**ANIMATRONICS FACE: PROFESSOR JAMES JOHNSON**



Youtube Link:

<https://youtu.be/qiCpQciHe40>

CJ Agbanlog

Dept. of Mechanical Engineering

San Antonio, TX, USA 78249

[Rowdy.runner2@utsa.edu](mailto:Rowdy.runner2@utsa.edu)

Sergio Alva

Dept. of Mechanical Engineering

San Antonio, TX, USA 78249

sergioalva17@yahoo.com

Victor Nelson\*

Dept. of Mechanical Engineering

San Antonio, TX, USA 78249

victor.m.nelson.17@gmail.com

Submitted as part of project for ME4543 Mechatronics, Fall 2018

TABLE OF CONTENTS

|  |  |
| --- | --- |
| Abstract | 3 |
| Literature Review | 4 |
| Supporting Structure | 6 |
| Joints and Motors | 6 |
| Sensors | 6 |
| Programming | 6 |
| Lesson learnt and suggestions | 7 |
| Personnel | 10 |
| Bill of Materials | 10 |
| References | 11 |
| Appendix A | 13 |

**ABSTRACT**

The purpose of this project was to create an animatronic face of an assigned engineering professor and use different sensors to move different facial features and have it say a soundbite. The structure of the face could be crafted using cardboard, wood and felt. In order to move the facial features, servo motors were attached to the mouth and eyebrows and connected to an Arduino and breadboard using jumper wires. The sensors were then to be connected and set up, and the code was then written and uploaded into the Arduino. An mp3 shield was to be connected to the Arduino holding a micro sd chip which would allow for the sound bite to be played. Once all of this was done, each of the sensors could then set off the movement of the servo motors attached to the assigned facial features.

**Section 1: Literature review**

**Design 1**

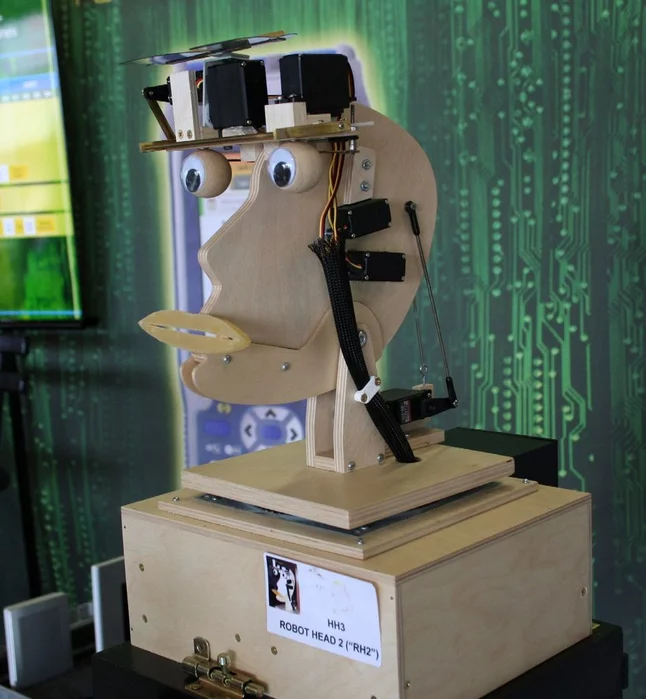
For our first animatronic face design, the group decided on using a flat sheet of plywood for the face similar to Figure 1 shown below with a few differences. The flat surface of the wood would be used for the face with the chose professor’s picture being traced into and placed on it. The parts that would move would then be separately cut out of the face and replaced with movable objects. Some of the main components used in this design include a servo controller, wires, power strip, micro switch, a single board computer and other items. Tools required to complete the project would include power drills, saws, wire strippers, a soldering gun, heat gun and other basic tools such as a hammers, screwdrivers and pliers. Sensors that would be incorporated into the design would be infrared sensors range sensors. The main purpose of the project for the creator was to stay busy during his free time. The instructions provided by the creators gave six main steps. The first step was to design the head. This was done using a ½ inch piece of plywood. The creator cut out the head design and then fitted the servo controllers onto the eyes, molded a wire to the lips to attach them to the head and inserted a screw and rod to the jaw and head to allow them to tilt. The second step dealt with the electronic portion of the project, with the main components being the servo controller and RAPU. The servo controller was to take the series signals from the RAPU and convert them into movements. The power supplies and speakers were placed inside a black box beneath the head with a fan dissipating any heat buildup. The third step called for connecting a spotlight to the head. This was not only to be used for lighting but also for mimicking certain sounds. The next step was to set up the robot head by attaching a power strip to the trunk and attaching the RAPU unit to the back and then plug in the power cable. The last step was to design a trunk to place all of the components required to power up the robot face. [3]

Figure RAPU Talking Hhead

**Design 2**

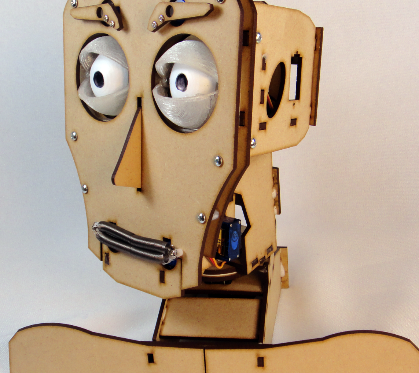
With regards to figure 2, the following design has thirteen motors that supports eyelids, eyeballs, eyebrows, lips, jaw and neck, the eyelids and the mouth. The moving parts are programmed using Fritz control Software for Windows utilizing Dot Network Framework in combination with Arduino. Firmware code is provided that allow customization of the 13 moveable parts. The Face is powered by an AC adapter or a battery pack; the other components include two standard servo motors, 11 micro servo motors, an Arduino Uno, servo shield, and bearing disk. The facial features can be modelled using a template to provide the exact dimensions. The goal of the head was to create a robotic puppet head that can be controlled using an app or a joystick [4]. The design would be modified to accommodate two sensors that could be programmed and added to control for movement. A distance sensor can be programmed to initialize the programmed movements, and a microphone in combination with a sound sensor can be used to initialize programmed movements to essentially make the face voice activated. Both sensors will be programmed using Arduino and initialize based on predetermined conditions.

Figure Fitz Talking Head

**Design 3**

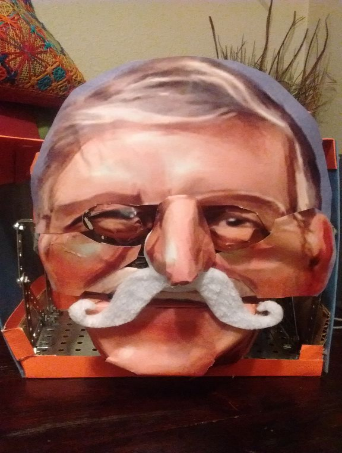
The third design researched was the Romo-bot animatronic face. The project would be used as a reference with the facial design being changed. This was done as a major project for an independent study. The material used for the structure of the face was a plastic mask but for the group’s project, wood would be used instead with a picture of the professor placed on it. The instructions that were provided called for the frame to be made out of an erector set, starting with the eye assembly. The frame was then constructed to support the eyes and required four motors. Two for each eye. One in the x direction and one in the y direction. The jaw was then made using a large servo motor to attach the upper and lower jaw allowing for mouth movement. Another motor was then used to give the nose movement. The next step was to include the audio using computer programs that converted the audio files to MP3 format allowing for face to talk. The eyes were then coded and then the wires were connected to the Arduino kit. The sensors that are planned on being used are touch and range sensors. [5]

Figure President Romo Head

**Section 2: Brainstorming (initial planning)**

The professor assigned to us is Professor James Johnson. To meet the requirements for two moving parts our goal is to have a (1) moveable bottom lip and jaw that corresponds with and (2) moveable eyebrows and potentially the eyes. For the chassis we intend to use a combination of wood and cardboard with a color print out of Professor Johnson’s face laid on the chassis. The cut out will most probably be glued onto the chassis. The head will be supported by a box with a 12”x12”x6” box that will have a UTSA shirt cut out laid on the box. In order to move the mouth, 9G micro servo motors will be connected and programmed using an Arduino and Arduino software and powered by a battery back with enough voltage to control the movements. The eyebrows and potentially the eyes will also be moved using the 9G servo motors. The sensors that are planned on being used are touch and distance sensors. These will also be powered by the batteries. When the animatronic face is touched, this will trigger the voice and movement of the mouth and eyebrows. The distance or range sensor that uses ultrasonic waves to determine distance in order to trigger the movements when something is placed within a certain distance. To implement the soundtrack, a recording of Professor Johnson’s voice will be made and converted into an mp3 file. The voice will be activated in combination with the touch and the motion sensor. Distance will be within three feet and the touch sensor will be mounted on the forehead. To meet the minimum 5 word requirement we plan to ask Professor Johnson what is acceptable during our next meeting. One potential statement we have is “I would have gotten away with it if it weren’t for you senior design kids” which we got from Scooby Doo.  The other potential statement we have is “I would have spent more than the 7 minutes allotted ripping you up.” However, we intend to first speak with professor Johnson to get his approval. To synchronize his soundtrack to the movement of the jaw we intend to study how a person’s jaw moves when saying the same statements.

**Section 3: Supporting structure**

The supporting structure was constructed using cardboard. A 12”x12”x6” box was made which would be used to house the Arduino, breadboard and wires. Tape and hot glue were used to help shape the box and keep it together. After this was completed, extra cardboard was used to create a vertical structure on top of the box. This is where the face was to be placed. The face was then printed and cut onto cardboard where the mouth and eyebrows were cut out, allowing for the servo motors to be connected and taped to. The face was then placed on the vertical structure and connected with tape and super glue. Wood was the glued and nailed onto the 12”x12”x6” box, giving it a nicer finish. Felt was then stapled onto the top of the box to cover what was left of the cardboard.

**Section 4: Joints and motors**

The joints that were chosen to be moved were the bottom jaw and both of the eyebrows. The joints were to be moved by the chosen sensors being triggered. Once the sensors were triggered, the connecting servo motors would then rotate each part. The first step in this process was to cut out and craft the desired facial joints being the eyebrows and the mouth. The servo motors would the be taped to each part and then connected to the breadboard and the arduino. The sensors were then also connected and the code was set up to allow the sensors to trigger the servo motors. The servo motors would then rotate at the given degrees.

**Section 5: Sensors**

The chosen sensors for our project were an ultrasonic sensor and a tilt sensor. The ultrasonic sensor was chosen for range detection. When an object was detected within the given range distance implemented in the code, the sensor was to detect it which in turn would set off the servo motors. This worked better with objects that were flat and at the distance specified in the code but also worked with any object just as long as the sensor detected it at that range even if the object was backed up to the desired distance. The tilt sensor allowed for the servo motor to be activated when a change in the angle of the sensor was detected. The tilt sensor consisted of a mercury ball inside of a clear, plastic casing that would move and light up an LED light that was connected to it when the angle was changed. Once the change in angle triggered the sensor, the servo motors would move the same way as the ultrasonic sensor.

**Section 6: Programming**

In order to code the servos that controlled the joints to respond to the sensors, the goal was to use the pre-existing libraries, Servo and NewPing to allow the Arduino MEGA 2560 to sync the moveable parts to the Professor’s sound clip. Within the sketch, the servos, and the tilt and ultrasonic sensor had to be assigned to a pin, and the keyword const int was used to define a pin on the Arduino MEGA to read the variables from the sensors that would provide feedback to the servos after being assigned in the void setup. The keyword int was used for the tilt sensor to read and write variables that corresponded to the position of the tilt sensor; the Boolean keyword was used to indicate whether the tilt sensor was reading a tilted position as true or false, which was necessary for the following else-if statements.

The next step was to use the NewPing library to define a maximum distance that the ultrasonic sensor would begin to read, which was chosen as 50cm. After calling on the library, the servos that were used for the joints were defined. Three servos were used, one servo for each eyebrow and one servo for the mouth; the servos for the eyes were ran together and responded to the inputs from the sensors together by being placed in series, the eyebrows were defined as servo3. The mouth was chosen as servo 1 and defined by using the Servo keyword.

Next a void setup was created to initialize the following void loop. The void setup is run once and it “initializes variables, pin modes, libraries etc.” [10]. Within the void setup, Serial.begin(9600) initialized the serial connection between the Arduino and the computer, servo. Attach attached a servo variable to a pin to relate the sensors to the servos, and pinMode was used configure the tilt switch to read inputs.

After the void setup, the void loop function was used to call on and loop two functions depending on whether or not the conditions were met. The first program defined the conditions for ultrasonic sensor to power the joints. An if else statement was used that looked at the distance an object was away from the ultrasonic sensor to illicit a response. Because the ultrasonic was rapidly sending and receiving outputs and inputs, the range to activate the servos was defined as greater than or equal to 35cm and less than or equal to 37cm, if these conditions were met, the servos would respond and the eyebrows and mouth would move. Within the if statement servo.write() was used to control the angle of the servo and delay() was used to indicate how long the servo would remain at the angle specified by servo.write. The servo angles and delay were selected to mimic the mouth position and timing of the opening and closing of the mouth to the sound clip provided by the professor. Servo.write() and delay() was also used to control the angle position and time of the servos powering the movement of the eyebrows. An else statement was used for outside of the range 35 to 37 cm range, effectively from 0-34 cm and 38-50cm that have the servos hold a position representing a resting position.

The second program defined by the void loop took advantage of the tilt sensor and an else if statement similar in structure to that of the ultrasonic sensor. The biggest change between the tilt sensor and the ultrasonic sensor was the conditions for the if else statements, and the tilt sensor used an if statement in the original if statement and an if statement in the else clause. The first if statement used Tilt Val read the digital input of the tilt sensor; for the if statement, tiltVAL == HIGH was used to tell the Arduino that the tilt sensor was in fact tilted. If the sensor was tilted, the Boolean defined the condition as true or false; if it was true, the tilt sensor was tilted, the same code that powered the mouth and eyebrows in the if statement of the ultrasonic sensor ran and the face joints moved.  If not, an else statement was used to keep the face at rest that used an if that maintained the face at rest if the Boolean was False.

The void loop continuously ran the codes for both sensors and the appropriate code would be called depending on what sensor was activated. The correct code was selected based on which pin read the variable, the pins and their corresponding sensor were defined using the const int function in the beginning of the sketch. The sensors only responded to the conditions defined in the if statements, otherwise the else statements were ran and the face remained at rest.

**Section 7: Lessons learnt and suggestions (1 page)**

**1.** Punctuation is very important when writing the sketch, if proper punctuation isn’t followed with respect to opening and closing brackets, opening and closing brackets, and semicolons, errors will populate causing you to read and reread your code causing you to second guess what you wrote. The reason punctuation is important is because it separates and at the same time groups the functions the Arduino is supposed to call. The problem was fixed by googling the errors and reading and trying to figure out the problem.

2. Structure is equally important. Libraries have to be defined followed by the pins, then the setup, then the loop. Within the setup, the proper initial conditions have to be defined as well as within the loops. Additionally, if-else statements have to follow a particular order and can vary depending on the sensor.

3. Wire management is very important. Staying organized, and grouping jumper cables will save a lot of headache downstream especially when trying to troubleshoot. Initially the range of the jumper cables were short and cables that were constantly being unplugged from the breadboard or Arduino would lead to servos that didn’t work and lots of questions. Similar colored jumper cables were used to extend the range and keep straight positive, negative, and input cables. Electrical tape was used to group together wires.

4. Sometimes a servo will not respond even though the sketch was successfully verified and uploaded. Sometimes the fix could be as easy as a wiring problem, sometimes it could as cumbersome as a coding issue.

5. A big lesson is not to plug your computer into the Arduino board when you have three 9 Volt batteries connected to the hardware.

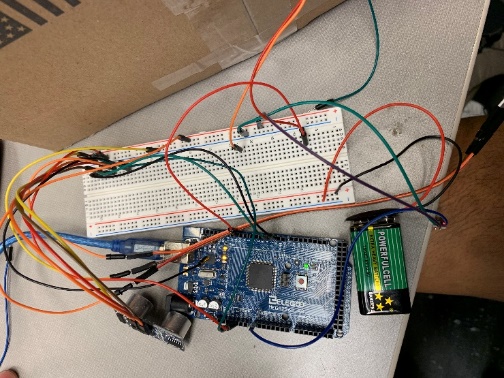
6. Things will never go as expected. Everything takes time and work to put in a decent project.

**B. How to improve**

1. The best way to improve is to code more using different goals, different sensors, and different responses.

2. The second-best way to improve is to stay organized and keep track of punctuation.

3. The third best way to improve is to have an idea of the structure you want to create

4. The fourth best way to improve is to become more verse in writing the conditions for if else statements.

5. Keep organized. Wire management helps trace potential problems to hardware issues. If hardware is connected, code can be checked next. If the next the project is run, and the code is good, the wires can be checked to determine bad jumper cables, or cables not connected. Figure 4 shows what cables started out as without using the whole breadboard. Figure 5 shows the more manageable set up on the breadboard and Figure 6 shows the cable set up on the Arduino Mega.

Figure Original Setup

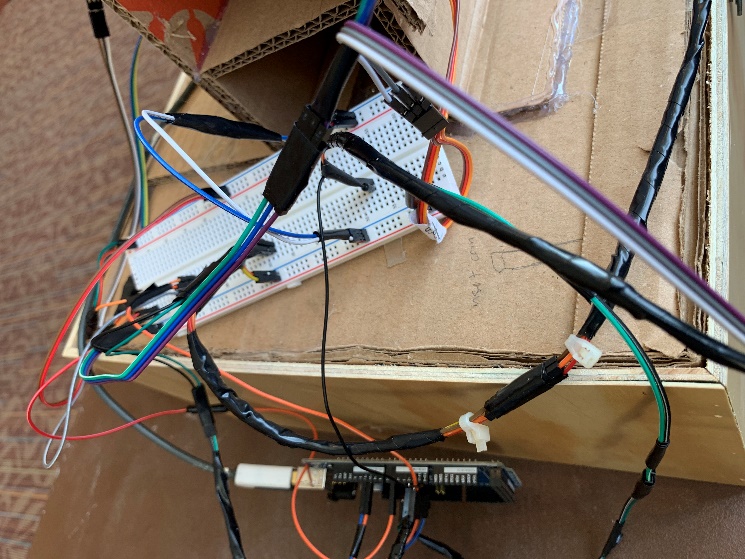


Figure Final Breadboard

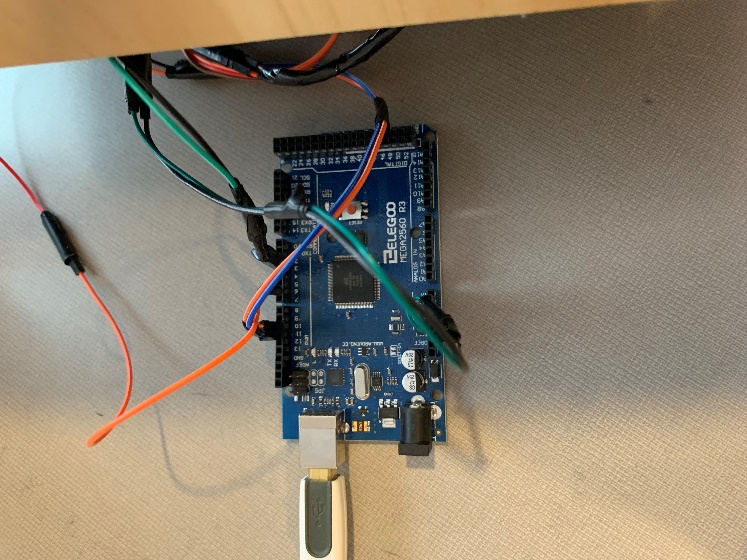


Figure 6 Final Arduino

**Section 8: Personnel and bill of materials**

**(a) Personnel**

|  |  |  |
| --- | --- | --- |
| **Task** | **Main Personnel** | **Secondary personnel** |
| Overall programming and integration | CJ Agbanlog | Sergio Alva |
| Chassis | Sergio Alva | CJ Agbanlog |
| Creating joints and motor interfacing | CJ Agbanlog | Sergio Alva |
| Face Design | CJ Agbanlog | Sergio Alva |
| Research | Sergio Alva | CJ Agbanlog |

**(b) Bill of Materials**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Description | Website/comment | Qty. | Unit $ | Total $ |
| 1 | Arduino MEGA 2560 | Provided | 1 |  |  |
| 2 | Wood | Scrap |  |  |  |
| 3 | Adafruit Music Maker MPS Shield for Adruino | <https://www.amazon.com/Adafruit-Music-Shield-Arduino-Stereo/dp/B01BT4N386/ref=sr_1_3?s=electronics&ie=>  UTF8&qid=1543908505&sr=1-3&keywords=arduino+mp3+shield | 1 | 32.49 | 32.49 |
| 4 | TowerPro MG90s Micro Server | <https://www.amazon.com/RioRand-MG90S-Metal-Geared-Helicopter/dp/B00M8SR0ZK/ref=pd_lpo_vtph_21_bs_tr_t_>  1?\_encoding=UTF8&psc=1&refRID=JDET01BNZ9RP14V  E7MGW | 4 | 16.99 | 16.99 |
| 5 | Universal Servo Cable Wire Connector Crimp Pin Kit | https://www.amazon.com/s/ref=nb\_sb\_noss\_1?url=search-alias%3Delectronics&field-keywords=servo+connectors | 1 | 8.99 | 8.99 |
| 6 | Cardboard | Home Depot | 1 | 5.04 | 5.46 |
| 7 | Breadboard | Provided | 1 |  |  |
| 8 | Multicolored Breadboard Dupont Jumper Cables | <https://www.amazon.com/Multicolored->  Breadboard-Dupont-Jumper-Wires/dp/B073X7P6N2/ref=sr\_1\_1\_sspa?s  =electronics&ie | 1 | 6.89 | 6.89 |
| 9 | Double Sided Gorilla Tape | Walmart | 1 | 5.83 | 6.31 |
| 10 | 9V Batteries |  | 8 | 25.94 | 28.08 |
| 11 | Hot Glue Sticks | Used to hold together the chassis | 1 | 2.67 | 2.89 |
| 12 | Packing Tape | Left Over |  |  |  |
| 13 | Super Glue | Walmart. | 1 | 1.77 | 1.91 |
| 14 | Felt | Walmart. Blue and orange felt used to cover some of cardboard. | 2 | .97 | 2.10 |
| 15 | Metal Pins | Walmart. Used to connect mouth joint to servo pin, connected to servo motor. Allowed mouth to swivel. | 1 | 1.24 | 1.34 |
| 16 | Cotton Balls | Walmart. Used for hair and goatee. | 1 | .97 | 1.05 |
| 17 | Notebook cardboard paper | Walmart. Used the back of legal pad for lightweight cardboard to be moved by the joints. | 1 | 2.44 | 2.64 |
|  |  | TOTAL |  | 112.23 | 117.14 |

**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]First Design Idea

<https://www.instructables.com/id/Talking-Animatronic-Robot-Head/>

[4] Design Idea Two

<https://www.instructables.com/id/FRITZ-ANIMATRONIC-ROBOTIC-HEAD/>

[5] Design Idea Three

<https://www.instructables.com/id/RomoBOT-Animatronic-Face-Robot/>

[6] Control servo with ultrasonic distance sensor and Arduino Uno, <https://www.youtube.com/watch?v=W6YNDse4-aw>

[7] Servo Motor SG-90, <https://components101.com/servo-motor-basics-pinout-datasheet>

[8] [HC-SR04](https://components101.com/ultrasonic-sensor-working-pinout-datasheet) Ultrasonic Sensor, <https://components101.com/ultrasonic-sensor-working-pinout-datasheet>

[9] Keyes KY-017 Arduino Mercury Tilt Switch: Tutorial, <http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/keyes-ky-017-arduino-mercury-tilt-switch-tutorial/>

[10] Void Set-UP

https://www.arduino.cc/reference/en/language/structure/sketch/setup/

**Appendix A: Code**

#include <Servo.h>

#include <NewPing.h>

int sensorValue = 0; // value read from the pot

int outputValue = 0; // value output to the PWM (analog out)

const int ServoPin = 11;

const int TriggerPin = 3;

const int EchoPin = 2;

const int analogInPin = A0; // analog input pin that the potentiometer is attached to

const int Brows = 10; // Servo 3

int tiltSwitch = 6; // Tilt Switch Input

int tiltVal; // variable to store tilt input

boolean bIsTilted ;// define numeric variables val

// 50 = maxDistance

NewPing sonar(TriggerPin, EchoPin, 50);

Servo servo1;

Servo servo3;

void setup () {

 Serial.begin(9600);

 servo1.attach(ServoPin);

 servo3.attach(Brows);

 pinMode (tiltSwitch, INPUT) ;// define the mercury tilt switch sensor output interface

}

void loop ()

{wheresvictor();

cj ();

}

void wheresvictor ()

{

 int cm = sonar.ping\_cm();

 Serial.println(cm);

 if (cm >= 35 && cm <= 37)

{

 // mouth closed START

 servo1.write(80);

 delay(100);

 // I

 servo1.write(130);

 delay(300);

 // can

 servo1.write(110);

 delay(300);

 //do

 servo1.write(90);

 delay(300);

 //some

 servo1.write(110);

 delay(90);

 servo1.write(90);

 delay(300);

 //thing

 servo1.write(110);

 delay(200);

 //with

 servo1.write(90);

 delay(200);

  //some

 servo1.write(110);

 delay(200);

 servo1.write(100);

 delay(200);

 // data

 servo1.write(110);

 delay(190);

 servo1.write(100);

 delay(190);

 // eybrow Start

 servo3.write(180);

 delay(90);

 servo3.write(30);

 delay(100);

 servo3.write(60);

 delay(150);

 servo3.write(80);

 delay(200);

 servo3.write(100);

 delay(100);

  // I

 servo1.write(130);

 delay(100);

 // can

 servo1.write(110);

 delay(200);

 //do

 servo1.write(90);

 delay(200);

 // no

 servo1.write(100);

 delay(200);

 //thing

 servo1.write(110);

 delay(190);

 //thing

 servo1.write(110);

 delay(190);

 //with

 servo1.write(90);

 delay(190);

 // no

 servo1.write(100);

 delay(190);

 // data

 servo1.write(110);

 delay(190);

 servo1.write(100);

 delay(190);

 // eybrow 2 Start

 servo3.write(180);

 delay(90);

 servo3.write(30);

 delay(100);

 servo3.write(60);

 delay(150);

 servo3.write(80);

 delay(200);

 servo3.write(100);

 delay(100);

}

else if (cm < 35 || cm > 37)

 {

     servo1.write(80);

   }

}

void cj ()

{

 tiltVal = digitalRead (tiltSwitch) ;// read the switch value

 if (tiltVal == HIGH) // Means we've tilted

 {

   if (!bIsTilted){

     bIsTilted = true;

    // I

 servo1.write(130);

 delay(300);

 // can

 servo1.write(110);

 delay(300);

 //do

 servo1.write(90);

 delay(300);

 //some

 servo1.write(110);

 delay(90);

 servo1.write(90);

 delay(300);

 //thing

 servo1.write(110);

 delay(200);

 //with

 servo1.write(90);

 delay(200);

  //some

 servo1.write(110);

 delay(200);

 servo1.write(100);

 delay(200);

 // data

 servo1.write(110);

 delay(190);

 servo1.write(100);

 delay(190);

 // eybrow Start

 servo3.write(180);

 delay(90);

 servo3.write(30);

 delay(100);

 servo3.write(60);

 delay(150);

 servo3.write(80);

 delay(200);

 servo3.write(100);

 delay(100);

  // I

 servo1.write(130);

 delay(100);

 // can

 servo1.write(110);

 delay(200);

 //do

 servo1.write(90);

 delay(200);

 // no

 servo1.write(100);

 delay(200);

 //thing

 servo1.write(110);

 delay(190);

 //thing

 servo1.write(110);

 delay(190);

 //with

 servo1.write(90);

 delay(190);

 // no

 servo1.write(100);

 delay(190);

 // data

 servo1.write(110);

 delay(190);

 servo1.write(100);

 delay(190);

 // eybrow 2 Start

 servo3.write(180);

 delay(90);

 servo3.write(30);

 delay(100);

 servo3.write(60);

 delay(150);

 servo3.write(80);

 delay(200);

 servo3.write(100);

 delay(100);

   }

 }

 else

 {

   if (bIsTilted){

     bIsTilted = false;

     Serial.println("not tilted");

   }

 }

}

**Appendix B: Arduino wiring**

You may use fritzing.org to create wiring diagrams.