STEM Capstone Notebook - Ben Mesnik and Matt Rubin

19
1
27

9/18/2015 - Where we are initially

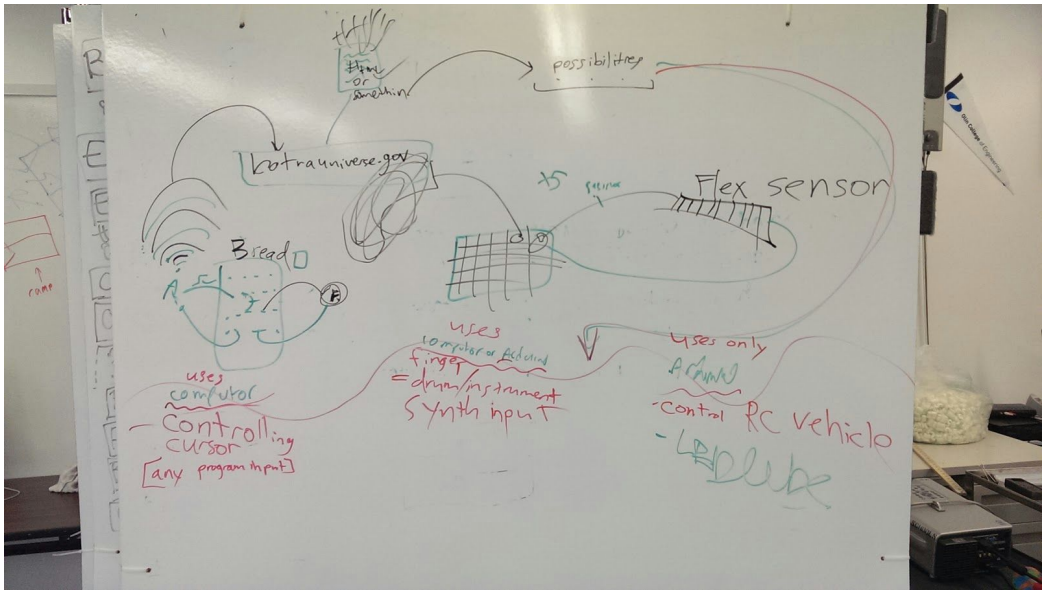
- preliminary setup: figuring out which materials to get
- we want to start initially with buying a single 4.5" (this length is long enough for measuring the flex of fingers) flex sensor to test with any arduino board
    - if/when we find success in our testing, we can purchase additional (4) flex sensors (5 in total, for all fingers on a hand), as well as the *Arduino Yún,* which has built-in capability to connect to the internet via Wi-Fi
        - we want to connect to Wi-Fi so we can wirelessly send the flex sensor data (it would be limiting if the glove had to be connected to a computer with wires, as you'd have to stay very close to the computer)

9/21/2015 - **potential uses for finished hand flex sensor device**

Ideas that involve our device using a computer, Arduino, or both to control something

Computer	**Arduino**	**Both/either**
controlling cursor	control RC vehicle	finger drum/other instrument synthesizer input
educational-type counting games for children	TV remote	LED/computer program for displaying amount of flex (*could be used for testing)*
	model hand, mimics finger bending	

(add list of *unknowns* for our creation)



9/24/2015 - What we have so far; *Arduino Test Code*

- we acquired one 4.5” flex sensor
- we wrote an Arduino program to display the amount of flex, but we don’t yet have an arduino, breadboard, or wires
- the code is below:

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int flexposition = analogRead(/*  whatever pin # we use */);
  Serial.print("Flex amount:");
  Serial.print(flexpostion);
  delay(20);
}
```

9/25/2015 - Idea about other applications of our device (beyond measuring finger bend)

- **Thought:** think of other applications for flex sensor; could have more uses than measuring finger flex
- could also measure flex of anything that bends
    - e.g. our wireless Arduino Yun device with a flex sensor could be attached to certain part/hardware that may change shape (bend or straighten) over time or due to a certain cause, and our device would detect when this part has deformed and it would then run a program and alert somebody

    - for example, maybe some kind of pipe or other part hidden behind a wall or someplace where people don't normally see it and where they wouldn't constantly want to check; our device would take care of the problem of being alert of when a part needs attention
    - multiple copies of our device could be placed on numerous parts in a building, for example, and they would alert people with their location in the building too

9/29/2015 - Initial Flex Testing

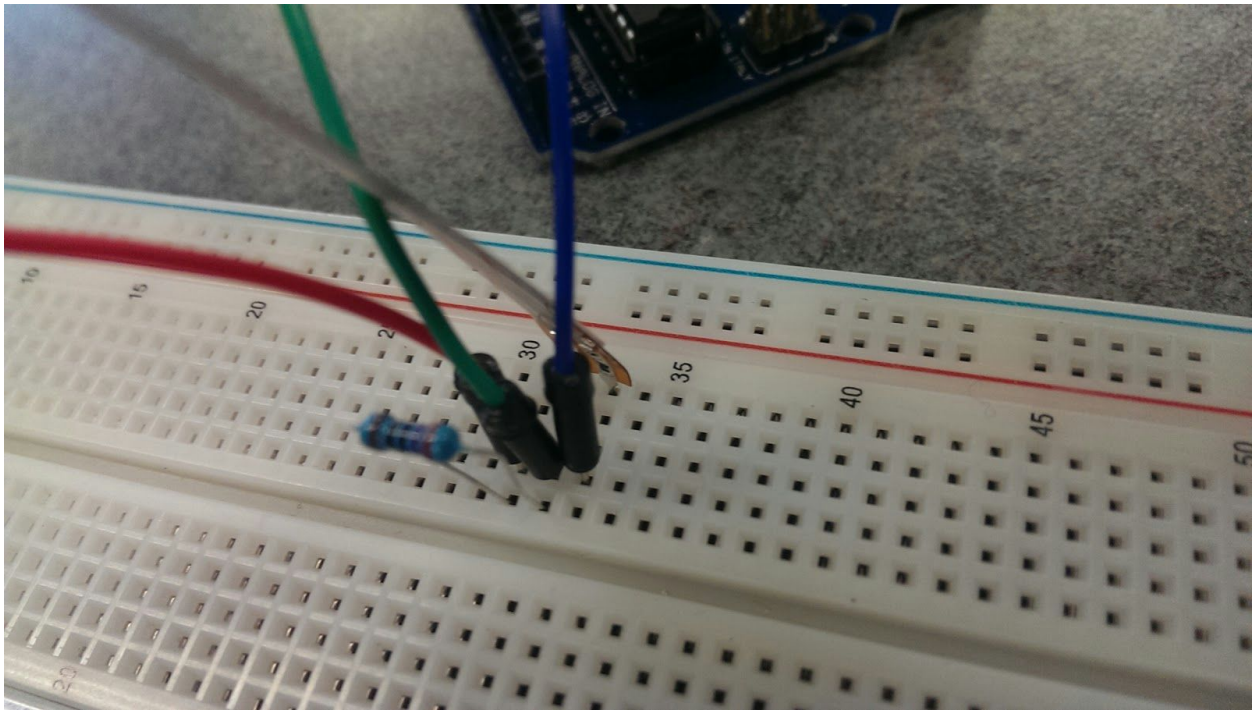
- we have now acquired an Arduino Uno, a resistor (270 ohms, 1% tolerance), some wires, and a breadboard
- the code we used to test it:

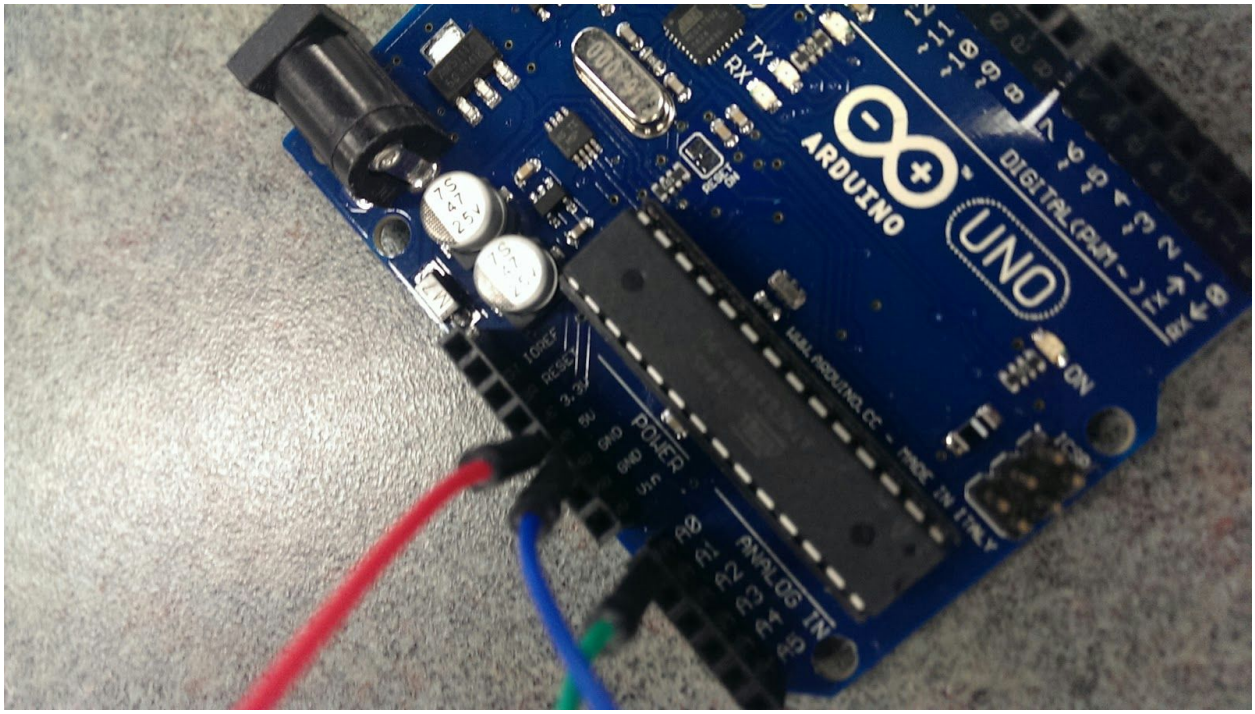
```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int flexposition = analogRead(14) - 1000;
  Serial.print("Flex amount:");
  Serial.print(flexpostion);
  delay(500);
}
```

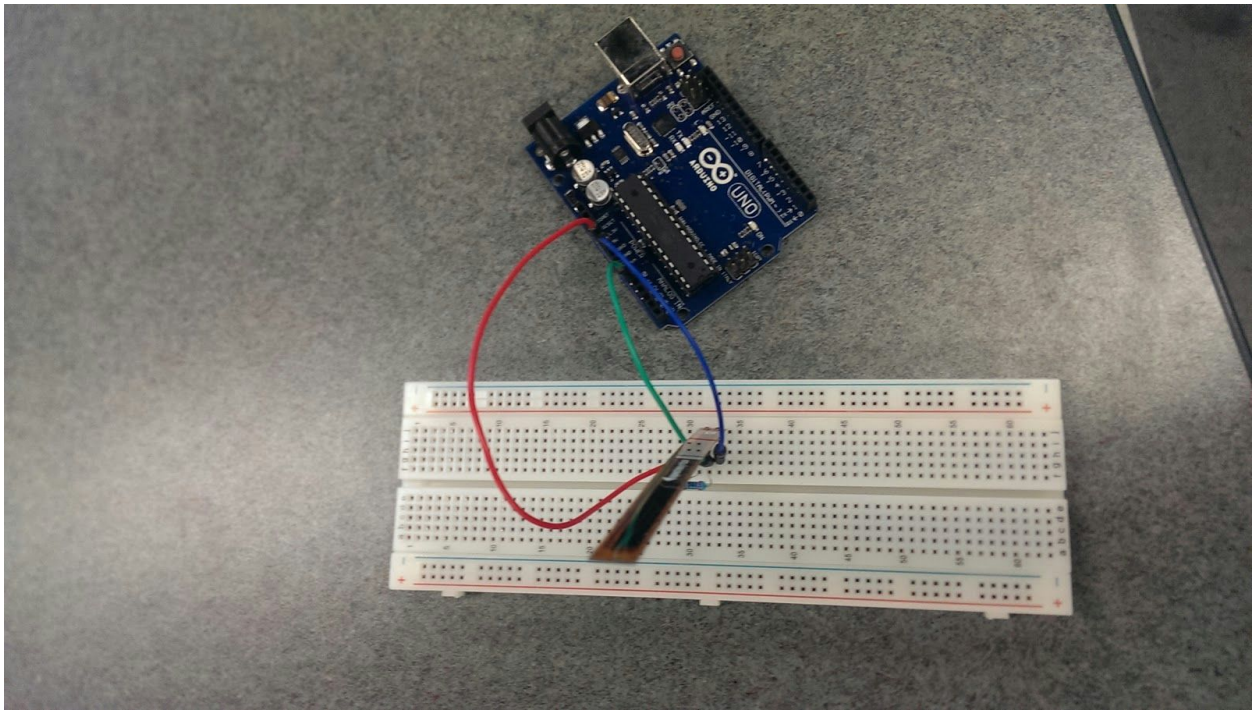
- what we need to advance:
    - glove
    - Arduino Yun (because this one has built-in WiFi capability)
    - more resistors and flex sensors (4 more)
    - micro SD card for Arduino Yun (it doesn't come with a lot of onboard memory)

9/30/2015 - Our upcoming plans

- we are currently waiting for the parts that we ordered to arrive
- meanwhile, we will research about creating javascript or html websites (to receive the flex data from the Arduino wirelessly)
- once we have our parts, especially the Arduino Yun, we will start programming it to connect to the computer wirelessly and we can start building our flex sensor circuits




ARDUINO
UNO
TX
RX
POWER
ANALOG IN
DIGITAL (PWM~)

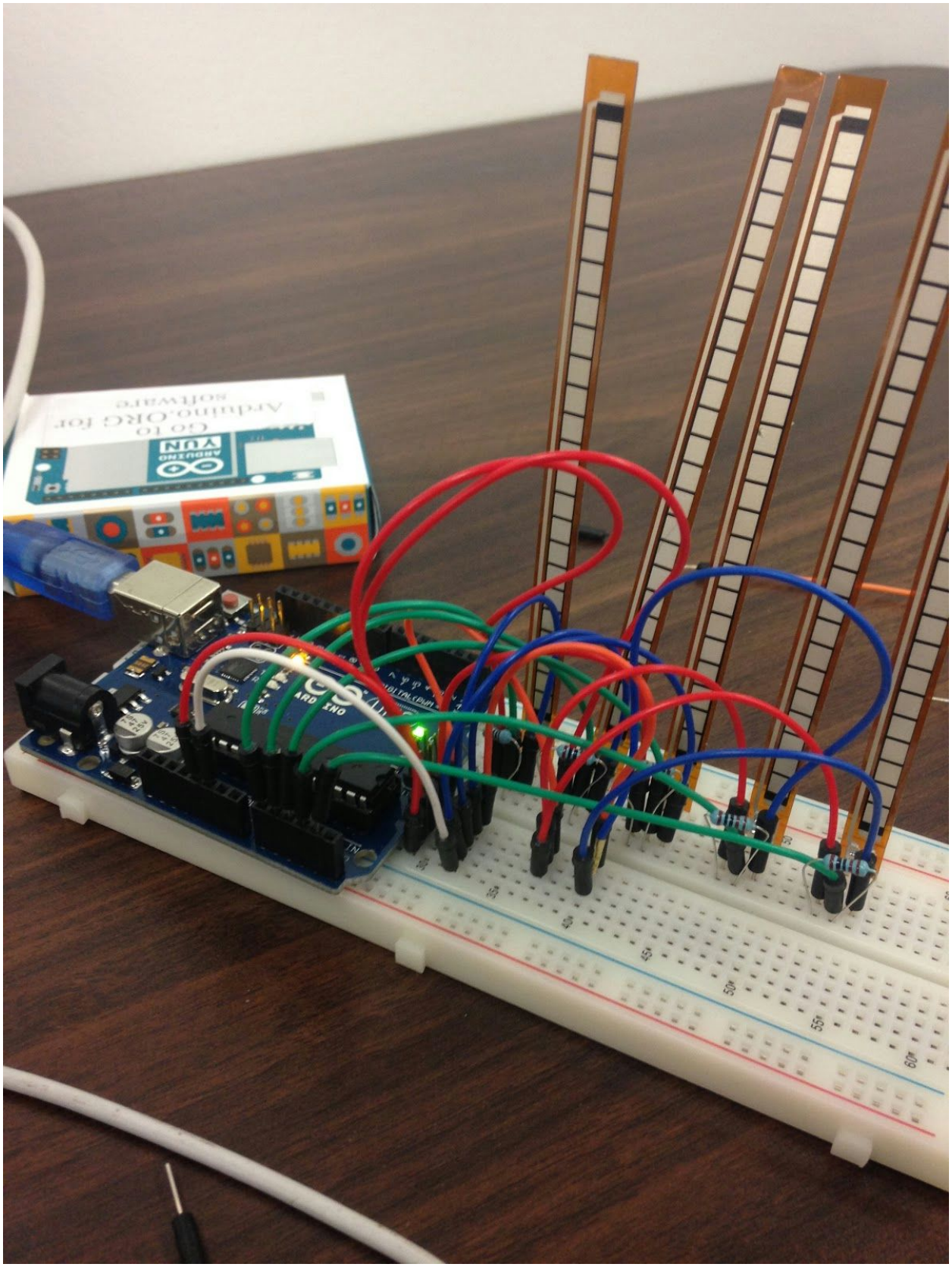

ARDUINO
UNO

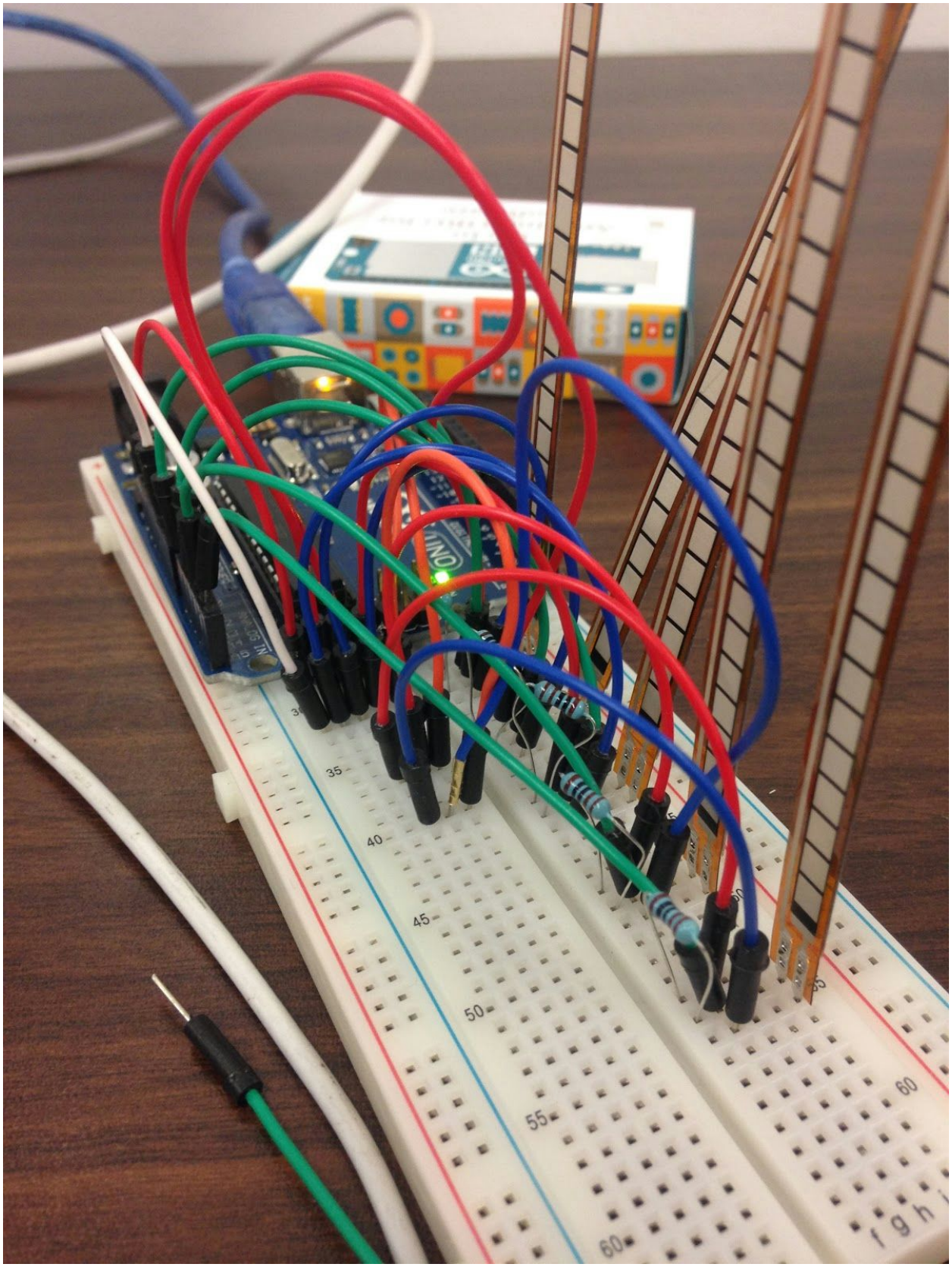
10/6/2015 - updated Arduino code

- the code:

```
int flexpositions[5] = {0,0,0,0,0};
int neutrals[5] = {0, 0, 0, 0, 0};     //neutral flex measurements for each sensor

void setup() {
  Serial.begin(9600);
  setNeutrals();     //initially, flex sensors should be all at 180 degrees to calibrate their values
}

void loop() {
  for(int i = 0; i < 5; i++)
  {
    flexpositions[i] = analogRead(i + 14) - neutrals[i];
  }
  for(int i = 0; i < 5; i ++)
  {
    Serial.print("Flex ");
    Serial.print(i+1);
    Serial.print(" : ");
    Serial.println(flexpositions[i]);
  }
  Serial.println();
  delay(1500);
}
void setNeutrals(){     //can be called anytime to reset what are considered neutral positions
  for(int i = 0; i < 5; i ++){
    neutrals[i] = analogRead(i + 14);
  }
}
```

- supports the input of all flex sensors at once (in an array)
- current photos of the breadboard setup below:


Go to
Arduino.ORG for
software
ARDUINO
YUN


35
40
45
50
55
60

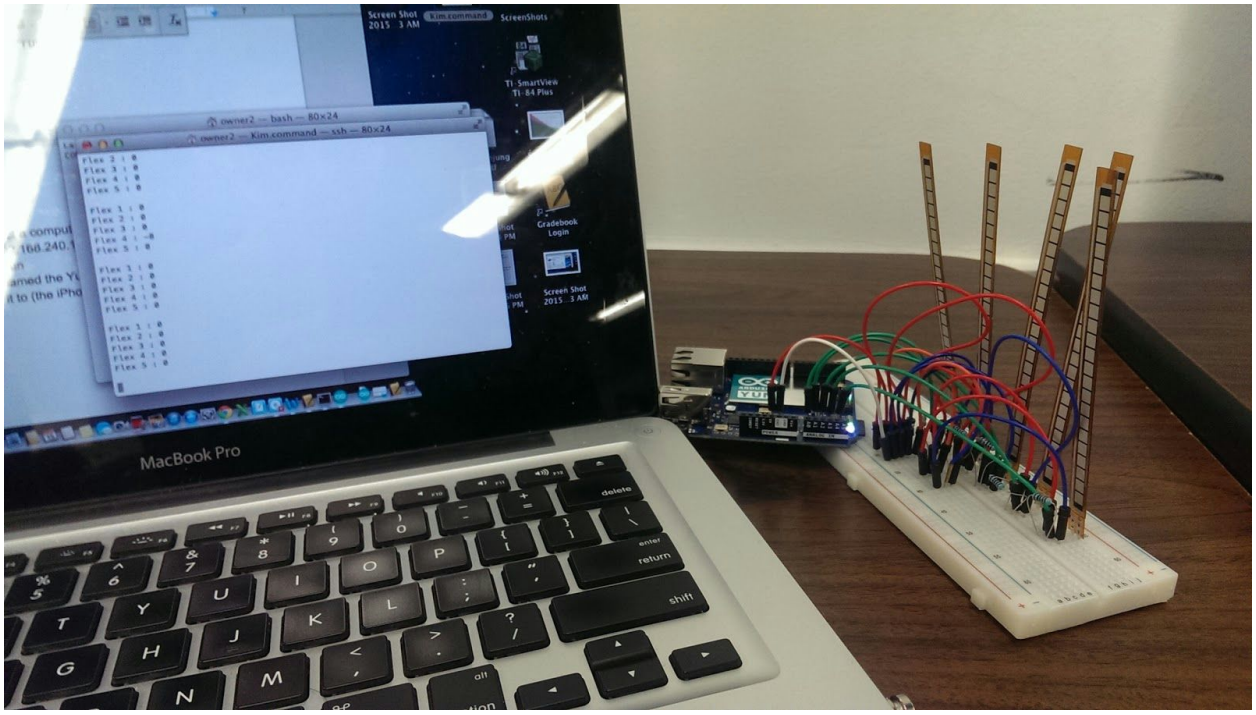
10/15/2015 - Progress Update

- we figured out we can't use the school's "NPS-Guest" wifi network to connect our Arduino Yun to the computer because devices cannot communicate with each other over a guest wifi network
    - the workaround we found was to use one of our cell phones as a wifi hotspot; that way, it is our own network over which we have complete control/ownership
- we have updated our code to read in the flex values for the Arduino Yun:

```
int flexpositions[5] = {0,0,0,0,0};
int neutrals[5] = {0, 0, 0, 0, 0};     //neutral flex measurements for each sensor
#include <Console.h>
void setup() {
 Bridge.begin();
  Console.begin();
  setNeutrals();     //initially, flex sensors should be all at 180 degrees to calibrate their values
while (!Console);

  Console.println("yo");
}

void loop() {
  flexpositions[0] = analogRead(A0) - neutrals[0];
  flexpositions[1] = analogRead(A1) - neutrals[1];
  flexpositions[2] = analogRead(A2) - neutrals[2];
  flexpositions[3] = analogRead(A3) - neutrals[3];
  flexpositions[4] = analogRead(A4) - neutrals[4];

  for(int i = 0; i <5; i ++){
    if(abs(flexpositions[i]) < 8)
      flexpositions[i] = 0;     //if flex sensor is approximately 180 degrees, set it back to 0 (for consistency)
  }

  for(int i = 0; i < 5; i ++)
  {

    Console.print("Flex ");
    Console.print(i+1);
    Console.print(" : ");
    Console.println(flexpositions[i]);
  }
  Console.println();
  delay(1500);
}
void setNeutrals(){     //can be called anytime to reset what are considered neutral positions
  neutrals[0] = analogRead(A0);
  neutrals[1] = analogRead(A1);
  neutrals[2] = analogRead(A2);
  neutrals[3] = analogRead(A3);
  neutrals[4] = analogRead(A4);
}
```

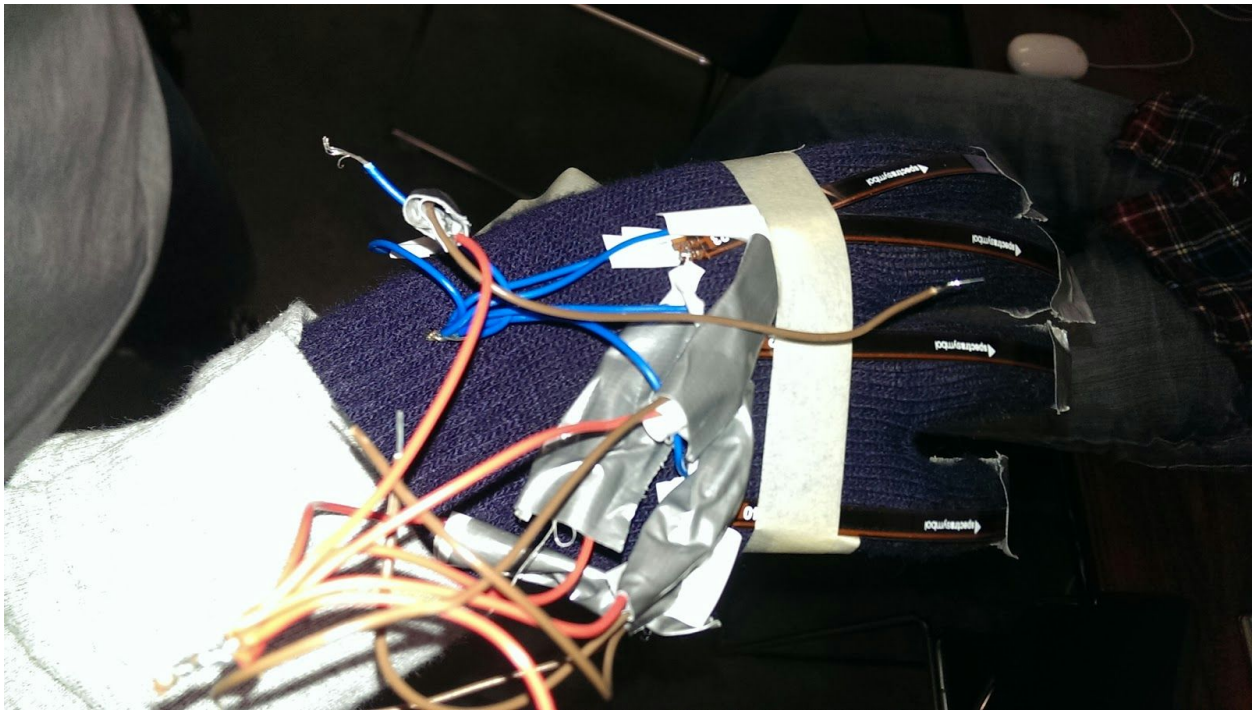
- in order to view the output of the Arduino Yun wirelessly on a computer that's on the same wifi network as the Yun, we enter the following command into the Terminal command line
    - ssh root@kimjungyun.local 'telnet localhost 6571' ["kimjungyun" is what we named our Arduino]

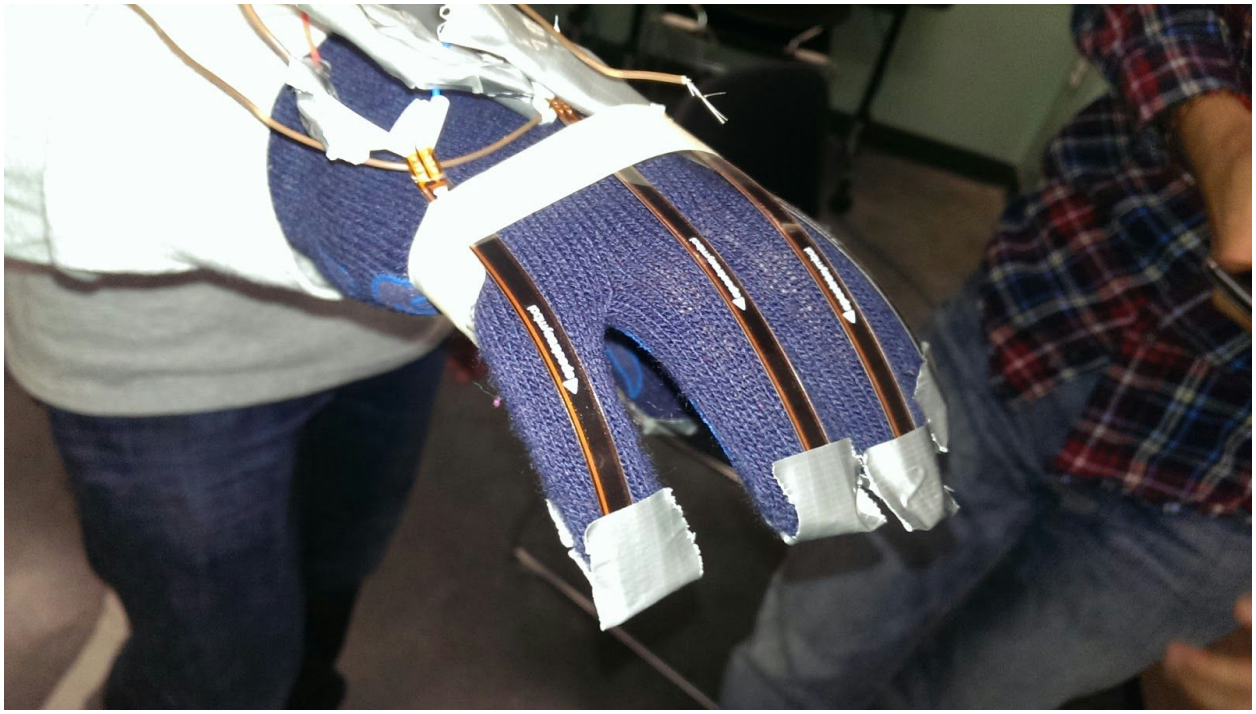
    - we then enter the password we chose for the Yun
- to initially configure the Yun, we had to connect a computer to the Yun as a wifi network
    - then, we opened up the IP address 192.168.240.1 into a browser to get to the configuration screen for the Arduino Yun
    - next, on this configuration page, we named the Yun and chose a password, and chose which wifi network to connect it to (the iPhone wifi hotspot)



10/21/2015

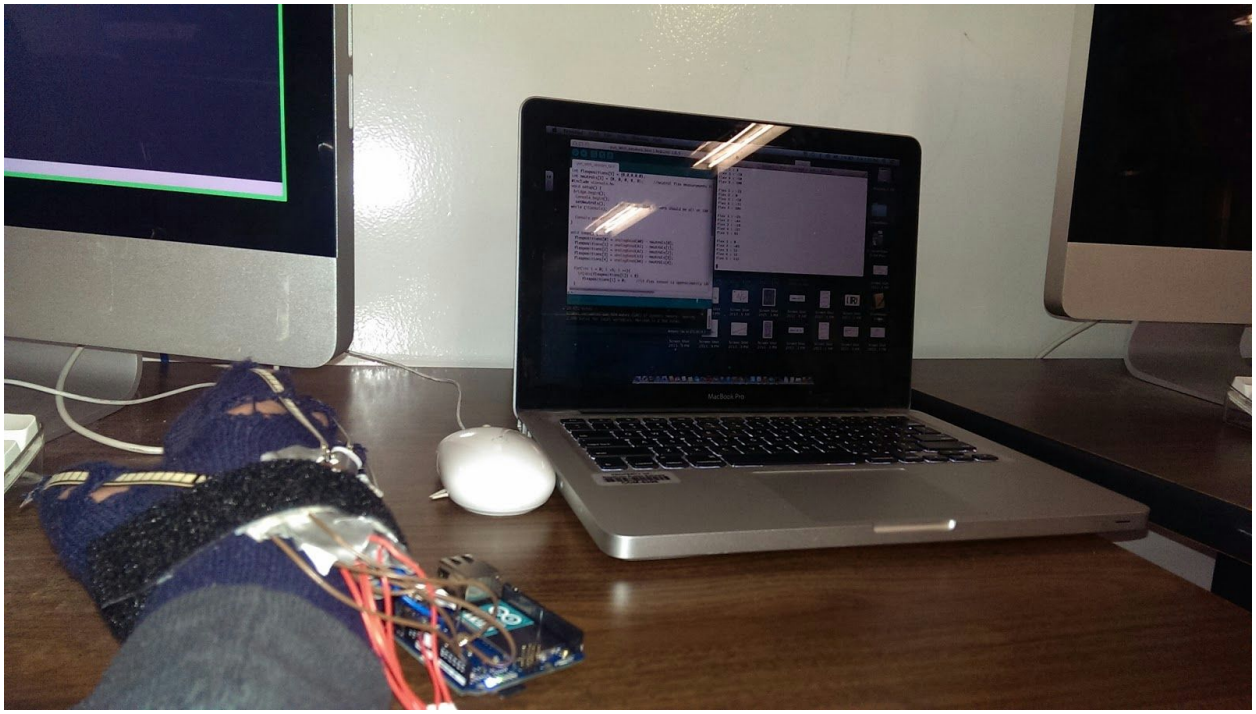
- We have recreated our 5 flex sensor circuit with wires that we soldered together off of a breadboard
- Then we added insulated electrical tape to the circuit
- Using duct and masking tape, we attached the circuit with all the sensors onto a glove
- Photos:

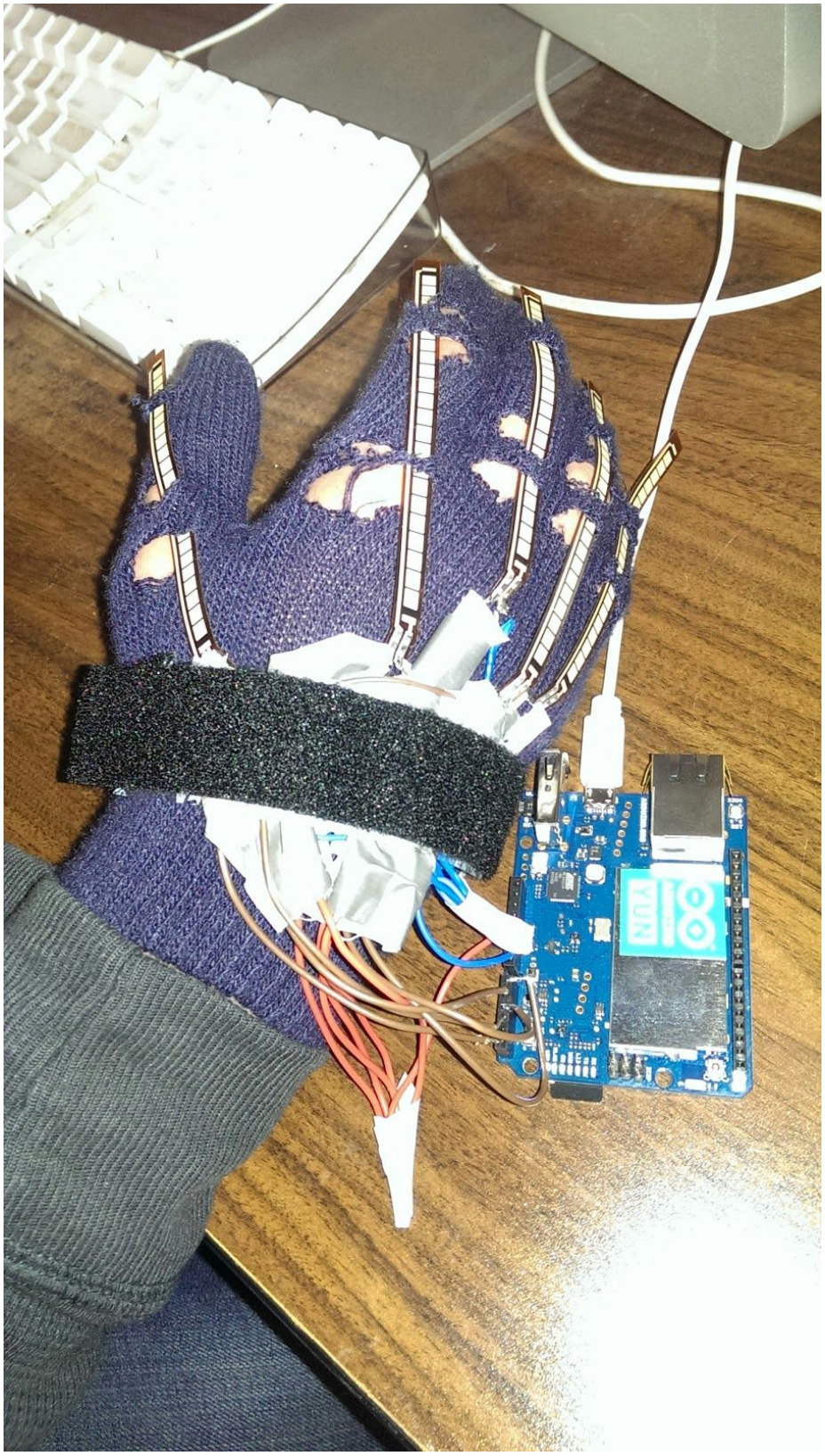




10/27/2015 - new glove prototype

- instead of relying on tape to attach the flex sensors and the wires to the glove, we have used velcro to attach the wires to the glove (the velcro also acts as a wrist strap) and we have cut holes in the glove and inserted the flex sensors through these holes
- this glove that we used, as pictured below, kind of got destroyed, and it's a bit too small and not a great glove to use for our final design
    - we will have to acquire a better glove




NINA

11/16/2015 - Test Program for sending data from arduino flex sensors to html page accessible by devices connected to same network as the arduino

```
/*
  variation of the built-in Arduino program, Temperature webpanel

 http://www.arduino.cc/en/Tutorial/TemperatureWebPanel

 */

#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>
#include <Console.h>

// Listen on default port 5555, the webserver on the Yún
// will forward there all the HTTP requests for us.
YunServer server;
String startString;
long hits = 0;

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

  // Bridge startup
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  Bridge.begin();
  Console.begin();
  while(!Console);

  digitalWrite(13, HIGH);

  // using A0 and A2 as vcc and gnd for the TMP36 sensor:
  pinMode(A1, INPUT);
  //pinMode(A2, OUTPUT);
  //digitalWrite(A0, HIGH);
  //digitalWrite(A2, LOW);

  // Listen for incoming connection only from localhost
  // (no one from the external network could connect)
  server.listenOnLocalhost();
  server.begin();



  // get the time that this sketch started:
  Process startTime;
  startTime.runShellCommand("date");
  while (startTime.available()) {
    char c = startTime.read();
    startString += c;
  }
}

void loop() {
  // Get clients coming from server
  YunClient client = server.accept();

  // There is a new client?

  if (client.connected()) { Console.print("We are connected to the client.");
    // read the command
    String command = client.readString();
```

```
    command.trim();        //kill whitespace
    Console.print(command);
    // is "temperature" command?
    if (0==0){//command == "temperature") {

      // get the time from the server:
      Process time;
      time.runShellCommand("date");
      String timeString = "";
      while (time.available()) {
        char c = time.read();
        timeString += c;
      }
      Console.print(timeString);
      int sensorValue = analogRead(A1);
      // convert the reading to millivolts:
      float voltage = sensorValue * (5000.0f / 1024.0f);
      // convert the millivolts to temperature celsius:
      float temperature = (voltage - 500.0f) / 10.0f;
      // print the temperature:
      client.print("Current time on the Y&uacute;n: ");
      client.println(timeString);
      client.print("<br>Current temperature: ");
      client.print(sensorValue);//temperature);
      client.print(" &deg;C");
      client.print("<br>This sketch has been running since ");
      client.print(startString);
      client.print("<br>Hits so far: ");
      client.print(hits);
    }

    // Close connection and free resources.
    client.stop();
    hits++;
  }

  delay(50); // Poll every 50ms
}
```

11/20/2015 - Updated Arduino code and HTML Webpage code & more

- *How the micro SD card in the Arduino Yun must be formatted*
    - Folder structure, starting from root folder of sd card:
        - arduino > www > datatohtml
    - the datatohtml folder contains two files:
        - index.html → sets up the webpage to which the Yun sends data
        - zepto.min.js → a file containing a javascript library used to make it simple to load the Yun's content onto a browser

Arduino code (datatohtml.ino):

```
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>

// Listen on default port 5555, the webserver on the Yún
// will forward there all the HTTP requests for us.
YunServer server;
```

```
String startString;
long hits = 0;

int flexpositions[5] = {0, 0, 0, 0, 0};
int neutrals[5] = {0, 0, 0, 0, 0};     //neutral flex measurements for each sensor

void setup() {

  // Bridge startup
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  Bridge.begin();

  setNeutrals();     //initially, flex sensors should be all at 180 degrees to calibrate their values

  digitalWrite(13, HIGH);

  // Listen for incoming connection only from localhost
  // (no one from the external network could connect)
  server.listenOnLocalhost();
  server.begin();

  // get the time that this sketch started:
  Process startTime;
  startTime.runShellCommand("date");
  while (startTime.available()) {
    char c = startTime.read();
    startString += c;
  }
}

void loop() {
  flexpositions[0] = analogRead(A0) - neutrals[0];
  flexpositions[1] = analogRead(A1) - neutrals[1];
  flexpositions[2] = analogRead(A2) - neutrals[2];
  flexpositions[3] = analogRead(A3) - neutrals[3];
  flexpositions[4] = analogRead(A4) - neutrals[4];

  // Get clients coming from server
  YunClient client = server.accept();

  // There is a new client?
  if (client.connected()) {
    // read the command
    String command = client.readString();
    command.trim();        //kill whitespace

    // get the time from the server:
    Process time;
    time.runShellCommand("date");
    String timeString = "";
    while (time.available()) {
      char c = time.read();
      timeString += c;
    }

    int sensorValue = analogRead(A0);
```

```
    client.print("Current time on the Y&uacute;n: ");
    client.println(timeString);
    client.print("<br>This sketch has been running since ");
    client.print(startString);
    client.print("<br>Hits so far: ");
    client.println(hits);

    //print sensor readings
    for (int i = 0; i < 5; i ++) {
      if (abs(flexpositions[i]) < 8)
        flexpositions[i] = 0;     //if flex sensor is approximately 180 degrees, set it back to 0 (for consistency)
    }
    client.print("<br>Flex 1: ");
    client.println(flexpositions[0]);
    client.print("<br>Flex 2: ");
    client.println(flexpositions[1]);
    client.print("<br>Flex 3: ");
    client.println(flexpositions[2]);
    client.print("<br>Flex 4: ");
    client.println(flexpositions[3]);
    client.print("<br>Flex 5: ");
    client.println(flexpositions[4]);
    client.println();

    // Close connection and free resources.
    client.stop();
    hits++;
  }
  delay(50); // Poll every 50ms
}

void setNeutrals() {    //can be called anytime to reset what are considered neutral positions
  neutrals[0] = analogRead(A0);
  neutrals[1] = analogRead(A1);
  neutrals[2] = analogRead(A2);
  neutrals[3] = analogRead(A3);
  neutrals[4] = analogRead(A4);
}
```

HTML code (index.html):

```
<!DOCTYPE html>
<html>
        <head>
        <script type="text/javascript" src="zepto.min.js"></script>
          <script type="text/javascript">
                function refresh() {
                        $('#content').load('/arduino/datatohtml);
                }
    </script>
        </head>
        <body onload="setInterval(refresh, 2000);">
                <span id="content">Wating for Kim...</span>
        </body>
        <style type="text/css">
```

```
            body{
              background-color: rgb(200, 50, 100);
            }

          </style>
</html>
```