EE 106B Lab 4: Soft Robotics

Chris Correa and Valmik Prabhu

April 16, 2018

1 Goal

The goal of this lab to to become more familiar with soft robotics, and the advantages and limitations of using soft robotics, instead of rigid manipulators. In addition, this lab will provide hands-on experience with sensor calibration and microcontrollers.

2 Hardware

In this lab, we will be working with the Soft Finger (Figure 1), created by Amir Mousavi. It is a Silicone Finger, which expands on one side when air is passed into the internal chambers. It can wrap around objects when pressurized.

3 Logistics, Safety, Questions and Help

We expect that all students in this class are mature and experienced at working with hardware, so we give you much more leeway than in EECS 106A. Please live up to our expectations

3.1 Groups, Robots and Accounts

Remember that groups must be composed of 2-3 students, at least one of whom must have completed EECS 106A.Robots should be reserved on the robot calendar here. The rules are:

1. **Groups with a reservation have priority.** You can go in and use (or continue using) the hardware whenever you like if no one has a reservation. However, you must pass along the hardware once the next reservation starts. Please be respectful and try to plan ahead; it's not cool to take the first twenty minutes of another group's section winding down.
2. **Do not reserve more than two hours at a time.** This is shared hardware and we want to ensure that all groups have enough time to complete the lab.
3. **One computer per group (if needed)** We only have ten lab computers, and have around thirty students. If more than ten students want to use the computers, please try to limit yourself to one computer per group. In addition, groups with robot reservations have priority

Figure 1: Soft Finger

on the computers next to their respective robots. Groups can accomplish working in parallel by using personal computers, file-sharing software like git, remoting into the lab computers, and/or using simulators like Gazebo or RViz.

3.2 Safety

Remember to follow the lab safety rules:

1. **Never work on hardware alone.** Robots are potentially dangerous and very expensive, so you should always have someone on hand to help if something goes wrong. This person should be another 106B student,though in cases when group schedules are highly incompatible, you may talk to a TA to arrange an alternate solution, such as bringing a friend (note that you would be entirely responsible for this person's conduct).
2. **Do not leave the room while the robot is running.** Running robots must be under constant observation.
3. **Always operate the robot with the E-Stop within reach.** If Baxter is going to hit a person, the wall, or a table, press the E-Stop.
4. **Power off the robot when leaving.** Unless you're trading off with another group, the robots should be powered down for safety. To power on/off the robot, press the white button on the back of the robot.
5. **Terminate all processes before logging off.** Leaving processes running can disrupt other students, is highly inconsiderate, and is difficult to debug. Instead of logging off, you should type:

   ```
   killall -u <username>
   ```

   into your terminal, where <username> is your instructional account username (ee106b-xyz). You can also use ps -ef | grep ros to check your currently running processes.
6. **Do not modify robots without consulting lab staff.** Last semester we had problems with students losing gripper pieces and messing with turtlebots. This is inconsiderate and makes the TAs' lives much more difficult.
7. **Tell the course staff if you break something.** This is an instructional lab, and we expect a certain amount of wear and tear to occur. However, if we don't know something is broken, we cannot fix it.

3.3 Questions and Help

If you experience software-related errors, please perform the following:

1. Document exactly what you did leading up to the error (commands, terminal output, etc.)
2. Post on Piazza with a description of the error, the above materials, and a description of what you did to try to fix it.

Chris and Valmik **Will Not** diagnose any programming errors without the above documentation. However, feel freeto ask us theoretical questions on piazza, during office hours, or after lecture or discussion.

3.4 Arduino Specific Logistics / Safety

1. Don't set the pump to be higher than some threshold (approximately 100). This is to prevent damage to the finger. 100 should be enough to bend the finger completely.
2. Make sure to always be in reach of the "Output on/off" switch on the power supply. If something goes wrong, hit this button immediately.
3. Be careful especially with the tube connecting to the finger. It is possible to puncture one of the internal chambers if you are too rough with the tube. Also be careful with the wires on the flex sensor.

4 Project Tasks

You'll be using the soft robotics circuit and finger in Cory 111 to solve the following tasks:

1. Calibrate the flex sensor and pressure sensor.
2. Create two separate PID controllers for the finger angle with the following sensors as feedback:
   - Flex Sensor that measures bending
   - Pressure Sensor
3. Compare the two controllers in terms of performance and usefulness.

5 Deliverables

1. Videos of your implementation working. Provide a link to your video in your report.
2. Provide the code in a zip file, to be uploaded with your writeup.
3. Submit a detailed report with the following:
   (a) Describe your approach and methods in detail, including a formal description of the controllers you used.
   (b) Let us know what of any difficulties and how you overcame them.
   (c) Summarize your results in both words and with figures. In particular, show:
      - Show plots of the step response of your system.
      - Discuss any noise you see in the sensor data. From where do you think it comes? How much do you think it impacted your controller performance? If it did, how would you deal with it in the real world?
   (d) What are some examples in which soft fingers would work and rigid fingers would not, and vice versa.
   (e) Bring in an object or two which you think a rigid finger would not be able to pick up and record a video of the soft finger being able to wrap around it.
4. BONUS: In addition to your report, write a couple paragraphs on the diculties you had performing the lab, paying particular attention to how the lab documentation could be improved. This course is evolving quickly, and we're always looking for feedback.

6 Working with Arduino

1. Plug the Arduino into the lab machine with the USB cable, and connect the breadboard to the power supply.
2. When you open the Arduino IDE, you'll need to select the Arduino Mega board under the "Tools" tab (Figure 2), and select the serial port corresponding to the arduino (Figure 3). (These may look slightly different depending on the port).
3. Write your code
4. To run your code on the arduino, click the check button then the arrow (Figure 4). The check button compiles and verifies your code, and the arrow flashes the code to the board.

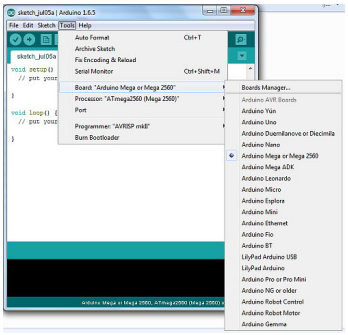

Figure 2: Arduino Mega Board

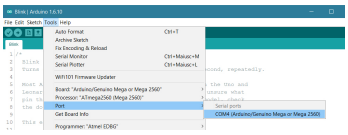

Figure 3: Choosing the Right Port

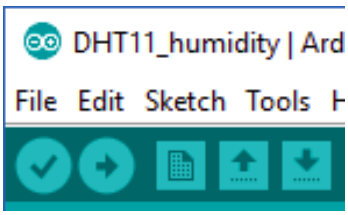

Figure 4: Choosing the Right Port

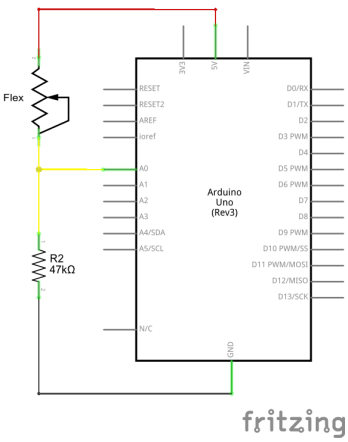

Figure 5: Flex Sensor

7 Flex Sensor

Every time you get on the system, ensure the circuit looks like Figure 5.

```
/*
    ******************************************************************************
Flex_Sensor_Example.ino
Example sketch for SparkFun's flex sensors
  (https://www.sparkfun.com/products/10264)
Jim Lindblom @ SparkFun Electronics
April 28, 2016

Create a voltage divider circuit combining a flex sensor with a 47k resistor.
- The resistor should connect from A0 to GND.
- The flex sensor should connect from A0 to 3.3V
As the resistance of the flex sensor increases (meaning it's being bent), the
voltage at A0 should decrease.

Development environment specifics:
Arduino 1.6.7
******************************************************************************/
const int FLEX_PIN = A0; // Pin connected to voltage divider output

// Measure the voltage at 5V and the actual resistance of your
// 47k resistor, and enter them below:
const float VCC = 4.98; // Measured voltage of Ardunio 5V line
const float R_DIV = 47500.0; // Measured resistance of 3.3k resistor

// Upload the code, then try to adjust these values to more
// accurately calculate bend degree.
const float STRAIGHT_RESISTANCE = 37300.0; // resistance when straight
const float BEND_RESISTANCE = 90000.0; // resistance at 90 deg

void setup()
{
  Serial.begin(9600);
  pinMode(FLEX_PIN, INPUT);
}

void loop()
{
  // Read the ADC, and calculate voltage and resistance from it
  int flexADC = analogRead(FLEX_PIN);
  float flexV = flexADC * VCC / 1023.0;
  float flexR = R_DIV * (VCC / flexV - 1.0);
  Serial.println("Resistance: " + String(flexR) + " ohms");

  // Use the calculated resistance to estimate the sensor's
  // bend angle:
  float angle = map(flexR, STRAIGHT_RESISTANCE, BEND_RESISTANCE,
                    0, 90.0);
  Serial.println("Bend: " + String(angle) + " degrees");
  Serial.println();

  delay(500);
```

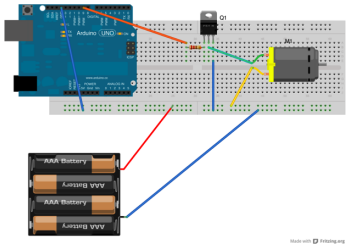

Figure 6: Choosing the Right Port

```
51 }
```

This is some starter code on how to read in the data from the flex sensor. You will need to calibrate the values for STRAIGHT_RESISTANCE and BEND_RESISTANCE. To do this, check what value is printed out when it is straight, and record that as STRAIGHT_RESISTANCE. Bend the sensor and do the same for BEND_RESISTANCE.

The flex sensor can sometimes output sporadic values. You may want to do some temporal averaging.

8 Pressure Sensor

Like the flex sensor, make sure the circuit looks like this before starting the system:

```
1  int incomingByte = 0;
2  #define sensor A0           // Pressure sensor
3
4  void setup(){
5    Serial.begin(9600);    // opens serial port, sets data rate to 9600
       bps
6  }
7
8  void loop(){
9    Serial.println(analogRead(sensor)); // prints pressure sensor value
10   delay(1000);
11 }
```

9 Pump

The pump acts exactly like a DC motor. Like the flex sensor, make sure the circuit looks like Figure 6 before starting the system.

```
1  int incomingByte = 0;
2
3  void setup(){
4    pinMode(3, OUTPUT);    // set mosfet gate pin to an output
5    Serial.begin(9600);    // opens serial port, sets data rate to 9600
       bps
6  }
7
8  void loop(){
9
10   if (Serial.available() > 0)
11   {
12   //read the incoming entered byte:
13   incomingByte = Serial.parseInt();
14
15   Serial.print("I␣received:␣");
```

```
16    Serial.println(incomingByte);
17              incomingByte = min(incomingByte, 100);
18    analogWrite(3, incomingByte); //write the user entered value(0-254)
          to pwm outdput
19
20    }
21    delay(1000); //You may change this to be whatever frequency you want
22  }
```

10 Scoring

For each task listed in Section 5, you'll be graded on a 0-5 score as explained in the table below.

Table 1: Grading Rubric for Lab 4

Score	Meaning
0	Did not attempt
1	Attempted but missing more than three deliverables
2	Complete with major error or missing two or three deliverables
3	Complete with moderate error or missing one deliverable
4	Complete with minor errors
5	All tasks completed satisfactorily with deliverables

Table 2: Point Allocation for Lab 4

Section	Points
Video	5
Code	5
Methods	5
Flex Sensor	5
Pressure Sensor	5
Results and Sensor Comparison	10
Extra Questions	5
Bonus Difficulties Section	5*

11 Submission

You'll submit your writeup(s) on Gradescope and your code through bCourses as a single .zip file. Your code will be checked for plagiarism, so please submit your own work. Only one submission is needed per group.