12/10/2016

**Final Report for Einstein Group (Team Six)**

**ECE 478 Fall2016**

**Students:**

**Naser Alshami**

**Srijana Sapkota**

**Table of Contents**

1. Introduction 2
2. Fuzzy Implementation in Einstein Robot 3
   1. Introduction 3

2.2 Fuzzy Table 4

2.3 Fuzzy Rule 4

2.4 Fuzzification 5

2.5 Defuzzification 6

* 1. Results and Link to fuzzy video 10

3. Arm design 10

3.1 Pictures 11

3.2 Hardware and software used 12

3.3 Degree of freedom 13

3.4 Torque calculations 13

4. Modifications 14

5. Challenges 15

6. Future Recommendations 14

7. References 15

APPENDIX 16

A. Results of fuzzy logic 16

B. Arduino code for fuzzy logic 17

C. Arduino Sensor Shield v5.0 Board Layout 20

D. Bill of Materials 21

**E. Arduino code for Einstein 22**

1. **Introduction:**

In this course, our job is to design realistic proportional right arm that consists fingers, palm, elbow, shoulder for Einstein robot in Intelligent robotics lab at Portland State University. The task is to attach the arm to Einstein robot and demonstrate some complete behavior using software developed by the previous Einstein team. We need to demonstrate fuzzy logic using controller with sensors as inputs and that we can control any degree of freedom using software.

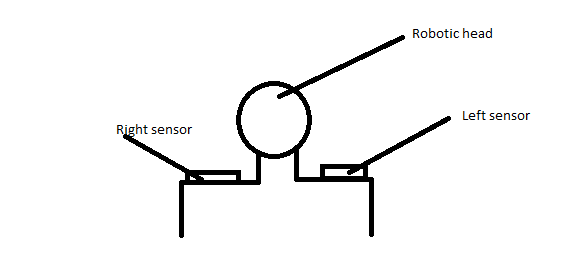
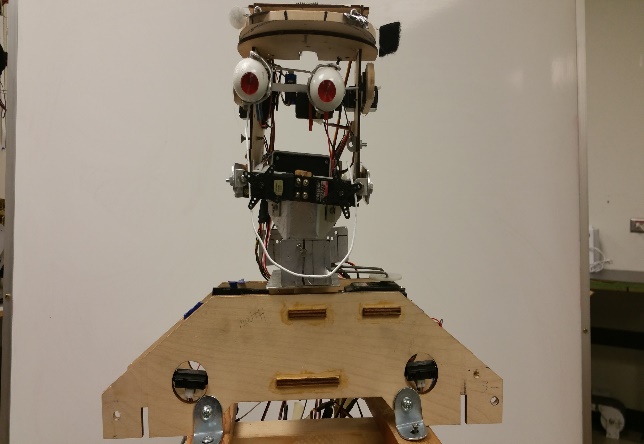
**2. Fuzzy implementation in Einstein Robot using IR detection sensors:**   
  
**2.1 Introduction:**

Fuzzy sets allow elements to be partially in a set unlike crisp sets where elements are either present in a set or not present in a set. In fuzzy set, membership value can range from 0 to 1 where o means element is not present in a set. When membership values are extreme (0 0r 1), it is equivalent to crisp sets. A membership function is a relationship between the values of an element and its degree of membership in a set. A fuzzy inference system is a system that uses fuzzy set theory to map inputs to outputs.

A rule based fuzzy controller with reactive behavior is implemented to perform robotic head movement. Controller provides a layer of robust high-level motor skills.

Einstein’s head is guided by sensor-info, sensor-based or behavior-based method is proposed. This approach takes the way of moving Einstein’s robot head by direct mapping between sensors and motors. Fuzzy logic unlike classical logic is tolerant to imprecision, uncertainty and partial truth. This makes it easier to implement fuzzy logic controller to nonlinear models than other conventional control techniques. Moreover, fuzzy controller offers a probability to mimic expert human knowledge.

Robot is equipped with a set of proximity infrared sensors. They are used real time to detect obstacles by measuring the reflected light. 2 proximity sensors are mounted in the shoulders of the robot faced to the front.



Robotic head motion based on a fuzzy controller

Sensor used is SHARP GP2D12:  
-> Effective range is 10 to 80 cm.

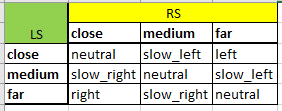
Under distance of 10 cm, the obstacle is considered to be very near for our robot and hence head should be at neutral position.

Fuzzy Control  
- (input membership function, fuzzy logic rules table, output membership function)

Fuzzy sets for sensors/ inputs ->(close, medium, far)

Fuzzy set for motors or motion for output-> (neutral, slow\_left, left, slow\_right, right)

**2.2 Fuzzy table:**



**2.3 Fuzzy rule (if..then)**

Fuzzy rules are a collection of linguistic elements that describe how FIS should make decisions regarding classifying an input or controlling an output.

Membership functions define what we mean by close, medium, far, neutral, slow\_left, slow\_right, left and right. The process of taking an input and processing it through a membership function is called fuzzification.There are two types of fuzzy combination “AND” and “OR”. In our project, we used “fuzzy AND” combination.

**1.** IF (LS IS CLOSE) AND (RS IS CLOSE) THEN NEUTRAL.**2.** IF (LS IS CLOSE) AND (RS IS MEDIUM) THEN SLOW\_LEFT.

**3.** IF (LS IS CLOSE) AND (RS IS FAR) THEN LEFT.

**4.** IF (LS IS MEDIUM) AND (RS IS CLOSE) THEN SLOW\_RIGHT.

**5.** IF (LS IS MEDIUM) AND (RS IS MEDIUM) THEN NEUTRAL.

**6.** IF (LS IS MEDIUM) AND (RS IS FAR) THEN SLOW\_LEFT.

**7.** IF (LS IS FAR) AND (RS IS CLOSE) THEN RIGHT.

**8.** IF (LS IS FAR) AND (RS IS MEDIUM) THEN SLOW\_RIGHT.

**9.** IF (LS IS FAR) AND (RS IS FAR) THEN NEUTRAL.

**2.4 Fuzzification**

During fuzzification, we mapped the inputs from a set of infrared sensors to values from 0 to 1 using a set of input membership functions.   
  
We computed fuzzy “AND” using the following:  
  
The fuzzy "and" is written as:

Undisplayed Graphic

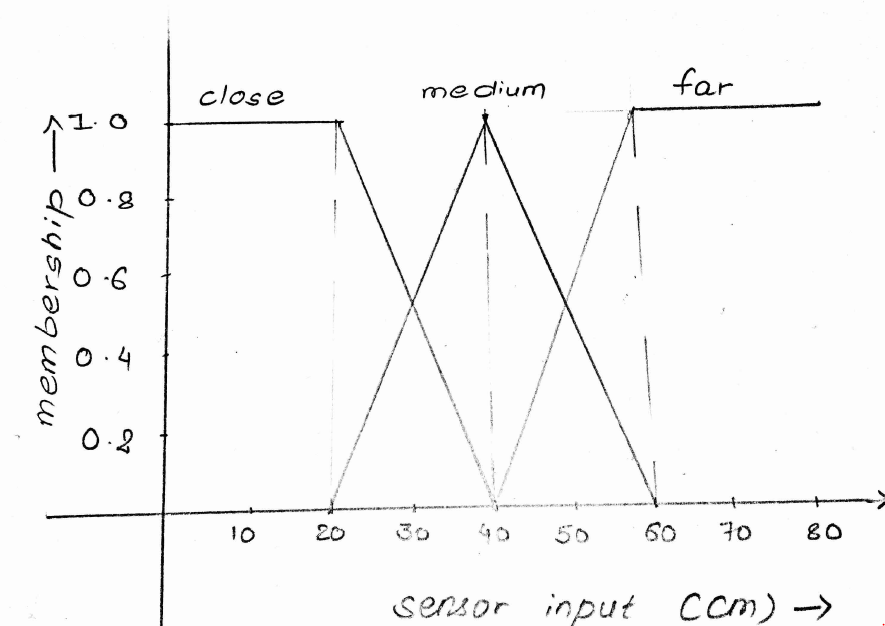
: This technique, named after the inventor of fuzzy set theory simply computes the "and" by taking the minimum of the two (or more) membership values. This is the most common definition of the fuzzy "and".  
  
It has the following properties:

The consequence of fuzzy rule is then computed by combining the fuzzified inputs with fuzzy “and” combination process.

Using triangular membership functions for simplicity. It means sum of the membership functions of any variables at any given point in the domain is equal to one.

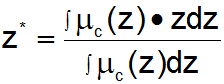
In fuzzy set , each element x is assigned with a degree of membership in , measured by a membership function

Membership function is zero when x does not belong to at all, on when belongs totally to and when belongs partially to the fuzzy set.



The outputs of all fuzzy rules were then combined to obtain one fuzzy output distribution.

**2.5 Defuzzification**Weighted average method for defuzzification: It is only valid for symmetrical output membership functions**.** It is formed by weighting each function in the output by its respective maximum membership value.

****

// Need membership graph for output

From the graph, membership values can be calculated as following:

1.At or below ,

2.From   
We solve two equations to get the membership values

1. Close membership function,

At So equation becomes:  
. So equation i becomes:

and , we get:-  
Or

And,

2. Medium membership function,

At , So equation becomes:  
At , . So equation becomes:

and , we get:-  
Or,

3. At ,

4. At and

We solve two equations to get the membership values

1. Medium membership function,

At So equation becomes:  
, . So equation becomes:

and , we get:-  
  
Or,

And,

2. Far membership function,

At So equation becomes:  
. So equation becomes:

and , we get:-  
  
Or,

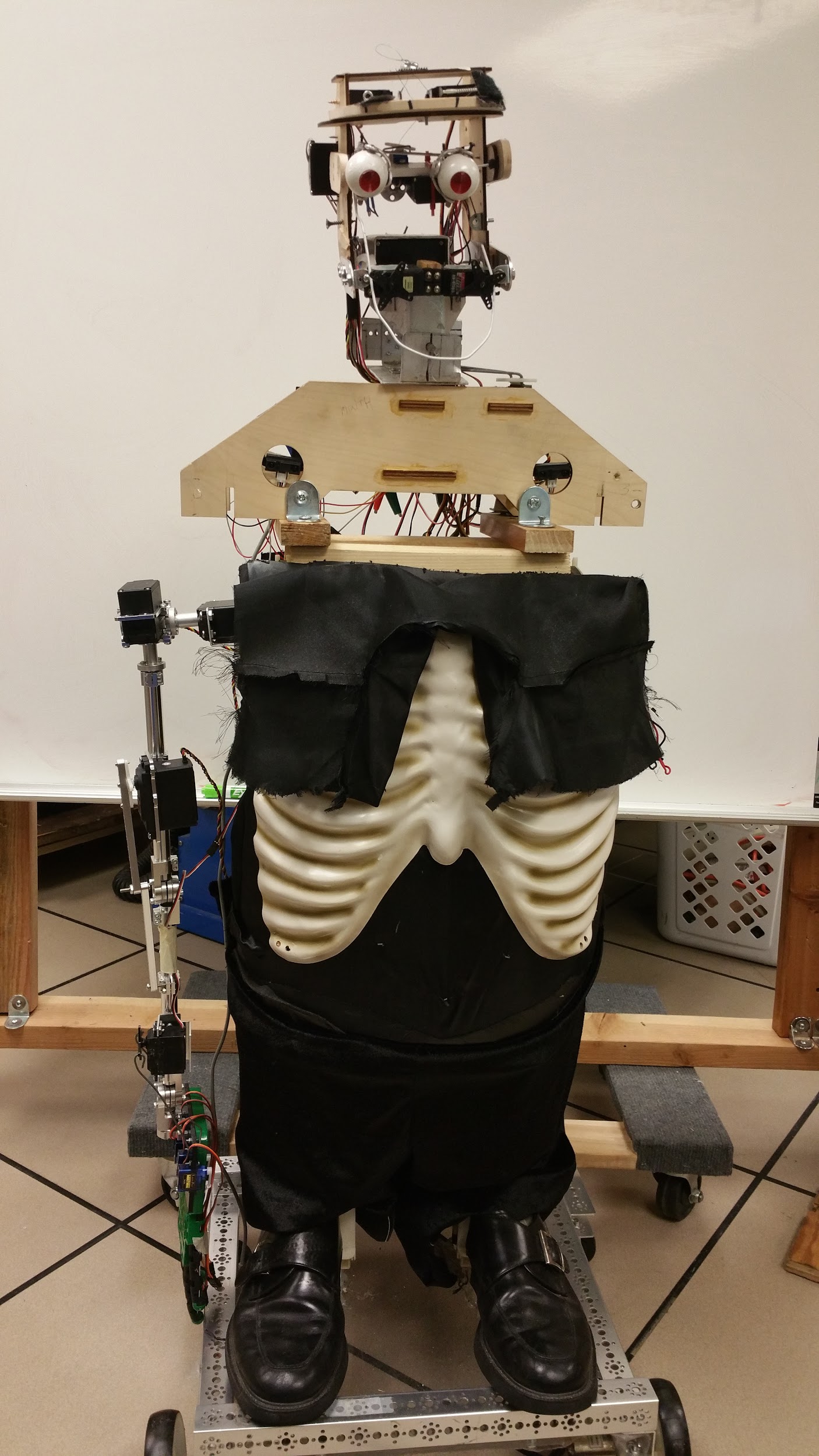
And,

Hence,

*5. At or above,*

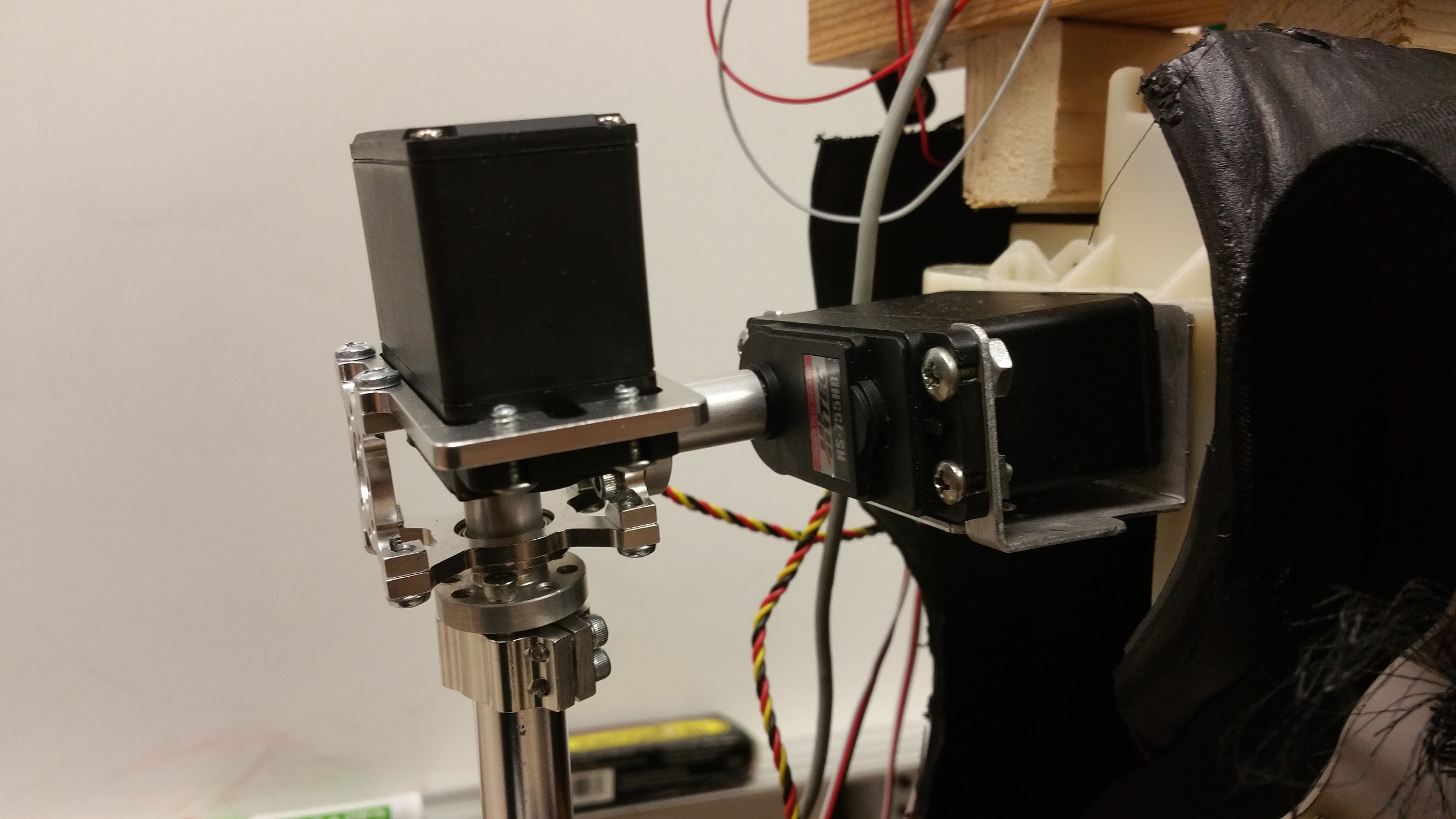
**2.6 Results And Link to fuzzy:**We successfully implemented fuzzy logic in robotic head.  
<https://www.youtube.com/watch?v=Ic8fPw8EWAc>

**3. ARM DESIGN**

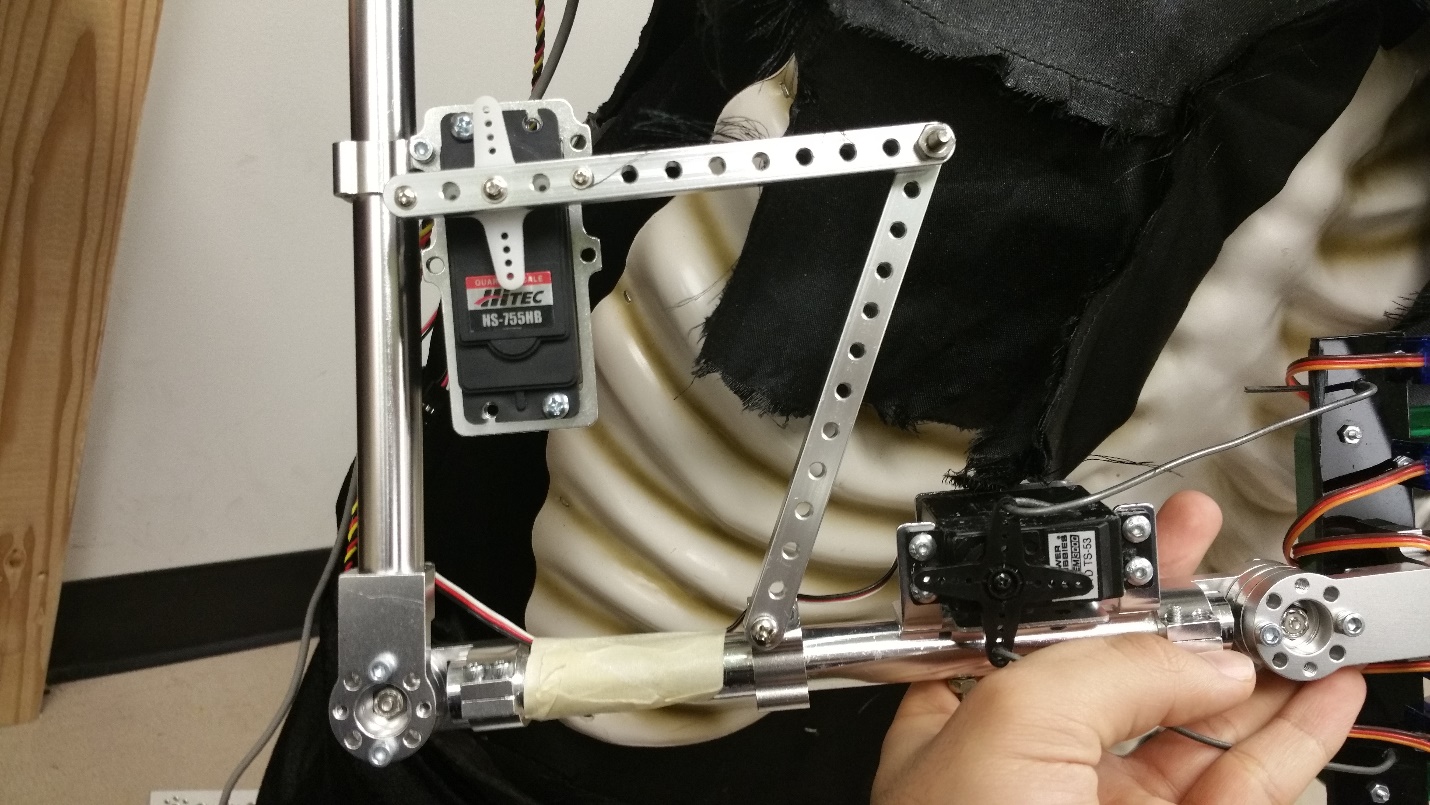
****

**3.1 Pictures:**

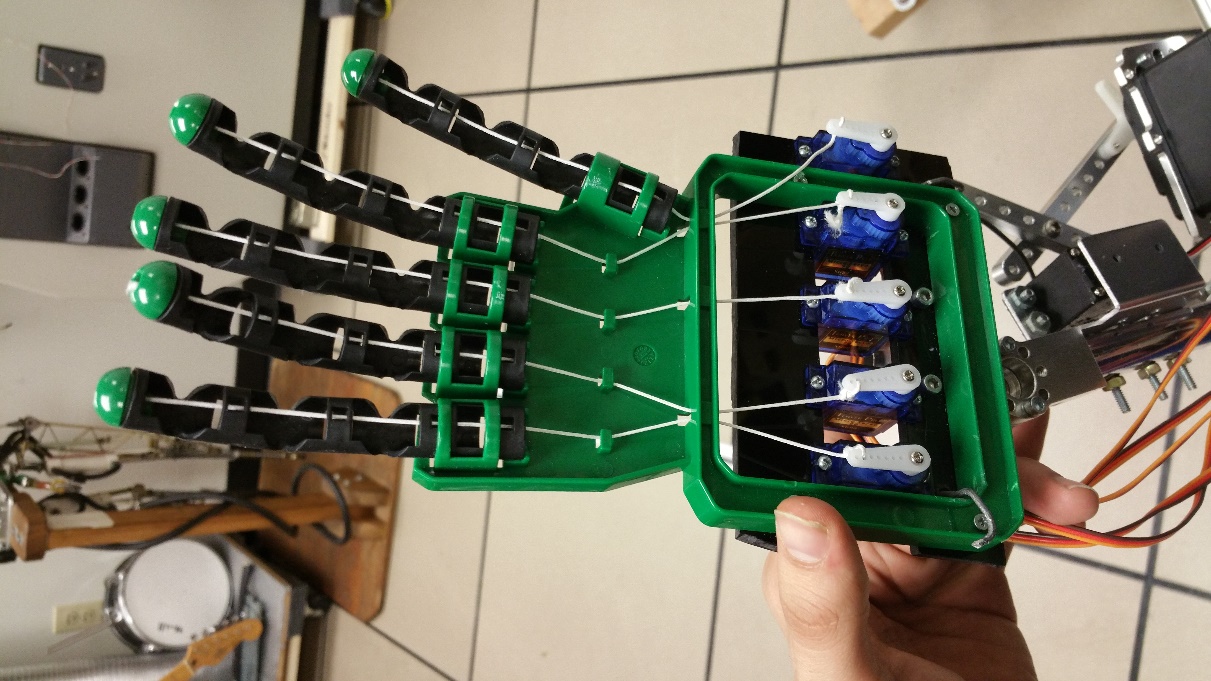
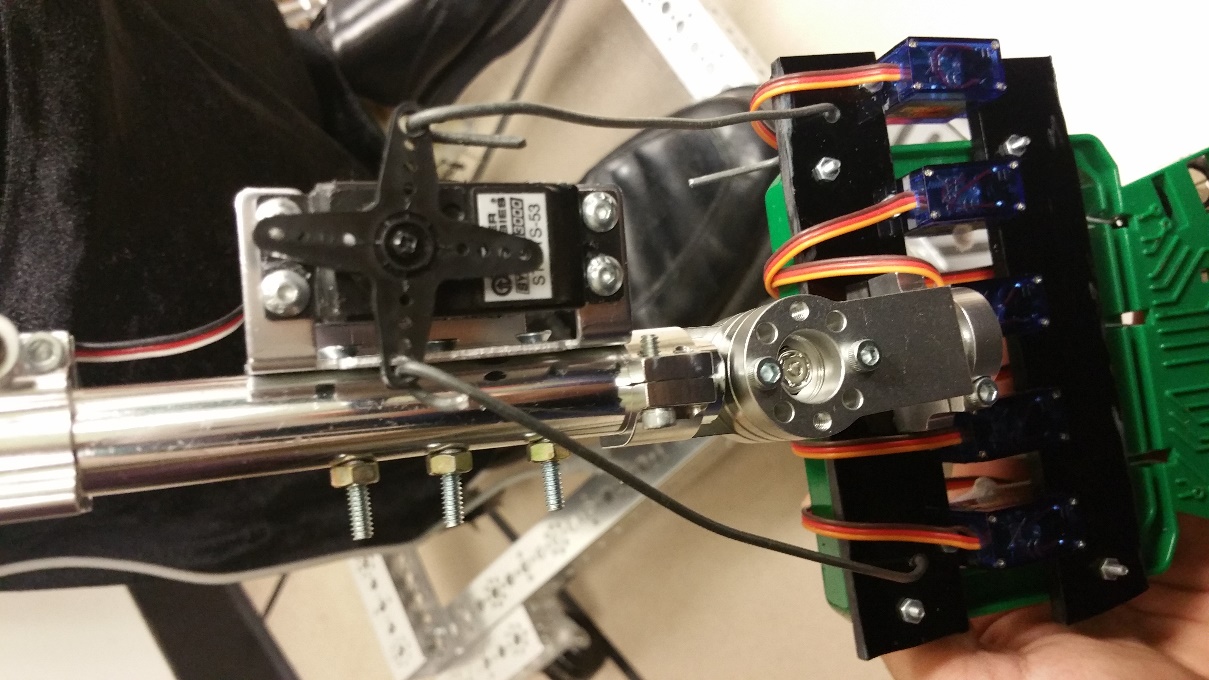
1. Shoulder:



2. Elbow:



3. Wrist and Hand:



**3.2 Hardware and software used:  
  
a.** **Arduino (ATmega 328 microcontroller)**

It is open source hardware and is relatively cheap ($24.95 currently) and has 14 digital input/ output pins with six of them providing pulse width modulation (PWM) outputs. We could simply connect to computer or power it with a AC-to-DC adapter or battery to start.

**b. Large hub shaft servo block (24T spline)**

We chose this part for our arm design as it helped to increase servo’s load bearing by helping to isolate load from servo spline and case.

**c. Servo shaft hub and Aluminum tubes**Servo shaft hub fits with spline and aluminum tubing serves as the arm structure.

**d. Bore 90 degree patterned clamping mount**  
To attach hollow tubes together and to provide more holding power for higher torque.  
  
**e. Dual pinch bolt face tapped clamping hubs**They work with gears, wheels and sprockets and offer more holding power for greater torque calculations.

**f. Swivel hub**As the name suggests, it helps any connected parts to swivel a full 360 degree. It has internal ball bearing for smooth motion.

**g. Large servo plate A**To attach gears and mechanical structures to servo.   
  
**h. Aluminium beams**They help to make unique connection like limiting elbow motion from a full swing.   
  
**i. Bore bottom tapped clamping mount**It has light weight with good strength and good torque holding power.

j**. Arduino Software**We started by testing previous team’s code for Einstein arm. There was power issue. After recognizing and fixing the issue, we tested each servos. Then we created functions for motion in the required direction for each of the servos in the robotic arm.

**3.3 Degree of freedom:**nine

**3.4. Torque calculations:**

Torque is tendency of force to rotate an object about an axis. It is made up of load and acceleration components of which load is constant.

Torque for the servos used from datasheet:

1. HS-755HB (total 2: used in shoulder and elbow)

Weight: 110g

At 6 V operating voltage,

Stall torque: 13.2 kg.cm

Standing torque: 10.5 kgcm

1. STD TS-53 (For wrist- 1)

Weight: 42.5 g

At 4.8 V operating voltage, torque = 3.02 kg.cm

1. SG90 micro-servo (5 for fingers)

Weight: 9 g

Stall torque: 1.8 kg.cm

1. HS-805BB (shoulder servo)  
   Weight: 152 g  
   Stall torque: 24.7 kg.cm at 6V operating voltage

There are 3 types of forces: vertical, horizontal and incline. We did vertical force calculation for our robotic arm design.

Vertical force, F(v) = W = mg

Where F = force of moving direction

m = mass of the load/ servo

g= acceleration due to gravity

Load torque = F X l , where l is the distance from center of rotation to force applied.

Mass of large hub shaft = 43.94 g

Mass of servo shaft hub = 9.9 g

Mass of remaining parts = approx 500g

Torque calculation for shoulder servo for our robot = F. l = Total mass in kg times acceleration due to gravity times length in meters =

((110+110+42.5+9\*5+ 43.94+9.9\*2+500)/1000)x9.8x0.65 = 6.07 kg.m =607 kg.cm(approx.)

Use of a tooth spline handles over 1.08 kg.m of torque without slipping.

**4. Modifications:  
  
1. Servos:**We tested out all servos previously used. The previous teams did not secure servo horns. We

used screws to secure all servos horns so they would not slip out during motion. We tested code and modified for better and desired motion for previously incorporated behaviors.

**2. Hand:**  
We replaced the strings in the hand with the more powerful string.

**3. Pins:**  
We disconnected unused servos from Arduino to make room for the servos in robotic arm. We labelled pins for the arm.

**5. Challenges:**

* We had to learn Arduino as we go so this took longer time than expected to be able to understand the connections from previous group.
* Handling torque to achieve required hand motion.

**6. Future Recommendations:  
  
1.** Use servo controller to be able to control multiple servos.

2. Replace the neck servo for a better neck movement (N10).

3. Bluetooth to send commands to robot.

4. Securing wire

5. Torque handling of arm to have a better functional complete arm

6. Upgrades to the head servos movements like the jaw servos that are unused.

7. Including a raspberry pi 3.

8. Jaw upgrade for mouth instead of using wire hanger.

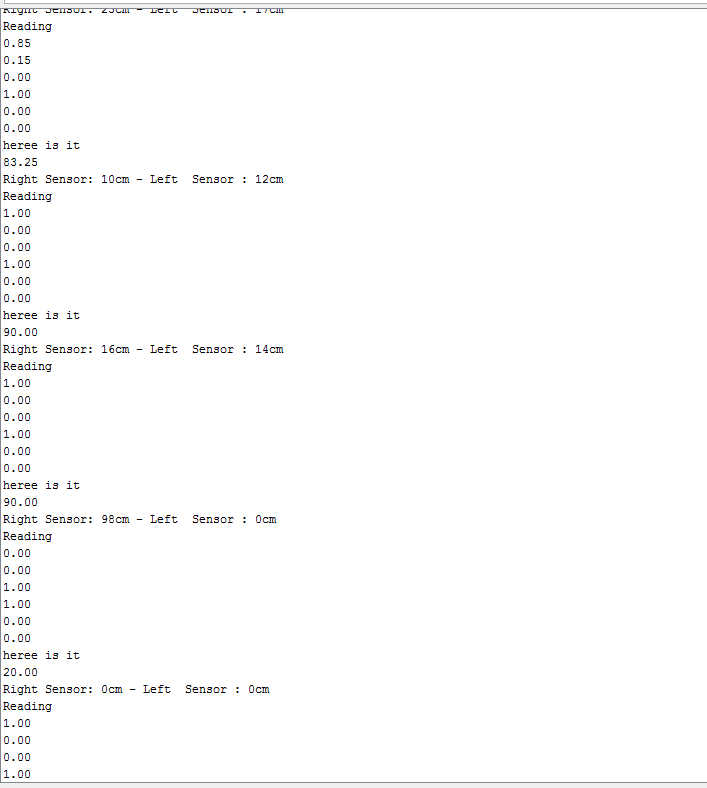
**7. References:**

1. <http://www.dasconference.ro/cd2010/data/papers/A35.pdf>

2. <http://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm>

3. [www.servocity.com](http://www.servocity.com/)

**APPENDIX:**

**A. Results of fuzzy logic (distance inputs and output rotation angle):**  
  
Arduino- Serial monitor observation:  


**B. Arduino Code for fuzzy logic:**

//Srijana Sapkota, Naser Alshami

//Group Homework Fall 2016

//Professor Marek Perkowski

//ECE478/578 Intelligent Robotics

//Effective Range: 10-80 cm

//Code for implementing fuzzy logic

//Our idea is to make neck move towards the obstacle based on infrared sensor distance inputs

// Outputs monitored using Serial Monitor tool in Arduino

#include <Servo.h> // Servo library

//final crisp outputs for our robot

#define NECK\_NEUTRAL 90

#define NECK\_SLOW\_LEFT 45

#define NECK\_LEFT 20

#define NECK\_SLOW\_RIGHT 120

#define NECK\_RIGHT 145

Servo N9;

char GP2D12;

char GP2D12\_b;

char a,b,x,y;

int rs\_val;

int ls\_val;

void setup()

{

Serial.begin(9600); //baud rate is 9600

N9.attach(9); //servo connected to head for our desired motion is named as N9 and is connected to digital pin 9 of arduino.

}

void loop()

{

GP2D12=read\_gp2d12\_range(0);

a=GP2D12/10;

b=GP2D12%10;

rs\_val=a\*10+b;

GP2D12\_b=read\_gp2d12\_range(1);

x=GP2D12\_b/10;

y=GP2D12\_b%10;

ls\_val=x\*10+y;

{

Serial.print("Right Sensor: ");

Serial.print(rs\_val);

Serial.print("cm - ");//

Serial.print("Left Sensor : ");

Serial.print(ls\_val);

Serial.println("cm ");//

}

//fuzzy implementation

float rs\_close, rs\_medium, rs\_far; //fuzzy inputs from right sensor

float ls\_close, ls\_medium, ls\_far; //fuzzy inputs from left sensor

float sum\_all\_mins, sum\_rotation\_times\_min;

float final\_fuzzy\_output;

float min1, min2, min3, min4, min5, min6, min7, min8, min9;

//if else condition will take care of the conditions based on distance inputs

//while ((ls\_val >=10 || ls\_val <=80) && (rs\_val >=10 || rs\_val <=80))

//{

if (rs\_val <= 20)

{

rs\_close = 1;

rs\_medium = 0;

rs\_far = 0;

}

else if (rs\_val > 20 && rs\_val < 40)

{

rs\_close = (-rs\_val/20.0) + 2.0;

rs\_medium = (rs\_val/20.0) - 1.0;

rs\_far = 0;

}

else if (rs\_val == 40)

{

rs\_close = 0;

rs\_medium = 1;

rs\_far = 0;

}

else if (rs\_val > 40 && rs\_val < 60)

{

rs\_close = 0;

rs\_medium = (-rs\_val/20.0) + 3.0;

rs\_far = (rs\_val/20.0) - 2.0;

}

else if (rs\_val >= 60)

{

rs\_close = 0;

rs\_medium = 0;

rs\_far = 1;

}

if (ls\_val <= 20)

{

ls\_close = 1;

ls\_medium = 0;

ls\_far = 0;

}

else if (ls\_val > 20 && ls\_val < 40)

{

ls\_close = (-ls\_val/20.0) + 2.0;

ls\_medium = (ls\_val/20.0) - 1.0;

ls\_far = 0;

}

else if (ls\_val == 40)

{

ls\_close = 0;

ls\_medium = 1;

ls\_far = 0;

}

else if (ls\_val > 40 && ls\_val < 60)

{

ls\_close = 0;

ls\_medium = (-rs\_val/20.0) + 3.0;

ls\_far = (rs\_val/20.0) - 2.0;

}

else if (ls\_val >= 60)

{

ls\_close = 0;

ls\_medium = 0;

ls\_far = 1;

}

//Serial.print(rs\_close, rs\_medium, rs\_far, ls\_close, ls\_medium, ls\_far, F, DEC, DEC, DEC);

Serial.println("Reading");

Serial.println(rs\_close);

Serial.println(rs\_medium);

Serial.println(rs\_far);

Serial.println(ls\_close);

Serial.println(ls\_medium);

Serial.println(ls\_far);

min1 = min(ls\_close, rs\_close);

min2 = min(ls\_close, rs\_medium);

min3 = min(ls\_close, rs\_far);

min4 = min(ls\_medium, rs\_close);

min5 = min(ls\_medium, rs\_medium);

min6 = min(ls\_medium, rs\_far);

min7 = min(ls\_far, rs\_close);

min8 = min(ls\_far, rs\_medium);

min9 = min(ls\_far, rs\_far);

sum\_all\_mins = min1+min2+min3+min4+min5+min6+min7+min8+min9;

if(sum\_all\_mins !=0)

{

sum\_rotation\_times\_min = (min1 \*NECK\_NEUTRAL) + (min2 \* NECK\_SLOW\_LEFT) + (min3 \* NECK\_LEFT) + (min4 \* NECK\_SLOW\_RIGHT) + (min5 \* NECK\_NEUTRAL) + (min6 \* NECK\_SLOW\_LEFT) + (min7 \* NECK\_RIGHT) + (min8 \* NECK\_SLOW\_RIGHT) + (min9 \* NECK\_NEUTRAL);

final\_fuzzy\_output = sum\_rotation\_times\_min/sum\_all\_mins;

Serial.println("heree is it");

Serial.println(final\_fuzzy\_output);

}

N9.write(final\_fuzzy\_output);

delay (1000);

//}

}

//this code reads in sensor inputs

//obtained from internet

float read\_gp2d12\_range(byte pin)

{

int tmp;

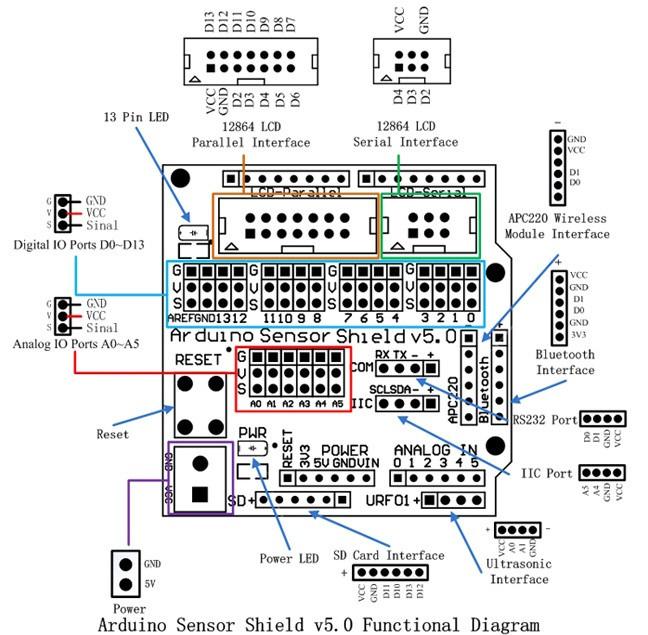
tmp = analogRead(pin);

if (tmp < 3)return -1;

return (6787.0 /((float)tmp - 3.0)) - 4.0;

}

**C. Arduino Sensor Shield v5.0 Board Layout**



**D. Bill of Materials:**

|  |  |  |  |
| --- | --- | --- | --- |
| Quantity | Description | Unit | Amount |
| 1 | (637118) Large Hub Shaft ServoBlock™ (24T Spline) | $28.99 | $28.99 |
| 1 | (525122) Servo Shaft Hub (0.500")  Servo Spline: 24 Tooth Spline | $9.99 | $9.99 |
| 4 | (545368) 5/8" Bore 90° Patterned Clamping Mount | $8.99 | $35.96 |
| 1 | (635270) 5/8" Aluminum Tubing  Length: 8.00" | $3.79 | $3.79 |
| 3 | (545344) Dual Pinch Bolt, Face Tapped Clamping Hubs, 0.770" Pattern | $5.99 | $17.97 |
| 2 | (545364) Swivel Hub | $6.99 | $13.98 |
| 1 | (575116) Large Servo Plate A | $5.99 | $5.99 |
| 1 | (585412) (13 hole) Aluminum Beams (2 pack) | $3.99 | $3.99 |
| 1 | (585548) 5/8" Bore Bottom Tapped Clamping Mount | $5.99 | $5.99 |
| 1 | Arduino UNO Rev 3 microcontroller | $17.00 | $17.00 |
| 1 | SunFounder DC 9V/650mA Power Plug Adapter for Arduino UNO R3 Mega 2560 1280 (6 Feet) | $6.99 | $6.99 |
| Subtotal | | | $157.63 |
| 1st Shipping & Handling | | | $6.99 |
| 2nd Shipping & Handling | | | $6.99 |
| 3rd Shipping & Handling | | | $6.99 |
| Total Shipping Cost | | | $20.97 |
| Grand Total | | | $171.61 |

**E. Arduino code for Einstein:**

// Einstein Head and Right Arm Arduino Code ECE478 - Fall16

// Naser Alshami and Srijana Sapkota

// Using arduino Uno R3 with Sensor Shield

// This is a build up and several enhancements on the previous group Arduino source code.

#include <Servo.h> // servo library

// Servo variables using wire labeling

Servo E7, E2; // E7 = Eyes Up/Down, E2 = Eyes Left/Right

Servo F4, F5, F6, F11, F12; // Fingers F4 = thumb

Servo N9, N10; // N10 = Neck Side Tilt, N9 = Rotate Head ("No" motion)

Servo S0, S1; // S0 = Shoulder Up/Down, S1 = Shoulder rotation

Servo J8, J3; // J8 = Elbow Joint, J3 = Wrist Joint

Servo U13;

int pos=0;

void setup(){

// assigning variable to i/o pins

S0.attach(0); // move shoulder up/down

S1.attach(1); // rotate shoulder

E2.attach(2); // moves eyes left/right

J3.attach(3); // wrist joint

F4.attach(4); // Thumb finger ,finger 1

F5.attach(5); // Finger 2

F6.attach(6); // Finger 3

E7.attach(7); // moves eyes up/down

J8.attach(8); // elbow joint

N9.attach(9); // rotate neck left/right "No"

N10.attach(10); // tilt neck side to side

F11.attach(11); // Finger 4

F12.attach(12); // Finger 5

N9.write(100);

N10.write(20);

pinMode(A2, OUTPUT); // left eye led

pinMode(A3, OUTPUT); // right eye led

digitalWrite(A2, HIGH); // Turn left eye LED on

digitalWrite(A3, HIGH); // Turn right eye LED on

//initial\_state\_arm();

//initial\_state\_hand();

}

void loop(){

// test();

// Arm movements functions

//elbow();

//wrist();

//wave();

//shoulderUp();

//shoulderRotate();

//counting();

//nofingers();

//all\_wave();

//dance();

//myturn();

// Head movements functions

// eyesON();

// eyesOFF();

// eyesBlink();

// eyeWink();

// browRight();

// shiftyEyes();

// eyesLeft();

// UpDownEyes();

// mouthOpenCloseSlow();

// mouthOpenCloseFast();

// motionNo();

// NeckTilt();

}

void test(){

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

J3.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

J3.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}

void myturn() {

S1.write(120);

// J8.write(30);

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

S0.write(pos); // tell servo to go to position in variable 'pos'

J8.write(pos/2);

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

S1.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

S1.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

S0.write(pos); // tell servo to go to position in variable 'pos'

J8.write(pos/2);

delay(15); // waits 15ms for the servo to reach the position

}

}

void sweep\_eyes() {

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

E2.write(pos); // tell servo to go to position in variable 'pos'

E7.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

E2.write(pos); // tell servo to go to position in variable 'pos'

E7.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}

void initial\_state\_arm() {

S0.write(160); // Shoulder down

S1.write(150);

J8.write(60);

}

void initial\_state\_hand() {

J3.write(100);

F4.write(180);

F5.write(180);

F6.write(180);

F11.write(180);

F12.write(180);

}

void all\_wave() {

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

J3.write(pos); // tell servo to go to position in variable 'pos'

S1.write(pos);

E2.write(pos); // tell servo to go to position in variable 'pos'

E7.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

J3.write(pos); // tell servo to go to position in variable 'pos'

S1.write(pos);

E2.write(pos); // tell servo to go to position in variable 'pos'

E7.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}

void elbow() {

J8.write(30); // Tight

delay(1000);

J8.write(60); // Middle

delay(1000);

J8.write(90); // Straight

delay(1000);

}

void wave() {

J3.write(10);

delay(500);

J3.write(160);

delay(500);

}

void nofingers() {

F4.write(0);

F5.write(180);

F6.write(0);

F11.write(0);

F12.write(0);

wave();

}

void wrist() {

J3.write(10); // Wrist rightwards

delay(1000);

J3.write(40);

delay(1000);

J3.write(70);

delay(1000);

J3.write(100);

delay(1000);

J3.write(130);

delay(1000);

J3.write(160); // Wrist leftwards

delay(1000);

}

void shoulderUp(){

S0.write(120);

J8.write(30);

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

S0.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

S0.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

// S0.write(30);

// delay(5000);

// S0.write(60);

// delay(5000);

// S0.write(80); // Shoulder straight

// delay(3000);

// S0.write(120);

// delay(5000);

// S0.write(150);

// delay(5000);

// S0.write(160); // Shoulder down

// delay(4000);

// S0.write(100);

// delay(4000);

// S0.write(40); // Shoulder up

// delay(4000);

}

void shoulderRotate() {

S0.write(160); // Shoulder down

delay(100);

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

S1.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

S1.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

// S1.write(175); // Near body

// delay(5000);

// S1.write(150);

// delay(5000);

// S1.write(120);

// delay(5000);

// S1.write(90);

// delay(5000);

// S1.write(60);

// delay(5000);

// S1.write(30);

// delay(5000);

// S1.write(50); // Away from body

// delay(5000);

}

void counting(){

for (int x = 0; x < 3; x++)

{

F4.write(180);

delay(1000);

F5.write(180);

delay(1000);

F6.write(180);

delay(1000);

F11.write(180);

delay(1000);

F12.write(180);

delay(1000);

F4.write(0);

F5.write(0);

F6.write(0);

F11.write(0);

F12.write(0);

delay(1000);

}

}

void dance(){

S0.write(100); // Shoulder straight

J8.write(10);

F4.write(180);

J3.write(180);

F5.write(180);

F6.write(0);

F11.write(0);

F12.write(180);

for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

S1.write(pos\*2); // tell servo to go to position in variable 'pos'

J8.write(pos);

N10.write(pos);

J3.write(pos);

E2.write(pos\*2); // tell servo to go to position in variable 'pos'

E7.write(pos\*2); // tell servo to go to position in variable 'pos'

F5.write(pos\*2);

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 90; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

S1.write(pos\*2); // tell servo to go to position in variable 'pos'

J8.write(pos);

N10.write(pos);

J3.write(pos);

E2.write(pos\*2); // tell servo to go to position in variable 'pos'

E7.write(pos\*2); // tell servo to go to position in variable 'pos'

F5.write(pos\*2);

delay(15); // waits 15ms for the servo to reach the position

}

}

/\*

void motionNo(){

neckNeutral();

delay(500);

neckTurnRight();

delay(500);

neckTurnLeft();

delay(500);

neckTurnRight();

delay(500);

neckNeutral();

delay(500);

}

void NeckTilt(){

neckTiltLeft();

delay(500);

// neckNeutral();

// delay(500);

neckTiltRight();

delay(500);

neckTiltLeft();

delay(500);

neckTiltRight();

//delay(500);

neckNeutral();

delay(1500);

}

\*/

void mouthOpenCloseFast(){

U13.write(100);

delay(200);

U13.write(150);

delay(200);

}

void mouthOpenCloseSlow(){

U13.write(100);

delay(500);

U13.write(150);

delay(500);

}

void mouthOpen(){

U13.write(100);

delay(200);

}

void mouthClose(){

U13.write(150);

delay(200);

}

/\*

void shiftyEyes(){

// eyesNeutral();

//delay(500);

eyesRight();

delay(500);

// eyesNeutral();

// delay(500);

eyesLeft();

delay(500);

}

void UpDownEyes(){

eyesUp();

delay(500);

eyesDown();

delay(500);

}

void neckNeutral(){

N9.write(90);

delay(500);

N10.write(60);

}

void neckTurnRight(){

N9.write(145);

}

void neckTurnLeft(){

N9.write(20);

}

void neckTiltLeft(){

N10.write(100);

}

void neckTiltRight(){

N10.write(30);

}

void eyesNeutral(){

E2.write(50);

delay(500);

E7.write(165);

}

void eyesLeft(){

E2.write(110); // E2.write(80);

}

void eyesRight(){

E2.write(10);

}

void eyesUp(){

E7.write(180);

}

void eyesDown(){

E7.write(15);

}

void browRight(){

B4.write(200);

delay(1000);

B4.write(10);

delay(1000);

}

\*/

void eyesBlink(){

eyesON();

delay(4000);

eyesOFF();

delay(500);

}

void eyesON(){

digitalWrite(A2,HIGH ); // Turn on left eye LED

digitalWrite(A3,HIGH); // Turn on right eye LED

}

void eyesOFF(){

digitalWrite(A2, LOW); // Turn on left eye LED

digitalWrite(A3,LOW); // Turn on right eye LED

}

void eyeWink(){

digitalWrite(A3,LOW); // Turn off right eye LED

delay(1000);

digitalWrite(A3,HIGH); // Turn on right eye LED

delay(1000);

}