RemoteLabs

USER MANUAL

Sep-2021

Control Systems (5ESD0)

Q1 2021-2022

Contents

1 Introduction

RemoteLabs is an on-line environment where students can login and connect to a physical set-up to run experiments. Remotelab set-ups are available on a 24/7 basis.

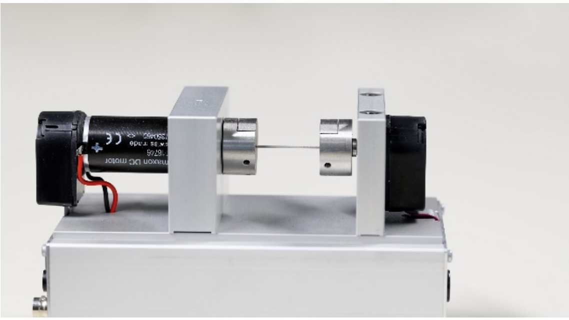
RemoteLabs has two distinct environments that are part of one framework:

1. Experiments can be specified and simulated using Matlab/Simulink on student laptops.
2. The Matlab/Simulink files developed in (1) can be submitted to the RemoteLabs server. Here a scheduler will manage the experiment requests and will deploy them on the real physical set-up. Experiment data will be collected and can be downloaded from RemoteLabs by the students for further analysis.

This User Manual will describe these two environments in the RemoteLabs Matlab-Simulink framework.

During the Control Systems course, students will run control experiments on two different set-ups:

The first one is a DC motor driving two masses coupled using a flexible axle, this setup will be referred to as FlexAxle in this document. With the second set-up a metal ball is levitated in an electromagnetic field, we will refer to this setup as MagLev. Below you see a picture of both set-ups.



Notes:

- This document is not about controller design. This document is about how to run a controller in simulation and on the actual setup in RemoteLabs using the RemoteLabs Framework.
- RemoteLabs uses Matlab/Simulink R2019b on the actual setup. Simulations using the RemoteLabs Framework can run on any version but files uploaded to the actual setup must be saved in the R2019b version.

2 The RemoteLabs MATLAB-SIMULINK Framework

2.1 Introduction

As a starting point for creating experiments for RemoteLabs, students can download from CANVAS the following Matlab/Simulink files:

For the FlexAxle set-up:

1. RL_FA_Controller_5ESD0.slx
2. RL_FA_Controller_5ESD0.mat
3. RL_FA_System_5ESD0.slx
4. RL_FA_TOP_SIM_5ESD0.slx
5. RL_FA_Busses_5ESD0.mat
6. condesign.m
7. controller.mat
8. TunableParams.m
9. Student_files.zip

And for the MagLev set-up a similar set:

1. RL_ML_controller.slx
2. RL_ML_controller.mat
3. RL_ML_System.slx
4. RL_ML_TOP_SIM.slx
5. RL_ML_busses.mat
6. controller_parameters_STUDENT.m

This section describes the function of the various files in the RemoteLabs Matlab/Simulink Framework.

The typical workflow when designing and testing controllers:

1) Model the plant
2) Design a controller
3) Verify the controller performance in simulation using a plant model
4) Test the controller on the real set-up
5) If needed go back to 1) (to get a better model) or 2) (to further tune the controller) and follow the steps again.

In the context of using RemoteLabs, we will now focus on steps 3) (simulation) and 4) (testing on real set-up).

2.2 RemoteLabs Simulation

You will do the simulation on your own laptop, you will not need RemoteLabs for this. However, you will set up your simulation environment in Simulink using the files listed above, such that your simulation files will be compatible with the RemoteLabs Framework.

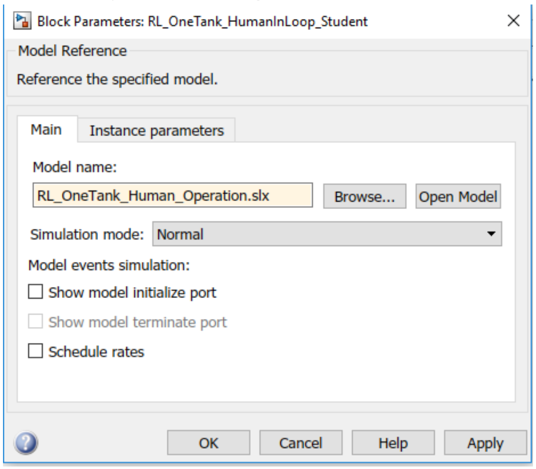
The file RL_##_TOP_SIM.slx is a Simulink model that you will use to do simulations of your controller using a dynamical model of the plant (plant being in this case the FlexAxle or MagLev system). This Simulink model contains two subsystems which are in fact ***referenced models***. A screen shot of the RL_FA_TOP_SIM.slx looks as follows:

These two subsystems in both RL_##_TOP_SIM.slx models each refer to another Simulink file. Initially, when you download these files from CANVAS, the left subsystem refers to RL_##_Controller.slx and the right one refers to RL_##_System.slx. As the filenames suggest, RL_##_Controller.slx will hold your controller implementation in Simulink while RL_##_System.slx contains your plant model. Both systems are connected through interface busses which are defined in RL_##_Busses.mat.

The files the subsystems refer to can be easily changed, which gives you the possibility to have various controllers and/or plant models stored in different files you can easily 'plug in' into the overall RL_##_TOP_SIM.slx model.

Changing the model reference is easy, proceed as follows:

- Right-click on the subsystem
- From the menu select: *Block Parameters (Model reference)*
- This will open the dialog box below.



Browse for the file you want to refer to and click OK.

Parameters used by the controller and system subsystems are either specified in the .mat files or generated by the matlab scripts (.m files) as part of the controller design exercise.

The .mat files have another function in RemoteLabs when running a controller on the real setup, as is described in the next section.

2.3 RemoteLabs Experiments

When simulation gives you the desired performance of your controlled system, you can try your controller on the real set-up. For this you will need to connect to RemoteLabs and upload a zipfile with all relevant files that define the controller. Details will follow but what you have to do is:

1) Create a .zip file containing:
    - RL_##_controller.slx (the same file as used in simulation)
    - RL_##_controller.mat (**<u>RemoteLabs specific file</u>**, contains information about the RemoteLabs user interface and must contain the execution time (=fixed step size) Ts. Also see section 3.
    - Any other .mat files needed to have a workspace that defines all the variables needed in the controller. Typically this will be the controller.mat file created by running the script controller_parameters_STUDENT.m.
2) Upload the .zip file in the RemoteLabs system

The hardware connected to the real set-up is called the HostPC. The HostPC runs Matlab-Simulink and uses a file called RL_##_TOP_HOSTPC.slx, which has a similar structure as RL_##_TOP_SIM.slx. It uses a subsystem for the controller that is again a model reference. The Simulink file it refers to will be automatically set to the .slx file you have uploaded through the .zip file.

Instead of the subsystem that holds the plant model, that subsystem is now an interface to the setup. The HostPC already has that subsystem loaded and you don't have to provide it. This interface expects the same busses definition as present in RL_##_Busses.mat. So it is crucial **to not modify** this busses definition.

The precise meaning of the RL_##_Controller.mat file and the possible other .mat files will be explained in chapter 3

The details on how to connect and work with the RemoteLabs user interface will be described in Chapter4.

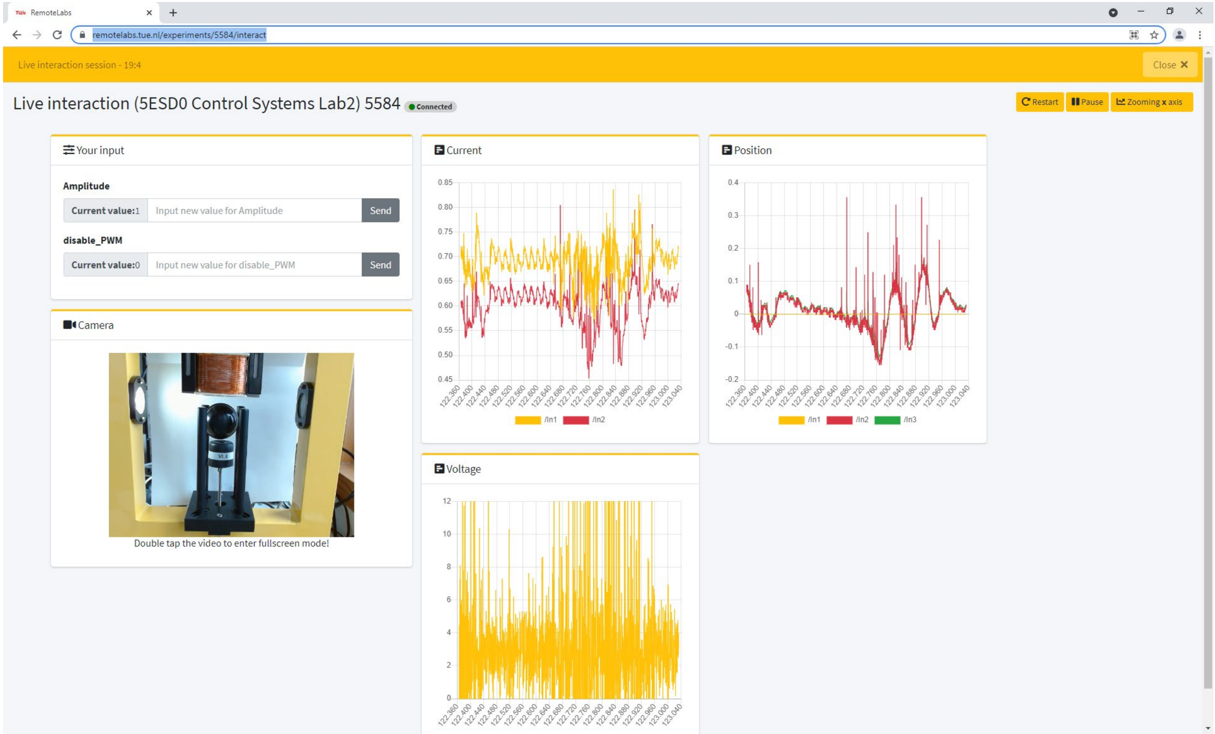
3 Shaping the RemoteLabs UI in Simulink

3.1 Introduction

The User Interface (UI) of RemoteLabs will have two distinct items:

1) **Numeric input fields**, here you can modify the values of parameters in your RL_##_controller.slx while the model is running, so in real time.
2) **Time plotters**, these show the time plots of signals in your RL_##_controller.slx.

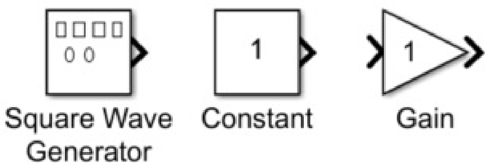
An example of a RemoteLabs UI while doing a MagLev experiment. Variables "Amplitude" and "disable_PWM" are numeric input fields and "Current", "Position" and "Voltage" are time plots.



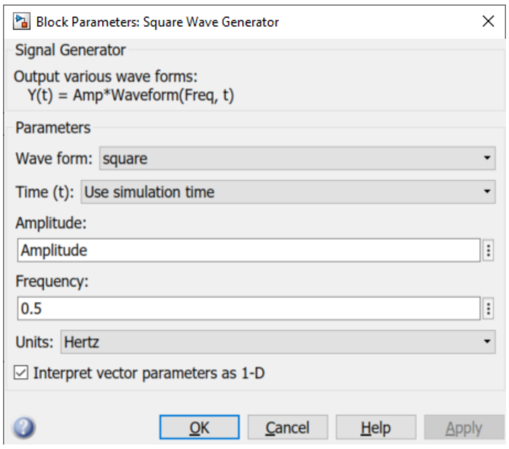
This chapter will explain how you can shape your RemoteLabs UI using certain configurations in your RL_##_controller.slx Simulink model and RL_##_controller.mat.

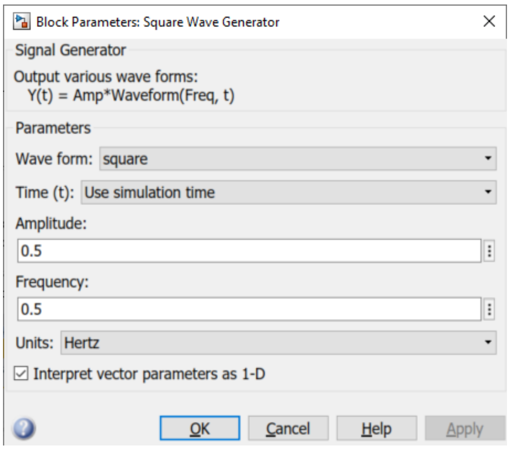
3.2 Tunable parameters (numerical input fields)

The Simulink file typically contains blocks like for example 'constant', 'gain' or 'signal generator' blocks, or other blocks that have parameters.



When you double-click a block in Simulink, for example the Square Wave Generator block, the following dialog box will open.





From the dialog box on the right you can see in that the parameter ‘Amplitude’ is set to ‘0.5’. You can put any value in, but you can also put a variable name here (‘Amplitude’ on the left).

In RemoteLabs, using variables for block parameters gives special functionality, namely:

When parameters are specified **using variables** instead of hard coded values, these parameters become **tunable parameters** in the RemoteLabs User Interface (UI). Every tunable parameter will have a **numeric input field** in the RemoteLabs UI but only if they are included in the file RL_##_controller.mat. This means they can be **modified in real-time** when the Simulink model is loaded and connected to the RemoteLabs physical set-up.

All the tunable parameters will need an ***initial value*** when the model is loaded into Matlab on the HostPC where it will be used for real time code generation. This initial value should be specified in the RL_##_controller.mat file which is also included in the .zip file you uploaded to RemoteLabs. This RL_##_controller.mat file is automatically loaded into workspace of the HostPC.

You can use the script TunableParams.m to create the RL_##_Controller.mat file (for FA) or use the save command in Matlab.

If your model has other parameters that you do not want to tune through the RemoteLabs UI, you have to specify them in another .mat file. The name of this .mat file you can freely choose. Also the name of the .slx file you up load you can freely choose, it does not have to be RL_##_controller.slx, but notice that:

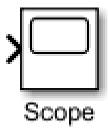
The .zip file you upload to RemoteLabs must contain **only one .slx file** and there should also be **at least one and only one .mat** file with **exactly the same filename** as the .slx file. All tunable parameters must be specified in this .mat file. Typically these names are **RL_##_controller.slx** and **RL_##_controller.mat**. (##=FA or ML).

There is another restriction you should be aware of:

Tunable parameters can only be **scalars of type double**. So for example no vectors or complex numbers are allowed.

3.3 Time plotters

In the Simulink block library, in the ‘Sinks’ part you can find a block called Scope.

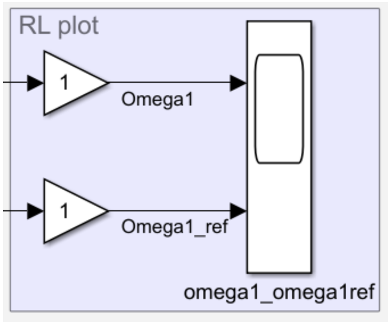


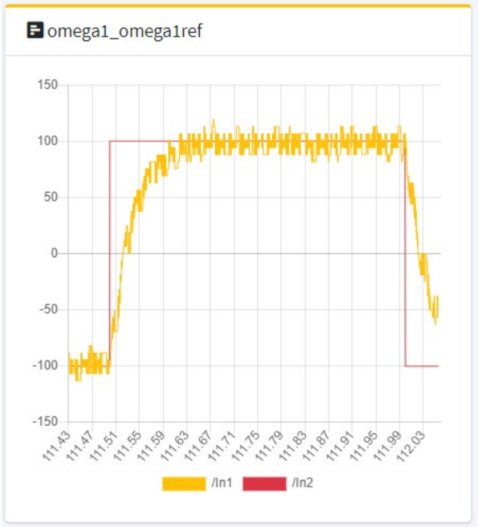
A Scope can be connected to any signal in your diagram. During simulation you will be able to view that signal as a time plot, when opening the scope before or after the simulation. However, for RemoteLabs a Scope as additional functionality, namely:

Signals **connected to a Scope Block** will be shown as **time plots** in the RemoteLabs User Interface (UI) and will be **recorded** in a data file that can be downloaded from the RemoteLabs system.

To make the scopes be visible as a time plot in the RemoteLabs UI, always connect a Gain Block with value '1' to every scope input as depicted below.

If you use a scope with 2 inputs, these two signals will plotted in one plot on the same y-axis.





3.4 Simulink Fixed Step size

When a Simulink model is executed on the real physical set-up, it will run with a so called Fixed Step Size. This is the time step at which the model will be evaluated. So for example if the Fixed Step Size = 0.01, it means that every 0.01 sec the input signals will be taken from the setup hardware interface, all blocks in the model will be evaluated (output values calculated based on inputs) and finally all output signals to the set-up hardware interface will be updated. So when you run for example a controller, this controller will run every 0.01 sec.

For RemoteLabs note the following:

You have to **set this Fixed Step Size** as a variable called **Ts** in in the same .mat file where you specify your tunable parameters (see section 3.2).

To summarize: File RL_##_Controller.mat defines which variables can be changed in real time in the RemoteLabs UI, apart from variable Ts which is used by the system.

4 FlexAxle specific information

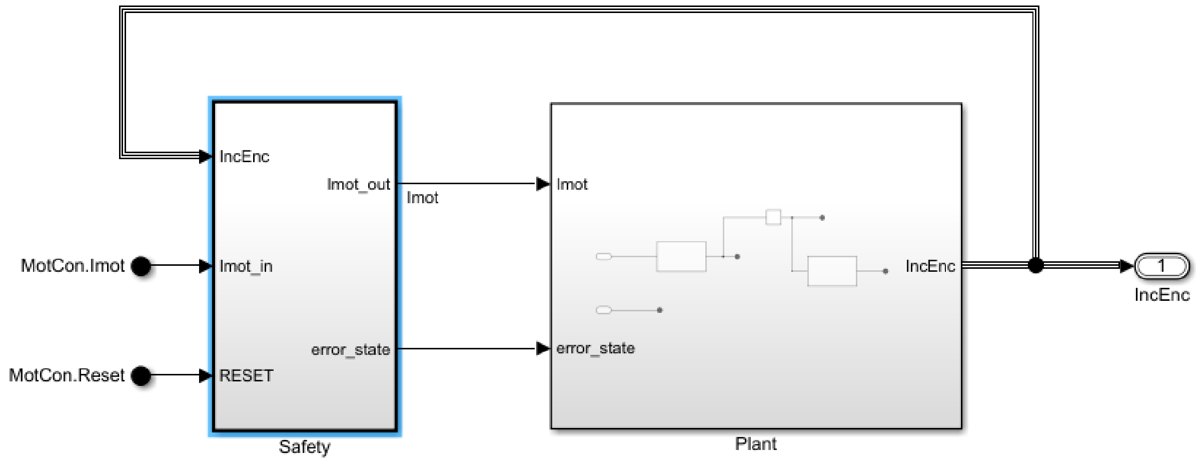
The FlexAxle software you have downloaded from CANVAS includes the following files:

1. RL_FA_Controller_5ESD0.slx
2. RL_FA_Controller_5ESD0.mat
3. RL_FA_System_5ESD0.slx
4. RL_FA_TOP_SIM_5ESD0.slx
5. RL_FA_Busses_5ESD0.mat
6. condesign.m
7. controller.mat
8. TunableParams.m
9. Student_files.zip

The generic meaning of all files, apart from condesign.m and controller.mat, have been explained in chapter 2 and 3. Apart from explaining the meaning and content of condesign.m and controller.mat, this chapter will look deeper into the content of RL_FA_System_5ESD0.slx and RL_FA_Controller_5ESD0.slx.

4.1 The FlexAxle system model

The model of the plant that is implemented in RL_FA_System_5ESD0.slx, when you open this subsystem, it looks as follows:



4.1.1 Safety system

To prevent the FlexAxle from getting damaged by unstable rotations of the masses or to high speeds or motor currents, a safety system is in place. This safety system is of course only useful when operating the real set-up, however, the system is also present in the simulation environment so you can already observe if the safety system gets triggered when simulating your controller.

The safety subsystem in the figure above will check if the following condition is true:

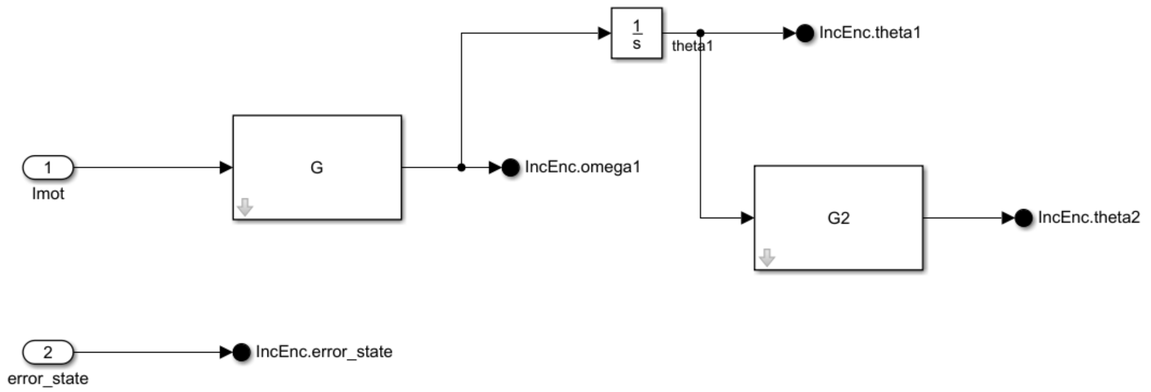
(omega1 > 400) OR (abs(theta1-theta2) > 0.5

If this condition is true the desired motor current (Imot) will not be passed through to the FlexAxle set-up.

The Safety System will also limit the Imot so it will not exceed +/- 1.83 [A].

4.1.2 The plant model

The actual model of the plant looks as follows:



It contains two LTI transfer function blocks, **G** holds the transfer function from Imot to omega1 and **G2** holds the transfer function from theta1 to theta2. These transfer functions need to be specified in Matlab Workspace before you can run the simulation.

Notice that you do not need to edit the RL_FA_System_5ESD0.slx file both the Safety and Plant subsystem are ready to use if Workspace is correctly filled.

4.2 The FlexAxle controller model

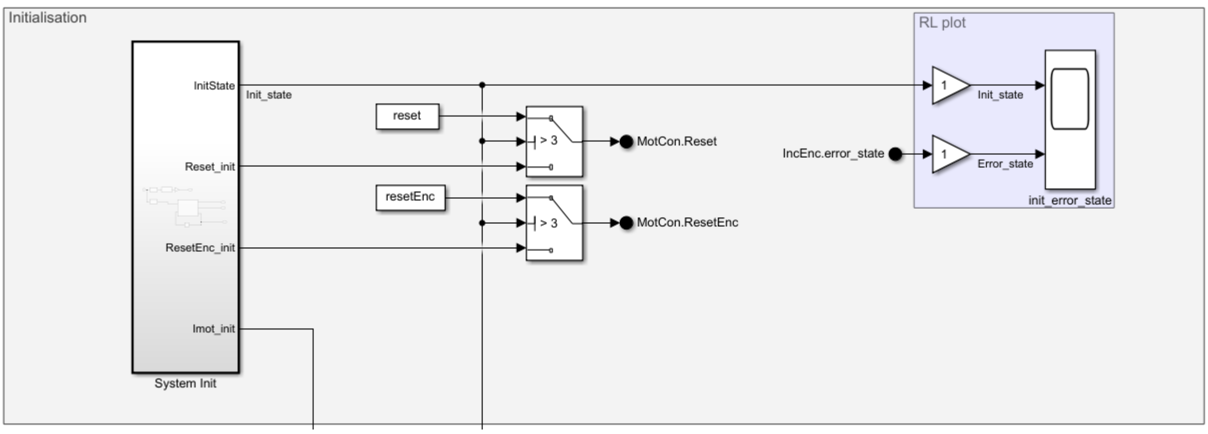
When you open the RL_FA_Controller_5ESD0.slx, you see two distinct areas:

1. Initialisation
2. Controller

4.2.1 Initialisation

At startup the FlexAxle system needs to be initialized. The motor will shortly spin and the position encoders will be set to zero. The signal called 'InitState' will show the progress of this initialization. When finished the InitState will be 4.

Do not change the subsystem called 'System Init' or any of the signal routings of the output signals of that subsystem.

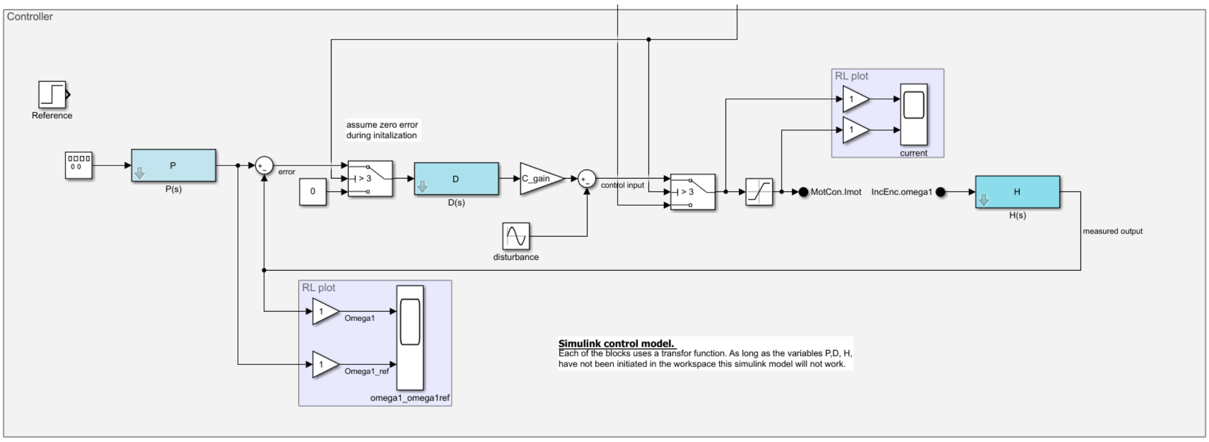


4.2.2 Resetting the system

When your controller settings caused the system to trigger the safety system (variable error_state goes from 0 to 1), you have to change the controller parameters back to values that worked before and reset the safety system. You can reset the state to operational by changing the value of reset from 0->1 or from 1->0, depending on what its current value is. You can monitor the value of error_state in the first time plot on the left side of the page.

4.2.3 Controller

The Controller area shows the control architecture. From the assignments you got for Lab1 will ask you to develop transfer functions for **P**,**D** and **H**. You do not need to change anything in the structure as shown below, apart from maybe adding different signal sources to the reference input of additional scopes if you need them.



Edit the file condesign.m to create a values for P, D and H. The comments in the file will guide you. When running condesign.m a file called controller.mat will be created, you need this file for doing experiments in RemoteLabs.

4.3 Creating the .zip file for RemoteLabs upload.

When you want to test your controller in RemoteLabs on the real set-up you have to upload a .zip file (see section 6.4) containing your controller and all .mat files needed to fill Matlab's Workspace. When creating your .zip file you can give it any name, it should include the following files:

1. RL_FA_Controller_5ESD0.slx
2. RL_FA_Controller_5ESD0.mat
3. controller.mat

The Student_files.zip that you can find on CANVAS also shows this as an example.

Make sure that the .zip file:

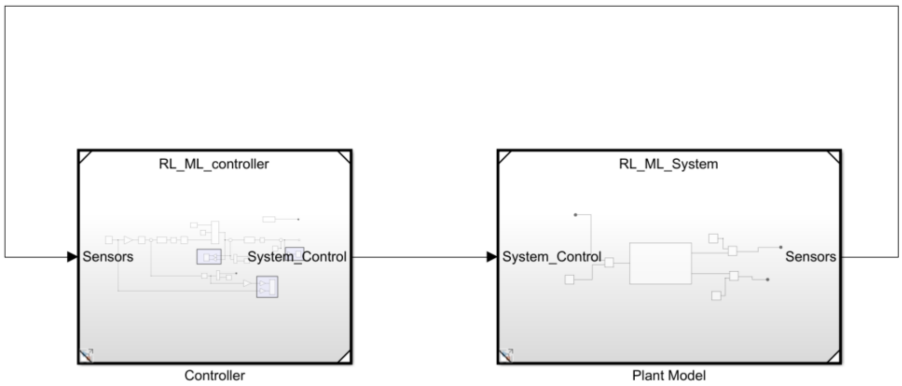
1. Contains only one .slx file
2. Contains at least .mat file with the same name as the .slx file, where this .mat file should at least contain the specification of Ts.

5 MagLev specific information

The MagLev software you have downloaded from CANVAS includes the following files:

1. RL_ML_controller.slx
2. RL_ML_controller.mat
3. RL_ML_System.slx
4. RL_ML_TOP_SIM.slx
5. RL_ML_Busses.mat
6. Controller_paramaters_STUDENT.

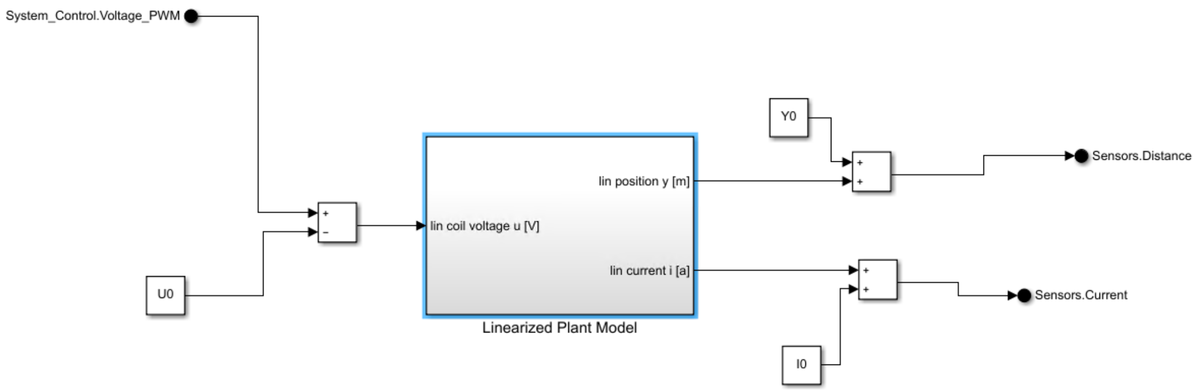
The generic meaning of all files, apart from controller_paramaters_STUDENT.m, have been explained in chapter 2 and 3. Apart from explaining the meaning and content of controller_paramaters_STUDENT.m, this chapter will look deeper into the content of RL_ML_System.slx and RL_ML_Controller.slx which are the referenced models in the RL_ML_TOP_SIM.xls as shown in this figure:



The file RL_ML_busses.mat define the Sensors and System_Control signals in the busses. Do not change this file.

5.1 The MagLev system model

The model of the plant that is implemented in RL_ML_System.slx, when you open this subsystem, it looks as follows:



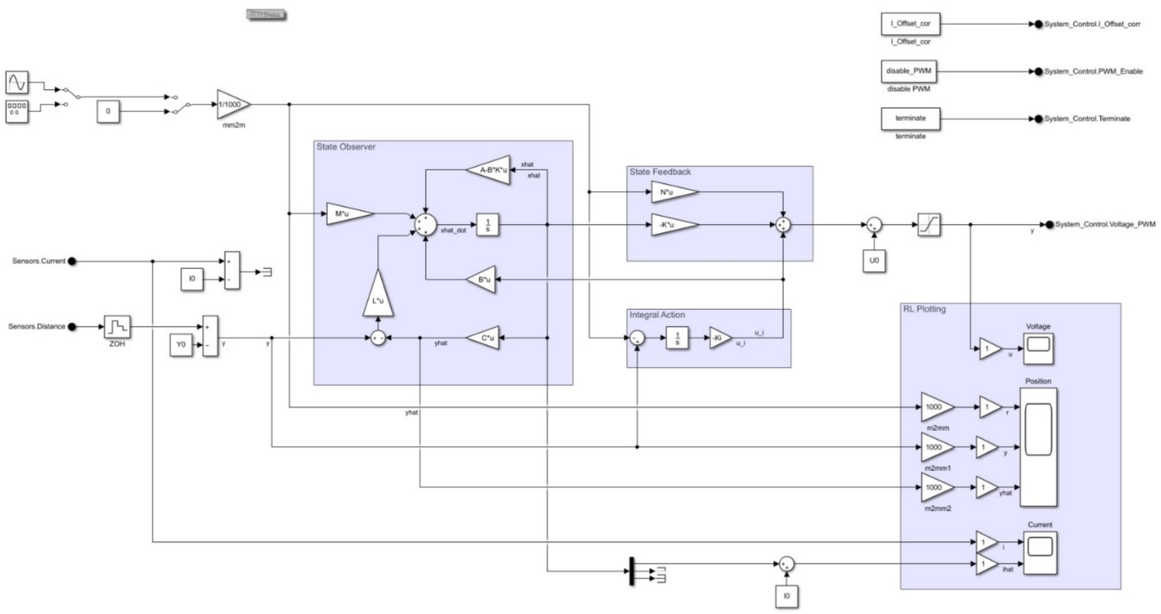
This diagram shows a subsystem containing the linearized plant model with its inputs and outputs added to the operating point defined by U0, Y0 and I0. Defining the linearized plant model is part of the assignment. Doble-click on the model to see the model-structure.

Do not change the bus inputs and outputs, *System_Control.Voltage_PWM*, *Sensors.Distance* and *Sensors.Current*.

5.2 The MagLev controller model

When you open the RL_ML_Controller.slx, you will find the Simulink diagram as shown below. You do not need to make any changes in this diagram, apart from the manual switches (depending on the position-setpoint you want to use) as you simply have to specify in Matlab's Workspace the parameters/matrices of the controller and observer and the operating point values $\mathbf{U_0}$, $\mathbf{I_0}$ and $\mathbf{Y_0}$.

The actual software implementation will be done in controller_parameters_STUDENT.m. The initial version of this file as you download it from CANVAS is partial filled but has a number of 'unfinished' parts in comments where you have to fill the correct code and values and remove the comment indicator (%). The file structure corresponds with the assignment description of Lab2.



The '**disable_PWM'** constant can be used to quickly disable the system in case your controller gives large oscillations to the sphere and might damage things. Please use it when you see the sphere moving wildly, while bumping into its limits. To disable the system make the disable_PWM = 1.

5.3 Creating the .zip file for RemoteLabs upload.

When you want to test your controller in RemoteLabs on the real set-up you have to upload a .zip file (see section 6.4) containing your controller and all .mat files needed to fill Matlab's Workspace. When creating your .zip file you can give it any name, it should include the following files:

1	**RL_ML_controller.slx**	is provided on CANVAS
2	**RL_ML_controller.mat**	Use Matlab save-command to include variable you want to change in the RemoteLabs UI. This file must also include Ts.
3	**controller.mat**	Will be created when running **controller_parameters_STUDENT.m**
4	**Any.mat**	Not required but you can add any parameter you want RemoteLabs to load in the Matlab workspace.

Make sure that the .zip file:

- Contains only one .slx file
- Contains at least .mat file with the same name as the .slx file, where this .mat file should at least contain the specification of Ts and any variables that you want to change during an experiment on the setup.

6 Using RemoteLabs

6.1 RemoteLabs Access

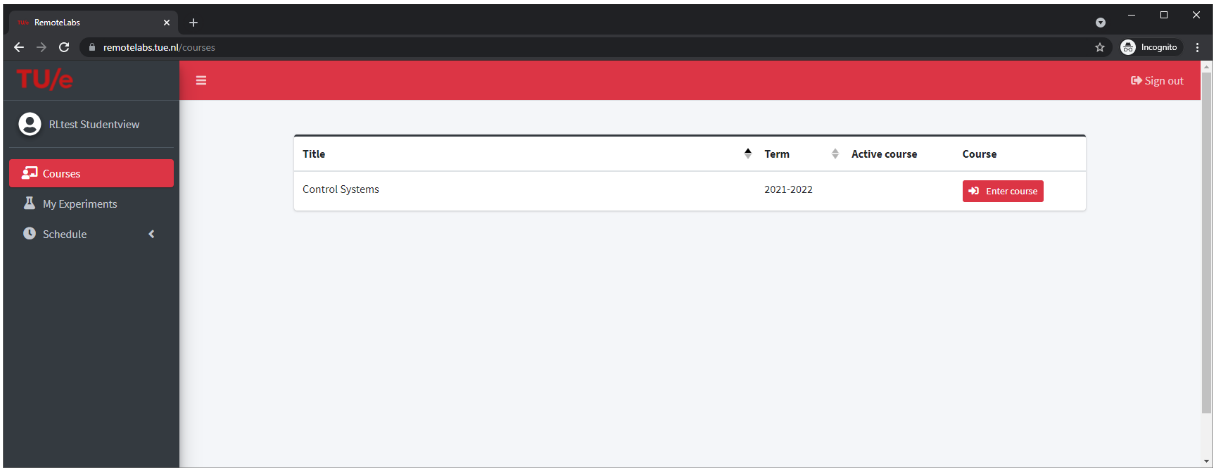
You can access the RemoteLabs system on: https://RemoteLabs.tue.nl/ (Use Chrome or FireFox) :

the page will show the RemoteLabs login:

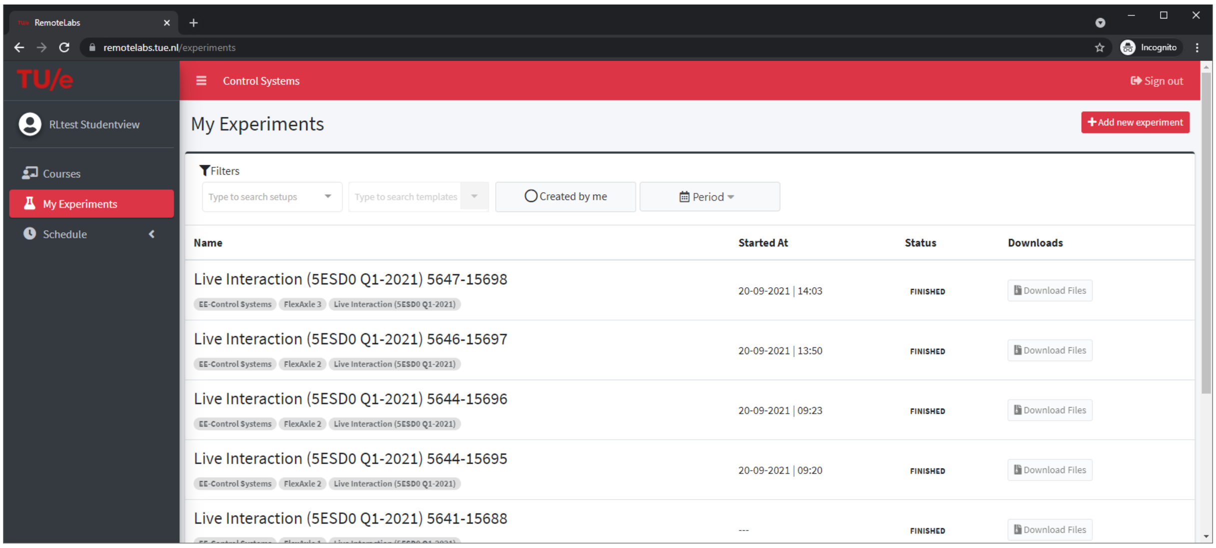
Log in using your TU/e account credentials.

6.2 Schedule an experiment

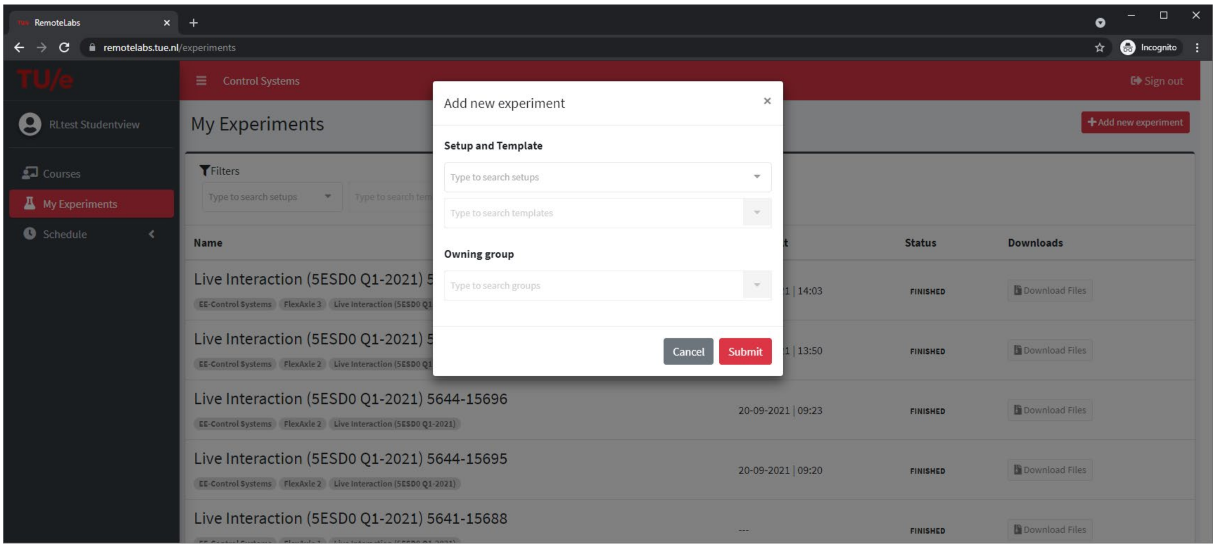
Select the course you are participating in.



Go to My Experiments and select the “Add New Experiment” button in the top right hand corner:



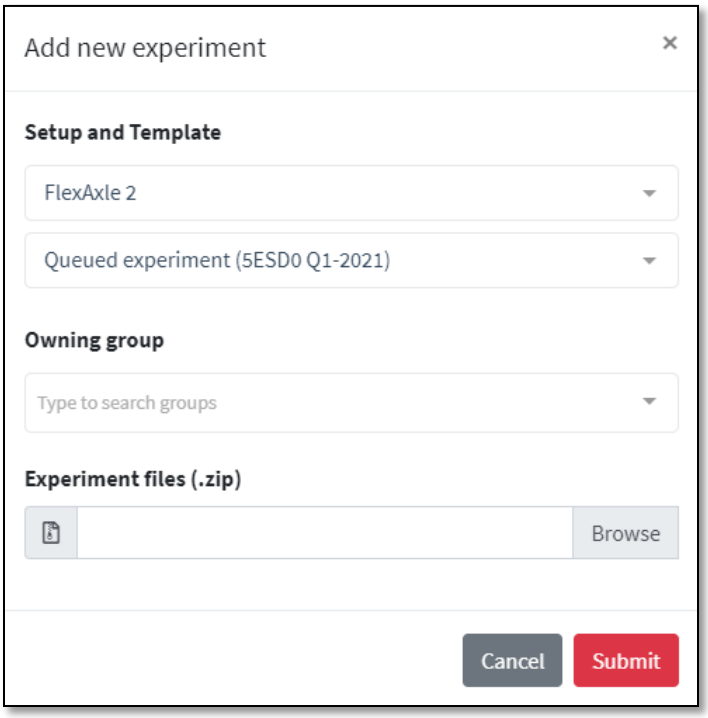
In the dialog box select the setup and the template you want to use (ignore the “Owning group” entry). The setups available will be FlexAxle1..4, and later MagLev1..4.



Depending on your choice of the template (a “Live”, “Lab visit”or “Manual”experiment, or a “Queued” or “Automated” experiment), the dialog box shows some additional entry-boxes as shown in the next sub-sections.

6.2.1 Schedule an Automated (Queued) Operation

Automated (Queued) Operation: using this type of experiment means you will not have any live interaction when the experiment is conducted. The RemoteLabs system will put your experiment request in a queue and will build real-time code using your uploaded .zip file with Matlab/Simulink files and run the experiment once it is your turn. The runtime of your experiment, including code building is set to **90 sec**. All signals that have a scope connected (see paragraph 3.3) will be stored in a datafile (.mat) that can be downloaded once the experiment has finished.



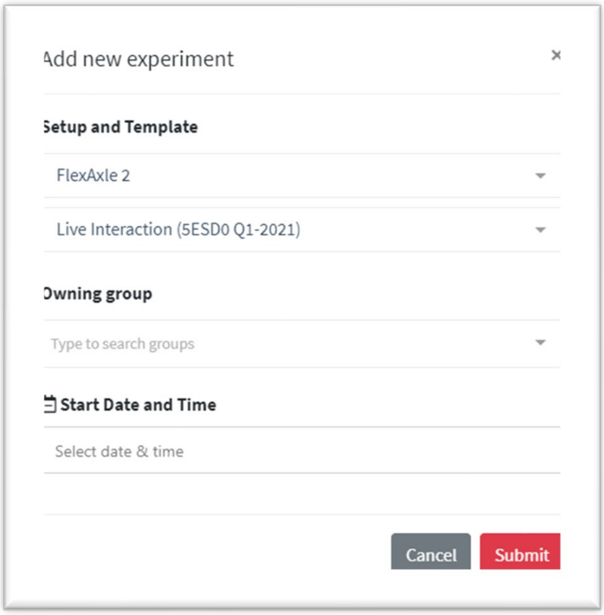
Simply upload your .zip file containing your controller as explained in paragraph 2.3, and press Submit.

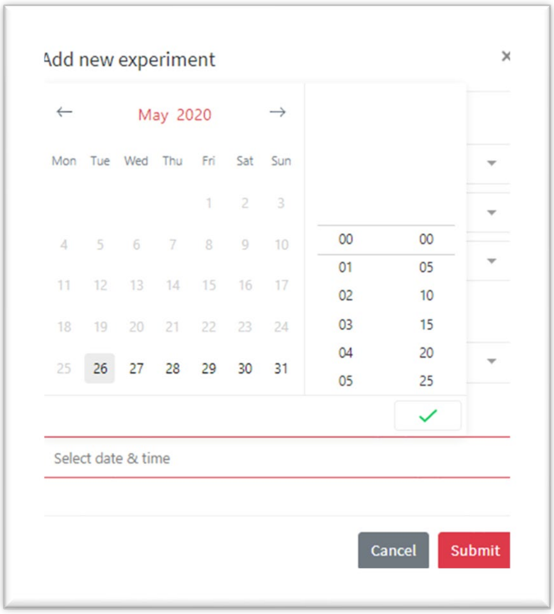
Your experiment will now be placed in the queue.

6.2.2 Schedule a Manual (Live interaction) Operation

Manual (Live interaction) Operation: using this type of experiment you will have the possibility to have live interaction while the experiment is running. You can tune parameters and monitor signals. You can use the Lab-visit template to book a session n the lab.

In the dialog box that opens, Click on Select date & time (do not edit the other fields).

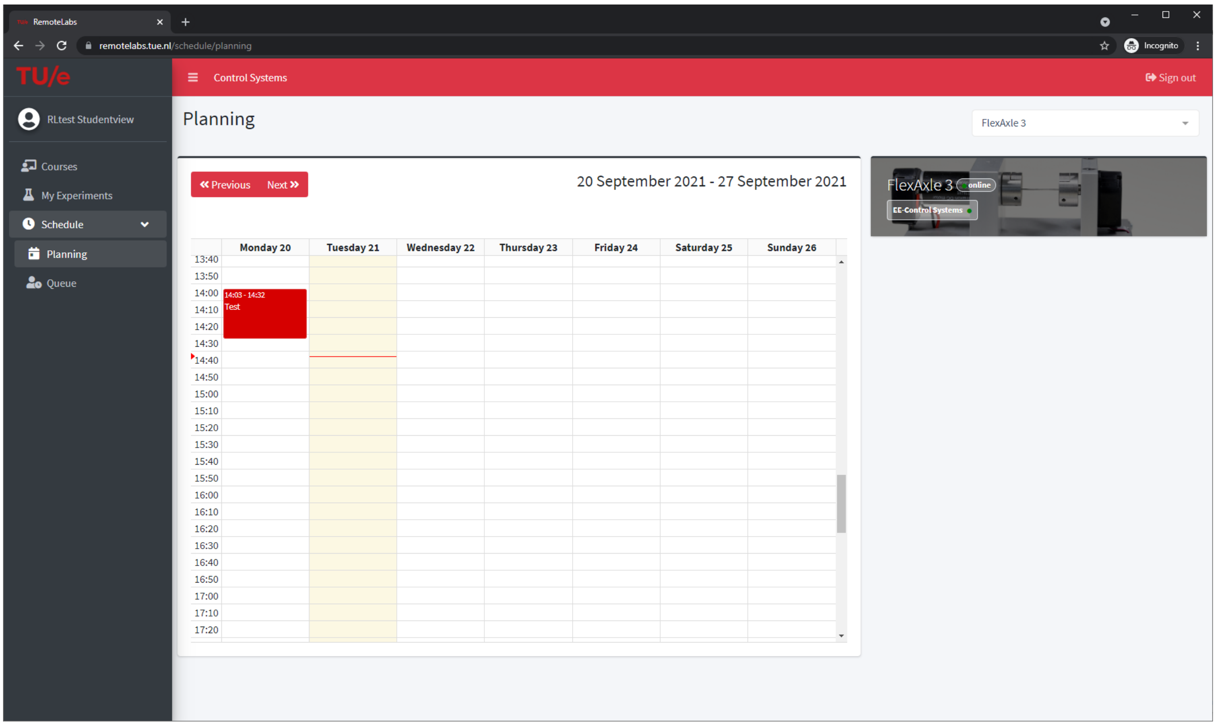




Select when you want to do your live interaction experiment, and press Submit.

6.3 Viewing the planner

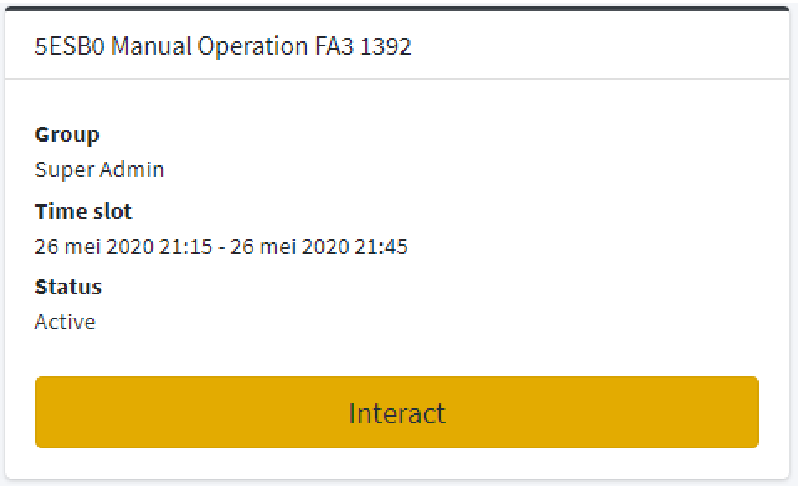
To get an overview of scheduled **Manual (Live interaction) Operations**, select from the main menu: Schedule/Planning. Ik top-right corner select the setup for which you want to see the planning.



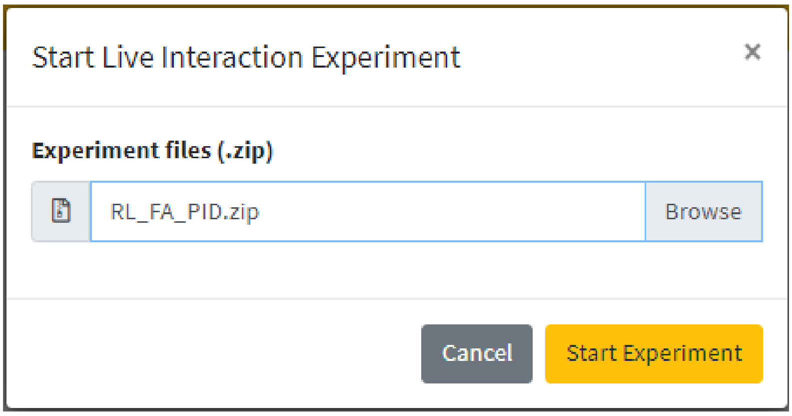
You will get a calendar overview where you can find back the time slots you have booked. In a similar manner you can get an overview of the queue of **Automated Operations** by selecting Schedule/Queue.

6.4 Starting Live interaction

When your time slot for live interaction has started, you can click the red block in the calendar that represents your time slot. You will see the following appear on the left side of the planning calendar:

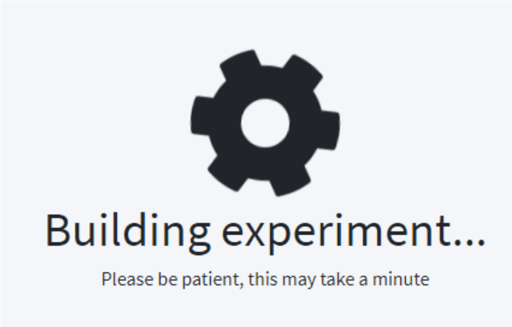


When you click the Interact button, you will get a dialog box where you can upload the .zip file that contains the files as discussed in section 5.3..



Next click Start Experiment.

Now your files will be unpacked and code will be build and deployed on the controller processor platform. During this process you will see:

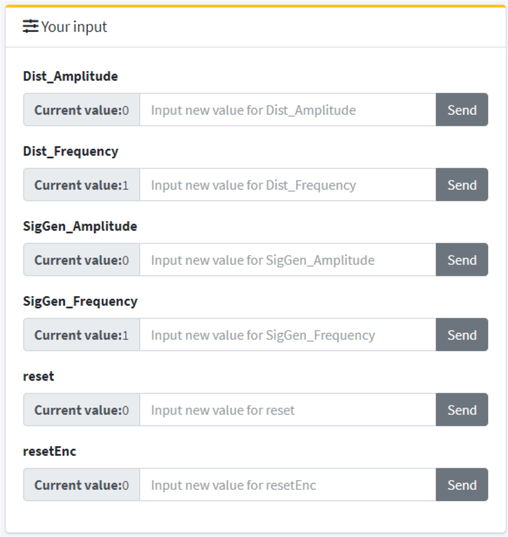


When finished the User Interface for live interaction will appear.

6.5 Live interaction

When your live interaction starts, you will see a User Interface (UI) with the following components.

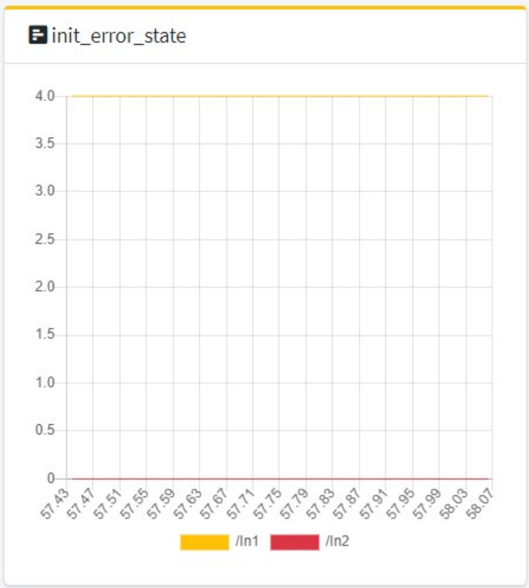
On the top left side you will find the input fields of all tunable parameters.

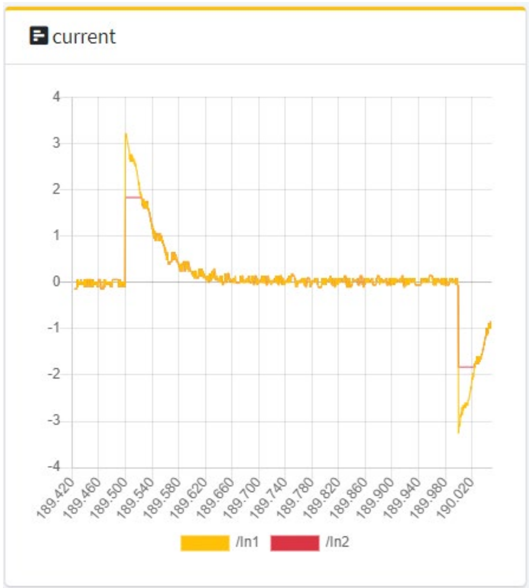


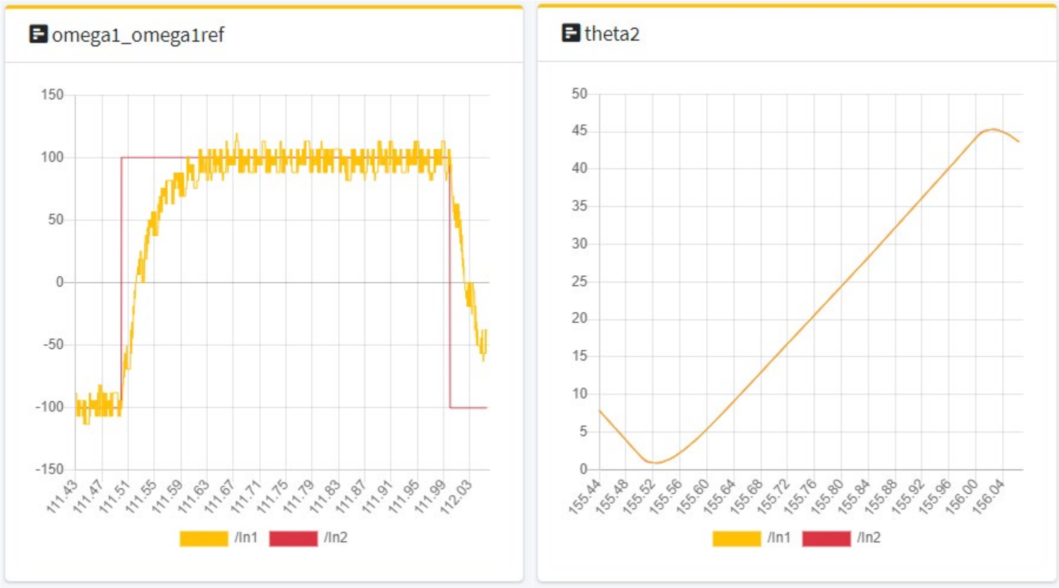
To change a parameter value enter the new value and press the Send button, note that pressing *enter* on your keyboard will not change the value!

On the right hand side you find all time plots based on the Scope block connections you have made in Simulink.

You can drag the plots to create a more convenient ordering on the page.







You can also use the Pause button: 

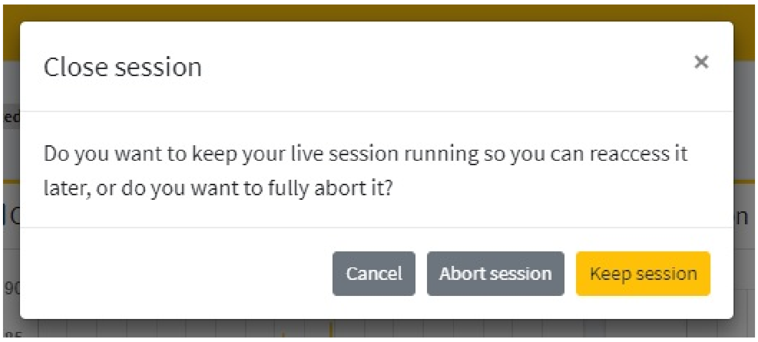
to freeze the plots so you can zoom (click in plot and use scroll wheel) and pan (keep left mouse key pressed and drag).

Zooming can only be done on the time axis and the yaxis using the switch button: 

Press the Play button: 

to switch back to live update mode again.

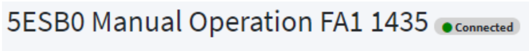
You can do multiple experiments during a live interaction time slot. For this use the Close button on the top right of the UI. Select “Keep sesion” if you want to do another experiment, or “Abort session” if you are finished and want to release the setup.



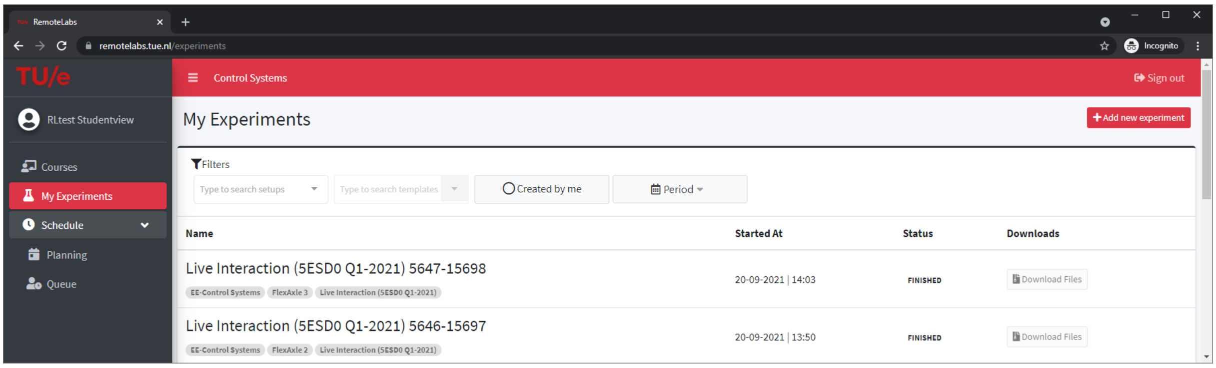
Be aware that when you start a Manual Operation you have a 30 min time slot to do multiple experiments. If you only do one large experiment (so 30 minutes without a restart) your data file will be very large. Measure for a short period and close the session. You will now have a much smaller data file that you can process and use for reporting.

6.6 Downloading and processing results

When you were doing the live interaction, the experiment ID was visible on the top of the page.



If you want to download the data file go to My Experiments:



This will give you a list of all experiments you did, with the most recent one on top:

Press the “Download Files” button in the downloads column to get the data file as a .zip. If status doesn’t show Finished, refresh the web page (F5).

The .zip file contains three files:

1. ***<exp ID>*_errors.zip** file: log file, useful when an experiment fails.
2. ***<exp ID>*_userInput.zip** file: The uploaded controller and associated files.
3. ***<exp ID>*_setupOutput.zip** file: Simulink compilation logfile and **experimentXX_<*exp_ID*>.mat** file.

The file **experimentXX_<*exp_ID*>.mat** contains a timetable with the measurement data. Load this file into matlab workspace and you can access the data as follows:

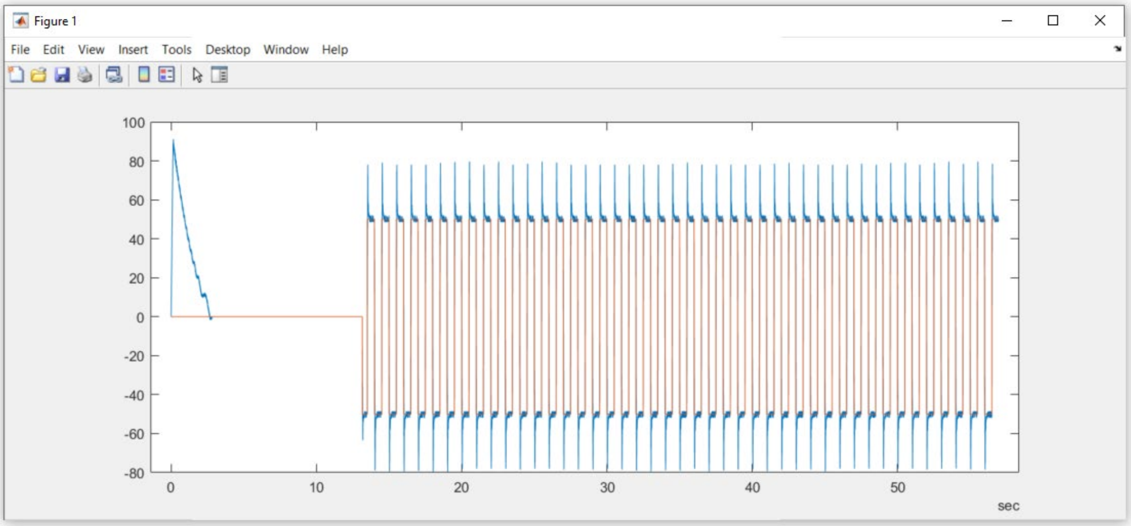
Get the variable names: >> **mdfData.Properties.VariableNames**

The columns can be accessed by index: **>> omega=mdfData.(6);** or **>> omegaref=mdfData.(7);**

To access the time-vector: >> **TimeStamp=mdfData.('Time');**

Screenhot of matlab command window:

```
>> mdfData.Properties.VariableNames

ans =

  1×10 cell array

  Columns 1 through 6

    {'Disturbance_In1'}    {'current_In1'}    {'current_In2'}    {'init_error_stat…'}    {'init_error_stat…'}    {'omega1_omega1re…'}

  Columns 7 through 10

    {'omega1_omega1re…'}    {'theta2_In1'}    {'theta2_In2'}    {'RunTime'}

>> omega=mdfData.(6);
>> omegaref=mdfData.(7);
>> TimeStamp=mdfData.('Time');
>> plot(TimeStamp, omega, TimeStamp, omegaref)
fx >>
```



7 Troubleshooting

If your uploaded .zip file causes errors and will not result in a successful experiment run, please check the following list to make sure you did things correctly:

1) Your zip file is of type **.zip** and not for example **.7z**
2) Your .zip file does not include a folder tree, so only zip the files and the folder their located in.
3) You been using **MATLAB R2019B,** or at least save the .slx file in that version.
4) Your .zip file contains **only 1 .slx file**
5) Your .zip file contains at **least 1 .mat file** with **equal name as the .slx file**
6) You have specified **Ts** in the .mat file referred at 4)
7) Your .mat files will fill MATLAB workspace with parameter values of **all** parameters present in your .slx file.