Getting started for Realtime Simulation with Simulink

In this document you will find all the information you need for designing a Realtime Simulation running on a Raspberry Pi developed with MATLAB Simulink. First read the chapters and start developing when you reach chapter “5. Step by Step Guide to create your Realtime Simulation”.

Content

Getting started for Real Time Simulation with Simulink ................................................................. 1

1. How to transfer data between the Raspberry Pi and your computer ........................................... 1

2. How to bring your model to the Raspberry Pi .......................................................................... 3

3. How to design a GUI for displaying data .................................................................................. 5

4. Start and Stop the Raspberry Pi model with your GUI ............................................................. 8

5. Possible errors ......................................................................................................................... 9

6. Step by Step Guide to create your Realtime Simulation ........................................................... 10

1. How to transfer data between the Raspberry Pi and your computer
To establish a datalink between the Raspberry Pi and the PC we need a communication via ethernet. There are two possible protocols. The first is the TCP/IP protocol. The other is the UDP protocol. Both protocols can be found in the so-called network layer, which is the fourth layer of the OSI model. Because UDP is a little easier to use and absolutely sufficient we will design our software with the UDP function.

In Simulink you will find the UDP-Block shown in Figure 0-1 by searching the library for UDP.

![UDP Blocks](image)
Because your simulation is running on the Raspberry Pi you have to use the UDP Send in its model to transfer the data to the UDP Receive function which you have to use in the GUI running on your PC.

Now you have to configure your UDP-Blocks as shown in Figure 0-2 and Figure 0-3.

**Figure 0-2 UDP Send config**

**Figure 0-3 UDP Receive config**
The UDP Receive Block could be difficult to understand so here is one small example. If you want to send three signals of the datatype double to your GUI, which you want to read every 100ms, your configuration of the UDP Receive should look like in Figure 0-4.

![Figure 0-4 UDP Receive example](image)

At this point there is maybe one question left: How is it possible to send more than one signal over UDP when there is only one input port? Therefore, you can use the blocks Mux and Demux. For the previous example your models could look like in Figure 0-5.

![Figure 0-5 Mux and Demux](image)

2. **How to bring your model to the Raspberry Pi**

After setting up the communication in your Raspberry Pi model it’s time to configure the settings for the hardware simulation. First you have to open the “Model Configuration Parameters”.

![Figure 0-6 Model Configuration Parameters](image)
Go to the section “Hardware Implementation” and select the Raspberry Pi board like in Figure 0-7.

![Figure 0-7 Hardware Implementation](image)

In the next step go to the “Solver” section and configure it like in Figure 0-8.

![Figure 0-8 Solver](image)

The last step would be to set the simulation time to infinity and then deploy the model to the hardware like in Figure 0-9.

![Figure 0-9 Deploy to Hardware](image)

When nothing went wrong a report file should pop up. If you receive an error at this point check your properties again and retry, otherwise ask your person in charge for help.
3. How to design a GUI for displaying data
In this section you will find a small “How-to” for creating a Graphical User Interface (GUI). First you have to create a new Simulink model. Create an area and go to the properties of the area and make it to a subsystem like in

![Click here to create an area](image)

*Figure 0-10 Area*
In Figure 0-11 you will see the GUI elements that you should use. In the figure you can see three different types of elements:

**Red:** Those elements are control elements. You won’t need them because you only have to display data.

**Orange:** This element could be used but is buggy. A work around could be to use a normal display that you know from the exercises.

**Green:** These elements could be used to display data. You can use them for your lab exercises.

*Figure 0-11 Dashboard elements*
If you then place one of the elements onto your area you have to double click your element. A popup should open. There you can connect your GUI element to a signal, also it’s always possible to define a minimum and maximum value for your element.

To connect your element to a signal open your Subsystem. There you have to click on the signal that you want to display like in Figure 0-12.

**Figure 0-12 Configure GUI element**

Next open the “Model Configuration Parameters” and go to the “Solver” section and do the configuration like in Figure 0-13.

**Figure 0-13 Model Configuration Parameters**
The last thing you need is the Set Pace block to slow down the simulation time. Search in the library browser for “Simulation Pace” and add the block to your model. Now configure it like in Figure 0-14.

![Simulation Pace](image)

**Figure 0-14 Simulation Pace**

4. Start and Stop the Raspberry Pi model with your GUI

The last thing you need to do is to create to different .m files to start and stop the model that is on your Raspberry Pi.

**Start model:**

```matlab
clear
h = raspberrypi; %clears the workspace
h.runModel('YOUR RASPI MODEL NAME'); %establish connection to Raspberry
Raspberry %starts the model on your Raspberry
clear h; %clears the h object
```

**Stop model:**

```matlab
h = raspberrypi; %establish connection to Raspberry
h.stopModel('YOUR RASPI MODEL NAME'); %stops the model on your Raspberry
clear %clears the workspace
```
Next you have to create Callback-Functions so that your GUI executes the two .m files by starting or stopping your model.

In the Simulink Editor, open the Property Inspector. Select View > Property Inspector. Go to the Properties tab, in the “Callbacks” section, select the callback you want to set.

![Property Inspector with Callbacks selected](image)

**5. Possible errors**

In this section you will find the most common errors.

- Can’t establish connection to the Raspberry Pi when trying to deploy model to the Raspberry.  
  **Solution:** Ask your supervisor for help.

- My GUI elements don’t do anything.  
  **Solution:** Check whether the “Model Configuration Parameters” of your Raspberry Pi model are configured correctly. Then do the same for your GUI. Next check whether the “Sample Time” of your UDP Receive block is set to the value 0.1. Then check the IP addresses of both UDP blocks. Then check whether the Buffer Size of the UDP Receive block is set correctly, do the same for the message length.

- My GUI elements are not running fluent or are jumping.  
  **Solution:** Check whether the “Model Configuration Parameters” of your Raspberry Pi model are configured correctly. Then do the same for your GUI. Next check whether the “Sample
Time” of your UDP Receive block is set to the value 0.1. Then check whether the Buffer Size of the UDP Receive block is set correctly, do the same for the message length. Make also sure that all the GUI elements you use are connected to a signal. If you still receive this behaviour set the “Sample Time” of your UDP Receive block to the value 0.2.

6. Step by Step Guide to create your Realtime Simulation

In this section you will find all the necessary steps to create your own Realtime Simulation for the Raspberry Pi. You should use your solution from exercise 7 and display the following:

Sim_Time[s], Lap_Time[s], Fastest_Lap, Fastest_Time[s], Lap_count, current_lap_distance[m], v[km/h], gear;

One possibility to display those signals could look like in Figure 0-16.

![Figure 0-16 Possible GUI](image)

Please design a linear gauge where it’s possible to see whether the car is in a curve or on a straight.

- First, it’s absolutely necessary that your simulation that you want to bring to the Raspberry Pi is working correctly. So, do the Lab exercise and then if you get the “OK” from your supervisor start with this exercise.

- This exercise could be really difficult because there are many sources of error. But it’s important that you try to solve the problem on your own by reading chapter “Possible errors”. If you get stuck please do not hesitate to ask.

- Create following files in a new folder:
  - Init.m -> For initialising all the variables in your Raspberry Pi model
  - StartRasp.m -> For starting the Raspberry Model. More information will come later.
  - StopRasp.m -> For stopping the Raspberry Model. More information will come later.
  - Ex[number of exercise]_rasp.slx -> Your model that will run on the Raspberry Pi.
Ex(number of exercise)_GUI.slx -> The GUI to display the data coming from the Raspberry Pi.

- Finish your Init.m file.
- Open Ex(number of exercise)_rasp.slx and build up an UDP communication to transmit all the signals that you need in your GUI. Refer to chapter “1. How to transfer data between the Raspberry Pi and your computer”.
- Next configure your Ex(number of exercise)_rasp.slx model and deploy it to the Raspberry Pi as described in chapter “3. How to bring your model to the Raspberry Pi”.
- Open Ex(number of exercise)_GUI.slx and develop your GUI by placing all the elements onto your area/subsystem you need. Don’t configure the elements at this point. Refer to chapter “4. How to design a GUI for displaying data”.
- Implement the Subsystem of your Ex(number of exercise)_GUI.slx. Refer to chapter “2. How to transfer data between the Raspberry Pi and your computer”.
- Configure your GUI elements like in chapter “4. How to design a GUI for displaying data”.
- Configure the in your Ex(number of exercise)_GUI.slx model “Model Configuration Parameters” like in chapter “4. How to design a GUI for displaying data”.
- Implement StartRasp.m and StopRasp.m. Then configure them as callback functions like in chapter “4. Start and Stop the Raspberry Pi model with your GUI”.
- Now your application is ready to run.