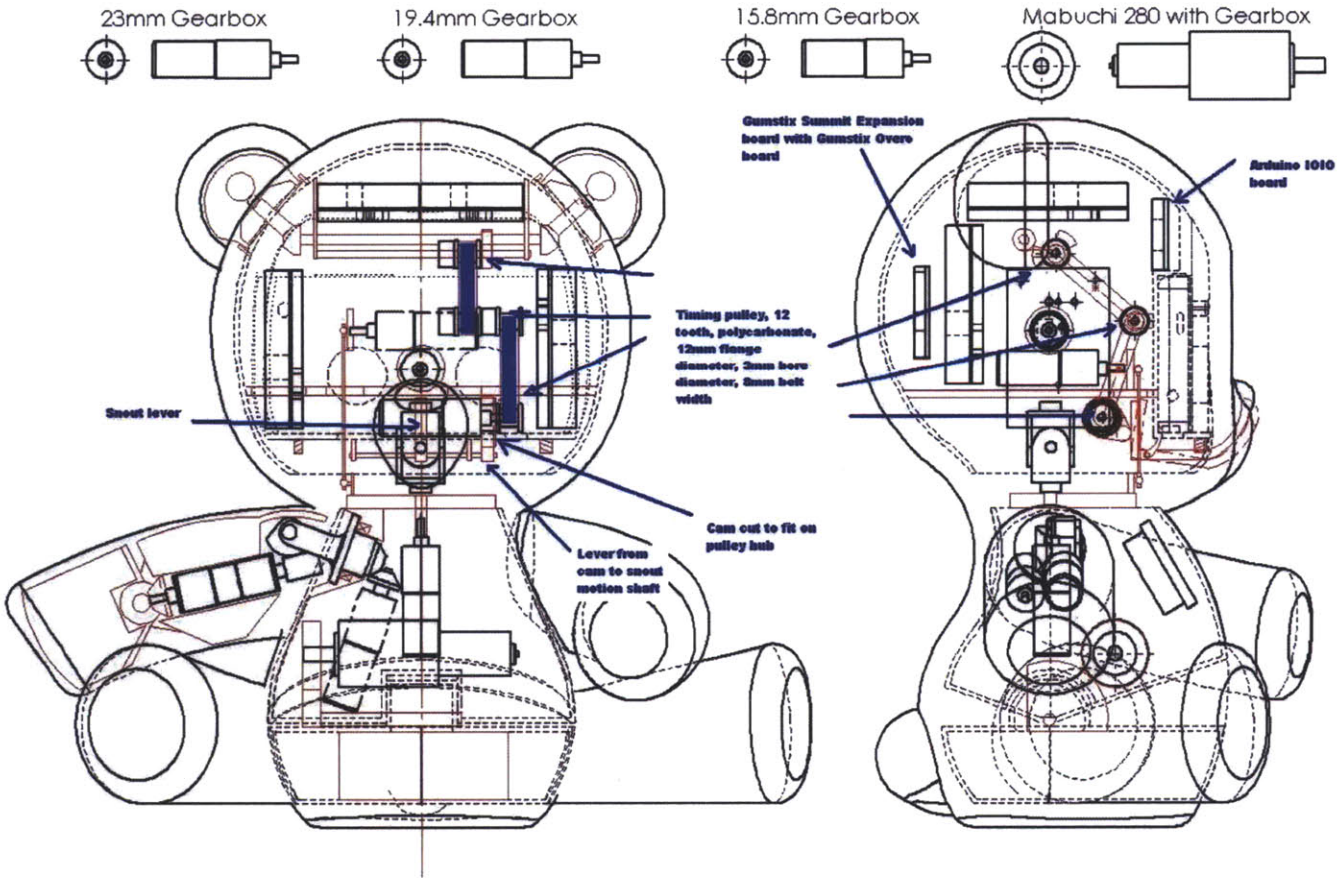



23mm Gearbox
19.4mm Gearbox
15.8mm Gearbox
Mabuchi 280 with Gearbox
Ear plate with embedded microphone
Bevel to spur gear for 45 degree spin
Motor board
Worm gear with single start and 10 teeth wheel
Bevel gear for elbow movement
Push to release springs for faceplate
Air bladder for pressure sensing
Spur gears with 1/2 gear reduction
Battery compartment
Belt drive
Universal joint
Cam system to operate ears and snout
Push bar for snout movement
Cut gear and shaft attached to base
Speaker

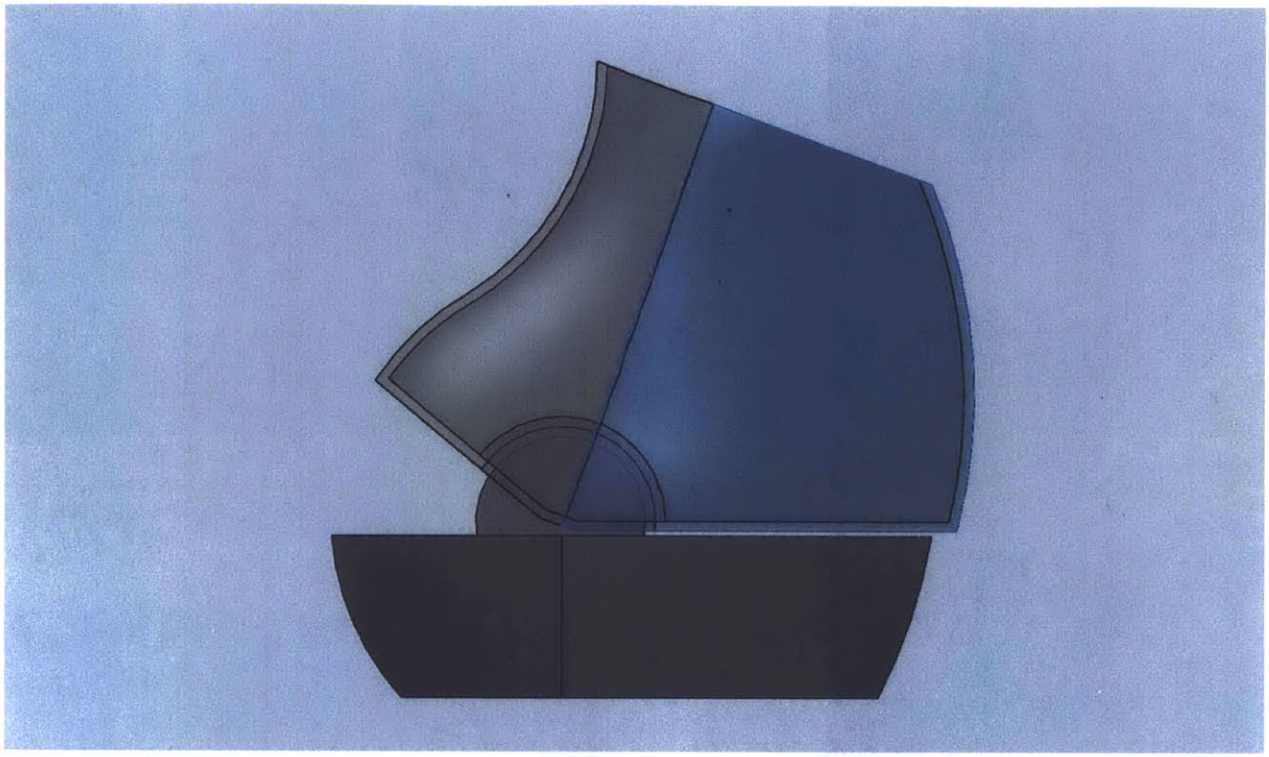

23mm Gearbox
19.4mm Gearbox
15.8mm Gearbox
Mabuchi 280 with Gearbox
Ear plate with embedded microphone
Bevel to spur gear for 45 degree spin
Motor board
Worm gear with single start and 10 tooth wheel
Bevel gear for elbow movement
Push to release springs for faceplate
Air bladder for pressure sensing
Spur gears with 1/2 gear reduction
Battery compartment
Belt drive
Universal joint
Cam system to operate ears and snout
Push bar for snout movement
Cut gear and shaft attached to base
Speaker

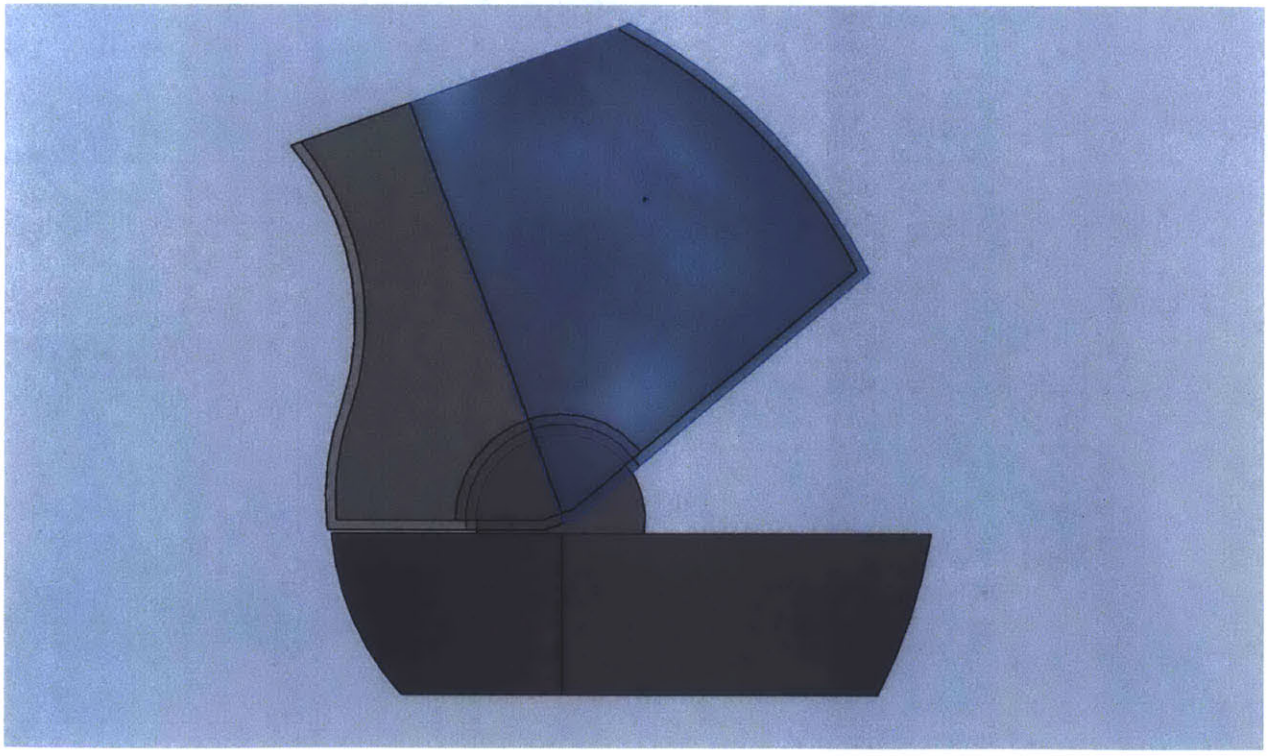

23mm Gearbox
19.4mm Gearbox
15.8mm Gearbox
Mabuchi 280 with Gearbox
Spur gear to turn +/- 40 degrees
Sectioned gear, 2:1 ratio
Bearings or linear guides
Torsion spring on snout lever shaft


23mm Gearbox
19.4mm Gearbox
15.8mm Gearbox
Mabuchi 280 with Gearbox
Spur gear with 2:1 ratio to move +/-40 degrees
74 degrees (72+1 degree on either side as buffer
Sectioned gear that had cut section to represent cam movement
Double belt system instead of single direct belt because single belt would be too long and may slacken
72 degrees
Cam system to push snout lever small amount (72 degree rotation from bottom to top for rest positions)


23mm Gearbox
19.4mm Gearbox
15.8mm Gearbox
Mabuchi 280 with Gearbox
Gumstix Summit Expansion board with Gumstix Overo board
Arduino 1010 board
Timing pulley, 12 tooth, polycarbonate, 12mm flange diameter, 3mm bore diameter, 3mm belt width
Snout lever
Cam cut to fit on pulley hub
Lever from cam to snout motion shaft

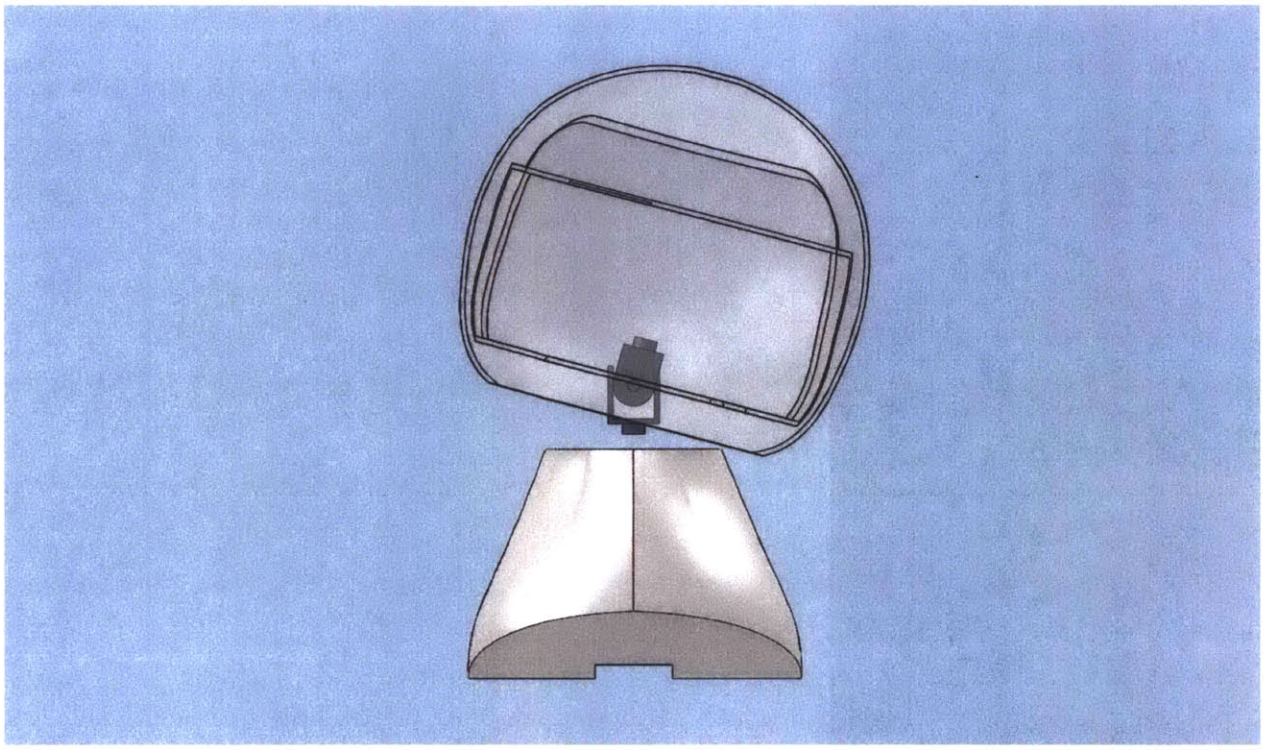
Waist Test Pictures

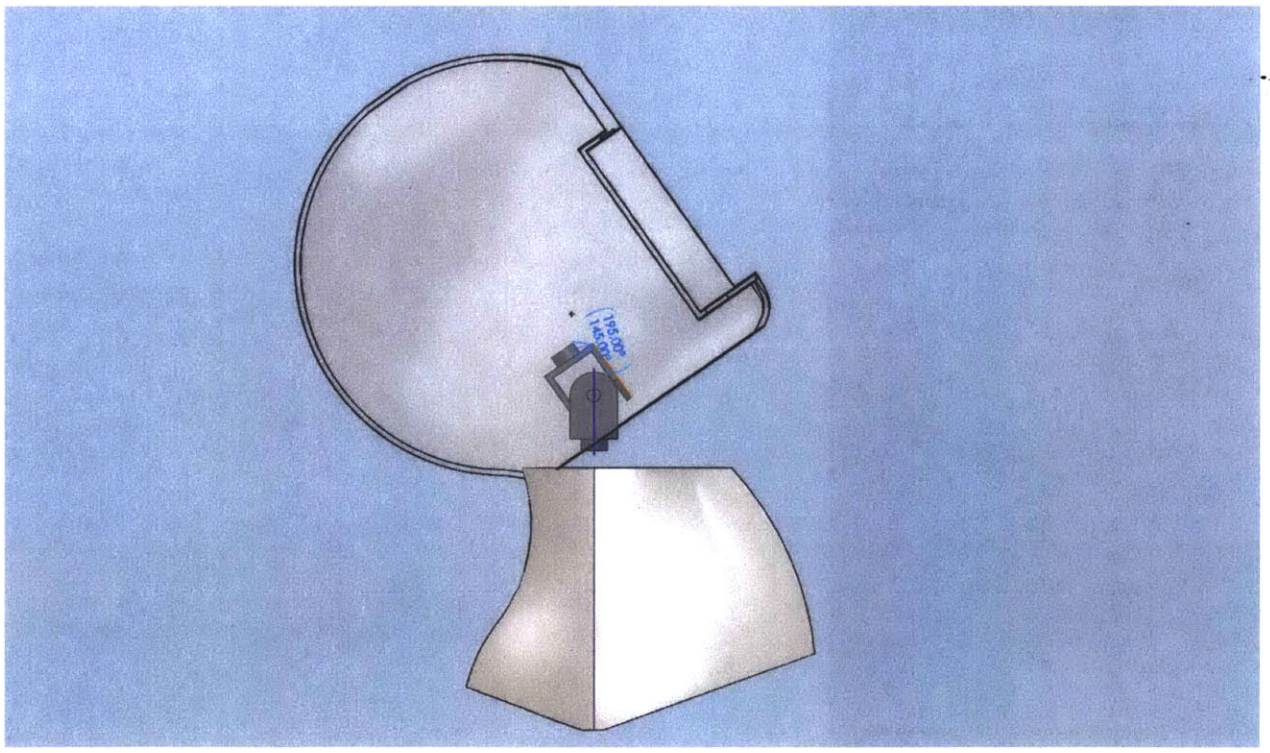


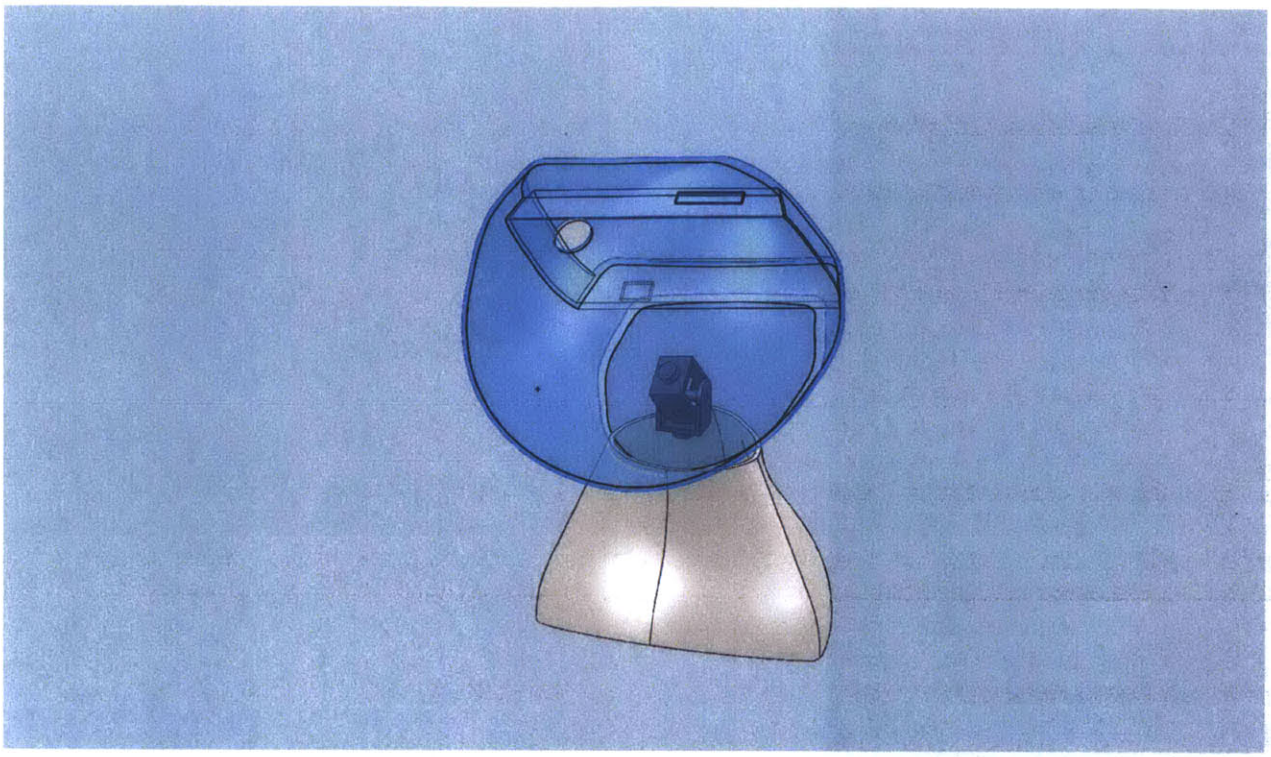


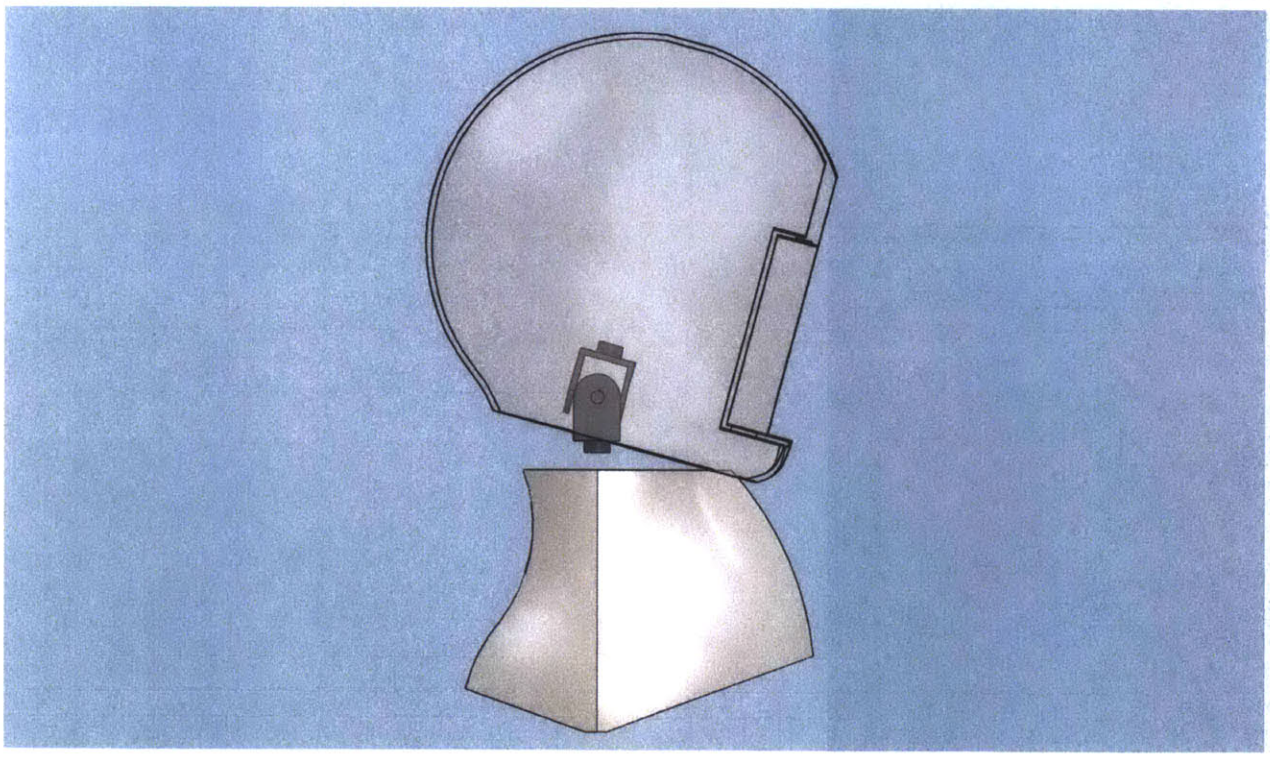
The following pictures depict a motion test of the waist at +/- 20 degrees for clearance.

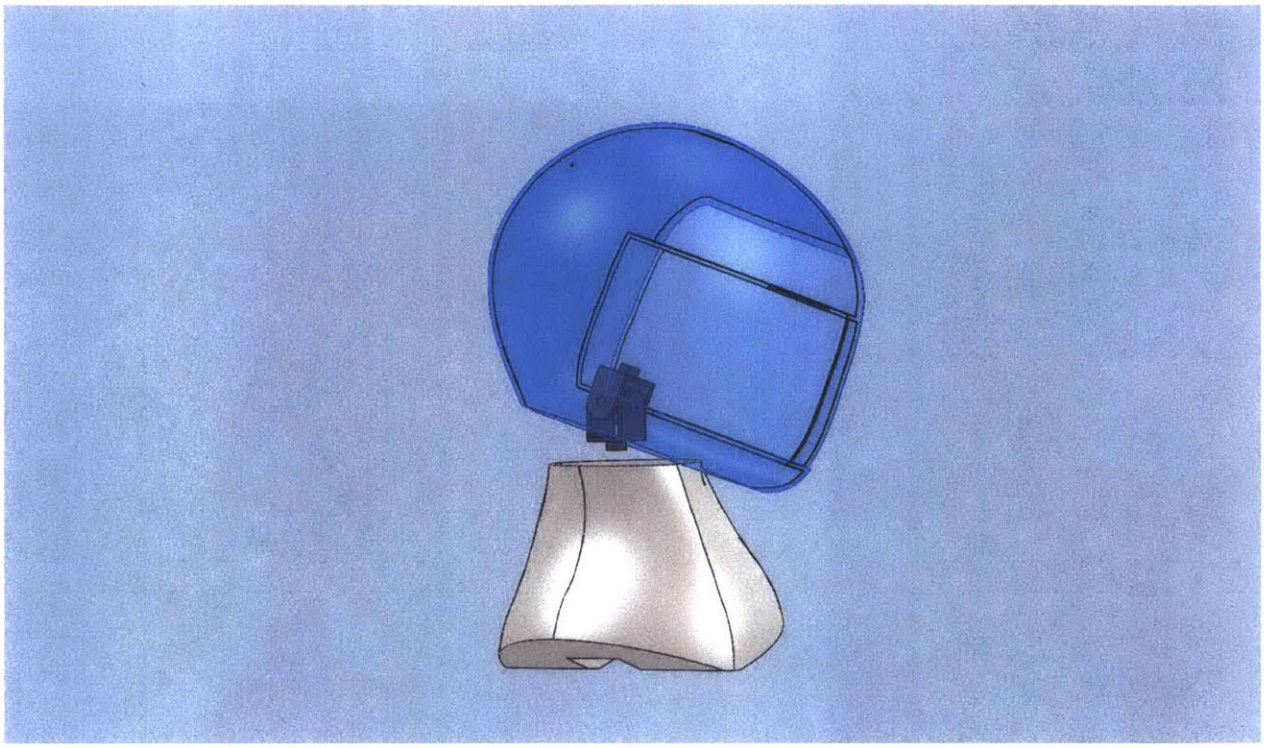
Neck Test Pictures





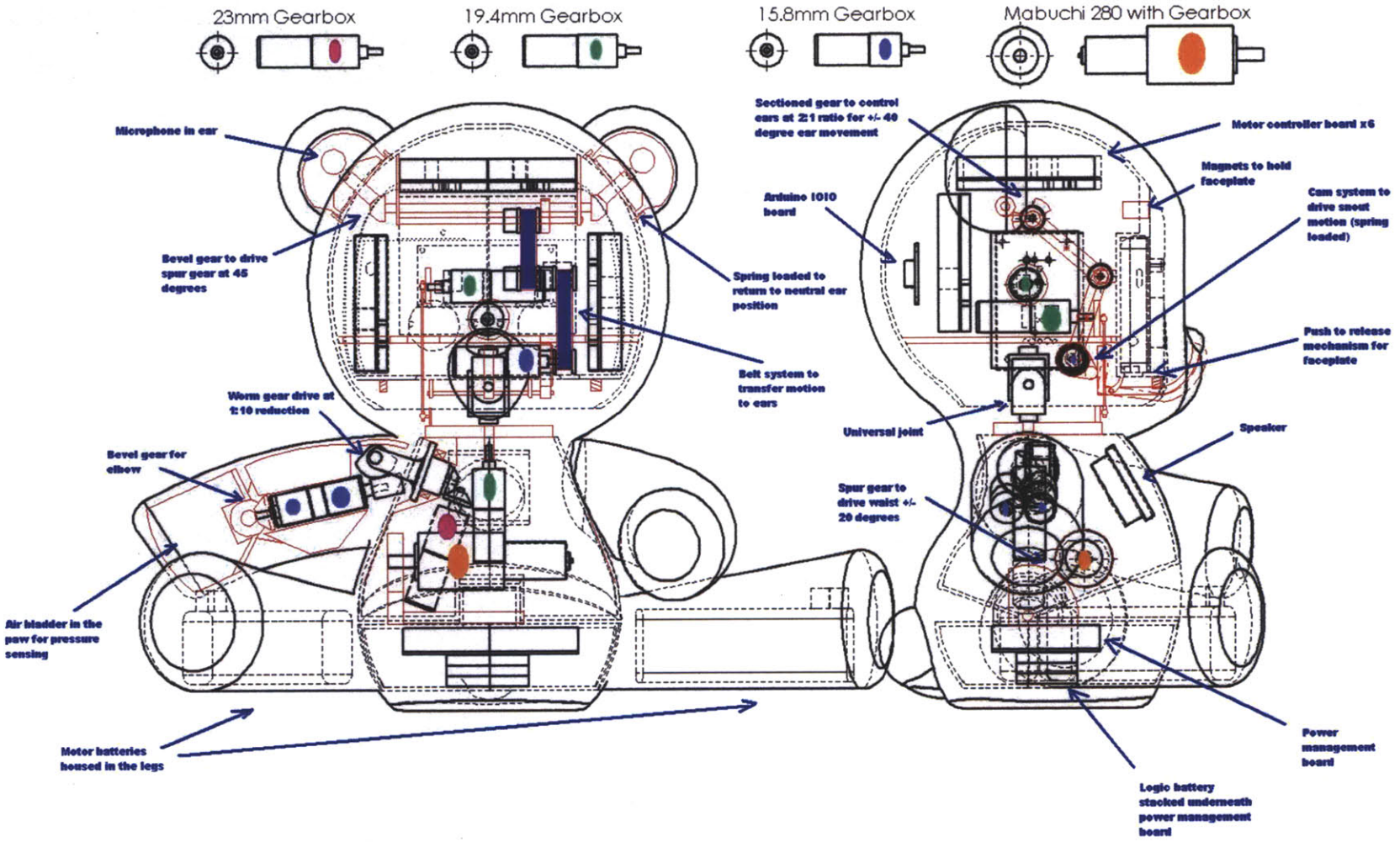






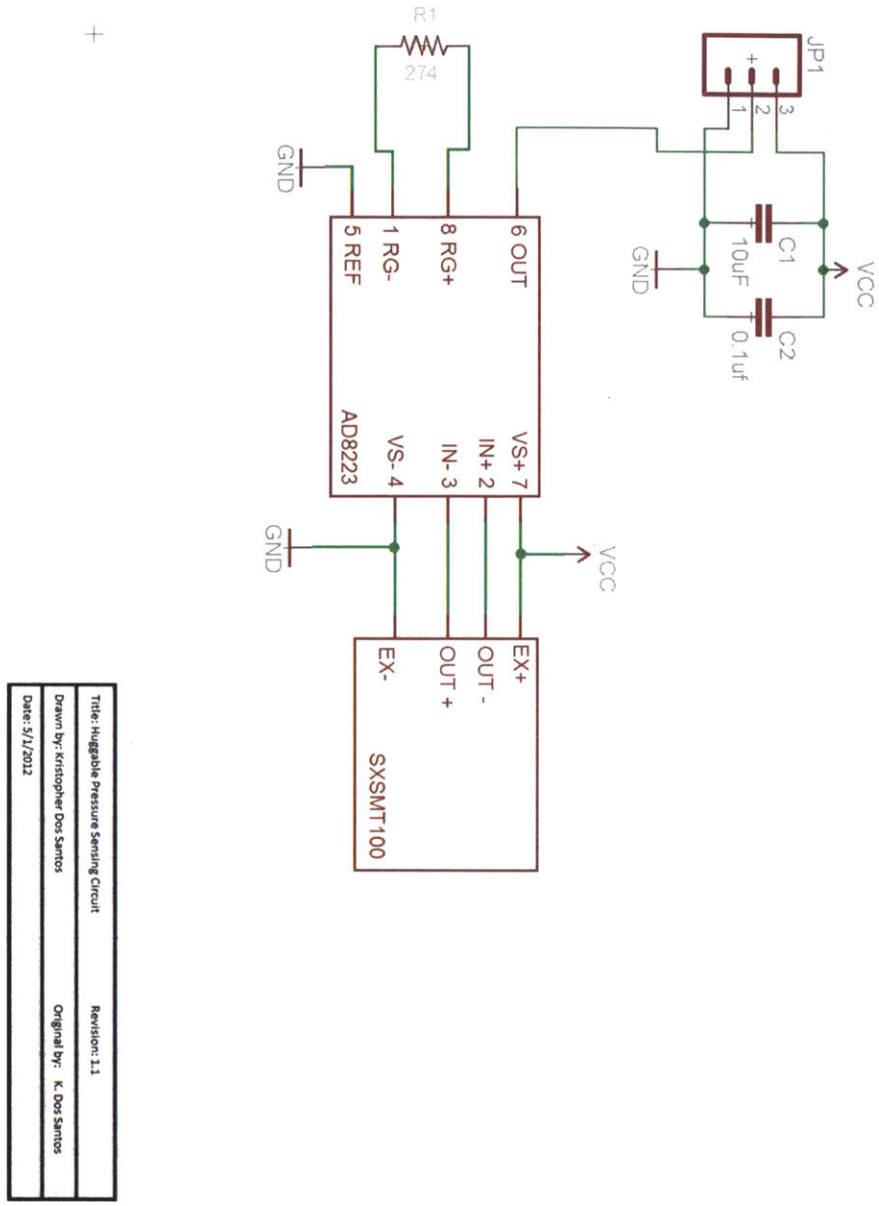
The following pictures depict a range of motion test for the neck at +/-15 degrees for left to right motion, and +35/-15 degrees for up to down motion. This was to make sure the head shell would not interfere with the body shell.

Appendix B: Final Mechanical Layout

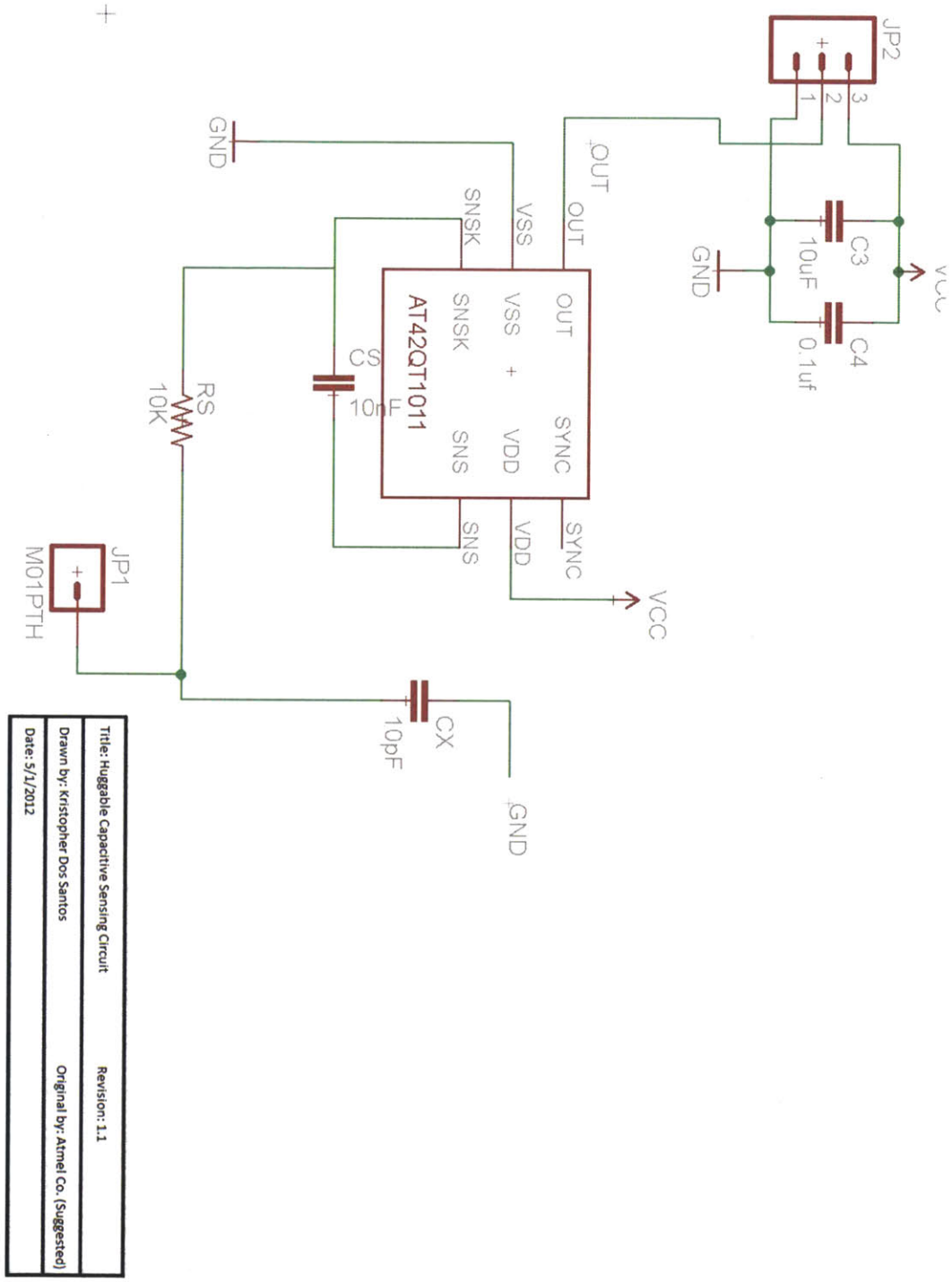

23mm Gearbox
19.4mm Gearbox
15.8mm Gearbox
Mabuchi 280 with Gearbox
Microphone in ear
Bevel gear to drive spur gear at 45 degrees
Worm gear drive at 1:10 reduction
Bevel gear for elbow
Air bladder in the paw for pressure sensing
Motor batteries housed in the legs
Spring loaded to return to neutral ear position
Belt system to transfer motion to ears
Sectioned gear to control ears at 2:1 ratio for +/- 40 degree ear movement
Arduino 1010 board
Universal joint
Spur gear to drive waist +/- 20 degrees
Motor controller board x6
Magnets to hold faceplate
Cam system to drive snout motion (spring loaded)
Push to release mechanism for faceplate
Speaker
Power management board
Logic battery stacked underneath power management board

Appendix C: Electrical Component Schematics

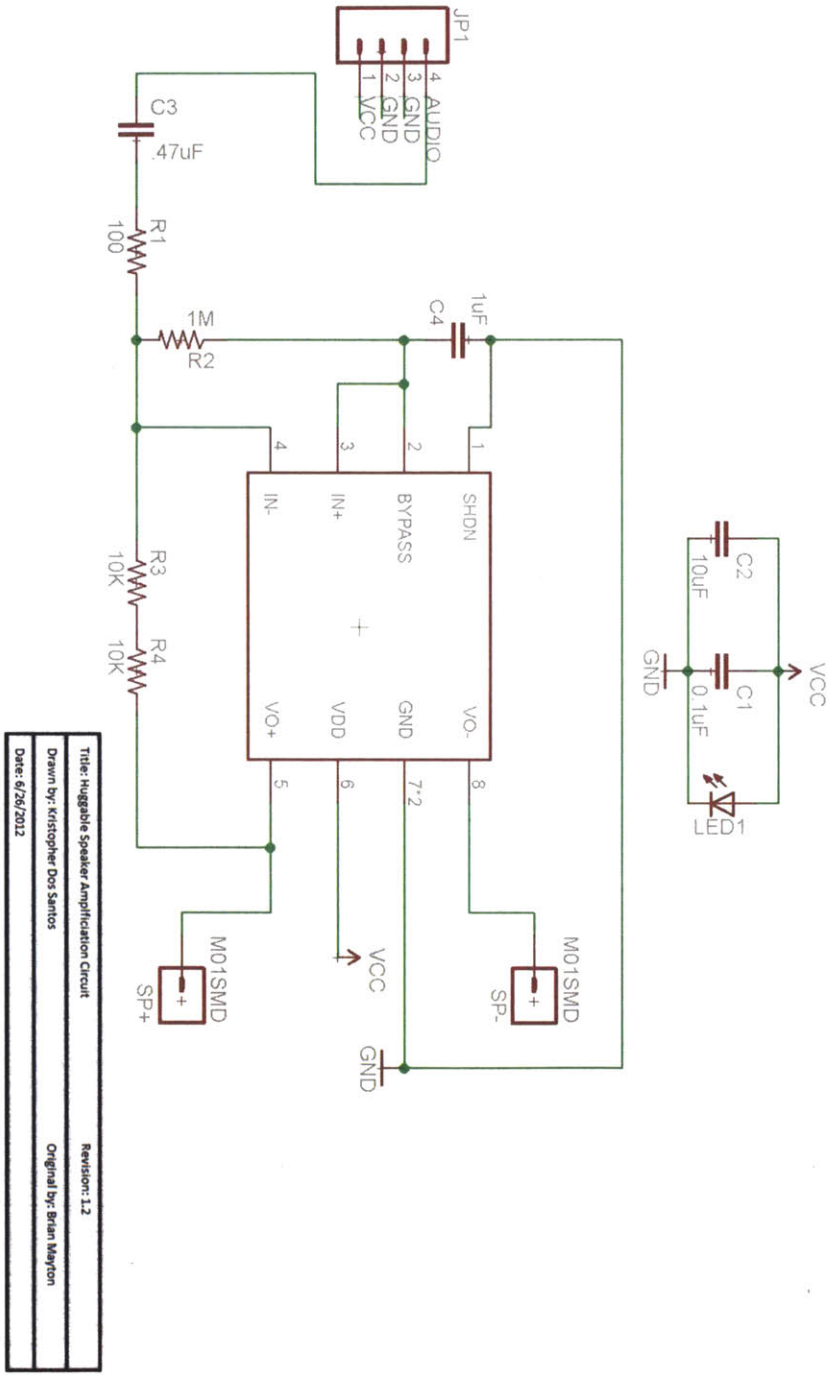
C.1 Pressure Sensing Circuit Schematic



C.2 Capacitive Sensing Circuit Schematic



C.3 Speaker Amp Circuit Schematic



Appendix D – Selected Source Code

D.1 .XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
    <MotorBoard
class="prg.interfaces.software.android.motor.AndroidIOIOSerialBus">
            <MiniBoards>
             <MiniBoard>
                 <id>1</id>
                 <fault_mode>OFF</fault_mode>
                 <Channels>
                     <Motor>
                         <channel>A</channel>
                         <pgain>150</pgain>
                         <dgain>30</dgain>
                         <igain>0</igain>
                         <deadband>0</deadband>
                         <maxstep>0</maxstep>
                         <output_divisor_pwr2>3</output_divisor_pwr2>

      <enable_rising_time_ms>2000</enable_rising_time_ms>
                         <direction>FLIPPED</direction>
                         <feedback_mode>POT_MODE</feedback_mode>
                         <control_mode>POSITION_MODE</control_mode>
                         <name>leftElbowBn</name>
                         <description>Left Elbow</description>
                         <center_ticks>490</center_ticks>
                         <max_ticks>490</max_ticks>
                         <min_ticks>280</min_ticks>
                         <ticks_per_unit>-150</ticks_per_unit>
                                 <current_limit>2000</current_limit>
                         <slow_enable_const>2000</slow_enable_const>
                     </Motor>
                     <Motor>
                         <channel>B</channel>
                         <pgain>100</pgain>
                         <dgain>30</dgain>
                         <igain>4</igain>
                         <deadband>0</deadband>
                         <maxstep>0</maxstep>
                         <output_divisor_pwr2>3</output_divisor_pwr2>

      <enable_rising_time_ms>2000</enable_rising_time_ms>
                         <direction>REGULAR</direction>
                         <feedback_mode>POT_MODE</feedback_mode>
                         <control_mode>POSITION_MODE</control_mode>
                         <name>leftShoulderUpDownBn</name>
                         <description>Left Shoulder</description>
                         <center_ticks>740</center_ticks>
                         <max_ticks>740</max_ticks>
                         <min_ticks>490</min_ticks>
                         <ticks_per_unit>-500</ticks_per_unit>
```

```
                        <current_limit>2000</current_limit>
                    <slow_enable_const>2000</slow_enable_const>
                </Motor>
            </Channels>
        </MiniBoard>
            <MiniBoard>
            <id>2</id>
            <fault_mode>OFF</fault_mode>
            <Channels>
                <Motor>
                    <channel>A</channel>
                    <pgain>80</pgain>
                    <dgain>15</dgain>
                    <igain>2</igain>
                    <deadband>0</deadband>
                    <maxstep>0</maxstep>
                    <output_divisor_pwr2>3</output_divisor_pwr2>

<enable_rising_time_ms>2000</enable_rising_time_ms>
                    <direction>FLIPPED</direction>
                    <feedback_mode>POT_MODE</feedback_mode>
                    <control_mode>POSITION_MODE</control_mode>
                    <name>leftShoulderRotateBn</name>
                    <description>Left Arm</description>
                    <center_ticks>430</center_ticks>
                    <max_ticks>680</max_ticks>
                    <min_ticks>430</min_ticks>
                    <ticks_per_unit>150</ticks_per_unit>
                            <current_limit>2000</current_limit>
                    <slow_enable_const>2000</slow_enable_const>
                </Motor>
                <Motor>
                    <channel>B</channel>
                    <pgain>80</pgain>
                    <dgain>30</dgain>
                    <igain>0</igain>
                    <deadband>0</deadband>
                    <maxstep>0</maxstep>
                    <output_divisor_pwr2>3</output_divisor_pwr2>

<enable_rising_time_ms>2000</enable_rising_time_ms>
                    <direction>FLIPPED</direction>
                    <feedback_mode>POT_MODE</feedback_mode>
                    <control_mode>POSITION_MODE</control_mode>
                    <name>headTilt</name>
                    <description>Head tilt</description>
                    <center_ticks>580</center_ticks>
                    <max_ticks>830</max_ticks>
                    <min_ticks>480</min_ticks>
                    <ticks_per_unit>-643</ticks_per_unit>
                            <current_limit>2000</current_limit>
                    <slow_enable_const>2000</slow_enable_const>
```

```
                </Motor>
            </Channels>
        </MiniBoard>
        <MiniBoard>
            <id>3</id>
            <fault_mode>OFF</fault_mode>
            <Channels>
                <Motor>
                    <channel>A</channel>
                    <pgain>100</pgain>
                    <dgain>30</dgain>
                    <igain>2</igain>
                    <deadband>0</deadband>
                    <maxstep>0</maxstep>
                    <output_divisor_pwr2>3</output_divisor_pwr2>

<enable_rising_time_ms>2000</enable_rising_time_ms>
                    <direction>REGULAR</direction>
                    <feedback_mode>POT_MODE</feedback_mode>
                    <control_mode>POSITION_MODE</control_mode>
                    <name>neckRotateBn</name>
                    <description>Head Rotate</description>
                    <center_ticks>770</center_ticks>
                    <max_ticks>900</max_ticks>
                    <min_ticks>640</min_ticks>
                    <ticks_per_unit>1334</ticks_per_unit>
                            <current_limit>2000</current_limit>
                    <slow_enable_const>2000</slow_enable_const>
                </Motor>
                <Motor>
                    <channel>B</channel>
                    <pgain>50</pgain>
                    <dgain>100</dgain>
                    <igain>0</igain>
                    <deadband>0</deadband>
                    <maxstep>0</maxstep>
                    <output_divisor_pwr2>3</output_divisor_pwr2>

<enable_rising_time_ms>2000</enable_rising_time_ms>
                    <direction>REGULAR</direction>
                    <feedback_mode>POT_MODE</feedback_mode>
                    <control_mode>POSITION_MODE</control_mode>
                    <name>snoutUpDownBn</name>
                    <description>Muzzle Wiggle</description>
                    <center_ticks>1023</center_ticks>
                    <max_ticks>1023</max_ticks>
                    <min_ticks>455</min_ticks>
                    <ticks_per_unit>-2000</ticks_per_unit>
                            <current_limit>2000</current_limit>
                    <slow_enable_const>2000</slow_enable_const>
                </Motor>
            </Channels>
```

```
        </MiniBoard>
        <MiniBoard>
            <id>4</id>
            <fault_mode>OFF</fault_mode>
            <Channels>
                <Motor>
                    <channel>A</channel>
                    <pgain>120</pgain>
                    <dgain>30</dgain>
                    <igain>5</igain>
                    <deadband>0</deadband>
                    <maxstep>0</maxstep>
                    <output_divisor_pwr2>3</output_divisor_pwr2>

<enable_rising_time_ms>2000</enable_rising_time_ms>
                    <direction>FLIPPED</direction>
                    <feedback_mode>POT_MODE</feedback_mode>
                    <control_mode>POSITION_MODE</control_mode>
                    <name>rightEarFlapBn</name>
                    <description>Ear Wiggle</description>
                    <center_ticks>615</center_ticks>
                    <max_ticks>680</max_ticks>
                    <min_ticks>500</min_ticks>
                    <ticks_per_unit>1000</ticks_per_unit>
                            <current_limit>2000</current_limit>
                    <slow_enable_const>2000</slow_enable_const>
                </Motor>
                <Motor>
                    <channel>B</channel>
                    <pgain>100</pgain>
                    <dgain>30</dgain>
                    <igain>0</igain>
                    <deadband>0</deadband>
                    <maxstep>0</maxstep>
                    <output_divisor_pwr2>3</output_divisor_pwr2>

<enable_rising_time_ms>2000</enable_rising_time_ms>
                    <direction>FLIPPED</direction>
                    <feedback_mode>POT_MODE</feedback_mode>
                    <control_mode>POSITION_MODE</control_mode>
                    <name>hipBn</name>
                    <description>Waist Bend</description>
                    <center_ticks>890</center_ticks>
                    <max_ticks>915</max_ticks>
                    <min_ticks>850</min_ticks>
                    <ticks_per_unit>-153</ticks_per_unit>
                            <current_limit>2000</current_limit>
                    <slow_enable_const>2000</slow_enable_const>
                </Motor>
            </Channels>
        </MiniBoard>
        <MiniBoard>
```

```
            <id>5</id>
            <fault_mode>OFF</fault_mode>
            <Channels>
                <Motor>
                    <channel>A</channel>
                    <pgain>80</pgain>
                    <dgain>15</dgain>
                    <igain>2</igain>
                    <deadband>0</deadband>
                    <maxstep>0</maxstep>
                    <output_divisor_pwr2>3</output_divisor_pwr2>

<enable_rising_time_ms>2000</enable_rising_time_ms>
                    <direction>FLIPPED</direction>
                    <feedback_mode>POT_MODE</feedback_mode>
                    <control_mode>POSITION_MODE</control_mode>
                    <name>rightShoulderRotateBn</name>
                    <description>Right Arm</description>
                    <center_ticks>750</center_ticks>
                    <max_ticks>750</max_ticks>
                    <min_ticks>630</min_ticks>
                    <ticks_per_unit>-150</ticks_per_unit>
                            <current_limit>2000</current_limit>
                    <slow_enable_const>2000</slow_enable_const>
                </Motor>
                <Motor>
                    <channel>B</channel>
                    <pgain>70</pgain>
                    <dgain>30</dgain>
                    <igain>0</igain>
                    <deadband>0</deadband>
                    <maxstep>0</maxstep>
                    <output_divisor_pwr2>3</output_divisor_pwr2>

<enable_rising_time_ms>2000</enable_rising_time_ms>
                    <direction>FLIPPED</direction>
                    <feedback_mode>POT_MODE</feedback_mode>
                    <control_mode>POSITION_MODE</control_mode>
                    <name>headUpDown</name>
                    <description>Head Nod</description>
                    <center_ticks>593</center_ticks>
                    <max_ticks>816</max_ticks>
                    <min_ticks>370</min_ticks>
                    <ticks_per_unit>-463</ticks_per_unit>
                            <current_limit>2000</current_limit>
                    <slow_enable_const>2000</slow_enable_const>
                </Motor>
            </Channels>
        </MiniBoard>
        <MiniBoard>
            <id>6</id>
            <fault_mode>OFF</fault_mode>
```

```
                <Channels>
                    <Motor>
                        <channel>A</channel>
                        <pgain>100</pgain>
                        <dgain>30</dgain>
                        <igain>0</igain>
                        <deadband>0</deadband>
                        <maxstep>0</maxstep>
                        <output_divisor_pwr2>3</output_divisor_pwr2>

      <enable_rising_time_ms>2000</enable_rising_time_ms>
                        <direction>FLIPPED</direction>
                        <feedback_mode>POT_MODE</feedback_mode>
                        <control_mode>POSITION_MODE</control_mode>
                        <name>rightElbowBn</name>
                        <description>Right Elbow</description>
                        <center_ticks>500</center_ticks>
                        <max_ticks>500</max_ticks>
                        <min_ticks>240</min_ticks>
                        <ticks_per_unit>-150</ticks_per_unit>
                                <current_limit>2000</current_limit>
                        <slow_enable_const>2000</slow_enable_const>
                    </Motor>
                    <Motor>
                        <channel>B</channel>
                        <pgain>100</pgain>
                        <dgain>30</dgain>
                        <igain>2</igain>
                        <deadband>0</deadband>
                        <maxstep>0</maxstep>
                        <output_divisor_pwr2>3</output_divisor_pwr2>

      <enable_rising_time_ms>2000</enable_rising_time_ms>
                        <direction>FLIPPED</direction>
                        <feedback_mode>POT_MODE</feedback_mode>
                        <control_mode>POSITION_MODE</control_mode>
                        <name>rightShoulderUpDownBn</name>
                        <description>Right Shoulder</description>
                        <center_ticks>300</center_ticks>
                        <max_ticks>550</max_ticks>
                        <min_ticks>300</min_ticks>
                        <ticks_per_unit>500</ticks_per_unit>
                                <current_limit>2000</current_limit>
                        <slow_enable_const>2000</slow_enable_const>
                    </Motor>
                </Channels>
            </MiniBoard>
        </MiniBoards>
    </MotorBoard>
</Root>
```

D.2 HuggableAndroidController

The following shows just a piece of the code used in the Huggable system. The highlighted area showcases the Analog Input handler.

```
package prg.content.huggable.core;

import ifrobots_android.platform.android.hardware.AndroidSpeaker;
import ifrobots_android.platform.android.ui.AndroidTextLabel;
import ifrobots_android.platform.android.ui.AndroidWindow;
import ioio.lib.api.AnalogInput;
import ioio.lib.api.IOIO;
import ioio.lib.api.exception.ConnectionLostException;

import java.io.FileNotFoundException;
import java.io.UnsupportedEncodingException;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.util.ArrayList;
import java.util.EnumSet;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import mcbmini.AndroidIOIOPSerial;
import mcbmini.MCBMiniBoard;
import mcbmini.MCBMiniConstants;
import mcbmini.MCBMiniConstants.Channel;
import mcbmini.MCBMiniConstants.ChannelParameter;

import prg.content.huggable.bluebear.BlueBear;
import prg.content.huggable.brownbear.BrownBear;
import prg.content.huggable.core.HuggableCommon.HuggableCharacter;
import prg.content.huggable.core.HuggableCommon.MotorSystemType;
import prg.content.huggable.utils.AndroidFacePoker;
import prg.content.huggable.utils.FaceMountedCameraCalculator;
import prg.content.huggable.utils.HuggableUtils;
import prg.content.huggable.vision.AndroidVisionSystem;
import prg.content.huggable.vision.HuggableVisionSystem.VisionModule;
import prg.innards.iNamedObject;
import prg.innards.appcore.InnardsDefaults;
import prg.innards.appcore.Launcher;
import prg.innards.appcore.ResourceLocator;
import prg.innards.appcore.android.AndroidMainActivity;
import prg.innards.graphics.Base;
import prg.innards.graphics.animation.HierarchyUtils;
import prg.innards.graphics.basic.BasicGeometry;
import prg.innards.graphics.basic.BasicHierarchy;
import prg.innards.graphics.basic.BasicSceneList;
```

```
import prg.innards.graphics.filereaders.DirectXReader;
import prg.innards.graphics.filereaders.indirecty.IndirectyLoader;
import prg.innards.math.linalg.Quaternion;
import prg.innards.math.linalg.Vec3;
import prg.innards.namespace.BaseTraversalAction;
import prg.innards.namespace.FindByName;
import prg.innards.network.ircp.IRCPConstants;
import prg.innards.network.ircp.IRCPSender;
import prg.innards.network.ircp.IRCPUDPManager;
import prg.innards.network.ircp.SafePacketHandler;
import prg.innards.util.log.Log;
import prg.innards.util.log.Log.ReportLevel;
import prg.innards.util.log.impl.FileLogger;
import prg.innards.util.log.impl.NetworkLogger;
import prg.innards.util.log.impl.TimedLogger;
//import prg.interfaces.hardware.mcbmini.MCBMiniBoard;
//import prg.interfaces.hardware.mcbmini.MCBMiniConstants;
//import prg.interfaces.hardware.mcbmini.MCBMiniConstants.Channel;
//import
prg.interfaces.hardware.mcbmini.MCBMiniConstants.ChannelParameter;
import prg.interfaces.software.android.graphics.AndroidBasicGeometry;
import prg.interfaces.software.android.graphics.AndroidGLController;
import prg.interfaces.software.android.graphics.AndroidGLLight;
import prg.interfaces.software.android.graphics.AndroidMaterials;
import
prg.interfaces.software.android.graphics.AndroidSkinningFactory;
import prg.interfaces.software.android.graphics.AndroidTextureUtils;
import
prg.interfaces.software.android.graphics.AndroidTouchSphereCamera;
import prg.interfaces.software.android.media.AndroidSoundManager;
import prg.interfaces.software.android.motor.AndroidMotorRenderer;
import prg.interfaces.software.android.network.AndroidMicStreamer;
import prg.research.motor.motorserver.MotorServer;
import prg.research.motor.motorserver.controller.MotorController;
import
prg.research.motor.motorserver.implementations.mcbmini.MCBMiniMotor;
import
prg.research.motor.motorserver.implementations.mcbmini.MCBMiniSerialBu
s;
import prg.research.motor.util.AutoDOFConfigurator;
import prg.research.motor.util.DOFManager;
import prg.research.rendering.MotorRenderingDOF;
import prg.research.speech.AndroidSpeechManager;
import android.content.pm.ActivityInfo;

import JSX.ObjIn;

public class HuggableAndroidController implements iHuggableCore {
      // Instance Variables
      //////////////////////
      protected boolean DEBUG;
      protected boolean connectedToRobot;
```

```
protected boolean enableASAP;
protected boolean IS_ANDROID;
protected boolean ASYNC_GL;
protected boolean has_network;
protected HuggableCharacter bearType;
protected long frameNumber;
protected EnumSet<MotorSystemType> motorSubSystems;

// Motor pipeline
protected BasicHierarchy hierarchy;
protected BasicGeometry geometry;
protected BasicHierarchy glRenderHierarchy;
protected BasicSceneList mainThreadSceneList;
protected MotorServer motorServer;
protected AutoDOFConfigurator dofConfig;
protected DOFManager dofManager;
protected AndroidMotorRenderer mr;
protected boolean motorsEnabled;

// GUI members
protected AndroidGLController glController;
protected AndroidWindow mainWin;
protected AndroidTextLabel versionLabel;
protected AndroidTextLabel updateLabel;
protected AndroidTextLabel debugLabel;
protected long lastPrintMS;
protected int updates;

// Sound stuff
protected AndroidSoundManager soundMan;
protected AndroidSpeaker speaker;
protected long audioPacketsReceived;
protected AndroidMicStreamer micStreamer;

// Network stuff
protected IRCPUDPManager netMan;
protected IRCPSender sender;

// GL camera stuff
protected FaceMountedCameraCalculator faceMount;

//log stuff
protected TimedLogger timedLog;

IOIO ioio;
AnalogInput pressRight;
AnalogInput capRightSide;
AnalogInput capRightLeg;
AnalogInput capRightArmSmall;
AnalogInput capRightArmBig;
AnalogInput pressLeft;
AnalogInput capLeftSide;
```

```
    AnalogInput capLeftLeg;
    AnalogInput capLeftArmBig;
    AnalogInput capLeftArmSmall;
    AnalogInput capLeftEar;
    AnalogInput capRightEar;
    AnalogInput capFrontHead;
    AnalogInput capBackHead;

    private AndroidIOIOPSerial ser;
    private Object ioio_lock;

    boolean analogInputSetUp = false;

    float pressRightValue = 0.0f;
    float capRightSideValue = 0.0f;
    float capRightLegValue = 0.0f;
    float capRightArmSmallValue = 0.0f;
    float capRightArmBigValue = 0.0f;
    float pressLeftValue = 0.0f;
    float capLeftSideValue = 0.0f;
    float capLeftLegValue = 0.0f;
    float capLeftArmBigValue = 0.0f;
    float capLeftArmSmallValue = 0.0f;
    float capLeftEarValue = 0.0f;
    float capRightEarValue = 0.0f;
    float capFrontHeadValue = 0.0f;
    float capBackHeadValue = 0.0f;
    // Data types
    //////////////

    // Initialization methods
    ////////////////////////
    /**
     * Initializes member defaults
     */
    public void launch() {
        IS_ANDROID = true;
        connectedToRobot = false;
        enableASAP = false;
        DEBUG = false;
        frameNumber = 0;
        motorsEnabled = false;
        ASYNC_GL = true;

        Log.println("---------- LAUNCH START ----------");
    }

    /**
     * Initializes a Huggable based on a given type
     */
    public Huggable initHuggable(HuggableCharacter c) {
        bearType = c;
```

```
            // Check if we are connected to a network
            String active_interface =
findActiveNetworkInterface(HuggableCommon.interface_candidates);
            if (active_interface != null) {
                  has_network = true;
            }
            has_network = true;
            // Initialize network
            if (has_network) {
                  initNetwork(c);
            }

            initRig(c);
            initUI();
            initFaceCam();
            //initTouchSphereCam();

            Huggable res = null;
            switch(c) {
                  case BROWNBEAR:
                        res = new BrownBear(hierarchy.getRoot(),
dofManager);
                        break;
                  case BLUEBEAR:
                        res = new BlueBear(hierarchy.getRoot(),
dofManager);
                        break;
                  default:
                        Log.println("ERROR: Invalid character");
                        System.exit(1);
            }
            res.setCharacter(c);
            return res;
      }

      /**
       * Initializes the android rig
       *    glController, hierarchy, dofs
       */
      public void initRig(HuggableCharacter c) {
            // Initialize GL
            glController = new AndroidGLController(!ASYNC_GL, 1, true);

            // Initialize lighting
            glController.getSceneList().addChild(new
AndroidGLLight("dragonLight"));

            // Build the hierarchy and geometry
            hierarchy = new BasicHierarchy("huggable hierarchy");
            if(ASYNC_GL)glRenderHierarchy = new
BasicHierarchy("huggable gl hierarchy");
```

```
            geometry = new AndroidBasicGeometry("huggable geometry");
            AndroidMaterials androidMaterials = new AndroidMaterials();
            AndroidSkinningFactory basicSkinning = new
AndroidSkinningFactory(geometry, glController.getSceneList());

            // Read in the .x file
            DirectXReader reader = new
DirectXReader(ASYNC_GL?glRenderHierarchy:hierarchy, geometry,
androidMaterials, basicSkinning);
            reader.loadFile(HuggableCommon.getRigPath(c));

            // Install hierarchy and geometry
            AndroidTextureUtils.setRepeatMode(geometry);

      glController.getSceneList().addChild(ASYNC_GL?glRenderHierarchy:h
ierarchy);
            glController.getSceneList().addChild(geometry);

            if(ASYNC_GL){
                  mainThreadSceneList = new BasicSceneList();

      hierarchy.setRoot(HierarchyUtils.copyHierarchy(glRenderHierarchy.
getRoot()));
                  mainThreadSceneList.addChild(hierarchy);
            }

            if (ASYNC_GL) {
                  nodesInMainHierarchy = new
ArrayList<Base.iTransform>();
                  nodesInRenderHierarchy = new
ArrayList<Base.iTransform>();
                  new BaseTraversalAction(){
                        @Override
                        protected boolean
actionImplementation(iNamedObject node){
                              if(node instanceof Base.iTransform){

      nodesInMainHierarchy.add((Base.iTransform)node);

      nodesInRenderHierarchy.add((Base.iTransform)FindByName.findFirst(
node.getName(), glRenderHierarchy.getRoot()));
                              }
                              return true;
                        }
                  }.applyAction(hierarchy);
            }

            // Set up the dof config / manager
            dofConfig =
AutoDOFConfigurator.getInstance(HuggableCommon.getRigPath(c),
hierarchy);
            IndirectyLoader.registerWithRegistry();
```

```
            //dofConfig.fixenateModel(hierarchy);
            dofManager = dofConfig.createDOFManager(hierarchy);
      }

      ArrayList<Base.iTransform> nodesInMainHierarchy;
      ArrayList<Base.iTransform> nodesInRenderHierarchy;

      /**
       * This is where we do platform specific initializations to the
creature
       * @param dragon creature to set platform specific settings
       * srcmarks: prg, robots
       */
      public void initPlatformComponents(Huggable bear) {
            // Start TTS and rec
            initSpeechManager(bear);

            // Initialize sound manager
            initSoundManager(bear);

            // Start up motors
            initMotorSystem();

            // Start up face poker
            initFacePoker(bear);
      }

      private void initSpeechManager(Huggable bear) {
            bear.setSpeechManager(new
AndroidSpeechManager(AndroidMainActivity.getActivity().getApplicationC
ontext()) );
            bear.getSpeechManager().setBear(bear);
      }

      protected void initFacePoker(Huggable bear) {
            glController.setOnTouchListener(new
AndroidFacePoker(bear));
      }

      private void initSoundManager(Huggable bear) {
            AndroidSoundManager sMan =
AndroidSoundManager.getSoundManager();
            sMan.init(bear.getCharacter());
            bear.setSoundManager(sMan);
      }

      /**
       * Creates android window, adds the gl controller, sets up debug
UI
       *
       * srcmarks: prg, robots
       */
```

```
	public void initUI() {
		// Create android window to contain GL controller
		mainWin = new AndroidWindow("Huggable Face");
		if (DEBUG) {
			versionLabel = new AndroidTextLabel("GL:-----");
			updateLabel = new AndroidTextLabel("Main:-----");
			debugLabel = new AndroidTextLabel("DEBUG");

			mainWin.addView(versionLabel);
			mainWin.addView(updateLabel);
			mainWin.addView(debugLabel);

			Log.setReportDetail(ReportLevel.EVERYTHING);
		} else {
			Log.setReportDetail(ReportLevel.SILENT);
		}
		mainWin.addView(glController);
	}

	/**
	 * Initializes Huggable communication
	 *
	 * srcmarks: prg, robots
	 */
	public void initNetwork(HuggableCharacter c) {
		byte robotID = IRCPConstants.HUGGABLE_ID; // Default is
huggable
		switch(c) {
			case BROWNBEAR:
				robotID = IRCPConstants.HUGGABLE_ID;
				break;
			case BLUEBEAR:
				robotID = IRCPConstants.HUGGABLE_ID;
				break;
			default:
				Log.println("ERROR: Invalid character");
		}

		netMan = IRCPUDPManager.createWithSuggestedID(robotID,
IRCPConstants.BEHAVIOR_MODULE_ID);
		Launcher.getLauncher().registerUpdateable(netMan);

		if (DEBUG) {
			Log.setReportDetail(ReportLevel.EVERYTHING);

			try {
				FileLogger nfl;
				try {
					nfl = new FileLogger("Log.txt");
					//Log.setLogger(nfl);
```

```
                              Log.setLogger(new
NetworkLogger(IRCPUDPManager.createWithSuggestedID(IRCPConstants.HUGGA
BLE_ID, IRCPConstants.PROJECTOR_CONTRO_MODULE_ID)));

                              System.setErr(nfl.getPrintstream());
                              System.setOut(nfl.getPrintstream());

                         } catch (UnsupportedEncodingException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                         }


                    } catch (FileNotFoundException e) {
                         Log.println("ERROR: Cannot initialize network
file logger");
                         e.printStackTrace();
                    }

               }
          }

          /**
           * Initializes handling and sending of streaming audio
           */
          public void initStreamingAudio(boolean send, boolean receive) {
               // Start streaming to the teleoperator
               if (send) {
                    micStreamer = new
AndroidMicStreamer(IRCPConstants.HUGGABLE_TELEOP_ID, true);

          Launcher.getLauncher().registerUpdateable(micStreamer);
               }

               // Set up handler for remote teleoperator data
               if (receive) {
                    audioPacketsReceived = 0;
                    speaker = new AndroidSpeaker();
                    IRCPUDPManager.addPacketHandler(new
SafePacketHandler("handle audio", null,
IRCPConstants.ANDROID.MAJOR_TYPE, IRCPConstants.ANDROID.AUDIO_DATA) {
                         public void safeHandle(ByteBuffer bb, int length)
{
                                   bb.order(ByteOrder.LITTLE_ENDIAN);
                                   speaker.write(bb, false);
                                   audioPacketsReceived++;
                         }
                    });
               }
          }

          /**
```

```
     * Sets up the motor server
     *
     * srcmarks: prg, robots
     */
    public void initMotorSystem() {
        // Render to our own network id, since motor server is
running in our own process
        mr = new AndroidMotorRenderer(dofManager,
IRCPConstants.BEHAVIOR_MODULE_ID, "Main Motors", DEBUG, has_network);
        mr.dontCheckForPresenceOfMotorSafetyMonitor();
        mr.disableSafetyCheck();
        mr.sendData(true);

        if (ASYNC_GL) {
            mainThreadSceneList.addChild(mr);
        } else {
            glController.getSceneList().addChild(mr);
        }
    }

    // Update methods
    ////////////////
    /**
     * Update GUI with Debug information if necessary
     *
     * srcmarks: debug, huggable, prg, robots
     */
    public void updateDebugGUI() {
        if (DEBUG) {
            long curTime = System.currentTimeMillis();
            if (lastPrintMS == 0) {
                lastPrintMS = curTime;
            }
            if (curTime - lastPrintMS > 1000) {
                float fps = (float)updates / ((float)(curTime -
lastPrintMS)/1000f);
                updateLabel.setText("Main: " + fps);
                lastPrintMS = curTime;
                updates = 0;
            }
            updates++;
            if (glController.getGLVersion() != null) {

    versionLabel.setText(glController.getGLVersion()+" fps: " +
glController.getFPS());
            }
        }
    }

    public void updateTouchIndicator() {

        if (ser.isInitialized()) {
```

```
                if (!analogInputSetUp) {
                //    ioio_lock = ser.getIOIOThreadLock();

                      synchronized(ioio_lock) {
                //          ioio = ser.getIOIO();

                            try {
                                  pressRight = ioio.openAnalogInput(33);
                                  capRightSide =
ioio.openAnalogInput(34);
                                  capLeftSide =
ioio.openAnalogInput(35);
                                  capRightArmSmall =
ioio.openAnalogInput(36);
                                  capRightArmBig =
ioio.openAnalogInput(37);
                                  pressLeft = ioio.openAnalogInput(38);
                                  capRightLeg =
ioio.openAnalogInput(39);   //
                                  capLeftLeg = ioio.openAnalogInput(40);
                                  capLeftArmBig =
ioio.openAnalogInput(41);   //
                                  capLeftArmSmall =
ioio.openAnalogInput(42);
                                  capLeftEar = ioio.openAnalogInput(43);
                                  capRightEar =
ioio.openAnalogInput(44);
                                  capFrontHead =
ioio.openAnalogInput(45);   //
                                  capBackHead =
ioio.openAnalogInput(46);
                            } catch (ConnectionLostException e) {
                                  // TODO Auto-generated catch block
                                  setDebugLabel("connection lost for
analog inputs");
                                  Log.println("connection lost for
analog inputs");
                                  e.printStackTrace();
                            }
                      }
                      analogInputSetUp = true;

                } else {
                      synchronized(ioio_lock) {
                            try {
                                  pressRightValue =
pressRight.getVoltage();
                                  capRightSideValue =
capRightSide.getVoltage();
                                  capRightLegValue = capRightLeg.read();
                                  capRightArmSmallValue =
capRightArmSmall.read();
```

```
                                    capRightArmBigValue =
capRightArmBig.read();
                                    pressLeftValue = pressLeft.read();
                                    capLeftSideValue = capLeftSide.read();
                                    capLeftLegValue = capLeftLeg.read();
                                    capLeftArmBigValue =
capLeftArmBig.read();
                                    capLeftArmSmallValue =
capLeftArmSmall.read();
                                    capLeftEarValue = capLeftEar.read();
                                    capRightEarValue = capRightEar.read();
                                    capFrontHeadValue =
capFrontHead.read();
                                    capBackHeadValue = capBackHead.read();

                                    setDebugLabel("pressRight:
"+pressRight.available() + " capRightSide: "+capRightSide.available()
+ " capRightLeg: "+capRightLeg.available());
                                    setDebugLabel("capRightArmSmall
:"+capRightArmSmall.available() + "   capRightArmBig:
"+capRightArmBig.available() + " pressLeft: "+pressLeft.available());
                                    setDebugLabel("capLeftSide:
"+capLeftSide.available() + " capLeftLeg: "+capLeftLeg.available() + "
capLeftArmBig: "+capLeftArmBig.available());
                                    setDebugLabel("capLeftArmSmall:
"+capLeftArmSmall.available());

                               } catch (InterruptedException e) {
                                    // TODO Auto-generated catch block
                                    setDebugLabel("interruptedException");
                                    e.printStackTrace();

                               } catch (ConnectionLostException e) {
                                    // TODO Auto-generated catch block
                                    setDebugLabel("connection lost");
                                    Log.println("connection lost");
                                    e.printStackTrace();
                               }
                          }
                     }
                }
           }

     private void initMotorServerConnected(){
          String motorConfigPath =
InnardsDefaults.getProperty("configFilePath");
          if (motorConfigPath == null || motorConfigPath.length() ==
0) {
               motorConfigPath = HuggableCommon.motorConfigPath;
          }
          motorConfigPath =
ResourceLocator.getPathForResource(motorConfigPath);
```

```
            // Initialize motor server
            motorServer = new MotorServer();
            System.out.println("HuggableAndroidController /
initMotorSystem: motorConfigPath: " + motorConfigPath);
            System.out.println("IOIO call: HuggableAndroidController");
            motorServer.initialize(motorConfigPath);
      }

      /**
       * Updates the GL controller
       */
      int cnt = 0;
      public void update() {
            if( cnt++ % 30 == 0 ){
                  IRCPSender sender =
IRCPUDPManager.getSenderForID(IRCPConstants.HUGGABLE_TELEOP_ID);
                  if( sender != null && motorServer != null){

      sender.startSubPacket(IRCPConstants.VISION.MAJOR_TYPE,
IRCPConstants.VISION.FACE_API_PACKET);

                        for (MCBMiniBoard board :
((MCBMiniSerialBus)motorServer.getMotorboards().get(0)).getMCBMiniServ
er().getBoards()) {
                              sender.write( board.getId() );
                              sender.write(
board.getChannelAParameter(ChannelParameter.ENABLED) );
                              sender.write(
board.getChannelBParameter(ChannelParameter.ENABLED) );
                              //sender.write(
board.getChannelAParameter(ChannelParameter.TARGET_TICK) );
                              //sender.write(
board.getChannelBParameter(ChannelParameter.TARGET_TICK) );
                              sender.write(
board.getChannelAParameter(ChannelParameter.CURRENT_TICK) );
                              sender.write(
board.getChannelBParameter(ChannelParameter.CURRENT_TICK) );
                        }

                        sender.finishSubPacket();
                        sender.finishPacketAndSend();
                  }
            }

            if (enableASAP && !connectedToRobot){
                  boolean connection =
AndroidMainActivity.getActivity().isConnectedToRobot();
                  if (connectedToRobot != connection && connection){
                        initMotorServerConnected();
                        connectedToRobot = connection;
                  }
```

```
            }

//          if (connectedToRobot && mr.isReadyToEnable() &&
!motorsEnabled) {
//                Set<String> jointSet = new HashSet<String>();
//                List dofsToEnable = dofManager.getRenderingDOFs();
//                for (int i = 0; i < dofsToEnable.size(); i++) {
//                      jointSet.add( ((MotorRenderingDOF)
dofsToEnable.get(i)).getName() );
//                }
//
//                mr.enableJoints(jointSet);
//                System.out.println("Enabling motors !");
//                motorsEnabled = true;
//          }

            if (connectedToRobot && mr.isReadyToEnable() &&
!motorsEnabled) {
                Set<String> jointSet = new HashSet<String>();
                List dofsToEnable = dofManager.getRenderingDOFs();
                for (int i = 0; i < dofsToEnable.size(); i++) {
                      jointSet.add( ((MotorRenderingDOF)
dofsToEnable.get(i)).getName() );
                }

                mr.enableJoints(jointSet);
                System.out.println("Enabling motors !");
                motorsEnabled = true;
            }

            // Update frame count
            frameNumber++;

//          if(frameNumber % 30 == 0){
//                System.out.println("Boards that are enabled");
//                for (MotorController mc :
((MCBMiniSerialBus)motorServer.getMotorboards().get(0)).getControllers
().values()) {
//                      System.out.print(
((MCBMiniMotor)mc).getMCBMiniBoard().toString());
//                      System.out.print(
((MCBMiniMotor)mc).getMCBMiniBoard().getChannelAParameter(ChannelParam
eter.ENABLED) +" ");
//                      System.out.print(
((MCBMiniMotor)mc).getMCBMiniBoard().getChannelBParameter(ChannelParam
eter.ENABLED) +" ");
//                }
//          }

            if (ASYNC_GL){
                Quaternion q = new Quaternion();
                Vec3 v = new Vec3();
```

```
                mainThreadSceneList.update();
                if(glController.obtainHierarchyUpdateLock()){

      //HierarchyUtils.setHierarchyPositions(glRenderHierarchy.getRoot(
), hierarchy.getRoot(), 1);
                      for(int i = 0; i < nodesInMainHierarchy.size();
i++){
                            Base.iTransform renderNode =
nodesInRenderHierarchy.get(i);
                            Base.iTransform mainNode =
nodesInMainHierarchy.get(i);
                            mainNode.getCurrentRotation(q);
                            mainNode.getCurrentTranslation(v);
                            renderNode.setRotation(q);
                            renderNode.setTranslation(v);
                      }
                      glController.releaseHierarchyUpdateLock();
                }
          }

          glController.update();

          //updateTouchIndicator();
     }

     /**
      * Sets the debug label to a given string
      * @param s the string the debug label will display
      * srcmarks: debug, prg, robots
      */
     public void setDebugLabel(String s) {
          if (DEBUG) {
               if (debugLabel != null && s != null) {
                    debugLabel.setText(s);
               }
          }
     }

     // Connection/disconnection delegates
     ////////////////////////////////////
     /**
      * Initializes a face-cam, which attaches to a character's face
for orthogonal projected display
      *
      * srcmarks: facecam, prg, robots
      */
     protected void initFaceCam() {
          if (frameNumber == 0) {

     AndroidMainActivity.getActivity().setRequestedOrientation(Activit
yInfo.SCREEN_ORIENTATION_LANDSCAPE);
```

```
                Base.iTransform faceMountBn =
((Base.iTransform)FindByName.findFirst("faceRootBn",
hierarchy.getRoot()));
                Base.iTransform forwardDirBn =
((Base.iTransform)FindByName.findFirst("eyesRootBn",
hierarchy.getRoot()));

                faceMount = new
FaceMountedCameraCalculator(faceMountBn, new Vec3(0, -1, 0), new
Vec3(0, 0, 1));
                faceMount.setAndroidCamera(glController.getCamera());
                glController.getRootSceneList().addChild(faceMount);
          }

          glController.getCamera().setNear(0);
          glController.getCamera().setFar(1000);
          faceMount.enable();
     }

     /**
      * Initializes a touch-sphere cam, which displays the entire
character
      *
      * srcmarks: touchcam, prg, robots
      */
     protected void initTouchSphereCam() {
          // Unregister the face mounted camera
          if (faceMount != null) {
                faceMount.disable();
          }
          glController.getCamera().setFOV(30);
          glController.getCamera().setIsOrtho(false);
          glController.getCamera().setFar(3000);
          glController.getCamera().setNear(1);

          glController.setOnTouchListener(new
AndroidTouchSphereCamera(glController.getCamera(), new Vec3(0, 20, 6),
60));
     }

     public static String findActiveNetworkInterface(String[]
interfaces) {
          String found_up = null;

          for (String iface_name : interfaces) {
                try {
                     NetworkInterface iface =
NetworkInterface.getByName(iface_name);

                     if (iface == null) {
                          continue;
                     }
```

```
                    if (iface.isUp()) {

      System.out.println("HuggableAndroidController /
findActiveNetworkInterface: iface_name (isUp): "+ iface_name);
                              found_up = iface.getDisplayName();
                    }
               } catch (SocketException e) {
                    e.printStackTrace();
               }
          }

          return found_up;
     }

     public void initVision(Huggable bear) {
          bear.setVisionSystem(new AndroidVisionSystem());
          bear.getVisionSystem().setDebug(DEBUG);
     }

     public void initVision(Huggable bear, VisionModule mod) {
          bear.setVisionSystem(new AndroidVisionSystem(mod));
     }

     public void initVision(Huggable bear, List<VisionModule> mods) {
          bear.setVisionSystem(new AndroidVisionSystem(mods));
     }

     public void vibrate() {
          AndroidMainActivity.getActivity().vibrate();
     }

     public void checkForComm() {
          sender =
IRCPUDPManager.getSenderForID(IRCPConstants.BEHAVIOR_MODULE_ID,
HuggableUtils.getIDFromCharacter(bearType));
     }

     public void setBodyPIDparams(int Pvalue, int Ivalue, int Dvalue,
int deadband, int maxstep){
          for (MotorController mc :
((MCBMiniSerialBus)motorServer.getMotorboards().get(0)).getControllers
().values()) {
               if (((MCBMiniMotor)mc).getMCBMiniBoard().getId() == 1
|| ((MCBMiniMotor)mc).getMCBMiniBoard().getId() == 2) {
                    for(Channel ch:
MCBMiniConstants.Channel.values()){

     ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelParameter(ch,
ChannelParameter.P_GAIN, Pvalue);
```

```
      ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelParameter(ch,
ChannelParameter.I_GAIN, Ivalue);

      ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelParameter(ch,
ChannelParameter.D_GAIN, Dvalue);

      ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelParameter(ch,
ChannelParameter.DEADBAND, deadband);

      ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelParameter(ch,
ChannelParameter.MAXIMUM_STEP, maxstep);


      //((MCBMiniMotor)mc).getMCBMiniBoard().setDirtyParameters(ch);
                        }

                  }
            }
      }

      public void setHeadPIDparams(int Pvalue, int Ivalue, int Dvalue,
int deadband, int maxstep){
            for (MotorController mc :
((MCBMiniSerialBus)motorServer.getMotorboards().get(0)).getControllers
().values()) {
                  if (((MCBMiniMotor)mc).getMCBMiniBoard().getId() == 3)
{

      ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelAParameter(Channel
Parameter.P_GAIN, Pvalue);

      ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelAParameter(Channel
Parameter.I_GAIN, Ivalue);

      ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelAParameter(Channel
Parameter.D_GAIN, Dvalue);

      ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelAParameter(Channel
Parameter.DEADBAND, deadband);

      ((MCBMiniMotor)mc).getMCBMiniBoard().setChannelAParameter(Channel
Parameter.MAXIMUM_STEP, maxstep);


      //((MCBMiniMotor)mc).getMCBMiniBoard().setDirtyParameters(Channel
.A);

                  }
            }
      }
```

```
	public String getBodyPIDparams(){
		String message = "";
		for (MotorController mc :
((MCBMiniSerialBus)motorServer.getMotorboards().get(0)).getControllers
().values()) {
			if (((MCBMiniMotor)mc).getMCBMiniBoard().getId() == 1)
{
				message += "P: "+
((MCBMiniMotor)mc).getMCBMiniBoard().getChannelAParameter(ChannelParam
eter.P_GAIN);
				message += ", I: "+
((MCBMiniMotor)mc).getMCBMiniBoard().getChannelAParameter(ChannelParam
eter.I_GAIN);
				message += ", D: "+
((MCBMiniMotor)mc).getMCBMiniBoard().getChannelAParameter(ChannelParam
eter.D_GAIN);
				message += ", dead: "+
((MCBMiniMotor)mc).getMCBMiniBoard().getChannelAParameter(ChannelParam
eter.DEADBAND);
				message += ", maxS: "+
((MCBMiniMotor)mc).getMCBMiniBoard().getChannelAParameter(ChannelParam
eter.MAXIMUM_STEP);
				return message;
			}
		}
		return message;
	}
}
```