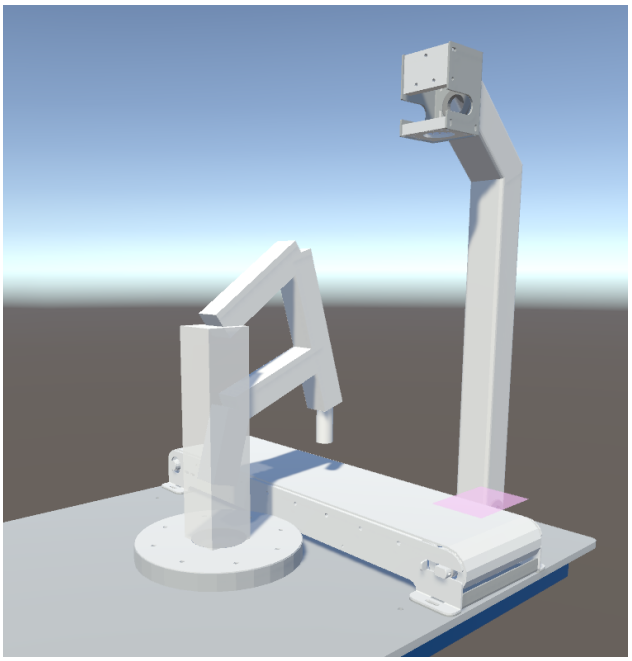
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Robot-arm simulator manual



J.H.M. van Eijk
G.P.M. van Hattum

January 30, 2025

Contents

1 Simulator overview

The simulator of the robot arm has been made with the purpose of giving students an additional tool to explore Stateflow. The intention is not to design or test feedback controllers and implement them on the real robot, as the simulator does not approximate reality sufficiently. Although the simulation capabilities of the simulator are decent, some limitations need to be kept in mind. For example, all sensor data, such as object detection, have been modeled to be stable and 100% accurate. In the real world, this is not possible, so this needs to be taken into account when translating the simulator Stateflow model to the robot Stateflow model.

1.1 Simulator capabilities

Building a simulator for any kind of system is not straightforward. Usually, it is impossible to recreate the behavior of the real system one-to-one in the simulator. In the end, the simulator is built on some assumptions and limitations. The most important of these assumptions and limitations can be found below:

- RTscope is not supported in the simulator.
- The sensors are stable and 100% accurate.
- The vacuum control and the LED ring are omitted.
- The camera detects one object at a time and measures in mm^2 instead of $pixels$.
- The simulation of the robot arm mimics real dynamics.
- The belt is perfectly parallel to the table.

Sections 3 & 4 describe the simulator and its corresponding Simulink file in more detail.

The simulator is supported from Matlab R2021a onward and has been tested on Windows machines. Compatibility with Linux or macOS is not actively supported by the course. It is recommended to first read the manual of the real robot before continuing with this manual.

2 Installation and Setup

To begin, download the zip file from Canvas and extract it. Ensure that the directories are not renamed, as this may impact the simulator's functionality. Once extracted, open the 'Simulink' directory in MATLAB and proceed with adding the required libraries and add-ons.

2.1 Libraries and Add-Ons

The simulator utilizes a custom library to create quintic polynomial blocks, similar to those used by the regular robot arm, but with adjustments tailored to the simulator. This library is automatically installed by running the 'Simulator_Install' function in the MATLAB command window from the './Robotarm_simulator/Simulink' directory.

The 'Simulator_Install' function will also verify if the following toolboxes are installed:

- Simulink
- Stateflow
- Instrument Control Toolbox

If any of these packages are missing, an error will be displayed, and the required package must be installed **manually**. To find and install add-ons, go to the Home tab in Matlab and, in the Environment section, click the 'Add-Ons' icon. The Add-On Explorer opens and displays the list of available add-ons. After installing the missing packages, rerun 'Simulator_Install' to confirm that all dependencies are correctly installed. A success message should appear if all toolboxes are installed correctly.

2.2 Simulator Installation

The simulator is included in the zip file. To start the simulator, click the 'Start Simulator' button in the 'robotarm_simulator_student.slx' file. Alternatively, you can start it from the MATLAB command window by executing 'runSimulator' in the 'Simulink' folder or navigating to the 'Simulator' folder and run 'Robotarm_simulator.exe'

Note: The Simulink model can only run when the simulator is active. Upon first starting the simulator, a Windows security alert will appear:

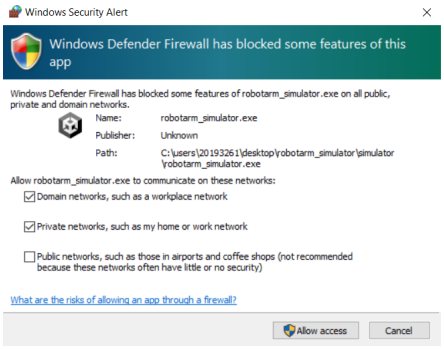


Figure 1: Windows Security Alert

Ensure the app is allowed to communicate on domain and private networks to enable the necessary communication ports. Simulink and the simulator communicate via UDP on ports 5005 and 5006.

If the dialog box does not appear, or if Simulink returns an error such as 'Unsuccessful open: Connection refused: connect', you may need to manually open the ports 5005 and 5006.

3 Simulator

When the simulator starts, UDP communication is started on ports 5005 and 5006. This can increase the CPU load, as well as power consumption and heat dissipation of your computer. It is advisable to close the simulator when making significant changes to your Stateflow model and to keep laptops plugged into a power source. The simulator also provides access to three additional windows: 'UDP Communication', 'Object Settings', and 'Statistics'.

3.1 UDP Communication

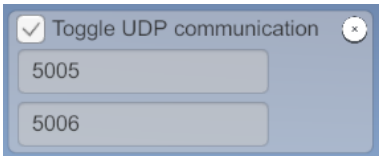


Figure 2: UDP Settings Window

This window allows you to manage the settings for UDP communication. You can stop the UDP communication by clicking the toggle button. Once stopped, the port numbers can be modified using the text fields below the toggle button. The top text field is linked to the "Receiving client" block in Simulink, and the bottom text field is linked to the "Sending client" block. These blocks are discussed in subsection 4.1.

It is recommended **not to change** these settings unless you have confirmed connectivity issues. Port changes should only be made if UDP communication on the default ports is unsuccessful. Changed settings **are not saved** upon exiting the simulator. If you modify these values, remember to update the corresponding UDP blocks in Simulink, as explained in subsection 4.1.

3.2 Object Settings

Figure 3: Object Settings Window

Objects can be added to the conveyor belt by clicking anywhere on the purple area. Although custom objects cannot be created for the simulator, you can modify the diameter and height (both in mm) of the discs. Once an object is configured, you can save the settings by entering a name in the 'Name' field and clicking 'Save'. To remove a saved object from the dropdown menu, select it and click 'Delete'. The 'Clear All' button will remove all live objects currently active in the simulator.

3.3 Statistics

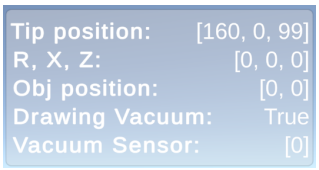


Figure 4: Statistics Window

The Statistics window provides real-time data about the simulator. You can monitor the following:

- Tip position (x, y, z) in mm
- Current motor rotations in the robot arm (R, X, Z) in radians
- Current object position on the belt (x, y) in mm
- Status of the vacuum system (True for vacuum, False for air blow)
- Current vacuum sensor reading in bar

This information helps track the state and performance of the robot arm and objects during simulation.

3.4 Camera viewpoints

At the bottom of the simulator, five buttons labeled 1 through 5 can be seen as depicted in the figure below.

Figure 5: Camera buttons

Using these buttons, the viewpoint of the camera can be changed. Viewpoint 1 is the default viewpoint when starting the simulator. Viewpoint 2 is roughly in the same position as where the camera would be in real life. Viewpoint 3 looks straight over the belt. Viewpoint 4 is a top-down view of the drop-off area. Viewpoint 5 is a front view of the robot arm. It is also possible to change the viewpoints using the corresponding numbers on your keyboard.

Additionally, it is possible to change the field of view for the cameras up to a certain point by using the scroll-wheel of a mouse. Touchpad scrolling is not supported.

4 Simulink

The Simulink model for the simulator is similar to the one used for the real robot arm, though the two are not interchangeable. You can run the Simulink file using the standard 'Run' button under the 'Simulation' tab. **Do not** alter the settings of the Simulink file. One benefit of using the simulator is that the Simulink Profiler can be used to analyze which blocks are consuming the most computation time.

Figure 6 shows the top level of the Simulink file. The contents of the controller subsystem are discussed starting in subsection 4.2. For now, we will focus on the 'Robot Arm Simulator' block, which is rarely accessed by users and may be ignored in most cases.

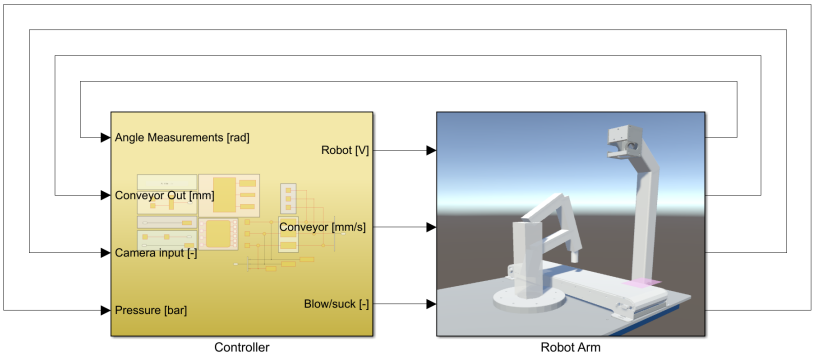


Figure 6: Top level of the Simulink file

When you enter the 'Robot Arm' block, you will see the view shown in Figure 7.

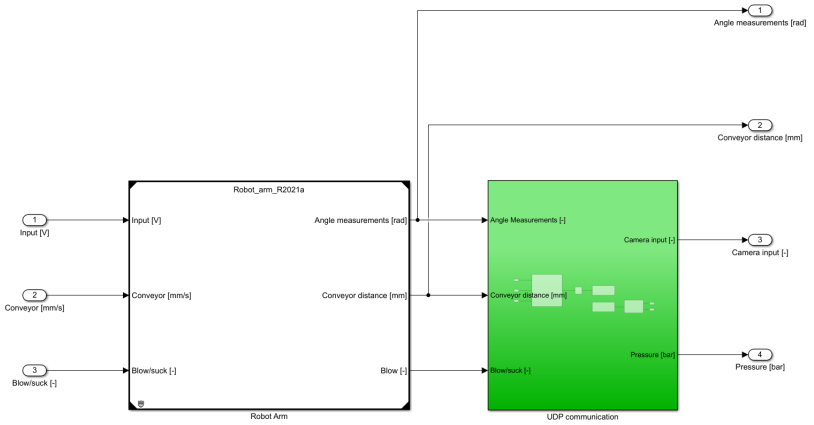


Figure 7: Contents of 'Robot Arm' block

The 'Robot Arm' block contains the dynamic properties of the robot arm simulator. Its contents cannot be modified, as it is a protected model. The 'UDP communication' block is discussed in subsection 4.1.

4.1 UDP communication

The UDP communication block manages data exchange with the simulator. If you modified any UDP settings as described in subsection 3.1, you need to apply those changes here as well. **It is not recommended to modify these settings unless necessary.** The default configuration works for over 99% of users. Figure 8 shows the contents of the 'UDP Data Interchange' block. If you changed the ports in subsection 3.1, update the 'Sending Client' and 'Receiving Client' blocks accordingly.

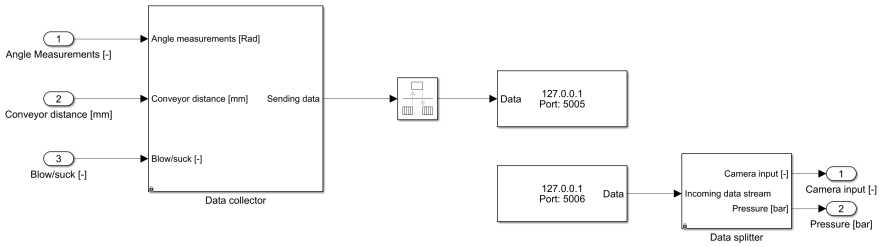


Figure 8: Contents of 'UDP data interchange' block

If the simulator is running, you can verify that the ports are correctly configured by opening the 'Sending Client' or 'Receiving Client' blocks and selecting 'Verify address and port connectivity'. This can assist in debugging if there is no available connection.

If the simulator is not running smoothly, you can adjust the pacing of the simulation to improve performance. By default, the simulation runs in real-time, with 1 second of simulation time matching 1 second of wall clock time. However, if performance issues arise, slowing down the simulation can help.

To change the pacing, go to the Simulink Toolstrip and locate the Simulation, Debug, or Modeling tab. From there, click the dropdown arrow next to the Run button and select Simulation Pacing. This will open a window, as shown in the figure below, where you can adjust the pacing settings.

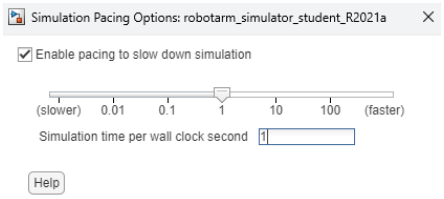


Figure 9: Simulation pacing options

By changing the 'Simulation time per wall clock second' variable to e.g. '0.75' slows your simulation down such that 3 simulation seconds takes 4 seconds in real time. It is not advisable to slow your simulation down by more than '0.5'.

The next sections all discuss the contents in the 'Controller' block.

4.2 Start Simulator

As described in section 3, the simulator can be started by pressing the 'Start Simulator' button. This button executes the 'runSimulator' function, which launches the simulator executable located in the 'Simulator' folder.

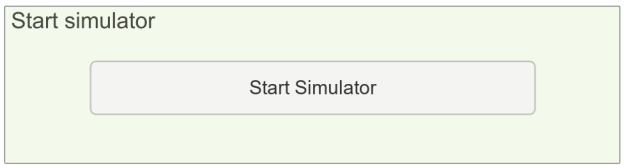


Figure 10: Start Simulator

4.3 Blow/suck vacuum

Similar to the real robot arm, the simulator is equipped with a vacuum gripper. The gripper can either draw a vacuum to grip an object or blow air to release it. Setting the 'Blow/Suck' output to 0 activates the

vacuum, while setting it to 1 activates the air blow. You can toggle between these modes manually using a switch that is double-clicked. To test this, double click the switch while the simulator and Simulink are running and observe the 'Drawing Vacuum' value change in the simulator's 'Statistics' window.

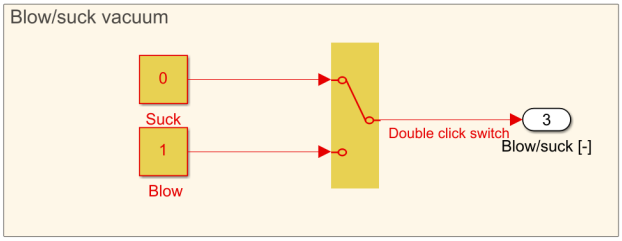


Figure 11: Blow/suck

Note: The simulator does not include a vacuum control loop to regulate the vacuum's strength. The vacuum pressure displayed in the simulator indicates whether an object is attached. A value of 0 means no object is attached, while a value of -0.25 indicates that an object is attached.

Figure 12: Pressure display

4.4 Conveyor control

The 'Conveyor Speed' block controls the conveyor belt's velocity, which can be set between -100 mm/s and 100 mm/s. The current position of the conveyor is displayed in real-time. The simulator does not use a conveyor control loop, as there is no hardware to compensate for.

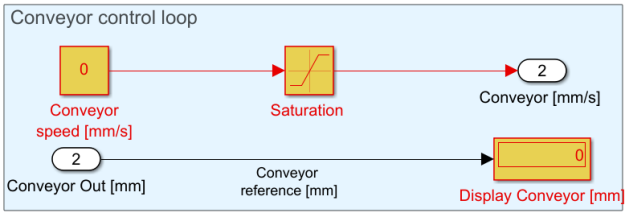


Figure 13: Conveyor control

4.5 Camera Input

The camera block in the simulator differs significantly from that of the real robot arm. For instance, the camera mask cannot be altered, and you won't be able to see a plot of the current camera view. Instead, objects in the simulator are uniformly red and automatically detected when placed under the camera. There's no need to start the camera separately.

The camera detects only one object at a time, and the simulator prevents multiple objects from being placed under the camera simultaneously. The camera outputs similar data to the real system, including object coordinates in millimeters (as a 2x1 vector) and object area in square millimeters (instead of pixels). The delay in the simulator is different from the delay on the real system as the delay in the simulator is defined as the difference in time between sending and receiving data to and from the simulator (in seconds).

While the camera block can be adjusted, this is usually unnecessary. **Note that** the simulator does not feature a controllable LED ring, as it is absent in the simulator.

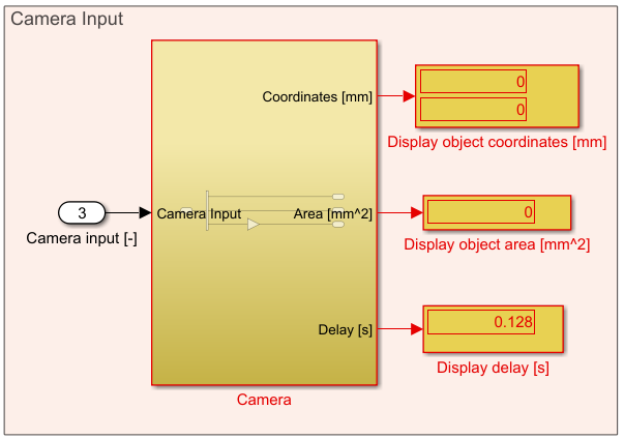


Figure 14: Camera input

4.6 Stateflow

The Stateflow block in the simulator is identical to the one used for the real robot arm. For more information on how to use Stateflow, refer to the robot arm manual.

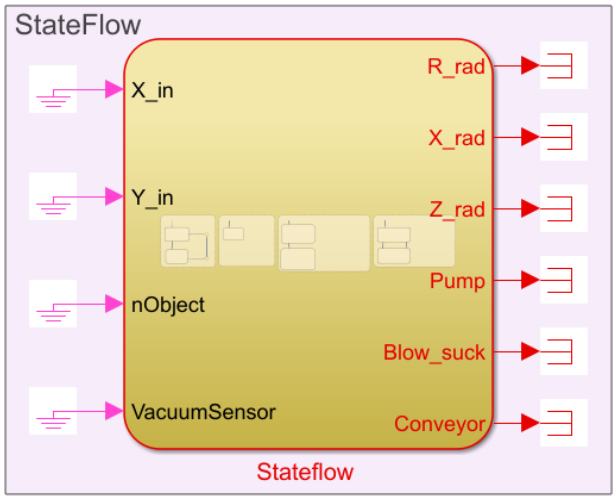


Figure 15: Stateflow

4.7 Controller

The controller section of the Simulink file is where you can implement control logic. It already includes a feedback loop, which sends control signals to the simulator (plant) and receives feedback measurements to close the loop. There are separate subsystems for each motor controller, as well as feedforward and reference inputs for each motor.

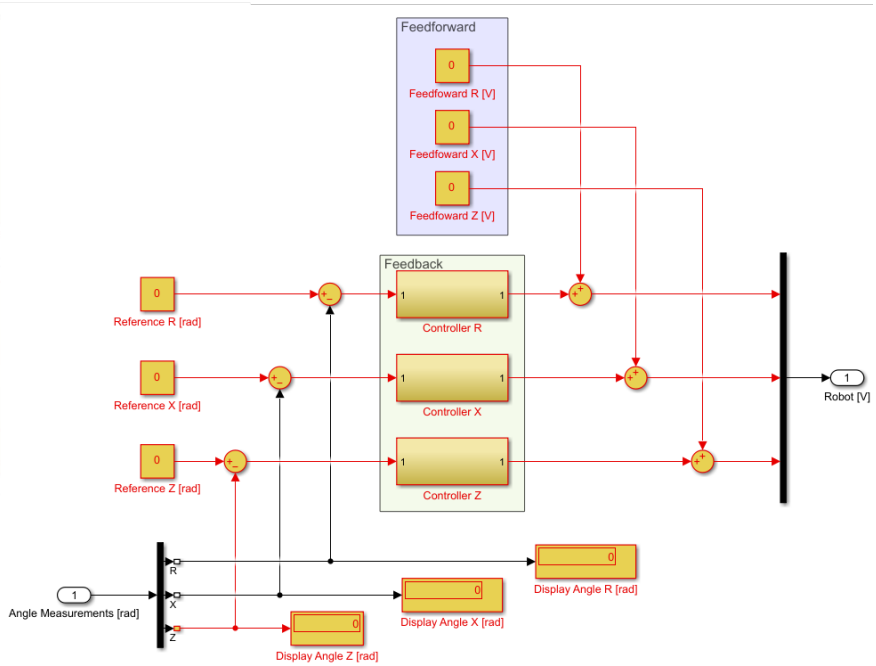


Figure 16: Controllers

While the simulator can run without controllers, doing so results in less accurate control. The robot arm and the simulator have nearly identical dynamic behavior, so a well-tuned controller for one should work well on the other. Nonetheless, it is advisable to validate your controllers rather than assume they will work flawlessly.

4.8 Trajectory generation

As with the real robot arm, the simulator uses custom blocks to generate smooth trajectories during operation. These 'Quintic Polynomial' blocks are discussed in the robot arm manual. It is important to note that the blocks used for the real robot arm are incompatible with the simulator. However, the usage is the same, with only some internal properties adjusted. Simulator-specific blocks are marked with the suffix '(simulation)', so ensure you use the correct blocks when adding them. These blocks can be found in the 'Robotarm Simulator Library' in the 'Library Browser'. For more details, refer to the robot arm manual.

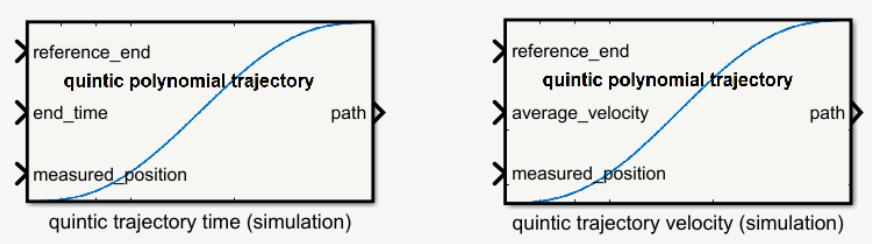


Figure 17: Quintic polynomial blocks

4.9 Measurement

As previously mentioned, RTscope cannot be used with the simulator. Therefore, it is not possible to measure the Frequency Response Function (FRF) of the robot arm using RTscope. However, you can measure the FRF using alternative methods. Controllers designed for the real robot arm should generally work with the simulator, as they share the same dynamic properties. Nonetheless, it is advisable to validate your controllers rather than assume they will work flawlessly.

4.10 Known errors and their interpretation

As of yet, there are two common errors that users are expected to encounter during the use of the simulator. These errors and their interpretation are found below.

Connection error

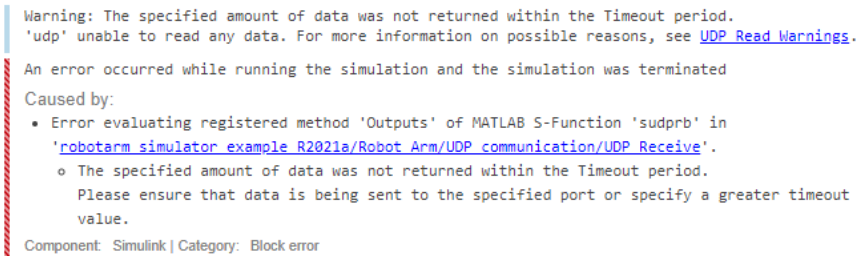


Figure 18: Connection error

This error occurs when the UDP communication between the simulator and Simulink is not active. This can have the following causes:

1. Simulink was started without the simulator running.
2. The simulator was stopped while Simulink was running.
3. The 'Toggle UDP communication' toggle is set to 'False'.
4. One or both UDP ports are not open for communication

If this error has appeared unexpectedly, make sure that:

1. The simulator is running
2. The 'Toggle UDP communication' toggle is set to 'True'.

If this does not resolve the error, try using the 'Verify address and port connectivity' button in the 'UDP data interchange' block as described in subsection 4.1 to verify port connectivity. When a port does not appear to be open, consider changing the UDP ports as described in subsection 3.1 and subsection 4.1.

Assertion error

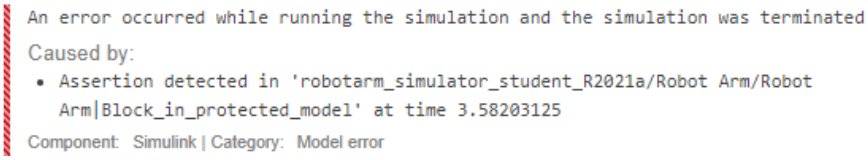


Figure 19: Assertion error

An assertion error appears when one of the robot arm motors exceeds its rotational limit. Figure 19 is an example of an assertion error. Unfortunately, the error does not indicate which motor has exceeded its limits. You may be able to find this by placing scopes on the input of the motors and seeing which one exceeds the limit.

5 Feedback

The simulator is a relatively new addition to the robot arm CBL. We welcome feedback from students who have used it during their project. If you have any questions or suggestions about the simulator during the course, feel free to consult one of the TAs. At the end of the project, a form will be distributed where you can submit your feedback.