# Table of Contents

**Project Overview** .......................................................................................................................... 4
  Problem Background and Needs Statement .................................................................................. 4
  Project Accomplishments ............................................................................................................. 4

**System Design Overview** ........................................................................................................... 5
  System Block Diagram – The Four Parts ..................................................................................... 5
  System Block Diagram – The Interfaces .................................................................................... 6

**Complete System Specifications** .............................................................................................. 7
  Part 1:  The Computer ................................................................................................................. 7
  Part 2:  The SV203 Microcontroller ............................................................................................ 18
  Part 3:  The Yano I/O Board ........................................................................................................ 22
  Part 4:  Yano ............................................................................................................................... 26

**Appendices** ................................................................................................................................... 28
  Appendix A:  Software Class and Method Outline ................................................................. 28
  Appendix B:  Hardware Logic Flow .............................................................................................. 33
  Appendix C:  Product Datasheets ............................................................................................... 33
  Appendix D:  Bibliography .......................................................................................................... 35
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>System Block Diagram</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2</td>
<td>The Computer</td>
<td>7</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Yano Control Center</td>
<td>8</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Parameterized Facial Expressions Screen</td>
<td>9</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Manual Motor Control Screen</td>
<td>10</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Sound Processing Screen</td>
<td>11</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Original Waveform</td>
<td>13</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Power Waveform</td>
<td>14</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Power Waveform Regions</td>
<td>15</td>
</tr>
<tr>
<td>Figure 10</td>
<td>The SV203 Microcontroller</td>
<td>18</td>
</tr>
<tr>
<td>Figure 11</td>
<td>SV203 Pin-out</td>
<td>21</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Yano I/O Board</td>
<td>22</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Yano I/O Circuitry</td>
<td>22</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Sensor Input Switch</td>
<td>23</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Yano I/O Board Pin-out</td>
<td>25</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Yano</td>
<td>26</td>
</tr>
</tbody>
</table>
Project Overview

Problem Background and Needs Statement
There has been a great deal of research done in the area of human-robot interaction to understand how a person interacts with a computer. Nonverbal communication, such as facial expression, provides us with characteristics about the expresser and emotion that is otherwise lost in flat speech alone. The ability to generate animated facial expressions together with speech is greatly important to many diverse application areas. For example, a deaf person could use an animated face as a lip-reading system. An autistic child could be positively affected from a robotic face in terms of social interaction, language development, and learning through structure and repetition.

Project Accomplishments
Our team delivered a computer-controlled robotic head taken from a Yano toy. Our final product contains all the necessary hardware, software, and user interface tools that are needed for its control.

The overall goal of this project was to create a robotic face capable of displaying human emotion accompanied with speech. Our final product accomplishes the following objectives:

1. We reverse-engineered Yano’s motors and sensors so we are able to move them to any desired position.
2. We developed a Windows-based graphical user interface that allows the user to move each motor in both directions to a desired position.
3. We researched the psychology behind the use of facial expressions to convey emotion and mimicked these facial expressions with Yano’s face.
4. We developed a graphical user interface that allows the user to select and display real human facial expressions.
5. We developed software to mimic speech with Yano’s mouth based on a measure of intensity from various wave files.
System Design Overview

The following is a general, functional description of each part and interface in our system. The details for specific software description, circuits, and pin-outs of the parts and the timing and logic of the interfaces are given later in the section titled, “Complete System Specifications.”

System Block Diagram – The Four Parts
1. Computer – this part is the computer from which Yano is controlled. System requirements are not great, but it should have at least a 450 MHz processor and 128 Meg. RAM. It must also have a serial port with which to communicate, and the libraries to run our program, built with Microsoft’s Visual C++ .NET.
2. SV203 Microcontroller – this part is the microcontroller we use to communicate with the computer via the serial port. It is designed by Pontech Inc. as a motor control board. It has
the ability to decode commands from the serial port, and from these set 8 digital motor 
control pins. It can also read in 5 analogue input pins and transmit their values back up the 
serial line.

3. Yano I/O Board – this circuit is designed by our team. It receives digital motor controls, and 
amplifies these to provide power and direction directly to Yano’s motors. It also interfaces 
with Yano’s motor limit switches and sends a TTL high or low signal back to the 
microcontroller indicating whether each switch is open or closed.

4. Yano – this part is the actual Yano toy. Yano can form many different expressions on his 
face by moving his eyelids, eyebrows, ears, cheeks, and mouth. The movement of the 
eyelids, eyebrows, and ears are tied together on one motor; and the cheeks and mouth each 
move independently on their own 2 motors. Yano provides feedback on the current position 
of its motors by way of limit switches that are triggered when a motor reaches one extreme or 
the other.

System Block Diagram – The Interfaces
A. Serial Port Output – this interface is from the computer to the SV203 microcontroller. The 
computer sends commands out of its serial port in the form of ASCII text strings.
B. Microcontroller Output – this interface is between the SV203 microcontroller and the Yano 
I/O Board. The microcontroller outputs logic that contains the controls for Yano’s motors.
C. Yano I/O Board Output – this interface connects the Yano I/O Board to the actual Yano toy. 
It has direct control over the motors inside Yano and can either run them forward, run them 
backward, or hold them at stop.
D. Yano I/O Board Input – this interface gets feedback from Yano’s motor limit switches. It is 
used by the I/O Board to determine if Yano’s switches are closed or open.
E. Microcontroller Input – along this interface, the Yano I/O Board sends the status of the limit 
switches to the microcontroller. It transmits analog values of 0V or 6V to the 
microcontroller’s analogue inputs.
F. Serial Port Input – the computer uses this interface to retrieve data from the SV203 
microcontroller. The computer sends a request to the microcontroller via the Serial Port 
Output interface to get the values of its analogue input pins, and then the microcontroller 
sends these values in ASCII text back up the Serial Port Input interface to the computer.
Complete System Specifications

The following section contains a complete specification and detailed design of each subsystem specified above. This includes circuit and logic diagrams, interfaces and pin-outs, and software processes with their inputs and outputs. The will be presented in the order of data flow in our system, and according to the summary given above.

Part 1: The Computer

*Functional Description* – The user interface serves three major functions:

1. Allows the user to directly control the movement of Yano’s eyes, cheeks, and mouth motors.
2. Provides parameterized control of Yano’s facial expressions by allowing the user to both select from a predefined set of expressions and to control his expression in terms of valence, arousal, and stance.
3. Allows the user to load a pre-recorded wave file and play it back as Yano mimics human speech based on the intensity of the wave file.

*Display Descriptions* – There are four dialog displays that the end user will be able to view: the menu, the facial expression select screen, the manual motor control screen, and the sound processing screen. Each of these screens provides a different way to control Yano’s motors.
Figure 3: Yano Control Center

Main Menu Dialog (Yano Control Center) - This menu will allow access to the parameterized facial expressions screen, the manual motor control screen, and the sound processing screen. After the user presses the “Close” button any of those three screens he/she will be taken back to the main menu.
Parameterized Facial Expressions Screen – This interface allows the user to move between expressions as defined by the three axes of the emotional state space. The user may also create his or her own customized facial expressions by defining Yano’s state of arousal, valence, and stance.

The three axes can be defined as follows:

- **Arousal** – to stir up, excite; provoke; to awaken from.
- **Valence** – the degree of attraction or aversion that an individual feels toward a specific object or event.
- **Stance** – the altitude or position of a standing person; mental posture; point of view; a station, a position.
Of the emotional states shown in the upper right of Figure 4, the valence prototypes have the strongest influence on lip curvature. Arousal prototypes influence the “height” of the face: the ears, eyes, and eyebrows. Stance prototypes affect the position the least, but appear to influence the openness of the eyes and height of the cheeks.

The user can move the sliders and the program will highlight the emotional state and check the box of the expression as one is reached. Alternatively, the user may check the box of the desired expression, and the sliders will move and highlight the emotional state accordingly.

**Figure 5: Manual Motor Control Screen**
Manual Control Screen - The manual control screen allows the users to move the motors to specific positions. Motor positions are shown using three progress bars and six end switch indicators (each motor has two end switches). The current and max positions are shown next to these progress bars as well. The user can either change the motor positions using the scroll bars at the bottom, or using the text boxes and the “Go” button. This screen also provides three calibrate buttons (one for each motor).

Sound Processing Screen – This screen uses voice/speech intensity to determine how Yano’s face will look as prerecorded speech is being played. The file name, file progress, and current intensity are also displayed.

Sound Analysis Software Considerations – We thought of two different ways to do the sound analysis. The first method was to monitor the sound card device, and to do real-time analysis on the data being sent to the sound card. The second method was to take a WAVE file, do the analysis ahead of time, and then play the sound.

The first method would be ideal because it would allow any format of file, and it would make real-time analysis easier. Unfortunately, all our research into this methodology was fruitless.

Because of this, we decided to go with the second method. WAVE files are supposed to be the easiest to parse since they only consist of a header and raw data, but the WAVE format is a fairly fragmented standard. It was hard to find good examples on the web, but eventually we found something that worked.
Sound Analysis Software Strategy – Our strategy is to skip the header, and assume the input data is a 16-bit mono .wav file. Then we can proceed to read samples one 16-bit word at a time.

The following is the algorithm to calculate the intensity of the WAVE file:
1. Calculate the sum-of-squares of $n$ samples and store in A.
2. Calculate the sum-of-squares of the next $n$ samples, and store in B.
3. Calculate the sum-of-squares of the next $n$ samples, and store in C.
4. If there are no more samples to be process, terminate analysis.
5. Add A, B, and C together, and divide by $(6n)$. This is the intensity.
6. $A = C$
7. Go to 2

A, B, and C form one window of $6n$ samples. The purpose of step 6 is to overlap the windows by 1/3 their size.
The following two figures provide an example of this algorithm on a small sample WAVE file.

Figure 7: Original Waveform
Figure 8: Power Waveform
Mouth Movement Algorithm – After calibration we can determine the maximum number of pulses needed to open or close Yano’s mouth. Because this number is usually 3 pulses, we decided to create 3 different regions in the power waveform. These are based on a percentage of the maximum intensity of any given waveform.

Figure 9: Power Waveform Regions

Region 1 is between 3% and 20%, region 2 is between 20% and 40%, and region 3 is anything above 40%. Anything below 3% is considered “noise” and should be ignored.

After defining these regions we developed the following algorithm:

1. If the audio is below region 1 then close the mouth. This is done by pulsing the motor to position 0.
2. If the audio is in region 1 then pulse the mouth 1/3 open. This is done by pulsing the motor to position 1. If the mouth is already at position 1 then pulse it to position 0.
3. If the audio is in region 2 then pulse the mouth 2/3 open. This is done by pulsing the motor to position 2. If the mouth is already at position 2 then pulse it to position 1.
4. If the audio is in region 3 then pulse the mouth all the way open. This is done by pulsing the motor to position 3. If the mouth is already at position 3 then pulse it to position 2.
NOTE: These positions are based on a percentage of the maximum number of pulses, and the maximum value of the power waveform. This allows the same algorithm to work with many different WAVE files. This algorithm is a simplified version of this idea.

The above algorithm basically opens the mouth to the proper position and oscillates until a new position is found. The mouth then moves to the new position and begins oscillating again.

Class Descriptions – There are eight main classes being used for the Yano project: CComm, YanoEngine, ManulControlDlg, YanoAdvancedDlg, MenuDlg, SoundProcessingDlg, Motor, and WaveOut. The most important of these classes is YanoEngine. See Appendix A for a detailed class and method outline.

YanoEngine (.cpp) - The most important class for this project. Provides the code necessary for adjusting the motors to the correct positions. It uses an instance of both CComm (used for serial I/O) and Motor (each motor is represented by an instance of this Motor class). The most important method contained in YanoEngine.cpp is adjustMotors(long destX, long destY, long destZ).

CComm (.cpp) - Used by YanoEngine to communicate with the microcontroller. Used for I/O over the serial cable.

Motor (.cpp) – Every motor is represented by an instance of this class. All instances of Motor are created and controlled by the same instance of YanoEngine.

MenuDlg (.cpp) – Creates the only instance of YanoEngine, CYanoDlg, and ManualControlDlg. Contains two buttons that open their corresponding dialogs. Later a third button will be added for the speech control dialog.

YanoAdvancedDlg (.cpp) – Contains nine buttons to change the facial expressions of Yano. Uses the YanoEngine class to control the motors.
ManualControlDlg (.cpp) – Allows the user to change the motor positions manually. Contains text box input and scroll bars for each individual motor. Also contains progress bars and end switch indicators. Once again this screen uses the YanoEngine class to adjust the motor positions.

SoundProcessingDlg (.cpp) – Analyzes wave file input and adjusts the motor positions accordingly. This dialog will uses the YanoEngine class for motor control and the WaveOut class for opening and playing audio. It allows Yano to mimic actual speech.

WaveOut (.cpp) – Used to open and play the wave file. This class uses CWave.cpp, CWaveBuffer.cpp, and CWaveDevice.cpp.
Part 2: The SV203 Microcontroller

Functional Description – The SV203 is the core microcontroller in our design. It communicates bi-directionally with the computer through the serial port to receive commands in ASCII text that dictate what it should do.

Circuit Description – This microcontroller is designed by Pontech Inc. and was purchased to facilitate the serial port communications with the computer. The details of the circuits that make the SV203 work are beyond the scope of this project. However, the important parts of the SV203, that are relevant to our system design, are described here, by their label in the above schematic.

J4. Motor Control Port – In our design, the Microcontroller Output Interface connects to the leftmost pin (the logic pin) of each 3-pin, horizontal group, and these are the only pins of interest to us in the port. These pins are the square ones next to the labels S1 through S8 on the schematic.

J3. A/D Input Port – The Microcontroller Input Interface connects to this port. They are numbered from the top left (square) pin from left to right and top to bottom, AD1 through AD5.
PWR1. **Power Port** – This is the power connection for the SV203. It runs off of the same 6-volt power supply that everything in our design does.

**J1. Serial Port** – This is the serial port connection for the SV203. The Serial Input and Serial Output interfaces use this port for communication between the computer and the SV203.

*Interface Description* – Seeing as how we did not create this microcontroller, the interesting parts, and the parts that must be fully specified for our design are the interfaces used on the SV203. This is a very versatile microcontroller and there are many ways it could be used, but we will only describe the specific ways we interface it in our system.

**Serial Port** – this port is used to interface the SV203 with the computer. The computer sends commands to the SV203 via this interface using RS232 communication standards and sending ASCII text as the data. We do not need to understand the details of the serial communication protocol, but we will specify the commands we use and what they do.

As stated, the commands sent are just text. They consist of one or two letters specifying the command, followed by a number specifying which pin it affects. Multiple commands can be strung together, and will be executed nearly simultaneously, up to 80 characters in length. All strings of commands must finish with a ‘\r’ or return character. Here is a list of the commands we use.

- **SVxM0** – where \( x \) is a number (1-8) of the desired motor control pin. This tells the SV203 that pin ‘\( x \)’ will be controlled with digital logic instead of specifying a waveform to broadcast out of it. The result is that we are now allowed to run the Pin Set and Pin Clear commands on this pin.

- **PSx** – sets motor control pin ‘\( x \)’ to TTL high (6V).

- **PCx** – sets motor control pin ‘\( x \)’ to TTL low (0V).
Dn – causes the board to delay any action for ‘n’ milliseconds. While the board is in delay mode, it will not be able to receive any new commands from or send anything out of the serial port.

PC1PC3PC5D300PS1PS3PS5 – a typical command we use to set the motor control pins. This one would cause pins 1, 3, and 5 to be set low, then 300 milliseconds later, they would return to high.

ADy – reads the value of the A/D Input Port and sends it back along the serial communication line.

**Motor Control Port** – This port sends the logic values for motor control to the Yano I/O Board. We specifically use the pins SV1 through SV6 only. These pins are initialized to be used for digital output and initially all set to high (6V). They are set by the microcontroller to either high (6V) or low (0V) when it receives a command for Pin Set (PS) or Pin Clear (PC) with the corresponding number afterward. The logic for which pins control each motor falls to the Yano I/O Board, and will be discussed later.

**A/D Input Port** – This port is also connected to the Yano I/O Board, but it receives input from the board instead. The data it receives is an analogue voltage, but used in a digital fashion. It will receive a high input (6V) if the limit switch it corresponds to is open, and a low value (0V) if the switch is closed. This port is read by the SV203 when it receives the A/D Port Read command (AD). The SV203 will transmit a value (0-255) back up the serial port indicating the voltage (0V-5V) on that particular input pin.
Pin-out – see the figure below for the pin layout of the SV203 microcontroller.

Figure 11: SV203 Pin-out
Part 3: The Yano I/O Board

Functional Description – The Yano I/O Board takes the logical motor control signals from the SV203, and uses them to power Yano’s three motors directly, in the desired direction. It also checks Yano’s motor limit switch sensors with some simple switching circuitry and sends back to the SV203 the status of each switch, open or closed.

Circuit Description – The circuitry in the Yano I/O Board is pretty simple, but very necessary. It is basically divided into 2 sections:

Motor Control – the Motor Control Port connects with the SV203 Microcontroller and receives logic controls for the motors. It gets 3 pairs of signals and runs each pair into a separate TC4424 H-Bridge. The purpose of the H-Bridge is to provide the power that the logic lines cannot. It takes two digital signals as input, and outputs to two pins either Vcc or Gnd, these pins connect to the ports that Yano’s motor plugs into.

Sensor Input – this circuit may look a little random on the schematic above, but what it boils down to is that the following circuit is created for each switch:

Figure 12: Yano I/O Board

Figure 13: Yano I/O Circuitry
So when Yano’s Switch is open, the Sensor Output will read a full 6V. But when the switch is closed, the Sensor Output will be bridged to Gnd and read 0V. There are five of these same switches on Yano, two for each limit of the mouth and cheek motors, and one for the eye motor to indicate when it reaches a certain point in the rotation. The eye motor is different because it doesn’t have limits like the other two; it can spin a complete 360 degrees and keep going without problem.

*Interface Description* – the Yano I/O Board has quite a few different interfaces. It has two interfaces with the SV203 Microcontroller, the Motor Control Input from the microcontroller, and the Sensor Output that it sends back to the microcontroller. It also has a number of interfaces with Yano, including three motor outputs and three sensor inputs.

**Motor Control Input** – This is the input port for the interface with the SV203. It has 6 pins, SV1 through SV6, corresponding to the same pins on the microcontroller. These pins will be either high (6V) or low (0V) depending on what the motors should do. They are paired off, two for each of the three motors. The mouth motor is controlled by SV5 and SV6, the cheeks by SV3 and SV4, and the eyes by SV1 and SV2. This port is the logical input for the H-Bridges that supply the power to the Motor Outputs that go to Yano. See the logic section for more details on the logical specifications.

**Motor Outputs** – There are three of these output interfaces, one for the eyes, cheeks, and mouth motors. Yano’s motors connect directly to these, so they source enough current to run the motors. They consist of two pins each, either one can have Vcc or Gnd on it. If both have the same value, the motor connected to it will not turn. If pin 1 had Vcc and pin 2 has Gnd, the motor will turn, and if it’s vice-versa, then the motor will turn the opposite direction.
Sensor Inputs – There are three of these interfaces that connect directly to Yano’s motor limit switches. The mouth and cheek switches behave the same way, and they eye switch is a little different. The mouth and cheek interfaces have three pins each. One pin connects to one side of both switches, and the other two pins connect to the opposite side of each switch. So for each motor (mouth and cheek) there is two switches, one for each extent (ie. moth fully open or fully closed, and cheeks fully up or fully down). For the eye motor, it can spin a full 360 degrees, so its sensor just marks a position around that rotation, so that the eyes can be calibrated to start at the same place. It does this with an infrared diode / infrared transistor pair located on opposite sides of a gear on the eye motor. The gear has a single hole in it, so that when it lines up between the diode and transistor, the infra red light triggers the transistor to allow current to flow. This sensor is interfaced by providing power to the LED and retrieving the state of the transistor in the same way as the other switches.

Sensor Output – This interface goes back to the SV203 microcontroller and contains data about Yano’s five motor sensor switches. They are numbered AD1 through AD5 and correspond to the same numbers on the SV203. The default state of the switches is open, and when that is the case, the Sensor Output port sends high (6V) back to the SV203. When a switch is closed, low (0V) will be sent for that switch. The mouth switches use AD3 and AD4, the cheeks use AD1 and AD2, and the eye switch is on AD5.
Pin-out – see the figure below for the pin layout of the Yano I/O board.

Figure 15: Yano I/O Board Pin-out
Part 4: Yano

Functional Description – This is the actual Yano toy. Yano’s face can move his eyelids, eyebrows, ears, cheeks, and mouth. He can move all of these at the same time to convey expressions, emotions, or even to make the motions of speaking. In our project, we control Yano to create expressions on his face that convey a wide range of emotions. It is also our goal to give him the ability to move his mouth in speaking motions along with someone talking into our software.

Circuit Description – Unfortunately for Yano, we tore out almost all of his internal circuitry at the beginning of our project. Our goal was to replace all of it with our own controls from the computer. However, we did leave the essential parts of the face control in tact. That is, we left the three motors that control his face, and the sensors on each motor that indicate when limits are reached. Each motor system is described below:

Eyes – Of the three motors, one controls the eyelids, eyebrows, and ears all at the same time. This motor can turn a complete 360 degree rotation and keep going; it has no limits on it. Each of the 3 facial features connected to this motor have a gear on them enabling them to move in different ways as the motor turns. As the motor makes one revolution, the eyes will open and close a few times, the ears will wiggle up and down quite a few times, and the eyebrows will move periodically. So the hardest work comes in finding out where the motor is, so we know which features will move and how when we turn the motor one way or the other. To help with this, there is an infrared light and receiver positioned on either side of a gear on this motor that has a hole at one place, so the infrared sensor will go off only when the motor is at one specific position. See the waveforms section for a more detailed description of how the features move as the motor turns.

Figure 16: Yano
Cheeks – The cheek movement is much simpler than the eyes. It simply involves some mechanics attached to a single motor that moves in one direction to raise his cheeks and the other to lower them. This motor, however, does not go all the way around, and has a limit switch at either extend (fully up and fully down) to indicate that the motor has reached the limit.

Mouth – The mouth motor works the exact same way as the cheeks, but its movement opens and closes the mouth. The switches for these two motors are very simple. They are just two pieces of metal separated that are pushed together by part of the plastic mechanics attached to the motors, thus closing the switch.

*Interface Description* – Yano has two main interfaces. Three that receive power for each motor, and three that sends back the status of the three motor’s limit switches. The motor interfaces simply will receiver either Vcc or Gnd on each of the two pins for any particular motor. The switch interfaces simply bridge together a unique pair of pins when one of the switches is closed. Details can be found below in the pin-out description.

*Pin-out* – the table below describes the colors of the wires corresponding to each of Yano’s motors and sensors.

Motors: Red/Black – Eyes  
Green/Black – Cheeks  
White/Black – Mouth

Sensors: Red/Green/Brown – Mouth  
Gray/Yellow/Pink – Cheeks  
Green/Yellow/Red/Brown – Eyes

See *Appendix B* for the complete hardware logic flow.
Appendix A: Software Class and Method Outline

a. CComm (.cpp)
   i. Used by YanoEngine to communicated with the microcontroller. Used for I/O over the serial cable.

b. YanoEngine (.cpp)
   i. This class is the core of the user interface. It does all the “work”.
   ii. initialize()
      1. Sets up the serial connection to the microcontroller
         a. Calls CComm::CComm(COM1, 9600) to set up the connection
      2. Sends commands to the microcontroller to enable digital i/o
         a. CComm::SendStr("SV1 M0\r");
            Sleep(3);
            CComm::SendStr("SV2 M0\r");
            Sleep(3);
            (continued)
         b. The software must allow 3 milliseconds for the microcontroller to process the commands.
      3. Creates three instances of the Motor class
   iii. shutdown()
      1. Closes the serial connection
         a. CComm::Close()
      2. Deletes the three instances of the Motor class
   iv. GetMotorXPos()
      1. Returns the position of motor X
   v. GetMotorXMasPos()
      1. Returns the maximum possible position of motor X
   vi. CalibrateMotor(int motorNumber)
1. Calls Motor::calibrate() on the specified motor
2. Actual calibration is done by the Motor class.

vii. conveyEmotion(long destX, long destY, long destZ)
1. Calls adjustMotors(long destX, long destY, long destZ)

viii. adjustMotors(long destX, long destY, long destZ)
1. This is the most important method for the entire UI
2. This method determines the current motor positions, desides which motors need to be adjusted, desides what directions the motors needs to be rotated, and begins to pulse the motors until they have reached the correct positions.
3. Pulses sent to the micro controller:
   a. Common command example
      i. CComm::SendStr("PC1PS2D200PS1PS2");
         Sleep(203);
      ii. The software must wait 200 milliseconds so the motor can turn and 3 milliseconds so the microcontroller can process the command.
      iii. This will clear pin 1 and set pin 2 for 200 milliseconds. After 200 milliseconds both pins will be set. This will rotate motor 1 counter clockwise for 200 milliseconds.
   b. Control Logic
      i. For example, if we want to open the mouth we clear SV5 and set SV6 (denoted by "PC5PS6"). This will cause the motors to rotate in the correct direction.
      ii. The diagram also shows the motor end switches and their corresponding voltages. If an end switch is hit the voltage will drop to a value less than 10.
   c. adjustMotors() will call CComm::SendStr() multiple times until all the motors are in the correct positions. Each call to SendStr() will pulse any number of motors for 200 milliseconds.
c. ManualControlDlg (.cpp)
   i. OnBnClickedCalMotorX()
      1. This calls YanoEngine::CalibrateMotor(X)
      2. Called when user presses one of the “Calibrate Motors” buttons
   ii. Other control methods call the usual YanoEngine::adjustMotors(long destX, long destY, long destZ) method
   iii. SetProgressValues()
      1. Sets the values of the three progress bars after the motors have been repositioned.
   iv. CheckEndSwitches()
      1. Sets the values of the end switch indicators.
      2. If an end switch has been triggered the checkbox will be checked.

d. YanoAdvancedDlg (.cpp)
   i. There are check boxes with corresponding facial expression images used to change the facial expressions of Yano.
   ii. The nine methods are OnBnClickedAccepting(), OnBnClickedAnger() and so on for the nine facial expressions.
   iii. Each of the nine “OnBnClicked” method calls YanoEngine::conveyEmotion(long destX, long destY, long destZ) which calls adjustMotors(long destX, long destY, long destZ)
   iv. There are three sliders controlling Yano’s valence, arousal, and stance by calling the method YanoEngine::conveyEmotion() when moved.

e. MenuDlg (.cpp)
   i. Creates the ONLY instance of YanoEngine, of CYanoDlg, and of ManualControlDlg.
   ii. Contains two buttons that open their corresponding dialogs
   iii. OnBnClickedManual()
      1. Displays the Manual Control Dialog (ManualControlDlg.cpp)
         a. See Figure 10
   iv. OnBnClickedFaces()
1. Displays the facial expression dialog (CYanoDlg.cpp)
   a. See Figure 9

f. **Motor (.cpp)**
   i. Every motor is represented by an instance of this class
   ii. All instances of Motor are created and controlled by the instance of YanoEngine created in MenuDlg’s constructor.
   iii. **Calibrate()**
       1. This method finds the motor’s maximum position and returns the motor to the first end switch (position 0).
       2. Runs the motor backwards until it hits an end switch, then it counts the number of pulses to hit the other end switch. This value is copied to m_maxPos.
   iv. **getCurrentPos()**
       1. Returns the current motor position (m_currentPos).
   v. **getDirection()**
       1. Returns the direction the motor should be run.
       2. setDirection(int destPos) must be called first.
   vi. **getMaxPos()**
       1. Returns the maximum position the motor can go to.
       2. calibrate() must be called first.
   vii. **needsAdjusting(long destPos)**
       1. Returns a true if the destination differs from the motor’s current location. Otherwise it returns a false.
       1. After the motor has been physically moved to a new position this method is used to set its new value in software. Usually follows a call to YanoEngine::adjustMotors(long destX, long destY, long destZ)

g. **SoundProcessingDlg (.cpp)**
   i. Controls Yano’s mouth based on an input wave file.
ii. Used to mimic actual speech.

iii. ProcessWaveform()
   1. Processes the wave file by computing the “sum-of-squares” power waveform.

iv. OnBnClLickedPlay()
   1. Plays the wave file and controls Yano’s mouth using the YanoEngine and WaveOut classes.
   2. Uses callback functions to update the display and to pulse Yano’s mouth motor.

v. OnBnClLickedOpen()
   1. Opens the wave file.
   2. Calls ProcessWaveform()

h. WaveOut (.cpp)
   i. Opens and plays a wave file.
   ii. Used by the SoundProcessingDlg class.
   iii. Contains basic wave file methods
       1. Open()
       2. Close()
       3. GetPosition()
       4. Play()
       5. Stop()
       6. IsPlaying()
   iv. Uses the following classes
       1. CWave
       2. CWaveBuffer
       3. CWaveDevice
Appendix B: Hardware Logic Flow

<table>
<thead>
<tr>
<th>Software</th>
<th>SV203</th>
<th>Yano I/O Board</th>
<th>Yano</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS1</td>
<td>SV1 to 5V</td>
<td>Eye - Red to 6V</td>
<td>Stop</td>
</tr>
<tr>
<td>PC1</td>
<td>SV1 to 0V</td>
<td>Eye - Red to 0V</td>
<td>Eye CW</td>
</tr>
<tr>
<td>PS2</td>
<td>SV2 to 5V</td>
<td>Eye - Black to 6V</td>
<td>Stop</td>
</tr>
<tr>
<td>PC2</td>
<td>SV2 to 0V</td>
<td>Eye - Black to 0V</td>
<td>Eye CCW</td>
</tr>
<tr>
<td>PS3</td>
<td>SV3 to 5V</td>
<td>Cheek - Green to 6V</td>
<td>Stop</td>
</tr>
<tr>
<td>PC3</td>
<td>SV3 to 0V</td>
<td>Cheek - Green to 0V</td>
<td>Cheeks Down</td>
</tr>
<tr>
<td>PS4</td>
<td>SV4 to 5V</td>
<td>Cheek - Black to 6V</td>
<td>Stop</td>
</tr>
<tr>
<td>PC4</td>
<td>SV4 to 0V</td>
<td>Cheek - Black to 0V</td>
<td>Cheeks Up</td>
</tr>
<tr>
<td>PS5</td>
<td>SV5 to 5V</td>
<td>Mouth - Red to 6V</td>
<td>Stop</td>
</tr>
<tr>
<td>PC5</td>
<td>SV5 to 0V</td>
<td>Mouth - Red to 0V</td>
<td>Mouth Open</td>
</tr>
<tr>
<td>PS6</td>
<td>SV6 to 5V</td>
<td>Mouth - Black to 6V</td>
<td>Stop</td>
</tr>
<tr>
<td>PC6</td>
<td>SV6 to 0V</td>
<td>Mouth - Black to 0V</td>
<td>Mouth Closed</td>
</tr>
</tbody>
</table>

| AD1 = 255 | AD1 at 5V | Cheek - Pink at 5V | Normal |
| AD1 = 0    | AD1 at 0V | Cheek - Pink at 0V | Cheeks Up |
| AD2 = 255  | AD2 at 5V | Cheek - Yellow at 5V | Normal |
| AD2 = 0    | AD2 at 0V | Cheek - Yellow at 0V | Cheeks Down |
| AD3 = 255  | AD3 at 5V | Mouth - Brown at 5V | Normal |
| AD3 = 0    | AD3 at 0V | Mouth - Brown at 0V | Mouth Up |
| AD4 = 255  | AD4 at 5V | Mouth - Green at 5V | Normal |
| AD4 = 0    | AD4 at 0V | Mouth - Green at 0V | Mouth Down |
| AD5 = 255  | AD5 at 5V | Eye - Green at 5V  | Normal |
| AD5 = 230  | AD5 at 3.8 V | Eye - Green at 3.8V | Eyes at Calibration |

Appendix C: Product Datasheets

SV203 Motor Controller Features:

- Controls 1 to 8 servos per board 8-bit resolution, value from 1 to 255,
- under one degree of servo position precision resolution
- Servo port can be reconfigured for digital output to drive on/off devices.
- Source/Sink 25 mA per pin
- Interface to PC through RS232 Serial port (2400 to 19200 baud).
- User definable board ID number (allowing multiple board to share same serial line).
- 5-Ch, 8-bit A/D input port for reading 0 - 5 Volts. (Control servo positions via Joystick/Pot)
- An SPI port for shifting in/out serial data
- Dimensions: 1.4 in X 1.7 in
- Servo Connectors: 3 pin sip. Futaba J-type connectors.
- Power supply: 7V (Up to 15V, but excessive voltage may damage servos)


TC4424 H-bridge Features:
- High Peak Output Current: 3A
- Wide Input Supply Voltage Operating Range:
  - 4.5V to 18V
- High Capacitive Load Drive Capability: 1800 pF in 25 nsec
- Short Delay Times: <40 nsec (typ)
- Matched Rise/Fall Times
- Low Supply Current:
  - With Logic ‘1’ Input – 3.5 mA (Max)
  - With Logic ‘0’ Input – 350 µA (Max)
- Low Output Impedance: 3.5Ω (typ)
- Latch-Up Protected: Will Withstand 1.5A Reverse Current
- Logic Input Will Withstand Negative Swing Up To 5V
- ESD Protected: 4 kV

Appendix D: Bibliography


   http://web.media.mit.edu/~stefanm/yano/

[3] TC4424 H-bridges
   http://www.microchip.com/1010/pline/analog/anicateg/power/mosfet/tc4_46/tc4424/