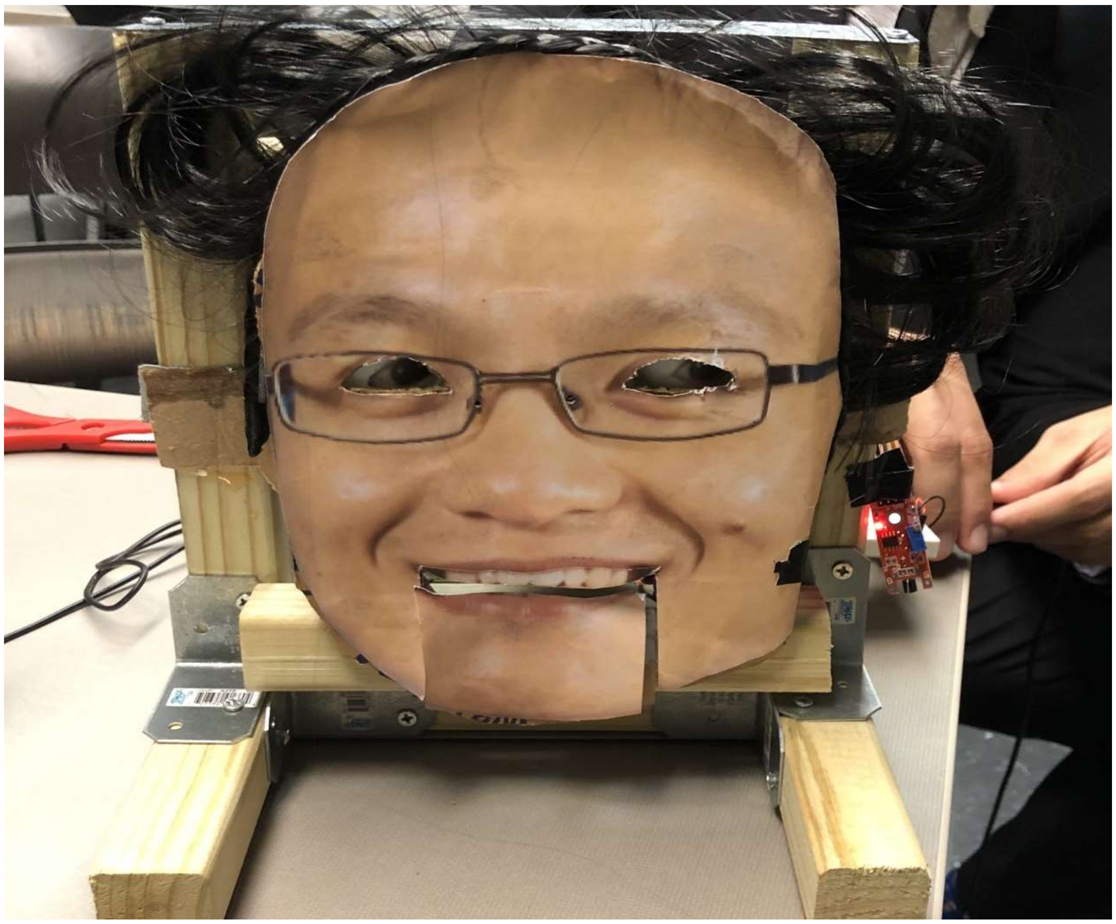
ANIMATRONICS FACE: DR. GAO



Carlos Quesada
Dept. of Mechanical Engineering
San Antonio, TX, USA 78249
cva308@my.utsa.edu

David Mata
Dept. of Mechanical Engineering
San Antonio, TX, USA 78249
djj105@my.utsa.edu

Nicolas Lara
Dept. of Mechanical Engineering
San Antonio, TX, USA 78249
cjx731@my.utsa.edu

Video link: https://youtu.be/YcXazh1cHIE

Submitted as part of project for ME4543 Mechatronics, Fall 2018

TABLE OF CONTENTS

ABSTRACT

The purpose of this project is to build an animatronic face that interacts with the user via sensors. The animatronic face built for this project was the face of Dr. Wei Gao, a Mechanical Engineering professor from the University of Texas at San Antonio. The goal of this project was to have a minimum of two moving parts with one being the mouth to resemble a human talking, and the other moving part to be determined by each team. For this project, the eyeballs were chosen to be the second moving part, both eyeballs having a rotational motion. The mouth and the eyes are both actuated by a total of four servo motors, two for the mouth and one per eyeball. Two sensors, one being a motion sensor and the other being a touch sensor, were used to trigger these servo motors. Each sensor activated the servo motors attached to the mouth and eyes to perform a short set of movements, while also having sound to represent the professor. Using the Arduino Mega for this project, the animatronic face was constructed and built to represent the professor with limited interactions. Although there were some difficulties in integrating the audio into the system, the animatronic face performed motions successfully and met the required specifications. The learning outcomes for the students was the ability to design a mechatronic system, carefully select sensors and actuators, the ability to program a microcontroller, and to successfully integrate these micro-controllers, sensors, actuators, to create a functional mechatronic system.

Section 1: Literature review

This section is composed of three different animatronic face designs found online to be used as a reference for the project at hand.

The first animatronic face design discovered is shown below in **Figure 1**. [1] This sophisticated design is composed of multiple servos installed throughout the entire assembly for multiple degrees of freedom and high range of movement. Eleven micro servo motors are used to emulate facial expressions involving the eyebrows, eyes, eyelids, and mouth. Two standard servos are used to control the neck, one for up and down movement of the head and other for left and right movement. The parts are precut from a solid wooden board for precision assembly. An Arduino microcontroller is used to control all the electronics and works in unison with an onboard software supplied by the manufacturer for servo calibration and facial movements. In addition, the assembly comes with an audible and motion editor for presetting recorded audio as an output to the assembly speaker. This assembly is not autonomous as movements are required through input hence the software provided, however, it provides an idea for constructing a similar face.

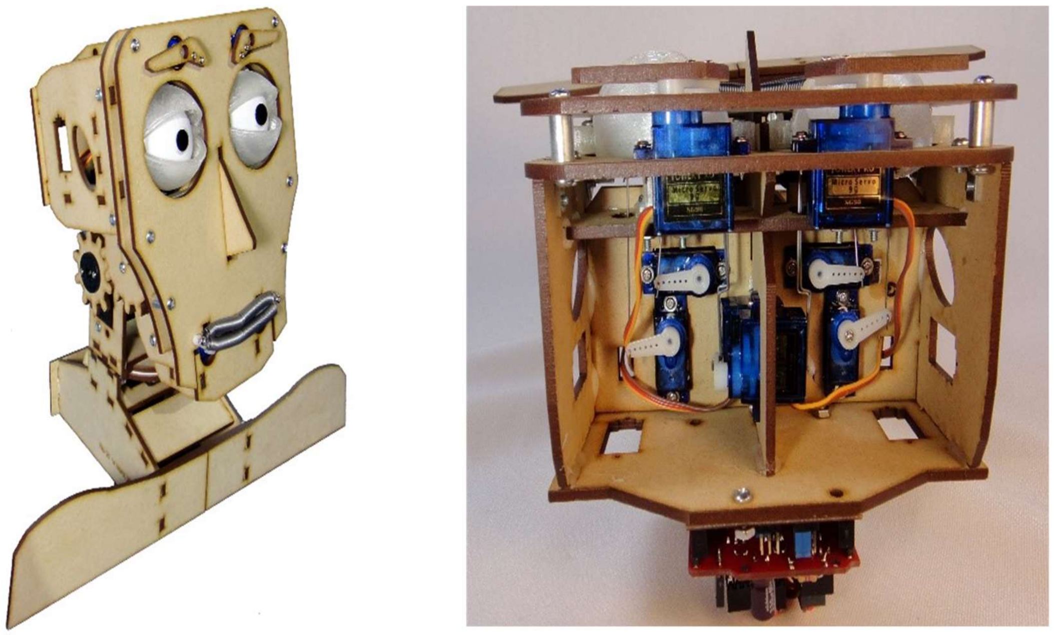


Figure 1. Animatronic Face Design One

The second animatronic face design found is part of a series of classes with master creature effects mechanic Craig Caton Largent and is shown in **Figure 2**. [2] This animatronic head consists of thirteen total servo motors; one servo for the jaw, two for the eyes (horizontal and vertical movement), two for the eyelids, two for the eyebrows, and six for the mouth (two for upper lip, two for bottom lip, and two for mouth corners). All servo motors are Hitec brand, Hitec HS-645MG, Hitec HS-82MG, and Hitec HS53 are the servo motors used. A clay mold is used to make the skull of the animatronic face and plywood is used to attach the servos together. This creation may give some help and guidance but is most likely not ideal for this project. This face goes multiple steps beyond by adding acrylic teeth, clay molding, and features for a Hollywood film, but the electronic setup could be useful.

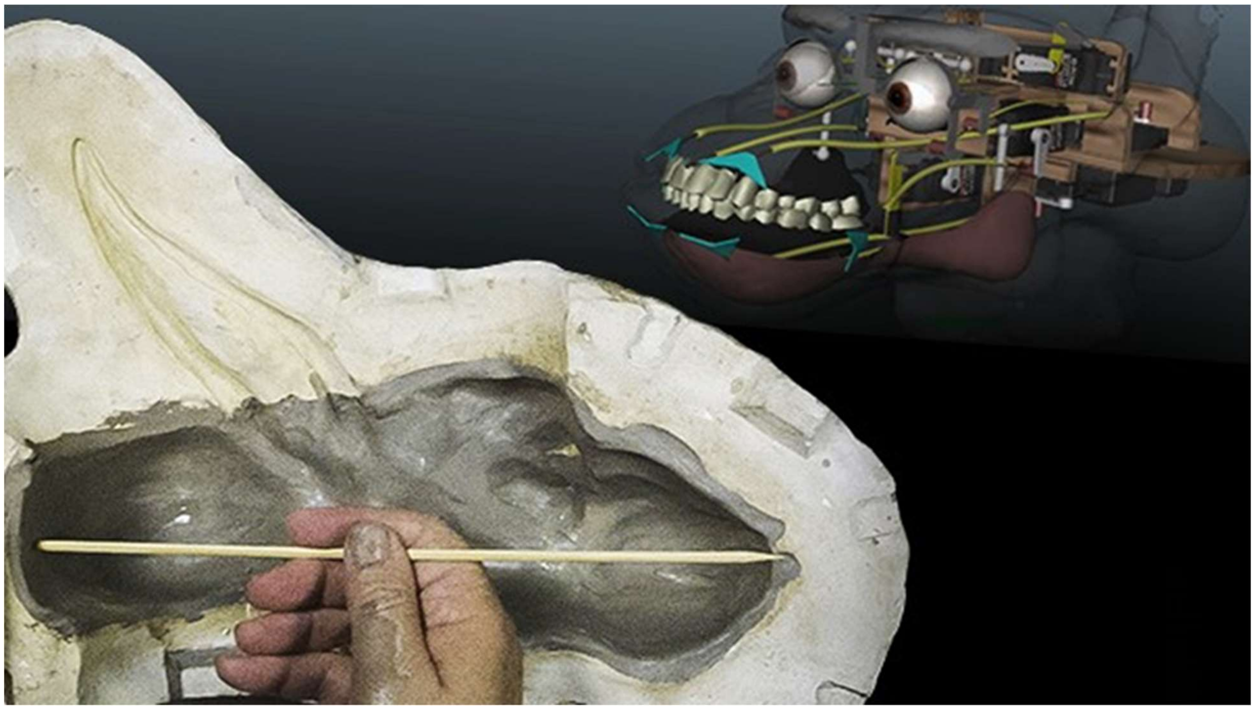


Figure 2. Animatronic Face Design Two

The third and final animatronic face design found online is more simplistic and can be seen in **Figure 3**. [3] This animatronic face was created to provide a detailed tutorial for those interested in recreating said face, or those who desire to use the given tutorial as reference for another project. Its functions include the movement of the eyes, eye brows, and mouth simultaneously while a voice recording plays as to imitate a person talking. The face was created with the use of the PIC18F452 microcontroller rather than an Arduino microcontroller. The movement of the face is actuated using seven servo motors. Four of the motors are used for the eyes, two are used for the eye brows, and one is used for the mouth movement. For this animatronic face, no sensors were used, rather movement was controlled completely through coding. Although, the simple design of the face structure does provide ideas that could be used for this project.

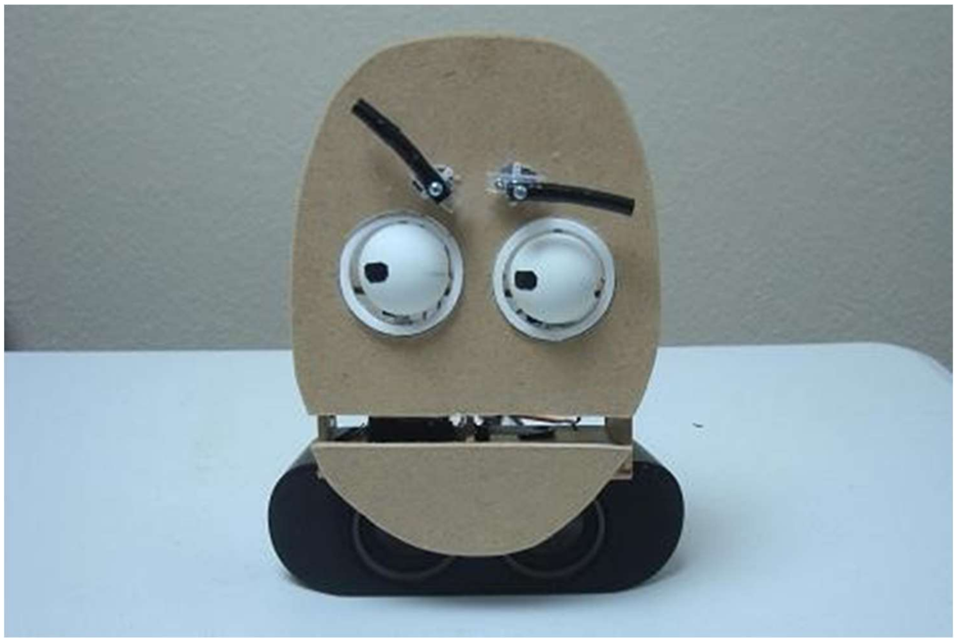


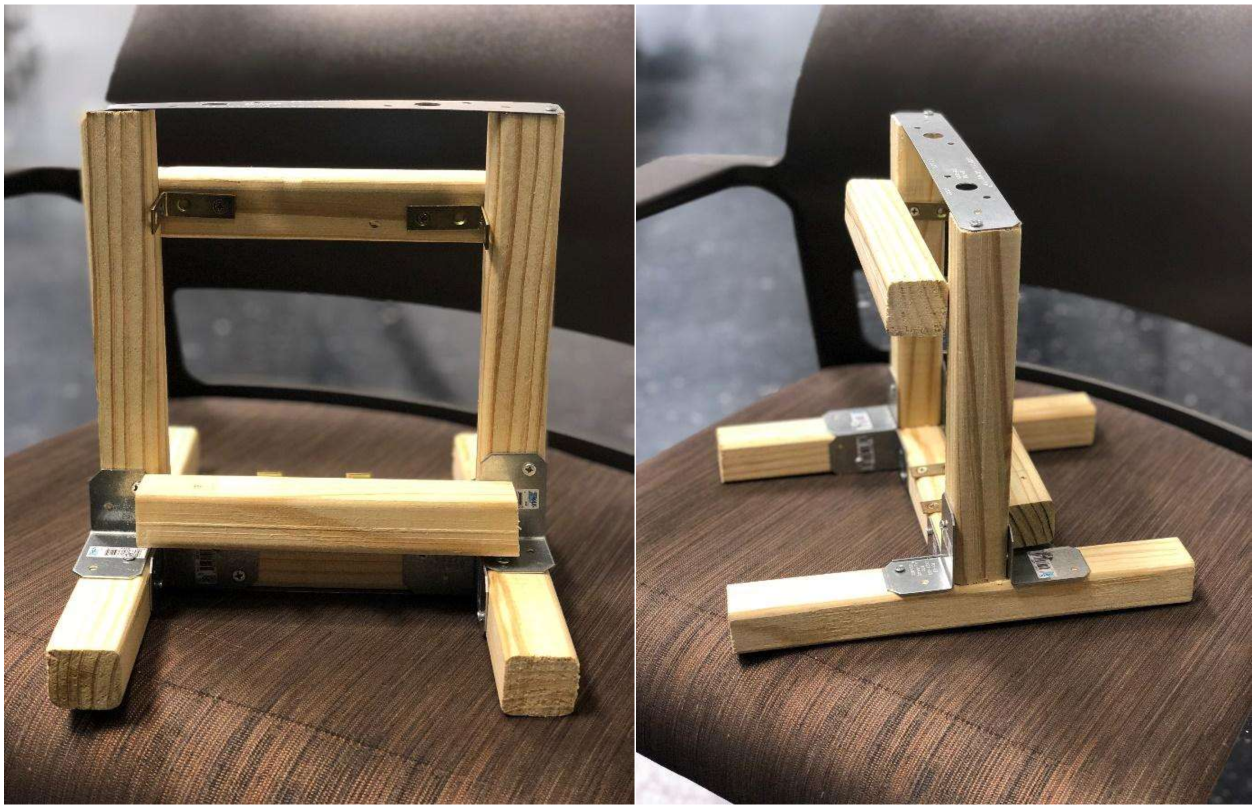
Figure 3. Animatronic Face Design Three

Section 2: Brainstorming

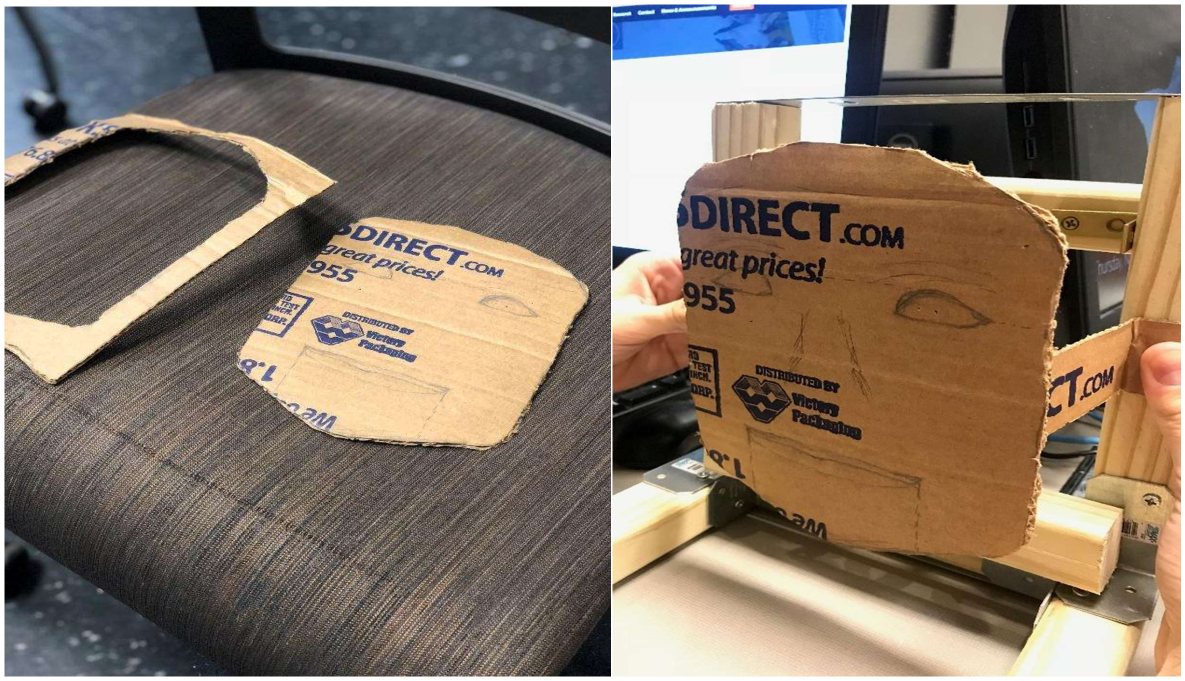
The design of the animatronic face in question will involve movement of the mouth and eyes. The mouth of the face will consist of various springs attached to both ends of it for retracting and its movement will be actuated with the use of a servo motor. The eyes of the animatronic face will not be directly connected to the main structure but will be very lightly gripped (to allow eye movement) inside the eye sockets. Similar to the mouth, the movement of eyes will also include the use of servo motors. These motors will be triggered with the use of two sensors. The first sensor that will activate the desired movements and audio will be a motion sensor. The second sensor will be a touch sensor, which will trigger a different set of movements and audio. The structure will be composed of thin malleable metal that can bent into a desired shape. The frame will be of rectangular shape with 2 or 3 servos attached. A frame will made to support the eyeballs and the servos controlling the movement of them will be placed behind the face. Another frame will be used to support the mouth and the servo controlling it. The facial structure will be made of thin cardboard that will be cut to approximate scaling of the professor and will be covered with a picture of the professor. Dr. Gao will be recorded saying, “The shaft should be made with a design safety factor of 2” and “Do not cheat on this project because I will find out” into a tape recorder or phone. The audio will be uploaded onto an Arduino unit and played via an audio player for the Arduino. The motion sensor will either be placed on the side of the head or in the back and the touch sensor will be placed on at the front of the structure. The Arduino will be powered with a portable battery that will also power all the other sensors and servos.

Section 3: Supporting structure

The supporting structure is composed of 6-12-inch pieces of wood all connected via various 90° metal connectors to produce a platform and frame supporting the face. The base of the structure is an H pattern (for sturdiness) with two more wood planks mounted right above for servo placement for the mouth and one for the eyeballs. The supporting structure is shown in **Figure 4** and **Figure 5**.As for the face structure, cardboard is used for its simplicity, and is cut into the shape of the provided professors head shape (Dr. Gao) shown in **Figure 6**. Holes for the eyes and the lower part of the mouth are drawn (for later cutting) to create eye sockets and to allow mouth movement. The face structure is mounted onto the supporting structure as shown in **Figure 7**.



Figures 4-5. Design of Supporting Structure



Figures 6-7. Design of Face Structure

Section 4: Joints and Motors

The entire face structure consists of only four joints; two for mouth (one on each corner), and two for the eyes (one joint per eye). All of the joints are actuated with the use of servo motors. Servo motors were chosen due to their ease of use and programming. The joints that make up the mouth of the facial structure are shown in **Figure 8**. Both corners of the mouth are attached to two separate servo motors that are mounted on the bottom front plank and each move down 30° and back up to their original position when activated. Due to the concern of the servos producing a torque within the mouth portion, they were glued for rigidity which held throughout the design process. The joints for the eyes can also be seen in **Figure 8**. Here each servo is directly attached to a ping pong ball (eye balls) via a wooden rod, such that when the servo rotates, the eyes rotate as well. To minimize friction between the ping pong balls and the cardboard, a hole was punched through each "eye ball socket" at a specific distance away as well as to compensate for the servo mounted below. This ensured only uniaxial twisting of the eyes. Simplicity and minimal cost was factored in this design as each joint is one moving part. This was opposed to a rack and pinion design (steering wheel assembly) composed of four or five moving parts further adding complexity and possibility for failure upon testing.

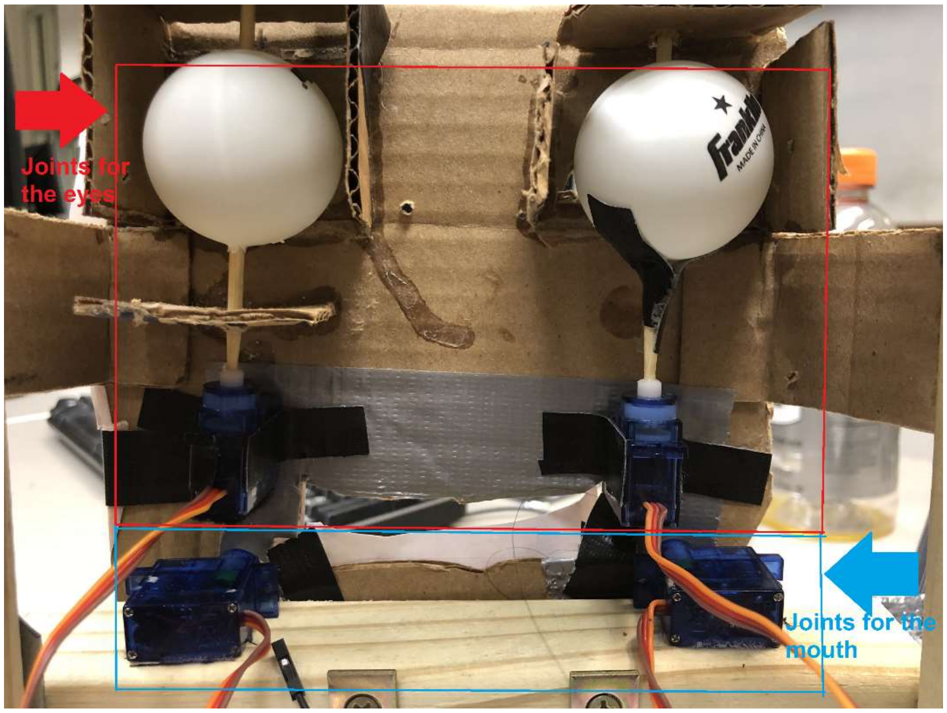
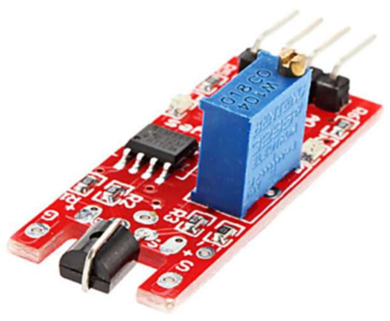


Figure 8. Joints for the Eyes and Mouth

Section 5: Sensors

The sensors used for this animatronic face design is a motion sensor and a touch sensor. These two sensors were chosen for their simplicity, their difference in their technology, and because they were readily available. Each of these sensors trigger a unique set of facial movements and audio when they detect a specified input (input is specified in the code). The motion sensor used in this project is shown in **Figure 9**. [4] This particular sensor is a Passive Infrared Motion Sensor (PIR), which detects infrared radiation. Thus, when there is a change in the amount of infrared radiation in front of the sensor, the PIR sensor detects it. The touch sensor used is shown in **Figure 10**. This touch sensor is a KY-036 Metal-touch module that is somewhat of a switch that triggers when the metal piece of the sensor is touched by a conductive body such as human fingers.



Figures 9-10. PIR Motion Sensor and KY-036 Metal-Touch Module

Section 6: Programming for Interaction

Programming for this animatronic face was completed using the Arduino IDE software in various steps to achieve proper interactions. Firstly, the proper libraries for the servo motors, SD card (for reading SD cards), Serial Peripheral Interface (SPI- allows communication with SPI devices with Arduino as the master device), and WAV audio playback had to be installed. Once these libraries were installed, each of servos were named (myservo, myservo2, etc.), and the initial directions for every servo was set at 60°. Coding for each of the sensors and motors was then required. Programming for the PIR sensor and the KY-036 Metal-Touch Module was completed by researching online for various codes that could be altered to fit the goals of this project. For motor programming, “if” statements were used in unison with the sensor coding to trigger the motors when the sensors read a specified value. For the PIR sensor, when motion was detected (change in infrared radiation), the input pin for the sensor would read “high”, which then triggered the servos to perform a provided set of movements. For the touch sensor, when the metal piece was touched/pressed, the input pin for the sensor would read a value ranging from 0-600 depending on how hard the metal piece was pressed (pressing harder reads lower values). Thus, using the “if” statement, the motors would be triggered to perform a certain set of movements if the value read less than 500. Programming of the audio into the code was done similarly to the programming of the sensors, in the sense that online research was conducted to find a base code to work from. However, due to complications within the WAV audio playback library installed, the audio did not play properly along with the servo movement. It was discovered late into the project that the library used was faulty and a whole different library and coding would be required in order to get the audio to work. Other than the integration of the audio code into the servo/sensor code failing, all other programming worked perfectly. The final code for this animatronic face can be found in **Appendix A**.

Section 7: Lessons Learned and Suggestions

1) **Lessons Learned-**
    I. Programming Issues-Learning how to compile multiple servos and sensors within the coding to obtain a desired output proved to be a difficult process. Previous experience with Arduino only comprised of simpler applications with less complexity. Thus, the challenges this project provided allowed an increase in programming experience and allowed a better understanding of how control systems work. One of the errors that occurred within the programming sector of this project was the verification of the code due to problem within one of the libraries needed for audio output. Due to the late discovery of this issue, there

was no fix to the problem.

II. Joint Design- This section involved a lot of creativity in the sense that it required multiple ideas to be considered for the final joint design. The joint design for the eyeballs took the most thought, as errors arose with the initial design. Multiple structure designs were considered for the eye joints such as a rack and pinion system, double side jointed eyeball controlled by a single servo (as such was the original design plan by the top and rear mounted plank featured in the structure) and other non-conventional methods. However, the main problem to overcome with the initial design was actuating linear motion through angular motion (servo motion) in attempt to get the eyes to rotate. It was then decided that directly connecting the eyeball and servo together via a skewer stick would be a better option.

III. Power issues- Due to the large amount of electronic components attached: 4 servos, 2 sensors, 1 amplifier, 1 SD card adapter, and a speaker. The power consumption would immediately shut off the Arduino upon uploading the code to it via USB on the computer. The main cause was the large amperage draw by all the components combined, thus surpassing the spec amperage output for the power source in use (computer usb port). Multiple attempts were done to fix this issue such as daisy chaining double A batteries and USB wall outlets composed of regular 5V, 0.5-1A output. The fix to this issue was a wall outlet plug that provided a 12V and 2A output. This new power source provided a sufficient amount of power to all the electronics including the Arduino MEGA itself.

2) Suggestions-

I. Building another replica cardboard face for the final presentation should have been completed as to show a neat and organized system of components with the same exact design without the cosmetic flaws. Performance and reliability of the system would have been prominent.

II. Using springs or rubber bands for the mouth and servo attached should have been done as opposed to using duct tape and wire to compensate for the outward stretching done by the servos when actuated. This would have made the design look more presentable and provide higher reliability.

III. Applying the use of an Arduino MP3 Shield as possible means to negate the coding issues and utilizing an entire different library.

IV. Using resistors to control the power draw of the electronic components thus also possibly negating the power issues.

Section 8: Personnel and bill of materials

(a) Personnel

Task	Main Personnel	Secondary personnel
Structure/Chassis Design	Carlos Quesada	Nicolas Lara
Motor/Joint Interfacing	Carlos/Nicolas/David	
Sensor Integration and Interfacing	Nicolas/Carlos	David Matamoros
Overall Programming/Integration	Nicolas Lara	Carlos Quesada
Cosmetics and Structure Integrity	David Matamoros	Carlos Quesada

(b) Bill of materials

No.	Description	Website/comment	Qty.	Unit $	Total $
1	Arduino MEGA 2560	Provided	1		
2	Wood (2x2 36”)	Home Depot	2	$0.87	$1.74
3	Metal Connectors	Home Depot	12	$0.70	$8.40
4	Screws (45 pack)	Hope Depot	1	$2.17	$2.17
5	Super Glue	Home Depot	2	$3.97	$7.94
6	Duct Tape	Home Depot	1	$7.97	$7.97
7	Skewer Sticks (100 pack)	N/A (Convenience store)	1	$5.57	$5.57
8	Ping Pong Balls (six pack)	N/A (Convenience store)	1	$1.93	$1.93
9	Servo 5 Pack	Amazon	1	$7.98	$7.98
10	Arduino LM386 5 pack - (for audio playback)	Amazon	1	$11.78	$11.78
11	Wig	Dollar Tree	1	$1.00	$1.00
12	Glasses	Dollar Tree	1	$1.00	$1.00
13	Cardboard	Provided	1		
14	Corrugated Plastic	Provided	1		

The total price for this project excluding the Arduino MEGA was $57.48

References

[1] Assembly Video/Software Link: https://kerkits.com/pages/fritz-support-page-assembly-softwareinstallation-etc

Assembly Parts Link: https://cdn.shopify.com/s/files/1/1320/7675/files/Fritz_assembly_Instructions_VERS3.pdf?1855 087446 633294682

[2] Assembly: https://www.stanwinstonschool.com/tutorials/character-animatronics-how-to-build-ananimatronic-head-part-1-mechanical-underskull

Parts Link: https://www.servocity.com/hs-645mg-servo https://www.servocity.com/hs-82mg-servo https://www.servocity.com/hitec-hs-53-servo https://www.servocity.com/hs-645mg-servo

[3] Assembly Mouth: http://www.pyroelectro.com/tutorials/animatronic_mouths/index.html

Assembly Eyes: http://www.pyroelectro.com/tutorials/animatronic_eyes/index.html

Assembly Eye Brows: http://www.pyroelectro.com/tutorials/robotic_eyebrows/

[4] Instructables. (2018, February 15). How PIR Sensor Work. Retrieved from https://www.instructables.com/id/How-PIR-Sensor-Work/

Appendix A: Code

```
#include <pcmRF.h>
#include <TMRpcm.h>  // WAV audio playback
#include <SD.h>
#include <SPI.h>


 /* This code sweeps a servo from 0 degrees to 180 when the PIR sensor detects motion.
   Special thanks goes to the author of the PIR sensor code, whose code helped tremendously
   in the making of this code and Instructable.
   author of PIR sensor code: Kristian Gohlke / krigoo (_) gmail (_) com / http://krx.at */
   // Audio Out - pin 46
 // SD card attached to SPI bus as follows:
 //  MOSI - pin 51
 // MISO - pin 50
 // CLK - pin 52
 // CS - pin 53

#include <Servo.h>


#define SD_ChipSelectPin 53 //Chip select is pin number 6
TMRpcm audio; //Lib object is named "music"

Servo myservo;  //creates a servo object
Servo myservo2;                       //a maximum of eight servo objects can be created
Servo myservo3;
Servo myservo4;
int pos = 60;        //variable to store servo position
int pos2 = 60;
//amount of time we give the sensor to calibrate(10-60 secs according to the datasheet)
 int calibrationTime = 15;

//the time when the sensor outputs a low impulse
long unsigned int lowIn;

//the amount of milliseconds the sensor has to be low
//before we assume all motion has stopped
long unsigned int pause = 5000;

boolean lockLow = true;
boolean takeLowTime;

int pirPin = 12;            //digital pin connected to the PIR's output
int pirPos = 13;            //connects to the PIR's 5V pin
const int slaveSelectPin = 53 ;

void setup(){
    // set the slaveSelectPin as an output:
  pinMode (slaveSelectPin, OUTPUT);
  digitalWrite (slaveSelectPin, HIGH);
  // initialize SPI:
  SPI.begin();
  audio.speakerPin = 46; //Auido out on pin 9
Serial.begin(9600); //Serial Com for debugging
if (!SD.begin(SD_ChipSelectPin)) {
Serial.println("SD fail");
return;
}
audio.setVolume(5);    //   0 to 7. Set volume level
audio.quality(0);        //  Set 1 for 2x oversampling Set 0 for normal
                         //music.play("filename",30); plays a file starting at 30 seconds into the track
  myservo.attach(4);    // eye 1 attaches servo to pin 4
  myservo2.attach(5);   // eye 2
  myservo3.attach(3);   // mouth
  myservo4.attach(2);   // mouth
  myservo3.write(60);
  myservo4.write(60);
  myservo.write(60);
  myservo2.write(60);
  Serial.begin(9600);  //begins serial communication
  pinMode(pirPin, INPUT);
  pinMode(pirPos, OUTPUT);
  digitalWrite(pirPos, HIGH);

  //give the sensor time to calibrate
  Serial.println("calibrating sensor ");
  for(int i = 0; i < calibrationTime; i++){
    Serial.print(calibrationTime - i);
    Serial.print("-");
    delay(1000);
  }
  Serial.println();
  Serial.println("done");

  //while making this Instructable, I had some issues with the PIR's output
  //going HIGH immediately after calibrating
  //this waits until the PIR's output is low before ending setup
  while (digitalRead(pirPin) == HIGH) {
    delay(500);
    Serial.print(".");
  }
  Serial.print("SENSOR ACTIVE");

}

void loop()
{
 int analogValue = analogRead(A0); // read the analog input
 delay(1000);
  Serial.println(analogValue);      // print it
  if(analogValue<500)        // Touch Sensor, "Make sure you contribute to the project"
{
  myservo.write(30);
myservo2.write(90);
 delay(1000);
   myservo3.write(90); // "make"
   myservo4.write(30);
   delay(200);
    myservo.write(60);
   myservo2.write(60);
   myservo3.write(60);
   myservo4.write(60);
   delay (250);
   myservo.write(60);   // "sure"
   myservo2.write(60);
   myservo3.write(90);
   myservo4.write(30);
   delay(200);
     myservo.write(60);
   myservo2.write(60);
   myservo3.write(60);
   myservo4.write(60);
     delay (150);
   myservo.write(30); // "you"
   myservo2.write(30);
   myservo3.write(90);
   myservo4.write(30);
   delay(200);
     myservo.write(30);
   myservo2.write(30);
   myservo3.write(60);
   myservo4.write(60);
   delay (200);
   myservo.write(60);    // "con"
   myservo2.write(60);
   myservo3.write(90);
   myservo4.write(30);
   delay(200);
   myservo.write(60);
   myservo2.write(60);
   myservo3.write(60);
   myservo4.write(60);
   delay(200);
   myservo.write(60);    // "tri"
   myservo2.write(60);
   myservo3.write(90);
   myservo4.write(30);
   delay(300);
   myservo.write(60);
   myservo2.write(60);
   myservo3.write(60);
```

```
myservo4.write(60);
 delay(150);
myservo.write(60);    // "bute"
myservo2.write(60);
myservo3.write(90);
myservo4.write(30);
 delay(300);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(150);
myservo.write(90);    // "to"
myservo2.write(90);
myservo3.write(90);
myservo4.write(30);
 delay(300);
myservo.write(90);
myservo2.write(90);
myservo3.write(60);
myservo4.write(60);
 delay(200);
myservo.write(60);  // "the"
myservo2.write(60);
myservo3.write(90);
myservo4.write(30);
 delay(250);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(150);
myservo.write(60);    // "pro"
myservo2.write(60);
myservo3.write(90);
myservo4.write(30);
 delay(200);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(200);
myservo.write(60);    // "ject"
myservo2.write(60);
myservo3.write(90);
myservo4.write(30);
 delay(200);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
```

```
}

  if(digitalRead(pirPin) == HIGH){  //if the PIR output is HIGH, turn servo

    /*turns servo from 0 to 180 degrees and back
    it does this by increasing the variable "pos" by 1 every 5 milliseconds until it hits 180
    and setting the servo's position in degrees to "pos" every 5 milliseconds
    it then does it in reverse to have it go back
    to learn more about this, google "for loops"
    to change the amount of degrees the servo turns, change the number 180 to the number of
degrees you want it to turn
    **/
    //  Motion Sensor, "The test is long, but i do not expect everybody to finish"
    {
   myservo.write(90);
   myservo2.write(90);
   myservo3.write(60);
   myservo4.write(60);
   delay(1000);
   myservo.write(60); // "the"
   myservo2.write(60);
   myservo3.write(90);
   myservo4.write(30);
   delay(250);
    myservo.write(60);
   myservo2.write(60);
   myservo3.write(60);
   myservo4.write(60);
   delay (75);
   myservo.write(60); // "test"
   myservo2.write(60);
   myservo3.write(90);
   myservo4.write(30);
   delay(200);
     myservo.write(60);
   myservo2.write(60);
   myservo3.write(60);
   myservo4.write(60);
     delay (150);
   myservo.write(90); // "is"
   myservo2.write(30);
   myservo3.write(90);
   myservo4.write(30);
   delay(250);
     myservo.write(90);
   myservo2.write(30);
   myservo3.write(60);
   myservo4.write(60);
    delay(200);
   myservo.write(60);    // "long"
```

```
myservo2.write(60);
myservo3.write(90);
myservo4.write(30);
 delay(300);
myservo.write(90);
myservo2.write(90);
myservo3.write(60);
myservo4.write(60);
 delay(250);
myservo.write(90);    // "but"
myservo2.write(90);
myservo3.write(90);
myservo4.write(30);
 delay(300);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(200);
myservo.write(60);    // "I"
myservo2.write(60);
myservo3.write(90);
myservo4.write(30);
 delay(300);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(200);
myservo.write(30);    // "do"
myservo2.write(30);
myservo3.write(90);
myservo4.write(30);
 delay(300);
myservo.write(30);
myservo2.write(30);
myservo3.write(60);
myservo4.write(60);
 delay(200);
myservo.write(60);    // "not"
myservo2.write(60);
myservo3.write(90);
myservo4.write(30);
 delay(300);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(200);
myservo.write(60);    // "ex"
myservo2.write(60);
```

```
myservo3.write(90);
myservo4.write(30);
 delay(250);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(200);
myservo.write(60);    // "pect"
myservo2.write(60);
myservo3.write(90);
myservo4.write(30);
 delay(300);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(200);
myservo.write(90);    //"Every"
myservo2.write(90);
myservo3.write(90);
myservo4.write(30);
 delay(300);
myservo.write(90);
myservo2.write(90);
myservo3.write(60);
myservo4.write(60);
 delay(100);
myservo.write(90);   //"body"
myservo2.write(90);
myservo3.write(90);
myservo4.write(30);
 delay(150);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(100);
myservo.write(60);    // "to"
myservo2.write(60);
myservo3.write(90);
myservo4.write(30);
 delay(150);
myservo.write(60);
myservo2.write(60);
myservo3.write(60);
myservo4.write(60);
 delay(100);
myservo.write(60);    // "finish
myservo2.write(60);
myservo3.write(90);
```

```
   myservo4.write(30);
    delay(100);
   myservo.write(60);
   myservo2.write(60);
   myservo3.write(60);
   myservo4.write(60);

    }
    if(lockLow){
      //makes sure we wait for a transition to LOW before further output is made
      lockLow = false;
      Serial.println("---");
      Serial.print("motion detected at ");
      Serial.print(millis()/1000);
      Serial.println(" sec");
      delay(50);
    }
    takeLowTime = true;
  }

  if(digitalRead(pirPin) == LOW){

    if(takeLowTime){
      lowIn = millis();          //save the time of the transition from HIGH to LOW
      takeLowTime = false;    //make sure this is only done at the start of a LOW phase
    }

    //if the sensor is low for more than the given pause,
    //we can assume the motion has stopped
    if(!lockLow && millis() - lowIn > pause){
      //makes sure this block of code is only executed again after
      //a new motion sequence has been detected
      lockLow = true;
      Serial.print("motion ended at "); //output
      Serial.print((millis() - pause)/1000);
      Serial.println(" sec");
      delay(50);
    }
  }
}
```