Bruno the Robotic Bear

By Aaron Brauner and Ian Panzer

Advisor: Dr. Hugh Smith
Department Chair: Dr. Hugh Smith
Department: CPE
College: College of Engineering
# TABLE OF CONTENTS

ABSTRACT ........................................................................................................................................... 4

BACKGROUND ..................................................................................................................................... 6

ROBOTS AND BRAINS .......................................................................................................................... 6

WHY A BEAR? ....................................................................................................................................... 7

INITIAL PROJECT CONCEPT ............................................................................................................ 7

SKELETAL CONCEPT ............................................................................................................................. 7

GUI CONCEPT ....................................................................................................................................... 9

SKELETAL DESIGN ............................................................................................................................... 11

POWER BOARD .................................................................................................................................. 17

arduino code ......................................................................................................................................... 23

GUI ....................................................................................................................................................... 25

CONCLUSION ....................................................................................................................................... 34

REFERENCES: ...................................................................................................................................... 36
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial Bear Concept Drawing</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Bear Initial GUI Design</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Entire Skeletal Structure</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Shoulder Elbow Assembly</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Shoulder Elbow Assembly Different Angle</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>Foot Design</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>CPE-200 Power Board</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>Initial Power Board Design</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>Modified PSU Circuit Board</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>Final Power Board Design</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>Partial Main State Machine</td>
<td>23</td>
</tr>
<tr>
<td>12</td>
<td>Example of Individual State</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>Manual Mode</td>
<td>25</td>
</tr>
<tr>
<td>14</td>
<td>Basic GUI</td>
<td>26</td>
</tr>
<tr>
<td>15</td>
<td>Advanced GUI</td>
<td>27</td>
</tr>
<tr>
<td>16</td>
<td>Serial Port Class for Communication with Arduino</td>
<td>29</td>
</tr>
<tr>
<td>17</td>
<td>Main GUI Code</td>
<td>33</td>
</tr>
</tbody>
</table>
Abstract

Creating homemade robots are becoming a popular hobby among engineers. With the dramatic reduction in costs over the past few decades, it is finally possible to build a personal automated mailbox. Microcontrollers needed to function as the brains of the robot are also becoming powerful enough that relatively complicated instructions could be programmed into the robot to handle a plethora of sensors and peripherals. In order to demonstrate how the Computer Engineering department models the synthesis of computer science and electrical engineering, the topic chosen was robotics, specifically, a stuffed bear robot.

The microcontroller chosen was the Arduino Mega 2560 Rev3, an inexpensive microcontroller manufactured in Italy. Due to the open-source nature of the Arduino, there is a large community of developers who have written numerous libraries for specialized functionality, such as timers, sensor reading, and music players. Also, the language used to program the Arduino is based off of C++ and is relatively simple to use. That, coupled with the low power requirements, made the Arduino an excellent fit for the project.

The skeleton for the robot was constructed out of PVC pipe, which is usually used for lawn sprinkler systems. After seeing and measuring the plush bear provided by Dr. Smith, the skeleton was designed in a modular fashion for easy disassembly, should that become necessary. A total of seven servos were added in order to provide motion. This allowed for the elbows, shoulders, ankles, and the neck to twist. Finally, both the Arduino and the power distribution board were mounted in a project box, to sequester the electronics from the rest of the robot and allow for sufficient heat dissipation.
In order to allow for some user interactivity, a Graphical User Interface (GUI) was created. The application was programmed in C# in order to take advantage of the tools that allow for quick creation of buttons and sliders. Also, C# has an easy way to handle serial communication, the preferred method for communicating with the Arduino. The GUI has two modes of operation, manual and automatic. In automatic, the bear goes through a series of motions. In manual, a user can send commands to the bear to do certain motions and move certain joints. The GUI provided an enjoyable source of interaction between the robot and the user.
Background

Robots and Brains

The idea of robots has been around for a long time. In general, a robot is a mechanical device that senses and manipulates its environments based on parameters that have been set by the programmer. In the modern era, robots have been used in factories, bomb disposal, entertainment, and prosthetics. Unfortunately, building robots for those complex situations is beyond the capability of the average person. However, building a simple robot is relatively easy and inexpensive, thanks to the high availability of skeletal supplies, servos and power circuitry, and easy-to-program microcontrollers.

In order for a robot to function, it needs some kind of computer to store and process the instructions provided by the user. For larger robots, a full-fledged computer would be a reasonable choice, because the robot could carry the computer and the processing power needed for all of the peripherals that can be attached to a larger platform. However, a computer does not make sense for a project of this scale. Something with a much smaller physical and electrical footprint is necessary, such as a microcontroller. A microcontroller is essentially a computer on a chip. It has a processor, memory, a clock, and input/output capabilities all in a single simple package.
Why a bear?

At the beginning of the year, Dr. Smith created the Animated Stuffed Animals lab. In the Winter Quarter 2013 he taught a sophomore level class on how to design and build a small robot. The students designed and built their own skeleton and then put that skeleton in a small stuffed animal. The class was designed to showcase Computer Engineering as a combination of circuit and program design. The class was a success and about ten smaller robotic animals were built. As he designed the class, Dr. Smith found several 53” tall plush bears and thought that putting robotic skeletons in them as well would make an excellent senior project. The initial design happened in Fall Quarter 2012 and the actual construction went from January 2013 through April, finishing around the 15th. At the university Open House in the middle of April 2013, visitors were able to see and interact with the bear, much to the delight of the younger children.

Initial Project Concept

Skeletal Concept

Initially when assigned the project, the idea was to have every joint within the bear move. This included the neck, arms, and legs. By having this freedom of movement the maximum number of motions could be programmed for the bear to perform. The movements that were planned out are: hugging, wave right and left arms, wave both arms simultaneously in the same and opposite directions, and move feet. After integrating sounds the motions of “Think” and “Rumbly in my Tummy” were also planned out. In order for these planned motions to execute the placement of the servos within the skeletal structure of the bear would have to mimic that of the human body. To accommodate some of the more complex motions two servos had to be
placed as each joint location in order to move up - down and right - left. All of these design ideas can be seen in figure 1.

![Figure 1: Initial Bear Concept Drawing]

As can be seen in figure 1, several servos have been depicted for several joints within the bear. The hugging, and waving motions used the servos within the neck, and arms. The “servo nod” was used for moving the head up and down. This servo was a crucial piece to the hugging motion. As the bear went into the hug motion, the bear was anticipated to look at the recipient of
the hug as if giving a caring gesture for that individual. This also represented the natural motion that a head gives when giving a loving hug between two individuals. The Hug/Wave and rotate servos in the arm were there to mimic the shoulder and elbow joints of a human. When any individual interacted with the bear natural look and feel was sought. Within the leg joints the feet were the targets for movement. As the bear was being envisioned to be sitting throughout the interaction with people, the feet were the only joint within the legs to have movement that made sense. Having the feet move was felt to be a quirky fun addition to the standard set of movements the bear was performing. This simple motion gives the impression that the bear was dancing and, therefore, was having fun with the individual controlling the bear.

**GUI concept**

How to control the bear was a primary concern for development. It was felt that in order for an individual to have fun and want to come back to use the bear in the future a GUI needed to be created which needed to be intuitive to use. When initially planned out the GUI was designed so that the user could move any joint within the bear to a specific position thus allowing them to control the bear fully. This is illustrated in figure 2. This figure shows how the user can move a joint to a specific position based on the angle the servo will move to. This was accomplished with slider bars in the center of the GUI. Another aspect of the GUI was the ability to perform a pre-programmed motion. A further section of the GUI allowed the user to play sounds and/or initiate motions, giving a personal feeling of attachment to the bear.
The design of the GUI was meant to inspire the user to be creative. The creativity component involved putting the individual joints in different positions and having the ability to basically come up with their own motion. Another aspect considered was the ability to save these custom motions for later so that future users could call on them to see how creative others had been. This was later scratched due to the complexity of trying to save these individual motions and how they would not necessarily be easy to control and reuse.

Beyond the GUI design the underlying motions are the key to the design of the bear. The bear has both a manual mode and an automatic mode that works by repeating a series of motions that it knows. Originally however, the design called for the bear to wake from a sleep state to follow individuals that had come across its field of vision thus allowing an interactive automatic approach. With these motions an idea for the bear talking to the user was proposed. The bear would say famous quotes from cartoon bears of their childhood. Yogi Bear © Hanna-Barbera
Productions Inc. and Winnie-the-Pooh © The Walt Disney Company were the two bears chosen to be utilized for sound bits. With these sounds coming from the bear motions could also be performed that went with the associated audio track. This effect of the bear talking to the user was meant to bring them closer to bear and feel as if it was not just another toy to play with.

**Skeletal Design**

For this design, a very simple human skeletal system was modeled. This layout of the skeleton provided the most stability as well as kept the weight of the design down to a minimum. Figure 3 shows the completed skeleton with seven servos.
Starting at the head there was a servo to control the neck’s side-to-side movement. This servo was a key component for the tracking motion of people walking across the field of vision of the bear. Next to this servo was another servo that enabled the head to move up and down, imitating a nod. This nodding type of motion allowed the user to feel more welcome to the bear. In the final design this servo was scratched as the excess weight in the head made the power draw from this servo alone too much for the power board to handle. The head had to be held in an up position from the natural slightly down position that it started at. In order for this servo to handle this control it would require 1A constant draw to hold the position of up. While this seems like a feasible amount of current, the power supplies initially used could only supply 1.5A each.

Still how is that there was not enough current. This comes from the fact that in normal motions the servos require 200mA of current to operate, and with eight servos in total plus the power for the Arduino Mega there was just not enough current available. Another problem that arose with the constant 1A current draw was heat. With constant current the servo was getting so hot, that there was concern that the stuffing of the bear might combust. After this consideration the nodding gesture was removed, as was the servo. For the head servo a PVC T was connected in order to give stability and weight distribution for the entire head. Out of the T are three PVC pipes that protrude to the sides or nose of the bear’s head. The strength of the PVC allowed for the head to be held more upright.

The shoulder - elbow joint design was tricky to get right. The question posed was how to create a joint that can both move up and down as well as left - right within a small confined area. The arms on the bear were short in comparison to that of a real human. With this in mind the elbow joint was moved up to the shoulder joint, allowing for more fluid movement and the
ability to perform the planned motions. As seen in figures 4 and 5 the shoulder and elbow are in close proximity.

![Shoulder Elbow Assembly](image)

**Figure 4: Shoulder Elbow Assembly**

As seen in this figure the shoulder joint is at a 90° angle to the main Skeleton. This allowed the arm to traverse up and down. The tricky part was including the elbow in this area. The solution developed was to have the elbow sit on end next to the shoulder servo allowing for left to right movement while not interfering with the shoulder up-down movement. This is visible in figure 4 as can be seen as the elbow sits at the end of the joint tube just out of the plain of rotation for the shoulder. The angular cut that was made in the joint area was a twofold improvement. First this design cuts down on the amount of weight in each arm thus lessening the
amount of current needed to move the arm. Secondly this design allowed for less material to bind when the shoulder was traversing up and down.

Figure 5: Shoulder Elbow Assembly different angle

From the views in figures 4 and 5 a better understanding of the relationship of shoulder to elbow is visible. In order to secure each servo into place for each joint, setscrews were utilized along the side of the servo. Two setscrews are enough per servo to ensure secure placement. If a future design was to be considered and the head were to have up and down movement included, this design would fit nicely into the head.
Moving to the base of the bear, a wide stance was used to provide the most stability when the upper section was in motion. Again, the sectional-design for the bear was modeled after the human body. The wide stance models after human hips which gives stabilizing or structural support in humans. Instead of going straight out and then turning 90° to move out to the feet a 45° gradual change was used.

The last joint designed was the feet. This design was the easiest design of the three major joint areas. The feet only needed to have the ability to move side to side and not have any up-down movement. Figure 6 shows how the feet were designed.
This design only needed to be able to wiggle the feet when called upon. The PVC pipe was able to handle this motion. The pipe sticks out to the side of the foot without actually touching the sides. By placing enough stuffing in the foot and moving the servo a wiggle was easily performed.

The final skeletal design decision was how to wire the bear in order for the joints to have power and signal. The PVC piping that served as the skeletal system provided a simple and effective conduit for running the wiring necessary to control each joint. Each set of three wires; red, black, and yellow were twisted together in order to be able to differentiate each set of joint wires from each other. The twisting of wires together also would cut down on electromagnetic interference.

There are several design improvements that could be made to the skeletal design. One primary design improvement would be to implement a four pillar main structure for support and stability. The single spine design was functional, however; this design does not support the big bear very well and leans quite a bit. By adding a four pillar design the load for the upper section of the bear can be spread more evenly over the pillars while also stabilizing the bear more. This four-pillar design would be like a cube in shape where all pillars are interconnected at the base and at the top of the structure. For branching out to each joint a T coupling can be utilized to extend out to the proposed joint from the structure. For every section that has a coupling attachment a setscrew should be utilized to secure the coupling in place. This will enable future rework of the design and allow for the skeletal frame to me modularized. Another design improvement that can be made is tubing to surround each joint so that stuffing from the bear does not get into the joint areas. This tubing however cannot interfere with normal joint
movement and therefore, is very hard to design for. The current format did not have a problem with the stuffing but it is foreseeable that an issue can arise.

In order for all the joints in the skeleton frame to move 5 - 6 Volts needed to be routed to each servo, as well as a pulse width modulated (PWM) signal. The power board design accomplished this task.

**Power Board**

In order to get the proper voltage to each servo, a power distribution board had to be designed and built. The power board has to have high voltage for each of the servos, a signal connection for each servo, and a ground plane, which also connected to the Arduino. The output voltage had to be between 5 and 6 volts, meaning some voltage regulation had to be done between the input and the servos. Rough calculations gave the current need for a single servo at approximately 200 mA, with spikes greater than or equal to 1A. Multiplied by eight servos, that made for a total current draw of at minimum 1.6 A. Thus, the external supply needed to be able source at least that much current and preferably quite a bit more, such as 2A. Also, the voltage regulators needed to be able to handle those current spikes that could occur when the servos move. All of these requirements made building the power board the most challenging part of the project.

The initial design was based on the board created by Dr. Smith for the Animated Stuffed Animals class shown in figure 7.
Figure 7: CPE-200 Power Board

The main features of this board were the single input, single regulator, and supply to the Arduino. There were several advantages to this design. Firstly, it was relatively simple to understand. The connections were easily traceable from input to the high voltage plane to the ground plane. Because of the simplicity, it was also easy to build. The sophomores, who have not taken any advanced circuit classes, were able to build this circuit. Unfortunately, there are also some disadvantages. While this design could be fine for the smaller robots, which use micro servos, the voltage regulator could not handle the current being pulled by the standard size servos used in the bear. The regulator was generating enough heat that component failure was a distinct possibility. Secondly, the power supplies provided by Dr. Smith were unable to provide enough current for the servos to move properly. These proved to insurmountable and this design was put aside.
The second design, shown in figure 8, expanded upon the basic version to divide up the current flow.

![Figure 8: Initial Power Board Design](image)

This version has four voltage regulators with two servos per regulator. Each regulation circuit was essentially a minimized version of the same circuit in the basic power board. It also supported two power inputs, giving two regulators per power supply. This cut the current draw per supply to 800 mA minimum. The main advantage for this design was that it split the current draw for the servos across two separate circuits. This reduced the current draw on the power supplies to be within their specifications. A related advantage was that it also reduced the amount of current passing through each regulator. Because each regulator was only supporting
two servos, the current draw was reduced to 400 mA at the minimum. The main disadvantage to this version was that it was a challenge to build. The components were in such close proximity that it was difficult to route all of the wires properly. Also, the servos were still trying to pull more current than the regulators could handle, causing significant amounts of heat. The regulator that fed the left arm was especially hot, enough so that one of us actually got a slight burn from the attached heat sink. Obviously there were changes that needed to be made.

The initial power supply maxed out at 1A with 9VDC, similar to the supplies provided to the students for the Animated Animal class. While it may have been sufficient for the smaller animals, this supply could not provide sufficient current for our project. The servos in the shoulders could not get enough current to lift the arms. With the second power board having two power inputs, adding another of these power supplies would hopefully alleviate the lack of current flowing into the circuit. It seemed that the supplies were able to deliver enough current if only one servo was moving on each circuit, but the load of two servos holding position and two moving, the supplies couldn’t handle it. The next attempt at a power supply was a 6V 12A power supply from a Dell laptop. However, this was short lived, because the power supply refused to provide power. There appears to be some communication that occurs between the laptop and supply to confirm that the branding of each component is the same and the bear was not a Dell. Finally, it was decided upon to try using a Power Supply Unit (PSU) from a PC. Two Molex connectors were attached to the power input and connected to the PSU. The PSU would only turn on if the motherboard were also powered so it was decided to leave the PSU within the computer. This setup worked wonderfully. However, the voltage regulators on the power board became extremely hot, because the PSU could source much more current than they were rated for. After using this setup at Open House, another PSU arrived for use with the project. In order
to streamline the operation of the bear, the supply’s internal circuit board was modified to remove extraneous power rails and to remove the need for a PC motherboard. Figure 9 shows the modified PSU circuit board. Instructions for how to modify the board can be found at http://www.gideontech.com/content/articles/196/1. All of the wires except for the 5V and ground rail needed were desoldered from the board.

This resulted in a trimmed down power supply that outputted a regulated 5V and 11A current. This made the voltage regulators on the power board superfluous and as such they were removed.
The power board’s final design was relatively simple. The power from the 300W PSU was routed into two inputs on the power distribution board. Then, each circuit split into two high voltage rails, each supplying up to four servos. All external regulators were eliminated and thus the heat output was significantly reduced. The servos were happy with the amount of current and voltage being supplied. Figure 10 shows the final power board design.

Figure 10: Final Power Board Design

The power board only routes the PWM signal for controlling the servo. The Arduino Mega 2560 Rev3 accomplished the generation of the PWM signal for control of each servo.
**Arduino Code**

Dr. Smith provided the initial version of the code used to control the bear. It was designed to easily allow a programmer to add servos as objects and use those objects to create relatively complex motions. The main loop was set up as a state machine, with each motion as a sub-machine. A portion of the main loop is shown in figure 11.

```
switch (state)
{
    case STOPPED:
        break;
    case COVER_EYES:
        if (cover_eyes() == DONE)
        {
            state = STOPPED;
        }
        break;
    case WAVE_LEFT_ARM:
        if (wave_left_arm() == DONE)
        {
            state = STOPPED;
        }
        break;
    default:
        break;
}
```

*Figure 11: Partial Main State Machine*

Each motion has its own state, which advances after the servos have finished moving to the instructed position. An example state from the WAVE_LEFT_ARM state machine is shown in figure 12.
case ARM_RIGHT:
    // arm is moving right, when done move it left
    if (If_eb.done() == DONE)
    {
        If_eb.moveLeft(MEDIUM);
        if (wave_count == 2)
            neck.moveMiddle(MEDIUM);
        state = ARM_LEFT;
    }
    break;

Figure 12: Example of Individual State

Adding more motions was as simple as creating more state machines, deciding on the number and types of states, and then choosing proper positions for the servos. However, for this project, there were several features that needed to be added.

The first step was to allow the Arduino to be able to communicate with an external computer. This was accomplished using serial communication through the attached USB cable. In order to start the connection, a basic handshake protocol was utilized. The Arduino waited for a specific string of characters to arrive on the serial port. If the string arrives, the Arduino sends back the proper response and jumps into auto mode. If the string did not arrive within 10 seconds, the Arduino turned off the serial port and went into auto mode. The key difference was that turning off the serial port disables manual mode. Once a connection was made, the Arduino called the serial_query function every cycle to check for a new command. The function differentiated between mode switches, motions, and specific joint movements and set the proper flags to cause the program to jump to the proper state machine. Serial communication was most often used in manual mode.

While having the bear act out a series of motions was interesting, the best part of the project was user interaction through manual mode. This mode was only available when a
connection has been established with an external computer. When a command for a motion has been received, the Arduino jumps to this section to decide what to do:

```c
unsigned char manual()
{
 static int state = STOPPED;
 while((state != DONE && state != NOT_MOVING) && type != 0)
 {
   if(type == JOINT)
     state = move_joint(part, degree);
   else if(type == MOTION)
     state = do_motion(part);
   move_all();
 }
 state = STOPPED;
 type = 0;
 part = 0;
 return DONE;
}
```

Figure 13: Manual Mode

Within the move_joint and do_motion functions are more switch statements to determine exactly which joint should move or which motion should be done. Like auto mode, manual is a state machine, checking to make sure that each state is finished before moving on to the next. Also, thanks to the addition of a speaker controlled by the external PC, sounds were added to the project. Some special motions were added which were related to the sounds. None of manual mode would be possible without the commands sent from the external GUI.

**GUI**

When the GUI was developed user experience was a primary factor in the design process. When designing the GUI to control the bear, user experience was considered to be the most important aspect of the design. Whenever a user would start using the GUI it was a requirement that no effort would be needed for them to know how to use and what to do to get a requested
A Windows Forms GUI design was determined to be the most familiar to users. From this decision C# was chosen as the language to use to program the GUI. Utilizing C# allowed direct manipulation of a Windows Forms environment. Figures 14 and 15 show a depiction of the two modes of operation for the GUI Basic and Advanced.

![Figure 14: Basic GUI](image-url)
When using the GUI every time the user clicks a button that is associated with a specific motion, the bear responded by performing a predefined action. To facilitate communication to the bear a Serial Port is being utilized to communicate with the Arduino Mega via USART communication. The figure 16 is a code snippet from the class that defines serial communication with the Arduino.

![Advanced GUI](image)

**Figure 15: Advanced GUI**
using System.IO.Ports;

namespace BearControlInterface
{
    class ArduinoCommunication
    {
        #region Class Variables
        readonly SerialPort _port;
        #endregion
        #region Constructors
        public ArduinoCommunication(String address)
        {
            _port = new SerialPort(address, 9600, Parity.None, 8, StopBits.One) { Handshake = Handshake.None };
        }
        #endregion
        #region Methods
        public Boolean OpenInit ()
        {
            try
            {
                _port.Open();
                _port.WriteLine("IDN?\n");
                var str = _port.ReadLine();
                return str.Equals("Bears rule!\r");
            }
            catch
            {
                return false;
            }
        }
        public Boolean Open()
        {
            try
            {
                _port.Open();
                return true;
            }
            catch (Exception)
            {
                return false;
            }
        }
        public void Close ()
        {
        }
    }
}
try
{
    _port.Close();
}
catch
{
}

public String Read ()
{
    return _port.ReadLine();
}

public void Write(String msg)
{
    _port.WriteLine(msg);
}

public Boolean DataAvailable()
{
    return (_port.BytesToRead - _port.ReadBufferSize) > 0;
}

#endregion

Figure 16: Serial Port Class for Communication with Arduino

Within this class are several methods that either open, close, read, or write to the serial port. Initializing the system serial port required information on the port number used, the parity bit, number of stop bits, and baud rate of the associated serial port needed to be known.

The next class in the GUI was the class that does the bulk of work within the GUI. For the GUI design each item was dragged and dropped through the Visual Studio editor. Each item was then given a name corresponding to what it was IE ChangeModeButton. The code below is only part of the class. A lot of redundant code has been omitted because of repeated logic. Also a lot of the conditional statements have been omitted because they also are repetition of the same logic. The only special using reference for this code is for Windows Media Player. The rest are the standard using statements used for Windows Forms environments.
using WMPLib;
namespace BearControlInterface
{
    public partial class BearInterface : Form
    {
        private void ModeChosen(object sender, EventArgs e)
        {
            if (BasicModeButton.Text.Equals("Auto"))
            {
                //Enter Auto mode
                //disable all buttons IE:
                BasicWaveRtButton.Enabled = false;
                AdvancedModeButton.Text = @"Manual";
                BasicModeButton.Text = @"Manual";
                outputListBox.Items.Add(@"Automatic Mode selected.");
                //send A to arduino for change mode
                _ac.Write("A");
            }
            else
            {
                //Enter Manual Mode
                // enable all buttons IE:
                BasicWaveRtButton.Enabled = true;
                AdvancedModeButton.Text = @"Auto";
                BasicModeButton.Text = @"Auto";
                outputListBox.Items.Add(@"Manual Mode selected.");
                _ac.Write("M");
            }
        }

        private void PlaySound(object sender, EventArgs e)
        {
            _timer.Reset(); //reset timer so auto mode is not enabled
            //how to play a sound
            if (sender.Equals(soundButton1) || sender.Equals(BasicSound1Button))
            {
                outputListBox.Items.Add(@"Pooh 1 Selected.");
                var a = new WindowsMediaPlayer { URL = @"c:\Sounds\pooh1.mp3" }; a.controls.play();
            }
        }
    }
}
private void PerformMotion(object sender, EventArgs e)
{
    _timer.Reset();
    // perform a motion
    if (sender.Equals(WaveLeftButton) || sender.Equals(BasicWaveLtButton))
    {
        outputListBox.Items.Add("Waving left arm.");
        _ac.Write(String.Format("JWL{0}\", (Char)0));
    }
}

private void ProgramStart(object sender, EventArgs e)
{
    _commWorker = new BackgroundWorker { WorkerSupportsCancellation = true }; _commWorker.DoWork += commWorker_DoWork;
    int i;
    for (i = 1; i < 21; i++)
    {
        _ac = new ArduinoCommunication(string.Format("COM{0}\", i.ToString(CultureInfo.InvariantCulture)));
        if (_ac.OpenInit())
        {
            CommPortLabel.Text = string.Format("COM {0}\", i.ToString(CultureInfo.InvariantCulture));
            _ac.Close();
            outputListBox.Items.Add(string.Format("Com Port set to COM {0}\", i.ToString(CultureInfo.InvariantCulture)));
            break;
        }
    }
    if (i >= 21)
    {
        /* No comm port found */
        MessageBox.Show("No communication with the arduino could be established exiting the GUI");
        Application.Exit();
    }
    _timer = new Stopwatch();
    _commWorker.RunWorkerAsync();
    BasicPanel.BringToFront();
}

void commWorker_DoWork(object sender, DoWorkEventArgs e)
{
    try
if (_ac.Open()){
    while (true) {
        if (BasicModeButton.Text.Equals("Auto")) {
            // we are in manual mode
            if (_timer.Elapsed.Minutes >= 3) {
                // too long in manual reset to auto
                BasicModeButton.PerformClick();
            }
            Thread.Sleep(TimeSpan.FromSeconds(1));
        }
        if (_ac.DataAvailable()) {
            String s1 = _ac.Read();
            // outputListBox.Items.Add(s1);
            if (s1.Equals("HLO\r")) {
                // hello greeting needs to be played
                var a = new WindowsMediaPlayer { URL = @"c:\Sounds\yogi1.mp3" }; a.controls.play();
            }
        }
    }
}

} catch (Exception) {
/* Comm failed */
    MessageBox.Show(@"Communication with the arduino failed exiting the GUI");
    Application.Exit();
}

private void GuiModeChossen(object sender, EventArgs e) {
    if (GuiModeButton.Text.Equals("Basic"))
    {
    }
This class has several items of note. One is the BackGroundWorker. This is how C# allows multi-threading. The check to see if the Arduino has sent or received messages is taken off the UI thread in order to keep that thread solely on the UI changes. Since the checking thread is not changing items on the UI thread no delegates need to be declared. If however, future changes are made any action taken on the UI thread must go through a delegate to invoke the item being modified to make that call thread safe. Another item to note is how sounds are played. A Windows Media Player object is initialized and used to play the sound right from the computer running the GUI. This is why a speaker was placed in the bear's mouth to play said sounds. The ProgramStart method is the first item called when this form is instantiated. The backgroundworker is created and communication to the Arduino is started. If no communication can be confirmed the GUI exits.

With all the controls routed the user has the option of placing the bear in auto or manual mode. The difference of the two modes is that in auto a preset motion tree is followed and repeated when a user is detected in front of the bear or until manual Mode is entered. In manual
mode the user was able to perform a host of motions or play a specific sound that corresponded to the performance of a specific motion also perform a motion. In the basic mode of operation the GUI has simple depictions of the motions to be performed. As well as the sounds were written out. In advanced mode there is a more finite control of the bear through sliders for each joint and buttons for sounds and motions. These buttons perform the same thing as those in the basic mode. They all point to the same listener.

Conclusion

Overall, the project was a success. While not all of the original ideas made it into the final project, the basic design held throughout. Throughout the course of the project, there were several changes that could have been made. Firstly, the skeleton would be designed and wired in a manner that allowed for easier adjustments to the individual pieces of pipe. When a limb was determined to be too long, it required a lengthy process to extricate the skeleton from the skin, unhook the servos on that limb from the rest of the system, remove the wires from the limb itself, alter the limb, hook everything back together, and stuff the skeleton back in the bear. The process is even more convoluted when a piece of the skeleton had to be replaced. Secondly, the amperage requirements for the servos would be determined before a complex power board was built. This would have saved time by making it clear that the regulators available were not sufficient for the project, causing a more suitable power arrangement to be sought out. Two specific lessons that have been learned through the course of this project. The first is the saying “Ground is ground is ground.” While it might be grammatically dubious, this phrase communicates an important part of electrical engineering. Namely, all of the ground planes on a project must be connected together, into a common ground, so to speak. Once that has been
done, it does not matter to which ground plane a component is attached, the ground is the same everywhere. The second lesson is commonly referred to as Murphy’s Law: “Anything that can go wrong, will go wrong.” In the day before the university Open house, a connection between the left foot piece and the servo was broken and a trace was broken on the power board. A few minutes before the Open House was scheduled to begin, the servo for the right shoulder broke completely. Fortunately, all of these obstacles were overcome, but they highlight the necessity of designing for the worst-case scenario. In closing, the creators of this project encourage others to try their hand at designing and building robotic bears.
References:


http://www.soundboard.com.sb/Yogi_Bear_sounds Yogi Bear Soundboard

http://arduino.cc/en/Main/arduinoBoardMega Arduino Mega Data Sheet


TO-220 7805 Voltage Regulator

http://www.gideontech.com/content/articles/196/1 PSU Modification

www.stackoverflow.com C# code help


Maxbotix EZ4 Ultrasonic Rangefinder

www.arduino.cc Arduino code help