DR. WAN'S ANIMATRONICS FACE:

Emiliano Rodriguez
Dept. of Mechanical Engineering
San Antonio, TX, USA 78249
pyn109@my.utsa.edu

Tony Amaro
Dept. of Mechanical Engineering
San Antonio, TX, USA 78249
aeb004@my.utsa.edu

George Gonzalez
Dept. of Mechanical Engineering
San Antonio, TX, USA 78249
george.gonzalez909@gmail.com

Submitted as part of project for ME4543 Mechatronics, Fall 2018

TABLE OF CONTENTS

ABSTRACT

The project is a user-interactive, voice enabled, cordless animatronic face of UTSA professor Hung-da Wan. The face consists of a controller that control the modes of the face and facial movements like the eyelids and eyes and mouth. At different modes, the animatronic face will do a different command based and the program. Some of the components of the face is micro servos, mp3 module, RGB LED, push buttons, joysticks, resistors, ultrasonic sensors, and wiring. The project resulted in 3 distinct modes that changed the way the face would react. There's a "free-mode" which the mouth, eyes and eyelids could each be controlled from the controller. The "eye testing" mode utilized an ultrasonic sensor to capture the distance in which an eye test sheet is away from the face. Depending on the distance, the face will play a sound bite of Dr. Wan attempting to read a line off the sheet while moving the mouth to mimic a talking motion. With the simplicity of a controller, the animatronic face is a fun, interactive and complex use of mechatronics.

Section 1: Literature review

The three animatronic faces found display different methods of construction in hardware and software used to create each project. This diversity is important as it will gather ideas and compare which methods will work for the task at hand. The most important factors conclude of motor selection, frame design and method of actuation.

Figure 1: Animatronic Neck (Obama face pasted over - left and Frame - Right)

The first animatronic face analyzed is a universal project in which any face can be pasted over the frame in order to replicate anybody's face [1]. This design is made up of medium density cardboard (MDF) [2] for the frame and utilizes servo motors for motion control. The interesting characteristic on this particular design is the wiring and logic control, as this project was manufactured before the arduino microcontroller existed, the creator used a PIC microcontroller which the code is uploaded via a device "programmer" [4]. The purpose of this animatronic was to show DIY makers that it was possible to make a versatile animatronic face with many capabilities, the point was also to display the movement limits for the head to show users what it's capable of handling.

Animatronics has been used in various films theme parks such as E.T, Terminator, and Disney Parks and many more. There purpose of these animatronics is to make operate like if they are real humans or creatures in real life. For the second animatronic face is made up with the frame of aluminium [5] shown in Figure 2

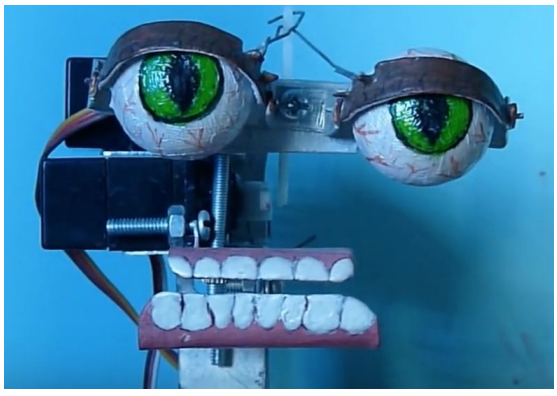

Figure 2: Animatronic face

Rather than using Arduino to make commands for this face they use 2 servo motors. The circuit was based on the 555 timer to provide pulse to the servo motors. A potentiometer was also used to vary the pulse width thus the angle of rotation. The servo motors were place in the to control the 55 timer. Hardest part of this project is construction the aluminum frame. On of the

servo motor is going to move the eyes and it eyelids an the other servo motor is going to move its mouth up and down motion. The eyes were made from rubber balls which were cut and painted.

In the spirit of Halloween, costume creation has transcended from putting on a blanket and calling yourself a ghost. Technology and creativity can result in unique costumes like the wearable animatronic headpiece below [6] . The face was designed using BowlerStudio, a robot development application, and contained 2 eyes and a mouth that was controlled with a Wii nunchuck, NodeMCU and servo motors. The project used BowlerStudio to CAD design and laser cut the pieces of the head and run a script for the NodeMCU to connect to the Wii nunchuck [7]. The servo motors were used to move the eyes and mouth by connecting a h-bridge and the NodeMCU. The purpose of the costume was for a contest of which the project won. It can be noted the devices were chosen because of their interfacing capabilities with BowlerStudio program working in Java 8.

Figure 3: Wearable Animatronic Head (left) and rest of costume (right)

Section 2: Brainstorming (initial planning)

The animatronic face should look as realistic as possible, this feature will be complicated as re-creating the facial structure of a person requires mold making and casting; another way this could be done is buying a mask and adding the professors facial features (glasses, hairstyle etc..). The moving parts would consist of both eyeballs and a mouth. Extra features of the face will include movable eyelids, and eyebrows as well as head movement. Those moving parts will require actuators and will move by servo motors. The sensors that will be used as input feedback will be buttons (touch sensors), remote controller or IR/ultrasonic sensors. The structure will be mostly made out of 3D printed material such as PLA or ABS, this will allow customization of the facial structure and other features. The face will be put on as a mask that fits over the structure. The soundtrack will be recorded by a smartphone and uploaded to a movie editing software to chop and edit the file so synchronization will be easier to adjust. The power supply for this system will either be portable (DC) or tethered (AC) in order to avoid power consumption by motors (which drain the most battery).

Section 3: Supporting structure

The supporting support of the animatronic will consist of ½ thick styrofoam poster board from Hobby Lobby. The styrofoam poster board was cut in 3 pieces, 2 pieces will be placed in the side of the face while one is place in the middle. The middle piece will have a nose like sticking out. A bracket for the eyes and eyelids, which the servo motors will be place, was

custom fabricated by using a 3D printer. The bracket will be placed between the end of the face with the middle of the face. The bracket will hold 4 servo motors and each once will have a specific job that will do.

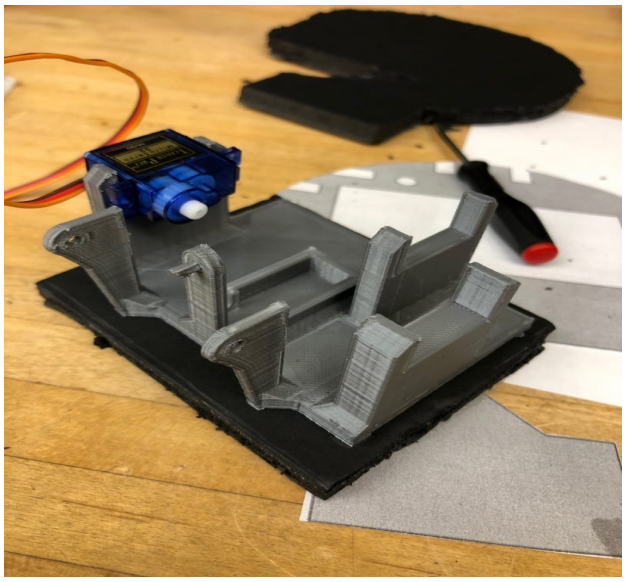

Figure 4: Bracket for the Servo motors

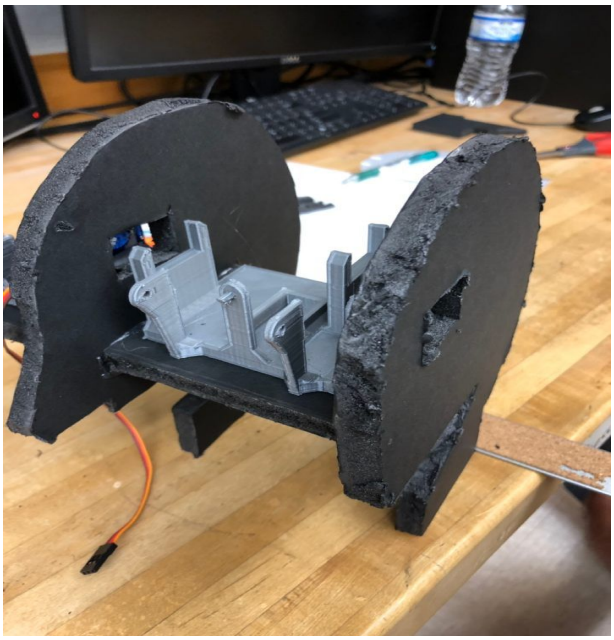

Figure 5: Side view of the support

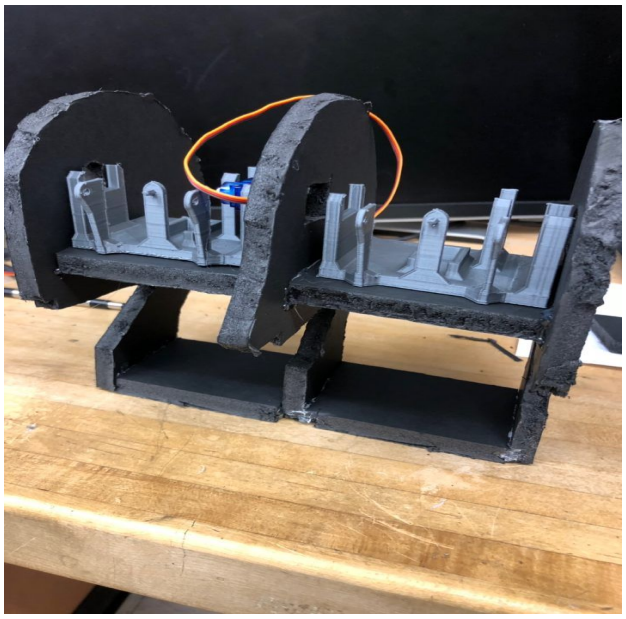

Figure 6: Full view of the support

Section 4: Joints and motors

In total the animatronic face has 9 motors to move specific parts of the face. The motors used were micro servo motors. These motors were located 4 for each side, and 1 is located at the bottom. These motors were used because it does not require a lot of power and movement for the motion of the blinking and eyes moving. To control the face in section 6 will be explain in more detail.

The servo located at the bottom is used to control the movement of the mouth. One side of the face the each motor had a job. The eye is connected to by 2 servo motors. Since the servo motors only move at the single direction, one will be place where the eye will only move the x-direction and the other servo motor will make the eye move the y-direction. The other 2 servo motors will move the eyelids, creating a realistic blinking person when being control. The eyes will be mounted on the bracket shown in figure 8 by using universal joints. The universal joints will then be placed on the bracket to connect to the eyes of the face and produce free x and y motion will being pivoted on the eye bracket. The mouth of the face used 3 components; the top set of teeth, the bottom set of teeth and the servo motor. The upper mouth piece was attached to the center vertical wall piece through super and hot glue so the piece is stationed. The bottom piece was mounted with two thin pieces of cloth to the back of the top mouthpiece. An additional universal joint was place on the back of the bottom mouthpiece to grant a pivot motion. A single servo motor was placed at the bottom of the lower mouth piece. The motion of the mouth is closed when the servo is completely pushing up on the bottom mouth. The mouth opens by rotating the servo until it is free from the mouth piece.

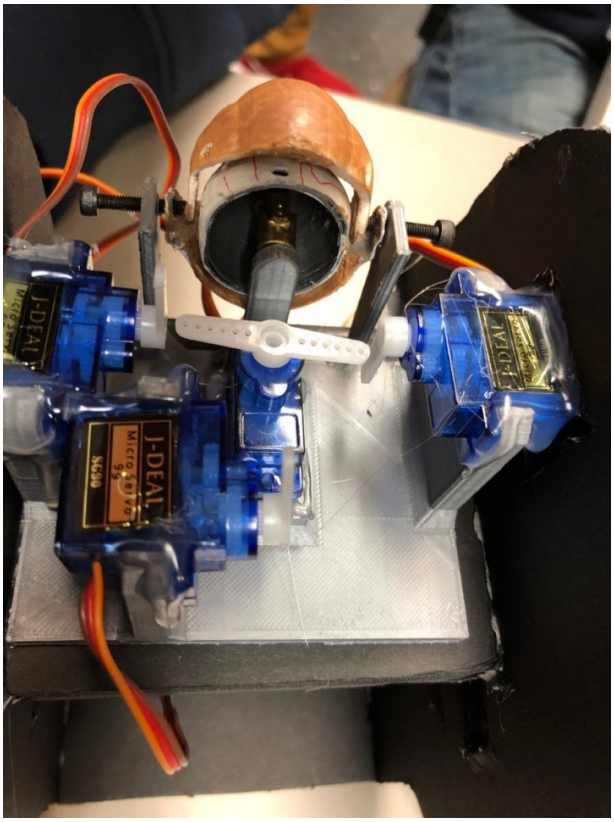

Figure 7: Back view of eye bracket showing the universal joint connection


Figure 8: Brackets of Eyes

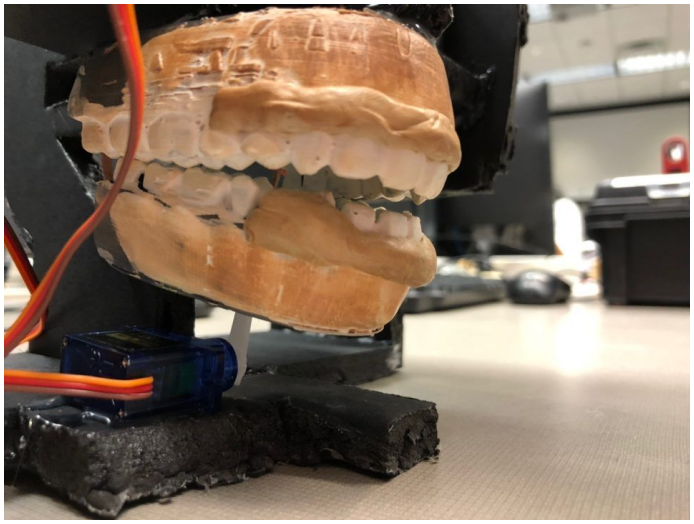

Figure 9 Mouth set up

Section 5: Sensors

The sensors used for this project will be the ultrasonic sensor and 2 push button, and RGB led. Theses sensors were selected because the push button when one being push, it controls the modes of the animatronic face and the second button is used for when being in mode 1, later explained, controls the eyelids which makes it blink. The RGB will indicate which mode is on. The ultrasonic sensor will be use for the second mode, later explained , for at a distance of 75 centimeter, the face will act on it own depending on what we make it do.

A simple joystick controller is designed face at mode 1. Mode 1 will be joystick mode and the color would be **red, green, blue**. During joystick mode, the face will face its freedom to be controlled from the eyes or eyelids.Also when the mode is set to 1, it will play a random track and actually start talking to you. When LED is Blue this will be mode 1. When LED is red its saying the the mode is in second. When the mode is red and blue the mode will be in 3.

For mode 2 an ultrasonic sensor is used for the distance reading. The ultrasonic sensor is used because the programing of the face require to do an eye exam. At a distance of 75 cm , the face will trigger and will close one eye and read the recording of the eye exam. After finishing the eye exam, at a distance of 100 cm, the face will close the opposite eye to perform an the next row of the exam.

For mode 3, is the same as mode 1 but without playing the voice track. The only difference of the mode 1 and mode 3 is there is an easter egg in the controller. If you hold the push button for over 8 seconds, it will start playing songs. A video is uploaded into Professor Bhounsule for recording purposes in case the day of the presentation the animatronic will fail.

Section 6: Programming for interaction

The programming involved consisted of several methods, classes and objects. Although some variables were global (in order to be used in all functions) the setup loop initiated and declared some variables used. The void loop begins by a conditional statement where if mode is greater than 3, then reset the arduino; this takes care of the problems in the mp3 module, such as fast looping of tracks (big problem). After, if the right button is pressed, mode increases

by 1. This allows the switch statement to call certain methods in the code. Mode 1 calls the joystick method along with a random mp3 sound clip. Method 2 calls the distance method, which syncs the sound and reads off a eye exam paper and the third one uses the joystick method with a secret easter egg mode (if you hold down the left button for more than 8 seconds it plays a compilation of well known songs. Finally, if the right button is pressed (mode button) then the mode is above 3 and the arduino resets.

The necessary libraries are the following for the code to work.

```
#include <Servo.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <DFMiniMp3.h>
#include <Adafruit_PWMServoDriver.h>
```

Figure: Libraries included

```
void loop() {
if(mode > 3)
  resetFunc();

Serial.println(mode);

if(digitalRead(2)==HIGH)
  mode++;

switch (mode) {
    case 1:
  mp3.stop();
  Serial.println("1");
  analogWrite(rPin, 0);
  analogWrite(gPin, 0);
  analogWrite(bPin, 100);
  joyStick();
      break;
    case 2:
        Serial.println("2");
  analogWrite(rPin, 100);
  analogWrite(gPin, 0);
  analogWrite(bPin, 0);
        distance();
      break;
    case 3:
  Serial.println("3");
  analogWrite(rPin, 100);
  analogWrite(gPin, 0);
  analogWrite(bPin, 100);
      joyStickOnly2();
      //do something when var equals 2
      break;
      default:
  Serial.println(mode);
  analogWrite(rPin, 0);
  analogWrite(gPin, 0);
  analogWrite(bPin, 0);
  pwm.setPWM(4, 0,130);
  pwm.setPWM(5, 0,455);
  pwm.setPWM(6, 0,420);
  pwm.setPWM(7, 0,420);
  pwm.setPWM(8, 0,420);
      break;
  }
  delay(180);
}
```

Figure: Void loop

Section 7: Lessons learnt and suggestions (1 page)

Describe using a numbered list, any lessons you learnt in the project. You should include things like, what went wrong and why, how you fixed the problem and so on. Keep the description brief and to the point.

1. 3D printing has its pros and cons. This project brought out the flaws in 3D printing such as its inability to print thin pieces and its flimsiness with detailed and small geometries. Our initial designs were to have the entire structure 3D printed however the vertical supports where too thin and the printer couldn't print properly. As a result black .5" thick foam poster board was purchased and was used for most of the structure. This gave us

enough support for holding the 3D-printed eye mounts and all the servos along with it. The eyes and the eye's mounting piece on the eye bracket were extremely flimsy. When trying to connect the two by universal joints, the glued in joints would become difficult to move and would cause the extruding piece that connects to the joint to break. This made it difficult to produce smooth x and y motion because the eye could not pivot from a stationary end of the universal joint. The solution was to use a small screw and attach it to the eye via super glue and that screw would be superglued to the universal joint. Precautions were taken of not placing the glue on the bearings of the joints to ensure free rotation.

2. One of the biggest challenges was soldering the components together to save space and being able to build a controller that was away from the motors. The controller's components such as the joysticks and buttons required soldering with resistors to properly function. This was a problem when the entire controller was assembled and soldered but resistors to the right and left bumper buttons were missing a resistor. The solution to the problem was to use a completely new set of buttons that would be soldered in series with a resistor and gluing them on top of the previous buttons. Soldering all the things to minimize space was difficult to work in such tight space and if the system was missing components then a whole new soldering process would have to take its place. However, through soldering the project was given the ability to have the most interactive design of the entire projects.

3. Getting the sound to work was one of the most difficult challenges especially when trying to sync it with the mouth's movement. This was solved by adding conditional statements in the code to be able to perform both motor movement and playing the audio file simultaneously.

4. The use of weekly milestones were helpful in keeping the project on track. This not only helped form our design as the weeks went on, but identified flaws in the structure and code. Understanding the constraints from our initial plans led us to modifying the design in order to meet the milestones, however completing the milestones gave us a straight enough path to complete the project by the time of our presentation.

Indicate using a numbered list at least two suggestions on how you can improve based on your experience with the project.

1. Make this a project competition like the battle of the bots previous year. For example students in the same lab will compete each other so at the end it will be the best face for each lab facing each other. The judging set on the complexity of the animatronic, the hardware use, sensors use. This will make the students engage students to work more and the best animatronic face for the lab gets the highest grade and the finals will be extra credit.

2. For sound, provided students with some examples because during the class several students were complaining that they couldn't get the sound to work. For examples for our project the sound quality is very low and to amplify the sound, we would increase the volume up to 400%. The problem by amplifying the sound, module need it an amplifier

for the sound to be smoother rather than a high pitch. This is one method the team could have improved on, by using a amplifier to be able to hear the output sound even during presentations. During our presentation, the animatronic face worked perfectly but the sound could not be appreciated due to the loud surrounding environment.

3. Implementing another servo motor to function as a neck would have been another improvement the team could've done. This would allow rotation of our face for a more complete product.

Section 8: Personnel and bill of materials

(a) Personnel

Task	Main Personnel	Secondary personnel
Eye bracket design	Emiliano Rodriguez	Tony Amaro
Mouth design	Tony Amaro	George Gonzalez
Supporting structure design	George Gonzalez	Emiliano Rodriguez
Face design	George Gonzalez	Tony Amaro
Motor/servor controls	Tony Amaro	George Gonzalez
Sensor Interfacing	Emiliano Rodriguez	Tony Amaro
Overall Programming	Emiliano Rodriguez	Tony Amaro
Controller design	Emiliano Rodriguez	George Gonzalez
Report Writing	All of the team	

(b) Bill of materials

No.	Description	Website/comment	Qty.	Unit $	Total $
1	Arduino MEGA 2560	Provided	1		
2	Micro Servos		9		
3	Mini MP3 Module	Amazon (pack of 2)	1	4.69	9.38
4	Speakers	Amazon (pack of 2)	2	7.87	7.87
5	Ultrasonic Sensor	Bought the kit while ago	1		
6	RBG LED				
7	Paper Clips	Found them			
8	Makeup	Dollargeneral	1	1.08	1.08
9	Screw Caps				
10	Styrofoam Poster Board and super glue	Hobby Lobby	1	14.05	14.05
11	Joysticks	Bought the kit while ago			
12	Push Buttons	Bought the kit while ago			
13	Eyes	3D Printed			
14	Eyelids	3D Printed			
15	Universal Joints	Amazon (pack of 3)	1	10.33	10.33

References :

[1] Animatronics Face of UTSA's President, Dr. Ricardo Romo, https://youtu.be/xkze1_hnam0

[2] ROMOBOT - ANIMATRONIC FACE ROBOT
https://www.instructables.com/id/RomoBOT-Animatronic-Face-Robot/

[3] Animatronic Neck - *PyroElectro.com*.
http://www.pyroelectro.com/tutorials/animatronic_neck/index.html

[4] Device Programmer - Ando
http://www.j-fsg.co.jp/pdf/Programmer_e.pdf

[5] How to make your own Animatronics Robot at home
https://www.youtube.com/watch?v=2bN3ZSUr-QQ

[6] HACKADAY.IO - Open Animatronics
https://hackaday.io/project/16193-open-animatronics

[7] HACKADAY.IO - Open Animatronics Instructions
https://hackaday.io/project/16193/instructions

Appendix A: Components Used

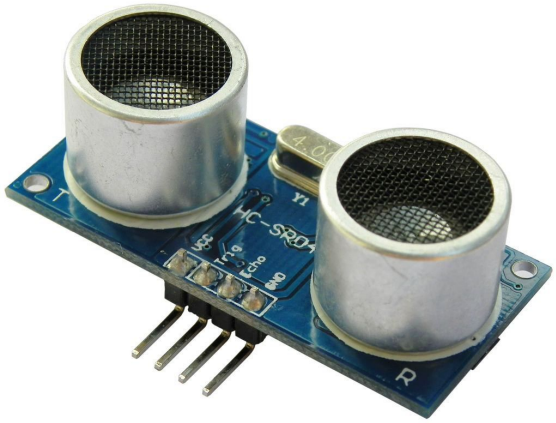

Figure # Ultra sonic Sensor

Figure ## Push Button

Figure ## RBG LED

Figure ## Joystick

Appendix B: Code

```
class Mp3Notify
{
public:
  static void OnError(uint16_t errorCode)
  {
    // see DfMp3_Error for code meaning
    Serial.println();
    Serial.print("Com Error ");
    Serial.println(errorCode);
  }

  static void OnPlayFinished(uint16_t globalTrack)
  {
//    Serial.println();
//    Serial.print("Play finished for #");
//    Serial.println(globalTrack);
  }

  static void OnCardOnline(uint16_t code)
  {
    Serial.println();
    Serial.print("Card online ");
    Serial.println(code);
  }

  static void OnCardInserted(uint16_t code)
  {
    Serial.println();
    Serial.print("Card inserted ");
    Serial.println(code);
  }

  static void OnCardRemoved(uint16_t code)
  {
    Serial.println();
    Serial.print("Card removed ");
    Serial.println(code);
  }
};

SoftwareSerial secondarySerial(10, 9); // RX, TX
DFMiniMp3<SoftwareSerial, Mp3Notify> mp3(secondarySerial);


void(*resetFunc)(void) =0;
void setup() {
  randomSeed(analogRead(A2)); //seeds random number for random sequence upon start
```

```
  pinMode(rPin, OUTPUT);
  pinMode(gPin, OUTPUT);
  pinMode(bPin, OUTPUT);
 pwm.begin();
 Serial.begin(9600);
 pwm.setPWMFreq(60);  // This is the maximum PWM frequency
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode (x, INPUT) ;
  pinMode (y, INPUT) ;
  pinMode (ym, INPUT) ;
  delay(500);

   mp3.begin();
   mp3.setVolume(30);
   mp3.playMp3FolderTrack(18);
}

void loop() {
if(mode > 3)
  resetFunc();

Serial.println(mode);

if(digitalRead(2)==HIGH)
  mode++;

switch (mode) {
    case 1:
  mp3.stop();
  Serial.println("1");
  analogWrite(rPin, 0);
  analogWrite(gPin, 0);
  analogWrite(bPin, 100);
  joyStick();
      break;
    case 2:
       Serial.println("2");
  analogWrite(rPin, 100);
  analogWrite(gPin, 0);
  analogWrite(bPin, 0);
       distance();
     break;
    case 3:
  Serial.println("3");
  analogWrite(rPin, 100);
  analogWrite(gPin, 0);
  analogWrite(bPin, 100);
```

```
digitalWrite(trigPin, HIGH);
delayMicroseconds(10); // Added this line
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;
Serial.println(distance);
pwm.setPWM(4, 0,300);
if(distance == 70) {
Serial.println("YES");
mp3.playMp3FolderTrack(5);

delay(100);
  pwm.setPWM(5, 0,420);
  pwm.setPWM(6, 0,500);
  pwm.setPWM(7, 0,420);
  pwm.setPWM(8, 0,500);

  pwm.setPWM(4, 0,300);
  delay(1500);
  pwm.setPWM(4, 0,130);
  delay(250);
  pwm.setPWM(4, 0,300);
  delay(1300);
  pwm.setPWM(4, 0,130);
  delay(100);
  pwm.setPWM(4, 0,300);
  delay(1250);
  pwm.setPWM(4, 0,130);
  delay(150);
  pwm.setPWM(4, 0,300);
  delay(1250);
  pwm.setPWM(4, 0,130);
  delay(150);
  pwm.setPWM(4, 0,300);
  delay(1700);
  pwm.setPWM(4, 0,130);
  delay(150);
  pwm.setPWM(4, 0,300);
  pwm.setPWM(4, 0,130);
  delay(150);
  pwm.setPWM(4, 0,300);
  delay(1250);
  pwm.setPWM(6, 0,420);
  pwm.setPWM(8, 0,420);
  delay(100);
  pwm.setPWM(6, 0,500);
  pwm.setPWM(8, 0,500);
  delay(10);
```

```
      pwm.setPWM(4, 0,130);
      delay(150);
      pwm.setPWM(4, 0,300);
      delay(1250);
      pwm.setPWM(4, 0,130);
      delay(150);
      pwm.setPWM(4, 0,300);
      delay(1300);
      pwm.setPWM(4, 0,130);
      delay(150);
      pwm.setPWM(4, 0,300);
      delay(3500);
      Serial.println("flag=1");
      flag = 1;
      delay(50);
    }

}

flag = 0;
     pwm.setPWM(5, 0,500);
     pwm.setPWM(6, 0,500);
     pwm.setPWM(7, 0,500);
     pwm.setPWM(8, 0,500);
delay(500);
for(int i=0; i<2;i++) {
     pwm.setPWM(5, 0,420);
     pwm.setPWM(6, 0,420);
     pwm.setPWM(7, 0,420);
     pwm.setPWM(8, 0,420);
     delay(200);
     pwm.setPWM(5, 0,500);
     pwm.setPWM(6, 0,500);
     pwm.setPWM(7, 0,500);
     pwm.setPWM(8, 0,500);
     delay(300);
     pwm.setPWM(5, 0,420);
     pwm.setPWM(6, 0,420);
     pwm.setPWM(7, 0,420);
     pwm.setPWM(8, 0,420);
     delay(100);
     pwm.setPWM(5, 0,500);
     pwm.setPWM(6, 0,500);
     pwm.setPWM(7, 0,500);
     pwm.setPWM(8, 0,500);
     delay(50);
}
```

```
  while(flag != 1) {
     if(digitalRead(2) == HIGH) {
  mode++;
   delay(500);
   return;
   }
     long duration, distance;
     digitalWrite(trigPin, LOW);  // Added this line
     delayMicroseconds(2); // Added this line
     digitalWrite(trigPin, HIGH);
     delayMicroseconds(10); // Added this line
     digitalWrite(trigPin, LOW);
     duration = pulseIn(echoPin, HIGH);
     distance = (duration/2) / 29.1;
     Serial.println(distance);
     pwm.setPWM(4, 0,300);
         if(distance == 100) {
     Serial.println("YES");
     mp3.playMp3FolderTrack(7);

       delay(500);
       pwm.setPWM(5, 0,500);
       pwm.setPWM(6, 0,420);
       pwm.setPWM(7, 0,500);
       pwm.setPWM(8, 0,420);

    for(int i=0;i<2;i++) {
       pwm.setPWM(4, 0,130);
       delay(100);
       pwm.setPWM(4, 0,300);
       delay(300);
    }
delay(1000);

       pwm.setPWM(4, 0,130);
       delay(100);
       pwm.setPWM(4, 0,300);
       delay(300);
       pwm.setPWM(4, 0,130);
       delay(100);
       pwm.setPWM(4, 0,300);
       delay(600);

       pwm.setPWM(4, 0,130);
       delay(100);
       pwm.setPWM(4, 0,300);
       Serial.println("here");
       delay(1300);
```

```
        pwm.setPWM(4, 0,300);
        delay(2800);
//hmmm IDK
        pwm.setPWM(4, 0,130);
        delay(80);
        pwm.setPWM(4, 0,300);
        delay(200);

        pwm.setPWM(4, 0,130);
        delay(80);
        pwm.setPWM(4, 0,300);
        delay(200);
        pwm.setPWM(4, 0,130);
        delay(150);
        pwm.setPWM(4, 0,300);
        delay(1700);
      Serial.println("CP  1");

        pwm.setPWM(4, 0,130);
        delay(150);
        pwm.setPWM(4, 0,300);
        delay(3750);

        pwm.setPWM(4, 0,130);
        delay(150);
        pwm.setPWM(4, 0,300);

        delay(1700);
        pwm.setPWM(4, 0,130);
        delay(150);
        pwm.setPWM(4, 0,300);

        delay(1500);
        pwm.setPWM(4, 0,130);
        delay(150);
        pwm.setPWM(4, 0,300);


        delay(2750);
        pwm.setPWM(4, 0,130);
        delay(150);
        pwm.setPWM(4, 0,300);

        delay(300);
        //Thats alot of guess work
        pwm.setPWM(4, 0,130);
        delay(150);
        pwm.setPWM(4, 0,300);
```

```
        pwm.setPWM(4, 0,300);

        delay(200);
        pwm.setPWM(4, 0,130);
        delay(150);
        pwm.setPWM(4, 0,300);


        delay(200);
        pwm.setPWM(4, 0,130);
        delay(150);
        pwm.setPWM(4, 0,300);


        flag = 1;
      }
   }
   pwm.setPWM(4, 0,130);
     delay(250);
     mp3.playMp3FolderTrack(8);
     delay(1000);
     pwm.setPWM(4, 0,300);
     mp3.stop();


        while(1) {
          delay(150);
          if(digitalRead(2) == HIGH) {
            mp3.stop();
            mode++;
            delay(500);
            return;
          }
        }
}

void easterEgg() {
  mp3.playMp3FolderTrack(17);
  delay(500);
}

void joyStickOnly1() {
  y = map(analogRead(A1), 0, 1023, 380, 300);
  x = map(analogRead(A0), 0, 1023, 240,330);

  ym = map(analogRead(A3), 0, 1023,500, 130);
```

```
void easterEgg() {
  mp3.playMp3FolderTrack(17);
  delay(500);
}

void joyStickOnly1() {
  y = map(analogRead(A1), 0, 1023, 380, 300);
  x = map(analogRead(A0), 0, 1023, 240,330);

  ym = map(analogRead(A3), 0, 1023,500, 130);


  if(digitalRead(4) ==HIGH) {
      pwm.setPWM(5, 0,455);
      pwm.setPWM(6, 0,420);
      pwm.setPWM(7, 0,420);
      pwm.setPWM(8, 0,420);
  }else {
    pwm.setPWM(5, 0,500);
    pwm.setPWM(6, 0,500);
    pwm.setPWM(7, 0,500);
    pwm.setPWM(8, 0,500);
  }
  pwm.setPWM(0, 0,x);
  pwm.setPWM(1, 0,x + 20);
  pwm.setPWM(2, 0,y);
  pwm.setPWM(3, 0,y + 15);
  pwm.setPWM(4, 0,ym);


    if(digitalRead(2) == HIGH) {
    mp3.stop();
    mode++;
    return;
  }
  delay(2);
}

void joyStickOnly2() {

  y = map(analogRead(A1), 0, 1023, 380,300);
  x = map(analogRead(A0), 0, 1023, 240, 330);

  ym = map(analogRead(A3), 0, 1023, 500,130);


  if(digitalRead(4) ==HIGH) {
      pwm.setPWM(5, 0,470);
```

```
void joyStickOnly2() {

  y = map(analogRead(A1), 0, 1023, 380,300);
  x = map(analogRead(A0), 0, 1023, 240, 330);

  ym = map(analogRead(A3), 0, 1023, 500,130);


  if(digitalRead(4) ==HIGH) {
      pwm.setPWM(5, 0,470);
      pwm.setPWM(6, 0,420);
      pwm.setPWM(7, 0,420);
      pwm.setPWM(8, 0,420);
  }else {
    pwm.setPWM(5, 0,500);
    pwm.setPWM(6, 0,500);
    pwm.setPWM(7, 0,500);
    pwm.setPWM(8, 0,500);
  }
  pwm.setPWM(0, 0,x);
  pwm.setPWM(1, 0,x);
  pwm.setPWM(2, 0,y);
  pwm.setPWM(3, 0,y);
  pwm.setPWM(4, 0,ym);
  delay(2);
  totalSecret = 0;
  secSecret = millis();
  while(digitalRead(4) == HIGH){
    Serial.println(totalSecret);
      pwm.setPWM(5, 0,470);
      pwm.setPWM(6, 0,420);
      pwm.setPWM(7, 0,420);
      pwm.setPWM(8, 0,420);
        totalSecret = (double)(millis() - secSecret) / CLOCKS_PER_SEC;
        if(totalSecret > 8)
            easterEgg();
          }
    if(digitalRead(2) == HIGH) {
    mp3.stop();
    mode++;
    return;
  }

}
```