A

SYNOPSIS

On PROJECT

ANIMATRONICS HANDS USING MECHATRONICS TECHNOLOGY

Submitted by

SIDDHANT PRATAP SINGH (1743120043)

AKANKHSA TIWARI (1743120003)

JAI HIND (1743120017)

UTKARSH PANDEY (1743120049)

KM ADAB ISTIAQ (1743120019)

RAJAN JOSHI (1743120033)

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

ELECTRICAL ENGINEERING

Under the guidance of

Dr. ABHISHEK MISHRA

Submitted to

Mr. SUSHIL KUMAR

B.N. COLLEGE OF ENGINEERING AND TECHNOLOGY, LUCKNOW

DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, UTTAR PRADESH

LUCKNOW

2020 – 2021

ACKNOWLEDGEMENT

I take this opportunity to express my profound sense of gratitude and respect to all those who helped me in this project.

Firstly, I would like to protract my gratitude to **Dr. Ashutosh Dwivedi (Director)** & Dr. Abhishek Mishra **(Additional Director and H.O.D. E.E. Department) of BNCET**, and all the faculty members of department for their valuable guidance in this project.

This project has seen its present day with the inspiration and guidance provided by **Mr. Sushil Kumar** (**Asst. Prof. E.E. Department)** the project in charge under whose guidance I carried out the present work.

At the end, I would like to say thanks to all the members of library & practical labs for providing the valuable information related to this project.

Above all I thank Almighty God for his manifold benevolences to carry out the study successfully.

SIDDHANT PRATAP SINGH (1743120043)
AKANKHSA TIWARI (1743120003)
JAI HIND (1743120017)
UTKARSH PANDEY (1743120049)
KM ADAB ISTIAQ (1743120019)
RAJAN JOSHI (1743120033)

CERTIFICATE

The undersigned certify that they have read and recommended to the Department of Electrical Engineering for acceptance, a project report entitled **"ANIMATRONICS HANDS USING MECHATRONICS TECHNOLOGY"** submitted by

SIDDHANT PRATAP SINGH (1743120043)
AKANKHSA TIWARI (1743120003)
JAI HIND (1743120017)
UTKARSH PANDEY (1743120049)
KM ADAB ISTIAQ (1743120019)
RAJAN JOSHI (1743120033)

of B.Tech. Final year, Electrical Engineering. In partial fulfillment for the degree of Bachelor of Technology in Electrical Engineering.

Mr. Sushil Kumar
Project In-charge
Department of Electrical Engineering
B.N. College of Engineering & Technology,
NH-24, Sitapur Road, B.K.T., Lucknow

Dr. Abhishek Mishra
Additional Director and Head of
Electrical Department
B.N. College of Engineering & Technology,
NH-24, Sitapur Road, B.K.T., Lucknow

Date of Submission:

ABSTRACT

Mechatronics is an upcoming technology that facilitates the machine to mimic the action of human and animals and produce more natural movement. For some time, we have been interested in making some sort of robot based on the Arduino platform. This project is the first phase of this longer-term desired effort. Anything is possible with the mighty power of Arduino. It's compact, it's straight forward, and makes embedding electronics into the world-at-large fun and easy.
Animatronics is the use of mechatronics to create machines which seem animate rather than robotic. Animatronic figures are most often powered by pneumatics (compressed air), and, in special instances, hydraulics (pressurized oil), or by electrical means. The figures are precisely customized with the exact dimensions and proportions of living creatures. Motion actuators are often used to imitate “muscle” movements, such as limbs to create realistic motions. Also, the figure is covered with body shells and flexible skins made of hard and soft plastic materials. Then, the figure can be finished by adding details like colours, hair and feathers and other components to make the figure more realistic.
We wanted to make a shadow robot from our curiosity. As the whole body of the robot would have been of much cost. Approximating the kinematics of the human hand was our top priority when developing this animatronic hand. Each joint of this hand has a movement range again the same as or very close to that of a human hand, including the thumb and even the flex of the palm for the little finger.

Keyword: Mechatronics, Arduino, Animatronics, Flex sensors, Power IC, Programming of the Arduino.

TABLE OF CONTENT

FIGURE OF CONTENT

INTRODUCTION

This project mainly designs regarding the construction of this Arduino based animatronic hand. Although more complicated and precise (more expensive) versions of this concept have been developed, this is a fun project with many potential applications. Interactive robot control of this level, I think, has many uses in industrial manufacturing, medical research, and anything you want to be able to do with precision that is unsafe to touch. The basic components of the hand and glove are the hand itself, the servos, the Arduino, the glove, and the flex sensors. The glove is mounted with flex sensors: variable resistors that change their value when bent. They're attached to one side of a voltage divider with resistors of a constant value on the other side. The Arduino reads the voltage change when the sensors are bent, and triggers the servos to move a proportional amount. The servo pulls strings that act as tendons, allowing the fingers to move.

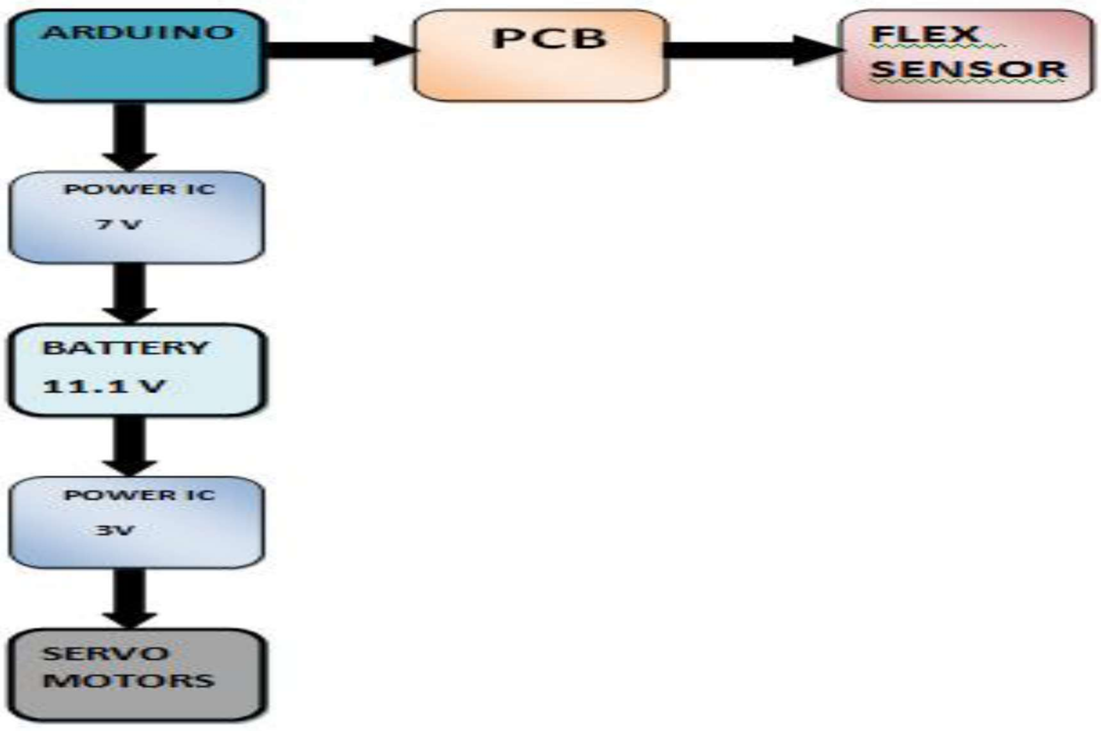


Figure 1: System Block Diagram

There are certain applications where a human hand is plays a significant role in a task, but pose life threatening risks when humans are subject to exposure. The objective of this work is to develop a robotic hand controlled by a human, in order to obviate human intervention in certain deleterious applications. The Animatronic Hand is a mechanical hand that is controlled with the help of an Arduino board. For instance, the device can be put to use in nuclear industries where hazardous substances that may pose a threat to his/her life, are handled. Also, the device can be employed in remote medical assistance wherein doctors can perform surgeries from a distant location, obviating the need of physical presence. The Prosthetic Animatronic Hand can also be helpful to physically disabled patients who have limited locomotory capabilities, thereby reducing the dependency on another person. According to a survey conducted in the United States, it is found that approximately 1 in 50 people or 5,596,000 people suffer from paralysis. Consider a patient who wants to operate a light or a fan switch, but cannot do so due to his disability. The robotic hand helps to augment the abilities of such patient's hand and thereby helps in operating the electrical switch. Hence, the motive of the robotic hand is to make the lives of individuals across various walks of life including paralytic patients, simpler and easier. Figure 2 shows an overview of the module.

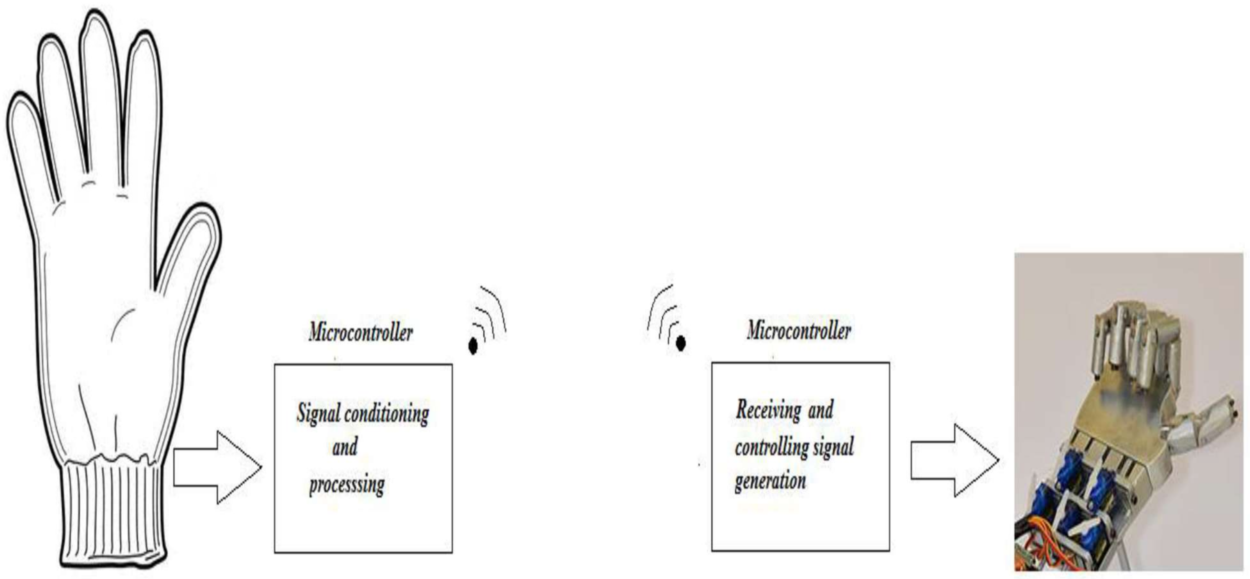

Microcontroller
Signal conditioning and processing
Microcontroller
Receiving and controlling signal generation

Figure 2: Overview of The Prosthetic Animatronic Hand

1. DESIGN

i. Working Principal of Arduino Hardware

The Arduino microcontroller is an easy to use yet powerful single board computer that has gained considerable traction in the hobby and professional market. The Demilune board features an Atmel ATmega328 microcontroller operating at 5 V with 2 KB of RAM, 32 KB of flash memory for storing
programs and 1 KB of EEPROM for storing Parameters. The clock speed is 16 MHz, which translates to about executing about 300,000 lines of C source code per second. The board has 14 digital I/O pins and 6 Analog input pins. There is a USB connector for talking to the host computer and a DC power jack for connecting an external 6- 20 V power source, for example a 11.1 V battery, when running a program while not connected to the host computer. Headers are provided for interfacing to the I/O pins using 22 g solid wire or header connectors.
The Arduino programming language is a simplified version of C/C++. If you know C, programming the Arduino will be familiar. If you do not know C, no need to worry as only a few commands are needed to perform useful functions. An important feature of the Arduino is that we can create a control program on the host PC, download it to the Arduino and it will run automatically. Remove the USB cable connection to the PC, and the program will still run from the top each time you push the reset button. Remove the battery and put the Arduino board in a closet for six months. When you reconnect the battery, the last program you stored will run. This means that you connect the board to the host PC to develop and debug your program, but once that is done, you no longer need the PC to run the program.
The power of the Arduino is not its ability to crunch code, but rather its ability to interact with the outside world through its input-output (I/O) pins. The Arduino has 14 digital I/O pins labelled 0 to 13 that can be used to turn motors and lights on and off and read the state of switches. Each digital pin can sink or source about 40 mA of current. This is more than adequate for interfacing to most devices, but does mean that interface is needed to control devices other than simple LED's. In other words, you cannot run a motor directly using the current available from an Arduino pin, but rather must have the

pin drive an interface circuit that in turn drives the motor. A later section of this document shows how to interface to a small motor.

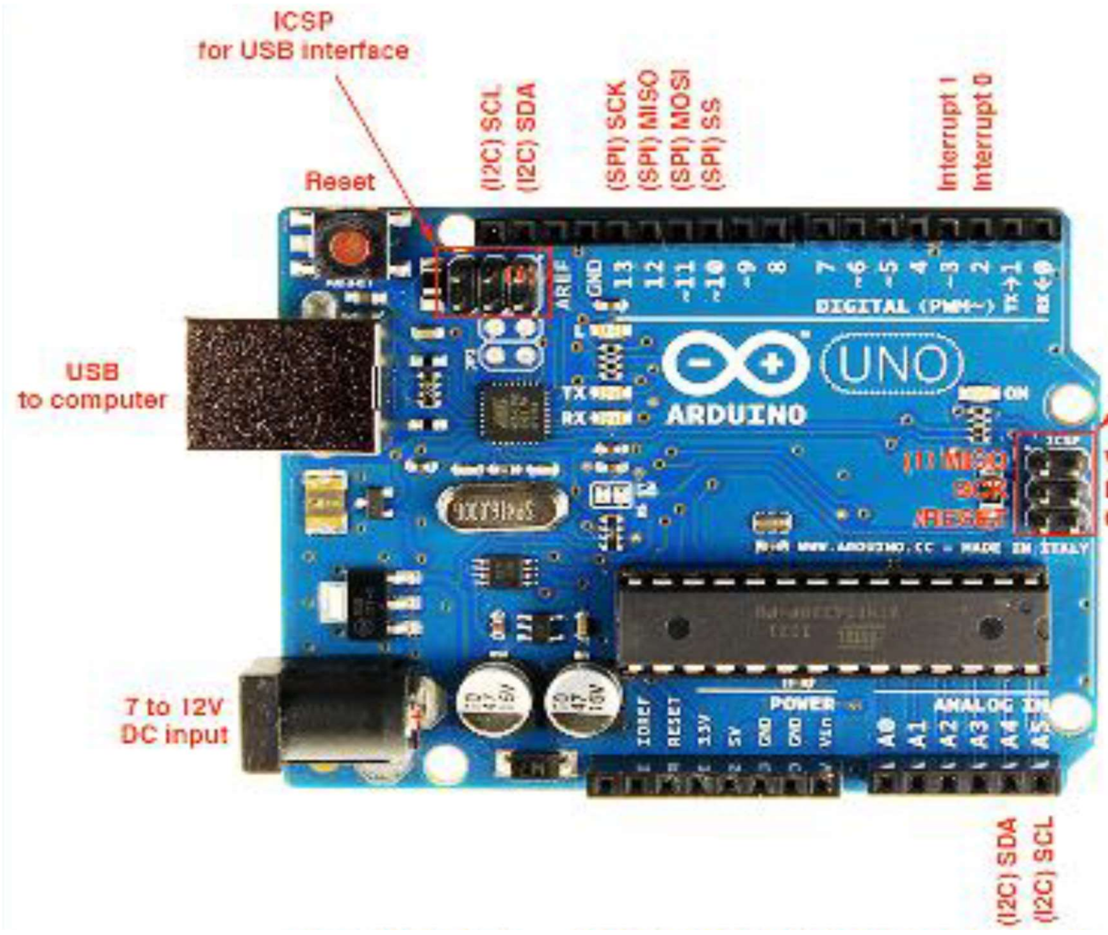


Figure 3: Arduino Hardware

To interact with the outside world, the program sets digital pins to a high or low value using C code instructions, which corresponds to +5 V or 0 V at the pin. The pin is connected to external interface electronics and then to the device being switched on and off. The sequence of events is shown in this figure.

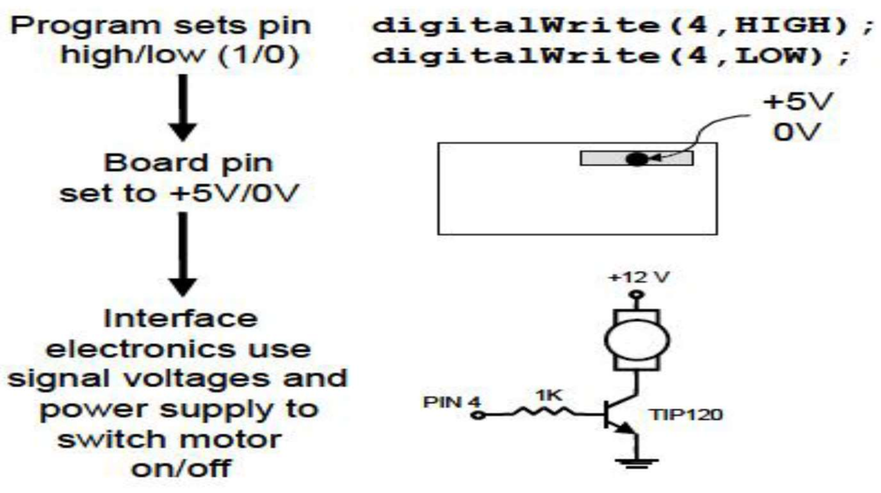


Figure 4: Configuration of Arduino Hardware

To determine the state of switches and other sensors, the Arduino is able to read the voltage value applied to its pins as a binary number. The interface circuitry translates the sensor signal into a 0 or +5 V signal applied to the digital I/O pin. Through a program command, the Arduino interrogates the state of the pin. If the pin is at 0 V, the program will read it as a 0 or LOW. If it is at +5 V, the program will read it as a 1 or HIGH. If more than +5 V is applied, you may blow out your board, so be careful. The sequence of events to read a pin is shown in this figure:

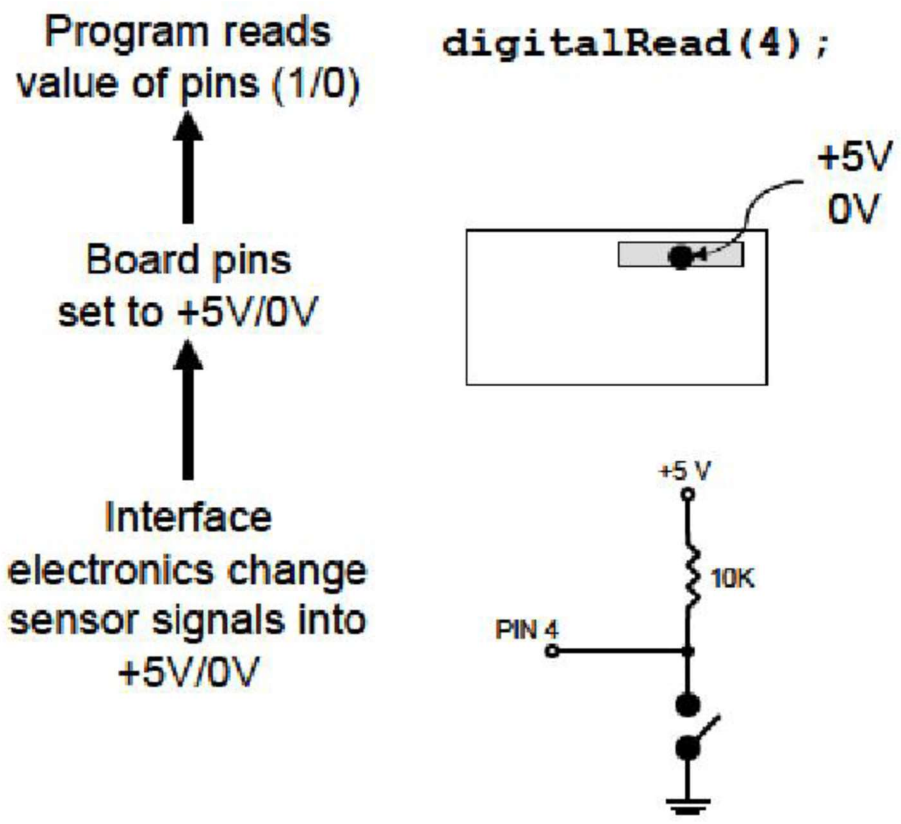

Program reads value of pins (1/0)
digitalRead(4);
+5V
0V
Board pins set to +5V/0V
+5 V
10K
PIN 4
Interface electronics change sensor signals into +5V/0V

Figure 5: Configuration of Arduino Hardware

Interacting with the world has two sides. First, the designer must create electronic interface circuits that allow motors and other devices to be controlled by a low (1-10 mA) current signal that switches between 0 and 5 V, and other circuits that convert sensor readings into a switched 0 or 5 V signal. Second, the designer must write a program using the set of Arduino commands that set and read the I/O pins. When reading inputs, pins must have either 0 or 5V applied. If a pin is left open or "floating", it will read random voltages and cause erratic results. This is why switches always have a 10K pull up resistor connected when interfacing to an Arduino pin. The reason to avoid using pins 0 and 1 is because those pins are used for the serial communications between the Arduino and the host computer. The Arduino also has six Analog input pins for reading continuous voltages in the range of 0 to 5 V from sensors such as potentiometers.

ii. Working Principal of Servo Motor

Unlike dc motors, with servo motors you can position the motor shaft at a specific position (angle) using control signal. The motor shaft will hold at this position as long as the control signal not changed. This is very useful for controlling robot arms, unmanned airplanes control surface or any object that you want it to move at certain angle and stay at its new position. Servo motors may be classified according to size or torque that it can withstand into mini, standard and giant servos. Usually mini and standard size servo motors can be powered by Arduino directly with no need to external power supply or driver. The servo has 3 wires:

Black wire: GND (ground)! RED wire: +5v ! Coloured wire: control signal

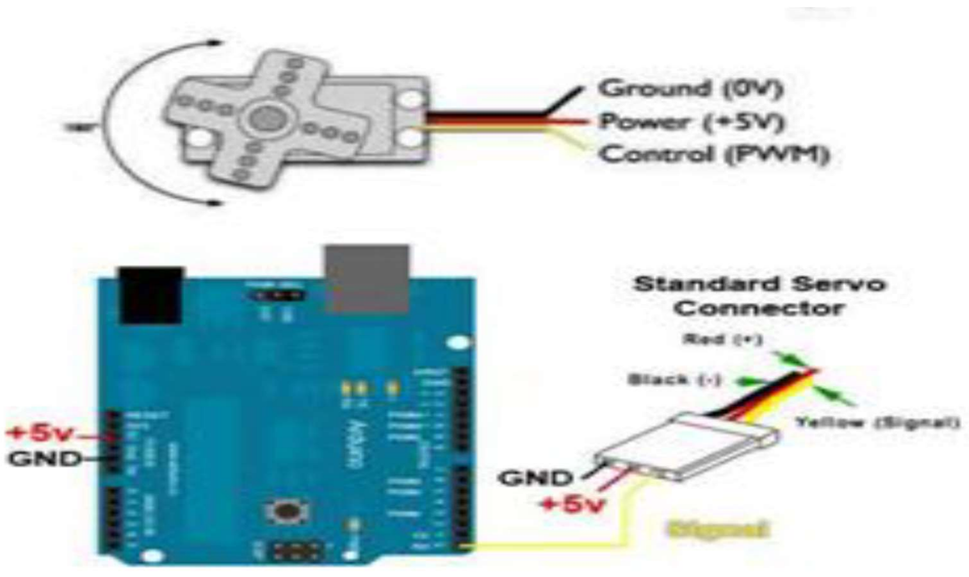


Figure 6: Servo Motor Connection

The third pin accept the control signal which is a pulse-width modulation (PWM) signal. It can be easily produced by all micro- controllers and Arduino board. This accepts the signal from your controller that tells it what angle to turn to. The control signal is fairly simple compared to that of a stepper motor. It is just a pulse of varying lengths. The length of the pulse corresponds to the angle the motor turns to. The pulse width sent to servo ranges as follows: Minimum: 1 millisecond ---> Corresponds to 0 rotation angle. Maximum: 2 millisecond ---> Corresponds to 180 rotation angles. Any length of pulse in between will rotate the servo shaft to its corresponding angle. For example: 1.5ms pulse corresponds to rotation angle of 90 degree. This is will explained in figure below:

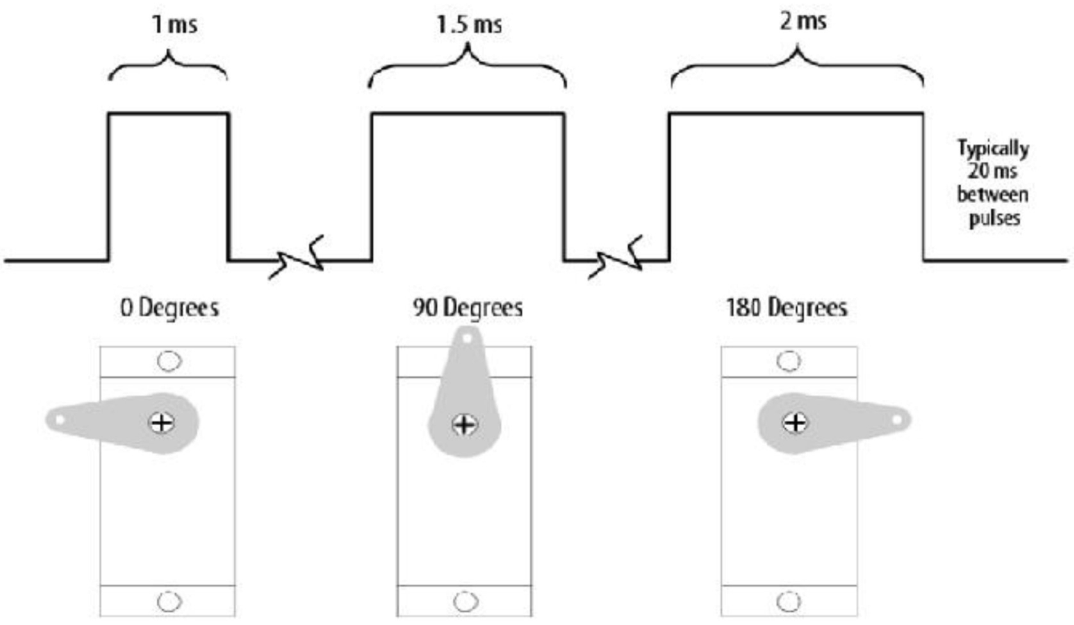


Figure 7: Rotating Configuration of Servo motor

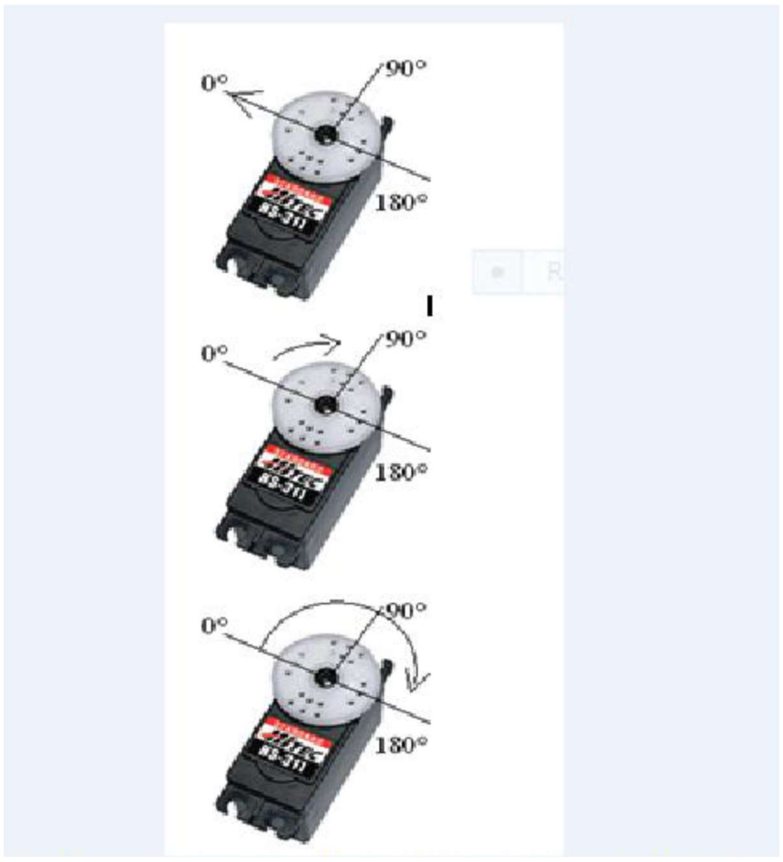


Figure 8: Rotational Configuration of Servo

iii. Working Principal of Flex Sensor

Flex sensors are sensors that change in resistance depending how much the sensor is bend. Sensors convert the change in bend to electrical resistance - the more the sensor bend, the higher the resistance value. Using the Flex Sensor is very easy. There are couple of different manufacturers in the market. Datasheet instructs you to use operational amplifier (OpAmps). That may be useful if you plan to use flex sensor as stand-alone device (without any microcontroller). Because We are using Arduino, we skipped all OpAmps and made a very simple circuit with only one additional resistor.

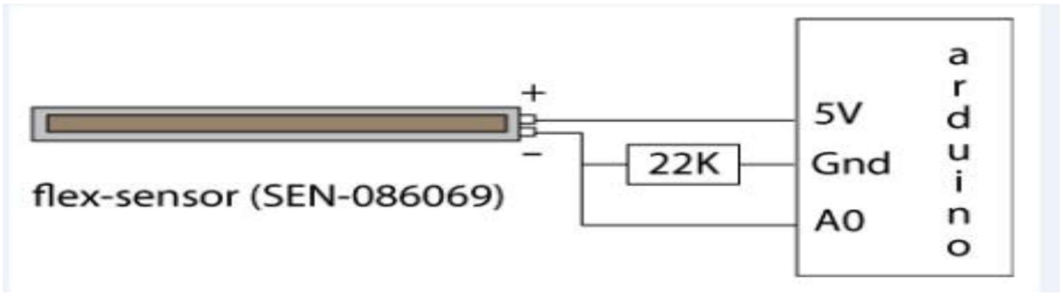


Figure 9: Simple Flex Sensor Circuit

Varying the value of the resistor will results different readings. With 22k Ohm resistor I will get values between 400-600. This works fine for us. In our code we assumed that all values above 400 mean that the sensor is bend. All values above 600 mean that sensor is nor bend. Note that Flex sensor give reliable readings ONLY if you bend it on the specific direction (usually towards on the text side of the sensor).

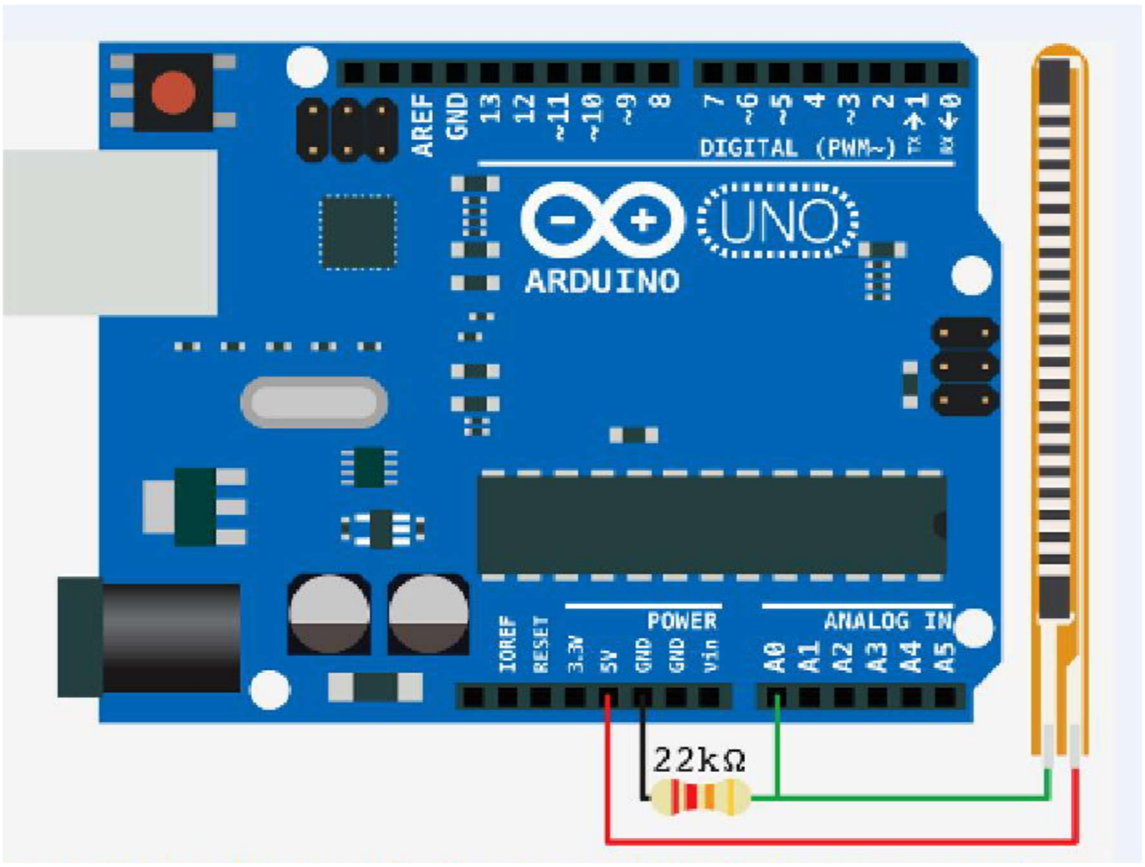


Figure 10: Flex Sensor Connection with Arduino

iv. Motor Control Using Arduino

a) Operating One Servo with Arduino:

Standard servo motor control using Arduino is extremely easy. This is because the Arduino software comes with a sample servo sketch and servo library that will get you up and running quickly:

1. Connect the black wire from the servo to the Gnd pin on the Arduino
2. Connect the red wire from the servo to the +5V pin on the Arduino
3. Connect the third wire (usually orange or yellow) from the servo to a digital pin on the Arduino.

# Important Notes:

1- It is not a good idea to connect a motor of any kind directly to the Arduino because it usually requires more power than the board can provide.

2- In our example, the servo is being used to demonstrate code and is not encountering any resistance. Note that you should use a standard or small size. if you are uncertain, check the servo's no-load current rating (it should usually be under 150mA).

3-You may need an external source of 5 or 6 volts when connecting multiple servos. Four AA cells work well if you want to use battery power. Remember that you must connect the ground of the external power source to Arduino ground.

b) Operating Two Servo with Arduino:

The Arduino can control two servos with the same ease as one. All it takes is creating a second instance (copy) of the Servo object, giving it a unique name. In this project we had to use six servos, five of which has to connect directly with Arduino at the same process of connecting one servo with Arduino. When wiring the solderless breadboard, be especially careful not to mix positive and negative leads to the servo. Reversing the power will permanently damage it. In order for everything to function properly, the ground connections for the Arduino and the servo battery supply must be connected together. This is shown in both the schematic and pictorial circuit views. Make sure to also properly orient the connectors for the servos when you plug them into the board. Servo power leads are color-coded, but the colours aren't universal. Here two servos are being connected with the Arduino in the figure given below: -

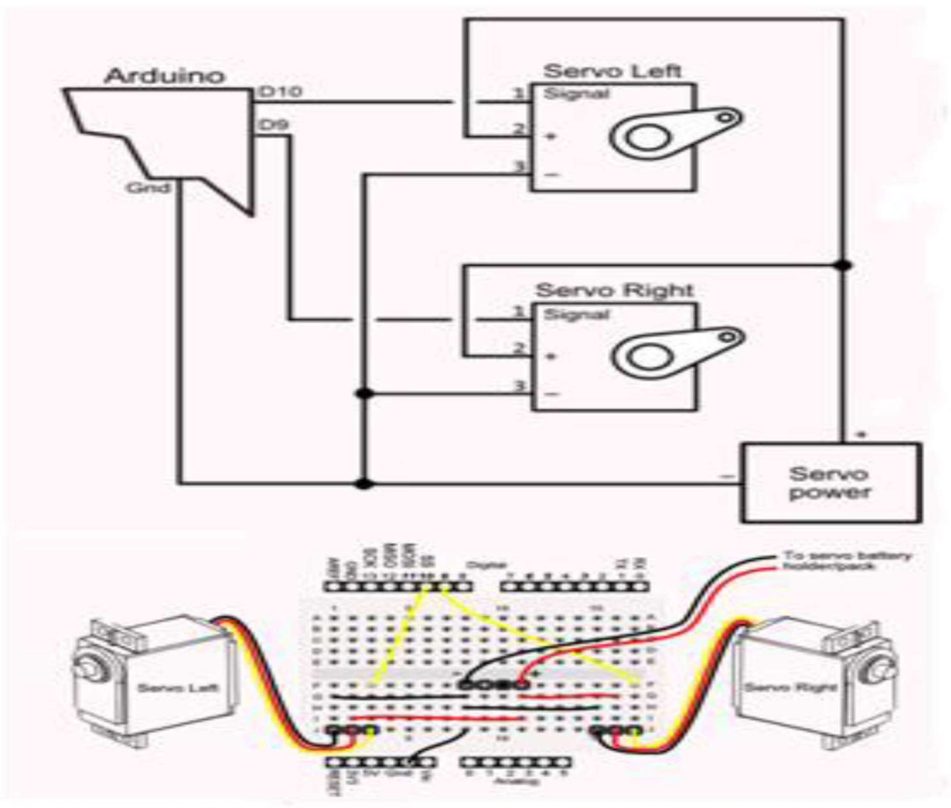


Figure 11: Operating Two Motors Using Arduino

c) System Algorithm

❖ Arduino Programming Coding Style

The Arduino runs a simplified version of the C programming language, with some extensions for accessing the hardware. All Arduino instructions are one line. The board can hold a program hundreds of lines long and has space for about 1,000 two-byte variables. The Arduino executes programs at about 300,000 source code lines per sec. Programs are created in the Arduino development environment and then downloaded to the Arduino board. Code must be entered in the proper syntax which means using valid command names and a valid grammar for each code line. The compiler will catch and flag syntax errors. before download. Sometimes the error message can be cryptic and you have to do a bit of hunting because the actual error occurred before what was flagged. Style refers to our own particular style for creating code and includes layout, conventions for using case, headers, and use of comments. All code must follow correct syntax, but there are many different styles we can use. Here are some suggestions:

* Starting every program with a comment header that has the program name and perhaps a brief description of what the program does.
* Using indentation to line things up. Function name and braces are in column one. Mark major sections or functions with a comment header line or two.

* Having just the right number of comments, not too few and not too many. Assume the reader knows the programming language so have the comment be instructive. Here is an example of an instructive comment **digitalWrite(4,HIGH) // turn on motor** and here is a useless comment **digitalWrite(4,HIGH) // set pin 4 HIGH** we need not comment every line. In fact, commenting every line is generally bad practice.
* Adding the comments when you create the code. If you tell yourself, "Oh, I'll add the comments when the code is finished", you will never do it.

❖ The Processing Code

The processing code of our project is given below:-

```
 #include<Servo.h>

 int flexSensorPin1 = A0;

int flexSensorPin2 = A0;
int flexSensorPin3 = A0;
int flexSensorPin4 = A0;
int flexSensorPin5 = A0;
int flexSensorPin6 = A0;
Servo
servo1,servo2,servo3,servo4,ser
vo5,servo6;
void setup(){
Serial.begin(9600);
servo1.attach(9);
servo2.attach(8);
servo3.attach(7);
servo4.attach(6);
servo5.attach(5);
servo6.attach(4);
pinMode(flexSensorPin1,INPUT);
pinMode(flexSensorPin2,INPUT);
pinMode(flexSensorPin3,INPUT);
pinMode(flexSensorPin4,INPUT);
pinMode(flexSensorPin5,INPUT);
pinMode(flexSensorPin6,INPUT);
}
         int pos=0;
void loop(){
int
fsr1=analogRead(flexSensorPin1)
;
//myServo.write(x);
//Serial.println(fsr);
int s1=map(fsr1, 605, 696, 180,
0);
Serial.println(s1);
if(s1>0 ||s1<180)
servo1.write((s1));
delay(100);
int
```

```
fsr2=analogRead(flexSensorPin2)
;
//myServo.write(x);
//Serial.println(fsr);
int s2=map(fsr2, 605, 696, 180,
0);
Serial.println(s2);
if(s2>0 ||s2<180)
servo2.write((s2));
delay(100);
int
fsr3=analogRead(flexSensorPin3);
        //myServo.write(x);

//Serial.println(fsr);
int s3=map(fsr3, 605, 696, 180,
0);
Serial.println(s3);
if(s3>0 ||s3<180)
servo3.write((s3));
delay(100);
int
fsr4=analogRead(flexSensorPin4)
;
//myServo.write(x);
//Serial.println(fsr);
int s4=map(fsr4, 605, 696, 180,
0);
Serial.println(s4);
if(s4>0 ||s4<180)
servo4.write((s4));
delay(100);
int
fsr5=analogRead(flexSensorPin5)
;
//myServo.write(x);
//Serial.println(fsr);
int s5=map(fsr5, 605, 696, 180,
0);
Serial.println(s5);
if(s5>0 ||s5<180)
servo5.write((s5));
delay(100);
int
fsr6=analogRead(flexSensorPin6)
;
//myServo.write(x);
//Serial.println(fsr);
int s6=map(fsr6, 605, 696, 180,
0);
Serial.println(s6);
```

```
if(s6>0 ||s6<180)
servo6.write((s6));
delay(100);
```

❖ The Coding Structure Analysis

In the program above, the very first thing that we did in the setup function is to begin serial communications, at 9600 bits of data per second, between our Arduino and our computer with the line: **Serial.begin(9600);** Next, initialize digital pin 8, the pin that will read the output from your button, as an input: **pinMode(A8,INPUT);** Now let’s move into the main loop of our code.

Main Loop

```
------------« Begin Code »------------
...
..
void loop()
{
//Move To 0 Degrees
myservo.write(10);
delay(5000);
//Move To 45 Degrees
myservo.write(45);
delay(5000);
//Move To 90 Degrees
myservo.write(90);
delay(5000);
//Move To 135 Degrees
myservo.write(135);
delay(5000);
//Move To 180 Degrees
myservo.write(170);
delay(5000);
//Move To 135 Degrees
myservo.write(135);
delay(5000);
//Move To 90 Degrees
myservo.write(90);
delay(5000);
//Move To 45 Degrees
myservo.write(45);
delay(5000);
}
------------« End Code »------------
```

2. SPECIFICATION

Hardware Details

Table 1: Cost Estimation

Sr. No.	Components	Specification	Quantity (Unit)	Price (in INR)
01	ARDUINO	MEGA 2560	1	999
02	SERVO MOTOR	SG90 (2.5K)	3	469
03	SERVO MOTOR	MIG996R (10K)	2	649
04	SERVO MOTOR	SMS4315 (14K TORQUE)	1	1050
05	BREAD BOARD	SOLDER LESS	1	170
06	JUMPER WIRE	MALE, FEMALE, MALE-FEMALE	3	420
07	FLEX SENSOR	4.5”	5	7500
08	BATTERY	11.1V	1	897
09	BATTERY CHARGER	N/A	1	1500
10	CHARGER CONNECTING CABLE	N/A	1	200
11	POWER IC	N/A	2	120
12	PCB	N/A	2	70
13	RESISTORS	10K, 100K	5	20
14	POT	N/A	2	69
15	STRING	N/A	5	200
16	NEEDLE & THREAD	N/A	1	100
17	SUPER GLUE	N/A	1	400
18	A POWER DRILL	N/A	1	3000
19	SPRING	N/A	10	100
20	RIGHT HAND GLOVE	N/A	1	200

21	WOOD	N/A	2	50
22	PLASTIC HUIS PIPES	N/A	4	200
Total Price				18383

3. IMPLEMENTATION AND CONSTRUCTION

Hardware Procedure

i. Set up the Sensor Circuit

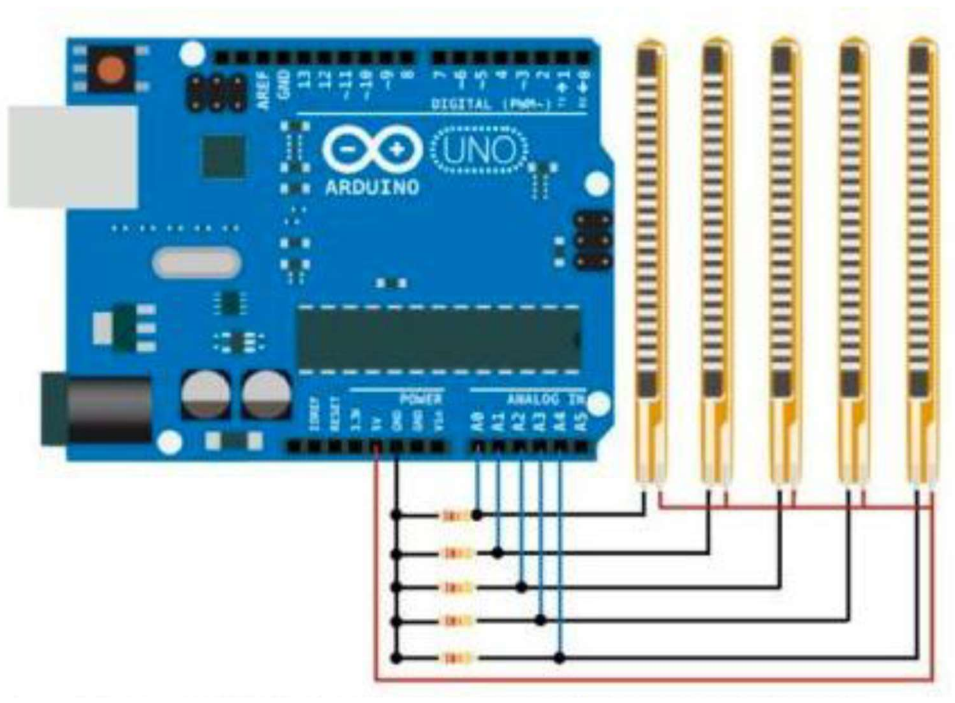


Figure 12: Construction of Sensor Circuit

The flex sensors require a circuit in order for them to be compatible with Arduino. It's a voltage divider: the flex sensors are variable resistors, and when paired with resistors of a static value, change in resistance (in this case bending the sensor) can be sensed through the change in voltage between the resistors. This can be measured by the Arduino through its Analog inputs. The schematic is attached (red is positive voltage, black is negative, and blue goes to the Arduino). The resistors in the photo are 22K. I color-coded the wires we used in the same way as the schematic, so we can see more easily. The main GND wire, which is connected to all the individual GND wires from the sensors, gets plugged into the Arduino's GND. The +5V from the Arduino goes to the main positive voltage wire, and each blue wire gets plugged into a separate Analog input pin.

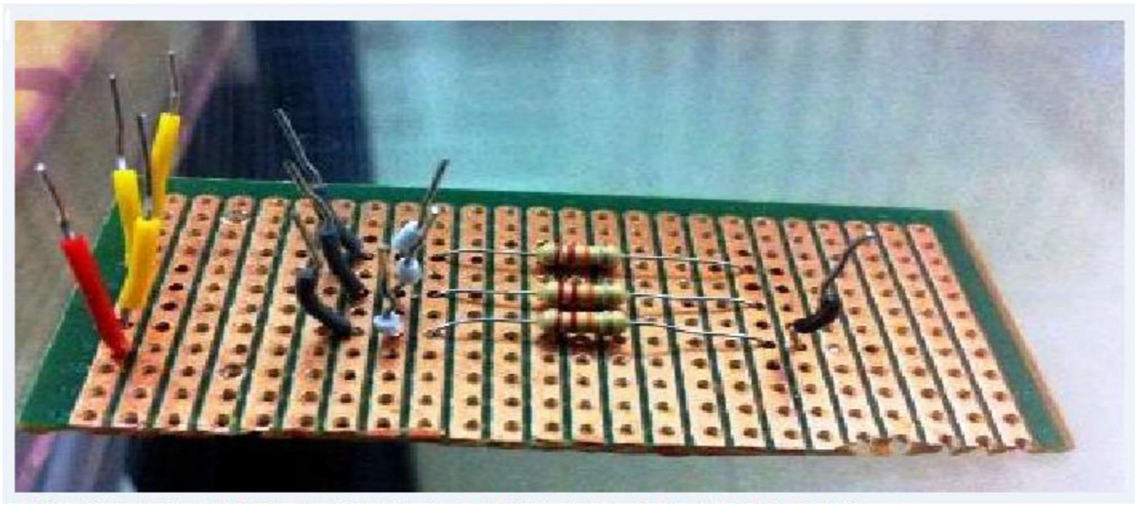


Figure 13: Soldiered the Sensor circuit on a PCB

Then we soldered the circuit onto a small PCB. One that could be easily mounted onto the glove.

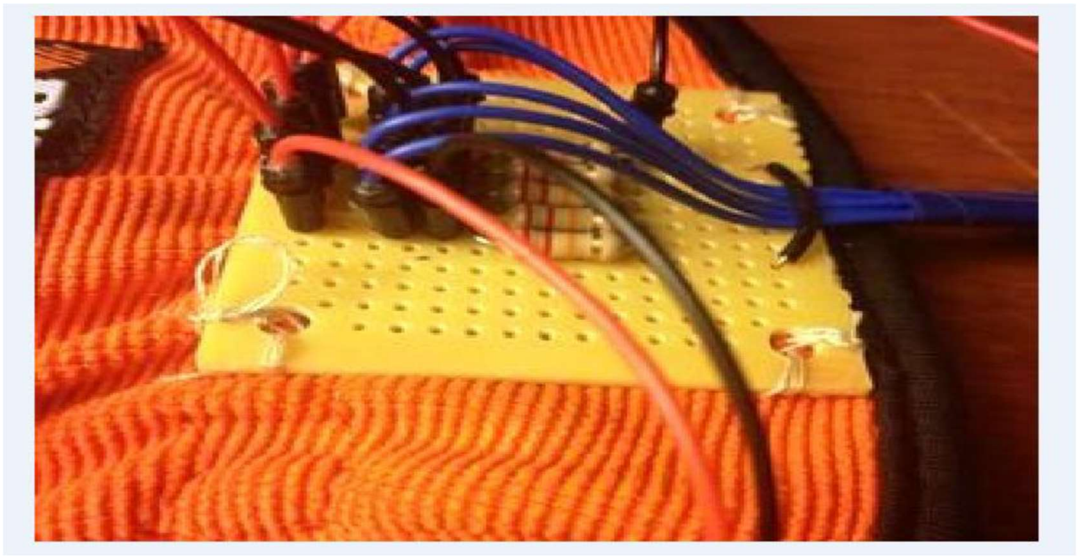
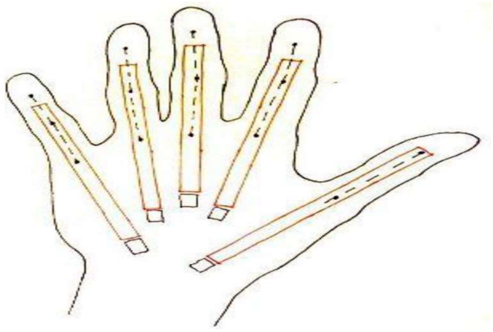


Figure 14: PCB Mounting on The Gloves

ii. Flex Sensor Mounting



Now it's time to mount the sensors and their circuit onto the glove itself. First, we drilled a tiny hole in the plastic of the sensors (at the top, once the resistive material has ended). Be sure not to hit the resistive material! Then, put on the glove and pull it tightly to your hand. On each finger, with a pencil or pen, make small lines over the tops of each joint/knuckle. This will tell you where to sew the sensors. Sew each sensor tip to the area of each finger just above where each of your fingernails would be (use

the hole you just drilled). Then, for each sensor, make loose loops around them with thread at both joints in each finger. Once each sensor is in place and slides under the loops of thread nicely. Then We sewed the PCB onto the wrist part of the glove tightly.
REMEMBER: for each step in this process, be sure We're not sewing the glove itself closed. That's quite a hassle.

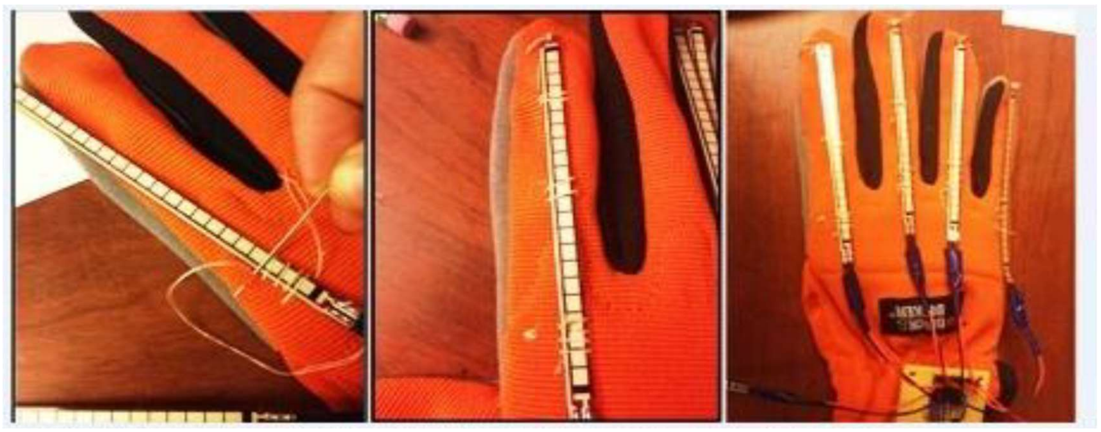


Figure 15: Flex Sensor Mounted on the Glove

iii. Hand and Servo Bed Construction

➢ FINGERS ASSEMBLY

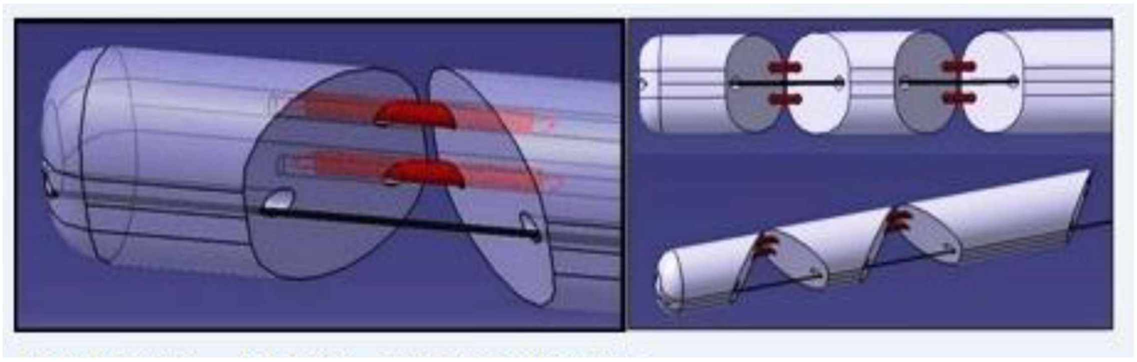


Figure 16: Finger Construction

We used plastic hollow pipes to construct the finger parts. When assembling the fingers, we made sure that the parts were oriented correctly before gluing. Also, we made make sure to re-drill the holes on the finger parts so the 3mm screws will act as hinge pins without causing friction. Then we connected the part using a string and screws. Then we kept the screws in with a dab of hot glue on the outside of the fingers.

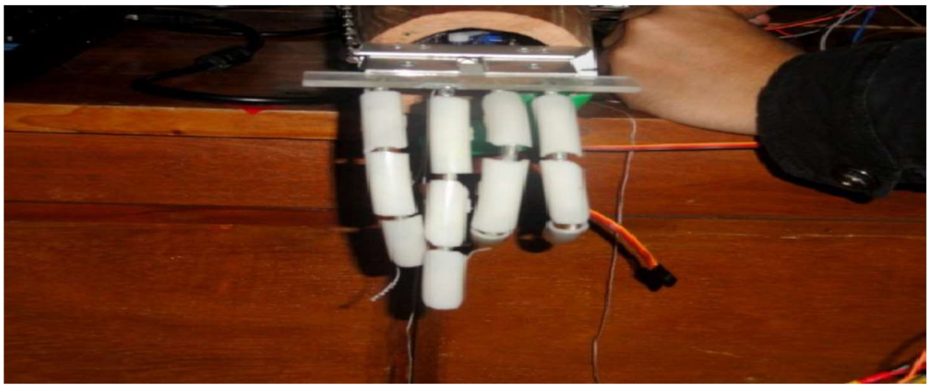


Figure 17: Fingers Assembly

➢ Servo Bed Construction and Servo Mounting

We have used a wooden piece to make the servo bed. Here in the bed we made 5 servo motors to be held perfectly within the bed. It’s just kind of a bed for the servos. Later we have to put the strings to connect the servos with each other. Here is the construction figure shown below: -

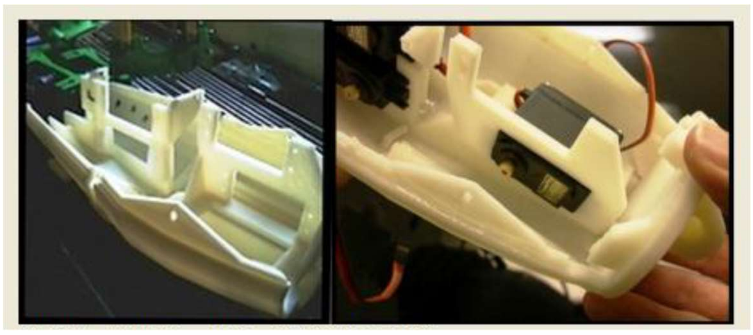


Figure 18: Servo Bed Construction

Here is the picture of our wooden aervo bed shown below: -

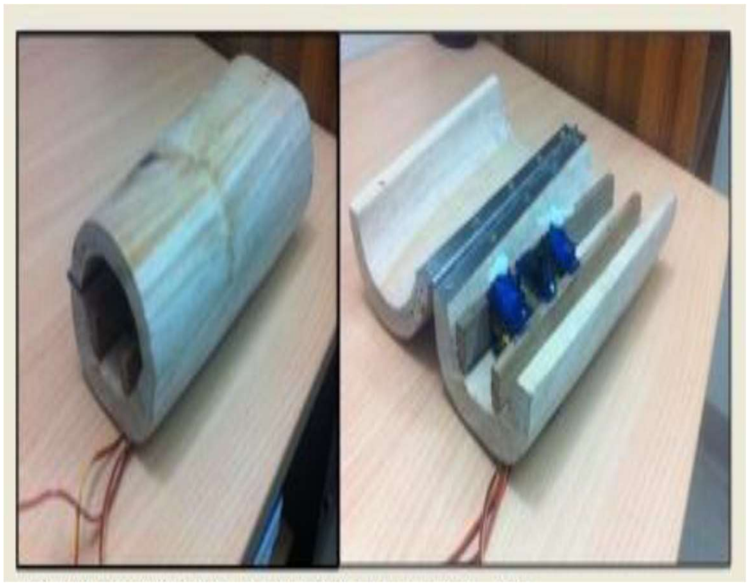


Figure 19: Wooden Construction Servo BED

➢ ADDING THE STRINGS

Adding the strings is by far the hardest and most tedious part of this project. It's simple in concept, but difficult to actually execute. Threading the fingers takes patience: remember that. The one difference between my installation of the strings and I used hot glue. To me, hot glue is more adjustable when calibrating each finger because it can be easily melted and re-hardened. We connected the servo motor in a way we could move our fingers with the exact comfort and flexibility. For that we calibrated the servo motors to connect the strings with the exact process
To calibrate each servo ring so it flexes and relaxes its finger when we wanted it to based on the input, first we plugged in our Arduino and servo battery and run the program. Then we put the glove on and flex the finger that corresponds to the servo we’re working on. We adjusted the servo ring so one hole is closest possible to the fingers and pulled the "relax" string of that finger as tightly as we can without bending the finger. We put it through the closest hole of the ring and glue it in place. Then, straightened my finger and pulled and secured the other string into the other hole. Then we repeated this process with each finger. It's important to make each string taut.

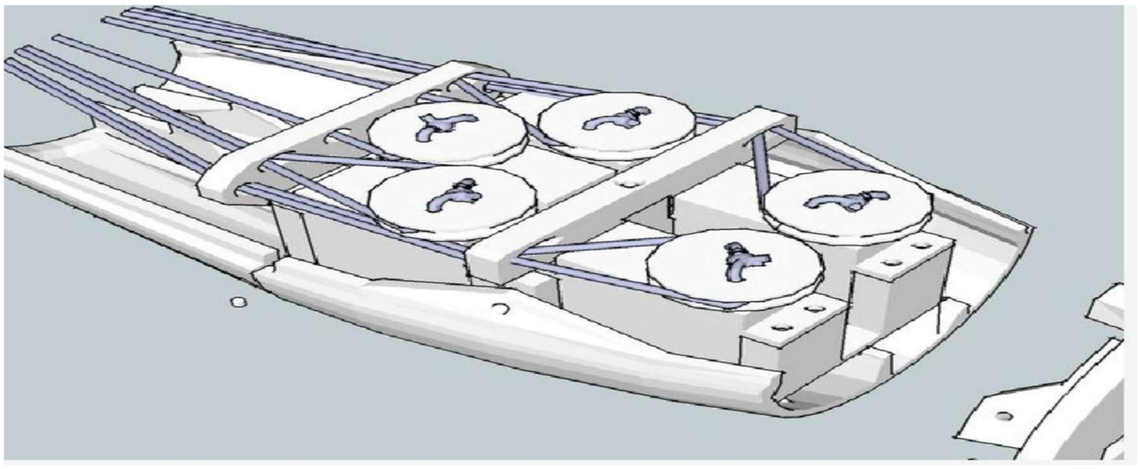


Figure 20: Adding the String with the servos

4. FUCTIONAL TESTING

A. SOFTWARE IMPLEMENTATION

a) Software Download

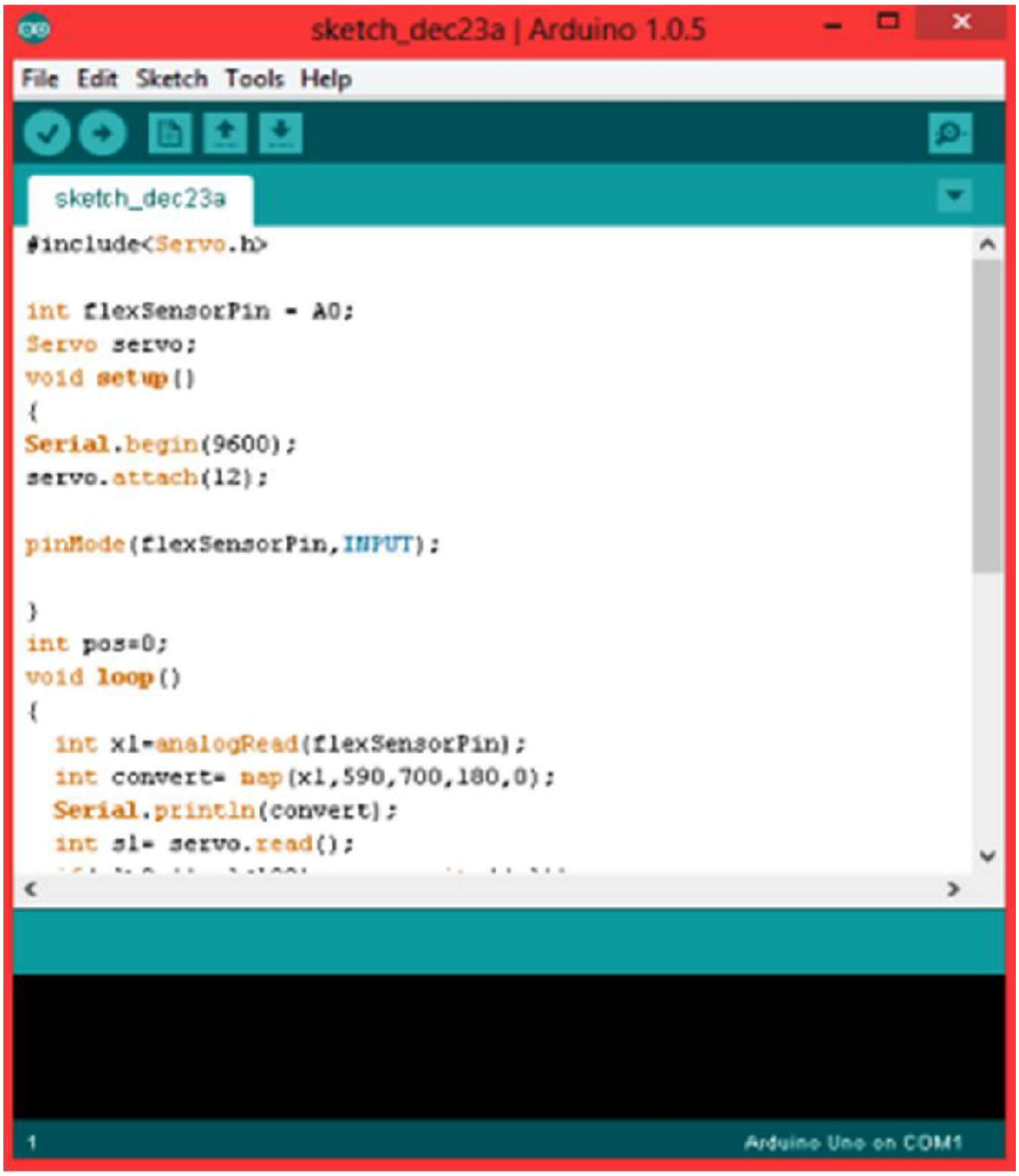
Following the instructions on the Getting Started section of the Arduino web site, http://arduino.cc/en/Guide/HomePage. we at first downloaded the latest version of Arduino, arduino-1.0.5- windows.

b) Software Installation

Going all the way through the steps to where we saw the pin 13 LED blinking. This is the indication that we had all software and drivers successfully installed and could start exploring with our own programs.

c) Software Execution

- Connected our Arduino to the computer with the USB cable. We did not need the battery for that time. The green PWR LED will light. If there was already a program burned into the Arduino, it will run.
- Started the Arduino development environment. In Arduino-speak, programs are called “sketches”, but here we will just call them programs. Our window would look something like this.
- Then we clicked the Upload button or Ctrl-U to compile the program and load on the Arduino board.



```
#include<Servo.h>

int flexSensorPin = A0;
Servo servo;
void setup()
{
Serial.begin(9600);
servo.attach(12);

pinMode(flexSensorPin,INPUT);

}
int pos=0;
void loop()
{
  int x1=analogRead(flexSensorPin);
  int convert= map(x1,590,700,180,0);
  Serial.println(convert);
  int s1= servo.read();
```

B. POWER ISSUE

i. CONNETCING A BATTERY

- For stand-alone operation, the board is powered by a battery rather than through the USB connection to the computer. While the external power can be anywhere in the range of 6 to 24 V (We used here a 11.1V Lithium ion battery), a standard 9 V battery is convenient.
- We then stablished Vin and Gnd connections on the board, it is better to solder the battery snap leads to a DC power plug and connect to the power jack on the board.

N.B Watch the polarity as you connect your battery to the snap as reverse orientation could blow out your board.

- Then we disconnected our Arduino from the computer. Then we connected a 11.1 V battery to the Arduino power jack using the battery snap adapter confirming that the blinking program runs.

This showed that we could power the Arduino from a battery and that the program we download ed runs without needing a connection to the host PC.

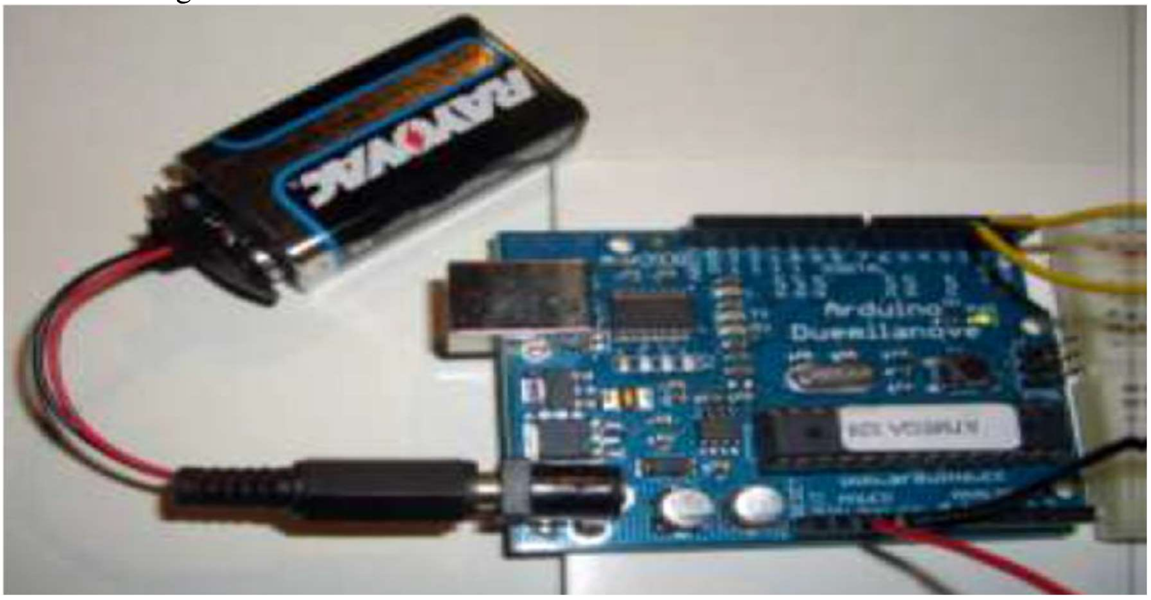


Figure 21: Connecting a Battery

ii. SETTING UP THE VOLTAGE OF POWER IC

Battery power input is 11.1V. This input is supplied to 2 power ICs. Through power ICs we have controlled the input voltage to get the desired output of 7V & 3V. As the threshold voltage of Arduino is 7V and the threshold voltage of servo is 3V.One power IC is set to get output of 7V. The output terminal of that power IC is connected to Vin and Gnd pin of Arduino. As the threshold voltage of servo motor is 3V. Here the power adjustment of the power IC is done by keeping the multi-meter positive and negative terminals into the input side of the power IC, IN+ & IN- and we varied the voltage with the help of power IC voltage-variable nob. Another power IC is set to get output of 3V. The output terminal of that power IC is connected to PCB and 6 terminals are made short with that so that 6 motors can get power supply. Here the voltage adjustment diagram of power IC is given below.

iii. FINAL STEP: WIRING OF THE TOTAL CIRCUIT WITH EXECUTION

After completing all the steps mentioned above we assembled all the stationary circuits together to build the main circuit for execution. The main circuit diagram is given below:-

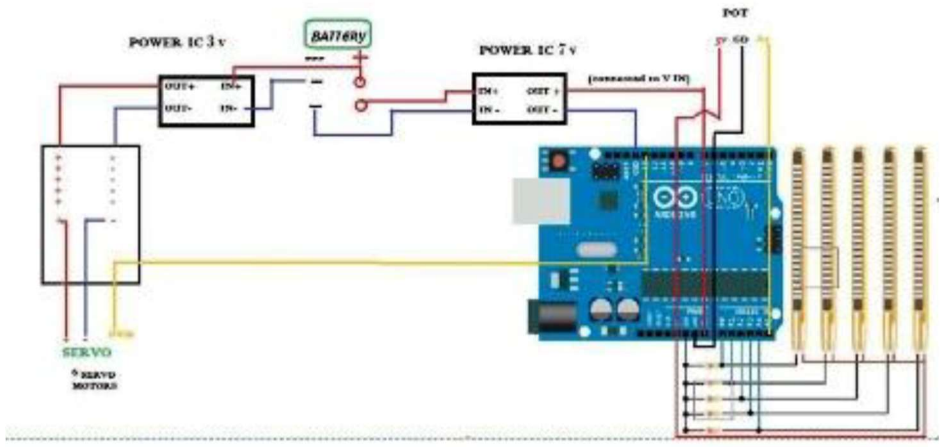


Figure 22: Wring of the total circuit

Application

The application of animatronic hands are numerous. They are listed below.

- ✓ Extra hand in chemical fields so that the skin diseases are avoided.
- ✓ Physically challenged people use it as an extra alternative hand.
- ✓ Two hands for the same operation can be utilized with this animatronic hand.
- ✓ Its applications in space for repair of space station are useful.
- ✓ Diffusion of bombs with these animatronic hands are preferred where there is high risk of lives.

Advantages

- ✓ Ease to handle in a simpler manner.
- ✓ Requires less maintenance.
- ✓ The technology doesn't bring any unemployment issues as it only safeguards the employees either be the chemical industry or the physically challenged persons.

Disadvantages

- ✓ Good knowledge is required by the worker who is working on the animatronic hand.

Future Scope

Digital camera can be places on the animatronic hands. It will record the motion of the hand if a robotic hand is at a distance which can bring many insights to the data.

Conclusion

We are glad that we chose to completing this project on the Arduino. It was our first real coding experience on this platform, and we can say that compared to writing C++, writing Wiring libraries for Arduino makes for a much more fun and Productive experience We are grateful that our time on the C++ taught us a lot about what is happening behind the scenes, but quit honestly it is nice to not have to worry about it so much. One thing we learned from this project is that servos and flex sensors in positioning, timing and environmental texture can lead to all sorts of undesirable readings. We were a bit disappointed with the performance of the SG90 servos in this particular use case, It required a lot of the fine-tuning to get readings accurate as the servo rotated. As stated previously, another area I need to look into is battery power. This project is a poor use case for a 11.1V battery. A better long-term portable power supply would include a higher efficiency in the output. That's why we also used a PC for the long-term power supply. Although the Animatronic hand did not operate with no errors, it is a great success overall. The Animatronic hand met all safety restrictions, easy to operate and energy efficient. This types of animatronic hand can be used for various purposes. The Animatronic Hand can be implemented in all the sectors where human interaction is needed, like-Handling of the explosive objects, performing various sophisticated operational jobs in the medical sectors, Industrial manufacturing etc. With more time and resources put for things like motors and base design we can carry a much larger payload and have a sturdier platform to carry things in. Much of this project could be used or improved upon by future EEE students.

REFERENCE

[1] The Absolute Beginner's Guide to Arduino http://forefront.io/a/beginners-guide-toarduino
[2] Principles of Robotics http://www.g9toengineering.com/resources/robotics.html
[3] Servo Motor | Servo Mechanism | Theory and Working Principle http://www.electrical4u.com/servo-motorservo-mechanism-theory-and-workingprinciple/
[4] DC Servo Motors | Theory of DC Servo Motor http://www.electrical4u.com/dc-servomotors-theory-and-working-principle/
[5] Introduction to Servo Motors http://mechatronics.mech.northwestern.edu /design_ref/actuators/servo_motor_intro.html
[6] DIY Robotic Hand Controlled by a Glove and Arduino by dschurman http://www.instructables.com/id/DIYRobotic- Hand-Controlled-by-a-Glove-and-Arduino/
[7] Animatronics http://en.wikipedia.org/wiki/Animatronics
[8] Simple Servo Bed for InMoov http://www.thingiverse.com/thing:65274
[9] AniHand - Animatronic Hand http://letsmakerobots.com/node/34639
[10] Calibrator: An Arduino library to calibrate sensors hooked to analog inputs http://julianvidal.com/blog/calibrator-anarduino-library-to-calibrate-sensorshooked-to-analog-inputs/
[11] Arduino - Begin. http://arduino.cc/en/Serial/begin
[12] Arduino learning pdf http://www.ele.uri.edu/courses/ele205/Arduino%20-%20Learning.pdf
[13] Arduino Microcontroller Guide http://www.me.umn.edu/courses/me2011/arduino/arduinoGuide.pdf
[14] Arduino functions http://playground.arduino.cc/Code/Function
[15] Structuring Your Code into Functional Blocks https://www.inkling.com/read/arduinocookbook-michael-margolis-2nd/chapter-2/recipe-2-10
[16] Arduino programming Problem solution http://stackoverflow.com/questions/12975895/arduinoargument-of-type-voidclassname-does-not-match-void-whi
[17] Blink Without Delay http://arduino.cc/en/Tutorial/BlinkWithoutDelay
[18] Operating Two Servos with the Arduino http://www.robotoid.com/appnotes/arduino-operating-two-servos.html
[19] Servo Problems With Arduino http://rcarduino.blogspot.com/2012/04/servo-problems-with-arduino-part-1.html
[20] Problem with arduino servo http://electronics.stackexchange.com/questions/93746/problem-witharduino-servo
[21] Arduino Servo control problem, external servo power source https://forum.sparkfun.com/viewtopic.php?f=32&t=24263
[22] Problem with multi servos on Arduino http://www.youtube.com/watch?v=iZFO_8hH7QY
[23]Multiple servo control http://www.instructables.com/id/Serial-Servo-Controller-wAduinoControlUp-To-1/
[24]Flex sensor http://mech207.engr.scu.edu/SensorPresentations/Jan%20-%20Flex%20Sensor%20Combined.pdf
[25] Arduino map() method http://stackoverflow.com/questions/9024124/arduino-map-method-why
[26] Arduino programming tutorial http://opensourcehardwaregroup.com/tutorial-09-reading-analogpins-and-convertingthe-input-to-a-voltage/
[27] Calibration of flex sensors http://www.mtbs3d.com/phpBB/viewtopic.php?f=138&t=17799
[28] Sensing A Bend With A Flex Sensor +Arduino http://bildr.org/2012/11/flex-sensorarduino/
[29] Arduino tutorial PWM http://www.youtube.com/watch?v=Y1QraI5i_XM

Marks Evaluation Sheet (7th Semester)

Presentation No.	**Faculty Name**	**Siddhant Pratap Singh**	**Akanksha Tiwari**	**Jai Hind**	**Utkarsh Pandey**	**KM Adab Istiaq**	**Rajan Joshi**
Presentation No. 1.	Dr. Ashutosh Dwivedi						
	Dr. Abhishek Mishra						
	Dr. Akhilesh Gupta						
	Mr. Sushil Kumar						
	Ms. Zeba Baktiyar						
	Mr. Priyank Srivastava						
	Mr. S. K. Singh						
	Mr. Jitendra Srivastava						
	Mr. Saurabh Bajpai						
	Mr. Kshitiz Srivastava						
	Dr. Ashutosh Dwivedi						

Presentation No. 2.	Dr. Abhishek Mishra						
	Dr. Akhilesh Gupta						
	Mr. Sushil Kumar						
	Ms. Zeba Baktiyar						
	Mr. Priyank Srivastava						
	Mr. S. K. Singh						
	Mr. Jitendra Srivastava						
	Mr. Saurabh Bajpai						
	Mr. Kshitiz Srivastava						
Presentation No. 3.	Dr. Ashutosh Dwivedi						
	Dr. Abhishek Mishra						
	Dr. Akhilesh Gupta						
	Mr. Sushil Kumar						

	Ms. Zeba Baktiyar						
	Mr. Priyank Srivastava						
	Mr. S. K. Singh						
	Mr. Jitendra Srivastava						
	Mr. Saurabh Bajpai						
	Mr. Kshitiz Srivastava						
Presentation No. 4.	Dr. Ashutosh Dwivedi						
	Dr. Abhishek Mishra						
	Dr. Akhilesh Gupta						
	Mr. Sushil Kumar						
	Ms. Zeba Baktiyar						
	Mr. Priyank Srivastava						
	Mr. S. K. Singh						
	Mr. Jitendra Srivastava						

	Mr. Saurabh Bajpai						
	Mr. Kshitiz Srivastava						

Marks Evaluation Sheet (8th Semester)

Presentation No.	**Faculty Name**	**Siddhant Pratap Singh**	**Akanksha Tiwari**	**Jai Hind**	**Utkarsh Pandey**	**KM Adab Istiaq**	**Rajan Joshi**
Presentation No. 1.	Dr. Ashutosh Dwivedi						
	Dr. Abhishek Mishra						
	Dr. Akhilesh Gupta						
	Mr. Sushil Kumar						
	Ms. Zeba Baktiyar						
	Mr. Priyank Srivastava						
	Mr. S. K. Singh						
	Mr. Jitendra Srivastava						
	Mr. Saurabh Bajpai						

	Mr. Kshitiz Srivastava						
Presentation No. 2.	Dr. Ashutosh Dwivedi						
	Dr. Abhishek Mishra						
	Dr. Akhilesh Gupta						
	Mr. Sushil Kumar						
	Ms. Zeba Baktiyar						
	Mr. Priyank Srivastava						
	Mr. S. K. Singh						
	Mr. Jitendra Srivastava						
	Mr. Saurabh Bajpai						
	Mr. Kshitiz Srivastava						
	Dr. Ashutosh Dwivedi						
	Dr. Abhishek Mishra						

Presentation No. 3.	Dr. Akhilesh Gupta						
	Mr. Sushil Kumar						
	Ms. Zeba Baktiyar						
	Mr. Priyank Srivastava						
	Mr. S. K. Singh						
	Mr. Jitendra Srivastava						
	Mr. Saurabh Bajpai						
	Mr. Kshitiz Srivastava						
Presentation No. 4.	Dr. Ashutosh Dwivedi						
	Dr. Abhishek Mishra						
	Dr. Akhilesh Gupta						
	Mr. Sushil Kumar						
	Ms. Zeba Baktiyar						

	Mr. Priyank Srivastava						
	Mr. S. K. Singh						
	Mr. Jitendra Srivastava						
	Mr. Saurabh Bajpai						
	Mr. Kshitiz Srivastava						