

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

Checklist for Lab 03:

In this lab, we will design continuous-time filter and discrete-time (digital) filters in Section 1, and start to play with Arduino in Section 2. So, you will need:

- (1) A computer installed with MATLAB and a USB port/adaptor.
- (2) Arduino Uno (The tutorial for this lab and follow-up labs will be based on Arduino Uno.)
It is acceptable to use other boards that work with MATLAB/Simulink. Check this link to find compatibility between MATLAB/Simulink and Arduino or Raspberry board if needed.
<https://www.mathworks.com/hardware-support/arduino-matlab.html>
<https://www.mathworks.com/hardware-support/raspberry-pi-matlab.html>
The pins are set up differently on different boards, so please be careful and always remember to check the pin map when you are using other boards.
- (3) It is recommended to install packages in Section 2 on your own computer or the computer you can use for follow-up labs.

Section 1. Filter Design

In this section, continuous- and discrete-time low- and high-pass filters will be introduced. Consider the example signal $f(t) = 0.05 \sin(2 \times \pi \times 5000t) + 2 \sin(2 \times \pi \times 50t)$, which includes two frequency components at 50 Hz and 5000 Hz with different amplitudes. The sinusoidal wave is plotted below.

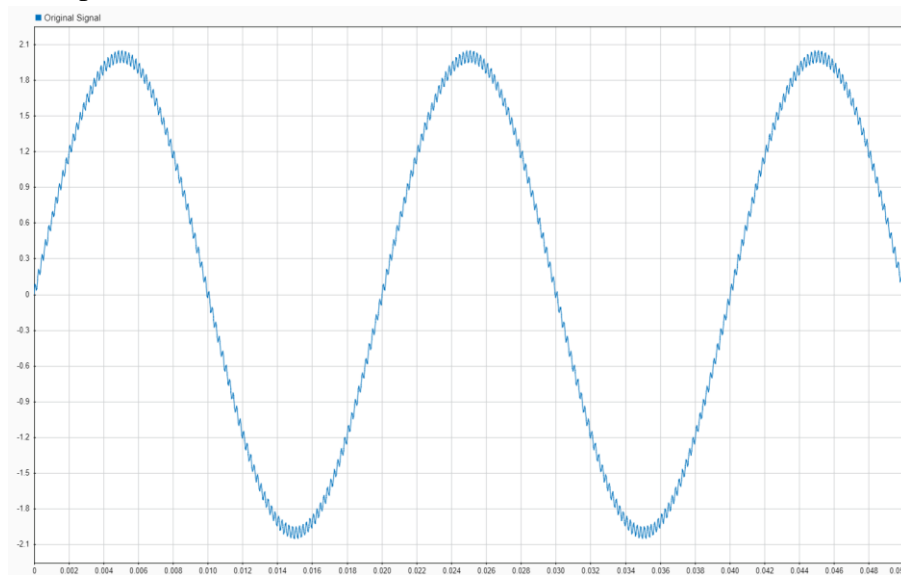


Figure 1. sinusoidal wave Form of Signal

To build a model consisting of signals and filters, “Sources/Clock”, “User-Defined Functions/MATLAB Function”, and “Continuous/Transfer Fcn” blocks from the Simulink library will be used. The model can be constructed as seen below.

- ★ Might be helpful to pull up prior lab assignments to reference while doing this lab
- ★ Seems that Simulink blocks can look different based on the block sizing and software version

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

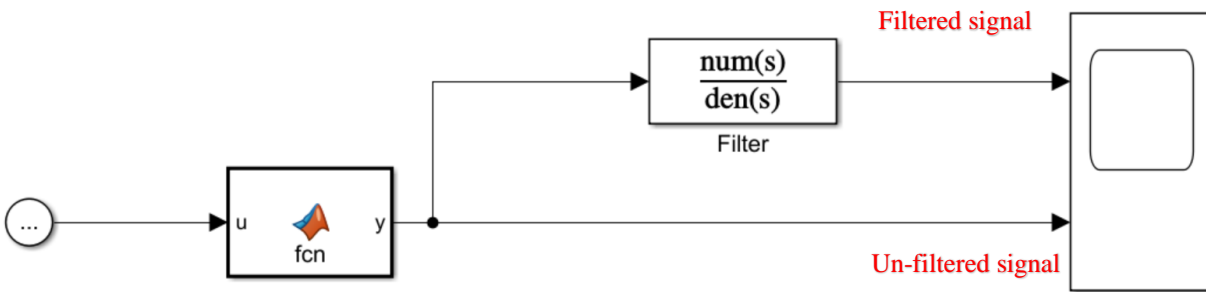


Figure 2. Model for Filter Design

The *MATLAB Function* allows us to define the equation between y and u by programming. Input u takes the time-serial point to the function block and output y is the outcome by the function block's programming. To realize (generate) the signal, we double click the MATLAB Function block to open the editor and type in the following code.

```
1 function y = fcn(u)
2
3 y = 0.05*sin(2*pi*5000*u)+2*sin(2*pi*50*u);
4
5 end
6
```

★ You exit out of the MATLAB Function code to return to the Simulink block diagram model by clicking the “Back” Button, “Up to Parent” Button, or selecting your Simulink model’s file name

Next, we focus on the design of filters.

★ Copy/pasting your Simulink model on the same canvas will be helpful so you can duplicate, adjust, and compare the two filters at the same time.

1.1 Continuous Low-Pass Filter

The low-pass filter can help to attenuate the high frequency signal and pass the low frequency signal.

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

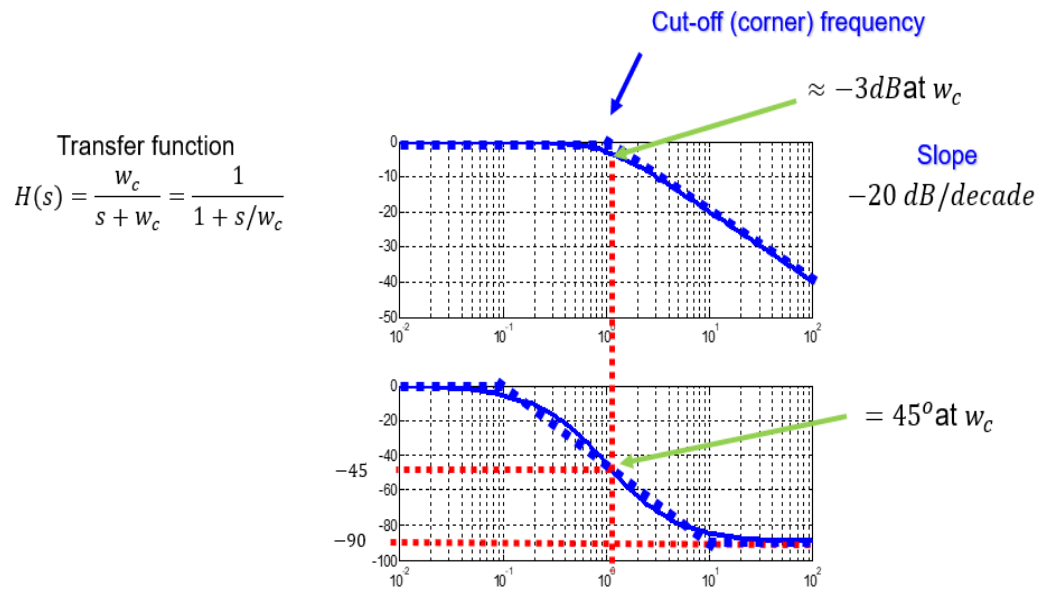


Figure 3. Continuous Low Pass Filter

The cut off frequency ω_c (in rad/s) needs to be higher than the expected/desired frequency and lower than the unexpected/undesired frequency. In this example, we choose $f_c = 200\text{Hz}$ that is in between 50 and 5000Hz. Hence the parameters of “Transfer Fcn” block need to be set as below.

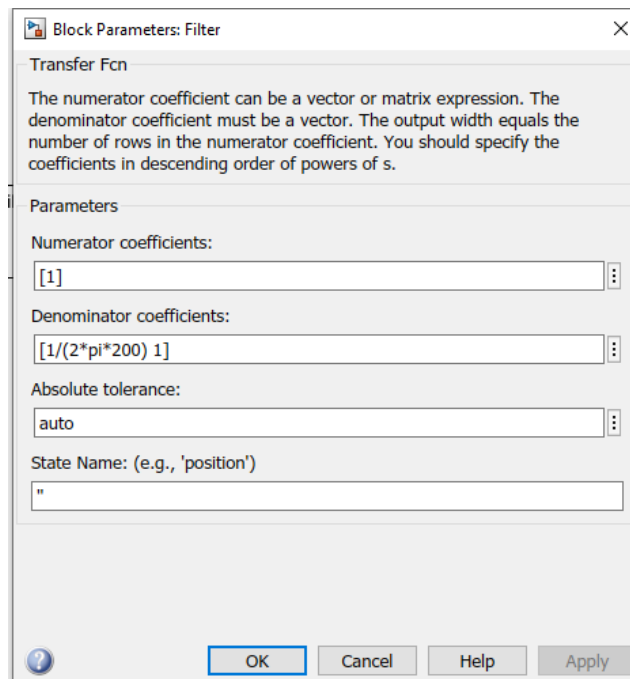
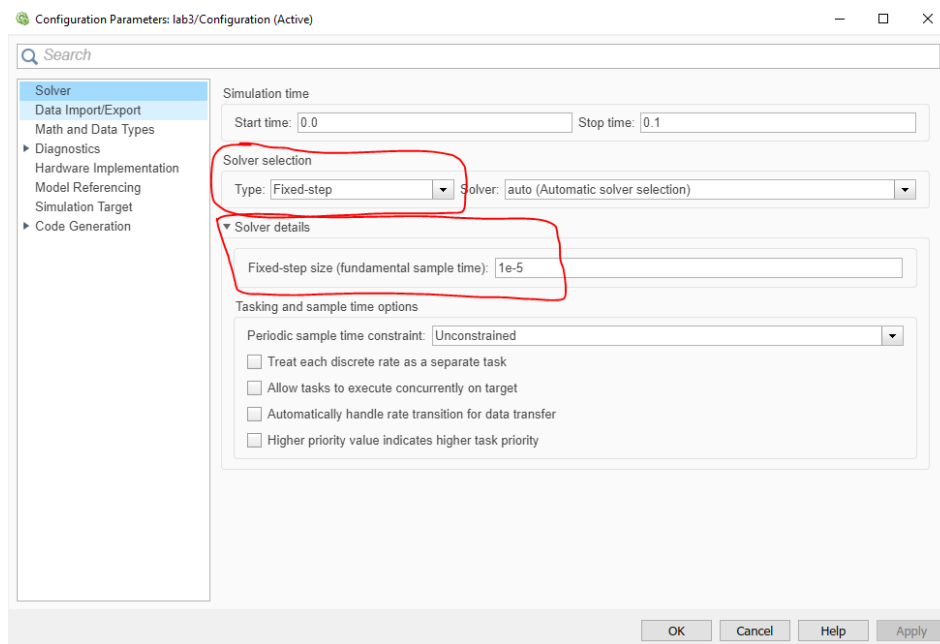


Figure 4. Low Pass Filter Parameter Configuration

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

Next, we set up the simulation step size. Go to the **MODELING** tab => 'Model Settings', select 'Fixed-step' in Solver options, and type '1e-5' as the fixed step. Then we set the stop time as 0.05.

(note: The step size should be chosen smaller than the period of the largest frequency component)



After running the model simulation, the comparison results are shown below. The high frequency component has been filtered out, and only the low frequency component is remained. However, a **phase delay** can be observed by comparing the un-filtered and filtered signal.

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

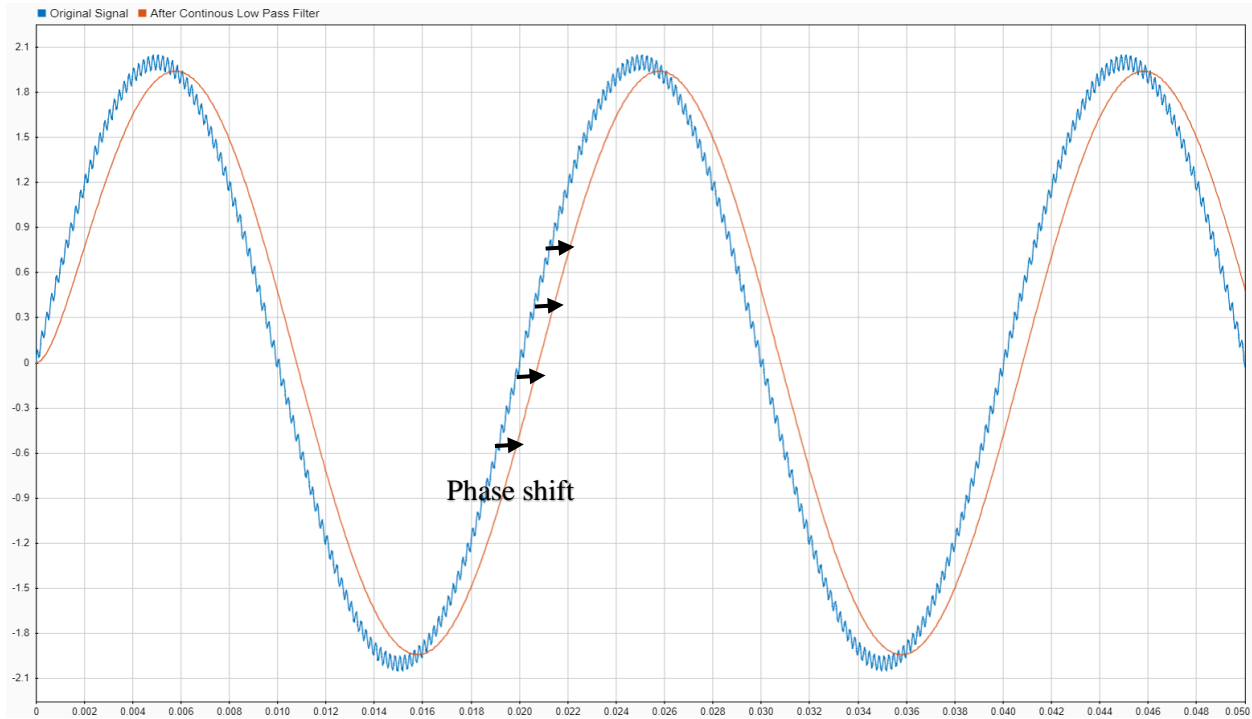


Figure 5. Comparison Results after Filter

1.2 Continuous High-Pass Filter

The high-pass filter can help to attenuate the low frequency signal and pass the high frequency signal. The continuous high-pass filter was introduced in the lecture.

$$\text{Transfer function} \\ H(s) = \frac{s}{s + w_c} = \frac{s/w_c}{1 + s/w_c}$$

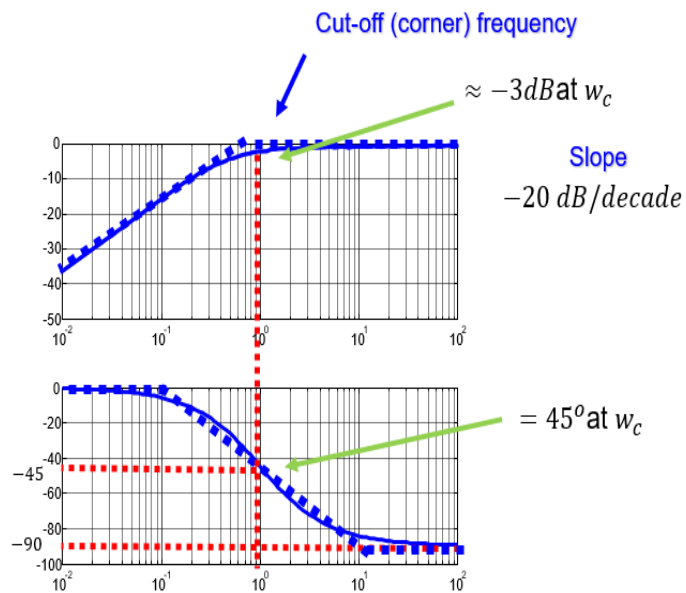


Figure 6. Continuous High Pass Filter

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

The cutoff frequency $\omega_c = 2\pi f_c$ (in rad/s) needs to be lower than the expected/desired frequency and higher than the unexpected/undesired frequency. In this example, we choose $f_c = 500$ which is between 50 and 5000Hz. Hence the parameters of “Transfer Fcn” block need to be set as below.

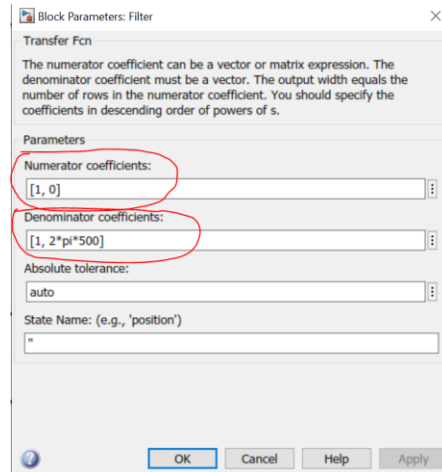


Figure 7. High Pass Filter Parameter Configuration

The responses comparison of unfiltered signal with filtered signal with 500 Hz cut-off high-pass filter is shown below. The low frequency portion has been attenuated, but the amplitude and phase shift are observed to be shifted from the original high-frequency component.

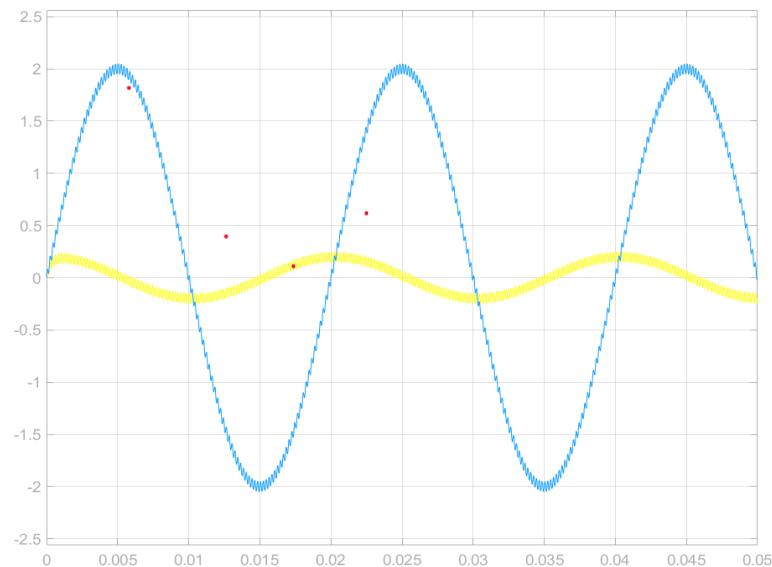


Figure 8. Comparison Results after Filter (disregard red dots)

In the following exercises, please apply the skills you learned from above instructions to complete the filter design and simulations.

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

Exercise 1: Continuous Low-pass filter

Now let us play with a new signal with three frequency components, $y = 5 \sin(2\pi \times 100t) + \sin(2\pi \times 2000t) + 0.5\sin(2\pi \times 8000t)$.

Modify the '*MATLAB Function*' block according to the new signal expression and design a continuous-time low-pass filter to **only** keep the lowest frequency portion (100 Hz) of signal.

What is the cutoff frequency that you will choose to design a low-pass filter?

Show the simulation results based on the filter design. Attach the Simulink model diagram, MATLAB Function code, and simulation results from the scope. (Make sure the parameters of the low-pass filter transfer function are visible).

- ★ You can make the transfer function equation visible by resizing the transfer function block
- ★ You can distinguish signals in the Scope with View => Legend

Exercise 2: Continuous High-pass filter

Now we still use three frequency components, $y = 5 \sin(2\pi \times 100t) + \sin(2\pi \times 2000t) + 0.5\sin(2\pi \times 8000t)$.

Design a continuous-time high-pass filter to **only** keep the highest frequency portion (8000 Hz) of signal.

What is the cutoff frequency that you will choose to design a high-pass filter?

Show the simulation results based on the filter design. Attach the Simulink model diagram, and simulation results from the scope. (Make sure the parameters of the high-pass filter transfer function are visible).

Exercise 3: Continuous Band-pass filter

We keep on using three frequency components, $y = 5 \sin(2\pi \times 100t) + \sin(2\pi \times 2000t) + 0.5\sin(2\pi \times 8000t)$.

Design a continuous-time band-pass filter (combine low- and high-pass filter. See lecture slides) to get the middle frequency portion (2000 Hz) of the signal.

What is the cutoff frequency that you chose for the low-pass filter?

What is the cutoff frequency that you chose for the high-pass filters?

Attach the Simulink diagram and simulation results from the scope (include 2 scope images: one zoomed out and one zoomed in).

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

In the next subsections, let us learn the discrete-time (digital) filter.

Again, we consider the example signal $f(t) = 0.05 \sin(2 \times \pi \times 5000t) + 2 \sin(2 \times \pi \times 50t)$, including two frequency components at 50 Hz and 5000 Hz with different amplitudes.

1.3 Digital Low-Pass Filter

The input of digital low-pass filter needs to be discrete, and hence, the signal is sampled and held. Zero-order sample-hold was introduced in the last lab and will be implemented here. Add the “discrete/Zero Order Hold” and “discrete/Discrete Filter” blocks into the model and connect them as below. The sample time of zero-order holder block is set to $T = 5.0 \times 10^{-5}$ second (sampling rate is 10 times higher than 5000Hz) for both “Zero Order Hold” and “Discrete Filter” blocks.

★ Optional: If you prefer making T a variable in Simulink, go to MODELING=> (DESIGN section) Model Explorer. In the Model Explorer window, you can select "Base Workspace" in the "Model Hierarchy" sub-window then go to Add => MATLAB variable. The center Contents sub-window will add a new variable "Var". double click on the columns of "Var" to change its name (to "T"), value, and data type as desired. You can then select "Apply" and exit out of the Model Explorer. In the Zero-Order Hold block, you can now set the Sample time as your variable "T" instead of "5*10^-5".

★ Newer versions of MATLAB/Simulink let you change the “Zero Order Hold” block sampling time but no longer lets you change the sampling time of the “Discrete Filter” block.

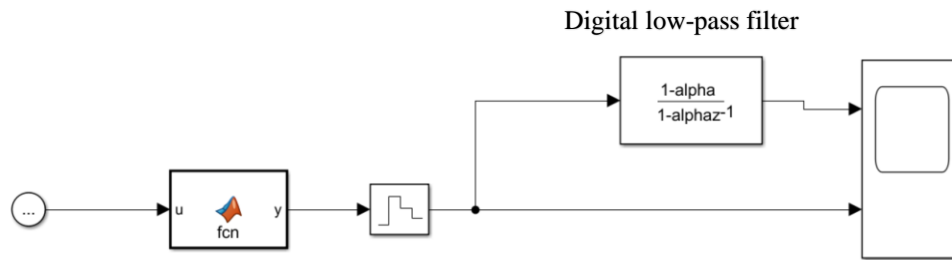


Figure 9. Model with Digital Low Pass Filter

The transfer function of digital low pass filter was introduced in the lecture notes.

$$\frac{Y(z)}{X(z)} = \frac{1 - \alpha}{1 - \alpha z^{-1}}$$

Figure 10. Digital Low Pass Filter Transfer Function

The cut off frequency $\omega_c = 2\pi f_c$ (in rad/s) needs to be higher than the expected frequency and lower than the unexpected. In this example, we choose $f_c = 200\text{Hz}$ that is between 50 and 5000Hz. The parameter α can be obtained by the following relationship between continuous- and discrete-time domains: (recall the lecture notes)

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

Continuous-time domain $\omega_c \rightarrow \alpha = e^{-\omega_c T}$ Discrete-time domain

In this case, $\omega_c = 2\pi \times 200 = 400\pi$ and $\alpha = e^{-400\pi \times 5.0 \times 10^{-5}} = 0.939$.

Hint: The command in MATLAB to calculate alpha is: `alpha=exp(-400*pi*5.0*10^(-5))`

Configure the digital filter as below.

Block Parameters: Discrete Filter

Discrete Filter

Independently filter each channel of the input over time using a discrete IIR filter. Specify the numerator and denominator coefficients in ascending order of powers of $1/z$.

A DSP System Toolbox license is required to use a filter structure other than Direct form II.

Main Data Types State Attributes

Filter structure: Direct form II

Data

	Source	Value
Numerator:	Dialog	[0.061]
Denominator:	Dialog	[1 -0.939]
Initial states:	Dialog	0

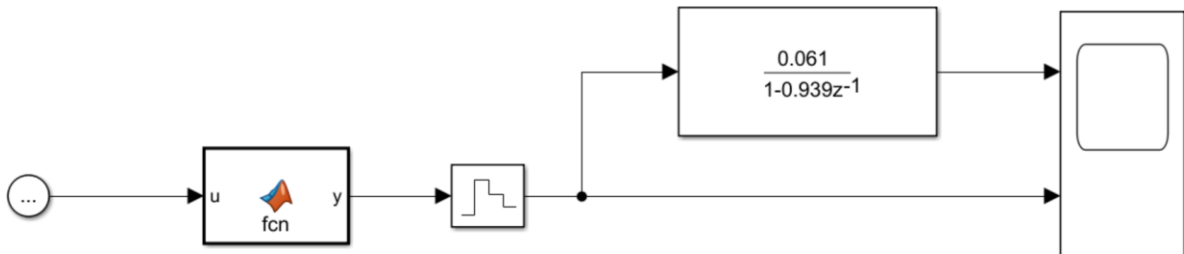
External reset: None

Input processing: Elements as channels (sample based)

☐ Optimize by skipping divide by leading denominator coefficient (a0)

Sample time (-1 for inherited): -1

OK Cancel Help Apply



MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

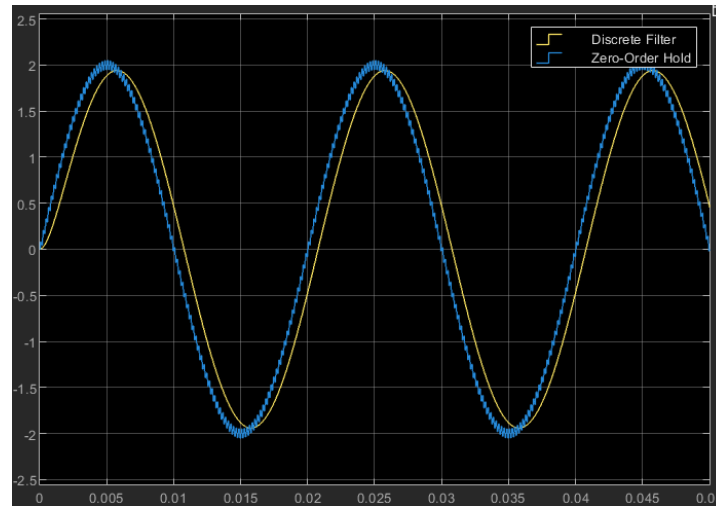
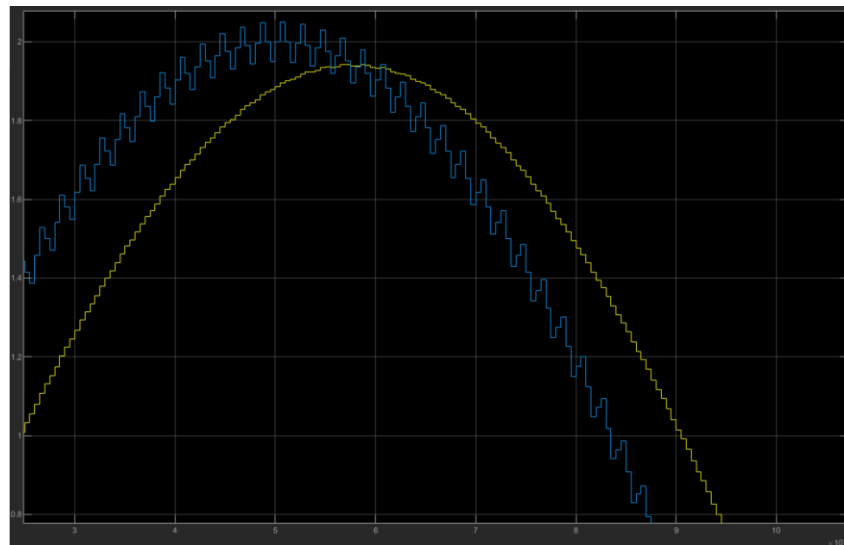
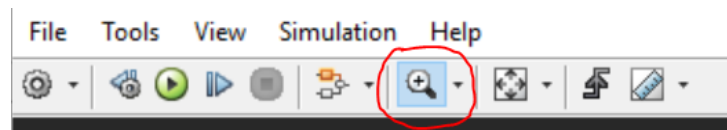


Figure 11. Comparison Results after Digital Filter

If we zoom in the figure by selecting magnifier icon and selecting the magnifying zone, then we can see sampled-and-holding signal after filter. The result is like the continuous-time filter case, but with sampled and holding signal.



1.4 Digital High-Pass Filter

The Simulink model is constructed as below to realize the digital high-pass filter by digital filter block. Then, we need to configure the parameters of the digital filter block.

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

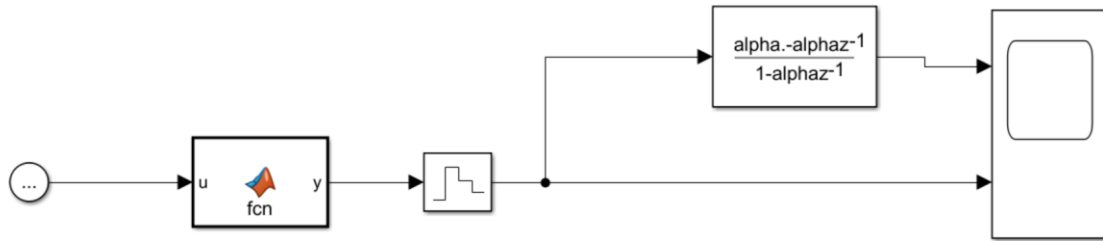


Figure 13. Digital High Pass Filter Models

For the high-pass filter, cut off frequency is selected to be $f_c = 1000\text{Hz}$ while the discrete transfer function of the high-pass filter needs to be modified as below. Thus, for high-pass filter, either one shown in Fig 13 can be adopt as the simulation model.

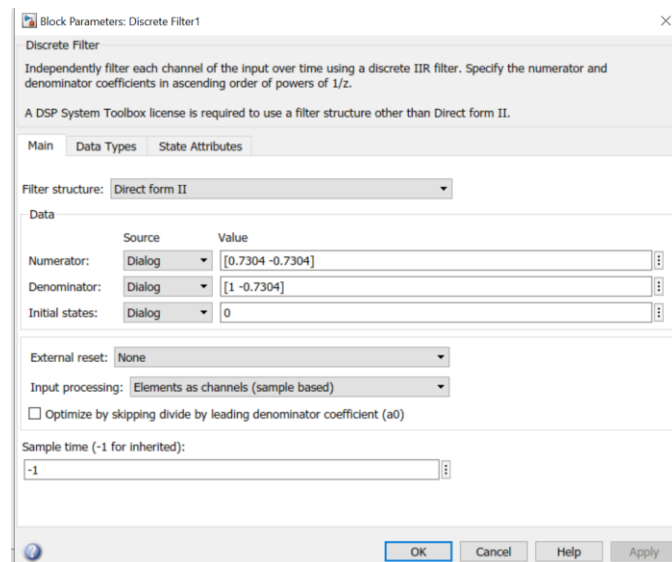
$$\frac{Y(z)}{X(z)} = 1 - \frac{1 - \alpha}{1 - \alpha z^{-1}} = \frac{\alpha(1 - z^{-1})}{1 - \alpha z^{-1}}$$

Figure 12. Digital High Pass Filter Transfer Function

where α is determined by

$$\alpha = e^{-2000\pi \times 5.0 \times 10^{-5}} = 0.7304$$

Configure the digital filter block parameters as:



After running the model simulation, the comparison results were shown as below and low frequency part has been filtered.

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

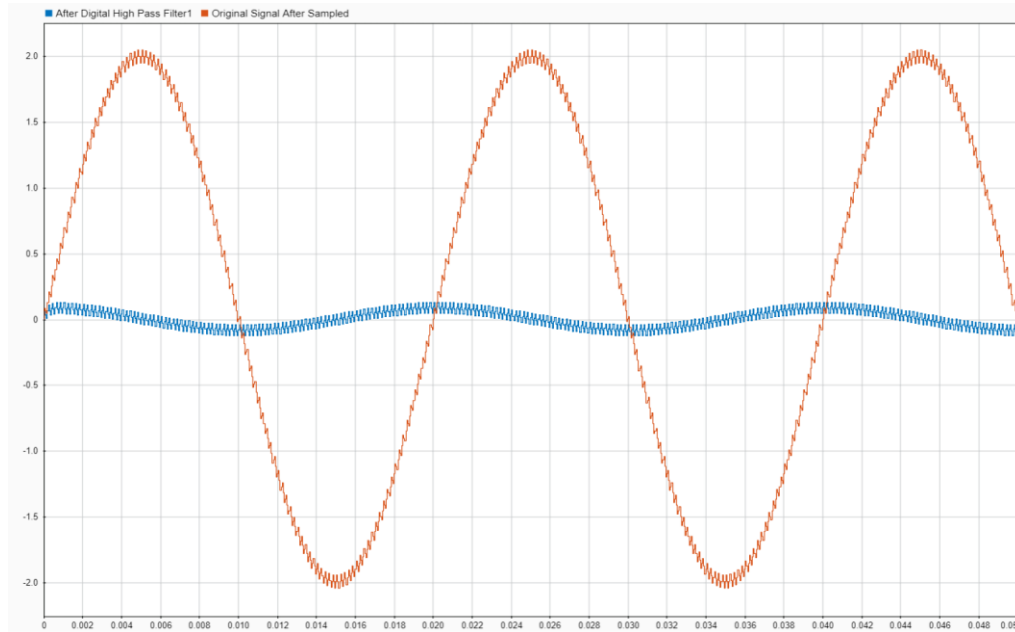


Figure 14. Comparison Results after Filter

Exercise 4: Digital Low-pass filter

The signal is $y = 5 \sin(2\pi \times 100t) + \sin(2\pi \times 2000t) + 0.5\sin(2\pi \times 8000t)$.

Design a discrete-time low-pass filter to get the lowest frequency portion of the signal using the following specifications:

Let us choose the cut-off frequency as 500 Hz, change sampling time of zero-order holder block to $T = 1.0 \times 10^{-5}$ second, what is the value of filter parameter α ? Show your calculation equations and α value.

Set stop time to 0.02 and show the simulation results based on the filter design. Attach the Simulink model diagram, MATLAB Function code, and simulation results from the scope. (make sure the low-pass filter transfer function is visible).

Exercise 5: Digital High-pass filter

The signal is $y = 5 \sin(2\pi \times 100t) + \sin(2\pi \times 2000t) + 0.5\sin(2\pi \times 8000t)$.

Design a discrete-time high-pass filter to get the highest frequency portion of the signal using the following specifications:

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

Let us choose the cut-off frequency as 5000 Hz, keep sampling time of zero-order holder block to $T = 1.0 \times 10^{-5}$ second, what is the value of filter parameter α ? Show your calculation equations and α value.

Set stop time to 0.02 and show the simulation results based on the filter design. Attach the Simulink model diagram, MATLAB Function code, and simulation results from the scope. (make sure the high-pass filter transfer function is visible).

Section 2. Introduction to Arduino

Over the years, Arduino has developed a variety of processor boards with different capabilities and functionalities. For instance, different MKR boards are designed for different communication protocols, such as Wi-Fi, Bluetooth and so on. The most popular board for beginners is the Arduino UNO based on the ATmega328P processor from ATMEL. UNO is the most used and robust board for the entire Arduino family, and it is perfect for getting started with the Arduino platform.

Processor boards released after the UNO uses more advanced microprocessors with more memory and enhanced input/output features, but for the most part they use the same pinout arrangements and work with existing add-on boards, called Shields or MotorCarriers, and various add-on components such as sensors, communication chipsets, and actuators.

2.1 Arduino Uno

For this class, we use Arduino Uno which is included in the Arduino starter kit. You are also encouraged to learn more info of MKR1000 and other boards.

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO



Figure 15. Arduino Uno

The pins on the Arduino boards can be used to control and communicate with external components, such as LEDs, motors, or sensors. Pins can be configured either as INPUT or OUTPUT. Pins configured as INPUT enable the board to read values from external components attached to the pin, such as the state of a button. When pins are configured as OUTPUT, the board will be able to control and communicate with actuators, such as lighting up an LED.

Pins configured as OUTPUT are said to be in a low-impedance state. In this configuration, the pin can provide current to other circuits. DC current pin can source up to 20 mA (50 mA for 3.3V pin) of current per I/O pin to other devices and circuits. This means that you cannot, for example, drive a motor directly from a pin because the current will not be sufficient. You will need to use other drivers (amplifier) to achieve different tasks like lighting up high power LEDs, driving motors, or activating relays.

2.2 Install the Arduino IDE

Arduino IDE is the most used programming interface for Arduino. The classic Arduino IDE can be downloaded from <https://www.arduino.cc/en/Main/Software>. Choose the version corresponding to the operating system in your computer and follow the instructions. More details about installation in different OS can be found in the link below.

<https://www.arduino.cc/en/Guide/HomePage>.

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

By installing the necessary files to compile programs for the Uno, you will also install the drivers needed for your computer to identify the board and communicate with it.

★After the IDE installation, popup windows might come up requesting to install Adafruit Industries LLC Ports and Arduino USB Drivers. These appear to be some of the drivers you need below.

- **Install Board Driver:** To be able to run programs on Arduino board, we need to install drivers of the board. Choose **Board** in the **Tools Menu** and then choose **Board Manager** on the top of the list of Boards. The following window will pop up. In the drop-down list of **Type**, choose **Arduino**. Further filter your search with key words ‘**Uno**’, then available drivers supporting Uno will show up. Install it for your first use in your personal computer.

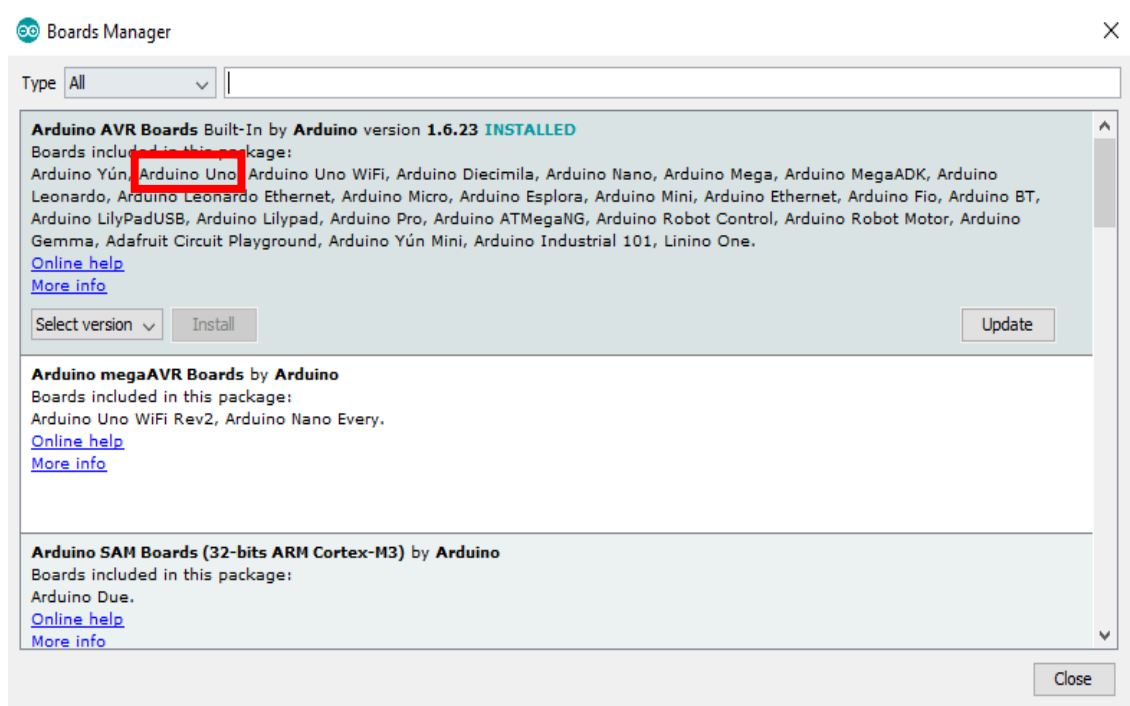


Figure 16. Board Manager of IDE

- **Install the Library:** Similarly, the **Library Manager** can be chosen in the **Sketch** menu (Include Library => Manage Libraries); see Library Manager below
More info can be found at <https://www.arduino.cc/en/Guide/Libraries>

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

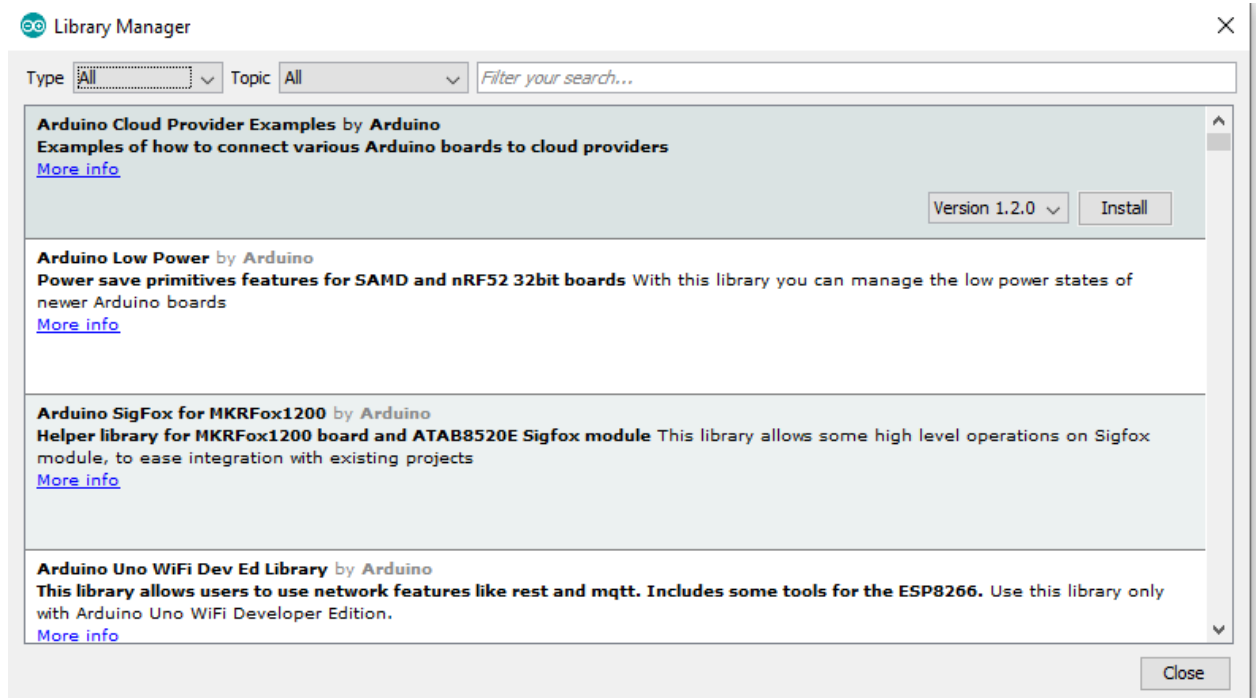


Figure 17. Library Manager

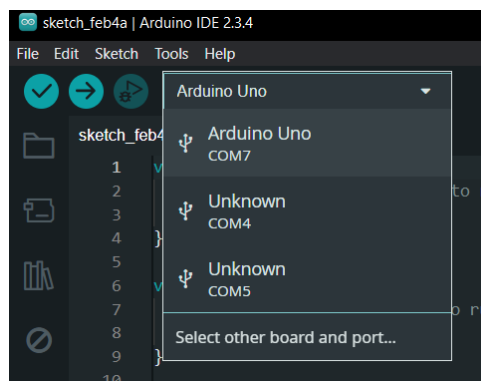
The library ‘**bridge**’ will be used for motor control in next lab. Type ‘bridge’ in search bar and install the bridge library.

Now connect your Arduino board to your computer!

Note: Operating systems in PCs handle the connection of new USB devices differently, so the COM number varies in different PCs and operating systems.

2.3 Programing by IDE (This section will be done during the lab time and used to verify you have installed the software successfully.)

(a) Double check Uno board is selected. Navigate & select ‘Tools/Board/Arduino AVR Boards/Arduino Uno’. Then on the main screen click the USB drop down menu and select the Arduino COM#.



MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

(b) Choose the 'File/Examples' and then go to 'Basics/Blink'. Then the Blink program code should pop up and is shown as below.

```
Blink$
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.


  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

Figure 18. Blink Program

The mode was setup to output so that we can set a pin to the desired state (on or off) for lighting the LED. Try to tune the value in the command 'delay()' to see the changes.

Click 'upload' icon , if the blinking can be observed, then your Uno Board can be programmed well by IDE.

Section 3. Communicate between Arduino and MATLAB/Simulink

3.1 MATLAB and Simulink Package Installation

Before installing the packages, make sure right click on MATLAB icon and

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

(Open MATLAB by ‘run as administrator’) (If you can... Might not be able to do on lab computers, it is possible to execute programs to the Arduino without ‘run as admin’)

In the lab00, the following MathWorks products are required to install. Please double check if you have all installed (click Add-Ons in MATLAB’s home tab then click “Manage Add-Ons” in the upper right-hand corner to see what you currently have installed) .

MATLAB®, Simulink®, Control System Toolbox™, Curve Fitting Toolbox™, DSP System Toolbox™, Image Processing Toolbox™, Instrument Control Toolbox™, Optimization Toolbox™, Signal Processing Toolbox™, Simscape™, Simscape Multibody™, Stateflow® and Symbolic Math Toolbox™.

After completing the installation, you will need to install several Add-ons that are used to support Arduino hardware. Select Add-Ons from the MATLAB Tool Bar as shown in the image below and click on Get Add-Ons.



Figure 19. Tool Bar of Matlab

Note: Install two packages in Add-Ons in the exact order provided below.

- MATLAB Support Package for Arduino Hardware: Acquire inputs and send outputs to Arduino boards and connected devices (you may be asked to do configuration after installation, choosing “setup now” will automatically bring you to the Hardware setup window in section 3.2).
- Simulink Support Package for Arduino Hardware: Run Simulink models on Arduino boards

3.2 Connection Setup for Matlab and Arduino

We will set up the Uno board in MATLAB by typing the following command to start:

```
>> arduinosetup
```

This will open the Hardware Setup window. You should select USB from the available options and click Next.

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

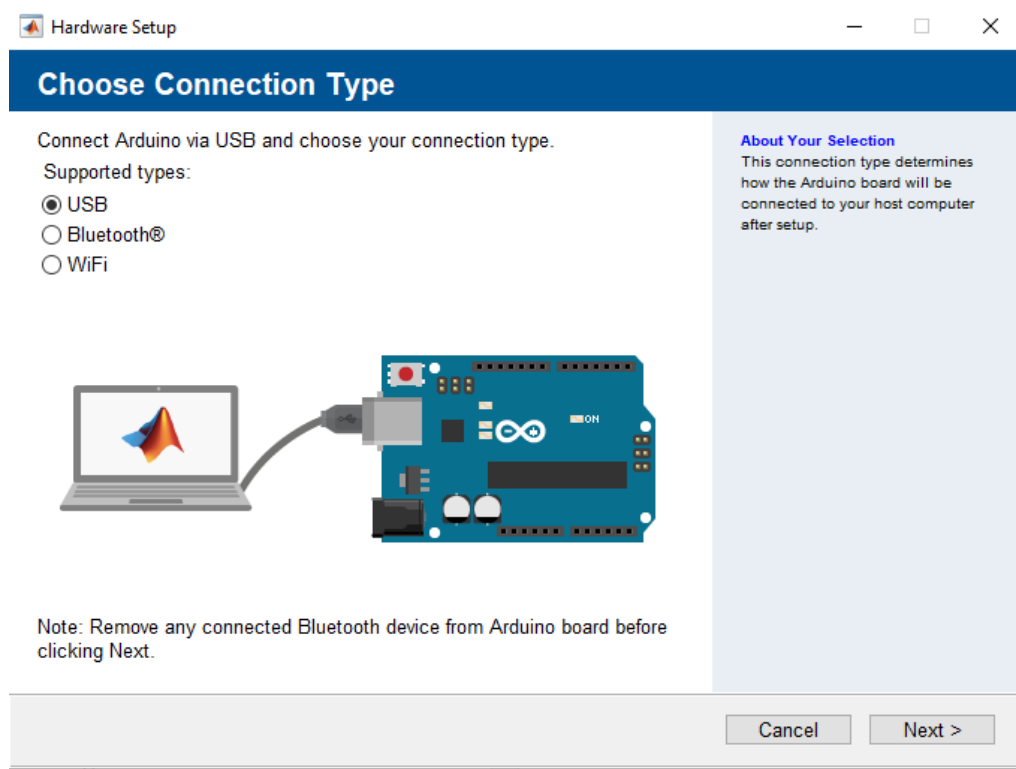


Figure 20. Hardware Setup Window

Next, you will select your Arduino Board Type, the COM# port through which you will communicate with it, and the external devices you will need to access. In this class we will use one DC motor and one rotary encoder. Set your board to Uno. Set your communication port to the COM port. Check that the *RotaryEncoder* box as that in Figure 21. Click **Program** (cancel any firewall popups if needed).

This may take several minutes to complete. After the libraries have successfully loaded to the Uno board, click Next and complete **test connection**. You are uploading firmware to the Uno board that will enable it to communicate back to your computer, where MATLAB will be executing commands with the help of the information that is captured by sensors and sending it back to actuators via the same communication channel.

Click next and finish to end the setup. Repeat for the Simulink support package (You do not need to download & install the PixyMon v2 package, so just click “Next” on the Third-Party/PixyMon installation pages).

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

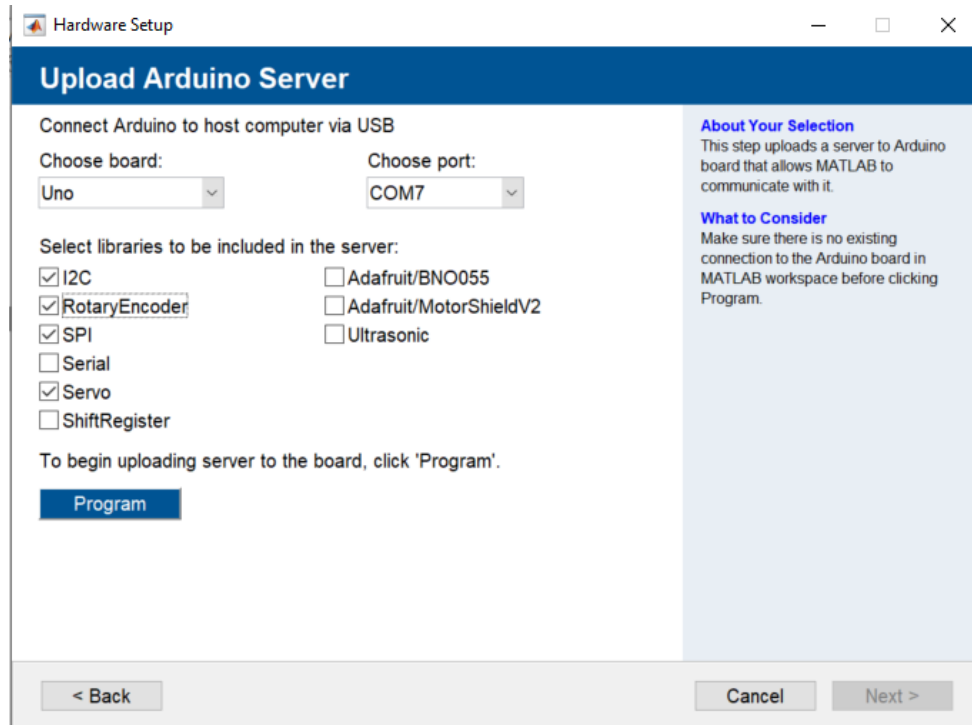


Figure 21. Upload Server Window

3.3 Communication between Simulink and Arduino (This section will be done during the lab session and used to verify your Simulink can communicate with Arduino. A spare board will be distributed in class)

Exercise 6:

After the hardware setup process is done, we need to check the status of Arduino.

Keep Arduino plugged in, and type 'arduino' in MATLAB command window. Then you will get something like this. Attach a screenshot of your own command window's output.

```
>> arduino

ans =

    arduino with properties:

        Port: 'COM7'
        Board: 'Uno'
        AvailablePins: {'D2-D13', 'A0-A5'}
        AvailableDigitalPins: {'D2-D13', 'A0-A5'}
        AvailablePWM Pins: {'D3', 'D5-D6', 'D9-D11'}
        AvailableAnalogPins: {'A0-A5'}
        AvailableI2CBusIDs: [0]
        Libraries: {'I2C', 'RotaryEncoder', 'SPI', 'Servo'}
```

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

Next, we try to illuminate the LED light via Simulink.

Step 1: Choose the **Pulse Generator** Block from Sources library and the **Digital Output** block from Simulink Support Package for Arduino Hardware library (in the “Common” subfolder). Connect those blocks as below.

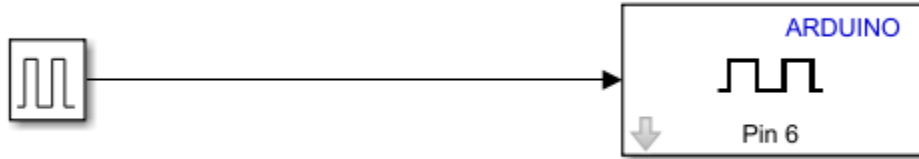


Figure 22. Simulink Model for LED Light

Step 2: Edit the parameters in Pulse Generator block as below. The amplitude is 1 and the period is 2 secs with a 50% pulse duty-cycle.

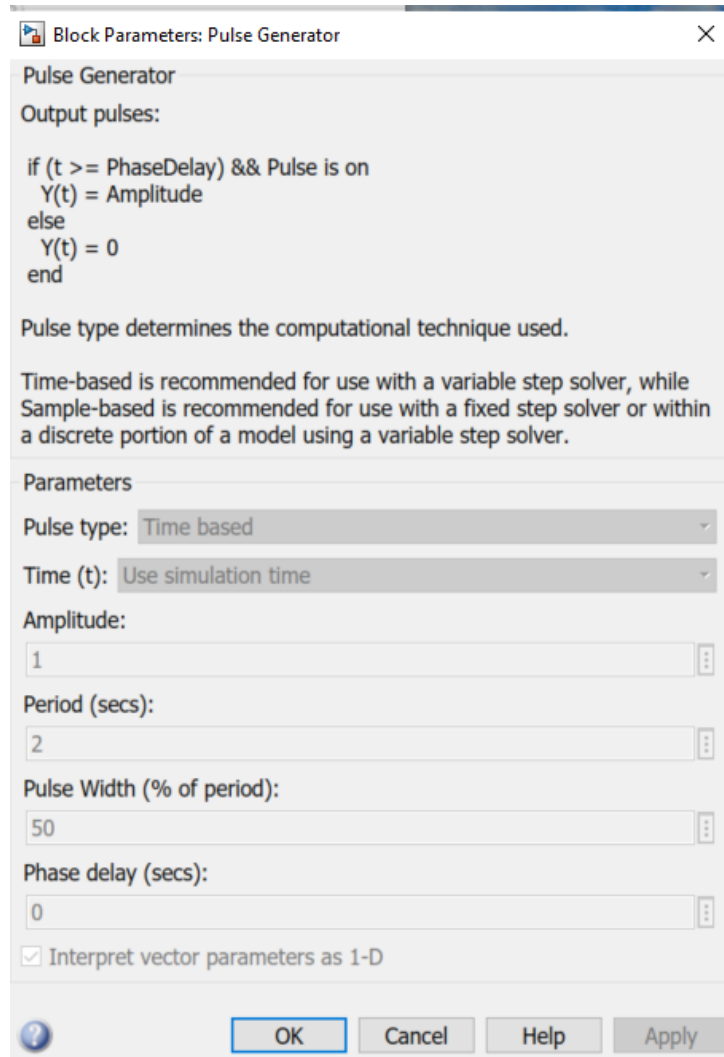


Figure 23. Pulse Generator Block

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

Step 3: Change the Pin number in the Digital Output block. The pin num of Uno LED should be **13** for illuminating LED lights.

Step 4: Change stop time to 'inf' for the simulation time.

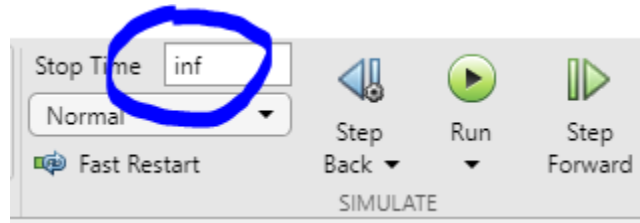

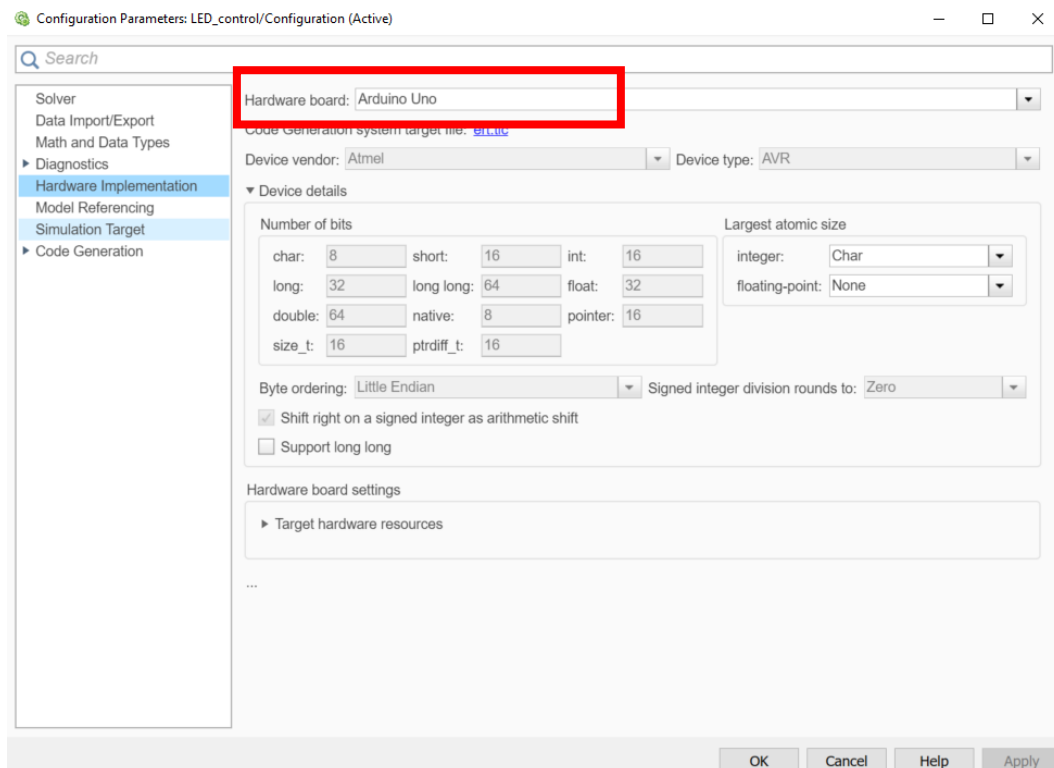


Figure 24. stop time

Step 5: Single click the icon  in the Simulink MODELING tool bar to configure parameters of the model. In the Configuration window, configure the following items:

- In the Hardware Implementation, choose Arduino Uno as the Hardware board.

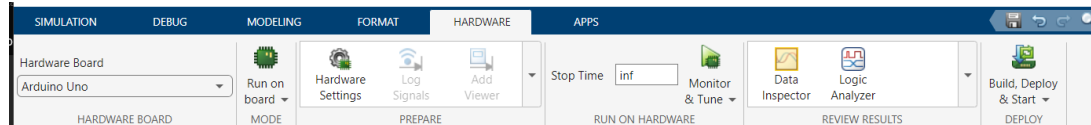


- In the Hardware board settings, go to **external mode**, choose Serial (XCP on Serial).
- If you have an older version of MATLAB, In hardware board setting, make sure Simulink IO is checked with **Enable Simulink IO**

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

- In the Slover, set up the **Fixed-step** size as ‘0.01’.

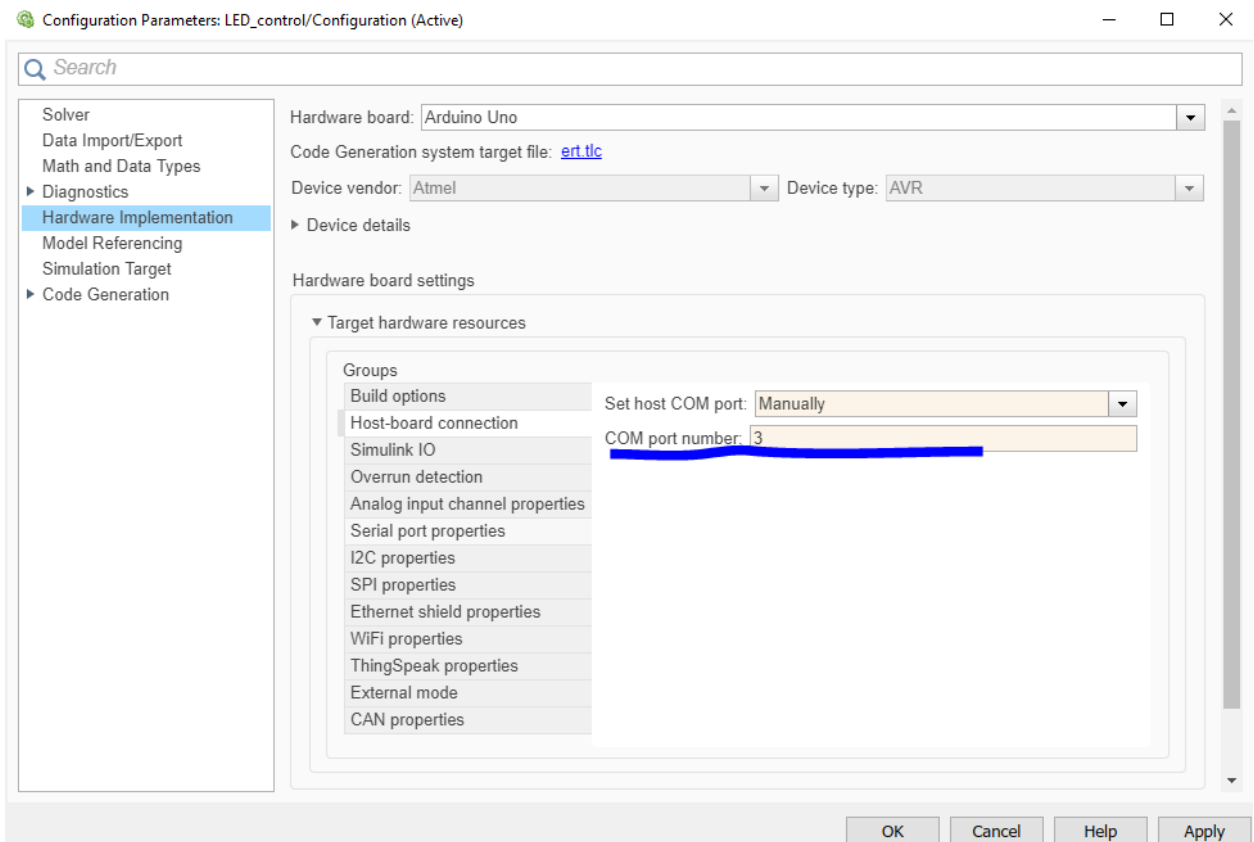
Step 6: Now the basic model has been created. In the **HARDWARE** tab, check in the **MODE** subcategory that you are in “Run on board” mode. In the **RUN ON HARDWARE** subcategory, click the “Monitor & Tune” button to launch the model.



If the build process completed successfully, the LED light in Arduino Uno board should blink. The LED is ON for 1 second and OFF for 1 second.

Debug Notes:

- If some errors with port connection pop up, we can go to Host-board connection and manually select COM port based on your own case.



- The **HARDWARE** tab has its own “Hardware Settings” in the **PREPARE** subcategory. Check that these are consistent with the **MODELING** tab’s settings that you set.
- Sometimes closing all applications, verifying you can still open & run Arduino IDE’s blink code, and reopening MATLAB or running MATLAB “as Admin” helps (also retyping ‘arduino’ in MATLAB command window might help).

MAE/ECE-5320 LAB 03: FILTER DESIGN AND INTRODUCTION TO ARDUINO

- In the HARDWARE tab you can use other modes and/or deploy your code (see the first link on pg. 1 for explanations) to run your code on the Board. The run button in the SIMULATION tab doesn't work because it's only sending commands to the simulation not to the board.
- It takes a while for Simulink to build, compile, generate, and execute code. You can look at the bottom left corner of the Simulink window to see its progress. Simulink's Diagnostic Viewer (click "view" in bottom center of Simulink window) can help with debugging too.
- Are the pulse generator settings correct?

Exercise 7: Arduino illuminating LED on Uno

Attach a photo of your own Arduino board with LED illuminated.