EEE 342 **Spring 2019**

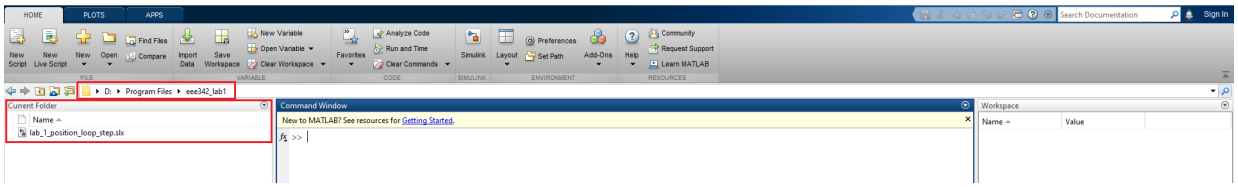
Lab 1 - System Identification

As in preliminary work, you will firstly need to identify the system by using step response of closed position loop. To obtain step response of your kit, Simulink files and hardware communication need to be configured. The steps in part-1 explain which adjustments you need to do to get samples from a physical system properly. In the second part, you identify the closed loop transfer function by using second order approximation on three different setups with the same DC motor. Lastly, in third part, as an introduction to frequency domain techniques, you will plot Bode diagrams of estimated closed loop TFs.

Part-1: Hardware Connection and Configuration

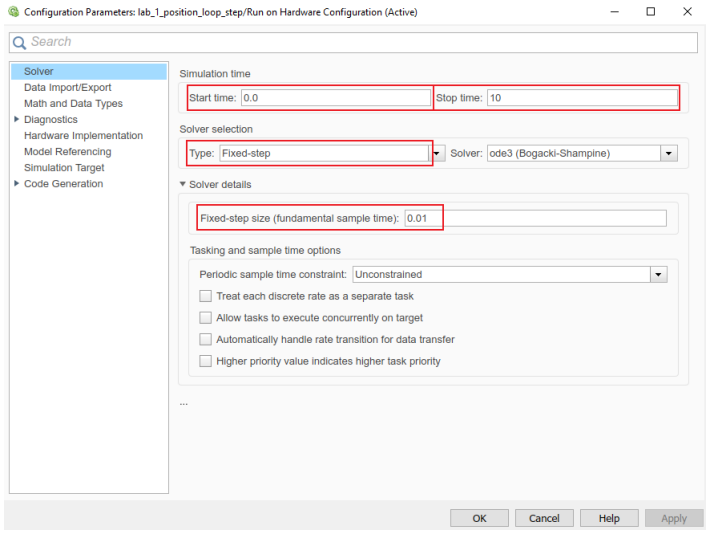
The sole purpose of this part is to assist you to congifure Simulink diagrams and provide a proper connection between the kit and Matlab. Therefore, you can skip this part in your reports except the plot of position vs time data obtained from hardware.

1. Connect the power source to your kit. This will cause the Arduino to run the last installed code.
2. Please close Matlab and plug your USB cable for Arduino and wait until a COM port is assigned. Then, you can open Matlab.
3. Create a folder and download `lab_1_position_loop_step.slx` file from moodle into this folder. Then, set your Matlab current workspace directory to the folder you created as shown below.

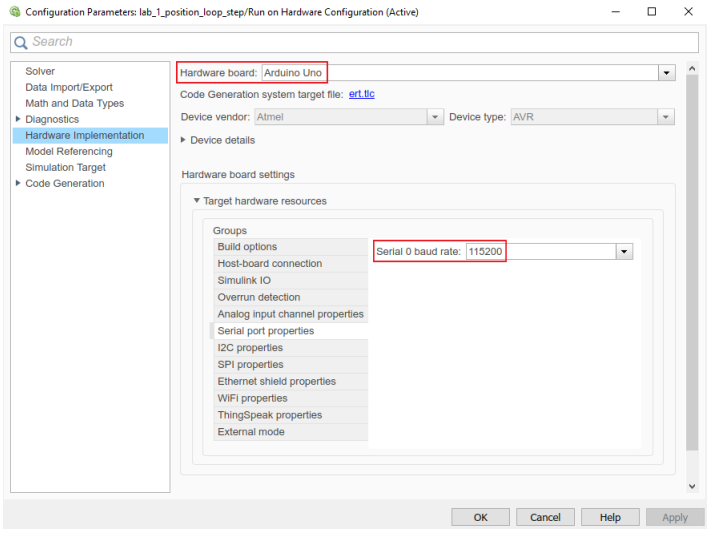


4. Open `lab_1_position_loop_step.slx` file and configure the voltage source (Step block): Step time: 0, Initial value: 0, Final Value: 90, Sample Time: 0.01.
5. The whole configuration part will be done in "Configuration" window. You can open this window by clicking CTRL+E or generic tools button or by opening "Model Configuration Parameters" under Simulation menu.

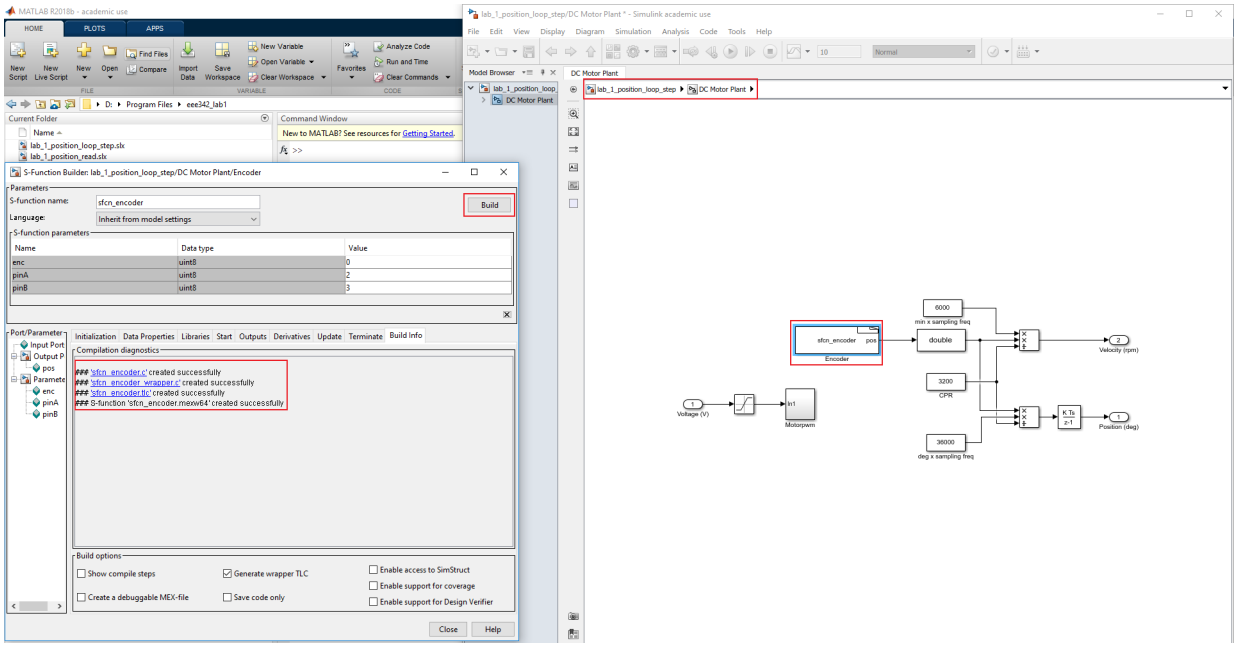
In order to provide a communication between Simulink and kit, you firstly need to identify the hardware you are using. This identification allows the Simulink to recognize pins of Arduino and compile accordingly. The kits contain Arduino Uno, therefore in hardware implementation pane in configuration window, you need to select Arduino Uno as hardware board. Also, in the same pane, set Serial 0 baud rate to 115200, which can be found in Serial port properties. The figure provides visual content for this description (note that these panels may vary based on versions of Matlab).



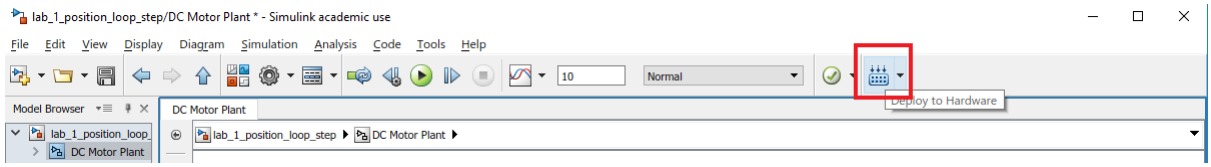
Now that we configured the communicated device, we need to set duration and sampling period. As you have done in preliminary work, this can be done by opening the Solver pane in Configuration window. Set Start Time to 0 and Stop Time to 10. Then, select the Type as Fixed-step in Solver selection panel. Lastly, open Solver details and set fixed-step size (fundamental sample time) to 0.01. The figure below provides visual content for this description.



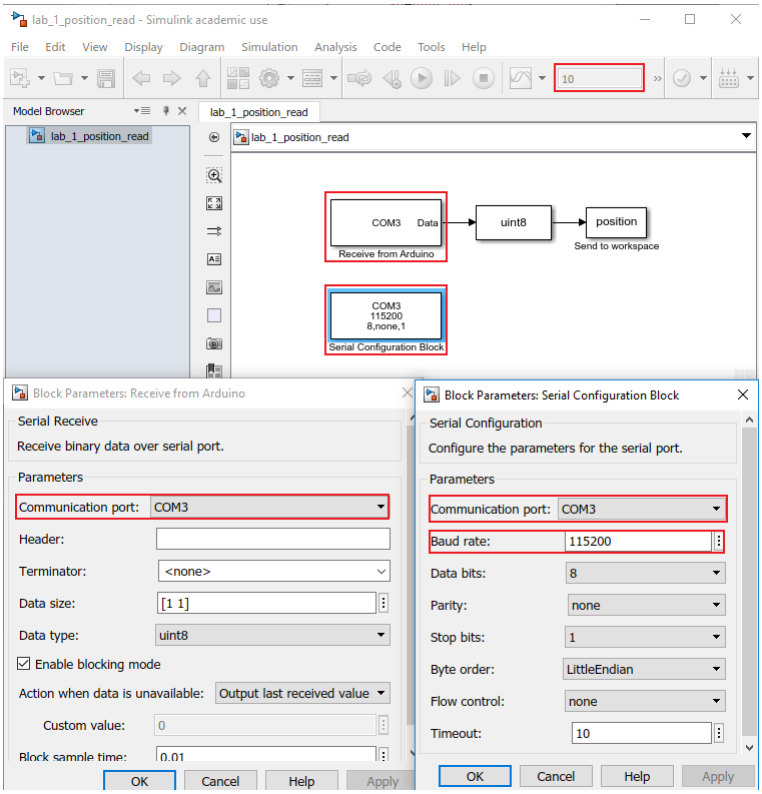
6. There is a final step to provide proper communication between Simulink and Arduino, that is to compile embedded code in `.slx` file. Open DC Motor Plant in `lab_1_position_loop_step.slx` and double click on sfcn_encoder block. Press the "Build" button on top right of the last opened window. You will need to see the following at the end of compile.



7. Close the last window and open DC Motor Plant in `lab_1_position_loop_step.slx`. Press the button with title "Deploy to Hardware", as shown below. Now, the embedded code is built within Arduino, and you will see that the DC motor runs. Observe that the pointer stays steady at 90 degrees plus the initial state. However, we are not receiving the data yet.



8. Download `lab_1_position_read.slx` to your folder and open it. This file will allow us to receive data. Open "Receive from Arduino" block and set Communication Port to **second** line (for example COM4). Repeat the same procedure for "Serial Configuration Block". You also need to set duration to 10, as can be seen from the following figure.



9. Apply these changes and close these windows. Open `lab_1_position_read.slx` again, enter 10 for simulation duration and click "Run" as shown below. After 10-15 seconds, you will see a *timeseries* data called "position". Save this data as `pos_1`

Check-1 Plot received data and show the figure to one of the TAs for the first check.

Part-2: System Identification in Time Domain

1. As you have done in your preliminary work, you will need to estimate first order approximation of DC motor by using second order approximation on received data (estimation of ω_n and ζ by using maximum overshoot and settling time).

$$\zeta = \frac{|\ln(M)|}{\sqrt{\pi^2 + (\ln(M)^2)}}, \quad \omega_n = \frac{4}{\zeta \times T_s}$$

   where M and T_s are maximum overshoot and T_s respectively. Note that M is the ratio between maximum overshoot value and steady state value (90), and T_s is the first time that the response is in 2% error band ($88.2 < y(t > T_s) < 91.8$ in our case).

   Show your calculations and sample points on figures in your report.

2. Download `lab_1_position_sim.slx` file to your folder. Note that in addition to your design in preliminary, this block diagram has friction and friction compensation models, to provide a more realistic result. Open this file and integrate your first order approximation to DC motor block.

3. Plot the response of simulation on top of the data you received from hardware. You can use the following lines:

```
figure; plot(position,'b','LineWidth',2);
hold on; plot(pos_sim,'r','LineWidth',2);
xlabel('Time (sec)'); ylabel('Position (degrees)');
title('Position vs Time');
legend('Hardware result','Simulation result');
```

Check-2 Show the figure to one of the TAs for the second check.

4. Change C_p and LPF in `lab_1_position_read.slx` to followings respectively, obtain responses for $r(t) = 90u(t)$ again as you did in tenth step of part-1, save received data as `pos_2,pos_3` respectively, and repeat items 1-3 for each setup.

   - $$C_p = 2, LPF = \frac{0.1}{0.6s + 1}$$
   - $$C_p = 2, LPF = \frac{0.1}{0.05s + 1}$$

Check-3 Show all 3 plots to TAs to receive your third check for this lab.

Part-3: System Identification: Introduction to Bode Plot

1. Plot the Bode diagrams of all 3 closed position loop transfer functions in one figure. You can use the following code, where you need to change initial parameters according to C_p, LPF you used for each data and the corresponding first order approximations you found in previous part.

```
% % Initial Parameters
Cp = [1, 1, 1]; % change this Cp(1) --> first Cp etc.
% LPF = K_LPF/(t_LPF*s+1)
K_LPF = [1, 1, 1]; % change this
t_LPF = [1, 1, 1]; % change this
% P = Km/(tm*s+1)
Km = [1, 1, 1]; % change this
tm = [1, 1, 1]; % change this

% % Plot Bode Diagram of all 3 CLs
figure;
for k=1:3
[w,Tp] = freq_data(Cp(k),K_LPF(k),t_LPF(k),Km(k),tm(k))
subplot(2,1,1);
semilogx(w,20*log10(abs(Tp))); grid on; hold on;
subplot(2,1,2);
semilogx(w,unwrap(angle(Tp))*180/pi); grid on; hold on;
end
subplot(2,1,1); ylabel('Mag (dB)');
title('Bode Plots of estimated closed loop TFs');
subplot(2,1,1); ylabel('Phase (deg)');
xlabel('Frequency (rad/s)');

function [w,Tp] = freq_data(Cp,K_LPF,t_LPF,Km,tm)
no_of_samples = 500;
w = logspace(-2,2,no_of_samples);
s = 1j*w;
Pv = Km./(tm*s+1);
Cv = K_LPF./(t_LPF*s+1);
Tv = (Pv.*Cv)./(1+Pv.*Cv);
Tp = Cp*Tv./(1+Cp*Tv);
end
```

Check-4 Show Bode plots of closed position loops in one figure to a TA for the last check.

In your report, you are expected to explain the work done in order. It needs to include all plots you drew, all mathematical equations you did, and all the results you obtained in the lab. You also need to comment on each result you obtained between lab checks. Do not forget to use report template and write introduction and conclusion parts.