

# RENSSELAER MECHATRONICS

## Lab 1: Blinking the LED

---

### Lab Objectives

- Install and verify Arduino software package for Simulink using a digital output to light a LED
- Communicate with the target board (Arduino) using external mode by changing the brightness of an LED with PWM

## Part 1: Arduino Toolbox Installation for Simulink

### Objective:

- Install and verify Arduino software package for Simulink using a digital output to light a LED

### Simulink Arduino Library Installation:

- Open MATLAB and type “supportPackageInstaller” in the MATLAB command window.
- The target installer window will appear. Select “Install from Internet”.
- If Arduino is not shown in the ‘Support Packages’ window, click Arduino in the ‘Support for’ window on the left.
- Select the Simulink package that supports Uno and Mega 256

[Run models on Arduino Uno, Mega 2560, Leonardo, and more boards.](#)

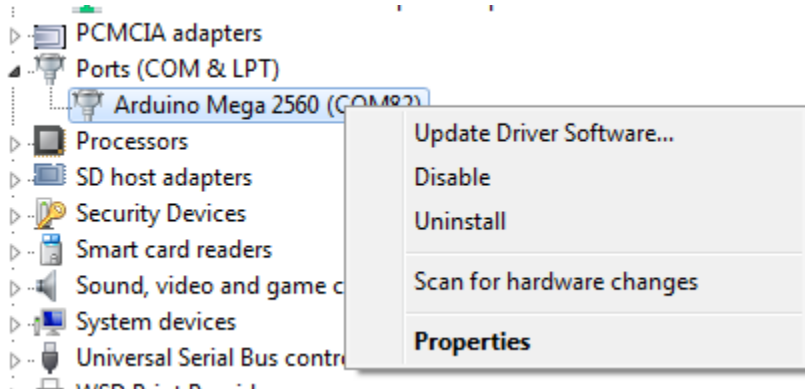
Simulink

Win32, Win64, Mac64, Linux64

- When prompted, log in using your MathWorks credentials. If you do not have an account you will need to create one.
- Follow the onscreen prompts to install the library.

### Arduino Mega 2560 Drivers' Setup:

- Connect your Arduino Mega board to the computer using the USB cable. Your computer will attempt to search for the necessary drivers online.
  - If your computer cannot find the right drivers do the following:
    - Go to Device Manager and look for your Arduino Mega board (Arduino Mega 2560). Your Arduino board will be under Ports (COM & LPT). Otherwise, it may be listed as “Unidentified Device” under ‘Other Devices’.
      - If you find an unidentified device, unplug the Arduino to see this device is removed from the list to verify that this unidentified device is your Arduino board. Plug you Arduino board back in.
    - Right click on Arduino Mega (or “Unidentified Device” ) and select “Update Driver and Software”
    - Select “Browse my computer for driver software”

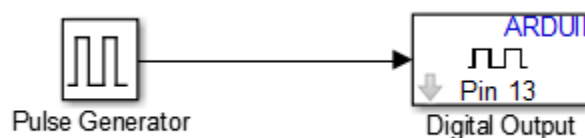
- Set the location to the installer folder for the Arduino software for example
  - C:\MATLAB\SupportPackages\R2015a\arduino-1.5.6-r2\drivers
- You will receive a prompt letting you know Windows cannot verify the publisher of the driver, select “Install this driver software anyway.”
- Windows will then install the necessary drivers for the Arduino Mega from this folder. A message will appear when the driver software has installed.
- If the driver still fails to install, download and install the newest Arduino IDE located here: <https://www.arduino.cc/en/Main/Software>
  - Install this software and the drivers when it requests it. This should overwrite any existing drivers that may be causing the trouble.
  - Uninstall the device:
 
  - Then unplug and plug the board back in. Windows should now automatically install the drivers. If not try the above procedure of manually installing the drivers using the installation path of the new Arduino IDE you just installed.
- If you are using windows 8 the driver installation might have a problem because the driver file is not digitally signed. [See http://mytechblog.com/tutorials/arduino/install-arduino-drivers-on-windows-8/](http://mytechblog.com/tutorials/arduino/install-arduino-drivers-on-windows-8/)

### How do I find the COM port for my Arduino Board:

- Go to Device Manager and look for your Arduino Mega board (Arduino Mega 2560). Your Arduino board will be under Ports (COM & LPT).

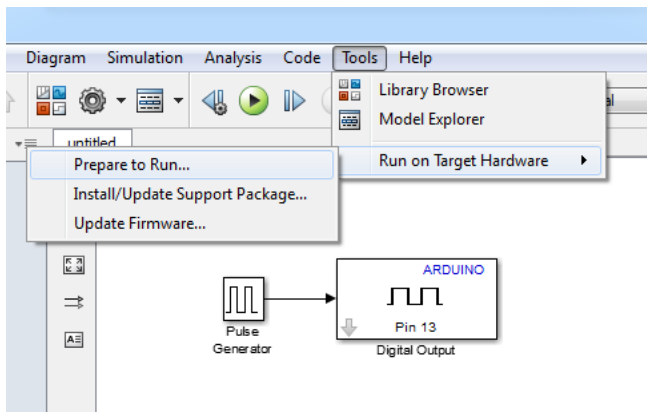
### Simulink Setup:

In order to ensure that your computer can properly communicate with the Arduino board, build and run the following Simulink diagram using the steps below:

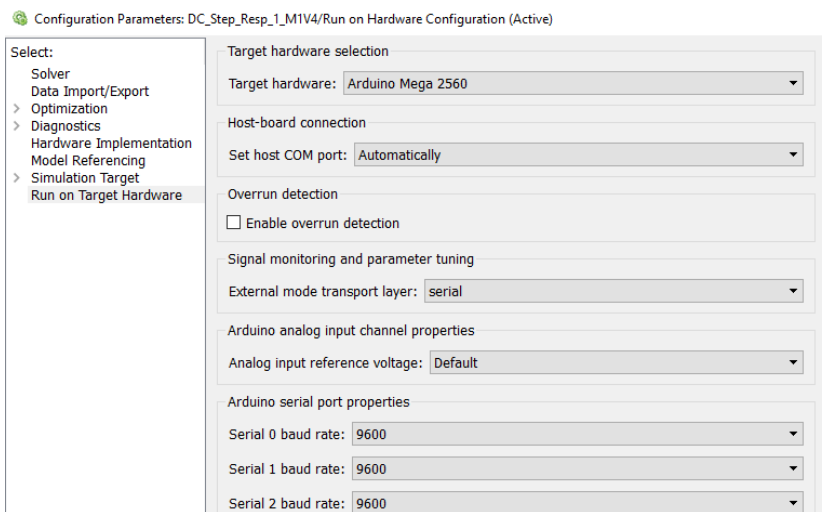



- 1) Open MATLAB then open a new Simulink model

- 2) Find and assemble the required blocks.
  - Pulse Generator is under View->Library Browser->Simulink->Sources
  - Digital Output is under View->Library Browser->Simulink Support Package for Arduino Hardware->Common
- 3) Change the output pin to 13, which is connected to the onboard LED. This is done by double clicking on the Digital Output block.
- 4) Double click on the Pulse Generator to change the amplitude to 1, the Period to 10, and the pulse width to 50%.
- 5) Under the tools menu go to 'Run on Target Hardware' and click 'Prepare to Run'. In the window that pops up you will need to select Arduino Mega 2560 in the drop down for Target Hardware.

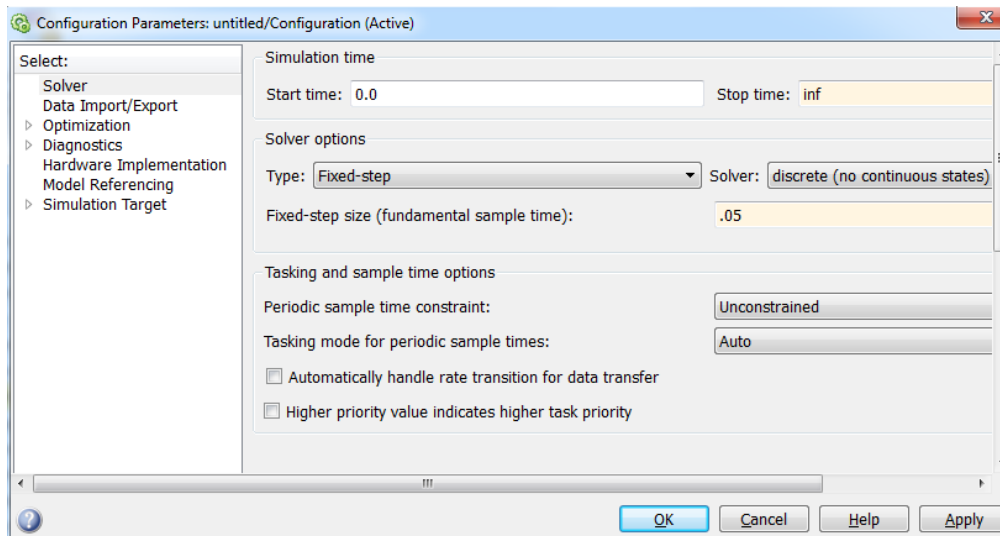


- “Set host COM port: “should be set to “Automatically”, click OK. You can specify the COM port your board if for some reason there are issues downloading the code.



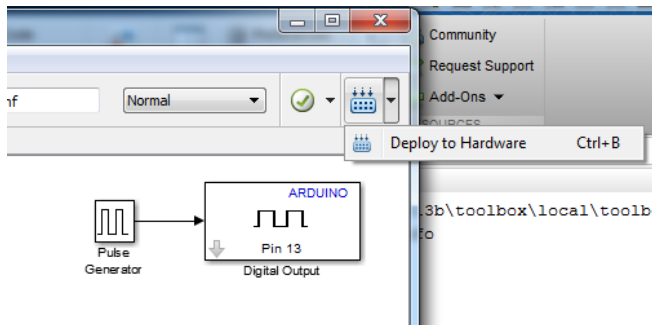
- Right click on the background of the Simulink window and select 'Model Configuration Parameters', or  in the top of the Simulink window.
- go to 'Solver' on the left hand menu and type 'inf' into the 'Stop time' spot so that the model will keep running. Make sure the solver options are set to “Fixed-step” and “discrete (no continuous states)”. The “Fixed-step size” should be 0.05. This is how fast in seconds the

control loop will execute. In this case, every 50 milliseconds it will execute the Simulink code.



6) Compile and download the code to the board:

- Click “Deploy to Hardware Button” or the keystroke Ctrl+B:



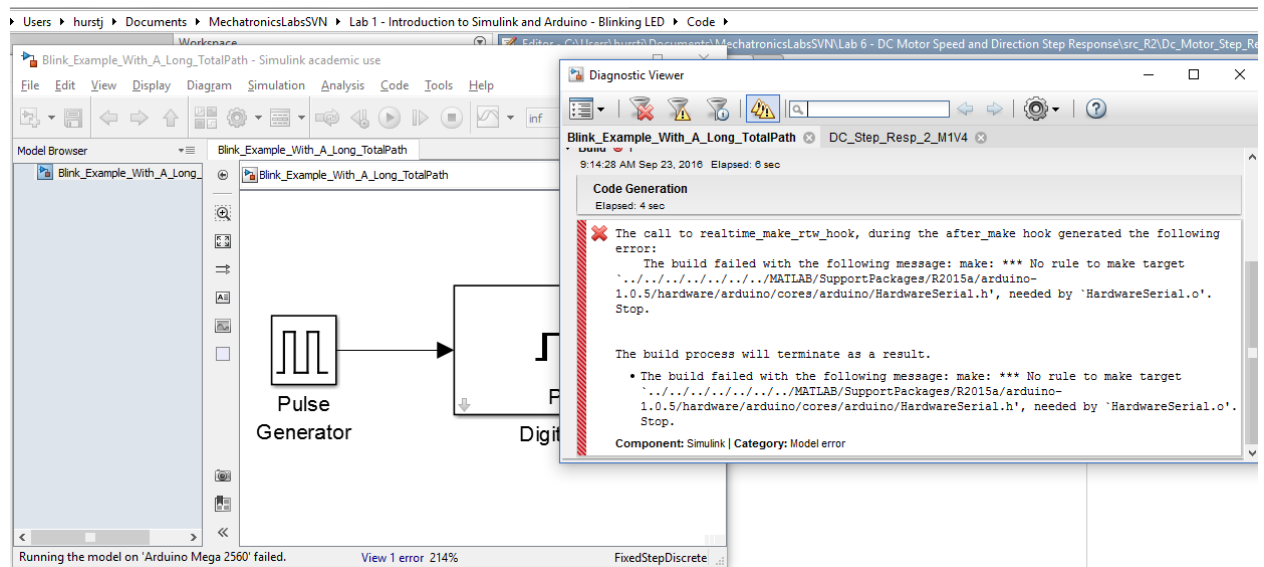
The “Deploy to Hardware button” will download the code to the board and **will not interact or connect with it**. In “Normal” Mode the code is downloaded to the board, and no information is passed between it and your computer unless your program specifically instructs the board to do so.

At this point the onboard LED connected to pin 13 should start blinking on and off at a very low frequency. This LED is on the lower board towards the USB connector. If not, check the troubleshooting section below.

## Troubleshooting

- If an error occurs indicating that there is no driver, follow the directions on the screen.
- If MATLAB cannot find the board find COM port number from the device manager and in ‘Configuration Parameters’ set the port manually to the correct number.
  - Also, try unplugging the Arduino USB cable and plugging it back in.
  - Can also try restarting MATLAB
- Some errors appear on the MATLAB workspace screen - if something is not working, check here – the error messages will indicate the problem

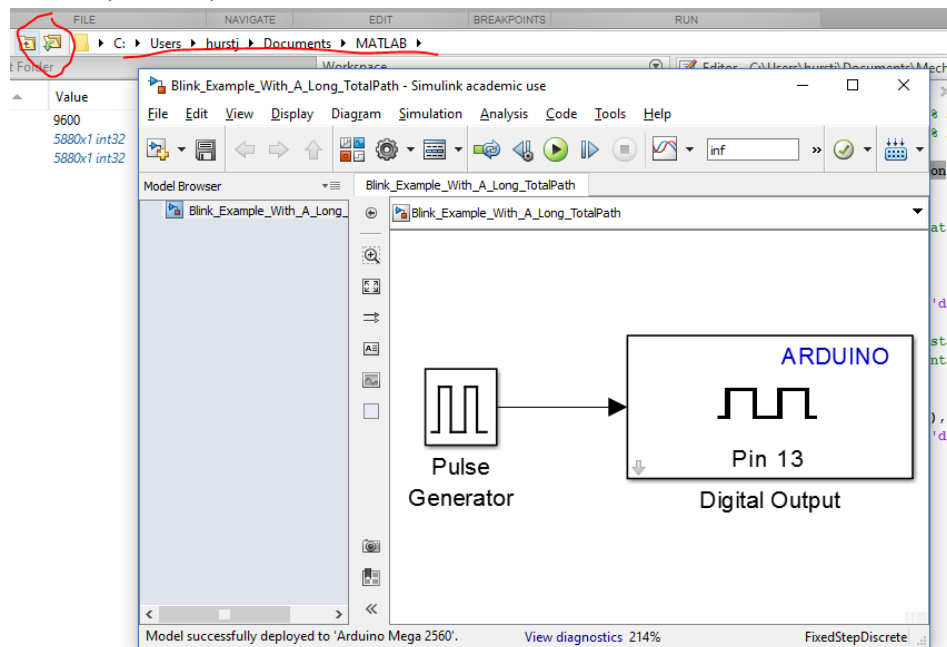
- If you get a code generation error like below:



This is because MATLAB is compiling in a directory where the directory and filename is too long. In the example above the compilation path is:

C:\Users\hurstj\Documents\MechatronicsLabsSVN\Lab 1 - Introduction to Simulink and Arduino - Blinking LED\Code\Blink\_Example\_With\_A\_Long\_TotalPath.slx

To remedy this you can try a shorter file name, or simply change (point) MATLAB to a shorter directory to compile in like:



It is the same file, but MATLAB will now compile everything in C:\Users\hurstj\Documents\MATLAB so the compiler won't have a path problem.

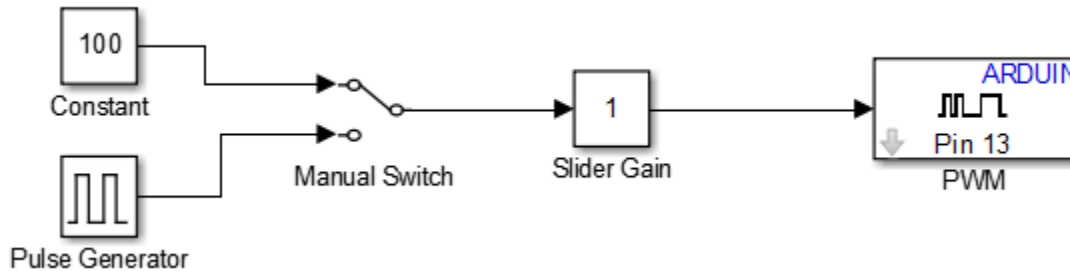
## Part 2: Communicating with External Mode

### Objective

- Communicate with the target board (Arduino) using external mode by changing the brightness of an LED with PWM

### Simulink

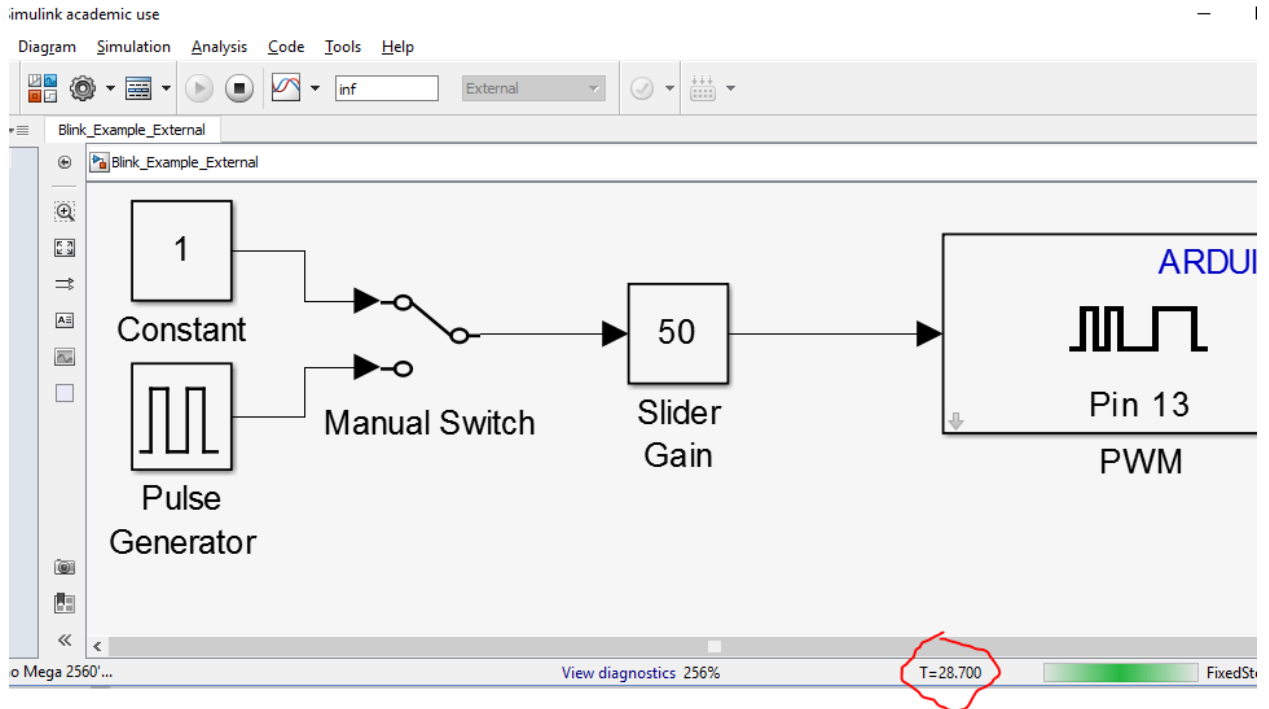
Modify your Simulink diagram to the following



- Pin 13 can also be set as a PWM instead of just an on/off digital output – replace the Digital Output block with the PWM block
- The “Slider Gain” can be found in View->Library Browser->Simulink-> Math Operations. Change the high value to 255 (the max value of the PWM)
- The “Manual Switch” can be found in View->Library Browser->Simulink ->Signal Routing
- The “Constant” can be found in View->Library Browser->Simulink -> “Commonly Used Blocks”

### External Mode

- External mode** and the green play button **allows bi-directional communication** to/from the application board to the PC, but can limit how fast your code can execute.
  - The “**Deploy to Hardware Button**” downloads the code and **does not connect or interact with it**, it can allow your code to run faster
- When external mode is selected and the green play button is pressed it downloads additional code to the board to allow this communication. This does increase the program size (requires more memory)
- When you use external mode, you can change parameters while the system is running
  - However, this has an impact on the performance due to the communication bandwidth, the fastest you can run in this mode and still meet your control loop time is approximately 30 milliseconds. Newer USB 3.0 ports and computers may run faster.
  - You can tell how fast you can run your code by examining the running clock in the Simulink diagram after you press the play button. When the code is running the green bar will be flashing and the time in seconds should be advancing:

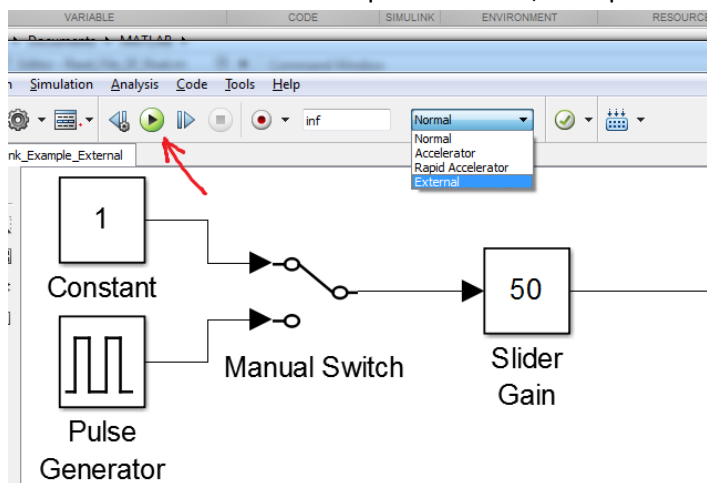


If this clock advances as fast as the normal real world clock your code is “keeping up”. If, for example, it can take 1 minute for the this timer to reach 10 seconds your code is not keeping up with real time and your requested sample time is too fast – it can not compute everything it needs to do fast enough.

- The values change on the Slider Gain and the Manual Switch can be changed while the program is running.
- The code can be run much faster (in the orders of 1 or 2 milliseconds) if external mode is not used, by using “Deploy to Hardware” instead of the green run button.

### Enabling External Mode:

- Select “External” from the drop down menu, then press the “Play” button.



Run the code and experiment with the setup. Observe the effects on the LED when changing the value of the Slider Gain, Manual Switch, and the Pulse Generator.

- While running the simulation, you can change the position of the switch by double clicking on the switch
- You can change the gain of the Slider Gain while running the simulation by double clicking it.
- Adjusting the Pulse Generator, experiment with all three parameters. Which increases brightness? What is the approximate range over which you can observe a difference?  
[Note: These parameters will be addressed in a later lab on PWM]

### Stopping the Simulation:

When you are running the simulation in external mode you are connected to the device and are sending data and receiving data.

- Use the green “Play” button to download and run the code in external mode
- Use the square “Stop” button to stop. Since the board is no longer connected it will continue doing the last action before it was disconnected.

### Debugging – cannot connect to device or cannot download code

- If you cannot connect to the device or there is an error when trying to download the code usually this means external mode was enabled and the simulation was stopped with the stop button or the USB cable was unplugged while being connected to the device. In this case you have a couple things to try that may successfully close the serial port
  - Disconnect and reconnect the USB Cable
  - Try the following commands at the Matlab command line (find connected instruments and close them):
    - `newobjs=instrfindall`
    - `fclose(newobjs)`
    - `delete(newobjs)`
  - Reset the device with the reset button or disconnect/connect the serial cable (this sometimes works)
  - Close and restart Matlab (this usually works)
  - Log off your machine, log back in, then restart Matlab (this almost always works)
- Make sure the COM port is correct in the “Model Configuration Parameters”

### Checkpoint:

To complete Lab 1 you will need to show the ability to control the blink of the LED in one program with both the ‘manual switch’ and the ‘slider gain’.



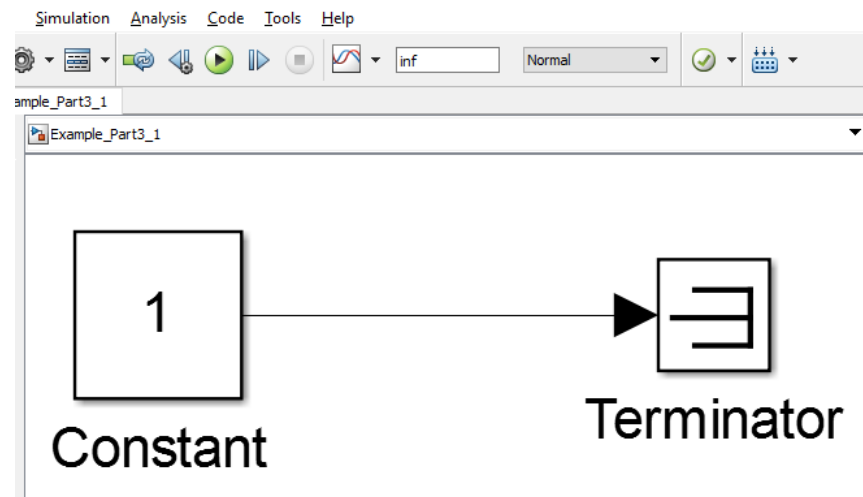
## Part 3: Evaluating code execution speed – how fast can I run my code

### Objective:

- Determine how fast your code can be run on the target hardware using “Enable overrun detection” in Simulink.

### Simulink:

Build the following model:



Note that this code essentially does nothing. The terminator is also in the commonly used blocks'. To determine how fast the hardware can execute a simple loop check the box “Enable overrun detection” and set the pin to 13.

Under the tools menu go to ‘Run on Target Hardware’ and click ‘Prepare to Run’. In the window that pops up you will need to select Arduino Mega 2560 in the drop down for Target Hardware. “Enable overrun detection” and set the pin to 13.

Configuration Parameters: Example\_Part3\_1/Run on Hardware Configuration (Active)

Select: Solver Data Import/Export > Optimization > Diagnostics Hardware Implementation Model Referencing > Simulation Target Run on Target Hardware	<b>Target hardware selection</b> Target hardware: <span>Arduino Mega 2560</span>
	<b>Host-board connection</b> Set host COM port: <span>Automatically</span>
	<b>Overflow detection</b> <input checked="" type="checkbox"/> Enable overflow detection Digital output to set on overflow: <span>13</span>
	<b>Signal monitoring and parameter tuning</b> External mode transport layer: <span>serial</span>
	<b>Arduino analog input channel properties</b> Analog input reference voltage: <span>Default</span>

If the hardware cannot execute your code in the same time you specify the digital output, in this case the LED on pin 13 will light up indicating that your code cannot be executed in the time you have specified. You can use any digital output, but pin 13 is conveniently connected to an LED on our board. For the example code above, start with a sample time of 0.01 milliseconds (.00001 seconds). Download the code with the deploy to hardware button (Normal mode, not external mode play button) and observe if LED 13 on the board lights up after downloading:

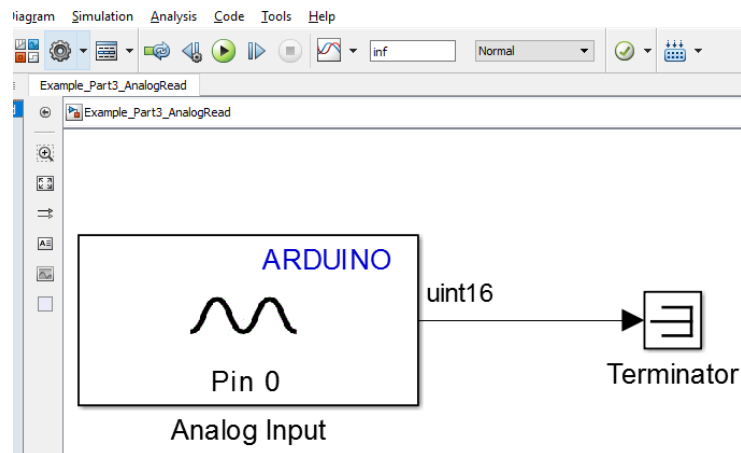
Configuration Parameters: Example\_Part3\_1/Run on Hardware Configuration (Active)

Select: Solver Data Import/Export > Optimization > Diagnostics Hardware Implementation Model Referencing > Simulation Target Run on Target Hardware	<b>Simulation time</b> Start time: <span>0.0</span> Stop time: <span>inf</span>
	<b>Solver options</b> Type: <span>Fixed-step</span> Solver: <span>discrete (no c</span> Fixed-step size (fundamental sample time): <span>.00001</span>
	<b>Tasking and sample time options</b> Periodic sample time constraint: <span>Unconstrained</span> Tasking mode for periodic sample times: <span>SingleTasking</span>
	<input type="checkbox"/> Automatically handle rate transition for data transfer
	(Empty section)

You should notice LED 13 will light up. This indicates the microprocessor can not run the code every .00001 second, so you have to specify a larger sample time

Keep increasing the sample time (try .02 milliseconds, etc.) until LED 13 does not light up after downloading. This is the fastest you the board can execute your code.

Using this procedure try and see how fast you can run the following, start at a sample time of 00.1 milliseconds.



In this fashion, you can determine how fast your code, or any part of your code takes to run on your hardware.

### Questions:

How can you determine how fast you can run code in External mode?

How can you determine how fast your code can run in Normal mode?

What is the difference between Normal mode and External mode?