Sensors and Motor Lab

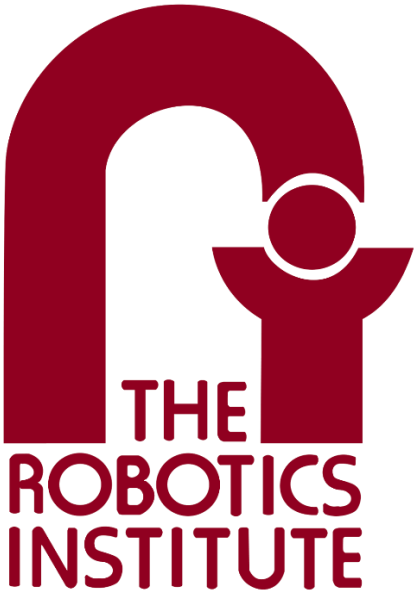
Individual Lab Report #1

VIVEK GOPAL RAMASWAMY

TEAM E- BEYOND SIGHT

Team Members: Chien Chih Ho, Pengsheng Guo, Oliver Krengel, Rohit Murthy, Vivek Gopal Ramaswamy

13th October 2017



Individual Progress:

As a part of the team, I was responsible for soldering the solarbotics motor driver kit, interfacing the potentiometer, servomotor and the slot sensor with the Arduino microcontroller and deriving the transfer plot for the IR sensor. My secondary job included hardware integration.

IR Sensor:

Part no: Sharp 2Y0A21YK0F

Figure 1. IR Sensor[1]

The IR sensor is used to measure the distance of an object. It calculates the distance based on the time of flight of the reflected light from the object. My part in this subsystem was to analyze and generate the transfer function for this sensor.

Setup:

The setup was simple. I had laid down the sensor on a scale (ruler), measured in inches. The IR sensor has three pins as follows

1) power pin

2) receiving pin

3) ground pin

The power pin was connected to a 5V pin of the Arduino and the ground pin was connected to both the Arduino's ground and to the ground of the multimeter. The receiving pin of the IR was connected to the positive terminal of the multimeter. The multimeter was set to measure the voltage in DC.

Process:

I used a small black box which was incrementally increased in steps of 1 inches and the output voltage given by the sensor for the corresponding distance was measured from the multimeter. The process was repeated until I collected 20 points. After that, the points were recorded in the excel sheet, which was further used in MATLAB to generate the plot. I used the fit function for line fitting and generating a linear model for the sensor.

The results are as follows:

Linear model Poly2

f(x) = p1*x^2 + p2*x + p3

Coefficients (with 95% confidence bounds):

p1 = 8.78 (0.073, 17.49)

p2 = -42.64 (-75.66, -9.614)

p3 = 57.62 (27.67, 87.57)

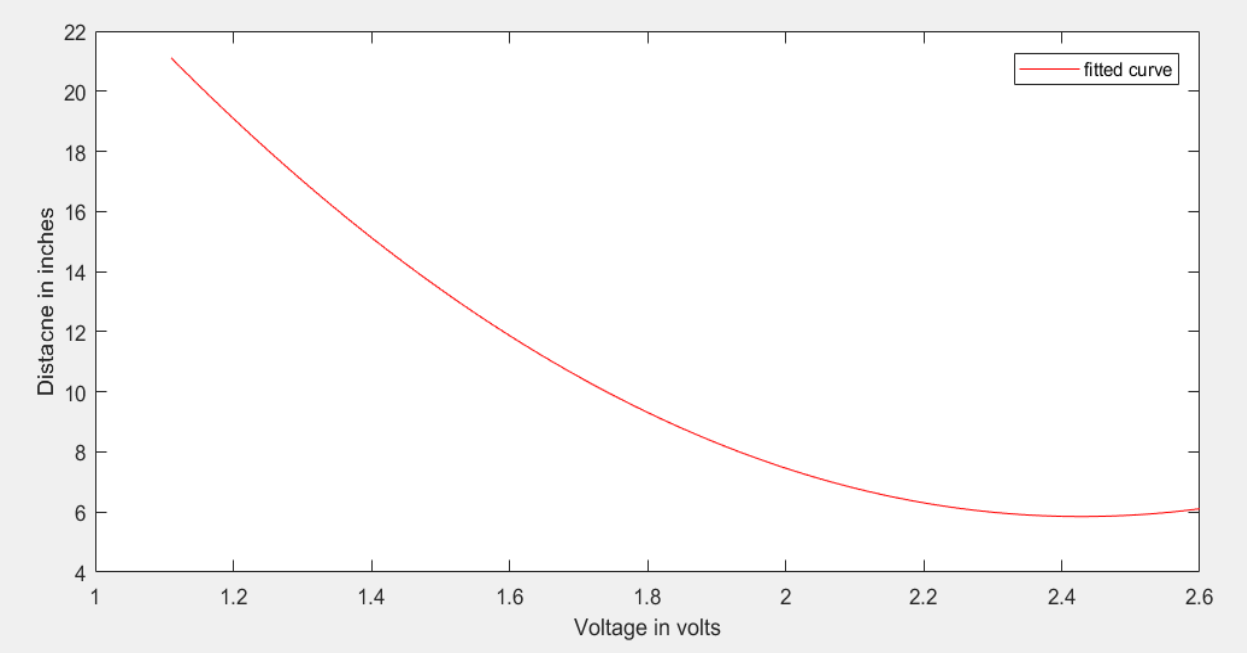


Figure 2. Transfer plot of the IR sensor

Servo Motor with Potentiometer:

Part no: HS-311

Figure 3. Servo Motor[2]

The servo motor has three pins.

1)Power

2)Gnd

3)Signal

The signal pin was connected to the digital pin 9 of the Arduino which supports the PWM function. The potentiometer also has 3 pins Power, Gnd and the output pin. The output pin was connected to the analog pin 2 of the Arduino. For the coding part, I had used the servo library[4]. The analog input was from 0 to 1023, as the ADC had a 10bit resolution. To map it I used the map function and mapped 0 to 1023 to 0 to 180 for the servo motor.

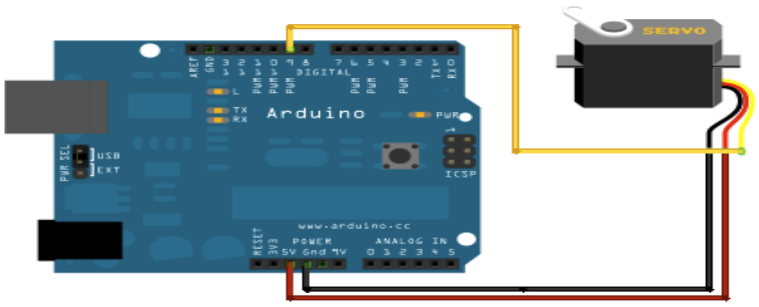


Figure 4. Servo Motor with Arduino[3]

Slot Sensor:

Part no: EE-SX1042

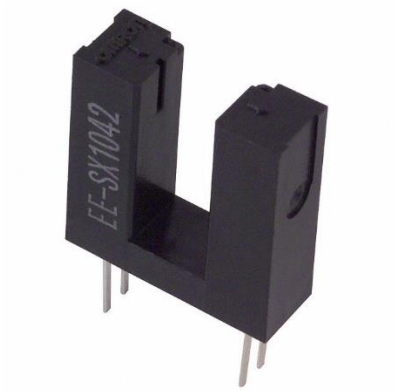


Figure 5. Slot Sensor[5]

Slot sensor also called as fork sensor are used to detect objects which pass through their arms. There are two arms, in which one of them acts as a transmitter and the other as a receiver.

Figure 5. Slot Sensor basic diagram

I used the data sheet[3] to refer the connection for the slot sensor.

Connection Diagram:

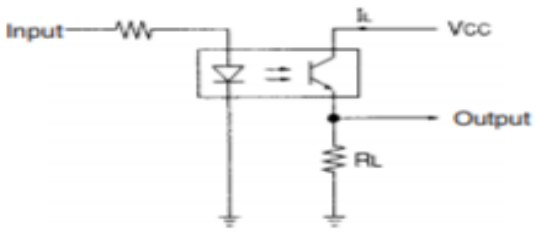


Figure 6. Connection Diagram for Slot sensor[6]

As per the datasheet, the RL was taken as 10-kilo-ohm and the input resistance was 100 Ohm. The input pin of the slot sensor and the Vcc pin was connected to 5V. The output pin was connected to digital pin 4 of the Arduino.

On the coding part, I initialized the slot sensor pin as input and read the state of it is using digital Write. Then using the basic 'if' conditions I printed serially that, whether the slot was open or closed.

Challenges:

Hardware integration was one of the biggest challenges for me as there were a lot of connections in each subsystem. I overcame this by having all the connection diagrams for each sensor interfaced from my team members. After this, I also had made a table of the pins which were used by the sensors. This setup made it very easy for me to properly integrate all the subsystem together. For easy debugging, the wiring was color coded.

The secondary challenge for me was on making my code compatible for integration with other team members code. To solve this, we as a team used a naming convention to name our variables and also had defined certain global variables which were important to the proper functioning of the program.

Final Setup:

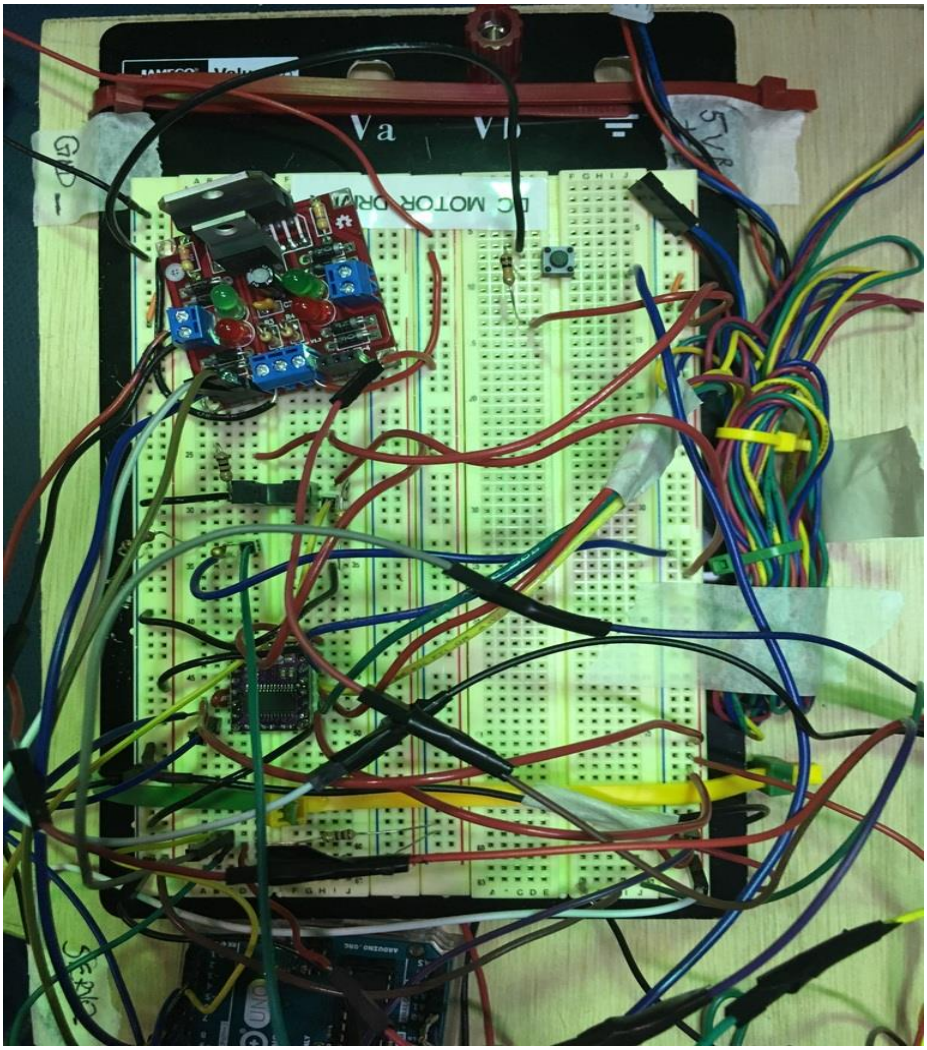


Figure 7. Final Setup

Team Work:

Vivek Gopal Ramaswamy: As a part of the team, I was responsible for soldering the solarbotics motor driver kit interfacing the potentiometer, servomotor and the slot sensor with the Arduino microcontroller and deriving the transfer plot for the IR sensor. My secondary job included hardware integration.

Chien Chih Ho: As a part of the team he was responsible for interfacing the stepper motor and the pushbutton with debouncing. He also played a key role in code integration and also helped us in debugging the GUI.

Pengsheng Guo: As a part of the team he was responsible for interfacing IR sensor, thermistor and controlling DC motor speed by IR. He also helped the team in code integration and debugging.

Oliver Krengel: As a part of the team he was responsible for interfacing encoder and DC motor and also, he helped us in hardware integration and code debugging.

Rohit Murthy As a part of the team, he was responsible for the development of GUI using processing. Also, he played a key role in code integration and debugging communication error between the GUI and Arduino.

Current Project Progress and Future Plans

For now, we have our mechanical cart and the telescopic pole assembled. LIDAR is interfaced with ROS and the point cloud is visualized using Rviz. On the software side, we have read a couple of algorithms such as MV3D and discussed the feasibility of implementation with our mentor Prof. David Held.

Our Future Plans include:

- Complete the mechanical assembly of the sensor mount
- Setup wireless communication with ROS and Run Team D's car
- Filter the point cloud from LiDAR and attempt background subtraction.
- Setup and calibrate the ZED stereocamera.

Task 7 (Sensors and Motor Control Lab) Quiz

1. Reading a datasheet. Refer to the ADXL335 accelerometer datasheet (https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf) to answer the below questions.
   - **What is the sensor's range?**
     Ans: Min-+/3g, Typ=+/-3.6g
   - **What is the sensor's dynamic range?**
     20log(0.707*3.6/150ug)
   - **What is the purpose of the capacitor C_{DC} on the LHS of the functional block diagram on p. 1? How does it achieve this?**
     Ans: Purpose is to achieve decoupling of accelerometer from the noise on the power supply.

     $$Z=1/j\omega C\Omega$$

     $$1j\omega C\Omega$$

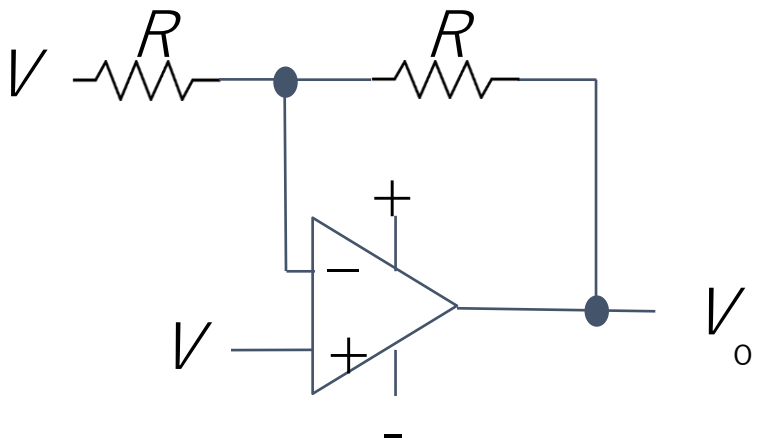
     where ω is the frequency, and C is the capacitance. So if there is a high-frequency noise in the power supply, the capacitor offers a low impedance path, thus removing the noise from the system. It is also called a low pass filter.
   - **Write an equation for the sensor's transfer function.**
     Ans: 1.5+0.3x
   - **What is the largest expected nonlinearity error in g?**
     Ans: 750ug/rms
   - **How much noise do you expect in the X- and Y-axis sensor signals when the sensor is excited at 25 Hz?**
   - Ans: +/- 0.3% full scale
   - **How about at 0 Hz? If you can't get this from the datasheet, how would you determine it experimentally?**
     Ans:

2. **Signal conditioning**
   - Filtering
     - **What problem(s) might you have in applying a moving average?**
       For filtering high-frequency noise, the moving average filter needs a bigger dimension window, which in turn induces a lag in the signal flow and this cannot be used in for applications in real time.
     - **What problem(s) might you have in applying a median filter?**

The problem with median filter is that it is computationally expensive, as the values are required to be sorted out even before applying the median filter. If the median filter is implemented by a quick sort algorithm the best case for time complexity is $\Omega(n \log(n))$.

- **Opamps**
  - In the following questions, you want to calibrate a linear sensor using the circuit in Fig. 1 so that its output range is 0 to 5V. Identify which of V1 and V2 will be the input voltage and which the reference voltage, the value of the reference voltage, and the value of Rf/Ri in each case. If the calibration can't be done with this circuit, explain why.
    - Your uncalibrated sensor has a range of -1.5 to 1.0V.
    - Your uncalibrated sensor has a range of -2.5 to 2.5V.

    Using wolfram alpha , to solve 2 equations with 3 variables, it gives no solution



1) vref =vin=V2
   uncalibrated sensor range -1.5 to 1.0V
   Here Rf=Ri is equal to 1
   Vref=-3 V
2) Solving 2 equations with 3 variables gives no solution, by solving using wolfram alpha

3. Control

- **If you want to control a DC motor to go to the desired position, describe how to form a digital input for each of the PID (Proportional, Integral, Derivative) terms.**

  Ans: **Proportional**: Difference between the current and the desired location.
  Integral: Summing up all the previous values
  Derivation: Difference between previous value and the current value.

- **If the system you want to control is sluggish, which PID term(s) will you use and why?**

Ans: I will use the Proportional term to increase the speed of the system, but I will be careful about not increasing too much as it would cause oscillation.

- **After applying the control in the previous question, if the system still has a significant steady-state error, which PID term(s) will you use and why?**
  Ans: I will use the Integral term to reduce the steady-state error. This is because the integral response will continue increasing unless the steady-state error is zero.

- **After applying the control in the previous question, if the system still has overshoot, which PID term(s) will you apply and why?**
  Ans: I will use the Derivative term to control the overshoot. This is because it will resist rapid change of the error function thus preventing overshoot. It will damp the system for overshoot and oscillation.

References

[1] https://www.sparkfun.com/products/242

[2]http://hitecrcd.com/products/servos/sport-servos/analog-sport-servos/hs-311-standard-economy-servo/product

[3] https://www.arduino.cc/en/Tutorial/Sweep

[4] https://www.arduino.cc/en/Tutorial/Sweep

[5] http://www.components-center.com/datasheets/9a/EE-SX1042-1.pdf

[6] http://www.components-center.com/datasheets/9a/EE-SX1042-1.pdf

Appendix

Code:

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = A2;  // analog pin used to connect the potentiometer
int slotpin=4;
int val;    // variable to read the value from the analog pin
int readinput=0;
float start_clock=0;
float stop_clock=0;
float timeresponse=0;

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  pinMode(4, INPUT);
  Serial.begin(9600);
}

void loop() {
```

```
 start_clock=micros();
 val = analogRead(potpin);           // reads the value of the potentiometer (value between 0 and
1023)
 val = map(val, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
 myservo.write(val);                  // sets the servo position according to the scaled valu
 readinput=digitalRead(4);
 if(readinput==1)
 {
  Serial.println("Slot Open");
 }else if(readinput==0)
 {

  Serial.println("Slot Closed");
  stop_clock=micros();
  timeresponse=stop_clock-start_clock;
  //Serial.println(timeresponse);
 }

 delay(15);                          // waits for the servo to get there

}
```