Sensor and Motor Control Lab
Individual Lab Report

Bikramjot Singh Hanzra
Team B – Auto Pirates
Teammates – Tushar Chugh, Shiyu Dong, Tae-Hyung Kim, William Seto
ILR01

October 16, 2015

Contents

1 Individual Progress

I have divided my individual contributions into 2 subsections, one section is for Task 6 and the other is for the project in general.

1.1 Task 6

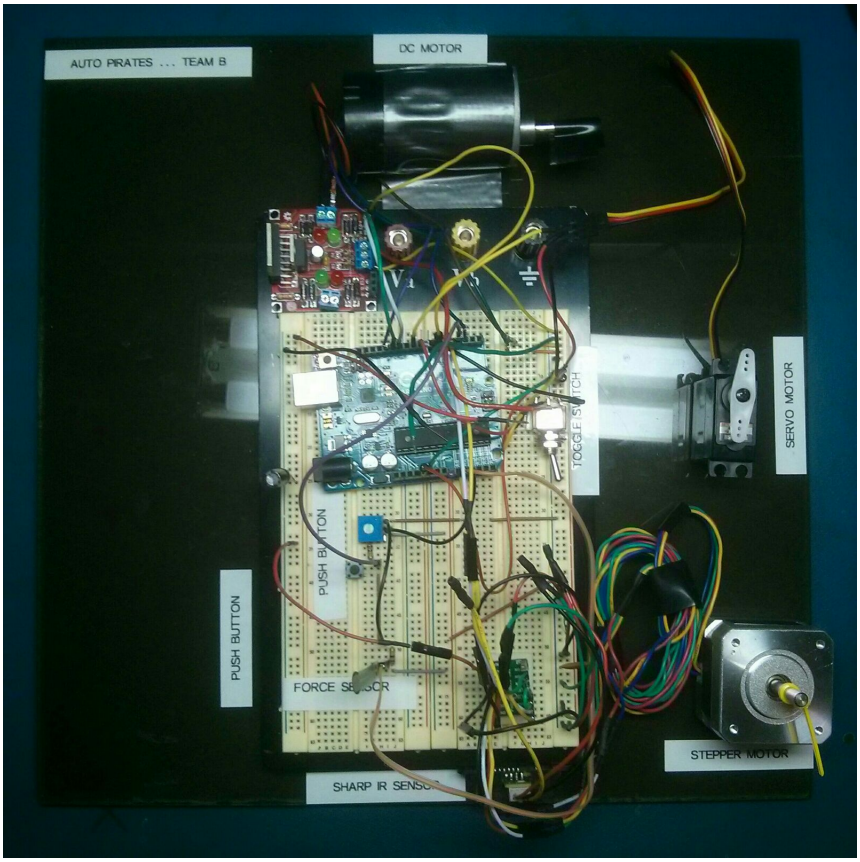

Figure 1: Sensors Lab Demonstration Setup

Figure 1 shows the physical connections of various sensors and motors to the Arduino Board. For the Task 6, my work was to -

- Write the code and setup the hardware for controlling the servo motor using the potentiometer reading.
- Write the code and setup the hardware for reading the state of the toggle switch.
- Write a MATLAB function to plot the graph of *ADC output vs Distance* for the Sharp IR sensor.

To control the servo, I used the Arduino Servo Library [1]. The task of controlling the servo motor is greatly simplified by using the Arduino Servo Library. The analog output of the potentiometer was connected to the `A2` analog pin of the Arduino board. In the code, the analog value of the

potentiometer is available in discrete values from 0 to 1023 and is needed to be mapped to a value between 0 to 180. The value between 0 and 180 is the angle at which the servo should rotate. The servo motor was connected to the pin 9 of the Arduino board.

The toggle switch was used to change the direction of rotation of the DC motor from clockwise to anti-clockwise direction and vice-versa. The toggle switch output was connected to pin 7 and pin 8 of the Arduino board. The code that I wrote for controlling servo motor and reading the state of the toggle switch is attached in Appendix 1.

The last task I did was plotting the graph of *ADC Output vs the Distance (in cm)* for Sharp IR sensor. The GUI dumped the data in a csv file which had 2 columns namely – *ADCOutput* and *Distance*. The MATLAB function (Appendix 2) reads the values from the CSV file and plots the graph.

In the Figure 2 shown below, the ADC output increases as the distance between between the object and the Sharp IR sensor decreases and vice versa. It can be clearly seen that the ADC output is a non-linear function of the distance. From the datasheet, we can infer the the output voltage varies from 0.3 ($10cm$) to 2.3 ($100cm$). Arduino board uses an ADC of 10 bits the voltage $(0 - 5)$ and the output is converted into discrete values from 0 to 1023.

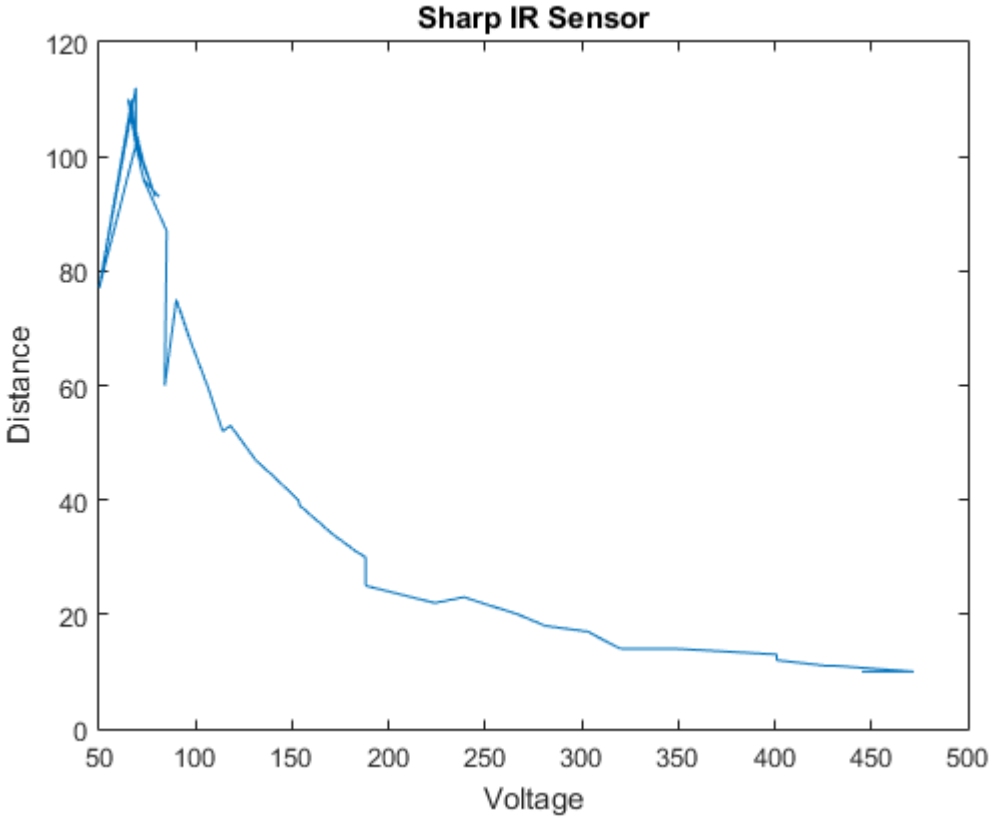

Figure 2: Plot of ADC Output vs the Distance (in cm) for the Sharp IR Sensor

1.2 Project

In the last week and before, I concentrated on learning the philosophy behind ROS and how to write applications using the ROS framework. Since the low-level control of the boat was written using the ROS framework, it was important for us to finish with the ROS assignment as soon as possible and become comfortable with ROS. After finishing the ROS assignment, I concentrated on the following task –

- We were getting some issues in compiling the ROS packages that were written previously by NREC engineers. I was able to troubleshoot the problem.
- Worked on Recording and Playing the IMU data and documented the procedure.

Also along with William and Shiyu, I am working on publishing the RADAR data on a ROS topic. As of now, there is no library available that can be used to get the data from the radar and we are working from scratch to finish this task.

2 Challenges

I have divided the challenges into 2 subsections –

2.1 Task 6

Initially the plan was to prevent debouncing on the toggle switch using internal interrupt but we had already used the 2 interrupts for –

- Handling debouncing on the push button. (Push button was used to change the state.)
- Reading the encoder values.

A possible solution was to switch to a board that had more interrupt pins but we decided to use polling to read the toggle switch output. So, in order to prevent debouncing I had to maneuver by reading data after a delay.

In order to draw the transfer function for the sensor, we needed to sample ADC output/sensor output voltage with distance of the obstacle from the IR-sensor. As Professor Dolan had pointed out in the review, we could have improved out transfer function plot. There was no major glitch in integrating the servo motor.

2.2 Project

During our work on the boat, initially we were unable to connect to the on-board computer on the boat. The on-board computer runs the `ROS MASTER` node and provides access to the various sensors on the boat. We had to take help from NREC engineers who had previously worked on the boat and the rest of the troubleshooting was done by us.

One major challenge, I faced while working with ROS was during the communication between 2 machines. In the ROS assignment, we had to pass the image messages from one machine to another. Passing the raw image frames was creating a lot of latency and the application was not running in

real time. I had to use compressed image messages in order to reduce the latency. Initially, I had thought that the latency was due to the CMU WiFi network but changing to a dedicated WiFi Router also did not help much.

3 Teamwork

The work done by each team member is summerized below –

- Shiyu Dong – Shiyu integrated the force resistive sensor with Arduino and also write the PID control for velocity for DC motor.
- William Seto – William integrated all the code in one single control loop and also integrated the sharp IR sensor.

  Shiyu and William worked collaboratively on tuning the position control of DC motor. Shiyu and William are also working with me on the radar.
- Tae-Hyung Kim – Kim worked on getting analog data from the potentiometer and also wrote code for controlling the stepper motor.
- Tushar Chugh – Tushar developed the GUI and worked on changing the states using the push button.

  Kim and Tushar are also collectively working on the path planning algorithm.

4 Work Overview for Coming Week

Over the course of next 2 weeks, I will be working on publish RADAR data on a ROS topic. I will be working with William and Shiyu to store raw data from the RADAR in a custom ROS message format. The custom ROS message will be published on a ROS topic. The obstacle detection algorithm will then subscribe to this code. I will also be working on updating the team website taking into account the review that was given for the CoDR report.

5 Appendix 1

Listing 1 contains the source code for controlling the servo motor and the toggle switch.

```
/*** BEGIN TOGGLE SWITCH DEFINES ***/

const int button_left = 8
const int button_right = 7

/*** END TOGGLE SWITCH DEFINES ***/

/*** BEGIN SERVO MOTOR DEFINES ***/

#include <Servo.h> // Include the servo motor headers

// Create servo object to control a servo
Servo myservo;
// Connect potentiometer to analog pin 2
const int potpin = 2;
// Servo resets itself to 90 in the beginning
volatile int servoAngle = 90;

/*** END SERVO MOTOR DEFINES ***/


/*** BEGIN SETUP CODE ***/

void setup() {
  /* BEGIN TOGGLE SWITCH SETUP */
  // Use the pull up resistors on pin 8 and 7
  pinMode(button_left, INPUT_PULLUP);
  pinMode(button_right, INPUT_PULLUP);
  /* END TOGGLE SWITCH SETUP */

  /* BEGIN SERVO MOTOR SETUP */
  // Attach the Servo Motor on Pin 9 to the Servo Object
  myservo.attach(9);
  /* BEGIN SERVO MOTOR SETUP */

  // Set the baud rate and begin Serial Communication
  Serial.begin (115200);
}

/*** END SETUP CODE ***/

/*** BEGIN SERVO MOTOR FUNCTIONS ***/

void processServo(int pos, int mode) {
  int pot_val = 0;

  switch (mode) {
    case 1:     //user-defined

      myservo.write(pos);
      servoAngle = pos;
      delay(15);
      break;
    case 2:     //sensor-defined
```

```
      // Read the value of the potentiometer (value between 0 and 1023)
      pot_val = analogRead(potpin);
      // Scale it to use it with the servo (value between 0 and 180)
      pot_val = map(pot_val, 0, 1023, 0, 180);
      // Set the servo position according to the scaled value
      myservo.write(pot_val);
      servoAngle = pos;
      delay(15);
      break;

  }
}

/*** END SERVO MOTOR FUNCTIONS ***/

/*** BEGIN DC MOTOR FUNCTIONS ***/

void processDCMotor(char dir, int value, char mode) {

  int vel;
  String message = "";
  if (dir == '1') {
    value = value * -1;
  }

  switch (mode) {
    case '0':
    // Position control
      if (stateChanged) {
        desiredMotorPos = encoderPos + value;
      }
      positionPID(desiredMotorPos);
      run_motor();
      break;

    case '1':
      // Velocity control
      velocityPID(value);
      run_motor();
      delay(48);
      break;

    case '2':
      // Switch direction of motor rotation
      if(digitalRead(button_left)==HIGH) {
        Motor_speed_value = 150;
      } else {
        Motor_speed_value = -150;
      }
      run_motor();
      break;

  }
}

/*** END DC MOTOR FUNCTIONS ***/
```

Listing 1: Source Code for Controlling Servo Motor and Toggle Switch

6 Appendix 2

Listing 2 contains the Matlab Code for plotting the Transfer Function.

```
function [x] = transferPlotSharp()

  % Read the CSV file
   file = csvread('TeamBAssignment5.csv',1,0);
   x = file(:,1);
   y = file(:,2);
   sys = tf(x',y');

  % Plot the function
   stepplot(sys);

  % Give the title and
   title('Sharp IR Sensor')
  # Give the axis Labels
   xlabel('ADC Output')
   ylabel('Distance')

end
```

Listing 2: Matlab Code for plotting the Transfer function

References

[1] Arduino Servo Library Reference
`www.arduino.cc/en/Reference/Servo`