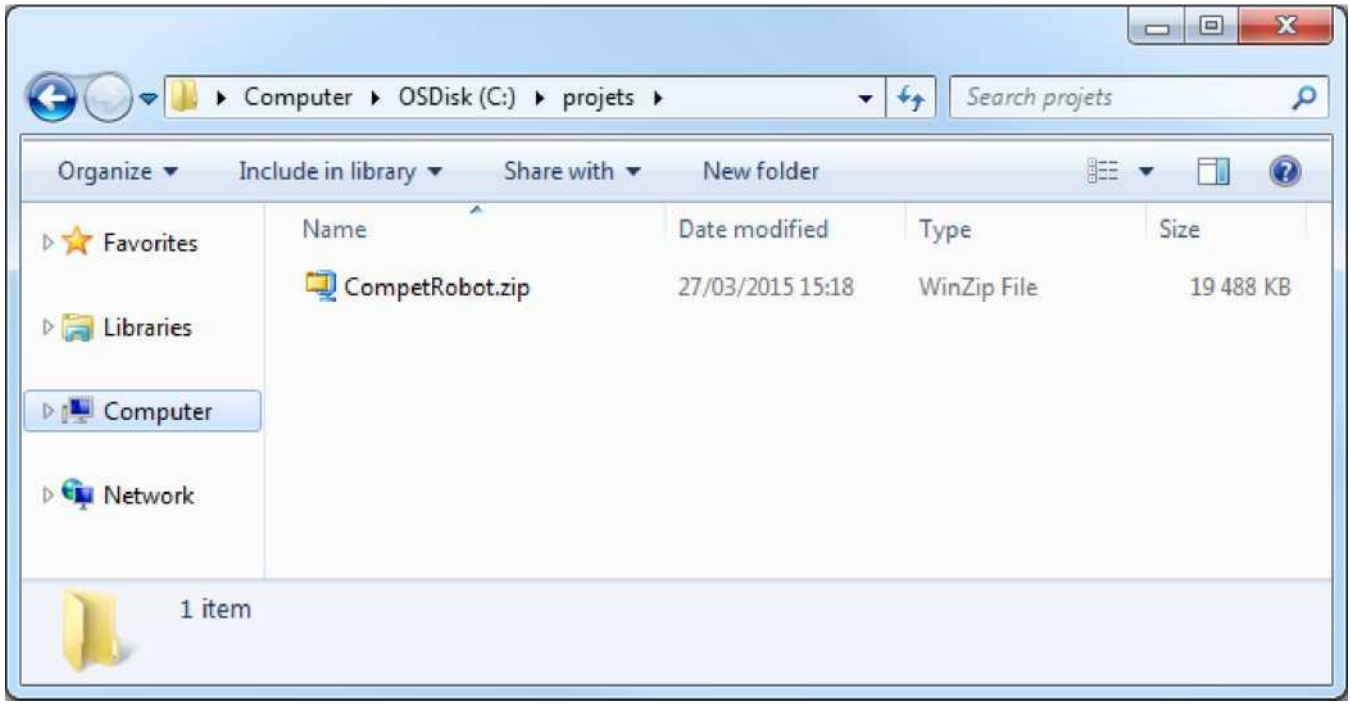
Mission on Mars Robot Challenge
Participant Guide

Downloading the project

The project on which the competition is based is available on the central MATLAB web site. Download the CompetRobot.zip archive containing the project with the technical elements you will need (robot simulation model, documentation, etc.). Once you have downloaded the archive, save in the directory of your choice. In the following screen shots, this is C:\projects, but you can use whichever directory you like.

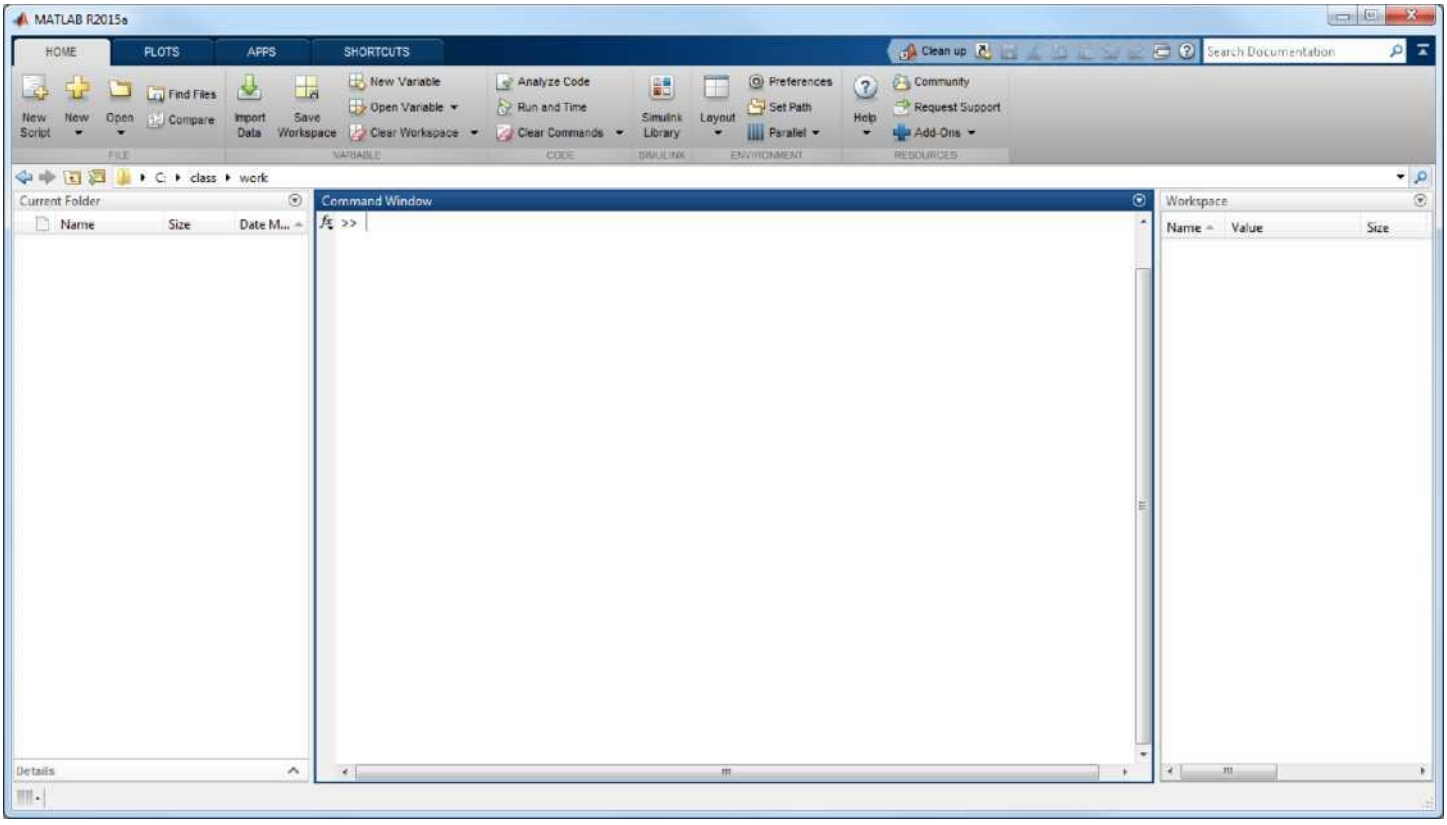


Running MATLAB

Once installed, MATLAB runs like any software application. You can go the Start menu and click on "All programs" then select the directory where you have installed MATLAB.

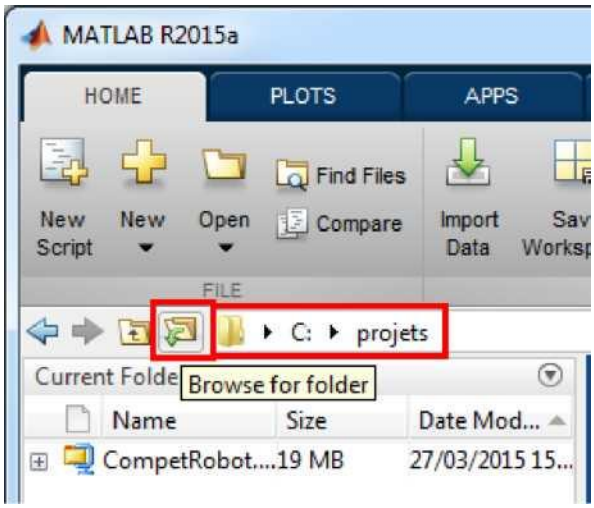
You should have an R2015b sub-directory containing the MATLAB R2015b executable used to run MATLAB:

When you click on this icon, the MATLAB "splash screen" (or download window) should open a few minutes while the application starts up. Then, you should see the following MATLAB interface:

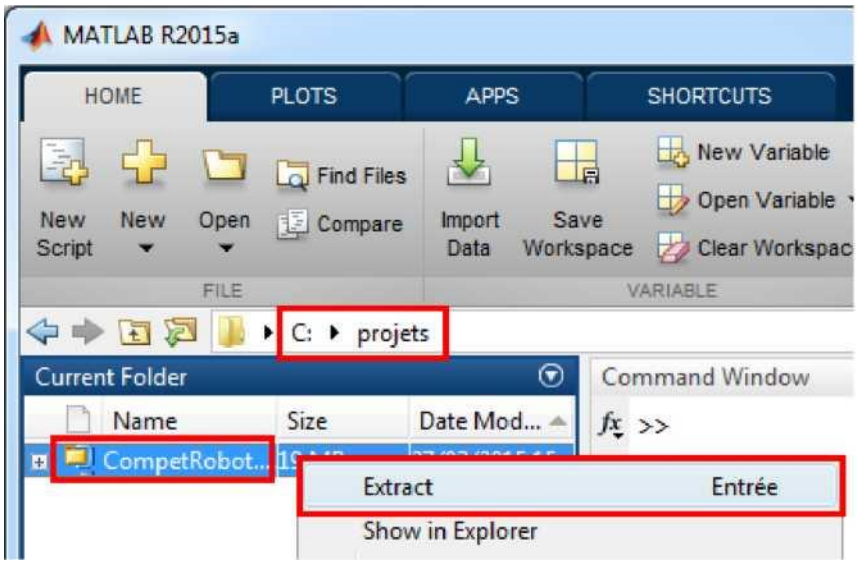


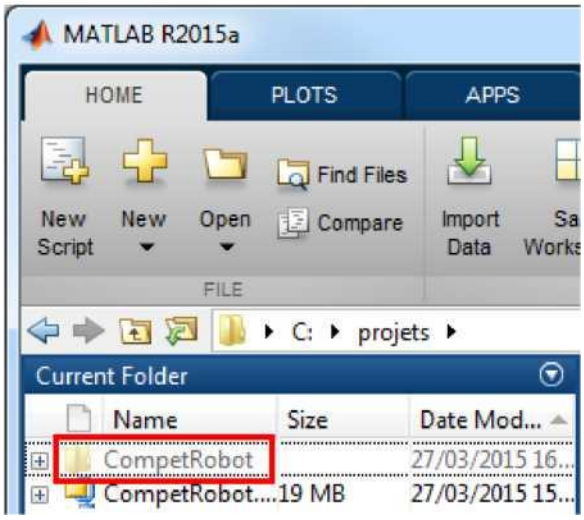
Extracting the Simulink project

Once MATLAB has started up correctly, go to the directory where you have copied the RobotCompet.zip archive. You can use the button as shown below:



Once in the right directory, you should be able to see the CompetRobot.zip archive you copied earlier in the MATLAB Current Folder window. Click right and select Extract in the contextual menu to unzip the archive. This should display the CompetRobot folder, as shown below:

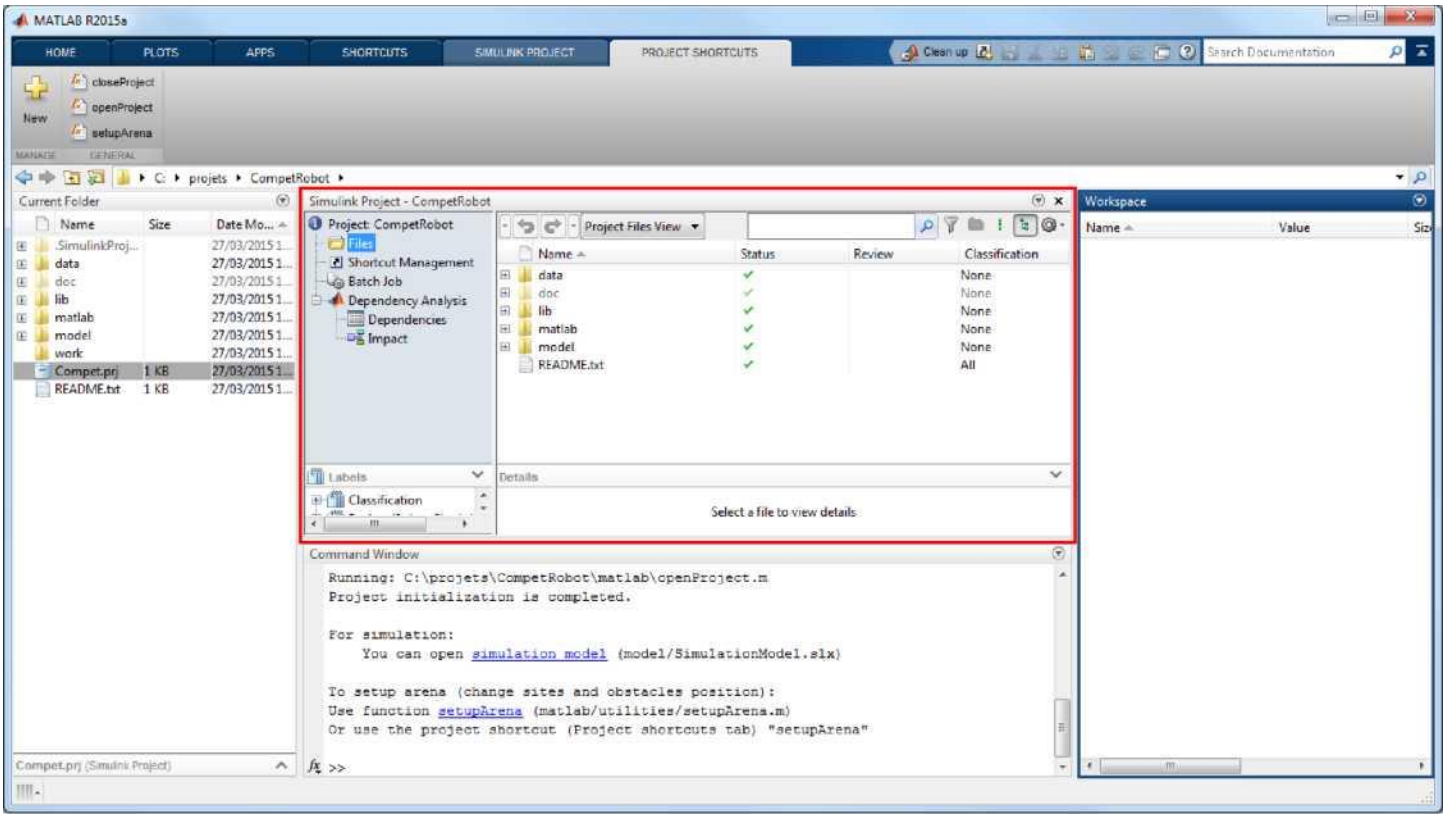




Opening the Simulink project

Once you have unzipped the CompetRobot folder, open it and double-click on the Simulink Compet.prj project file:

The Simulink CompetRobot project window should then open in the MATLAB interface. In our example, it opens in the middle of the interface screen, but this may not be the case on your screen:



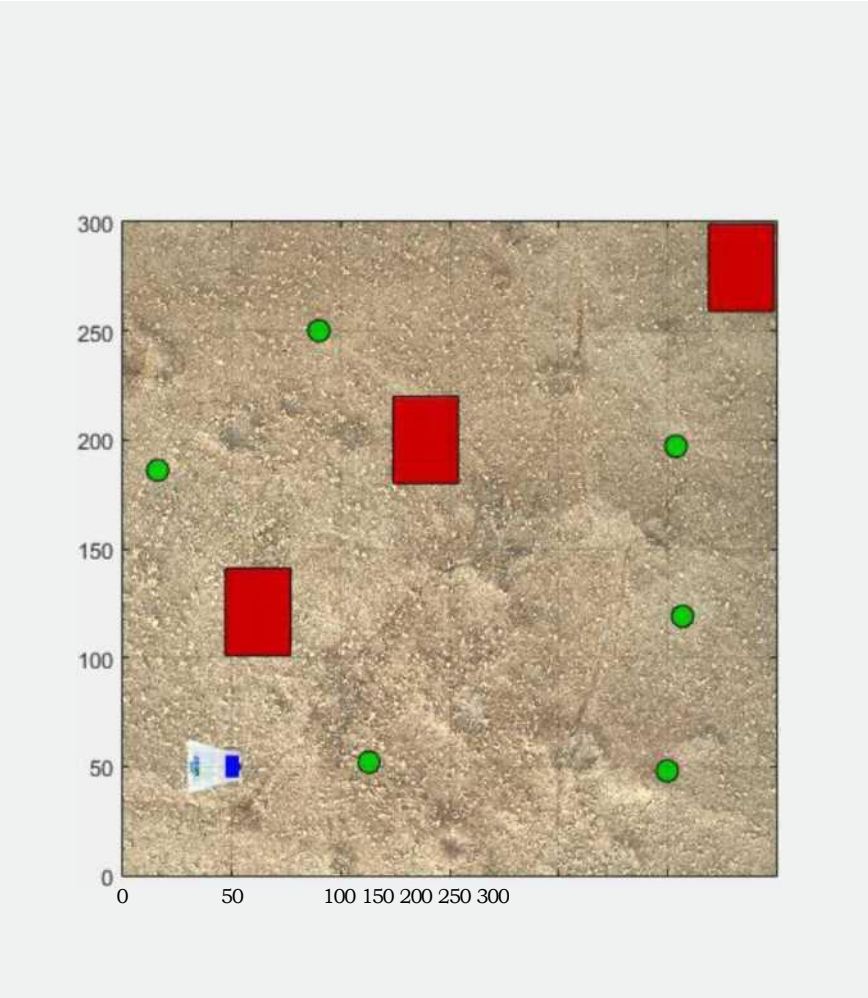
Once you have opened the project, you can do two things:

- Manage different configurations in the arena where the robot is going to perform,
- Improve the robot command algorithm in the Simulink model.

These two stages are described in the following part of this document.

Managing arena configurations

We will provide a user-friendly tool you can use to change the position of the site locations and obstacles in the arena. This is what it looks like:



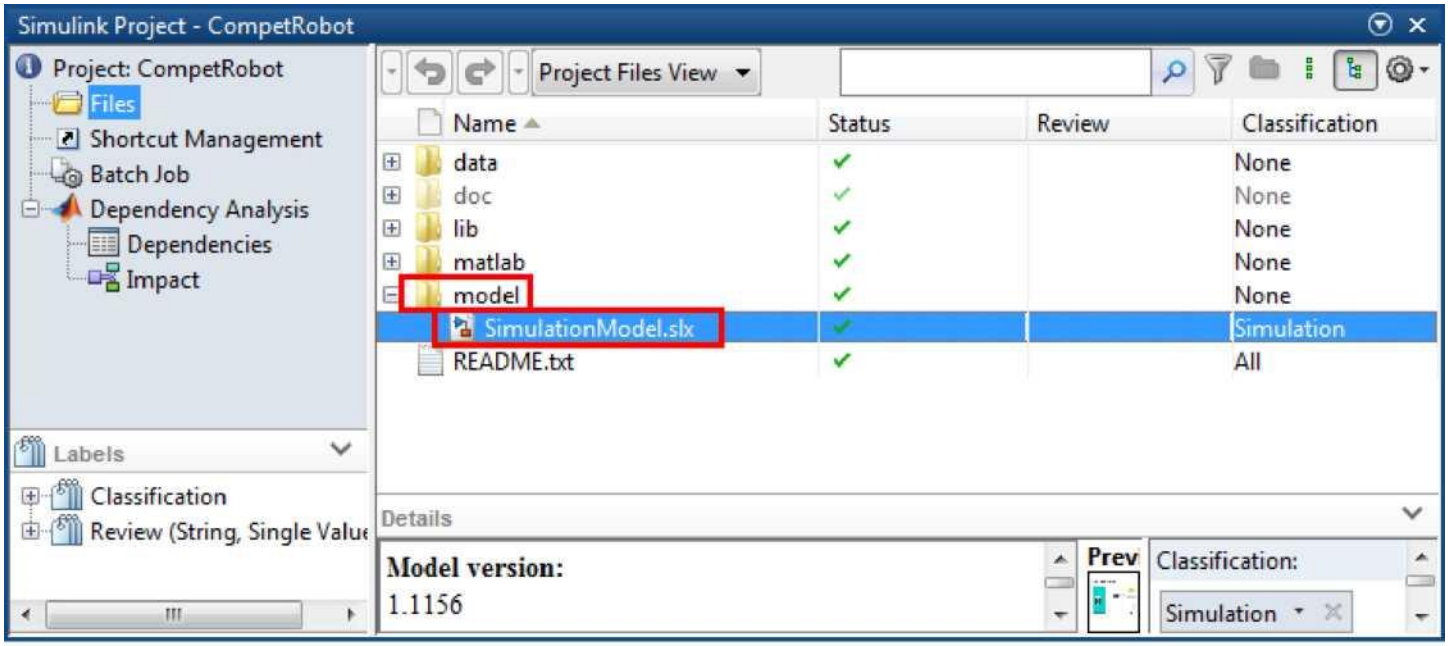
To manage arena configurations, simply use the setupArena shortcut (available in the toolbar, in the `Project` Shortcuts tab).

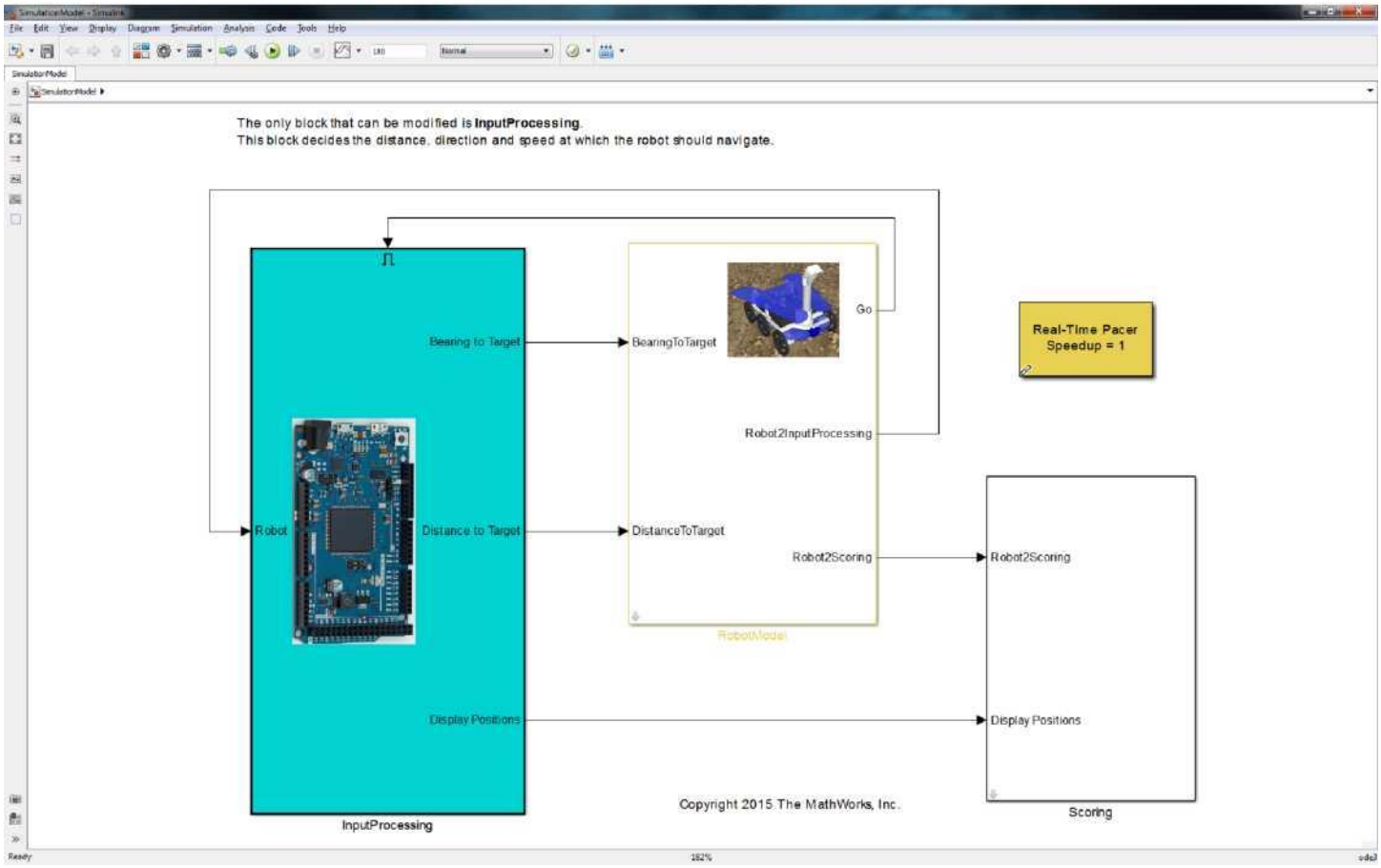
You will find detailed documentation in the Doc\SetupArena.html directory.

Remember that you will not know where the sites and obstacles are located on the day of the competition.

Opening the robot simulation model

Go to the model directory and find the SimulationModel.slx file. Double click on the file to open the model in the Simulink interface window.





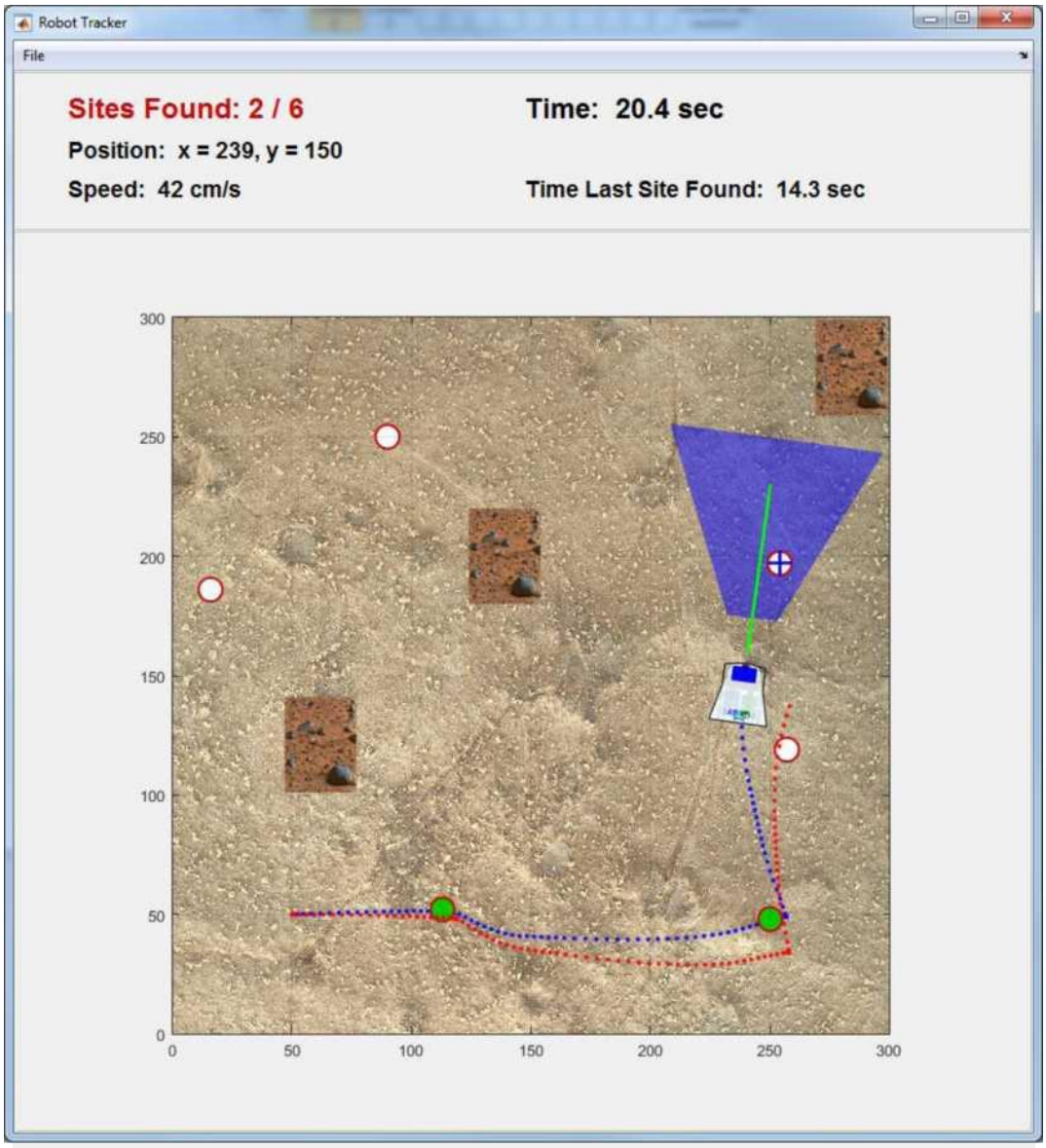
This window displays the simulation model.

Launching the simulation

Before studying and modifying the model, it is important to understand how to run a simulation, so you can see what happens when you modify and experiment with the model.

Click the "Run" button shown in the screen shot below, to run the simulation. This button checks for any errors in the model and if none are found, runs the simulation.

The program checks and compiles the model, then opens the following window.



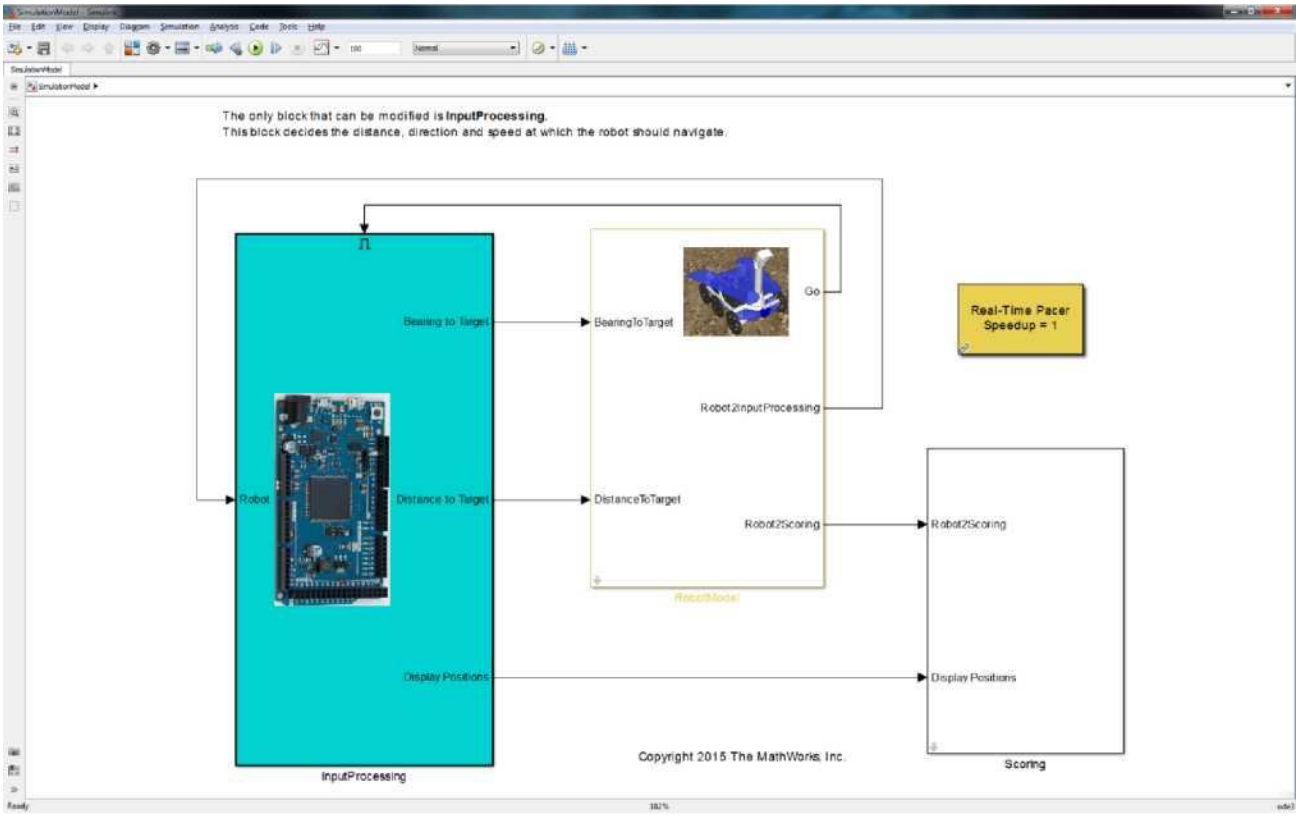
The simulation window:

- Circles show the sites the robot has to visit:
    - The white circles are the sites that the robot has not yet visited.
    - The green circles are those that the robot has already visited and where it has remained stationary for at least 3 seconds.
- Obstacles are represented by stones. If the robot touches one of the obstacles, the mission stops immediately...
- The robot's trajectory is shown by:
    - The blue dotted line, which shows where the robot has actually travelled in the arena.
    - The red dotted line, which shows where the robot thinks it is according to the wheel sensors. The two lines are not identical because the robot tends to slip on the arena surface.
- The blue box in front of the robot shows what the embedded camera can see. When the camera can see several sites, the distances and angles are available for processing.
- The line in front of the robot shows the position of the distance sensor. It is green if no obstacle is detected, i.e. if no obstacle is within a distance of 10 to 80 cm. If an obstacle is within that range, the line turns red and displays the distance to the obstacle. The distance sensor works like a radar and constantly bounces a signal back and forth to scan its environment.
- This window also shows:
    - The number of sites found by the robot.
    - The robot's current position and speed.
    - Time elapsed since the robot started out.
    - Time elapsed between the start and the last site visited.

You can modify the model and run as many simulations as you like to immediately see how your modifications affect the robot's behavior.

Model overview

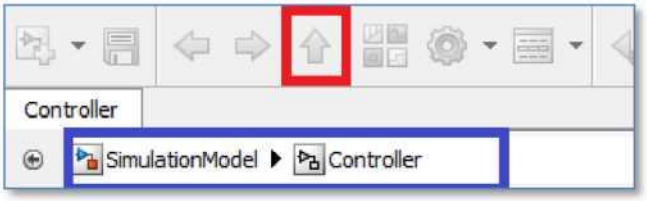
The figure below shows the system simulation model as it is displayed on opening. It includes the program that will control the robot as well as an environment modeling algorithm



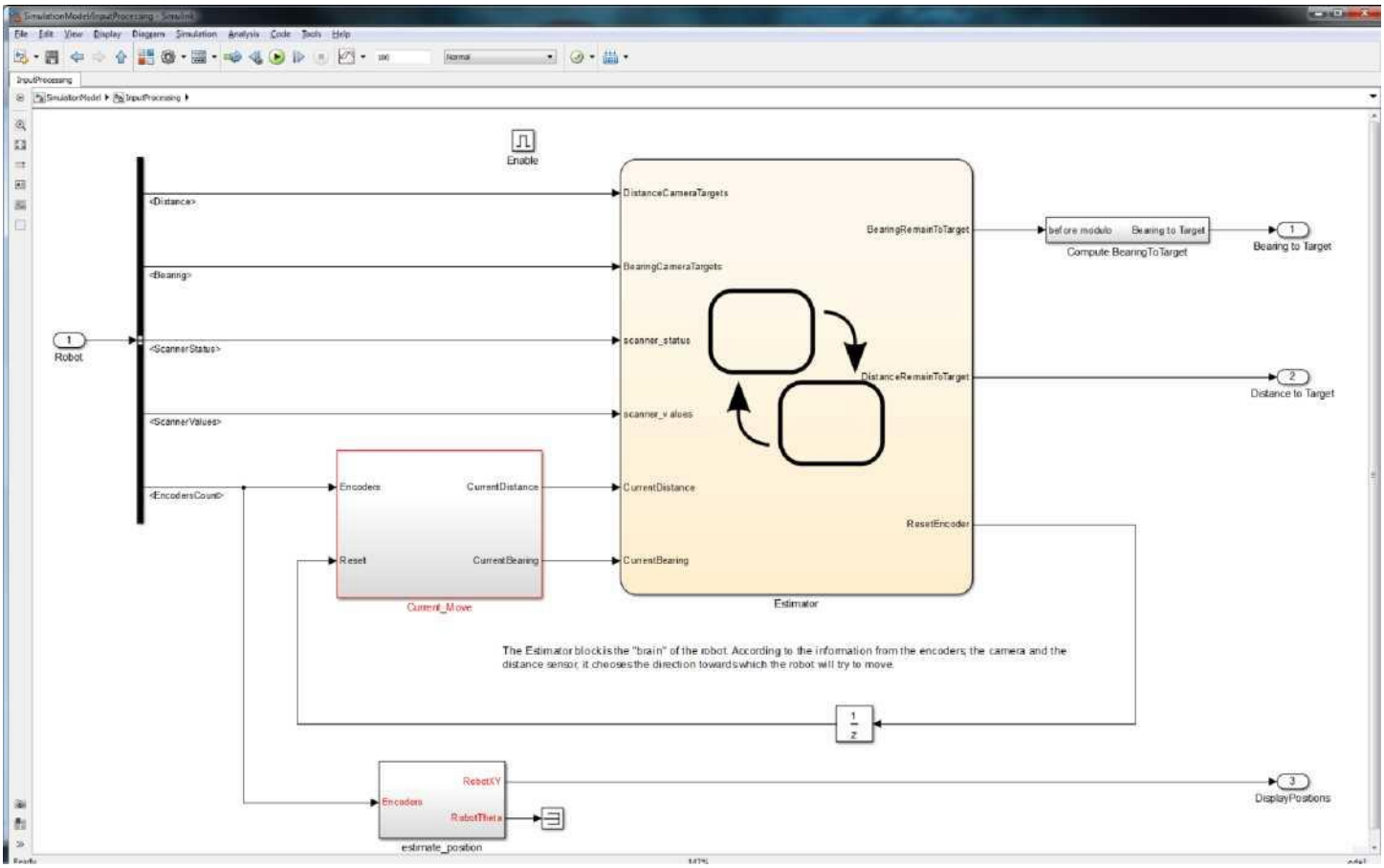
The model consists of 3 main sections:

1. **The blue "InputProcessing" unit**: This implements the robot's movement strategy. This unit determines which way and how far the robot will move, on the basis of its estimated position, the information supplied by the camera and the distances measured. This is the only unit where you can improve the model.
2. **The "Robot Model" unit**: This models the robot's physical behavior and its sensors. You must not try to modify this unit.
3. **The "Scoring" unit**: This models the electronic judge and manages the graphics simulation window. You must not try to modify this unit.

Double click on each unit to display the contents. When a sub-unit is open, you can go back to the higher level by using the "up" arrow in the toolbar (shown in red below) or in the navigation bar (in blue in the next figure).



The “InputProcessing” unit



The robot’s path logic is in the “Estimator” unit, shown as a Stateflow state diagram. The MathWorks Stateflow product is a decision logic modeling and simulation environment based on finite state machines.

Entry data for this unit are:

- Information supplied by the camera, i.e. a table containing the distances and the angles of the sites that the camera can see in relation to the front of the robot.
- Information from the encoders located on the robot’s 2 front wheels.
- Position and distance measured by the distance sensor.

Information from the encoders (left and right wheel rotation) used to estimate the robot’s current position, in the “Current Move” and “Estimate_Position” units. You must remember, however, that these are only estimates. So if the actual size of the wheels is not exactly identical to the estimated wheel size, or if the wheels slip, these estimates will be inaccurate and errors will tend to accumulate.

The logic implemented in the model provided is straightforward:

- If the camera detects a site, the robot will move towards it, navigating “by sight”.
- If the camera detects no site within its field of vision, the robot moves straight ahead.
- If the distance sensor detects an obstacle right or left when the robot is on the move, the trajectory is corrected in the opposite direction. If the distance sensor detects an obstacle straight ahead, the robot stops and turns until it finds a space free of obstacle, then sets off again.

The robot

The "Mars Rover" robot you will use in the competition will be provided by MathWorks on the day of the final, on May 25th. You are not allowed to compete with your own robot.
Each team having reached the final will test their model for the first time on the robot on the day of the competition.

It is a 3-wheel robot with 2 driving wheels and a free wheel. This means it can move in a straight line or turn round on the spot.
The Mars Rover robot is 24 cm wide, 24 cm long and 22 cm high.

The robot consists of the following parts:

- An Arduino DUE board
  - This is the board that will use your algorithms.
  - The Arduino DUE board controls the navigation and the movement strategy used to reach each site.
  - It is equipped with a daughter board to control motor power: DFRobot motor Shield.
- A Raspberry Pi board
  - Using the image flow from the connected camera, this board will identify any sites within the camera's field of vision. The sites are represented by colored markers on the ground.
  - By calculation, the board will determine the distance in centimeters of the detected sites and their angle compared to the robot (0° facing, positive angle left and negative, right).
  - This information will then be sent in table format to the Arduino board via I^2C interface.
  - This assembly will be provided and requires no modifications.
- A Webcam
  - The webcam is directly connected to the Raspberry Pi to provide vision.
  - It supplies a continuous image flow but the processing time is approximately 200-300 ms.
- Two CC motors
  - The motors are equipped with 70 mm dia wheels.
  - They are fitted with encoders with a 636 step resolution per wheel revolution.
  - The wheels are 165 mm apart.
  - The Arduino board will use this information to estimate the distance covered and the robot's angle.
  - Please note that the robot's wheels may slip and that it is not foolproof.
- A distance sensor
  - This is mounted on a servo motor used to turn the sensor. Measurements are taken in a continuous sequence: centre, right, left, centre, etc. The time lapse between 2 measurements can vary from 200 to 400 ms.
  - The sensor provides angle position information on the basis of the last measured value in relation to a table containing the 3 distance values.
  - It provides a distance measurement in centimeters from the obstacles and from the side of the arena.
  - The detection range is 10 to 80 cm.
- A 6000 mAH battery can power the robot for a whole day.

The planet Mars is represented by a 3 meter by 3 meter arena. The arena will contain:

- Green dots: they stand for locations on Mars that the robot has to pass over and where it must stop for 3 seconds to validate their exploration.
- Obstacles to avoid.

Conclusion

The model provided is a base model that you will need to improve.

Here are a few ideas on how to improve the model, in ascending order of complexity:

- Optimize the robot's speed parameters.
- Improve the site search algorithm so it can find more sites by taking into account factors such as camera time lag when the robot is moving.
- Create a new navigation algorithm using the robot's absolute estimated position to build a map of its environment.

But you are of course free to come up with all kinds of other answers. However, bear in mind that the robot's computing power is very limited and that complex calculations may take too long to perform. We therefore recommend that you don't use double precision floating-point integers or too many trigonometry calculations.