Carnegie Mellon University

Individual Lab Report #1

Author:
Yuqing Qin

Team C: COBORG
Gerald D'Ascoli | Jonathan Lord-Fonda | Yuqing Qin | Husam Wadi
Feng Xiang

February 25, 2021

Table of Content

1 Individual Process

1.1 Sensors and Motors Lab

In the sensor and motor lab, my work mainly focused on designing simple circuits and developing microcontroller codes for sensors, specifically the flex sensor and IR distance sensor. The flex sensor is a resistance sensor whose value depends on the bending angle. The IR sensor outputs different voltage values based on the distance to the object. I also integrated the code for the flex sensor and servo motor so that bending the flex sensor will control the movement of the servo motor.

1.1.1 Sensor

The flex sensor I used in the lab is SEN-10264, which is a resistance sensor. Its working mechanism is shown in Figure 1. As the sensor is bent, the resistance value for the sensor increases. From the datasheet, I knew that the flat resistance for this sensor is 25kOhms. To test the real bending resistance range, I designed a voltage divider circuit with a resistor (R=10kOhms) in series. The power supply is 5V from the microcontroller (Arduino Uno) and the output voltage for the flex sensor is connected to the analog pin of Arduino. The value reading from the analog pin is from 700 to 860 when I bend the sensor from 0 to 90 degrees. As with the 10-bit microcontroller settings, the corresponding voltage for the sensor is from 3.41V to 4.20V. Therefore, the real resistance range for this flex sensor is from 21kOhms to 53kOhms.

Also, the output voltage from the flex sensor is noisy, so I added capacitors (C = 100nF) between each resistor and implemented a moving average filter from the analog readings to smooth the output. By testing on different window sizes for the average filter, I found the proper window size to smooth the output.

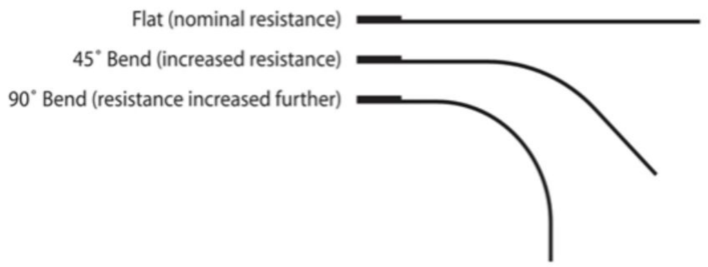
Flat (nominal resistance)
45˚ Bend (increased resistance)
90˚ Bend (resistance increased further)

Figure 1. Flex sensor resistance changing with angles

1.1.2 Motor

The motor I used for this lab is a servo motor. The servo motor has a built-in library in Arduino, which is called “Servo”. It allows me to directly write the moving degrees into the motor. The moving range of the servo motor is from 0 to 180 degrees. Therefore, I map the flex sensor output to the motor degrees. Moreover, the power for the servo motor is 5V, so I connect the power pin with the 5V pin on the Arduino to get enough power supply.

1.1.3 Circuit and Code

To prototype the circuit, I utilized TinkerCAD software to simulate it and later test it on a breadboard. The voltage divider circuit is shown in Figure 2. The output voltage for the flex sensor is connected to the analog pin A1 on Arduino, and the servo motor is connected to the digital pin 9 on Arduino. The resistor for the voltage divider is 10kOhms, and the capacitors to filter the noise are both 100nF.

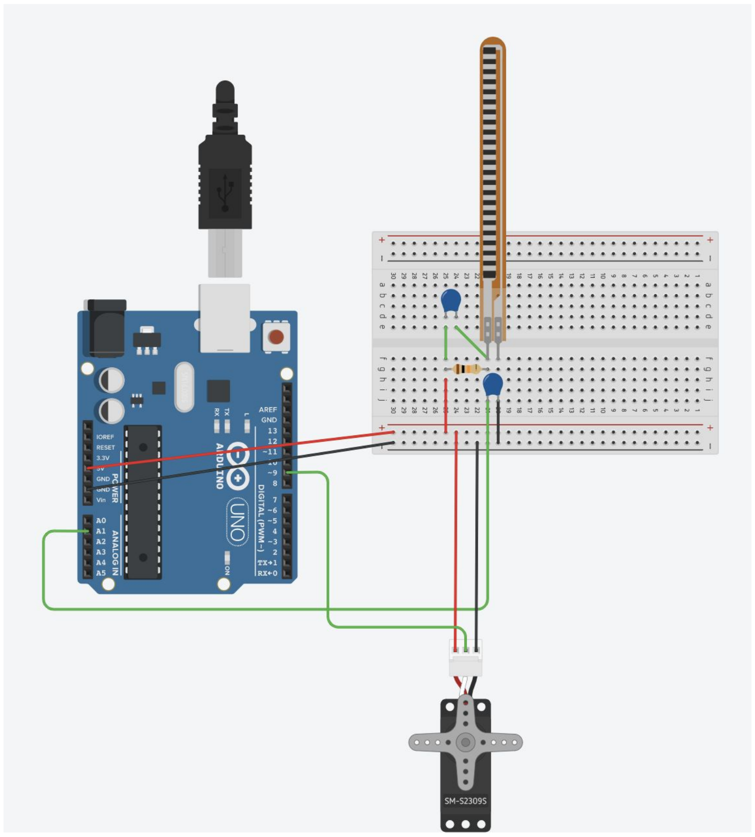

Figure 2. Circuit diagram

The code for the microcontroller is attached in Appendix A. It contains the code for reading flex sensor input and writing the mapped degree into the motor. Also, the moving average filter code is implemented in the code as well. I also worked on testing the IR sensor and converting the IR sensor output to the physical unit (cm). The code I contributed to is also shown in Appendix A.

1.2 MRSD Project

My work from the last semester mainly focuses on the perception subsystem implementation, specifically on the hand detection part. During the last few weeks, I worked on the YOLO v3 hand detection model setup and testing. I found someone has already trained the YOLO on several hand datasets, so I downloaded the pre-trained YOLO v3 model with weights and configuration files into my workspace. I tested on my CPU setting, the YOLO v3 tiny model works well and it is much faster than the YOLO v3 model. Therefore, we decided to use the tiny version of YOLO to do the hand detection.

Moreover, I launched the ROS node for the Intel Realsense D435i depth camera. I set up the camera and launched the ROS node for the camera. By checking the available topics connected to the node, I found the raw RGB image topics can be fed into the YOLO model and get detection results. Therefore, I am currently working on the ROS node build for YOLO and connecting the YOLO node to the camera node.

2 Challenges

2.1 Sensors and Motors Lab

During the lab, the main challenge I had is to filter the noise for the flex sensor. The noise will make the motor sensitive to the sensor output. I both applied capacitors between the resistors and also applied an average filter to the output. As a result, using two capacitors can effectively smooth the output, and an average filter can also help with the output.

2.2 MRSD Project

The problem I had for the past few weeks is about the running speed for YOLO v3. It does not fulfill the performance requirements. We set up the performance requirement for the run time within 5 seconds, but currently, the YOLO v3 running on the CPU is slow. I further tried different models, also tried YOLO v4 and YOLO v3 tiny. It turns out that the tiny version of YOLO is much faster even on CPU settings. At the same time, the accuracy of hand detection is good enough for our project.

3 Teamwork

3.1 Sensor and Motor Lab

Team Member	Sensor	Motor	Motor Lab Contribution
Feng	Potentiometer	Servo motor	Mapped potentiometer sensor analog output to servo-motor input. Collaborated with Yuqing to develop a control relationship between potentiometer and flex sensor with servo motor output. Helped combine code together to get arduino-only arduino code state.
Jonathan	Ultrasonic	Stepper motor	-Soldered DC motor controller board -Wrote switch debouncer code -Wrote code structure -Wrote program for stepper motor -Wrote program for ultrasonic sensor
Gerry	IR	Dc motor	-Wired DC motor and motor controller to each other and arduino -Debugged electrical issues -Wrote PID control of DC motor using encoder for state and IR sensor as input for desired state -Wrote program to control stepper motor position using the ultrasonic sensor as input.
Husam	GUI		-Created a ROS publisher and subscriber node in Arduino to send/receive data -Created a serial interface between ROS and the Arduino using “rosserial” -Created a custom message class “CMU” that handled sensor data outputs -Created a URDF that controlled the motor outputs through joint_states publisher -Created a visualization in RVIZ to show the GUI motor control output -Created a RQT GUI that contained RVIZ and 5 plots that displayed all the sensor

		outputs in real time

Table 1. Teamwork Contribution on Motor Lab

3.2 MRSD Project

Team Member	Work Description for COBORG
Feng	Developed pipeline between MoveIt and HEBI API and Coborg motors Developed URDF and initialized moveit to move robot in RViz
Jonathan	-Wrote the initial BOM for the current system -Created initial ROS node map -Sketched out main state node -Installed Linux/ROS and completed ROS training -Tested the arm's max lift at full extension -Reviewed Jason's 3-D model of COBORG
Gerry	Developed prototype of voice system. Generated a custom implementation of pocketsphinx on the CoBorg system and created custom libraries and recognition files to tailor the voice system to the CoBorg's specific requirements and functions.
Husam	-Manage and updating project timeline to adapt to the semester load -Used scrum/agile methodologies through kanban boards to map out weekly project work -Deepened ROS interconnectivity knowledge by researching state machines in ROS and going through ROS tutorial examples.

Table 2. Teamwork Contribution on COBORG

4 Plans

For the next few weeks, I will be mainly focusing on the ROS node (Goal_getter), which should align the bounding box information to the point cloud to obtain the 3D position of detected hands. Also, I will work on the tracking camera (Intel Realsense T265) and set up ROS nodes for it. Furthermore, I will combine it with the D435i Depth Camera to get the 3D position relative to the world frame.

Appendix A Code

```
#define WINDOW_SIZE 10
#include <Servo.h>

int flexPin = A1; // flex sensor
int IRpin = A2; // IR sensor

const int SERVO_PIN = 9;
Servo servo;

// average filtering
int readings[WINDOW_SIZE];
int sum = 0;
int index = 0;

int avgFilter(int reading)
{
  sum -= readings[index];  // remove the earliest reading
  readings[index] = reading;  // update the reading array
  sum += readings[index];   // add new reading to the sum
  index = (index+1) % WINDOW_SIZE;   // move the “index” pointer to the next
  return sum/WINDOW_SIZE;
}

void setup(void) {
  Serial.begin(9600);
  pinMode(SERVO_PIN,OUTPUT);
  servo.attach(SERVO_PIN);
  for (int i=0; i< WINDOW_SIZE; i++){
    readings[i] = 0;
  }

}

void loop(void) {
// IR sensor
//  int reading = analogRead(IRpin);
//  int average = avgFilter(reading);
//  Serial.println(average);
//  float voltage = (average/1024.0)*5;
//  float dist = 50.6/(voltage-0.173);   // cm
//  Serial.println(dist);
// Flex sensor  (resistance VS force)
   int val = analogRead(flexPin);  // 700-860 -> 3.45V-4.29V (0-1023 -> 0-5V) . R = 10k . Rx =
22k to 60k
  int avg = avgFilter(val);
  Serial.println(avg);
  float angle = map(val, 700, 860, 0, 180.0);
  servo.write(angle);

  delay(100);
}
```

Appendix B Quiz

1. Datasheet
   a. Sensor’s range: ±3 g,
      In the datasheet, “All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed”, so the range/span (based on “input signals outside this range are not guaranteed to meet the sensor’s specifications” in slide) for sensor is ±3 g
   b. Dynamic range/full scale: 6 g (guaranteed)
   c. Adding Cdc is to filter out the high-frequency noise from the power supply (decoupling). Capacitors will act as a low resistor when high-frequency noise comes so that it will allow the noise coming through the capacitor to smooth the power supply.
   d. Transfer function: Vout = 1.5V + 300mV/g * a
   e. largest expected nonlinearity error: 0.3% * 6g = 0.018g
   f. 25Hz noise: From the datasheet note

      $$rms\ Noise = Noise\ Density \times (\sqrt{BW \times 1.6})$$

      150*sqrt(25*1.6) = 0.00095g
   g. Not mention the 0Hz noise in the datasheet. The smallest bandwidth is 0.5Hz. We can connect the sensor to the Arduino, 0Hz means the sensor is kept the same stage (either no input or same input), so no sensor input and measure the changes from the Arduino output.

2. Signal Conditioning
   a. Filtering
      i. Moving average filter: 1. Sensitive to the large jump(outliers), will not entirely get rid of the outliers 2. Lag behind the real noisy data

      ii. Median filter: 1. Its computation cost is large because every time needs to sort the window 2. Sometimes may not be able to remove the noise(outliers) if have a lot of consecutive noises (wide outliers)

   b. Op-amp:
      i. -1.5V -> 0V , 1V -> 5V
         V1 is reference voltage, V2 is Vin, as a result:
         Vout = Vin*(1+Rf/Ri) -Vref(Rf/Ri)
         1+Rf/Ri = gain = (5-0)/(1-(-1.5)) = 2
         => Rf/Ri = 1
         => Vref = -3V

ii. -2.5V -> 0V , 2.5V -> 5V
If V1 is reference voltage, V2 is Vin, as a result:
Vout = Vin*(1+Rf/Ri) -Vref(Rf/Ri)
1+Rf/Ri = gain = (5-0)/(2.5-(-2.5)) = 1
=> Rf/Ri = 0
Not possible to calibrate this range to (0V,5V)

If v1 is VIn, V2 is reference voltage:
Vout = -Vin*(Rf/Ri) +Vref(1+Rf/Ri)
Rf/Ri = gain = 1
Not possible to find Vref to fulfill the equations

3. Control:
   a. For Kp, the output from DC encoder, will be subtracted to the desired position, use the error for position as the input for Kp
      For Kd, need to estimate the velocity of the motor, and subtract with the desired velocity, and use the error for velocity as the input for Kd
      For Ki, integrate the position error from start as the input for the Ki term.

   b. Sluggish: increase Kp to reduce rising time, quicker to get the desired position
   c. Steady-state error: increase Ki to make the steady-state error closer to 0
   d. Overshoot: Increase Kd to reduce the overshoot