UNIVERSITY OF TEXAS AT SAN ANTONIO

# Robot Face Independent Study Project

## RomoBOT and Elle-phant

**Geoffrey Toombs and James Schopfer**

**5/3/2016**

**Instructor: Pranav Bhounsule**

**In partial fulfillment of Independent Study at UTSA**

**Contact info: Geoffreytoombs@yahoo.com, James_Schopfer@yahoo.com**

# Contents

# The “Robotic Face” Build: RomoBOT

## Purpose:

This project is formed around a robotic face that entertains and has the ability to interact with the user, from young children to adult age robotic enthusiasts.

## Objectives:

To develop a package of control board, sensors, and functional features (i.e. speech via MP3 shield) that would control a robotic face and have the capability to interact with the user.

## Hardware:

| MATERIALS LIST | |
|---|---|
| Erector set | PVC board |
| 5x servo motors (90g, micro) | Ping pong ball |
| 1x larger servo motor Futaba 3003 | Wooden shaft |
| Arduino Mega | Plastic marble |
| MP3 Arduino shield | Push-rods (hobby) |
| Plastic mask | Ball joint for rods (hobby) |
| Computer Speakers | Micro SD Card |
| Wires | Soda bottle |
| MDF Board | Acrylic |

| Tools Needed | |
|---|---|
| Allen Wrench | Hot Glue Gun |
| Drimmel tool | Wire Strippers |
| Soldering Iron | Butane torch |
| Solder | Epoxy |
| Drill | Wood glue |

## Software:

Arduino or equivalent. (All code can be viewed in Appendix A. Wiring diagram can be seen in Appendix B. NOTE: diagram does not follow precise pinout of RomoBOT code. Pin assignment is the only difference.)

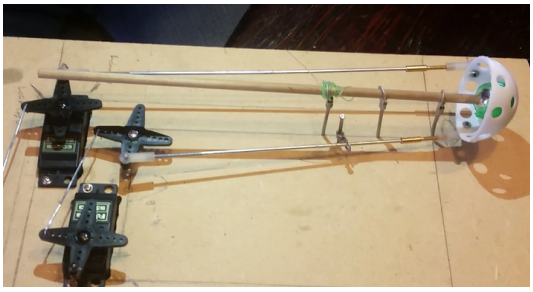
## The RomoBot:

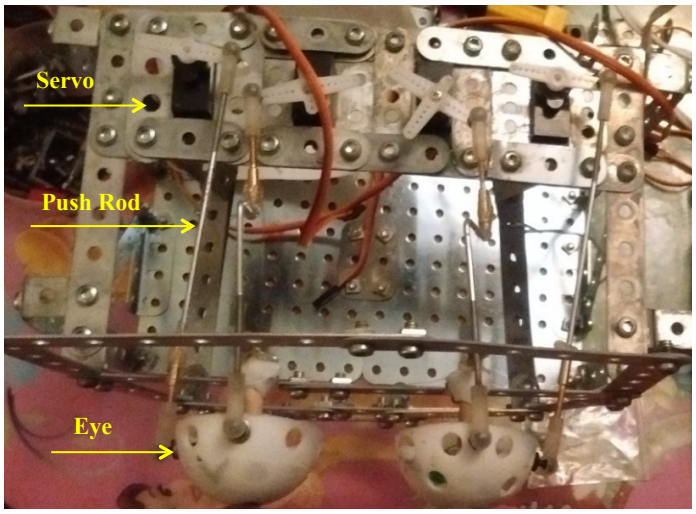
The development was a long process of trial and error, but I will stick to the successes and try not to dwell on the impediments. I will tell you other possible thoughts we had while making it

as well as some areas for improvement. Here is the final product, prior to adding sensors and a wooden box.

https://www.youtube.com/watch?time_continue=14&v=xkze1_hnam0

The frame of the robot was built using an Erector set. The Erector set gave me an easy to assemble frame for the robot. In constructing the eye assembly the ball socket method was chosen over the 2-axis gimbal, but both were considered and prototyped. The ball socket was created with a soda bottle, a plastic marble and a wooden rod. The plastic marble received a hole drilled to the size of the wooden rod. The rod was then glued into place using wood glue. The fabricated plastic receivers were glued to the wooden shafts. The plastic whiffle golf ball was cut in half using a Dremel multi-tool. The fabricated ball-receivers were attached to the center of the whiffle ball using epoxy putty.






*Figures 1 & 2: (Left) Ballsocket prototype; (Right) Final eye assembly*

Next, a frame was constructed using the erector set to support the eyes and required motors. 4 motors were used for the eye movement (each eye has 2 motors; 1 for x-axis movement, 1 for y-axis movement). The positions of the eyes are critical; they need to be positioned behind the face and symmetric. Each of the eye motors was adjusted using the "motor adjustment" Arduino file. The frame supporting the eye assembly was added to a base, while allowing space for the rest of the face.

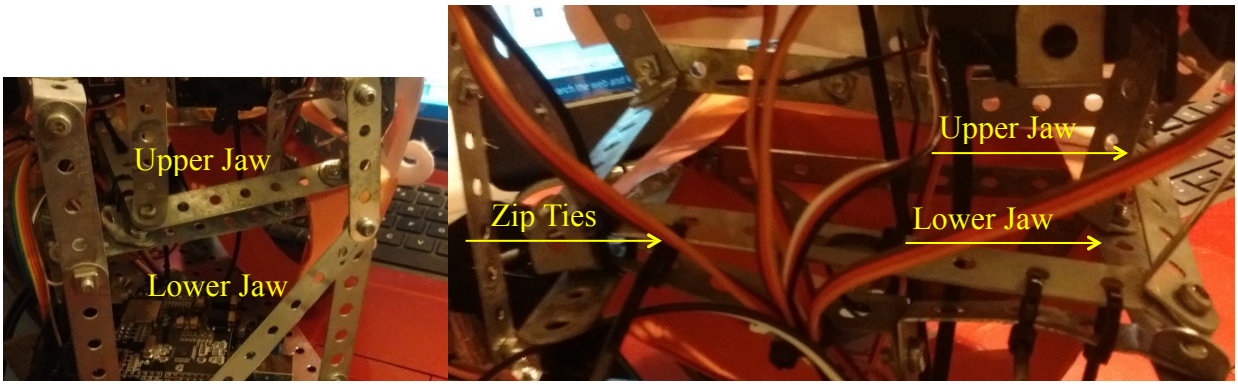
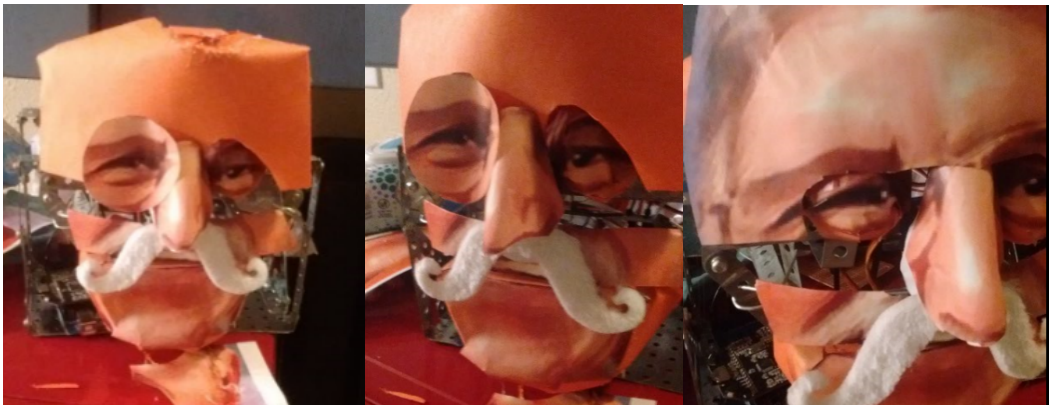



*Figure 3 & 4: Pictures of Jaw*

A lower mandible was constructed (bottom jaw) with the erector set. This piece was secured to the upper jaw using zip-ties to create a hinge. This could have been done many other ways, but this was functional. A large servo motor was attached inside the mouth securely to the upper jaw and the servo horn was wired to the lower jaw. The plastic mask was used to create a structure to glue the printed face to. Orange file folders were used where the mask didn't fit. The eye balls were covered with the printed eyes that were cut from the printed images. A total of 2 color pictures (8.5" by 11") were printed of an illustration by Michael Hogue of Doctor Ricardo Romo (the current President of UTSA) to cut up and glued to the facial frame.



*Figures 5, 6 & 7: Appling the Eye Cover*

The nose and mustache required another motor to give this assembly the movement required to wiggle. The motor, adjusted to the same orientation as the others, was zip tied and hot glued to secure it. Special care needs to be taken to allow movement of eyes, nose, and mouth without any impediments.

As for speech, an attempt of using syllables to distinguish when the mouth opens and closes was used to no avail. It was decided to use the maximum (yellow dot) and minimum (red dot) seen in the picture below to simplify the movements. When there was a high spot the mouth should be open, and when it was closest to the center minimum it should be closed. This took some trial and error, but it got close.

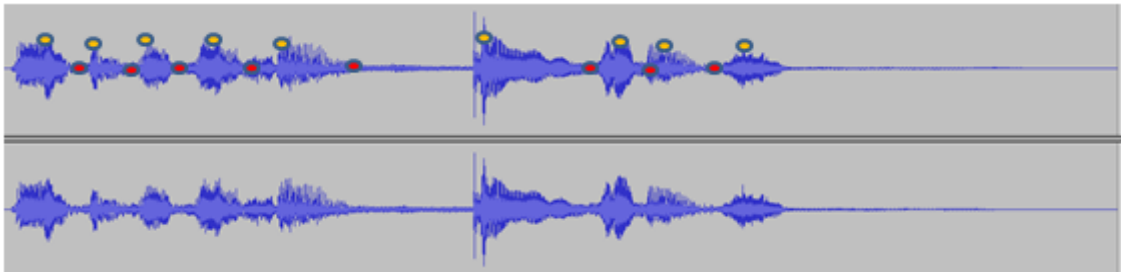


*Figure 8: Speech Pattern*

The audio was a sample taken from a YouTube video using various softwares to turn the video file into an audio file and then cutting to the proper length. The original address of the video was copied and pasted on the Zamzar,com website so the video could be downloaded and manipulated (follow instructions on website). The program, Audacity, was used to turn the video file into an audio file and it allowed the audio to be clipped and saved as a MP3 file to a micro SD card. The MP3 shield was then **carefully** attached **(**try to keep from bending the pins) to Arduino Mega.

Finally, a box was made out of MDF with a clear back made from acrylic. The dimensions are 12” by 9.5”. The wood was cut with a jigsaw and assembled with wood glue and clamps. The acrylic was cut with an acrylic cutting tool to keep plastic from cracking. The hinges were screwed to the wood and some epoxy putty was used on the acrylic side due to interference created by the screws.

*Figures 9 and 10: (Left) Outer frame with back; (Right) RomoBot*

## The Robotic Arm and Elephant Face, Elle-phant:

### Purpose:

Add the capability to physically move objects and demonstrate the versatility of the robotic face platform with other faces.

### Objectives:

The addition of a 2 DOF or 3DOF arm to use with the robot face. Develop a second robotic face for the control board, sensors, and features of the Romobot to integrate with.

### Hardware:

Servo motors, Futaba 3003 and miniservo, control board (Arduino Uno to mega), structure material (cheap & lightweight wood, plastic, metal) and associated hardware.

### Software:

C++, compatible with Arduino, Raspberry Pi

To begin the prototype process ordered a MeCon Robot Arm, with windows interface to control arm. The Windows GUI would not function, the servos are underpowered and at the 0 degree

position the arm can get stuck. Disassembled the MeCon arm to provide materials for the (Figure 8,9) 1st iteration of prototype.

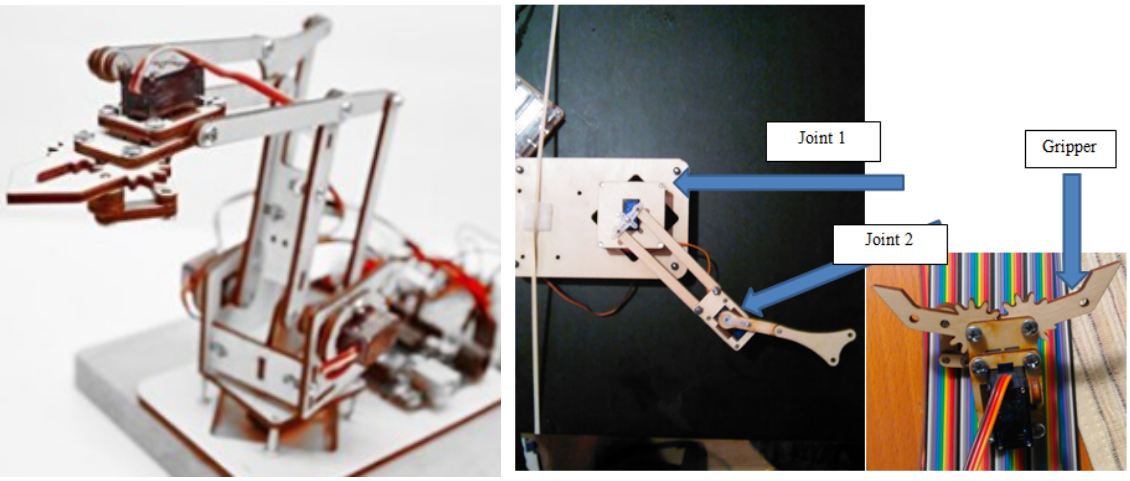
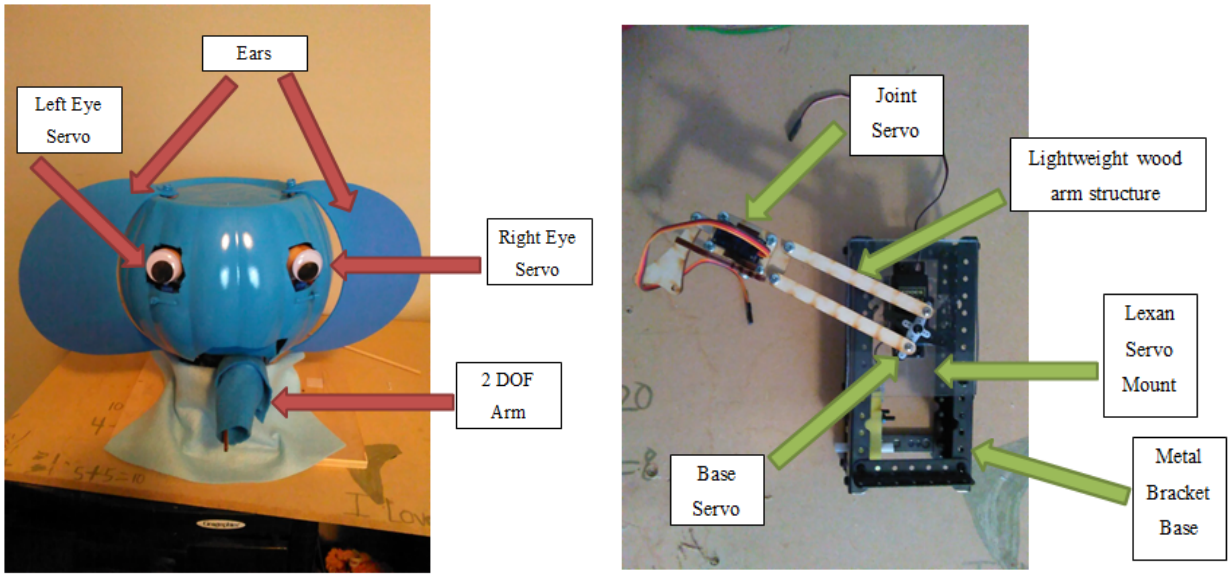



*Figure 11, 12, and 13: (Left) MeCon DIY Robot Arm; (Middle) 1st prototype; (Right) Gripper*

The next iteration had upgraded motors to Futaba 3003, downloaded Matlab with board for inverse kinematics, better structure material; interface with robotic face. The results were: improved torque with the Futaba 3003 motor. Matlab would not load properly onto the Arduino board, so simple kinematics used with trigonometry. The structure material Lexan proved to be similar to light weight wood used in the arm as far as performance and utility, transparency useful when assembling.

The construction of the elephant robotic face was made with a recycled Halloween plastic pumpkin, the ears from a foam visor, and the trunk "skin" from blue felt. The eyes were constructed from handle knobs and craft doll eyes, all hot glued and mounted to servo horns. The eyes were actuated by two servo motors and allowed lateral movement from left to right. The robotic arm had 2 degrees of freedom and actuated by servo motors. The rotational motor at the base, where the trunk meets the face, was a servo motor with greater torque than the one used in the initial robotic arm. The gripper was removed because the gear mechanism would not work properly nor did the mount. The structure used was a combination of erector set style metal brackets, lightweight wood, and Lexan. The metal brackets provided a base that allowed many different attachment points. The wood was light enough to stay within the servo motor's torque

capabilities. The Lexan was simple to modify as a mount for the servo at the base of the arm. See the figures below.




*Figures 14 and 15: (Left) Elephant robotic face; (Right) 2 DOF robot arm*

**Robotic arm and Elephant Conclusion**

The robotic arm was integrated with the elephant face to enable the robotic face platform to move or manipulate objects without constructing a body or additional structure. The elephant face was simple to construct with readily available materials. This platform allows for simple improvements to integrate with both the Arduino board and physical hardware. These additions to the robotic face, the Romobot, can widen both the appeal and functional capabilities.

## Changes and Improvements (in no particular order)

- Use a small speaker instead of the computer speakers for RomoBOT. This would allow the speaker to be hidden inside the head.
- Integrate a web camera into the eye
- Use a raspberry pi to allow for more complex movements

- Use stepper motors to allow for more precise and fluid movement. This is really more of an assumption since I have never used steppers, but from my understanding steppers would allow greater precision.
- I would put infrared sensors and more motors to allow the face to track you.
- Work on using this robot for Skype phone calls.
- Add eyelids with random blinks

# Appendix A – Code

## 1. Motor Adjustment

```
// use this program for inital calibration to 90 degrees, and
unplug arduino.
#include <Servo.h>

Servo Test1;

void setup()
{
  Test1.attach(2);  // attaches the servo, to digital pin 2
}

void loop()
{
  Test1.write(50);
  delay(1000);
  Test1.write(90);
  delay(3000);
  Test1.write(110);
  delay(1000);
}
```

## 2. Romo “Complete Movement” Motor Test

```
/*geoff toombs attempt to turn 2 motors at once.
2/5/16*/
/*move at same rate - in tandum*/


#include <Servo.h>

Servo R_eyebrow; //Right eye brow
Servo L_eyebrow; // Left eye brow
Servo X_eyes; // moves eyes on x axis
Servo Y_eyes; // moves eyes on y axis
Servo mouth; // mouth

void setup() {

R_eyebrow.attach(6);
L_eyebrow.attach(5);
X_eyes.attach(3);
Y_eyes.attach(4);
mouth.attach(7);

}

void loop()
{
 /*
 ----------
EYEBROWS---------------------------------
 */
 // puts eyebrows to 90 degrees... origin. This allows
for full range of emotions.
 R_eyebrow.write(90);
 L_eyebrow.write(90);
 delay(1000);

// Angry
//both eye brows slant down towards nose

 R_eyebrow.write(120);
 L_eyebrow.write(60);
 delay(1000);

// Surprised
// both eye brows slant upward towards forhead

 R_eyebrow.write(60);
 L_eyebrow.write(120);
 delay(1000);

//  Origin

 R_eyebrow.write(90);
 L_eyebrow.write(90);
 delay(1000);


/*
----------EYES---------------------------------
*/

/*
----------X axis RIGHT/LEFT------------------------
*/


        // puts eyes at origin (0 degrees)
 X_eyes.write(90);
 Y_eyes.write(90);
 delay(1000);

 X_eyes.write(120);
 Y_eyes.write(90);
 delay(1000);

 X_eyes.write(90);
 Y_eyes.write(90);
 delay(1000);

 X_eyes.write(60);
 Y_eyes.write(90);
 delay(1000);

 X_eyes.write(90);
 Y_eyes.write(90);
 delay(1000);
/*
----------Y axis UP/DOWN------------------------
*/
 X_eyes.write(90);
 Y_eyes.write(90);
 delay(1000);

 X_eyes.write(90);
 Y_eyes.write(120);
 delay(1000);

 X_eyes.write(90);
 Y_eyes.write(90);
 delay(1000);

 X_eyes.write(90);
 Y_eyes.write(60);
 delay(1000);

 X_eyes.write(90);
 Y_eyes.write(90);
 delay(1000);




/*
----------MOUTH--------------------------------
*/
        // puts mouth in origin (0 degrees)
 mouth.write(110);
```

```
  delay(500);
  mouth.write(60);
  delay(500);
  mouth.write(110);
  delay(500);
  mouth.write(60);
  delay(1000);
  mouth.write(110);
  delay(500);
}
```

## Romo Code – Complete

```
// using TaskScheduler to run mp3 and motors simultaneously
// createList was used with Seeed mp3 shield for audio.
// arduino mega is used
// 2 motors for each eye, 1 for x-axis, 1 for y-axis
// one motor for the mouth
// one motor for the mustache

#include <SD.h>
#include <SPI.h>
#include <arduino.h>
#include <MusicPlayer.h>
#include <Servo.h>
#include <TaskScheduler.h>

// Callback methods prototypes
void t1Callback();
void t2Callback();
void t3Callback();
void t4Callback();


//Tasks
Task t1(500, 1, &t1Callback);
Task t2(500, 1, &t2Callback);
Task t3(2000, 1, &t3Callback);
Task t4(2000, 1, &t4Callback);

Scheduler runner;


Servo mouth;
Servo mustache;
Servo R_X_eye; // moves eyes on x axis
Servo R_Y_eye; // moves eyes on y axis
Servo L_X_eye; // moves eyes on x axis
Servo L_Y_eye; // moves eyes on y axis

void setup() {
  Serial.begin(115200);
  Serial.println("Romo_mouth");

  runner.init();
  Serial.println("Initialized scheduler");

  runner.addTask(t1);
  Serial.println("added t1");

  runner.addTask(t2);
  Serial.println("added t2");

  runner.addTask(t3);
  Serial.println("added t3");

  runner.addTask(t4);
  Serial.println("added t4");

  delay(500);

  t1.enable();
  Serial.println("Enabled t1_mp3");
  t2.enable();
  Serial.println("Enabled t2_talking");
  t3.enable();
  Serial.println("Enabled t3_mustache");
  t4.enable();
  Serial.println("Enabled t4_roll_eyes");

  player.begin(); //will initialize the hardware and set default
mode to be normal.
  player.setPlayMode(PM_SHUFFLE_PLAY); //set mode to
play shuffle
  player.scanAndPlayAll(); //If the current playlist is empty,it
will add all the songs in the root directory to the playlist.


  mouth.attach(38);//green
  R_X_eye.attach(30);//brown
  R_Y_eye.attach(32);//red
  L_X_eye.attach(36);//orange
  L_Y_eye.attach(34);//yellow
  mustache.attach(40);//blue

}
void loop() {

  runner.execute();
}

// -------------------------------------
mouth-------------------------------
//90 degrees is closed
void open_mouth() {
  mouth.write(60);
  delay(100);
}

void close_mouth() {
  mouth.write(110);
  delay(100);
}


//-------------------------------------
volume-------------------------------------
void adjustVolume(void) //User-defined function
{
  unsigned int vol_temp = analogRead(A4);
  unsigned char volume = vol_temp / 12;
  if (volume == 0x55) volume = MAXVOL; //MAXVOL =
0xfe;
  player.setVolume(volume);
}

/*
    ----------Task Scheduler "mp3"-----------------

*/
void t1Callback() {
  delay(0);
  player.play(); //starts mp3
  Serial.print("mp3 - play: ");
  Serial.println(millis());
}

void t3Callback() {
  delay(0);
  wiggle_mustache();
  delay(500);
```

```
  Serial.print("wiggle wiggle wiggle: ");
  Serial.println(millis());
}

void t4Callback() {
  delay(500);
  roll_eyes();
  delay (500);
  Serial.print("rolleyes: ");
  Serial.println(millis());


}
// ---------- Task Scheduler "talking"----------
void t2Callback() {
  Serial.print("talking started: ");
  Serial.println(millis());

  //"welcome"
  open_mouth();
  delay(100);
  close_mouth();
  delay(50);
  Serial.print("1st breath: "); // 0.285 seconds
  Serial.println(millis());

  //"to"
  open_mouth();
  delay(50);
  close_mouth();
  delay(30);
  Serial.print("2nd breath: ");// 0.2 sec
  Serial.println(millis());

  //"UT..."
  open_mouth();
  delay(130);
  close_mouth();
  delay(10);
  Serial.print("3rd breath: ");// 0.185 seconds
  Serial.println(millis());

  //"..SA"
  open_mouth();
  delay(100);
  close_mouth();

  Serial.print("4th breath: ");// 0.190 seconds
  Serial.println(millis());

  // "go"
  open_mouth();
  delay(200);

  close_mouth();

  Serial.print("5th breath: ");// 0.118 seconds
  Serial.println(millis());

  //"runners"
  open_mouth();
  delay(250);
  close_mouth();

  Serial.print("6th breath: ");// 0.311 seconds
  Serial.println(millis());

  // total time 1.760 seconds + 0.035 seconds = 1.795 seconds


  // total time = ???
  // total time of audio = 3.5 + seconds
}
//--------------X axis RIGHT/LEFT-------------------------
// LOOK FORWARD
void look_forward() {
  // puts X axis at origin (90 degrees)
  R_X_eye.write(90);
  L_X_eye.write(90);
  R_Y_eye.write(90);
  L_Y_eye.write(90);


}

//Look RIGHT
void look_right() {
  R_X_eye.write(120);
  L_X_eye.write(120);
  R_Y_eye.write(90);
  L_Y_eye.write(90);

}

//LOOK LEFT
void look_left() {
  R_X_eye.write(60);
  L_X_eye.write(60);
  R_Y_eye.write(90);
  L_Y_eye.write(90);


}

//LOOK UP
void look_up() {
  R_X_eye.write(90);
  L_X_eye.write(90);
  R_Y_eye.write(120);
  L_Y_eye.write(60);

}
//LOOK UP_right
void look_upRT() {
  R_X_eye.write(120);
  L_X_eye.write(120);
  R_Y_eye.write(130);
  L_Y_eye.write(50);

}
//LOOK UP_left
void look_upLT() {
  R_X_eye.write(60);
  L_X_eye.write(60);
  R_Y_eye.write(130);
  L_Y_eye.write(50);
```

A4

A5

```
}
//LOOK DOWN
void look_down() {
  R_X_eye.write(90);
  L_X_eye.write(90);
  R_Y_eye.write(60);
  L_Y_eye.write(120);

}
//LOOK DOWN-RIGHT
void look_downRT() {
  R_X_eye.write(120);
  L_X_eye.write(120);
  R_Y_eye.write(50);
  L_Y_eye.write(130);
}
//LOOK DOWN-left
void look_downLT() {
  R_X_eye.write(60);
  L_X_eye.write(60);
  R_Y_eye.write(50);
  L_Y_eye.write(130);

}
//ROLL EYES
void roll_eyes() {
  look_right();
  delay(250);
  look_downRT();
  delay(250);
  look_down();
  delay(250);
  look_downLT();
  delay(250);
  look_left();
  delay(250);
  look_upLT();
  delay(250);
  look_up();
  delay(250);
  look_upRT();
  delay(250);
  look_forward();
  delay(250);
  Serial.print("roll_eyes");
  Serial.println(millis());


}
// WIGGLE
void wiggle_mustache() {
  mustache.write(90);
  delay(50);
  mustache.write(70);
  delay(50);
  mustache.write(110);
  delay(50);
  mustache.write(70);
  delay(50);
  mustache.write(110);
  delay(50);
  mustache.write(90);
  delay(50);
  mustache.write(70);
  delay(50);
  mustache.write(110);
  delay(50);
  mustache.write(70);
  delay(50);
  mustache.write(110);
  delay(50);
  mustache.write(90);
  Serial.print("wiggle_mustache: ");
  Serial.println(millis());
}
```

# Appendix B