Phase 3 Lab Exercise #3 – Filtering Data

INSTRUCTIONS

Complete the exercises below and upload them to Canvas as a single MATLAB script file using the naming convention "ENGR131_21S_P3_Lab##_abc123.m", replacing abc123 with your Case ID, and ## with the two-digit lab number.

For example, if Dr. Williams were submitting this lab, it would be ENGR131_21S_P3_Lab03_mrw8.m

For your script, please perform the following:

1. You may use the code and notes from class, the textbook, MATLAB's documentation, and anything you find using Google to solve these problems.
2. Use comments as appropriate to indicate your thoughts and how your code works (or is supposed to work). This is 5 points (9%) of your grade.

BACKGROUND

Servo motors are devices that contain a DC motor, gearbox, position sensor (usually a potentiometer, sometimes an encoder), and circuitry to provide position control. The position is input by a micro controller sending it a pulse width modulated (PWM) signal between 0 and 255 which corresponds to 0-100% of the full range of the servo (generally 180º or so). In the case of Arduinos, there is a specific command and associated library that allows users to specify the desired position (between 0º and 180º) and the microcontroller will automatically generate the correct PWM output signal. The servo will hold this position as long as it is instructed to by the microcontroller and is not physically overpowered by an external force.

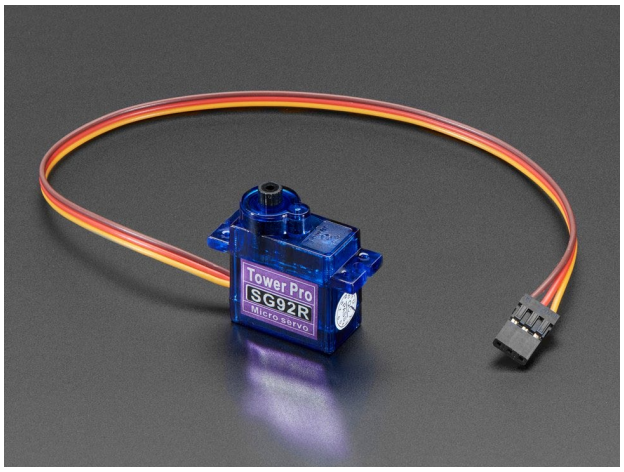

Figure 1. Photograph of a servo motor like the one in your Elegoo kit.

One downside is that they also generate a large amount of electromagnetic noise that can get picked up when recording data. The more the servo switches direction, the more instances of noise. In this lab exercise, we're going to use a servo set to move to random locations to corrupt our light sensing recording and Matlab to clean up the recorded data.

QUESTIONS

There are 4 questions for this lab.

1. Arduino Setup (10 Pts)

A. For this lab, you will be using the light sensor to collect data similar to the Phase II and III Lab 1 exercises. Start by recreating the light sensing portion using the photoresistor circuit from ENGR 131 21S-IN-030-21 (Arduino Basics).

B. In the Arduino IDE, open the Servo Sweep example

C. Working from your analog input with serial output code, add the items below, using the servo sweep example as a reference:
   i) Include the servo.h library
   ii) Create the servo object
   iii) Define a variable to hold the value of the position of the servo
   iv) In the setup() function, attach the servo to digital pin 9
   v) In the loop() function add the following:
      (a) Use the random command to assign a random value between 0 and 180 to the servo position variable
      (b) Write this position to the servo
      (c) Use a delay of 10 at the end of this function

D. Connect the servo to your Arduino as shown in Figure 2.

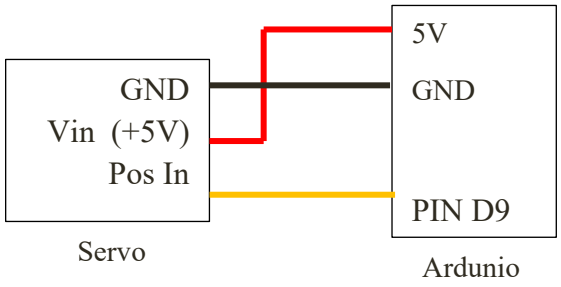

Figure 2. Schematic showing how to connect the servo to the Arduino. Note that on some servos, the ground wire may color coded as brown. The connections for the photoresistor circuit is not shown for clarity.

2. Revise GUI (10 Pts)

A. Starting with your GUI from Phase III Lab 2, save as a new file and remove all controls (and their callbacks) EXCEPT for the following (Figure 3):
   i) Record time knob (revised for 5, 10, 15, and 20 seconds)
   ii) Edit Field (Numeric). Rename this to be FilterCutoff
   iii) Recording progress gauge
   iv) Ready lamp
   v) Record button
   vi) Plot button
   vii) Plotting axes

B. Add the following controls to the GUI
   i) Radio button group with three buttons. Label these Raw, FFT, and Filtered
   ii) Numeric slider. In the Property Inspector (lower right when in Design View) set the limits for the slider to be 1,5 and the MajorTicks to be 1,2,3,4,5

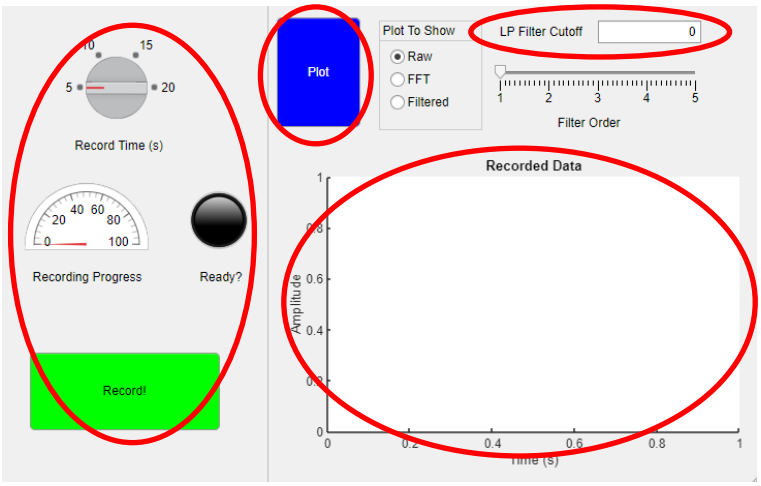

Figure 3. Example of final system GUI. Circled items are those retained from the Phase III Lab 2 GUI. Your final GUI may look different than as shown, but must have these controls.

3. CREATE FUNCTIONS (15 PTS)

In Code View mode, create the following functions:

A. A function that computes the Fast Fourier transform (complete with shifting and absolute value) of the data passed to it. The output of this function is the transformed and formatted data.

B. A function that computes and low pass filters the recorded data. The cutoff frequency comes from the value of the numeric edit field and the order from the slider. The output should be the filtered recorded data.

4. CALLBACKS (20 PTS)

A. Revise the callback for the record button to include:
   i) Compute dt (average time step between samples) from the array of recorded time stamps.
   ii) Compute the sampling frequency, Fs (1/dt)
   iii) Save both dt and Fs as app properties

B. Revise the plot button to plot the recorded data based on the radio button selected:
   i) If Raw is selected, plot the recorded signal as it was recorded
   ii) If FFT is selected, call the function created in 3.A and pass the recorded data app property to the function. Plot the resulting output using x-axis limits of 0 to 20 Hz.
   iii) If Filtered is selected, call the function created in 3.B and plot the filtered data.

For all plots, change the axes properties such that the plot has the correct title, axis labels and limits.

Hint:

- *app.<AxesName>.Title.String = '<Insert Title Here>'*
- *xlim(app.<AxesNames>,[0, <recording time>], etc.)*

OPERATION

To demonstrate your system, connect the Arduino. The servo should be moving to random positions and making a lot of noise (auditory and electrical). While sweeping the light over the photoresistor as in Phase III Lab 1, record 5 seconds of data and view it – is it noisy? Plot the Fourier transform of the signal - is there a spike at about 1 Hz? Apply a low pass filter that is appropriately high enough to cut out the noise while preserving the light data as much as possible. How does the order of the filer affect the plot? What happens to the plot of the Fourier transform if you record for a longer period of time?

Revision	Description	Date
A	Original Document	4/15/2021