Robotic Kinect Bear

A Senior Project for

The Computer Engineering Department of

California Polytechnic State University, San Luis Obispo

Advised by Dr. Hugh Smith

24 May, 2013

by

Sagiv Sheelo

Dennis Waldron

& Spencer Lines

Table of Contents

I Introduction

The goal of this project was to create an interactive 53”-tall robotic teddy bear to showcase various aspects of Cal Poly’s computer engineering degree. The interactive element took the form of a Microsoft Kinect for Windows sensor, which provided body and face position tracking of the user. Computation was then performed on a PC to convert the raw positional data to joint angles for servo positions (a servo being a motor that can move from 0° to 180°). Those angles were then converted to a message and transmitted to an Arduino Mega controller (a miniature computer with multiple input and output connections). The Arduino then converted that message into signals that servos understand called pulse-widths. Finally, those pulses were sent from the Arduino to each servo by control wires, updating each joint in the bear’s skeleton to the new angle. Using this configuration, we were able to have the robot mimic the user’s face and arm positions in real time, effectually creating a teddy-bear mirror of the user.

Figure 1: Robotic Bear Presented at Open House

The goal of this project was not to just create a robotic teddy bear, but also to showcase the skillset of a computer engineer. This project provides a good example of that skill set by highlighting the interplay between software and hardware which characterizes the computer engineering field. By blending the skills of an electrical engineer with those of computer science, computer engineers becoming effective 'hardware-programmers' in the technology realm. As such, we dealt with power draw, programming in the various C languages, servos, soldiering, and code revision control (keeping track of collaborative code updates and resolving conflicting code changes). During this interplay of hardware and software, we were able to exploit

the flexibility of our computer engineering degree to successfully handle the diversity of this project.

II Design Components

a. Overview

Since this was an interactive product, the user became the central factor in our design decisions and the starting point for our process flow as seen in the following diagram.

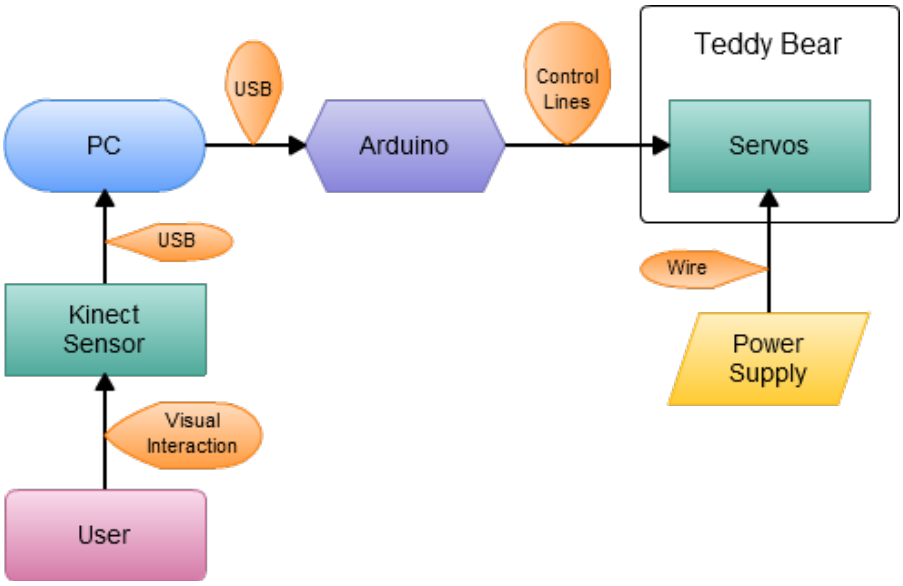

Figure 2: Process Flow Diagram

We wanted the interaction to be as intuitive for the user as possible, which prompted us to use visual interaction technology rather than physical controllers such as joysticks and buttons. It seemed most intuitive to have the user's arms control the bear's arms and the user's head control the bear's head.

The second overarching design goal was to have the bear latency be as low as possible. A “mirror” which drastically lags behind the user is a very ineffective mirror, not to mention harder to control precisely. This prompted us to use more ‘if-logic’ in our code to reduce unnecessary function calls, and to use higher communication rates over the USB interface as well as smaller messages.

Since we were pressed with an early deadline for our project, most of the rest of the design choices were made based on ease of implementation. The respective factors motivating each design decision will now be discussed in detail in the following sections.

b. Bear

i. Skeleton

Figure 3: Bear Skeleton

The teddy bear does not have an internal frame to keep it supported. We needed a skeleton inside the bear that can hold the servos and support the movements we want the bear to have. We decided to construct the skeleton of the bear using PVC pipe. PVC is light enough so that our servos can move it easily but also strong enough to support the teddy bear's stuffing and cloth. PVC is also easy to work with; it is easy to cut to the desired length. There are also many joints available to connect PVC pipes together at various degrees.

Our skeleton consists of a base that runs through the legs of the bear to keep the bear stable on a flat surface. There is a spine coming off of the base that keeps the rest of the bear upright. At the top of the spine there is a joint for the head that controls the up/down and left/right movements of the head. There are arm PVC pipes coming off of each side of the spine to reach the shoulder of the bear. At the shoulder there is a joint that holds the arm PVC pipe of the bear.

Originally we were going to have an elbow or hand joint that would give the arm a greater range of motions, but we decided against it. The teddy bear is simple and its arms are not large enough to allow for a great range of motion. By giving the arms only shoulder motion in the arm height and arm rotation motions, we were able to see the most impressive result that looked realistic for a teddy bear.

ii. Joints and Servos

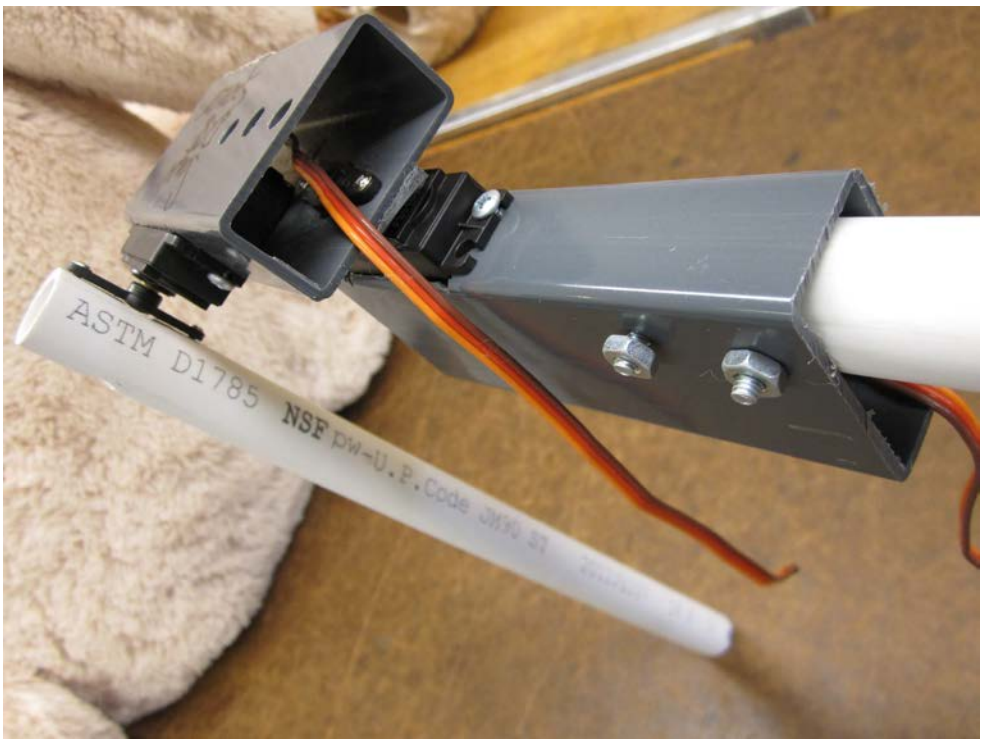

Figure 4: Arm Join Servo Configuration

The servos needed to be mounted to the PVC pipe so that they can control the bear's skeleton. We needed to construct servo mounts so that the servos could be mounted together to represent a joint of the bear. For both the head and arm servos, this meant finding a configuration where the servos emulated a range of motion that a teddy bear would have.

The head joint consisted of a mounted servo that controls the left/right motion of the head attached to the spine. Another servo was connected to the mount that controls the up/down motion of the head. A piece of PVC was attached to the up and down servo, and it went into the nose of the bear so that the servos would be able to move the entire head of the bear.

The shoulder joints used to move the arms of the bear also consisted of two servos. A first servo mounted to the shoulder PVC pipe rotated the arm across the bear's body. A second servo was attached to that that moved the bears arm up and down.

iii. Wiring

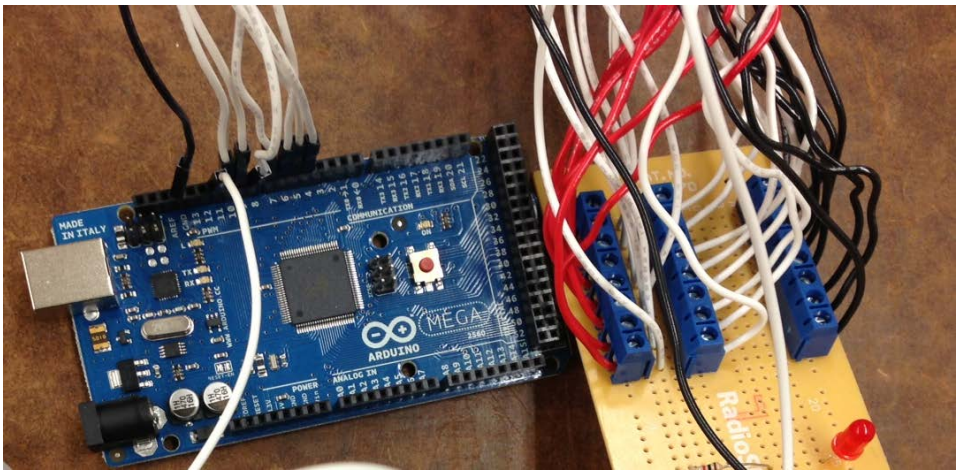

Figure 5: Arduino Mega and Power Board

The servos require three lines each: power, ground, and control. We braided three solid core copper wires together and clamped them to a three pin harness on each end for each servo. This allowed us to easily remove the cables from the servos mounted to the bear for storage. The two power pins went to our power board that supplied 5V and ground. The control pin went straight to the Arduino Mega board where its pin was correlated in the Arduino code to the servo that it was controlling. The Arduino Mega and the power board also shared a common ground.

iv. Bear

Figure 6: Unstuffing the bear

We cut open the back of the bear and removed all of the stuffing from the body, arms, and head. We inserted the base of the bear into the legs first, attaching the spine next. We then were able to connect the head and arms of the bear to the spine. After all of the skeleton was in the bear cloth we placed some of the stuffing back in to make the bear look like a teddy bear. The stuffing of the head was first placed in a mesh cloth bag so that it would be able to be moved by the PVC pipe connected to the head servos.

c. Sensor

i. Kinect

Figure 7: Kinect for Windows, src: http://blog.mic-belgique.be/wp-content/uploads/2012/04/KinectforWindows-Sensor-facing-forward_h_cL.jpeg

To track the physical position of the interactive user we implemented a Microsoft Kinect for Windows sensor. Originally, we had planned on using a standard Microsoft Kinect sensor from an Xbox 360 video game unit, but that sensor required a proprietary adapter and power unit and lacked the performance enhancements of the newer Windows unit. Specifically, the new Kinect for Windows unit has improved firmware which decreases the minimum detection distance from the standard 80cm down to 40cm, as well as providing a standard USB interface optimized for use with a PC rather than an Xbox. Also the Standard Development Kit for creating new Kinect applications is designed specifically for the Windows Kinect Sensor, although it is backwards compatible with the older Xbox sensor.

As for hardware, the Kinect has an infrared emitter, an infrared receiver, a color camera with a 1280x960 pixel resolution, four microphones, and a motor for tilting the unit up and down. The IR emitter and receiver are used to create a depth map and the color camera is used to locate the anatomical positions of the user including the yaw and pitch of the face. The full list of anatomical positions is demonstrated in the following diagram from Microsoft's website:

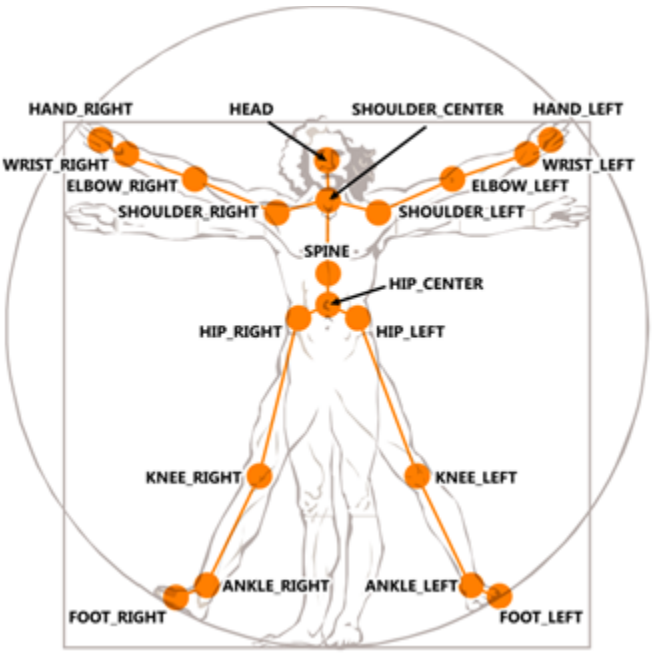

Figure 8: Kinext Capabilities, src: http://i.msdn.microsoft.com/dynimg/IC534688.png

By using these positions tracked in the 3D coordinate system, we were able to calculate the angles of the joints in real-time. While the tracking system was running solo, we experienced millisecond delays between user movement and the tracked display image on the computer terminal. So, the Kinect was more than capable of realtime user mirroring, with the more significant system lag being introduced in the message passing procedure and the updating of servo positions.

d. Processing and Flow Control

i. Computer

The computer software that will run the Kinect and communicate with the Arduino Mega controller is written in C# for the Windows operating system. This was chosen over C++ and Visual Basic because of its similarities to Java, which all three developers on this project were more familiar with.

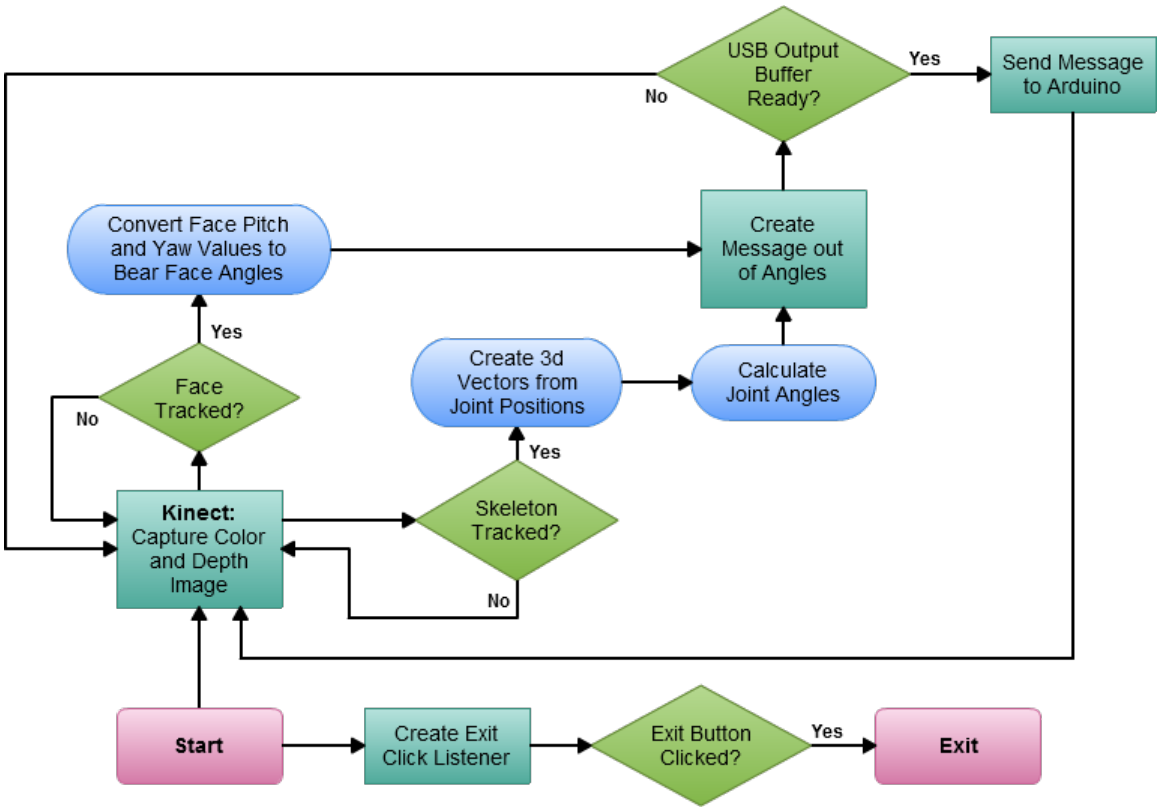

Figure 9: C# Flow Diagram

ii. GUI

We determined that the graphic user interface (GUI) needed to include a live feed of the color video from the Kinect, as well as buttons that would allow a user to specify if the bear is active and how it is interacting with individuals.

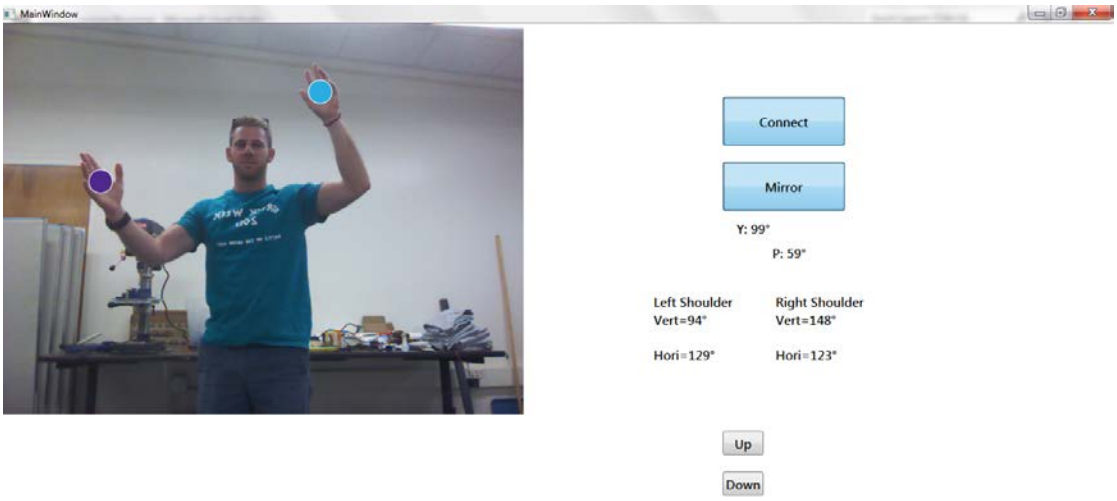

Figure 10: GUI of User Program

iii. Arduino

The Arduino Mega is responsible for the communication to the servos about their current position. Because the Arduino Mega has no physical interface that a human can interact with, we decided that communication to the microcontroller would be through the USB port to the computer using a shell like interface. This design provided several advantages: automatic power to the microcontroller, ease of testing, and minimal communication change when the Kinect software was driving the bear.

Automatic Power: This simplified our design because the Arduino Mega would be powered using the USB lines from the computer. If the input power lines on the Arduino Mega would have been used, then a separate 7V rail would need to be created on the power supply exclusively for the Arduino Mega. Since the servos use a 5 volt rail this would have been wasteful.

Ease of Testing: Because communication was in a shell-like format over the USB port, PuTTY could be used in order to give a testing console to the board. In order for a user to move a servo, a tester would only need to execute a command that the Arduino would interpret and execute. For example, the command "lsv 90" would move the left servo responsible for vertical movement to 90 degrees.

Minimal Configuration Change When Kinect Driven: The only difference between communication to the Arduino using PuTTY and the Kinect code would be the unnecessary echo of character presses back from the Arduino (which allowed the user to see what was being typed in PuTTY). For this purpose "echo on" and "echo off" commands were added.

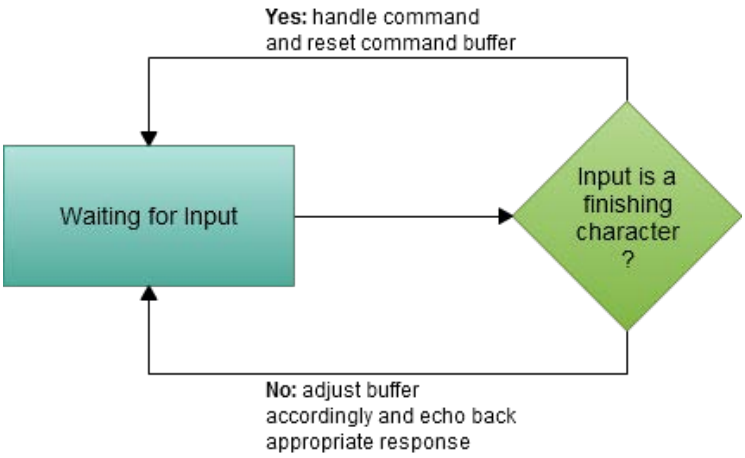

Figure 11: Arduino State Diagram

III Construction and Implementation Details

a. Microcontroller Implementation

The Arduino Mega executable was written using the Arduino IDE in order to take full advantage of the Arduino libraries for prototyping that are provided. This provided the Servo library, allowing us ease of communication to the servos, and the Serial library, allowing us ease of communication to the computer.

Communication with servos is through pulse widths, with a minimum pulse width corresponding to 0 degrees and a maximum pulse width corresponding to 180 degrees. We discovered that in order to obtain the full 180 degrees of motion the minimum pulse width needed to be set to 450 ms and the maximum to 2400 ms.

A character buffer array of size 80 was used in order to hold incoming communication from the computer. As characters came in they would be

added to the buffer and, if echo mode was on, would also be sent back over the USB connection. Some characters of interest include the line feed and carriage return characters (when one of the two arrived then both had to be echoed back for PuTTY to display properly) which would signal the end of a line and to execute the command and the backspace character which would move a pointer for the current index in the buffer backwards.

When a command was entered, the strncmp() function was used in order to determine what type of command had been received. If it was a command that moved a specific servo, then the atoi() function was used on a portion of the string in order to obtain the angle. The microcontroller would then take appropriate action, usually by calling the Servo.write() command.

Even though non-echo mode usually suppressed writing back to the Serial port, one special command was included that would write back. The command "InitArduinoComm" always caused "InitArduinoCommResponse" to be returned. This was used on Kinect code startup by sending the command to any available communication ports. If the port responded with the appropriate reply, then that communication port would be used.

b. Kinect Implementation

i. Algorithms used to obtain angles

The Kinect API provided face and skeletal tracking information that had to be processed into angles for individual servos in the bear. The face tracking provided pitch and yaw already, so they only had to be adjusted and restricted to the allowed pitch and yaw for the bear's head. On the other

hand, the skeletal tracking provided spatial points in the XYZ space for individual joints.

In order to obtain angles for the shoulder servos, the XYZ joint coordinates needed to be reduced to angles. This was done by first creating a vector (origin based) for each arm by subtracting the wrist point from the shoulder point. One angle is calculated by finding the angle of the arm vector imposed onto the XZ plane and using the Math.Atan2() function. The second angle is obtained by finding the angle between the vector and the imposed XZ vector obtained previously. These are then adjusted in order to match the conventions for the bear's arm servos.

ii. Graphic User Interface

The graphic user interface (GUI) was created in order to provide meaningful feedback to us as developers as well as those driving the bear. Some functionality was also introduced in order to better allow control over the bear.

The main section of the GUI was a color image of what the Kinect was currently seeing, with two balls superimposed over the image over the hands of the person driving the bear. This was the main indication to determine who had control of the bear and if the skeletal tracking was working reliably. It also allows users to determine if they need to reposition themselves so that the camera can pick up more motion.

The GUI also displays the current angles that have been calculated for the bear. This allows for quick visual verification of the calculated angles and that the bear is moving correctly.

The GUI also contains several buttons. One button engages the bear, initiating the imitation of the human user. Another button allows the toggling of pure imitation (movement of right arm triggers movement of bear's right arm) and mirroring imitation (movement of right arm triggers movement of bear's left arm, as you would expect when looking in a mirror). Two button were also included in order to control the pitch of the Kinect.

c. Construction

i. Materials

PVC types (1" x 2" and 0.75" round), screws, nuts

ii. Servo Mounts, Arms, Neck

Servo mounts for the shoulders were created by taking rectangular PVC and cutting out a window about the size of a servo so that the servo can be slipped in and screwed down into the PVC. For the first servo mount on each shoulder, round PVC pipe was inserted into the rectangular PVC and then two holes were drilled through both PVC. Screws were then put in through the holes in order to guarantee a secure hold. The second servo mount on each shoulder has a servo arm screwed into the rectangular PVC, which connects to the servo in the first mount. The second servo is then connected to a long round PVC pipe which has a servo arm attached in order to act as the arm of the bear.

Servo mounts for the neck were created by putting the round PVC pipe that connects to the frame into a rectangular PVC pipe, drilling two holes, and screwing it down, similar to the shoulder mounts. The difference in this mount is that a servo window wasn't cut, but instead inserted into the opening of the rectangular PVC. A second mount is created in the same manner as the second mount of the shoulders and attached to the first servo. The second servo of the neck is also attached to a long round PVC pipe, which is then attached to the inside of the bear's nose.

iii. Bear Frame

The bear's frame is based off of a simple 't' shape, two pieces of round PVC coming out horizontally connecting to the servo mounts, one piece of round PVC coming out vertically to the neck mount, and one round PVC coming out vertically down to a base of round PVC pipe that keeps the structure stable.

IV Problems and Solutions

a. Faulty Servo

i. Problem

The smaller servos we were originally going to use for tilting the "hand" caused all our other servos to start moving radically.

ii. Investigation

We knew one of the servos was causing the strange behavior because it was short circuiting. In order to narrow down which servo was causing it we plugged in one servo at a time. Once we narrowed down to one of the small hand servos we tried replacing it with a different small servo. This servo caused the same behavior and on analyzing the two both were short-circuiting.

iii. Solution

We found two small servos that were operating correctly and used those in our bear. In the end, we decided against having hand servos and did not use the small servos at all.

b. Overheating Servo

i. Problem

The servos in the head of the bear overheated and caused their plastic mounts to melt.

Figure 12: Overheated Servo Melting

ii. Investigation

We first noticed that something was wrong when we smelled a strong odor of melting plastic in the lab. When we went to check the servos we noticed that they were extremely hot and that the mount was melted. Upon further investigation we figured out that the servos were being sent angles to move to that were past what the bear fabric will allow. The servos trying to move to an angle while being physically stopped from doing so caused them to overheat.

iii. Solution

After getting two new servos and creating new mounts for them, we had to fix the code so that there are maximum angles that the arduino will attempt to move the servos to. Now the arduino code that controls the servos has limits that it will enforce on the servos so that they will not be moved to angles beyond their physical limitations. We manually moved each servo until we saw that it was pulling on the bear fabric. That is the point where we set our servo limit in the code.

c. Power

i. Problem

The six servos seemed to be draining all the power from the power board, thus causing them to fail and become unresponsive for a time.

ii. Investigation

We realized that the combined weight of the PVC pipe, bear stuffing, and bear fabric, more current was needed than was being provided when multiple servos were moving, most notably the head servos.

iii. Solution

The solution to this problem was to use a power supply for a desktop computer. It could provide plenty of current at the needed voltage, and therefore fixed the issue.

d. Arm Servo Configuration

i. Problem

The original placement of the servos created a "pole" (similar to the poles on the earth) on the arm movement to the side of the bear. This created an issue if the bear was flapping its arms up and down to its side.

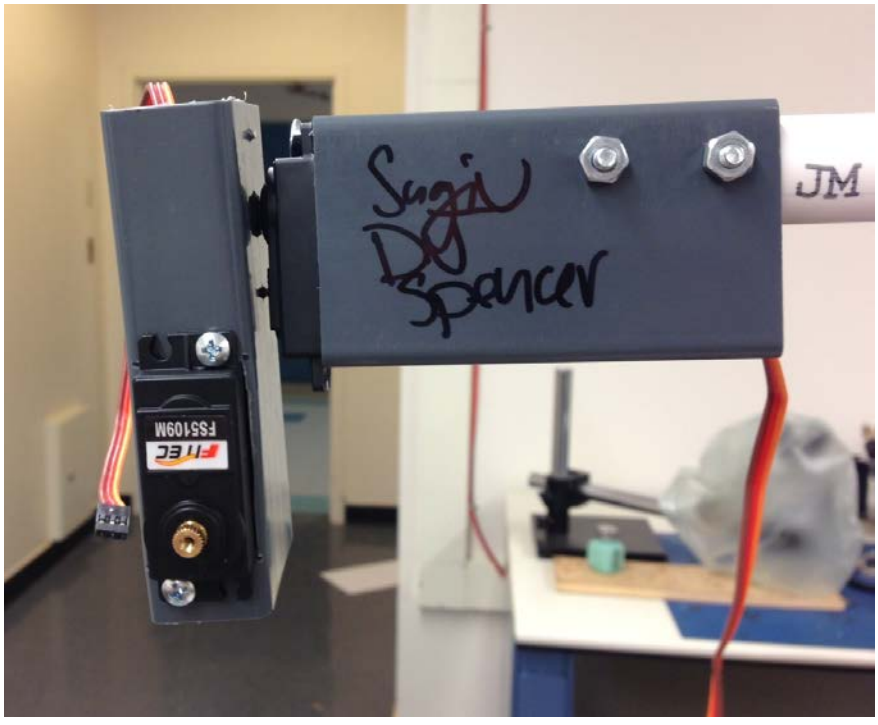

Figure 13: Old Arm Servo Configuration

ii Solution

A new placement of the servos had to be done to move the poles of arm rotation to directly above and below the servos, allowing freedom of motion when performing almost any movement.

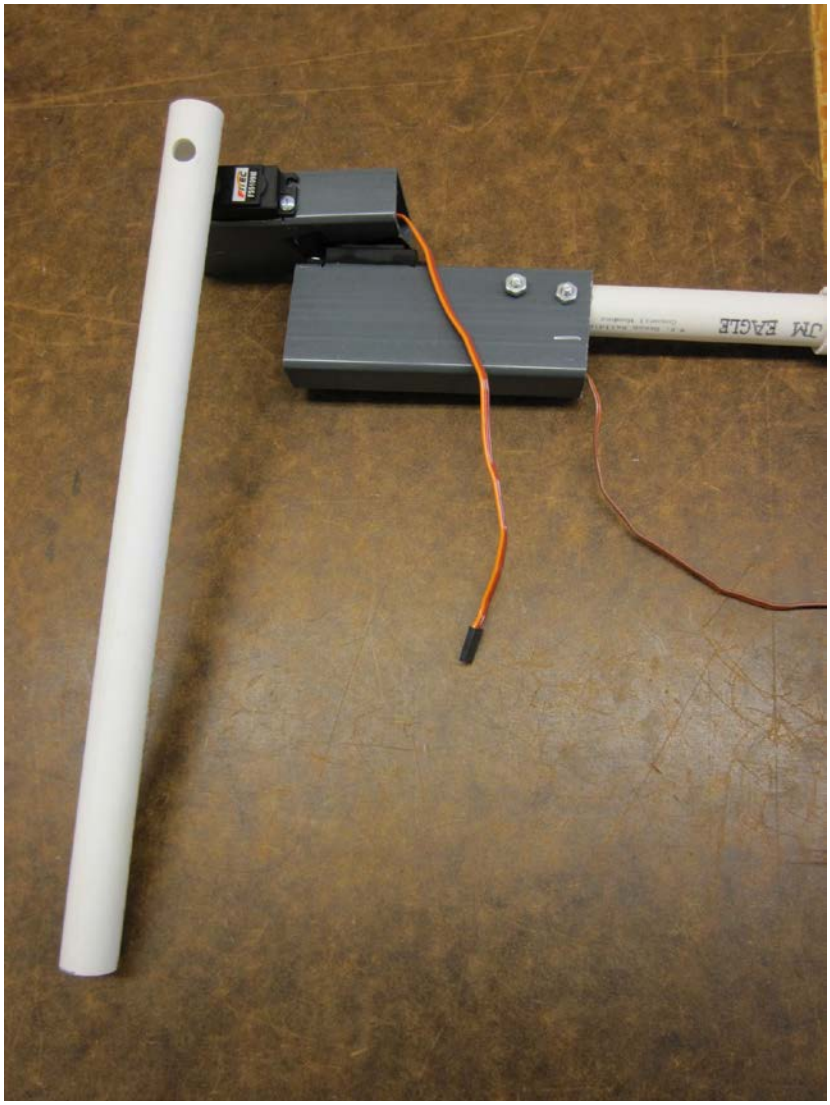

Figure 14: New Arm Servo Configuration

V. Final Thoughts

This was an excellent project to exhibit what it means to be a computer engineer. We were able to take hardware and software and use it to make something new and fun to interact with, and the creation process was also entertaining and rewarding. Thanks for taking an interest in what we did!

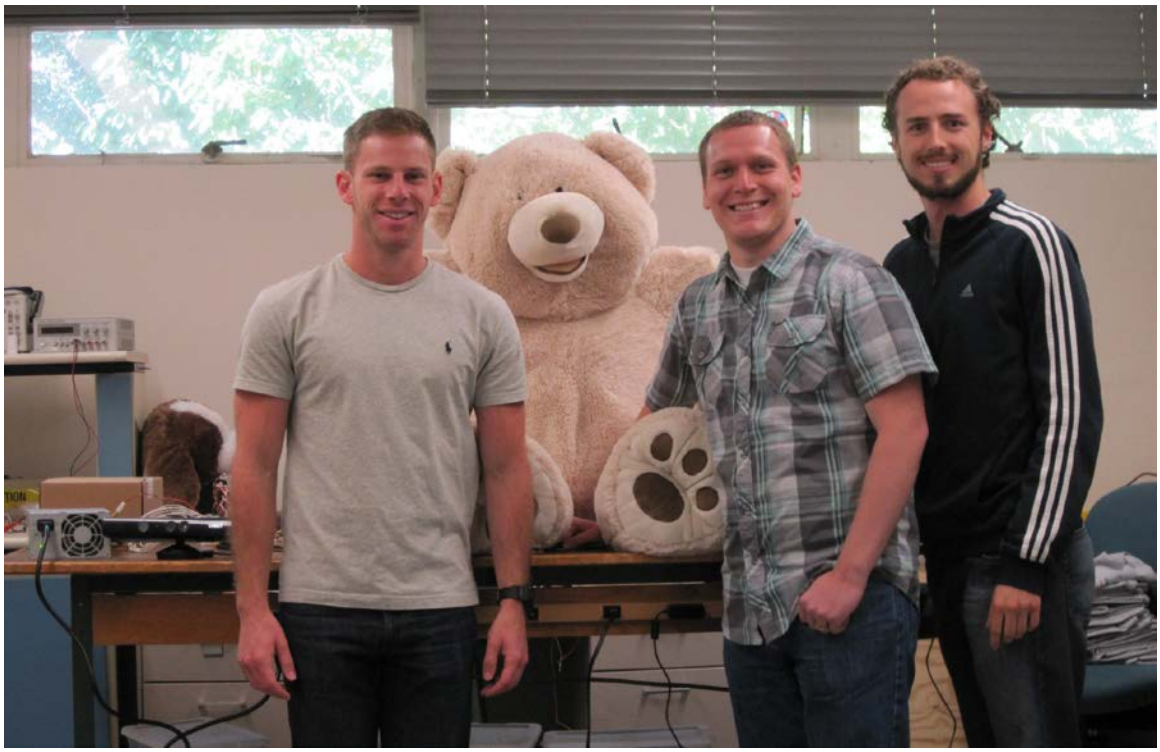

Figure 15: Group members from left to right: Sagiv Sheelo, DJ Waldron, Spencer Lines