SY202 – Cyber Systems Engineering

Report due on Tuesday 16-April-2019

TERM PROJECT #01: Elevator Closed-loop Control

Abstract

In this lab project, you will systematically investigate and design closed-loop position controllers for an elevator system. To accomplish this, you will use feedback from a calibrated position sensor and pulse width modulation (PWM) motor control, with controllers implemented on an mbed microcontroller using the C programming language. The objectives of this lab are to:

(1) Synthesize and put in practice all concepts discussed throughout the course, from the implementation of closed-loop systems to the use of actuators, sensors, and microcontrollers;
(2) Identify and understand the hardware necessary to control and sense within a cyber physical plant;
(3) Apply mathematical and experimental tools to calibrate and use a sensor;
(4) Understand the importance of real-time control;
(5) Highlight the differences between logic control and some classical control techniques (PID feedback control); and
(6) Collect data and analyze system performance based on experimental results.

Introduction

In this project, you will explore closed loop position control of a simple electromechanical system. The system you will control is a small elevator, driven by a DC motor and pulley, with position feedback from both an infrared height sensor and motor encoder. The project will be divided in **two parts.**

During the first part, you will collect data from both sensors at different heights. Using this data you will then calibrate the sensors, i.e., you will find a mathematical relationship between the output of the sensors (voltage or encoder counts) and the height of the elevator. Then, you will need to select the best sensor to use based on noise and reliability.

During the second part, you will design and implement four different controllers to regulate the position of the elevator according to some expected performance criteria. The controllers are: a logic controller, a proportional controller, a proportional-integral controller, and a proportional-integral-derivative controller. You will then collect data from experiments and compare the performance of all four control systems.

Part I: Sensor Calibration [Timeline: 26-27 March]

The elevator has two sensors. At the top of the elevator cage (see Figure 1 - left), there is an infrared range sensor namely GP2D12. The GP2D12 provides an analog voltage output which is a function of the distance from the sensor to the elevator. The GP2D12 analog output (a yellow wire) should be connected to pin #p20 in your mbed, while its black wire should be connected to ground (GND). The second sensor is an encoder mounted on the DC motor (see Figure 1 - right). The encoder doesn't directly measure position or distance from the elevator. Instead, it measures the angular rotation of the motor in "counts" which can be used to estimate the distance travelled by the elevator. The encoder output utilizes two wires that should be connected to pins #p5 (yellow wire) and #p6 (white wire).

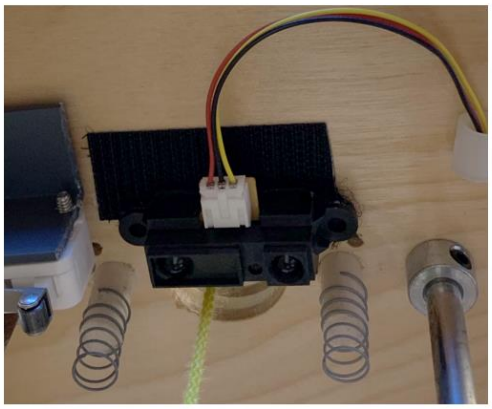
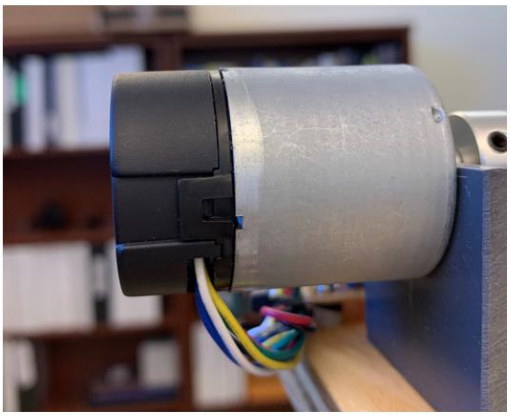

Figure 1. GP2D12 Infrared Sensor (left) and Motor Encoder (right).

Task I: Collect data from both sensors using the mbed.

Write an mbed code that (1) takes 10 analog voltage readings from the infrared range sensor and 10 count readings from the encoder and (2) displays all 10 readings for each sensor along with the average for both sensors. Add a wait function of 0.05 seconds between each sample to allow enough time for each sensor to generate a new reading. The program should wait for the user to enter a key once he/she is ready to collect data at the desired height. Give the user the option to end the program. By averaging 10 readings per sensor, you reduce the probability of sensing errors.

> **Note:** Before you start any experiment, you will need to make sure that the elevator is being supplied by approximately 12 V (+/- 0.5 V).
>
> **Note**: When using `AnalogIn` to read the infrared sensor value, the voltage reading is a float value automatically normalized by the mbed maximum voltage, which is 3.3 V. Therefore, to get the right voltage reading, multiply the infrared range sensor reading value by 3.3 V. Use the `Mbed_Elevator_Calibration_template.txt` as template.
>
> **Note**: To read the encoder, include the library `QEI.h` in the preamble. To declare the encoder, use: `QEI EncoderName(p#A,p#B,NC,64,QEI::X4_ENCODING)` in the preamble. Note that `EncoderName` can be any name of your choice. Use p5 and p6 as pin numbers for `p#A` and `p#B`, respectively. Use `EncoderName.getPulses()` to read the encoder within the main function. `EncoderName.getPulses()` returns the position of the motor as an integer value. Cast as a float, i.e., `Value = (float)EncoderName.getPulses();`

Once you have a working mbed program that is able to read, average, and display data from both of your sensors, fill out Table 1. Manually (by rotating the motor shaft), position the elevator at the given height. Make sure that the cable is not loose but straighten. Make sure not to block the view of the infrared sensor with your hand and that the encoder counts are positive (if you are getting negative numbers, you will need to manually rotate the motor in the other direction). Also, note that the power supply needs to be ON in order to supply power to the the motor encoder.

Table 1. Sensor Calibration

Height (in)	Average Reading		Height (in)	Average Reading	
	Infrared Sensor (V)	Encoder (Counts)		Infrared Sensor (V)	Encoder (Counts)
7			17		
8			18		
9			19		
10			20		
11			21		
12			22		
13			23		
14			24		
15			25		
16			26		

Note: The sensitivity of the infrared sensor is higher between 10 inches and 24 inches. You should notice that the voltage remains almost constant for heights around or below 10 inches and that they suddenly drop after (or around) 24 inches.

Note: When resetting the mbed; make sure that the elevator always starts at the lowest position. The encoder gets reset to zero whenever you reset the mbed or cut power to the mbed.

Task II: Use MATLAB to find polynomial fit for both sensors.

Calibrate the Infrared Sensor:

1. Copy the results in a MATLAB script. Then, plot the results for the infrared sensor as Height (in) vs Average Voltage (V), similar to Figure 2 (left).

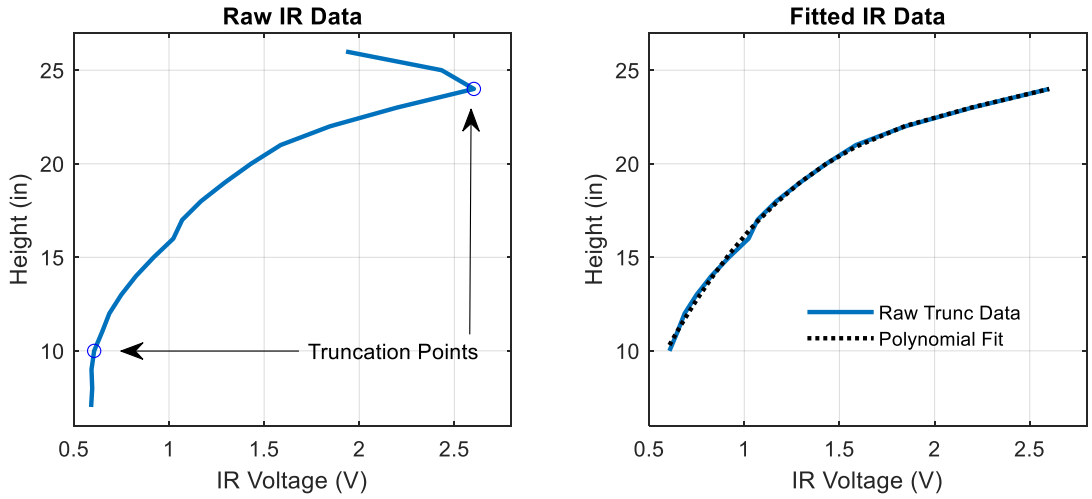

Figure 2. Sensor Calibration

2. Note that the plot is not a one-to-one function, meaning that for some values of "x" (i.e., voltage) you can have more than one value in "y" (height). This is a problem, as a reading from the sensor can be interpreted as two different values. You need to truncate your height and voltage vectors to those values

where the voltage and height have a one-to-one relationship. For instance, consider the example in Figure 2 (left). The voltage is increasing from height = 10in to height = 24in. Before 10in, it remains practically constant (not increasing) and above 24in it starts to decrease. Define a new truncated height vector (`TruncatedHeight`) in MATLAB that only includes heights for which the voltage values are increasing. Similarly, define a new truncated voltage vector (`TruncatedIRVoltages`) that includes the respective voltages.

3. In MATLAB, use the `polyfit` function to fit a polynomial to your data. `polyfit` will create a polynomial that takes the analog measurement data and approximates the height. The polynomial is of the form: `height` $= p_1x^N + p_2x^{N-1} + \ldots p_{N+1}$, where x is the analog measurement (voltage) and p_i are the coefficients generated by polyfit. Here, N is the order of the polynomial. For the infrared sensor use a value for N between 2 and 4. Plot in the same figure the raw truncated data along with the fitted line (see Figure 2 (right)). How well does the fitted line match the actual data?

   Note: Fitting a polynomial to a data set can be accomplished by the following code (use `help polyfit` and `help polyval` for more details):

```
p = polyfit(TruncatedIRVoltages, TruncatedHeight,N);
fittedHeight = polyval(p, TruncatedIRVoltages);
plot(TruncatedIRVoltages, fittedHeight,'—k')
```

   The output of the polyfit function contains the coefficient of your Nth order polynomial. Enter: `format long g` into the command window of MATLAB and then print these coefficients to your workspace. Record the coefficients using all the decimal places given.

Calibrate the Motor Encoder:

4. Repeat step 1 and 3 but using the values from the encoder counts rather than infrared voltage. Note that in the case of the encoder, you do not have the same nonlinearity as seen with the infrared sensor, therefore you DO NOT need to truncate your data and you can skip step 2. Furthermore, note that the relationship between encoder counts and heights is almost linear. For the polynomial fit, use N=1 which correspond to a line equation. Make sure to record your coefficients.

Task III: Display to TeraTerm position of elevator in inches using both sensors.

Write a new mbed program that continuously samples the voltage from the infrared sensor and the counts from the encoder, then converts the sensors reading into height using the polynomial equation obtained in Task II. The program should display to Tera Term the position of the elevator using both the infrared sensor and the encoder. Add a pause (i.e., wait) of 0.2 sec between samples to force a new reading. Since you will be using this conversion from sensor readings to inches in Part II, you may write the conversion as C++ function.

Manually move the elevator up and down, while running the program. Observe the accuracy of both sensor. In your opinion, which sensor is more precise and accurate? Which sensor better reproduce/repeat the same result for the same height? Which sensor has more noise?

Summary for Part I

You will need to keep the following data and plots to be included/used in your report:

1. Check with your Instructor after the completion of each Task.
2. Keep record of Table 1 and mbed programs from Task I and Task III.
3. Two plots like Figure 2 (right) with the raw and fitted data lines for both the infrared sensor and encoder.
4. Equation (along with coefficients) for your polynomial fits for both sensors.