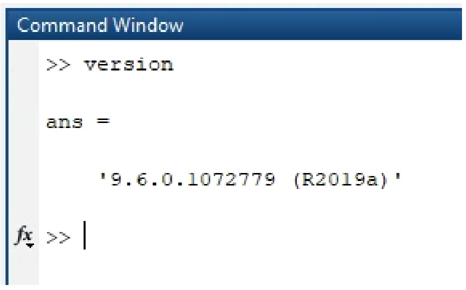
D5 Algorithm Simulation: Virtual Test Bed

1 Prerequisites

You should have MATLAB 2017b and above installed. If you haven't you may install the latest student version at software.soton.ac.uk. You should also have gcc installed in your laptop for C/C++ compilation.

1.1 How to check your MATLAB version

The version number should appear at the top right corner of MATLAB when you start up the software. If it is not shown, you can type 'version' in the MATLAB command window to get the version.



```
>> version

ans =

    '9.6.0.1072779 (R2019a)'

>> |
```

1.2 Check downloaded files

The most important files that should be available are as shown:

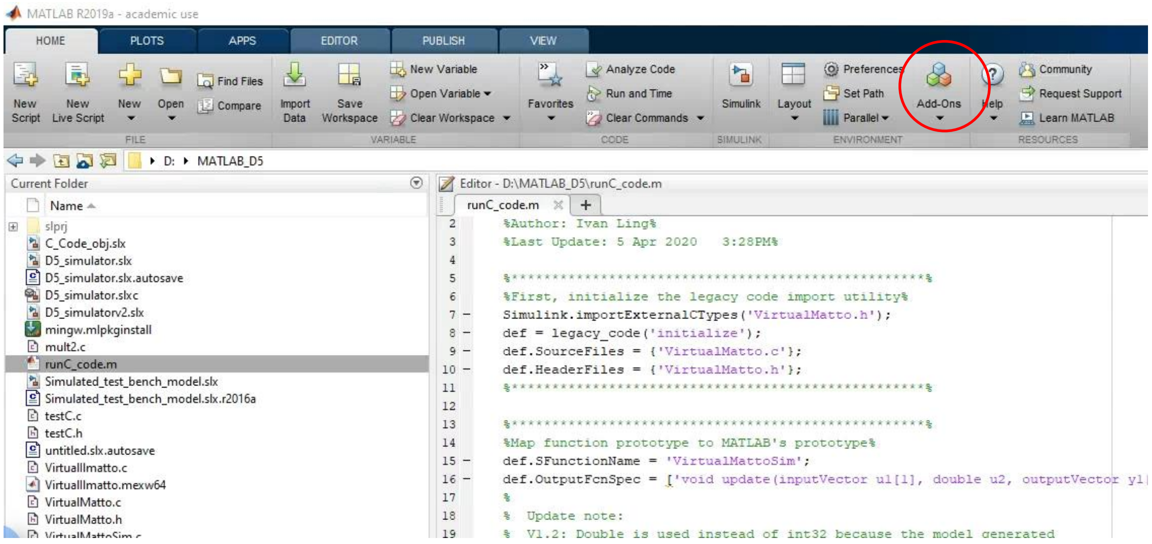
- Slprj folder - This is the Simulink project folder
- D5_simulator.slx – This is the 2019a version of the Simulink model. Use this if you have the latest MATLAB installed
- D5_simulator_compabilitymode – Use this version if you have an older version of MATLAB. Note that you should not use anything older than 2017a or else the code will not work.
- Mingw.mlpkginstall – This is the MinGW C compiler support package for **2019a**. If you are using older version, please refer to the next section for installation guide.
- D5tbico.ldi & d5vmico.ldi – Resource files needed for the D5_simulator.slx to work. Make sure they are in the same folder as your D5_simulator.slx.
- runC_code.m – A MATLAB script written to convert the VirtualMatto.c and VirtualMatto.h into a SIMULINK block
- VirtualMatto.c & VirtualMatto.h – The C code and header files where you write your algorithm. Please define all user-defined functions within the ".c" file only, and do not edit the ".h" file.

After the first run, you may notice a few additional files created. These are the module files that contain the precompiled C code blocks.

Author: Ivan Ling (tyl1m20)
Revision: 1st version

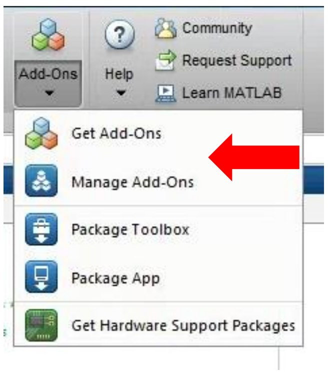
2 MinGW Support Package Installation

To install the MinGW Support Package in MATLAB, go to your MATLAB Add-On Manager

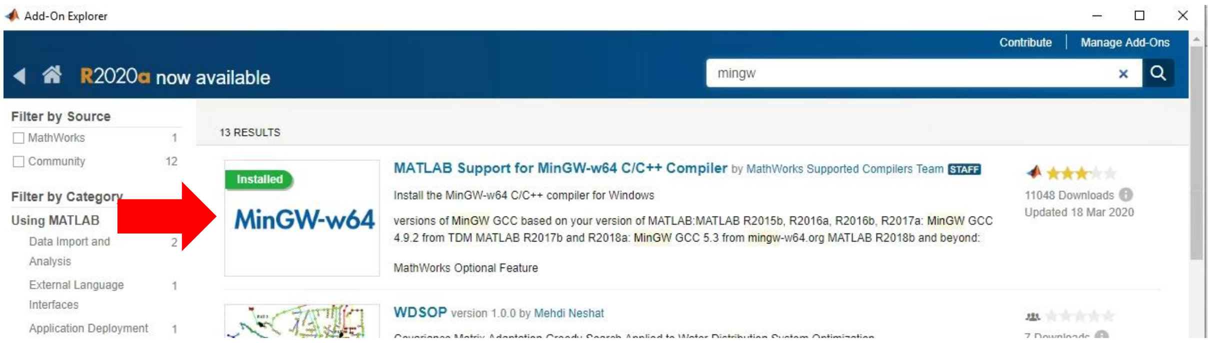

MATLAB R2019a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Add-Ons
Editor - D:\MATLAB_D5\runC_code.m

```
%Author: Ivan Ling%
%Last Update: 5 Apr 2020   3:28PM%

%****************************************************%
%First, initialize the legacy code import utility%
Simulink.importExternalCTypes('VirtualMatto.h');
def = legacy_code('initialize');
def.SourceFiles = {'VirtualMatto.c'};
def.HeaderFiles = {'VirtualMatto.h'};
%****************************************************%

%****************************************************%
%Map function prototype to MATLAB's prototype%
def.SFunctionName = 'VirtualMattoSim';
def.OutputFcnSpec = ['void update(inputVector u1[1], double u2, outputVector y1
%
%  Update note:
%  V1.2: Double is used instead of int32 because the model generated
```

Select “Get Add Ons”



When the Add-On Manager pops out, search for “Mingw” and you should be able to see a packaged labelled MATLAB Support for MinGW=w64 C/C++ Compiler.



MATLAB Support for MinGW-w64 C/C++ Compiler by MathWorks Supported Compilers Team STAFF

Install the MinGW-w64 C/C++ compiler for Windows

versions of MinGW GCC based on your version of MATLAB:MATLAB R2015b, R2016a, R2016b, R2017a: MinGW GCC 4.9.2 from TDM MATLAB R2017b and R2018a: MinGW GCC 5.3 from mingw-w64.org MATLAB R2018b and beyond:

MathWorks Optional Feature

11048 Downloads
Updated 18 Mar 2020

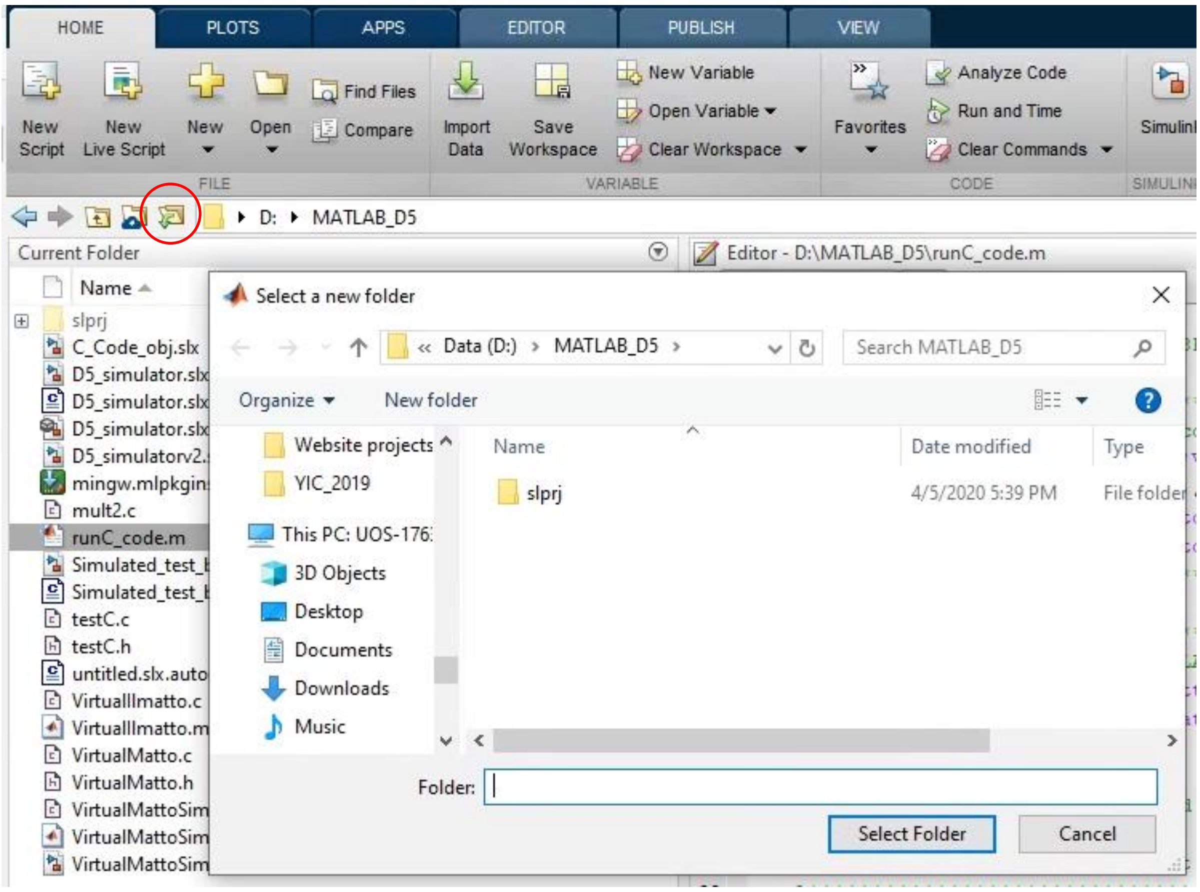
WDSOP version 1.0.0 by Mehdi Neshat

In your case, if the green “Installed” indicator is present, you don’t need to proceed further. If it is not present, click on the package, and you should reach an installation page. Download the required file and install the MinGW package. If you are unable to download, and happen to be using a x64 version, you may install by clicking on the “mingw.mlpkginstall” file found in the D5 Simulator ZIP folder.

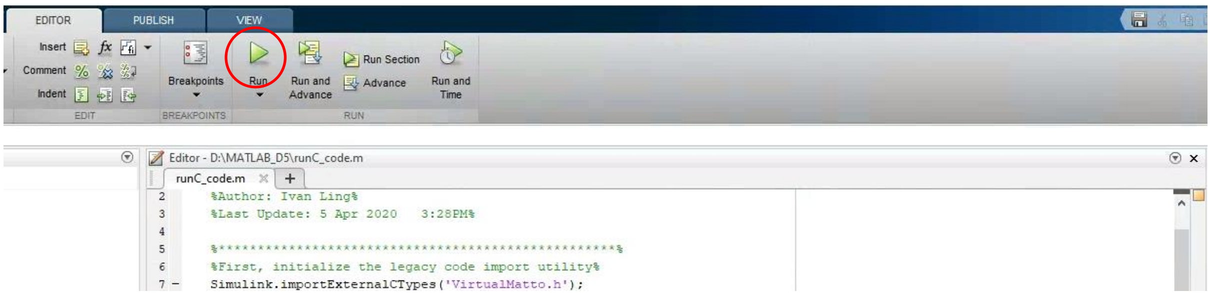
Once the file has been installed, you can proceed to compiling C codes within MATLAB.

3 Generating SIMULINK Block from C programs

In the MATLAB interface, navigate to the D5 Simulator directory by clicking on the select folder button. You should have the D5 Simulator ZIP file unzipped in a convenient location in your PC by now.



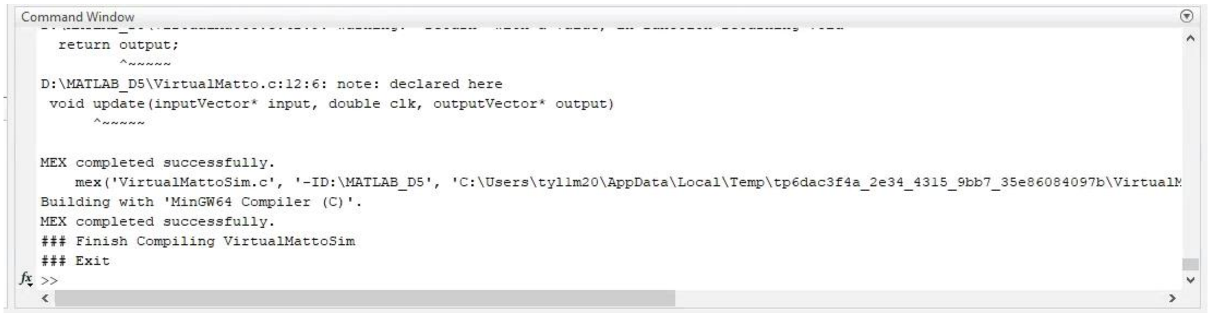
Within the D5 Simulator folder, you will be able to find a file named "runC_code.m". Open the that file with MATLAB.



```
%Author: Ivan Ling%
%Last Update: 5 Apr 2020    3:28PM%

%**************************************************%
%First, initialize the legacy code import utility%
Simulink.importExternalCTypes('VirtualMatto.h');
```

In the "Editor" tab, select "Run". This will attempt to compile the prewritten VirtualMatto.c and VirtualMatto.h file. These two files contains the bare-bone template of the simulator's C-code.

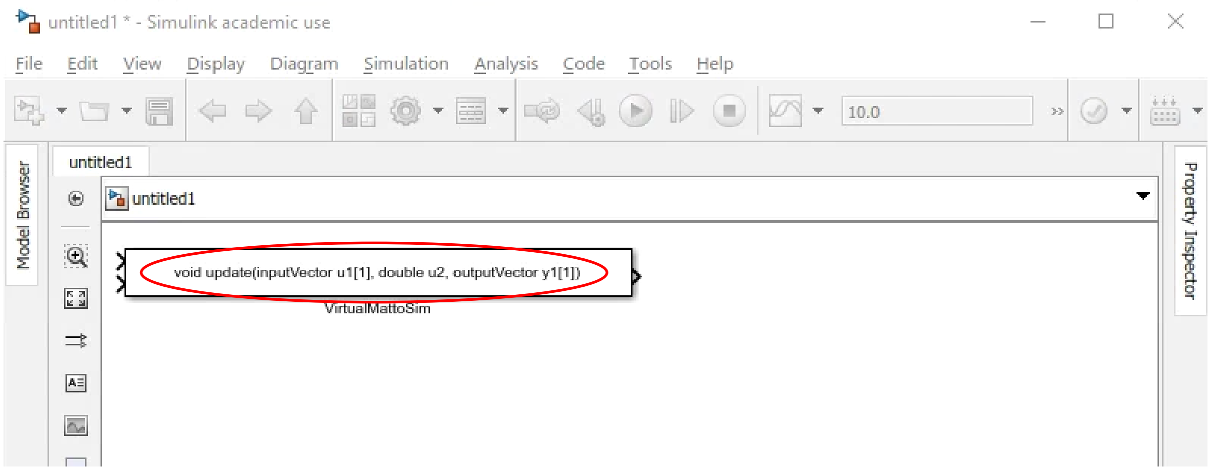
You should be able to see the following if the process completed successfully.



```
  return output;
         ^~~~~~
D:\MATLAB_D5\VirtualMatto.c:12:6: note: declared here
 void update(inputVector* input, double clk, outputVector* output)
      ^~~~~~

MEX completed successfully.
    mex('VirtualMattoSim.c', '-ID:\MATLAB_D5', 'C:\Users\tyl1m20\AppData\Local\Temp\tp6dac3f4a_2e34_4315_9bb7_35e86084097b\VirtualM
Building with 'MinGW64 Compiler (C)'.
MEX completed successfully.
### Finish Compiling VirtualMattoSim
### Exit
>>
```

A common problem at this stage is permission error, where MATLAB is unable to create the required file at the specified directory. If that happens, make sure you run MATLAB at an elevated permission (Run as Admin), or move your working directory to a folder where you have full access rights.

After the compilation has completed successfully, a SIMULINK block will pop up. If it doesn't, please check and ensure you have SIMULINK installed properly.



void update(inputVector u1[1], double u2, outputVector y1[1])

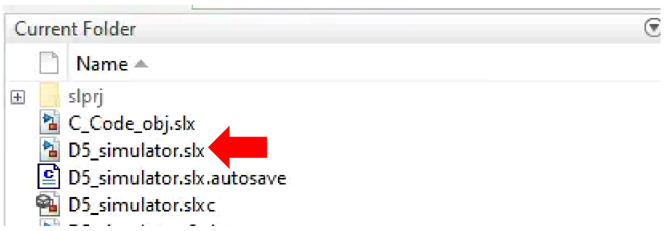
VirtualMattoSim

When this pops up, right-click on the block and select "copy". You will need to insert this into the SIMULINK model later.

Author: Ivan Ling (tyl1m20)
Revision: 1st version

4 Inserting the compiled C-code block into SIMULINK

On MATLAB’s main interface, select the “D5_simulator.slx” file and open it.

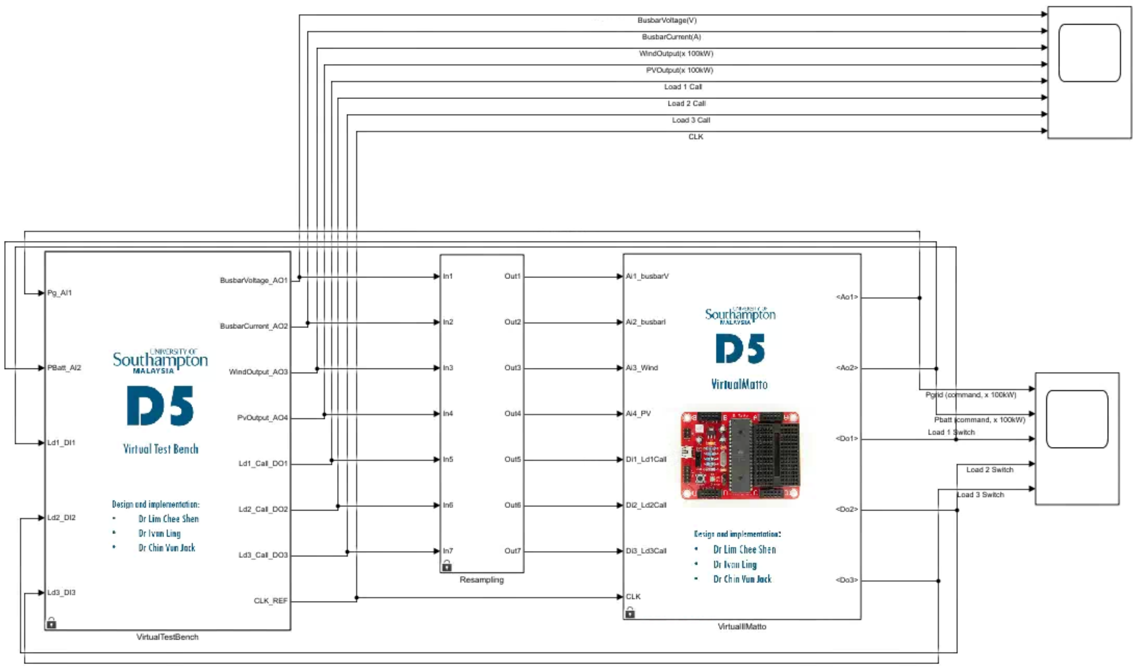


Current Folder

Name

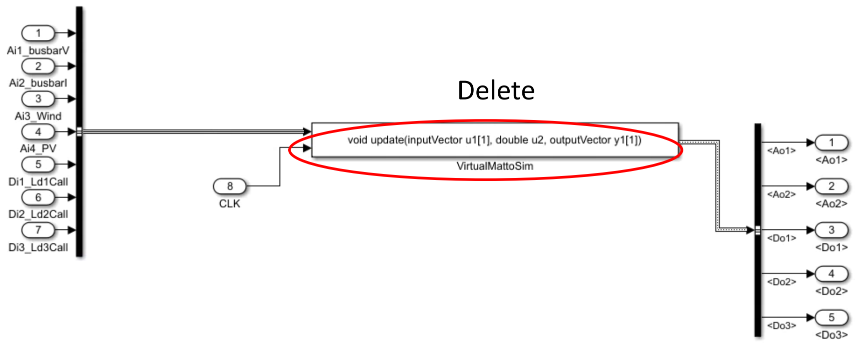
- slprj
- C_Code_obj.slx
- D5_simulator.slx
- D5_simulator.slx.autosave
- D5_simulator.slxc

You should be able to see the model in the following configuration:

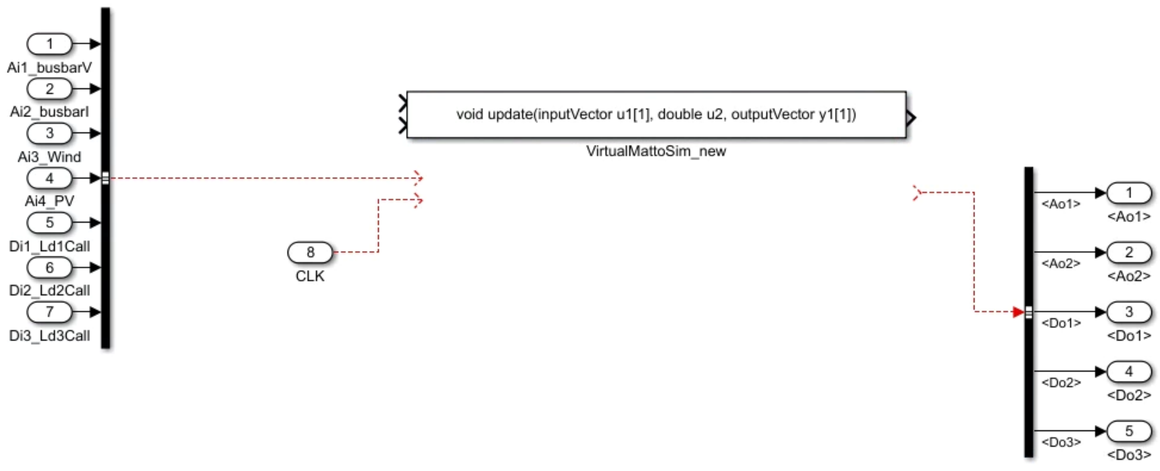


The model is split into two main parts, with a “Resampling” block between them to synchronize the input. The “TestBench” block contains a pre-coded load profile which is similar (but not identical) to the LabView Testbench described in your lab sheet. The “VirtualIlMatto” block contains the C-code to generate the necessary responses. Instead of displaying to an OLED screen, all the signals are tapped out into a virtual oscilloscope for viewing.

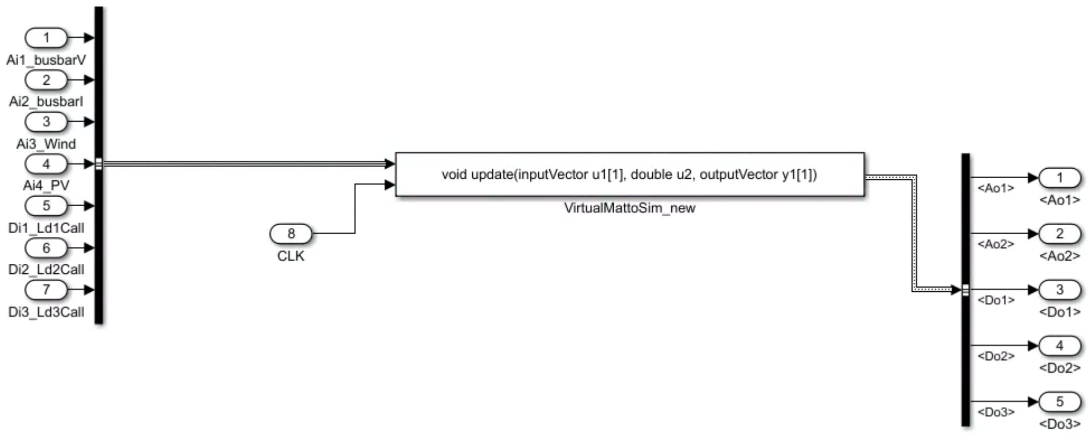
To update the code inside the “VirtualIlMatto” block, double click on the block and delete the previous C-code block.



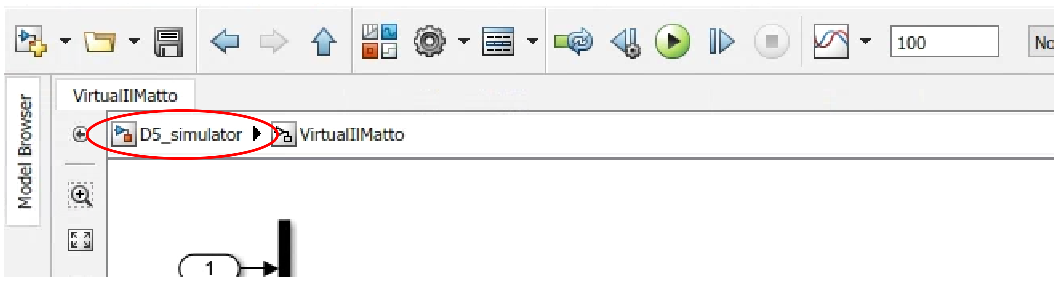
Then, replace the deleted block with the new block by connecting all the virtual wires.



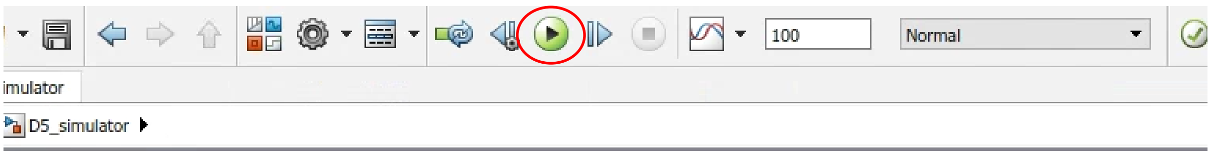
Make sure the CLK signal is connected to the input port at the bottom on the left side, and the input bus is connected to the top input port. There is only one output port, so just connect that to the output bus. You should have something like this when you are done:



Double click on the "D5_simulator" block at the top left of the window to go back to the simulator view. *Note: I renamed the block with "_new". In your case, the new and old block are the same.*



To verify that everything is working fine up to this point, press the “Start Simulation” button located at the toolbar.

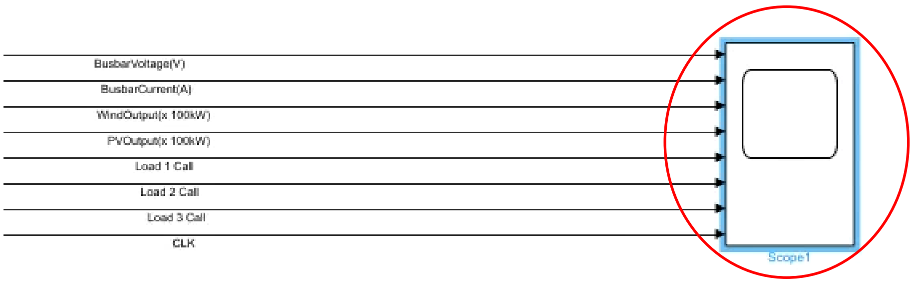


At this stage, if there are no error messages, you are good to go.

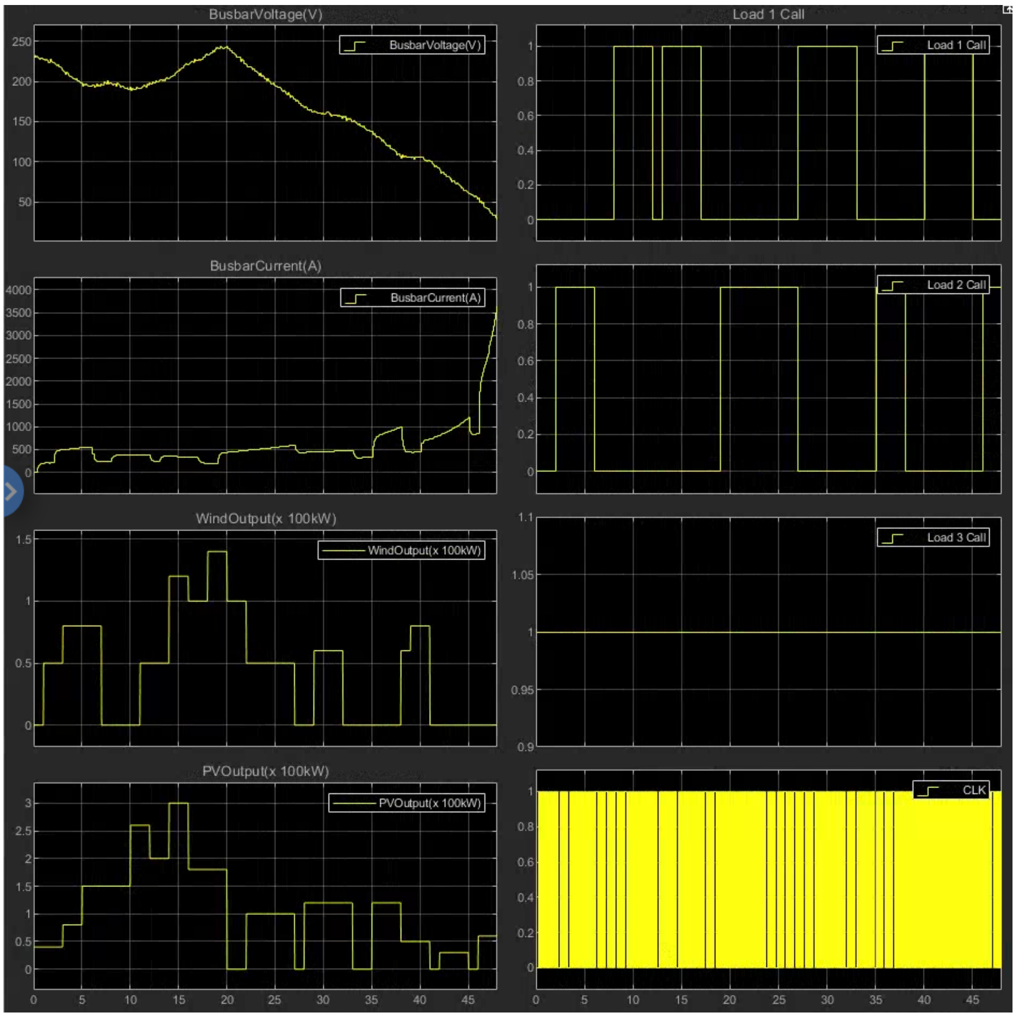
Author: Ivan Ling (tyl1m20)
Revision: 1st version

5 Viewing simulation results

The result of the simulation is viewable by clicking on the scopes connected to the components.



Sample output signal from test bench:



Author: Ivan Ling (tyl1m20)
Revision: 1st version

6 Editing the C code

To edit the C code, open up “VirtualMatto.c” with your preferred code editor. You should be able to see the skeleton code written to get you started.

```
#include "VirtualMatto.h"

int triggerFlag = 0;


//***************************************************************
// STUDENT VARIABLES GO HERE
//***************************************************************

//----------------------
// example

double temp; // Creating a variable.
//NOTE: For simulation to work, use only "double" as the data type.
// for boolean values, use 1.0 to represent "True" and 0.0 to represent "False"


// end example
// ----------------------

//***************************************************************
void update(inputVector* input, double clk, outputVector* output)
{

	if (clk == 1 && triggerFlag == 0) //Only trigger at edge
	{
		triggerFlag = 1;
		//***************************************************************
		// STUDENT CODES GO HERE

		//----------------------
		// example
	 temp = 0;

		//Setting Output
		output->Do1 = 1 ;
		output->Do2 = 1 ;
		output->Do3 = 1 ;
		output->Ao1 = 0 ;
		output->Ao2 = 0 ;

		//Getting Input
		temp = input->Ai1;
		temp = input->Ai2;
		temp = input->Di1;
		temp = input->Di2;
		// end example
		// ----------------------

		// STUDENT CODE END
		//***************************************************************
	}
	else if (clk == 0)
	{
		triggerFlag = 0;
	}
	return output;

}
```

In order to ensure that the testbench runs at a higher speed than the virtual Il Matto, a CLK signal is generated by the “TestBench” block. The Il Matto should only run when the CLK signal is triggered. When triggered, the virtual Il Matto will run the codes written between the “STUDENT CODES GO HERE” comment and the “STUDENT CODE END” comment. You may insert your code here. Take note that if you declare variables here, it’ll be overwritten in the next loop cycle. If you want the variable to persist, you should declare it as a global variable in the location indicated (“STUDENT VARIABLES GO HERE”).

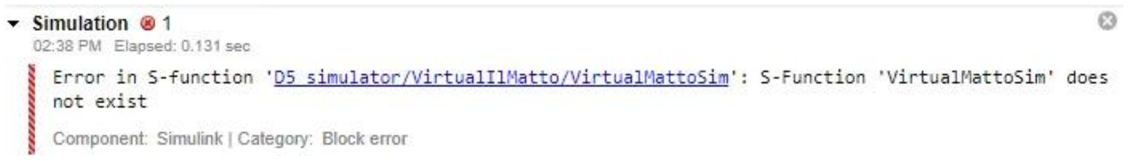
Instead of using Digital IO, ADC and PWM signals, the simulated Il Matto communicates directly using the ports Ai1-A14 for analogue input, Di1-Di3 for digital input, Ao1-Ao2 for analogue output and Do1-Do3 for digital output. The summary of the pin mapping is as shown below:

Variable Name in C	Connected test bench port	Description
Output->Ao1	Pg_AI1	Power generation control
Output->Ao1	PBatt_AI2	Battery control
Output->Do1	Ld1_D1	Load 1 switch
Output->Do2	Ld1_D2	Load 2 switch
Output->Do3	Ld1_D3	Load 3 switch
Input->Ai1	BusbarVoltage_AO1	Busbar Voltage
Input->Ai2	BusbarCurrent_AO2	Busbar Current
Input->Ai3	WindOutput_AO3	Wind output
Input->Ai4	PvOutput_AO4	Solar PV output
Input->Di1	Ld1_Call_DO1	Load 1 call
Input->Di2	Ld2_Call_DO2	Load 2 call
Input->Di3	Ld3_Call_DO3	Load 3 call
CLK	CLK_REF	Reference clock signal to drive the Il-Matto

The test code has no real meaning. Feel free to delete everything between the comments and replace them with your own algorithm. The only difference is now you MUST declare all interface variable as “double” regardless of whether it is a digital or analogue value. This is so that the blocks can communicate with other blocks within the SIMULINK testbench model. For this purpose, you should assume your analogue IO to be able to receive the full voltage value, without needing to do an ADC conversion. In the same way, your output should be able to provide full voltage value as well.

7 Common errors

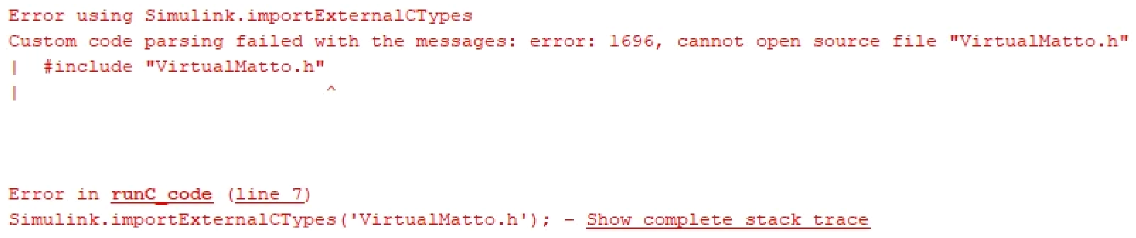
7.1 Simulation error



```
Simulation ⊗ 1
02:38 PM  Elapsed: 0.131 sec
Error in S-function 'D5 simulator/VirtualIlMatto/VirtualMattoSim': S-Function 'VirtualMattoSim' does
not exist
Component: Simulink | Category: Block error
```

If this error occurs during simulation, it means you have not generated the correct C-code block for SIMULINK. Try generating the blocks again (Refer to Section 3).

7.2 Compilation error



```
Error using Simulink.importExternalCTypes
Custom code parsing failed with the messages: error: 1696, cannot open source file "VirtualMatto.h"
|  #include "VirtualMatto.h"
|                          ^


Error in runC_code (line 7)
Simulink.importExternalCTypes('VirtualMatto.h'); - Show complete stack trace
```

If the error shown above occurs during compilation, please ensure that you have sufficient rights to access the current MATLAB directory, and also ensure that the VirtualMatto.c and VirtualMatto.h files are located in the correct MATLAB directory.

8 Notes

We are aware that at with the current situation, you might not be able to implement your full algorithm due to the limitations of the simulator. Nonetheless, try your best to debug and to solve the problems. You may need to tweak your algorithms slightly to make it work in the simulator. If you've done so, do record it and include the changelog in the report.

If you need further technical support on the simulator installation, feel free to contact Dr Ivan Ling at ivan.ling@soton.ac.uk. For questions regarding the test-bench model, please direct them to Dr CS Lim at c.s.lim@soton.ac.uk.

Author: Ivan Ling (tyl1m20)
Revision: 1st version