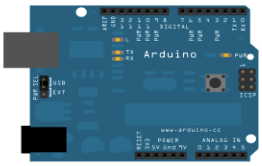


ECEN – 5613 EMBEDDED SYSTEM DESIGN

Hand Movement Detection using Arduino Uno

Final Project Report

Table of Contents

Chapter

3

Introduction and Overview

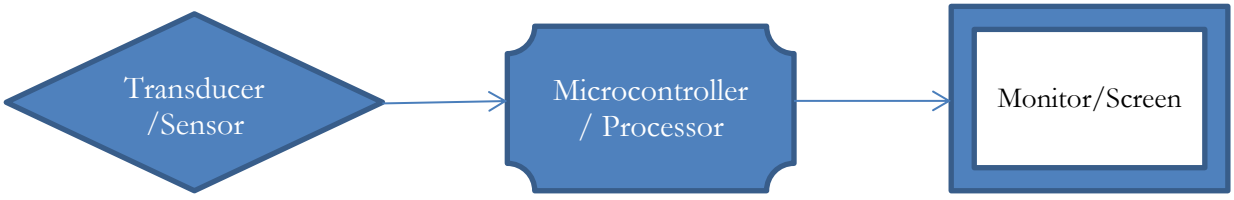
The objective of the Project is to implement flex resistors to detect grasp movement of the hand using a glove assembly and a suitable controller

The idea for the motion tracking for the hand struck me when I used the gesture based controls on my Laptop's touchpad and wanted to create a device which could be interfaced with a computer to control the functions of the computer using gesture based operations. I believe as the display screen technology has improved in recent times to allow multi-span monitors to be used for a single process. As the visual interface size increases there will be a need to appropriately control the next generation of multidimensional/multi-span visual devices with gesture based controlling devices which would me more efficient and natural to basic human operations. Humans have used gestures as a form of non-verbal communication for conveying important messages and are a function of the human body, processed sometimes conscious or as an emotional reflex. Hence as operations to be performed on the computers become more and more complex, a gesture based solution to the problem is a suitable way for solving the above problem.

Overview

To achieve the whole objective as part of the ESD project would be too ambitious and time consuming. Hence I decided to split the objective to a smaller portion; my basic goal would be to interface sensors to obtain change in hand movement. This would involve designing hardware for the same.

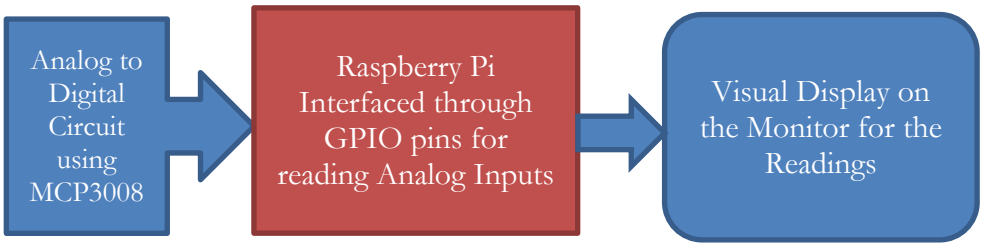
A controller which would be capable to be interfaced with the sensors and display it over a monitor/screen. Below diagram depicts the overview of the concept.



Chapter

4

Attempt to Design with Raspberry Pi



My initial choice for execution of the project was Raspberry Pi for the controller. I chose Raspberry Pi because it had the right amount of features required for achieving the project goals.

A HDMI output for displaying on a standard monitor screen. Its shared GPU memory which was sufficient enough to generate graphics for the visual representation and a super-fast processor at an amazingly low cost.

By following the quick-start guide available on the website I was able to fire up the Raspberry Pi with the devices I had. Additional purchase of the SD card had to be made for the Operating System to be loaded.

The Raspberry Pi manufacturers provide the Raspbian "Wheezy" Operating system which is a LXDE (Light X11 Desktop Environment) which is a fast and low resource consuming operating system suitable for a board like this but provides a multitude of functionality.

Once I had the Operating System running my next goal was to work towards testing and controlling the GPIO pins.

This is where I ran into too many Snags!

Issue with GPIO pins and A/D conversion

An easy approach to getting values from the flex resistor would be to use an Analog to Digital Converter Circuit for the Raspberry Pi Interface.

I planned to use the MCP3008 as done in a tutorial provided by "adafruit.com"

Link: http://learn.adafruit.com/reading-a-analog-in-and-controlling-audio-volume-with-the-raspberry-pi/connecting-the-cobbler-to-a-mcp3008

I managed to build the circuit on the breadboard and while testing applying any input to the analog Channels on the A/D converter would reset the Raspberry Pi.

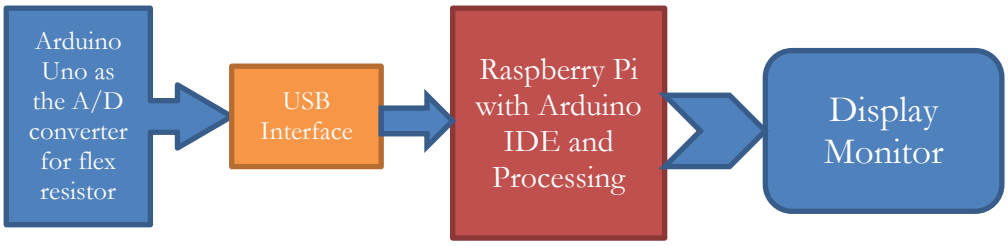
I could not figure out why this was happening. I believe that no over voltage protection provided on the onboard circuitry was causing this to happen.

Script writing on the Raspberry Pi

The second problem I ran into was that coding for all the pins in C code for the raspberry pi on such as short time would be really time consuming.

All the example codes I saw on the Raspberry Pi website were written in Python script. Which seemed like the most efficient way of Coding for this task. Since I had already squandered most of my time on the hardware implementation of the Raspberry Pi and was not functioning successfully. I abandoned the Plan to use the Raspberry Pi for The A/D conversion process.

Using the Arduino Uno as Analog to Digital Converter along with the Raspberry Pi



Since I already had a Linux based operating system running on the Raspberry Pi, my next approach was to utilize the Raspberry Pi's USB port to connect an Arduino Uno board and user it as an Analog to Digital Converter and communicate serially.

Since the software for the Arduino Uno are java based (i.e. Arduino IDE and Processing) I installed the Arduino Software using the following guide available online on the raspberry Pi

http://raspberrypi.homelabs.org.uk/raspberrypi-the-arduino-development-tool/

Arduino IDE is a Utility which is similar to a combination of Eclipse/SDCC programming editor, Atmel FLIP and Tera Term the serial monitor , All combined in one. Arduino IDE is a derivative of the "Processing" utility. . (Description and download links for both are made available in Utility Folder)

And I managed to install the processing utility software onto the Raspberry Pi using the following guide available online

http://www.designspark.com/blog/processing-on-the-raspberry-pi

Once the Arduino and Processing were installed onto the Raspberry Pi I ran to the following problems:

A) The Arduino software was unable to burn code onto the Arduino flash memory. Although serial communication was working and ASCII messages from the Arduino were being displayed on the "Serial Monitor" of Arduino IDE running on the Raspberry Pi

B) The Processing software would not communicate over the serial port because the RX/TX libraries in java ver. 6 are incompatible with the Processing software. Copying the native java RX/TX library did not solve the issue.

After facing above problems I decided to abandon the Raspberry Pi as the Microcontroller.

Why I did not go for another A/D circuit

The cost of the project had gone over $110 and I did not wish to spend any more on the same. The flex resistors being the most expensive parts had thrown me out of budget.

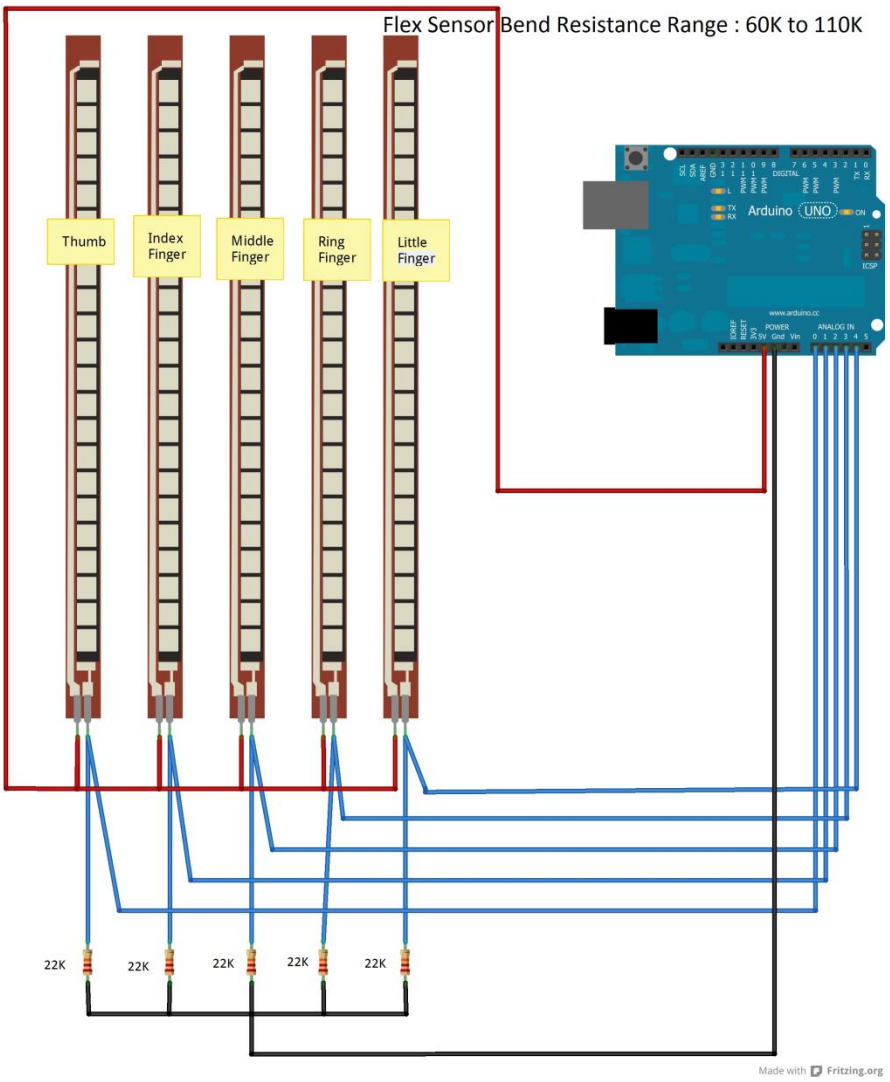
I already had an Arduino Uno board from a previous project and planned to use it for further implementation instead of going for new hardware.

Chapter

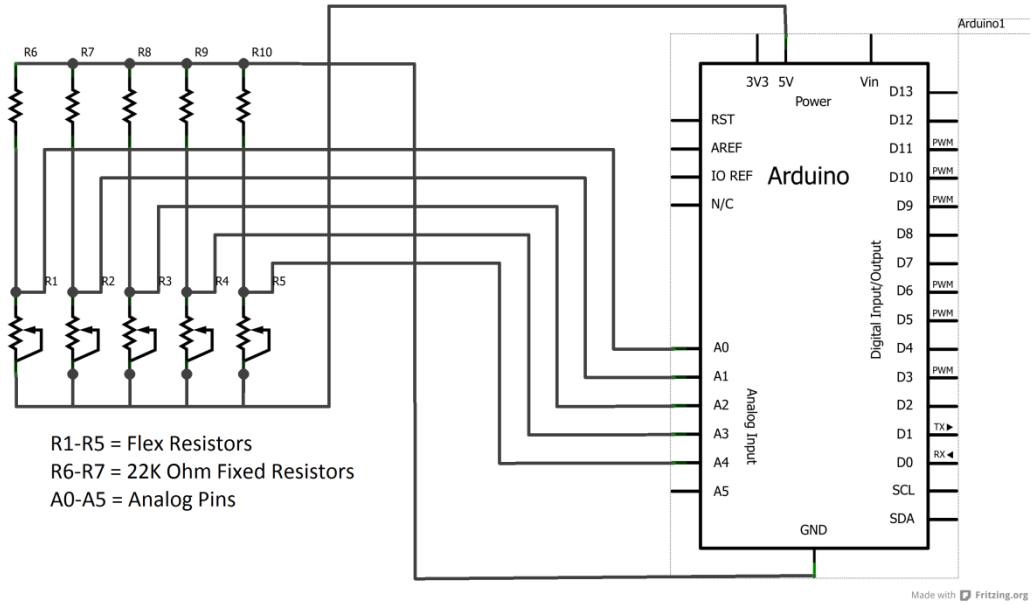
5

Design with Flex Resistor and Glove Assembly

Below are the Layout Diagram for the Flex resistor and Glove Assembly.



Circuit Schematic:



Original Size Schematic and Layouts are available in their respective folders.

As show in the schematics the Flex resistors R1 to R5 are used in combination of the R6-R10 which are 22k fixed resistors connected in a Voltage divider fashion to obtain as stable analog voltage at the A0 – A4 analog input pins on the Arduino.

Resistor R1 to R5 is the flex resistor for the thumb, index, middle, ring and little finger respectively.

The Sparkfun website from where the Flex resistors were purchased also has a link to the following website in the product description:

http://bildr.org/2012/11/flex-sensor-arduino/

This link provides the code leveraged for creating the program used to obtain the analog values and map it for creating the movement detection code and also forms a base for the gesture detection and the game implementation.

Below is the snipped of code leveraged from the website for the implementation

```
int flexSensorPin = A0; //analog pin 0

void setup(){
  Serial.begin(9600);
}

void loop(){
  int flexSensorReading = analogRead(flexSensorPin);

  Serial.println(flexSensorReading);


  //In my tests I was getting a reading on the arduino between 512, and 614.
  //Using map(), you can convert that to a larger range like 0-100.
  int flex0to100 = map(flexSensorReading, 512, 614, 0, 100);
  Serial.println(flex0to100);

  delay(250); //just here to slow down the output for easier reading
}
```

For well-defined explanations on the above functions please refer to the Function reference for Arduino/Processing provided in the datasheet folder.

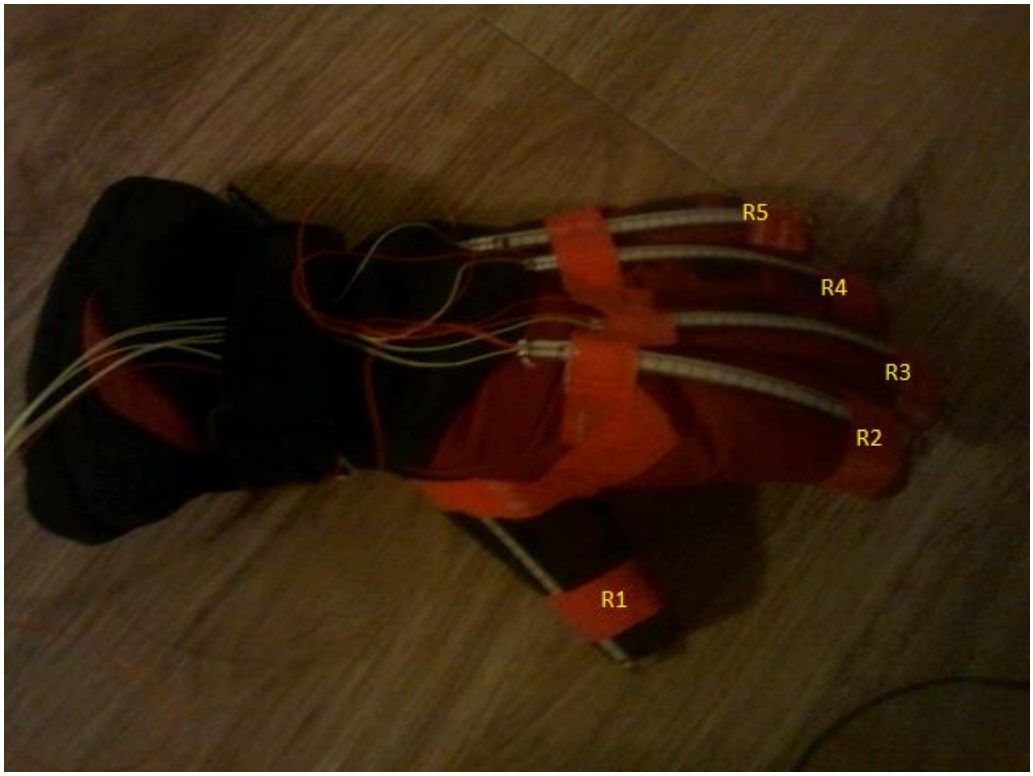
Using the above code I was able to create the code which could obtain the raw values from the flex resistor using the Arduino's inbuilt 10 bit Analog to Digital converter which provided a resolution from 1 to 1023 bits and map the values to between 0 and 100.

For the raw values I was able to get values from 600 to 800~850 hence the mapped values would extend from 0 to 70~80 for a maximum flex.

Calibrating the flex resistors was tough than I had imagined as movement of the flex resistors on the glove would cause a permanent shift in their position and would change the readings.

This was interfering when calibrating to an accuracy of 10 degrees every time the glove would be worn on removed there would be some changes in the resistors.

Below is a picture of the actual implementation of the flex resistors on the Glove



Chapter

6

Code Implementation on the Arduino IDE and Processing utilities

There are two sets of codes developed for this project

1. Code for movement Detection and Gesture detection

   The code developed in Arduino IDE for the movement and Gesture detection is as follows:

```
//Global Declaration of Analog Pins
int f0 = A0; //analog pin 0 connected to the thumb flex Res.
int f1 = A1; //analog pin 1 connected to the index finger
int f2 = A2; //analog pin 2  connected to the middle finger
int f3 = A3; //analog pin 3 connected to the ring finger
int f4 = A4; //analog pin 4  connected to the Little finger

void setup(){
  //Initialize the serial port at 9600 baud
  Serial.begin(9600);
}

void loop(){
  //read values from analog pins
  int fr0 = analogRead(f0);
    int fr1 = analogRead(f1);
      int fr2 = analogRead(f2);
        int fr3 = analogRead(f3);
          int fr4 = analogRead(f4);

//code continued on second page
```

```
/* DEBUG CODE TO DISPLAY RAW VALUES ON SERIAL MONITOR/TERMINAL
Serial.print(" fr0 RAW:");
 Serial.print(fr0);
 Serial.print(" fr1 RAW:");
 Serial.print(fr1);
 Serial.print(" fr2 RAW:");
 Serial.print(fr2);
 Serial.print(" fr3 RAW:");
 Serial.print(fr3);
 Serial.print(" fr4 RAW:");
 Serial.print(fr4);
*/


 //I was getting raw values from 680 to 1024 from the ADC
 //Using map(), you can convert that to a larger range like 0-100.
 int fr0map = abs(map(fr0, 680, 1024, 0, 100));
  int fr1map = abs(map(fr1, 631, 1024, 0, 100));
   int fr2map = abs(map(fr2, 683, 1024, 0, 100));
    int fr3map = abs(map(fr3, 697, 1024, 0, 100));
     int fr4map = abs(map(fr4, 707, 1024, 0, 100));

 /*DEBUG CODE TO PRINT MAPPED VALUES TO THE TERMINAL
 Serial.println();
 Serial.print(" fr0map:");
 Serial.println(fr0map);
  Serial.print(" fr1map:");
 Serial.print(fr1map);
  Serial.print(" fr2map:");
 Serial.print(fr2map);
  Serial.print(" fr3map:");
 Serial.print(fr3map);
  Serial.print(" fr4map:");
 Serial.print(fr4map);
 Serial.println();
*/
```

The above patch of the code is responsible for establishing Serial communication between the Arduino and the PC's Serial Monitor.

It read the Analog Output from the Analog to Digital Converter and maps it to the 0 to 100 values

```
//CODE FOR DISPLAYING FINGER FLEXES
if(fr0map>=45 && fr1map<=45 && fr2map<=45 && fr3map<=45 && fr4map<=45)
Serial.println("Thumb Curled");
if(fr1map>=45&& fr0map<=45&& fr2map<=45&& fr3map<=45&& fr4map<=45)
Serial.println("Index Finger Curled");
if(fr2map>=45&& fr0map<=45&& fr1map<=45&& fr3map<=45&& fr4map<=45)
Serial.println("Middle Finger Curled");
if(fr3map>=45&& fr0map<=45&& fr1map<=45&& fr2map<=45&& fr4map<=45)
Serial.println("Ring Finger Curled");
if(fr4map>=45&&fr0map<=45&& fr1map<=45&& fr2map<=45&& fr3map<=45)
Serial.println("Little Finger Curled");
```

The above patch of code checks the mapped values of each finger and outputs the corresponding output to the serial monitor.

Above detection is for only single finger movements.

```
//CODE FOR DISPLAYING GESTURES
if(fr0map<=45 && fr1map>=45 && fr2map>=45 && fr3map>=45 && fr4map>=45)
Serial.println("Thumbs Up");
if(fr0map>45 && fr1map<=45 && fr2map>=45 && fr3map>=45 && fr4map>=45)
Serial.println("Pointing to Something");
if(fr0map>=45 && fr1map>=45 && fr2map>=45 && fr3map>=45 && fr4map<=45)
Serial.println("Restrooms on the First Level");
if(fr0map>=45 && fr1map>=45 && fr2map>=45 && fr3map>=45 && fr4map>=45)
Serial.println("Grasp");
if(fr0map<=45 && fr1map<=45 && fr2map>=45 && fr3map>=45 && fr4map<=45)
Serial.println("Rock On!");
//delay to slow down the output for easier reading on the terminal
  delay(750);
```

The above code is used to perform gesture detection

2. The Arduino and Processing code for Game implementation

Since I have implemented the game to be controlled by only the thumb flex movement. I have disabled the other Analog Inputs.

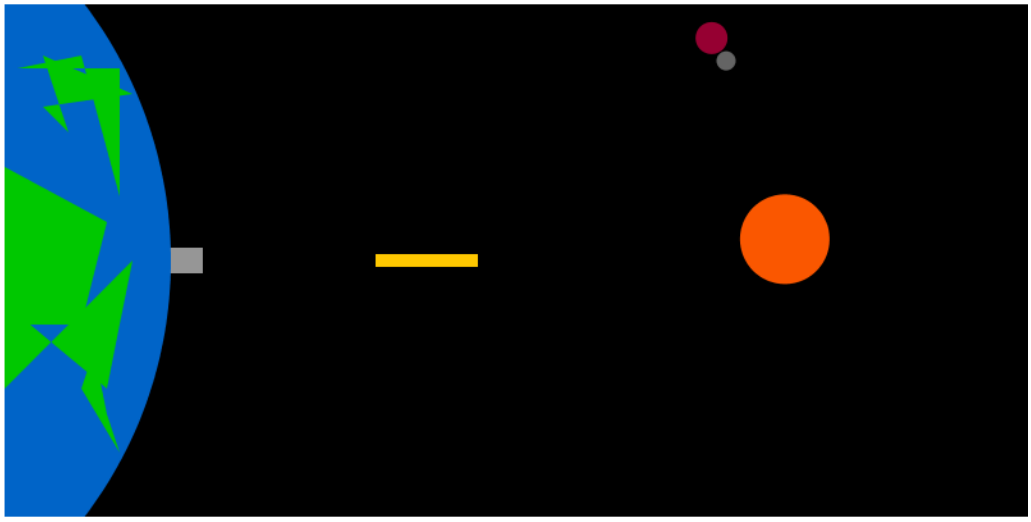
The Code running on the Arduino just repeatedly reads the Thumb flex resistor values maps them to the 0 to 100 range and sends it to the Serial Port as shown below:

```
// Global Declaration of Analog Pin
int f0 = A0; //analog pin 0

void setup(){
  //Starts Serial Communication at 9600 baud
  Serial.begin(9600);
}

void loop(){
  //Reading the Analog Pin
  int fr0= analogRead(f0);

  //Crating the mapped values
  int fr0map = abs(map(fr0, 680, 1024, 0, 100));

 //
  Serial.println(fr0map);
  delay(50);
}
```

A delay of 50 millisecond is added just to reduce the high baud rate which causes false positives that trigger the game to run erratically.

The second part of the code runs on the computer using the Processing Software.
This code is responsible for generating the graphics for the game and also reading the serial port.

The game implementation consist of a Globe and cannon/ missile launcher
Which shoots into the Sun represented by the Orange circle



The Game objective and how it operates is that the user flexes the thumb to shoot a missile from the cannon.

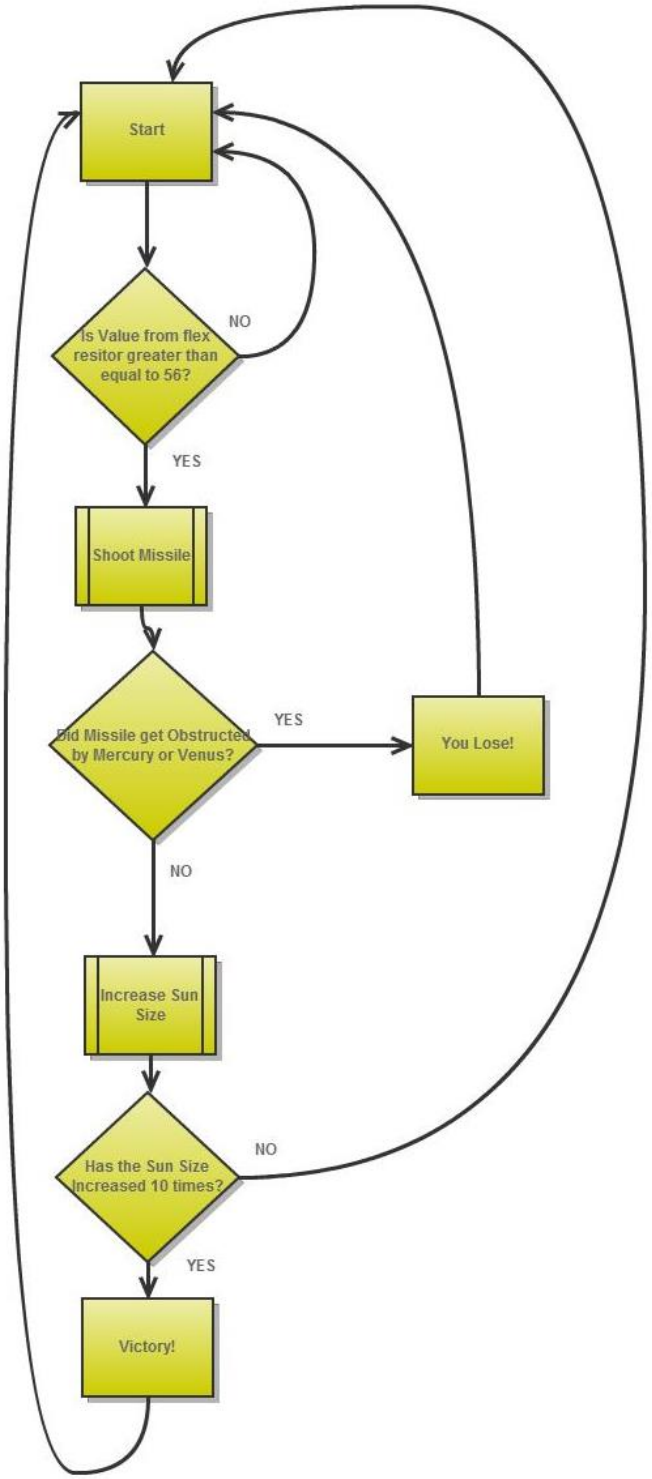
He has to time it correctly so that the planets Mercury and Venus do not obstruct the yellow beam.
If the yellow beam gets obstructed the user Loses!

If it Hits the sun the Sun grows bigger and the speed of the revolving planets increases.

The size of the sun keeps increasing 10 times for consecutive hits then the user wins and the game resets.

The algorithm for the game is as explained easily in the chart shown below:

Flowchart for the Game Operation:



The code for the game is leveraged from the website:

http://www.openprocessing.org/sketch/134

The code has been modified to include serial communication and flex resistor control with the following snippets of code

```
//Imports the serial library for processing
import processing.serial.*;

//Defining the Serial port variable
Serial myPort;
```

Above snipped imports the serial library for communication with Arduino.
Declaring a serial Buffer "myPort" for storing values.

```
void setup()
{
 println(Serial.list());
 // Opening the serial port for running at 9600 baud:
 myPort = new Serial(this, Serial.list()[0], 9600);
 size (1080,400);  //resolution of the window
 smooth ();
 noStroke ();
}
```

The above code lists the serial port and initializes the port on which Arduino is connected at 9600 baud. The resolution/windows size for the game is changed too.

```
while (myPort.available() > 0) {
   int inByte = myPort.read();  //Storing the data obtained serially into intByte Variable
   println(inByte);            //Prints the value in the Serial Monitor

 if(inByte>=56) {          //Setting the Resistance value for the thumb Flex Resistor
  if (misslePath == 120) { // Prevents interrupt of missile trajectory
    misslePath = cannonEdge;
  }
  }
 }
```

Above code checks if any data is available at the Serial port. If present stores into the "intByte" variable and checks its value if greater than or equal to 56. If condition is satisfied it shoots a missile towards the sun.

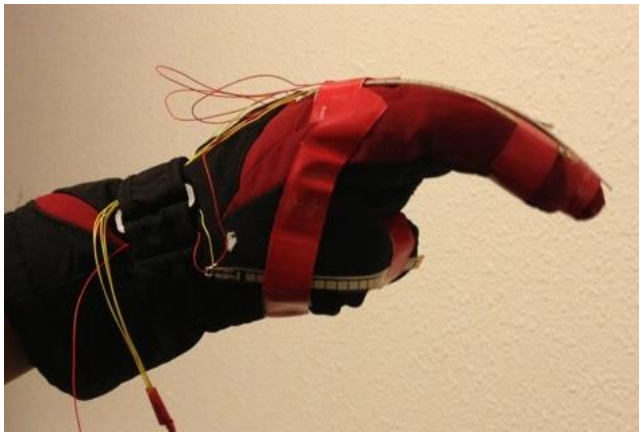
Chapter

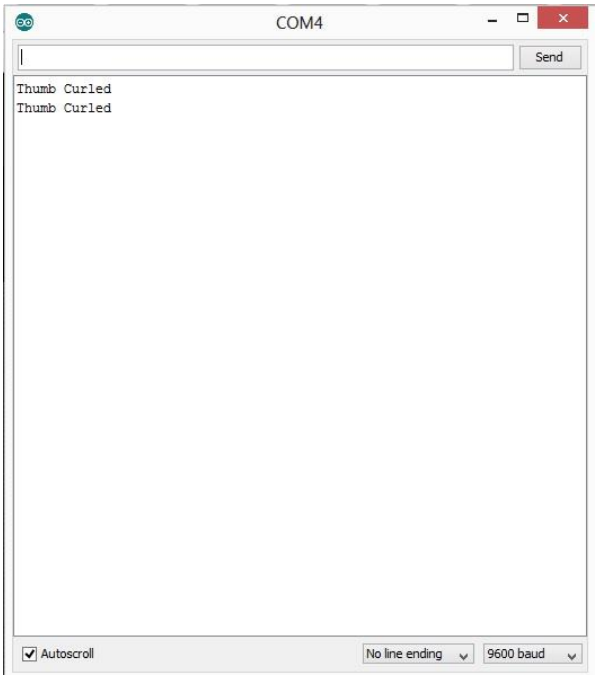
7

Results

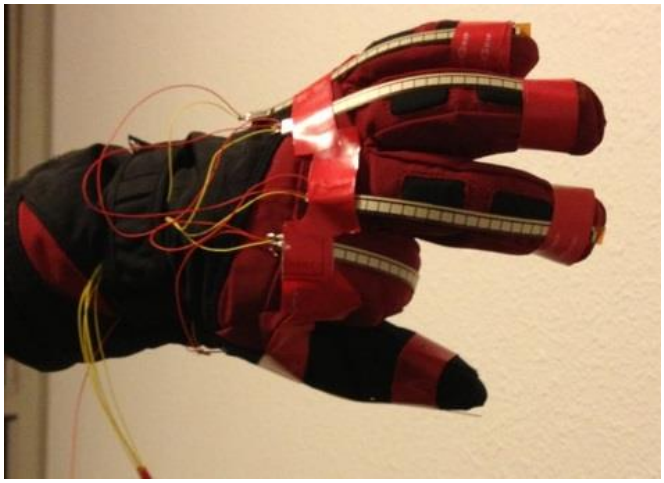
For the results I obtained are as follows:

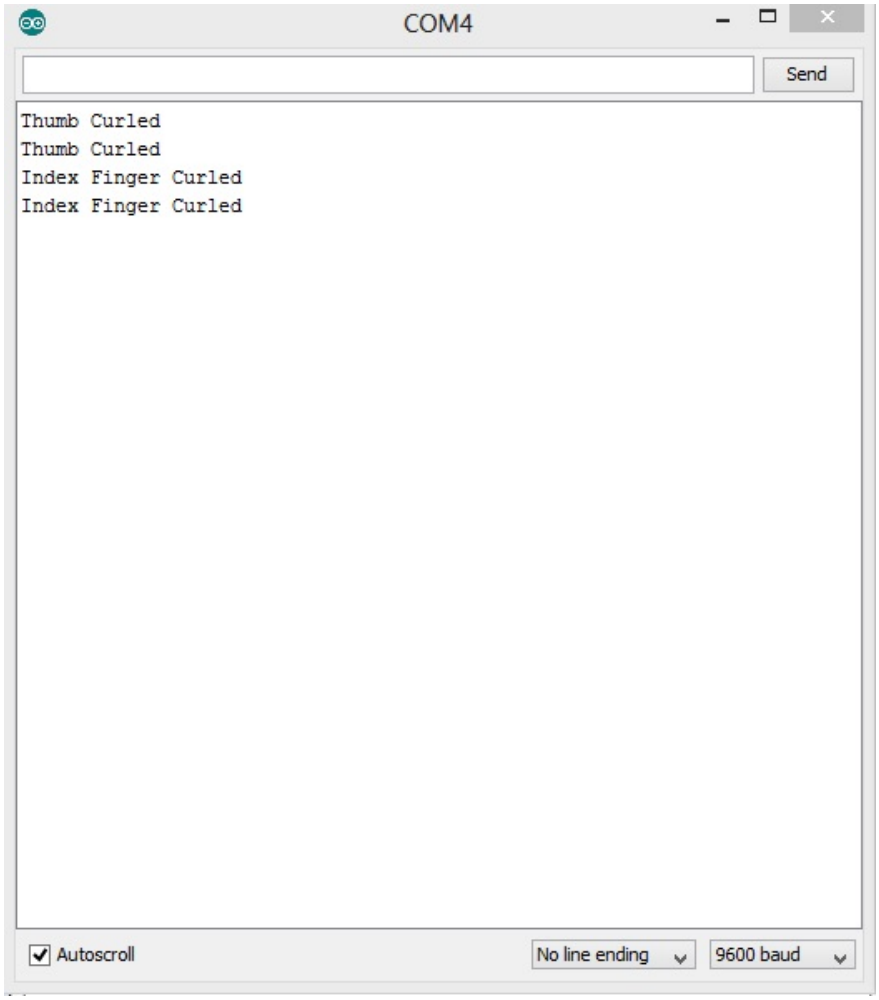
Thumb Curl and corresponding terminal/Serial Monitor Output:



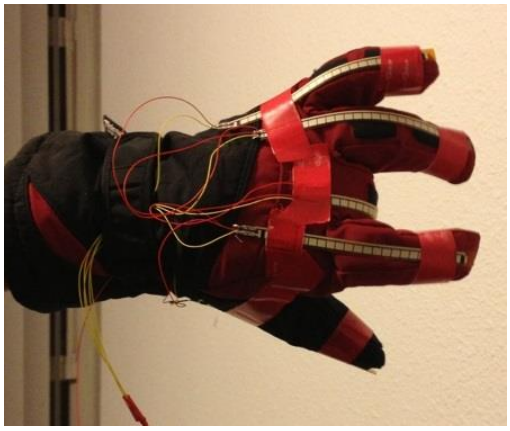


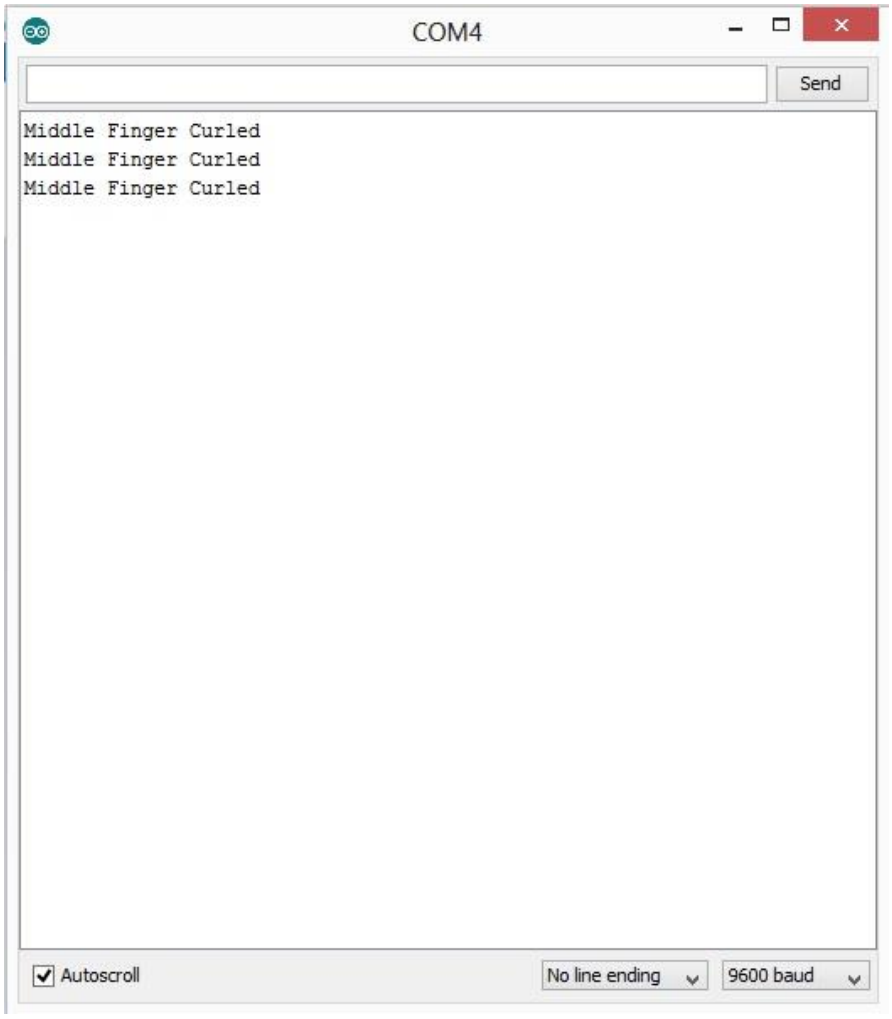
Curled Index Finger:



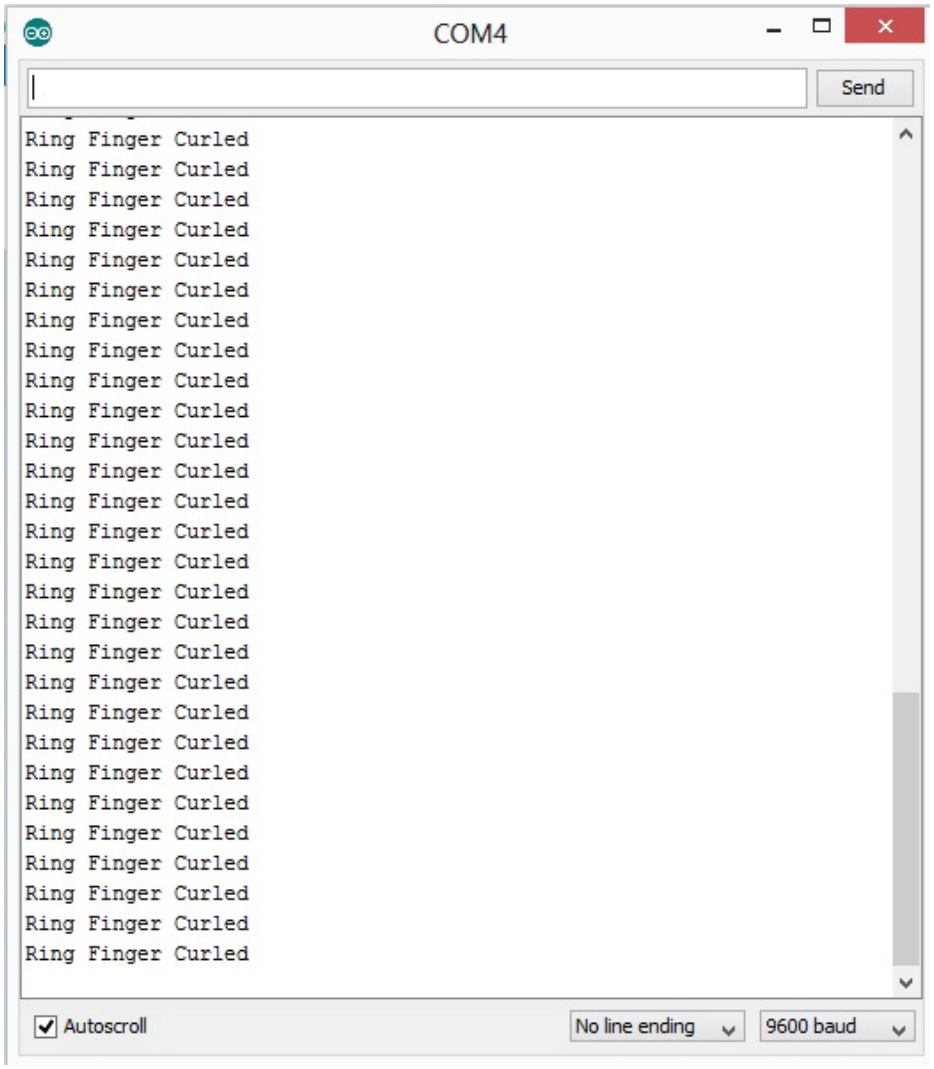


Curled Middle Finger:

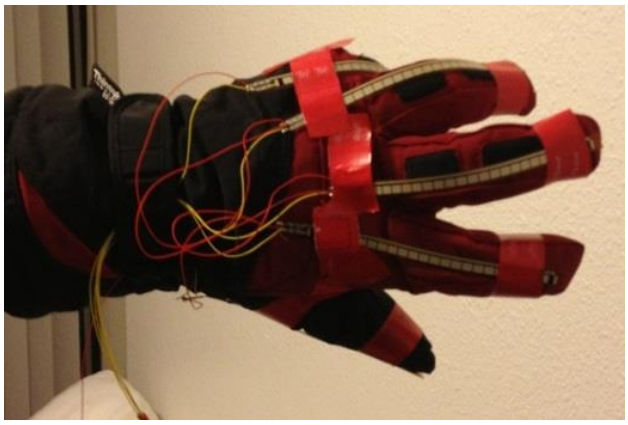


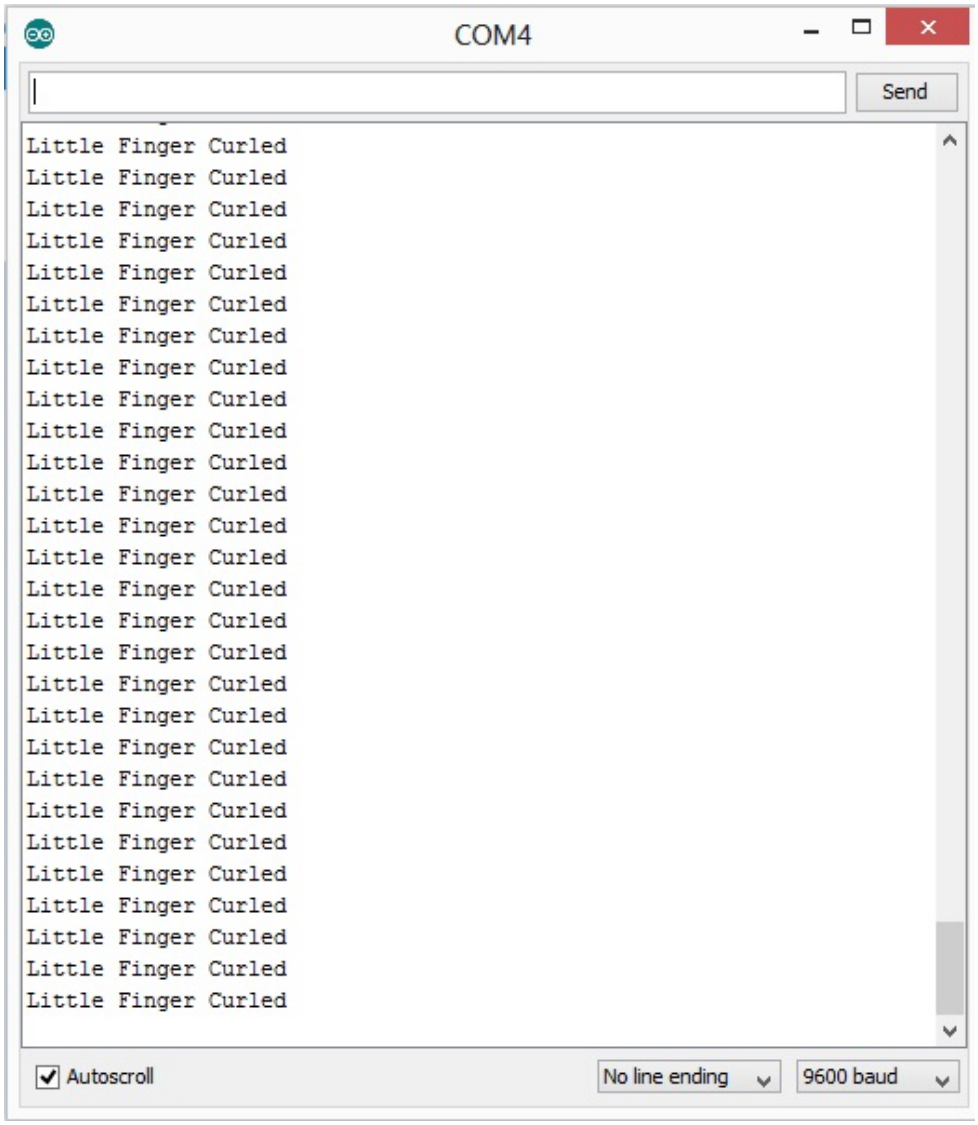


Curled Ring Finger:

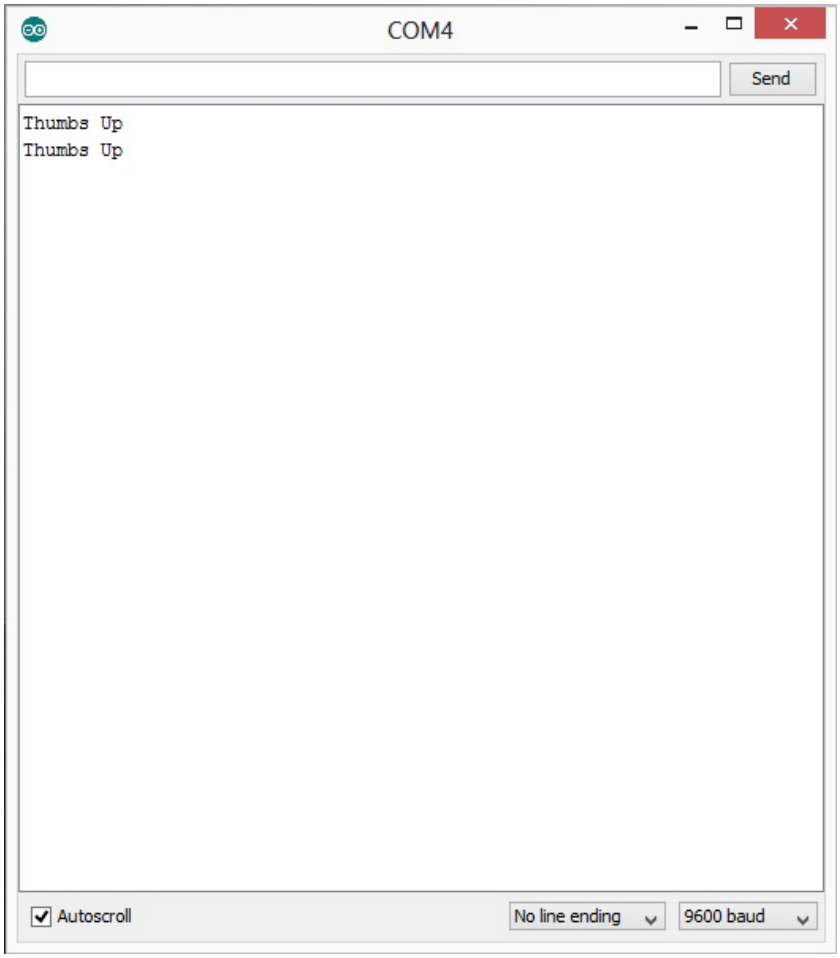


Curled Little Finger:

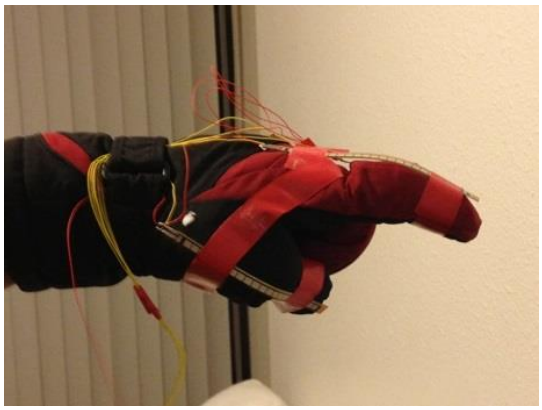


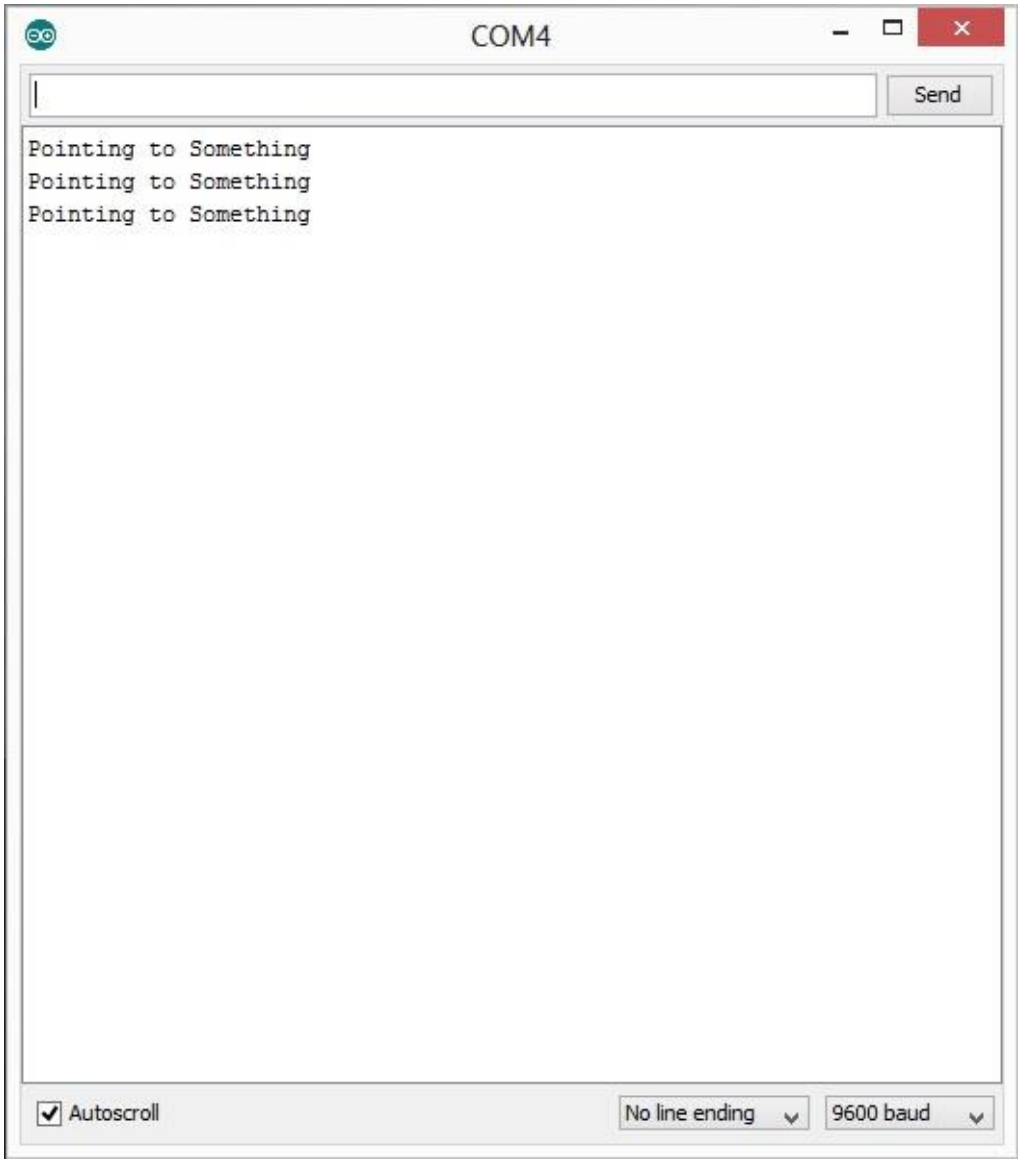


Thumbs Up Gesture:

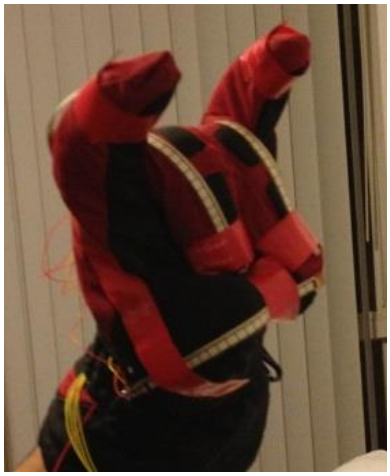


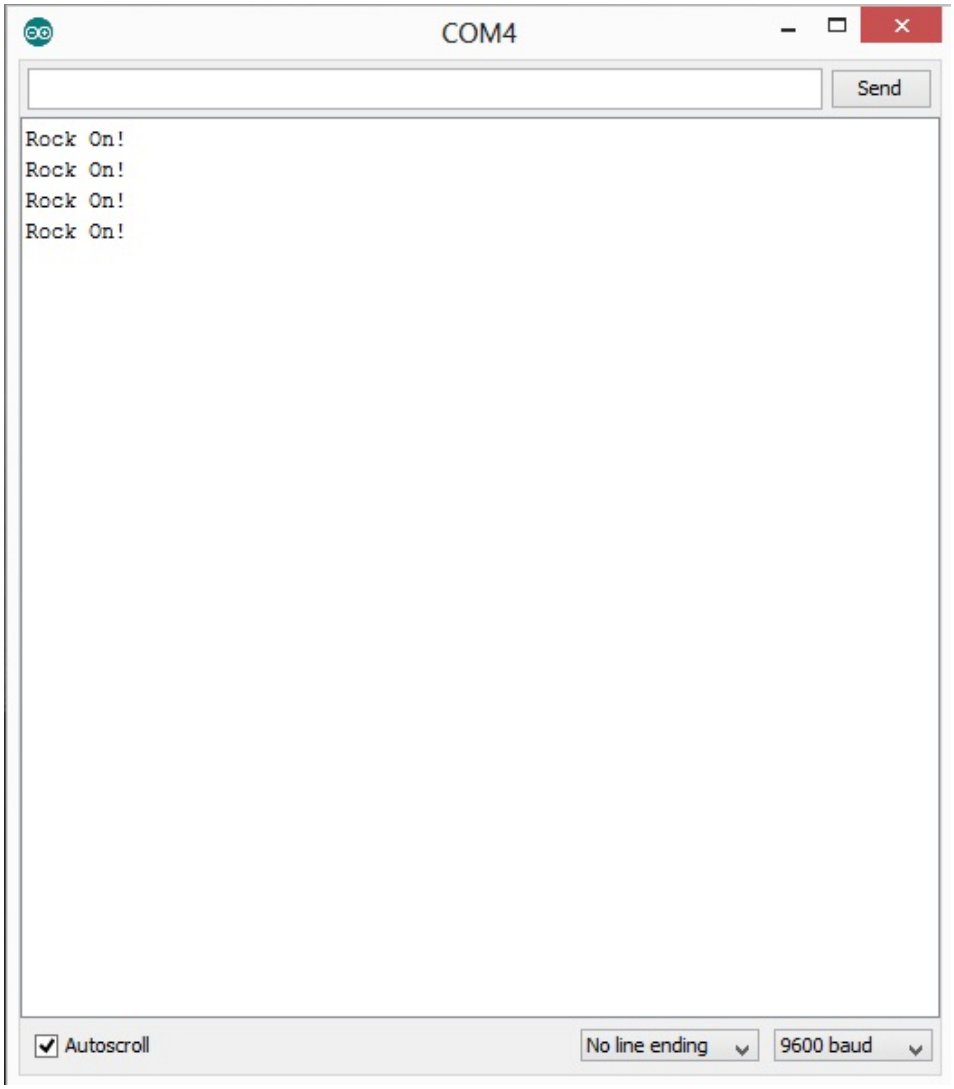
Pointing to Something:



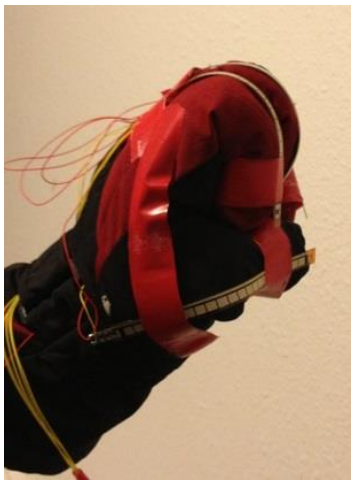


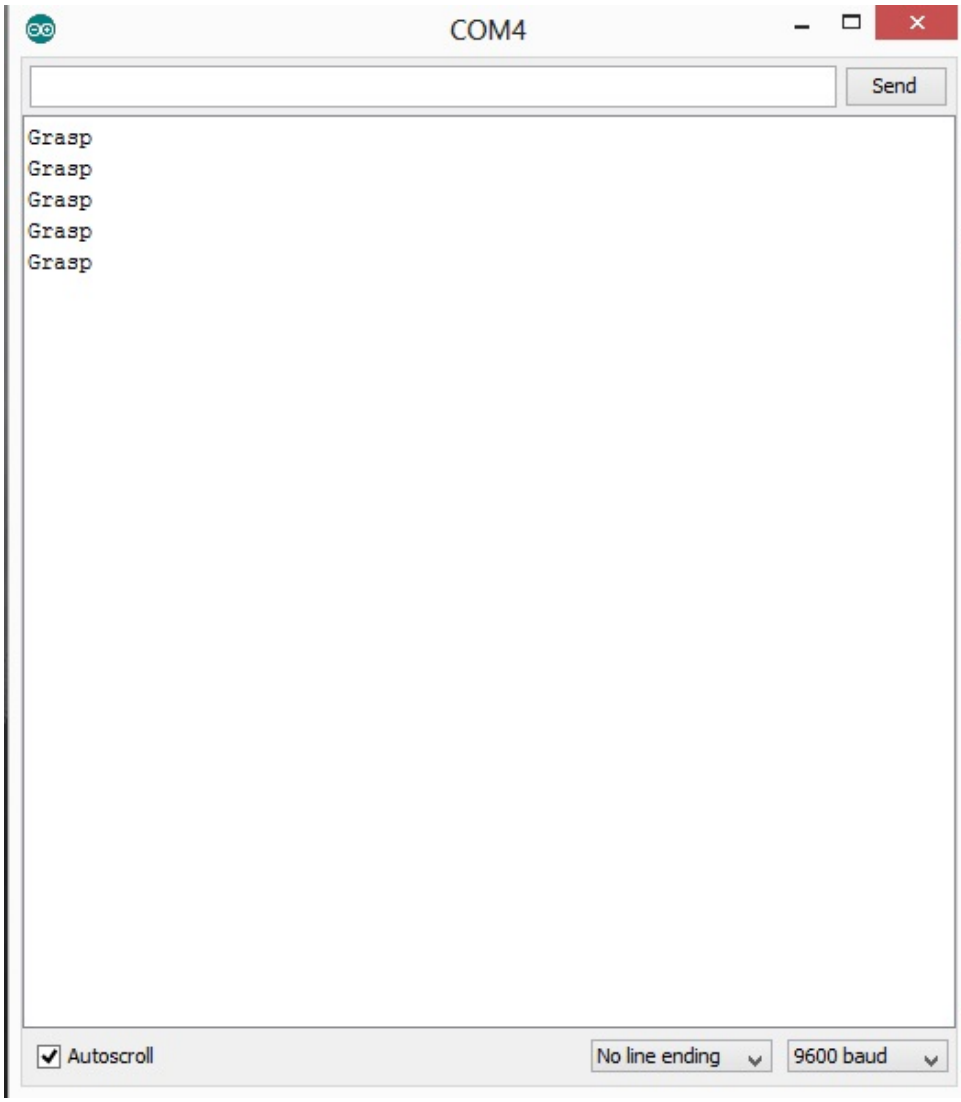
Rock On Gesture:





Grasp Gesture: (All Fingers Curled)





For the Result for the Gaming Portion Please view the available video in the Video Folder.

Chapter

8

Conclusions/ Lessons Learnt

In conclusion the project worked out successfully and I was able to implement the hand detection precisely to obtain change of 10 degree bends.

A requirement for auto-calibration is required though for this system which cannot be implemented in code as I realized.

Future Development

I plan to add accelerometer and gyroscope to Hand detection assembly so it can detect the hand movement in 3-Dimensions.

My second objective would be to use a controller which has inbuilt Analog to Digital Converter and also USB plug and play interface so I can use the Hand movements to control a mouse pointer.

Chapter

9

Acknowledgements

I would like to thank the instructor, Prof. Linden McClure for designing such a rigorous course and answering my questions during the labs and the project creation.

I would also like to thank the TA's for all their help during the labs and their patience for answering my questions and debugging help in the labs.

To colleagues/friends who helped me when neither the instructor nor TA's were available I would like to extend my thanks during those graveyard shifts in the Labs.

I enjoyed the coursework very much and would recommend anyone interested in the embedded system field to take it up.

Finally I would like to thank the authors of the code and guides I based my project build.

References

LXDE information
http://lxde.org/
http://en.wikipedia.org/wiki/LXDE
Raspberry Pi Quick start Guide
http://www.raspberrypi.org/
http://www.raspberrypi.org/wp-content/uploads/2012/12/quick-start-guide-v1.1.pdf
Raspbian Wheezy
http://www.raspberrypi.org/downloads
GPIO pin control for the raspberry Pi
http://elinux.org/RPi_Low-level_peripherals

Chapter

10

Appendix

APPENDIX A)

Bill of Materials

Raspberry Pi	$45.00
Flex Resistors	$74.00
Resistors 22K	$00.00
Arduino Board	$75.00
Glove	$12.50
Pin Headers	$00.50
SD Card	$12.00
HDMI Cable	$06.00
---	---
Totals	$225.00

APPENDIX B)
Schematics

APPENDIX C)
Code

APPENDIX D)
Datasheets