EEE 304 Lab: Filtering with an Embedded Computer (Extra Credit)

1. Bill of Materials

In this laboratory experiment, you will build and test a simple Digital filter on an embedded platform. The idea is to generate a few tones, filter them and observe the output by listening to the sound generated by the tones on a speaker/headphone.

In order to carry out this experiment you will be required to obtain the following:

Table 1: List of parts to be purchased

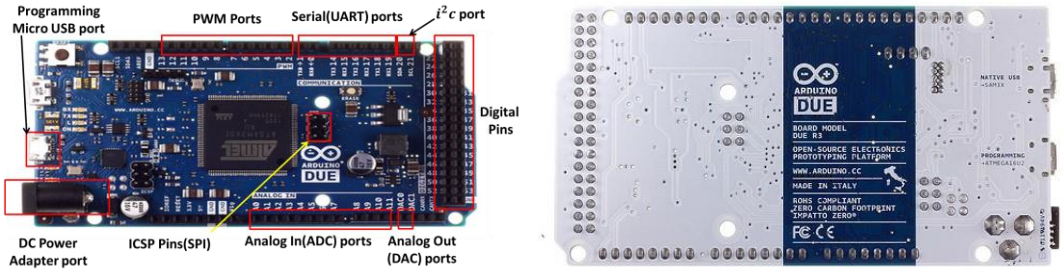
Item	Model	Quantity
Arduino	Due	1
Micro USB cable	Any android phone charger	1
Headphones	Any headphone used for listening to music. Preferably one that has 20-40 Ohm impedance.	1
Crocodile clips (optional)	Only two clips are needed	2
Digital Multimeter (optional)		1
Set of wires	Male to Male Jumper wires	10
Breadboard	Any breadboard	1

You will require at least the quantities mentioned in the table to successfully carry out the experiment. The hyperlinks attached to the **item names** are for example only and you can purchase these items from anywhere you like.

You will also need MS Visual Studio to work as a compiler, instructions on how to install is posted on Blackboard.

2. Introduction to Arduino Due

The Arduino Due is a microcontroller board based on a 32-bit ARM core microcontroller (Atmel SAM3X8E ARM Cortex-M3 CPU). It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), an 84 MHz clock, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button. Unlike other Arduino boards it operates at 3.3V. It can be powered using a micro USB port (programming port) or AC-DC adapter.



(a) Top view of Arduino Due (b) Bottom view of Arduino Due

Figure 1: Top and bottom view of the Arduino Due.

The 12 analog inputs of the Arduino Due have a voltage range of 0-3.3V. It can support a resolution of 12 bits on each ADC channel. On the other hand, the two DAC channels have a voltage range of $\frac{1}{6}(3.3)$ to $\frac{5}{6}(3.3)$ V. You must keep this limitation in mind while doing

experiments with the DAC on the Arduino Due. The resolution of the DAC is again 12 bits on each channel.

The Due has 4 serial ports (0-3). Serial port 0 is directly connected to the USB programming port and any data can be passed down or read from the Due using that link and any terminal program on the computer (the Arduino IDE has such a program for convenience). All the serial ports operate at 3.3V as high and 0V as low.

All the necessary pinouts of the Arduino Due are shown in Fig. 1. For more details, please visit https://www.arduino.cc/en/Main/arduinoBoardDue.

3. Software Preparation

3.1 Installing the Simulink Toolbox for Arduino Due

Start by downloading the Arduino IDE software from: http://arduino.cc/en/Main/OldSoftwareReleases Preferably use the 1.5.8 software, the BETA is actually quite stable. Unzip the Arduino onto a folder. Next, plug in the micro USB end of a cable to the programming port (the one closest to the power jack) and the other end of the cable to the PC. Install drivers for the board using device manager; the drivers can be found under the unzipped Arduino folder inside the Drivers folder. Once the drivers are installed, you should see it listed under a COM port inside Device Manager in Windows.

The next step is to install the Simulink Package. In the MATLAB command prompt, type the following and hit enter.

```
>> supportPackageInstaller
```

The support package installer window will appear as shown in Fig. 2.

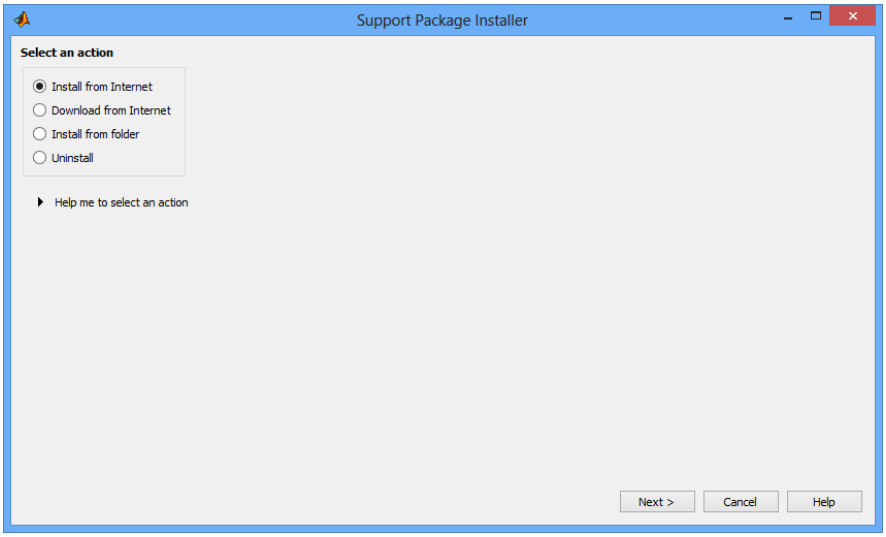


Figure 2: Simulink and Matlab Support Package Installer.

Check the Install from Internet radio button and click next and the following screen should appear (Fig. 3).

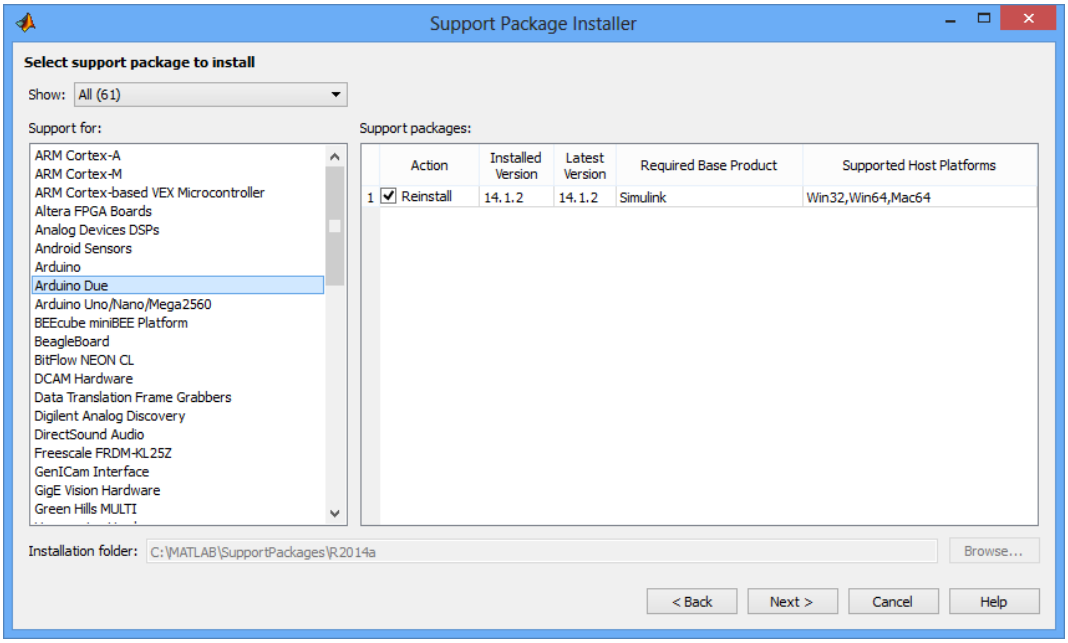


Figure 3: Choose Support Package to install.

Although the Figure shows "reinstall", if the Package had never been installed previously on the computer it will show up as "install". MATLAB 2016b will have two options; you can select both of them. The next page will ask for a valid Mathworks account to log in (fig 4). If you do not have one, you can create it for free using this link: [Matlab Account Creation]

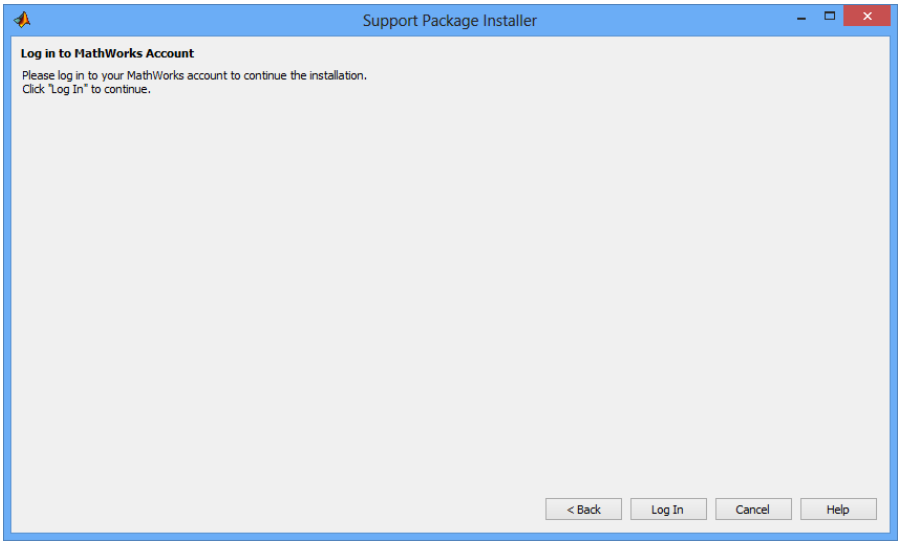


Figure 4: Log in to Mathworks Account.

The next few screens will ask for permission to install and choose installation folders. Complete these steps until the installation process ends.

3.2. Setting up Model for Running on Arduino Due

Start by creating a new Simulink Model. Then, under "Tools", select "Run on Target Hardware" and click on "Prepare to run" as shown in Fig 5.

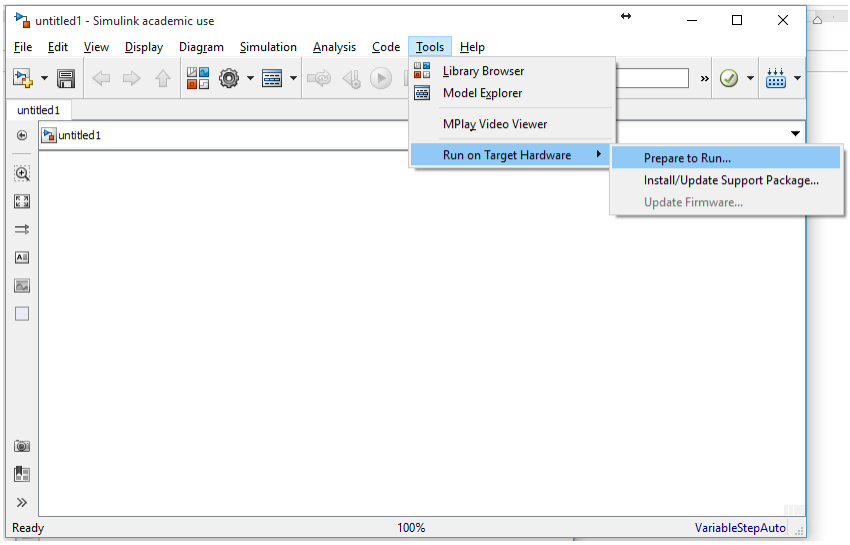


Figure 5: Prepare to run on target hardware.

A new window will appear where you will select "Arduino Due" as shown below.

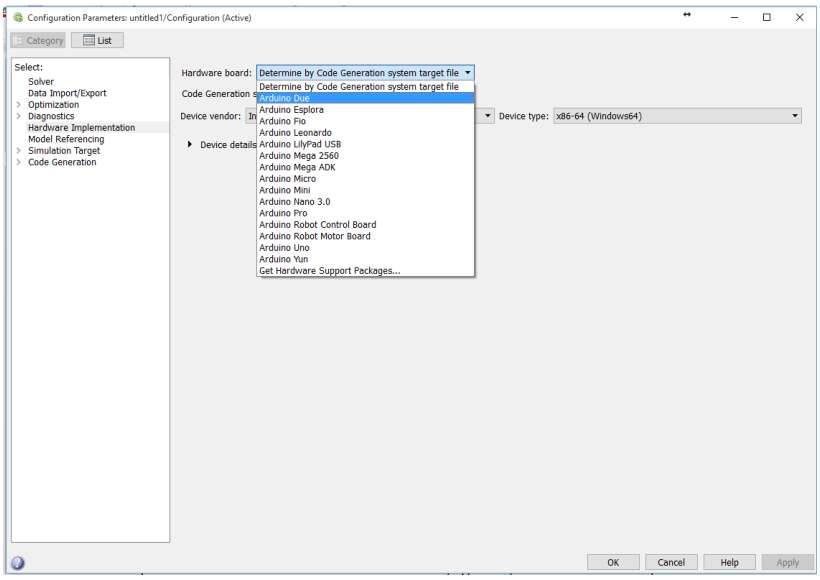


Figure 6: Choosing Arduino Due as target hardware.

The window will change and bring up new menus. Under "Hardware Board Settings", select "Serial port properties". Change baud rate for all serial ports to its highest value, i.e. 115200. At this point, the Simulink Model is ready for code to be built and downloaded to an Arduino Due.

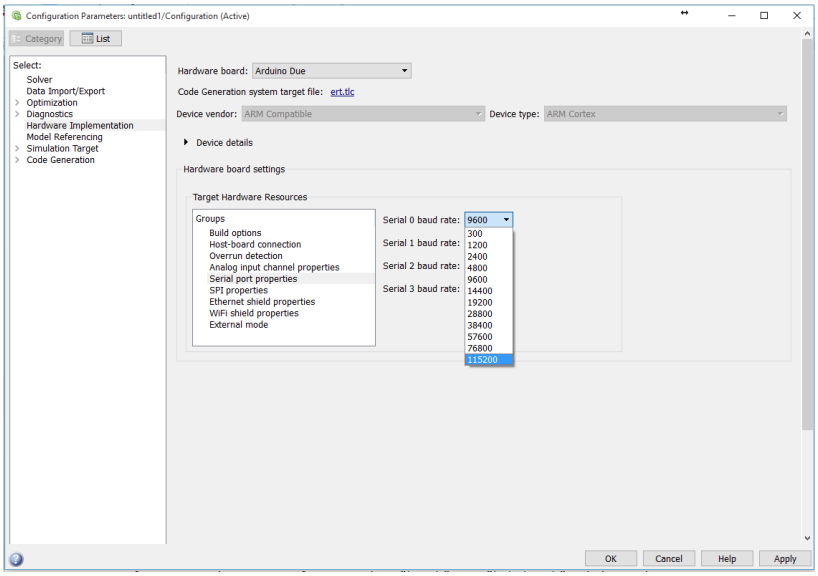


Figure 7: Change serial port baud rate.

The last bit is to setup the solver parameters for the model. On the left side of the same window, select "solver" and configure this page as shown below.

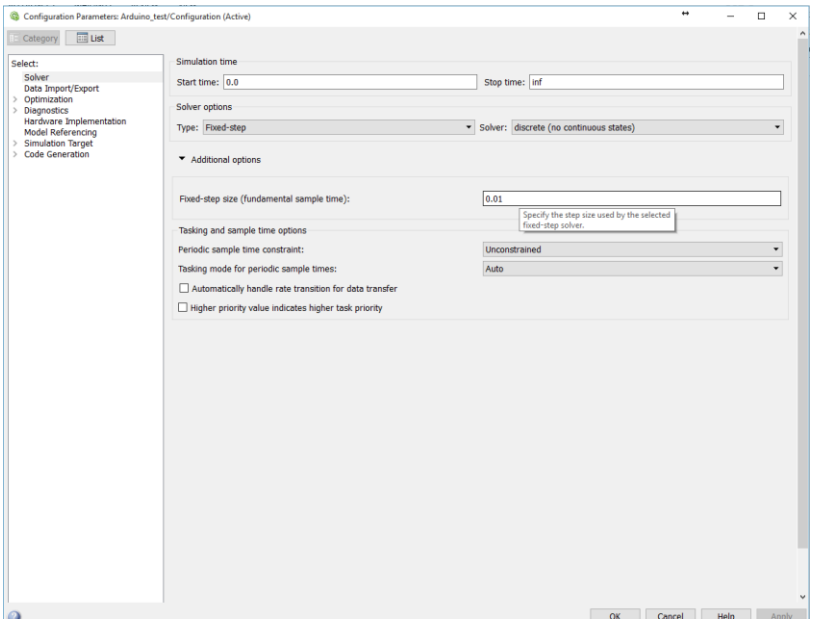


Figure 8: Set Solver parameters

The Fixed-step size is going to be the same as the sampling rate used in your models.

At this point, you can connect the Arduino Due to the computer using the USB to micro-USB cable. Windows should ask to install drivers, go through the prompts and install the drivers. To ensure that the drivers were correctly installed, right click on windows "Start" and select "Device Manager". There under "Ports (COM & LPT)" you should see an item listed as "Arduino Due Programming Port (COMX)", where x is a number e.g COM3. If this shows up, the Arduino is installed and ready.

In order to download code to the Arduino it must be connected to the PC.

4. Lab Procedure

4.1 Building the Circuit

Using the parts purchased, connect the circuit as shown in Fig. 9. If you are unsure of the pin labels or for any details, you may look up datasheet for the parts online.

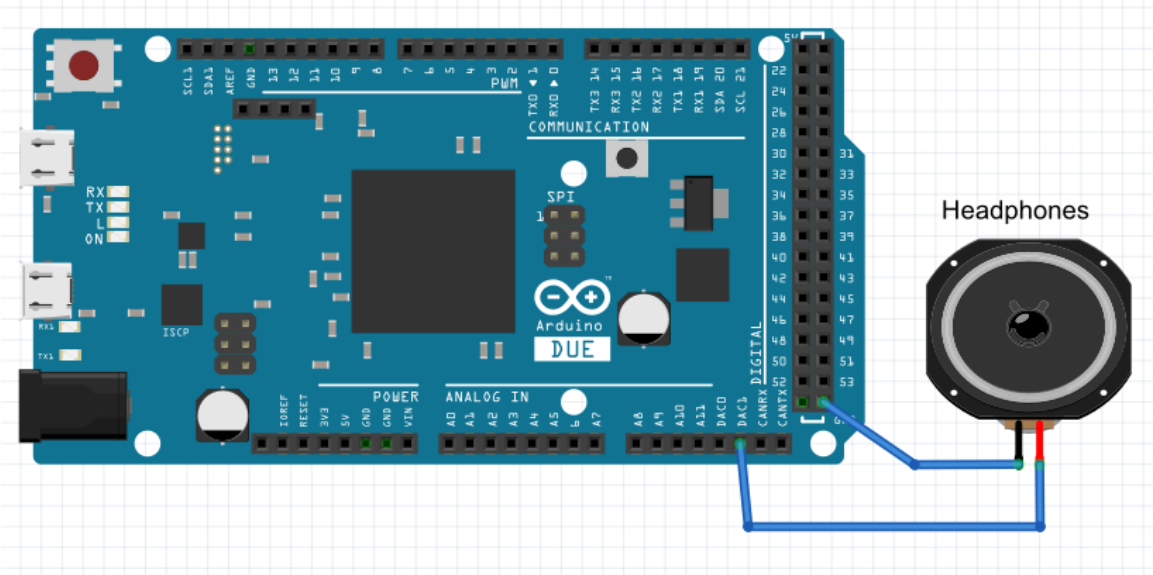


Figure 9: Building the circuit.

Before connection the headphone to the Due, it is a good idea to measure the impedance. Typical four pin connectors on stereo headphones have connections as shown in the diagram to the right. If your headphone has only three terminals then you probably do not have volume controls on them. Connect an ohmmeter (from a digital voltmeter) to the GND and Right or Left channel and measure the impedance. Any number between 20 – 40 Ohms is acceptable. **Do not** use a speaker with less impedance because it will burn out the DAC channel on the DUE. If you have purchased the headphone from the AMAZON link, you will see that the link specifies the headphone has about 32Ohm impedance.

In order to connect the headphone to the DUE, you may use two crocodile clips to connect to the headphone and have wires connect from the clips to the GND and DAC1 pin of the Arduino DUE. Alternatively, a make shift measure would be to wind thin wires around the headphone connectors and insert them to the DUE. Be sure to connect GND of headphone to GND of DUE and any left or right channel of the headphone to the DAC1 pin.

4.2 Building the Simulink Test Model

The Simulink model needed to perform this exercise will be provided to you. You do not need to build anything by yourself. Let us start with a simple test model. In your working directory, download and save the "**Arduino_test.slx**" model file, then open it.

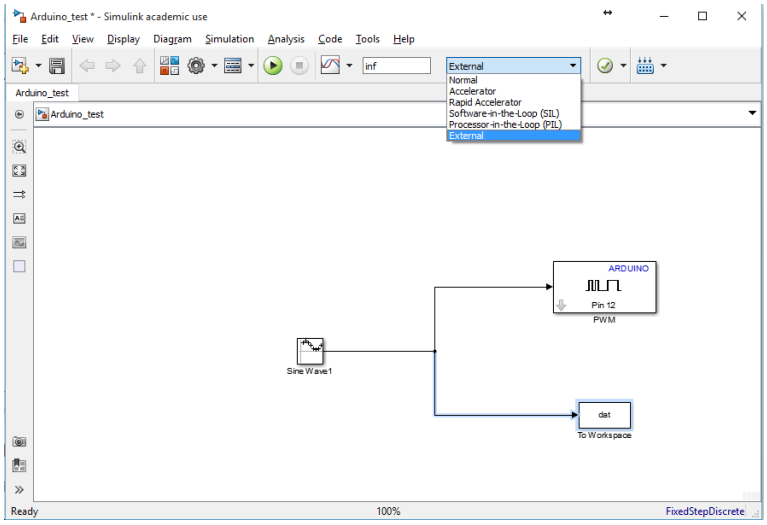
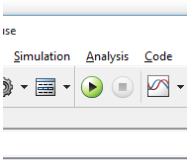


Figure 10: Arduino Test

As shown in Fig. 10, select "External" mode. The Simulink platform allows two modes of compiling code for Arduinos and other target hardware.

- External mode – The model runs on the target hardware but Simulink is constantly downloading and uploading data to and from the board. With this method, you can display and log signals as well as change certain parameters in the model. The disadvantage of this method is that high sampling rates are not supported. Also, the model will not run if the Arduino loses power and its memory.

- Normal Mode (Deploy to Hardware) – In this mode the code is built and downloaded onto flash permanently. No message passing occurs between the Arduino and Simulink, so no data display or parameter changing is possible. Very high sampling rates are supported under this mode and the code is permanently in the flash memory so the Arduino can be used when it is not connected to the computer.



For the first test case, we will be using external mode. Once external mode is selected in the model window, hit the green "Run" button that looks like a play symbol shown in the right figure. The model should run for about 10 seconds and then stop. While the model is running, the brightness of the onboard LED on the Arduino DUE should go between 0 and 100% in a sinusoidal timing pattern; this is due to the source being sinusoidal. Look for the LED next to the USB programming port. There are four LEDs named **RX**, **TX**, **L** and **ON**. The **L** LED is the one we used here which is connected to PWM pin 13.

When you run the model, the sine block generates a sinusoid between 0 and 255. The reason behind this is that the PWM output of the Arduino can accept numbers between 0 and 255 (8-bits) that represents an RMS voltage between 0-3.3V on the pin. That voltage causes the **L** LED to glow at different intensities.

When the model is done running, look at the MATLAB workspace. There should be a variable named "data". This variable has saved time in its first column and the value of the sinusoid in its second column. To plot this data, enter the following command in MATLAB

```
>> plot(data(:,1),data(:,2))
```

TASK 1: As part of your report, provide the plot you obtain from running this command.

4.3 Building the Simulink Experiment Model

Let us start by looking at what the code does in this Simulink experiment. Four sine waves are generated with frequencies 80, 200, 500 and 800 Hz and amplitude 0.5 peak for all. Since the Arduino DUE DAC can only output positive voltages. The sinusoids are shifted by 2 upwards first. The gain of 400 is applied to modify the output numbers from between 0-4 to about 0–1600. Note that the DAC block in Simulink takes value between 0-4095 since the DUE DAC is 12 bits. This number is converted by the DAC in the DUE to a range of 0.55V – 2.75V as mentioned earlier in this manual. Instead of converting the sinusoid to a range of 0-4095, we chose 0-1600. The reason behind this is that although the DAC can go up to 2.75V, it cannot deliver enough current. When the speaker with impedance 30 Ohm is connected, the DAC gets loaded and the maximum voltage it can output before it starts clipping is about 1.2V. 0-1600 range in the DAC output ensures the signal stays small and does not get clipped.

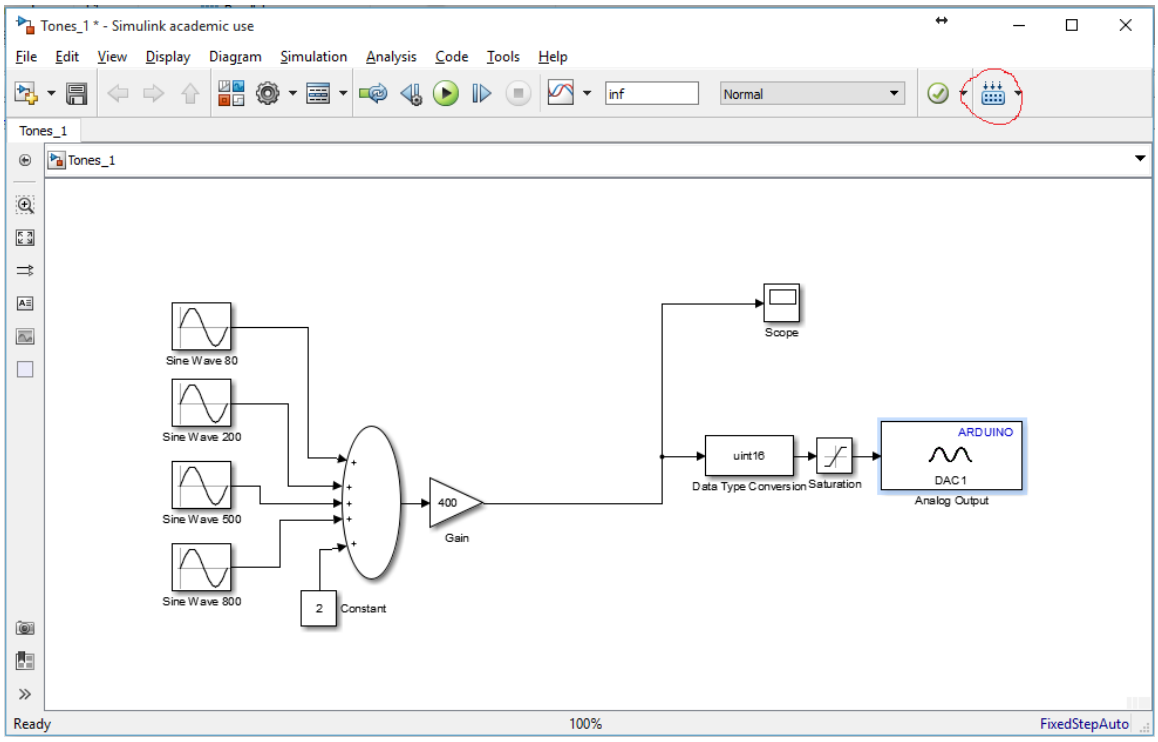


Figure 11: Generating sound

Open "**Tones_1.slx**" model file. Make sure your Arduino is connected to the PC via the USB cable. Then click on "Deploy to Hardware". This will compile code and download it onto the Arduino. Once the process is complete, if the headphone is connected correctly to the DAC1 pin you should be able to hear a tone.

TASK 2: As part of your report, answer the following question – Does the audio heard on the headphone sound like it is a single tone or a mixture of frequencies?

4.4 Filtering the Sound

Open and "Deploy to Hardware" the model file "**Tones_2.slx**" as shown in Fig. 12. In this case, we have applied a number of different filters to the same sum of sinusoids generated in the last section. The filters can be activated by grounding either or both of the Digital pins 7 and 8 (see Fig. 13).

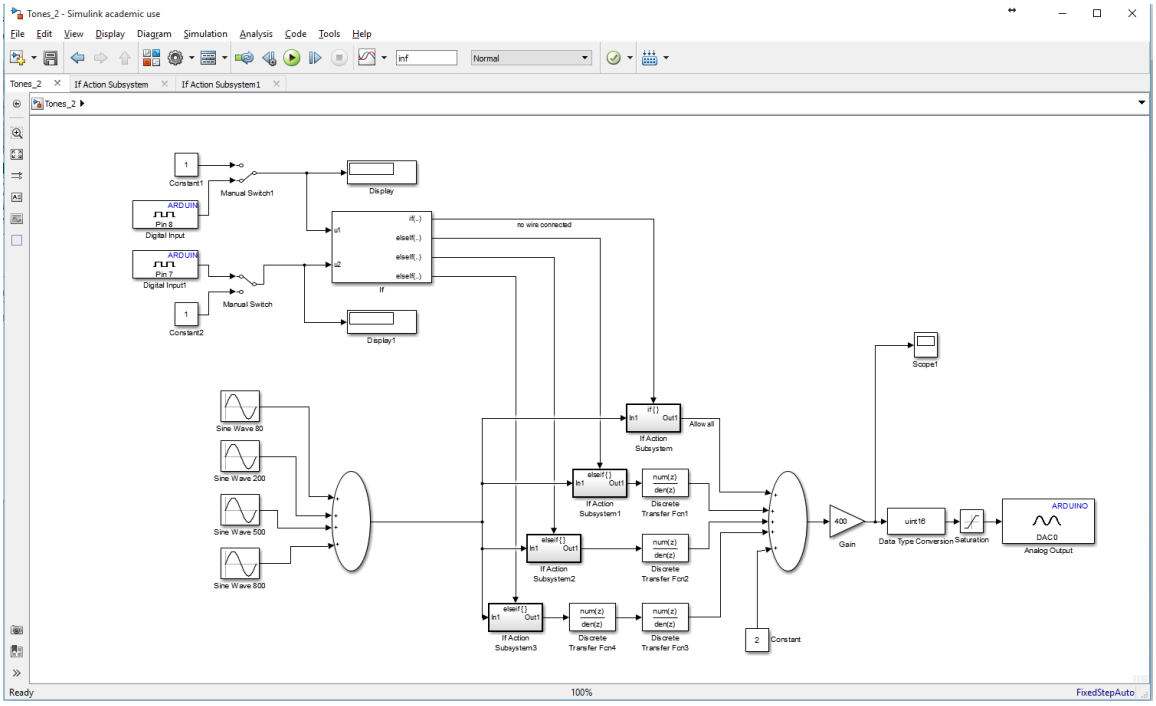


Figure 12: Filtering.

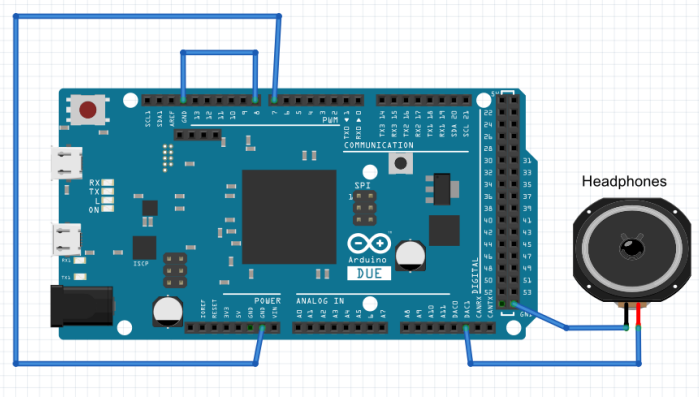


Figure 13: Connecting pin 7 and 8.

Fig. 13 shows the case when both pin 7 and 8 are grounded. By default the Arduino Digital Inputs read 1 (high) when no wire is connected, so by connecting a ground to the digital pin makes it read 0 (low). The actions that the model will perform is summarized in Table 2.

Table 2: Connecting pin 7 and 8 to activate one of the "If Action Subsystem" blocks

Pin 7	Pin 8	Action
Not Grounded	Not Grounded	The sum of sines will pass through the first "If Action Subsystem" from the top (pass through)
Grounded	Not Grounded	The sum of sines will pass through the second "If Action Subsystem"
Not Grounded	Grounded	The sum of sines will pass through the third "If Action Subsystem"
Grounded	Grounded	The sum of sines will pass through the fourth "If Action Subsystem"

The first "If Action Subsystem" from the top simply allows the signal to pass through without filtering and the sound can be heard on the headphone. This will make the audio sound exactly like you heard in the last section.

For the next three "If Action Subsystem", three different types of filters have been designed. Your task later on will be to identify which is which but for now you will know that **in no particular order**, there is

- A band stop filter to kill all signals so nothing is heard.
- A band pass filter to allow only low frequencies (80, 200Hz).
- A band pass filter to allow only high frequencies (500,800Hz).

Any time you make connections to pin 7 and 8 using Table 1, only one of the "If Action Subsystem" is activated allowing signal to pass through and getting filtered and producing a sound output.

TASK 3: As part of your report, connect wires to pin 7 and 8 as per the following table and report on what type of frequencies you can hear. Choose between **high**, **low**, **all** and **none**.

Pin 7	Pin 8	Frequencies heard
Not Grounded	Not Grounded	
Grounded	Not Grounded	
Not Grounded	Grounded	
Grounded	Grounded	

By now you may have found out what type of signal is being filtered. This is exactly how an equalizer filters signals when you listen to music. The last task in this lab is to design some filters

TASK 4: Before you rip everything apart, take a photo of your hardware set up (Arduino DUE, the headphone and wiring) and attach it to the lab report to serve as a proof of you completed the experiment.

TASK 5: As part of your report, design the following digital filters using the butter command in MATLAB. Provide MATLAB code and Bode plots of these filters. Choose order 4 and sampling rate of 5 kHz.

- A band stop filter to kill all signals so nothing is heard.
- A band pass filter to allow only low frequencies (80, 200Hz).
- A band pass filter to allow only high frequencies (500,800Hz).

EXTRA CREDIT LAB 1: LAB REPORT GRADE SHEET

Name _____

Instructor Assessment

Grading Criteria	Max Points	Points Lost
Template		
Neatness, Clarity, and Concision	2	
Lab Description	8	
Description of Assigned Tasks, Work Performed & Outcomes Met		
Task 1	8	
Task 2	8	
Task 3	8	
Task 4	8	
Task 5	8	
Lab Score (out of 50)	**Points Lost**	
	Late Lab	
	Lab Score	