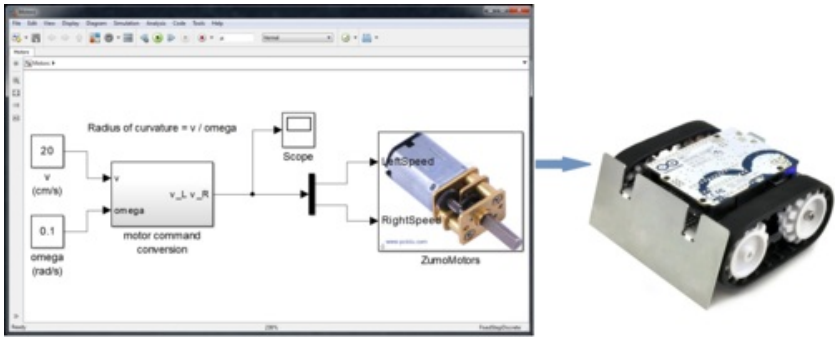
adafruit learning system

How to program a Zumo Robot with Simulink

Created by Anuja Apte



Last updated on 2015-03-13 11:15:06 AM EDT

Guide Contents

Overview

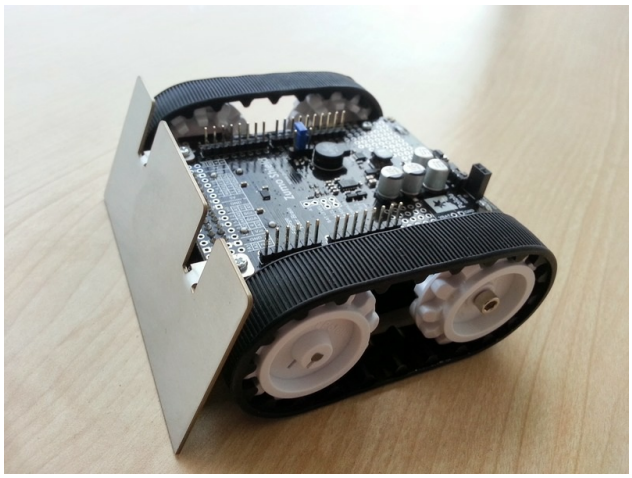
This tutorial covers how to use Simulink to program a Zumo Robot powered by an Arduino Uno. This guide will walk you through the steps to move a Zumo Robot along a specific trajectory.

Here are the specific steps to get there:

1. Install Simulink Support package (http://adafru.it/drt) for Arduino
2. Download the Zumo Robot Library (http://adafru.it/dru) for Simulink
3. Create a Simulink Model move the Zumo Robot
4. Build and download the model to see the robot in action

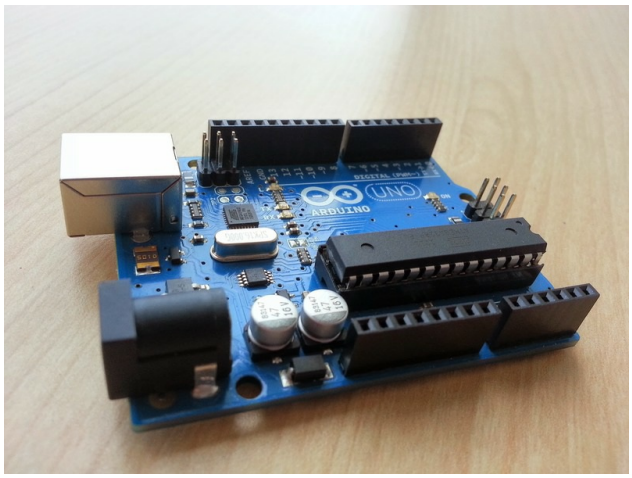
The tutorial is a second in a series on using Arduino with Simulink. For a quick introduction to Simulink, refer to Set up and Blink - Simulink with Arduino (http://adafru.it/drv) tutorial. While this tutorial uses the Zumo Robot,a similar Simulink model and the same workflow can be used to control any robot that uses a Simulink supported Arduino board

Hardware



1) Assembled Zumo Robot Kit

The assembled Zumo Robot Kit serves as a shield for the Arduino Uno R3 and also includes the chassis for the Zumo Robot. To use the Zumo, you can simply add a programmed Arduino Uno and attach 4 AA batteries and you are all set!



2) Arduino Uno R3

The Arduino Uno R3 serves as the “brains” of the Zumo Robot and is not included as a part of the assembled kit. In the tutorial, we will program the Arduino Uno using Simulink and then add it to the Zumo Robot.

3) USB Cable

The cable will be used to connect an Arduino Uno to a computer running Simulink

4) 4 AA Batteries

Software

List of Software components:

1. **MATLAB and Simulink (R2014a)**

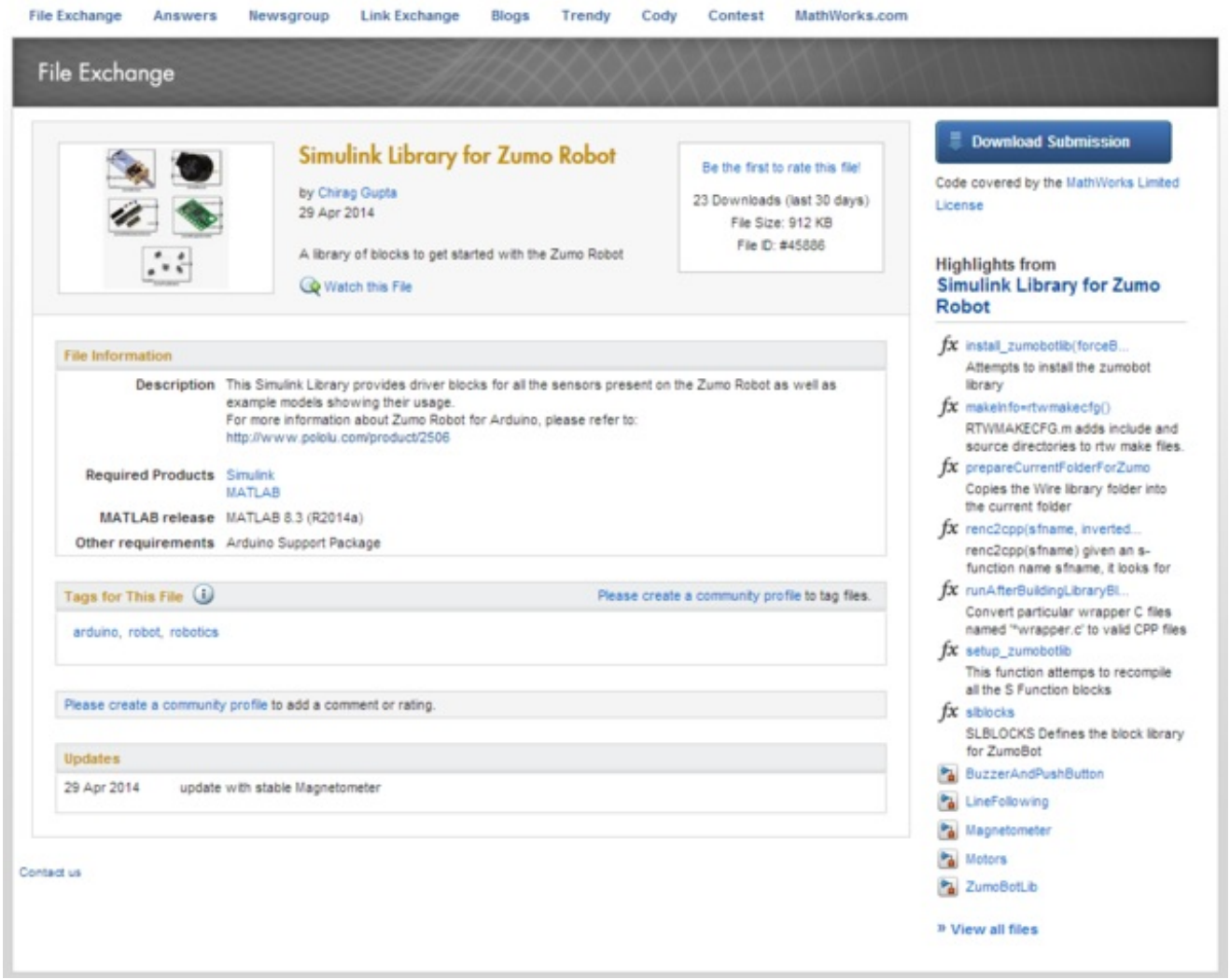
- Buy the Student version for $99 (http://adafru.it/drw)
- Buy a Home-use license for $194 (http://adafru.it/drx)
- See commercial and other licensing options (http://adafru.it/dry)

2. **Simulink Support Package for Arduino (http://adafru.it/drz)**

- Follow this tutorial to complete installation: Set up and blink - Simulink with Arduino (http://adafru.it/eI0)

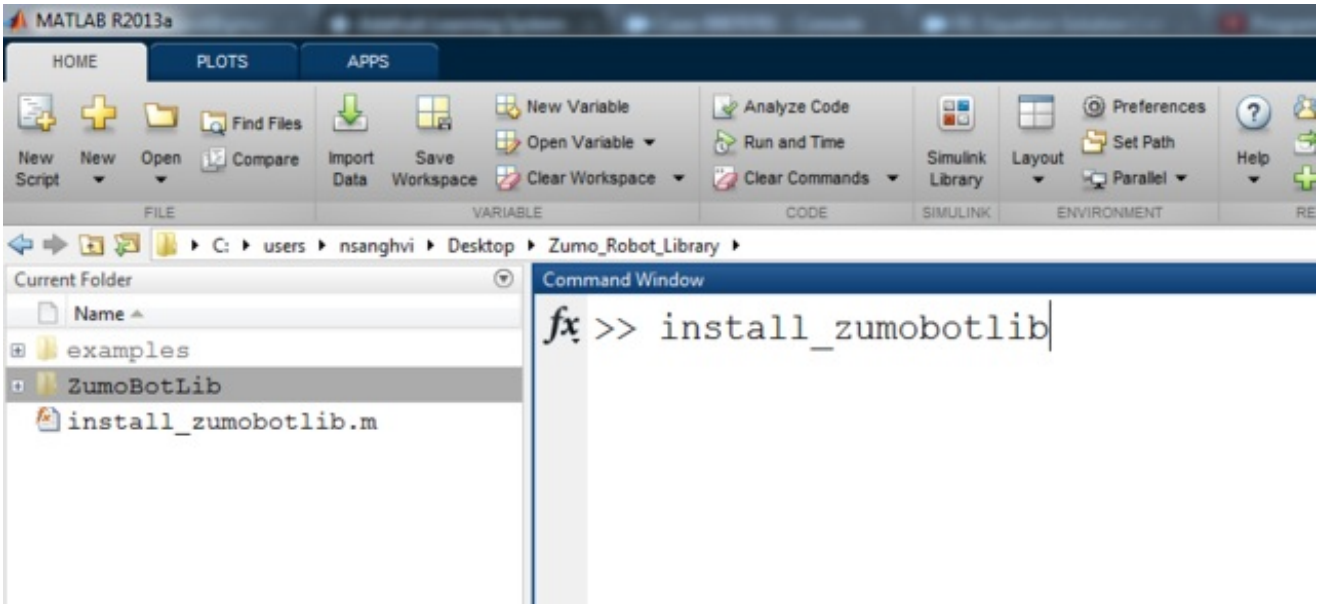
3. **ZumoBot Simulink Library (http://adafru.it/dru)**

- The Zumobot Simulink Library is a collection of blocks used to interface specifically with different components of the Zumo Robot.
- Download this library from MATLAB Central File Exchange.



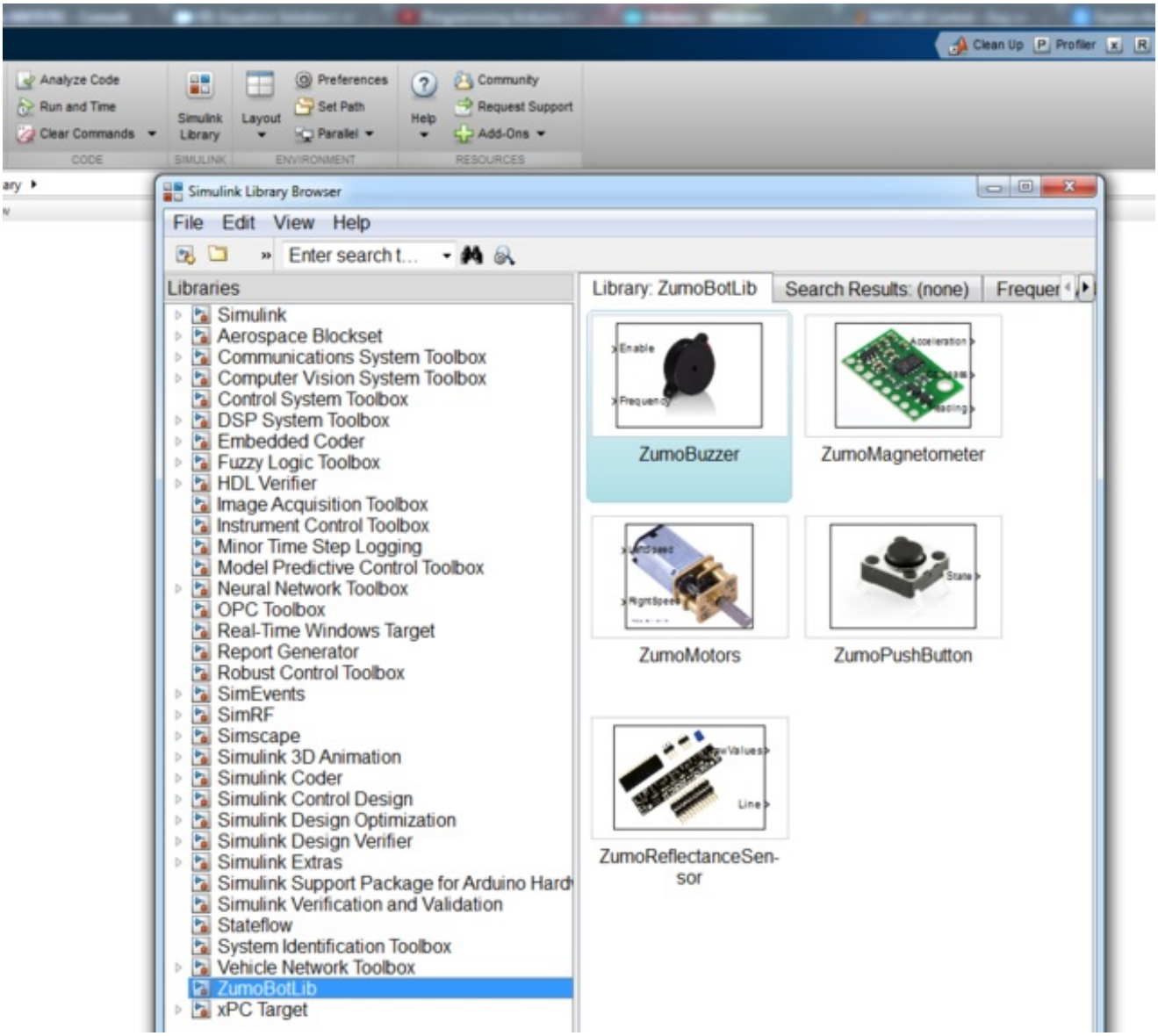
To install the library:

- Unzip the downloaded file to your working directory.
- Open MATLAB, and navigate to the directory named '**zumo_library'** in the unzipped folder.
- Install the library by typing **install_zumobotlib** in the MATLAB command window
- If you face any issues related to installation, refer to the text file '**README.txt**' for instructions related to installation.



To verify successful installation of the ZumoBot Simulink Library,

- Click on the Simulink Library Icon on the MATLAB Toolstrip to launch the Simulink Library Browser
- In the Library Browser window, check if ZumoBotLib is included in the list on the left.


Clean Up
Profiler
Analyze Code
Run and Time
Clear Commands
CODE
Simulink Library
Layout
Preferences
Set Path
Parallel
SIMULINK
ENVIRONMENT
Help
Community
Request Support
Add-Ons
RESOURCES
Simulink Library Browser
File Edit View Help
Enter search t...
Libraries
Simulink
Aerospace Blockset
Communications System Toolbox
Computer Vision System Toolbox
Control System Toolbox
DSP System Toolbox
Embedded Coder
Fuzzy Logic Toolbox
HDL Verifier
Image Acquisition Toolbox
Instrument Control Toolbox
Minor Time Step Logging
Model Predictive Control Toolbox
Neural Network Toolbox
OPC Toolbox
Real-Time Windows Target
Report Generator
Robust Control Toolbox
SimEvents
SimRF
Simscape
Simulink 3D Animation
Simulink Coder
Simulink Control Design
Simulink Design Optimization
Simulink Design Verifier
Simulink Extras
Simulink Support Package for Arduino Hard
Simulink Verification and Validation
Stateflow
System Identification Toolbox
Vehicle Network Toolbox
ZumoBotLib
xPC Target
Library: ZumoBotLib
Search Results: (none)
Frequer
Enable
Frequency
ZumoBuzzer
Acceleration
ZumoMagnetometer
LeftSpeed
RightSpeed
ZumoMotors
State
ZumoPushButton
Line
ZumoReflectanceSen-sor

Simulink Model

Now that we have the software environment set up, the next step is to create a model for the system.

The curved trajectory along which the Zumo Robot moves can be defined by the angle 'omega' at which the Zumo Robot is turning at an instant and the velocity 'v' at which the robot is moving forward.

Using simple kinematic equations, the mathematical relation is given by the following equations:

$$V_L = V - \omega L/2$$
$$V_R = V + \omega L/2$$

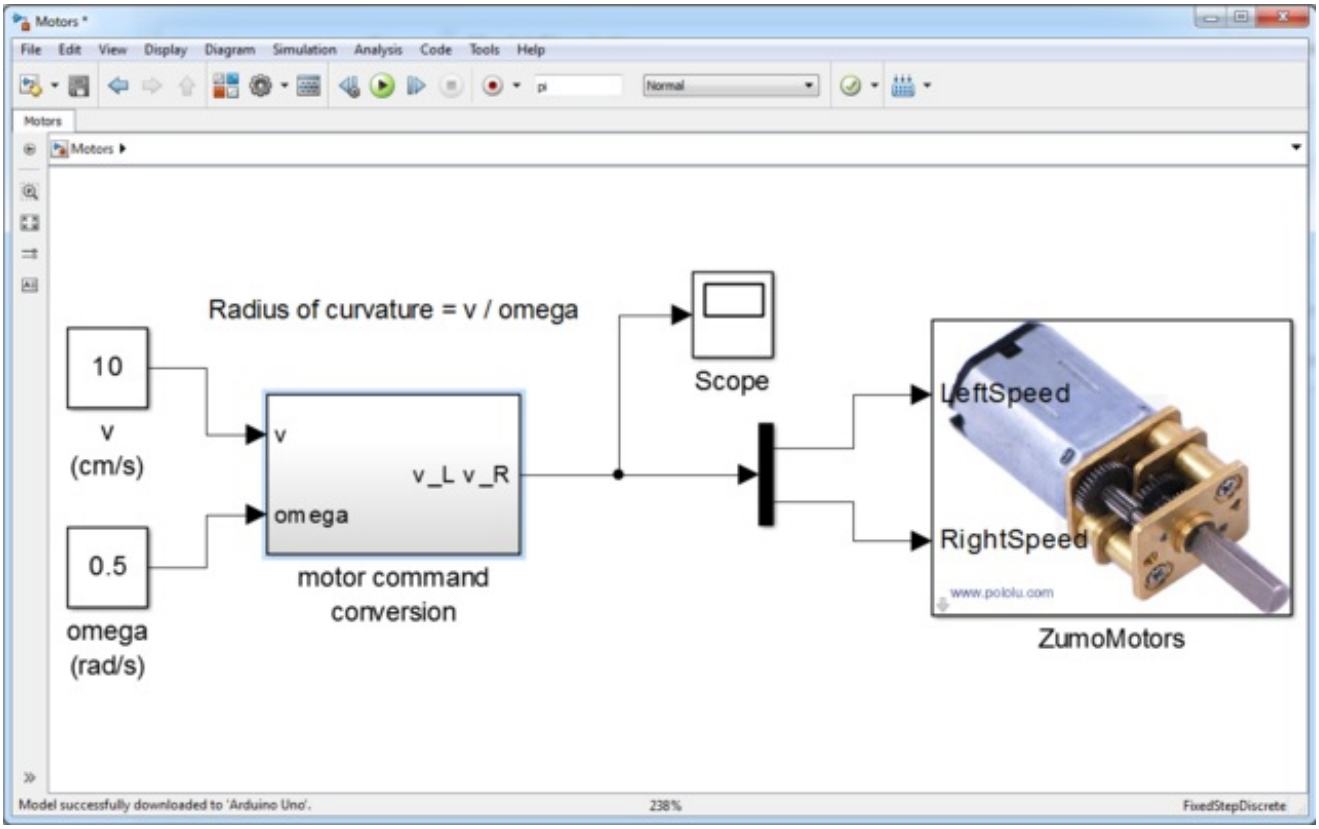
where,

V is the forward velocity
V_R is the right wheel velocity
V_L is the left wheel velocity
L is the length of the axel
ω is the angular velocity

Or in matrix notation,

$$\begin{bmatrix} V_L \\ V_R \end{bmatrix} = \begin{bmatrix} 1 & -L/2 \\ 1 & L/2 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix}$$

The ZumoBot simulation library includes an example Simulink model for this tutorial. To open the example model navigate to the '**examples**' folder, and type **Motors** in the MATLAB command window.



The model accepts velocity '**v**' and turning angle '**omega**' as inputs and using the subsystem named '**motor command conversion**' converts the two inputs into left and right wheel velocities.

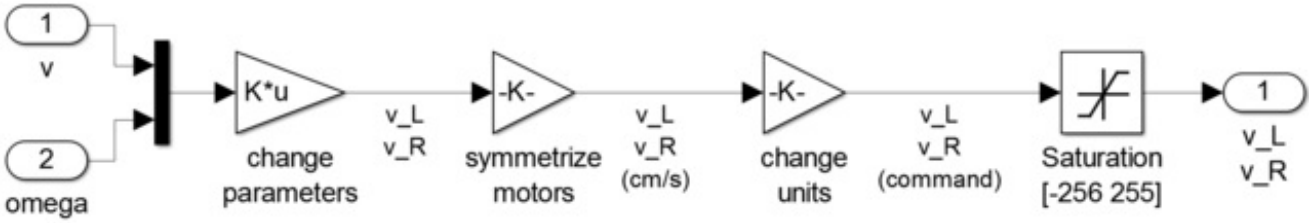
A subsystem is a collection of Simulink blocks and is a neat way to organize blocks for complex models. It is the block diagram equivalent of writing functions for your code.

The '**model command conversion**' subsystem implements the math we just saw above in Simulink. To look at the implementation, double click on the subsystem block.

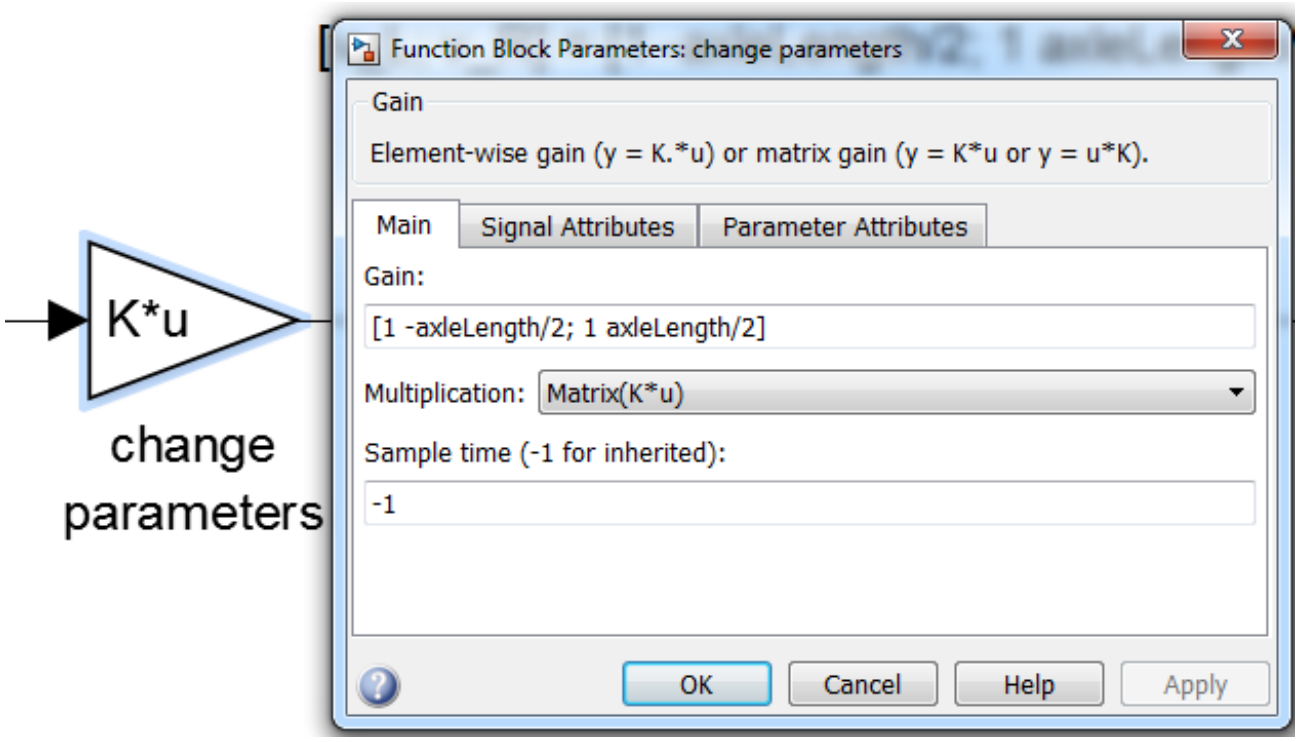
$$v_L = v - \text{omega} * \text{axleLength}/2$$
$$v_R = v + \text{omega} * \text{axleLength}/2$$

or in matrix notation,

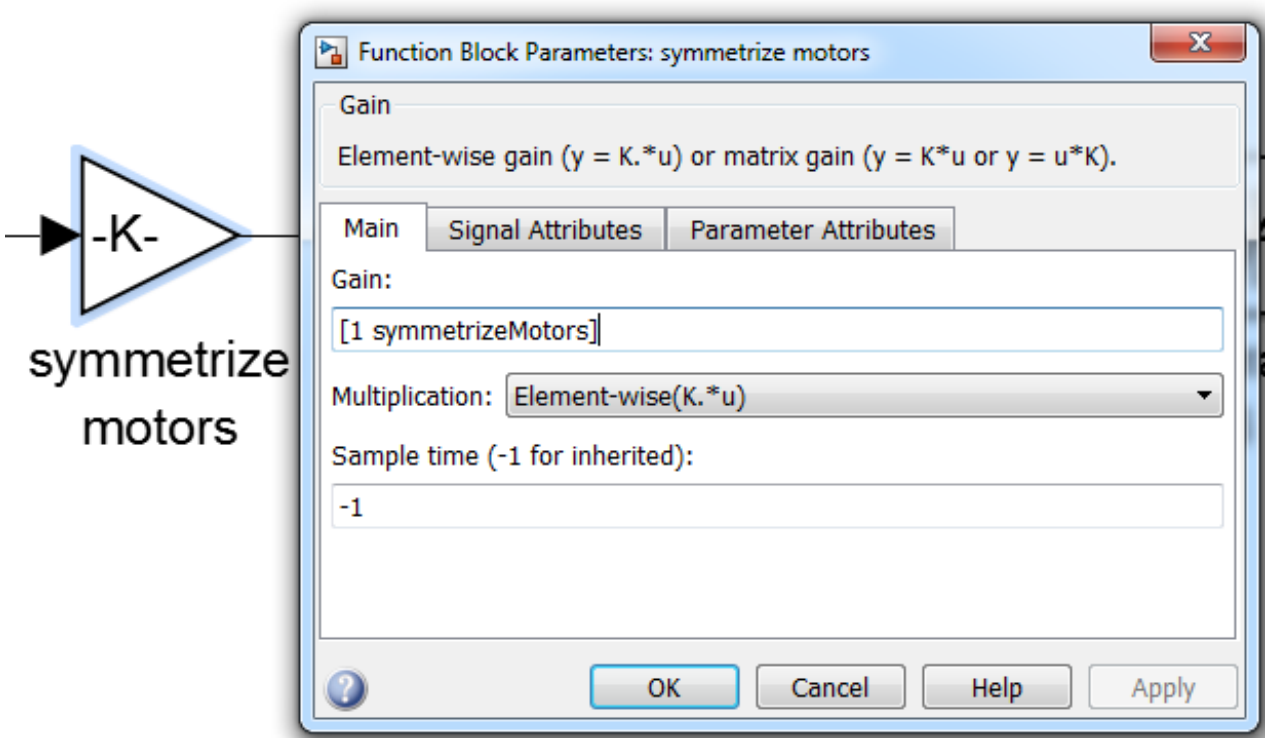
$$[v_L; v_R] = [1\ -\text{axleLength}/2;\ 1\ \text{axleLength}/2] * [v; \text{omega}]$$



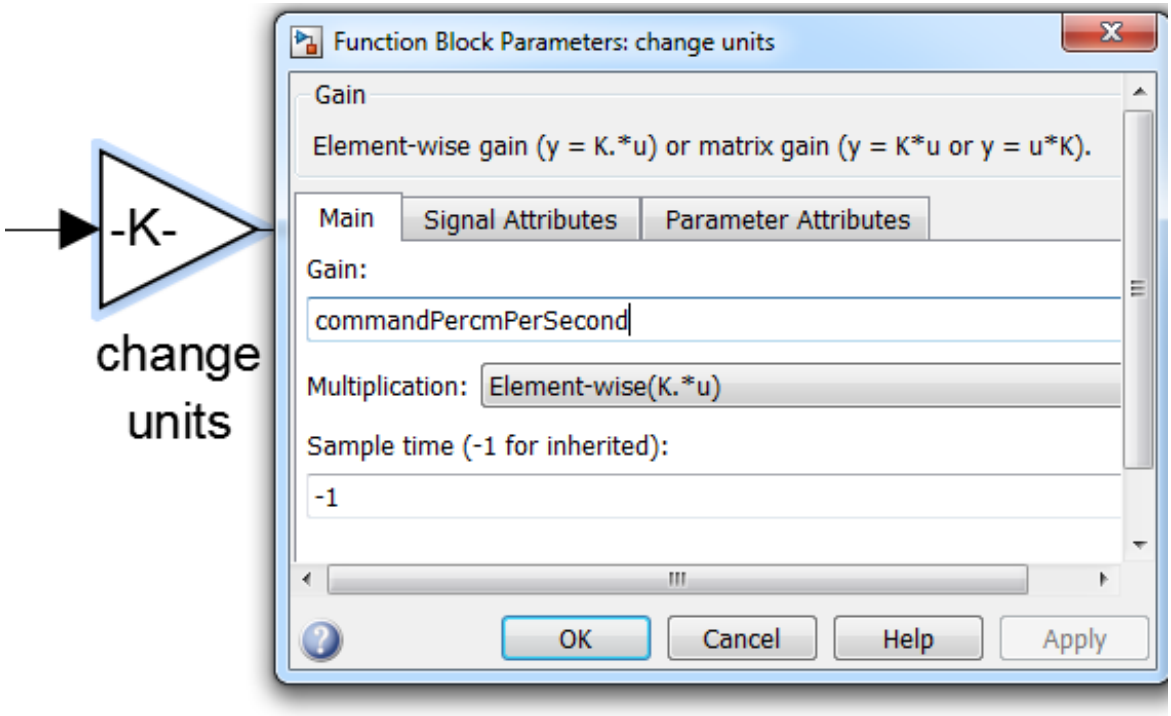
The first block named '**change parameters**' performs the matrix multiplication operation we saw in the equations above. To look at the block properties, double click on the block.



The second block named '**symmetrize motors**' accounts for the fact that not all motors are identical. This block multiplies the one of the wheel velocities with a scaling factor called '**symmetrizeMotors**' which can be tuned depending on the differences between the motors.

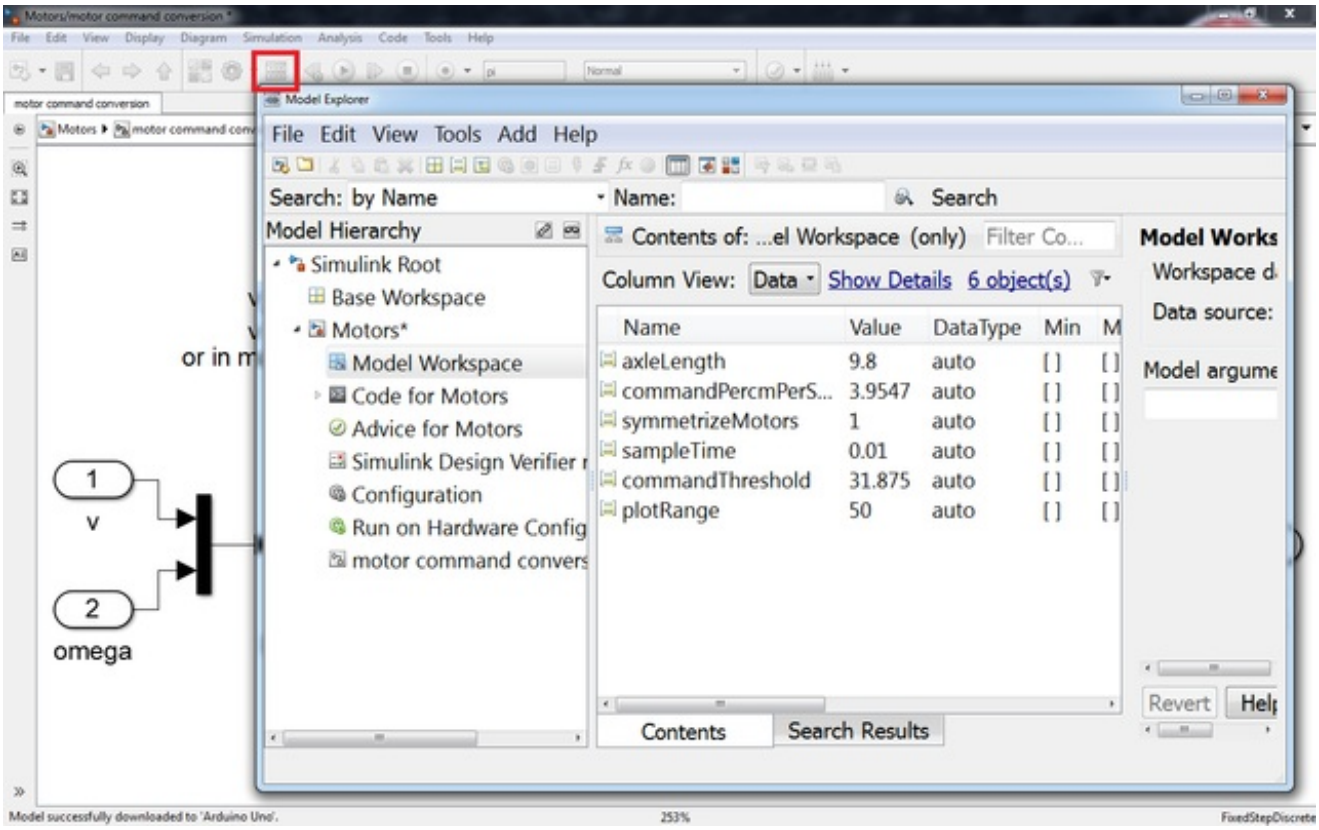


Up till this point, we were dealing with real world units, however the Zumo Robot understands commands in integer values ranging from -256 to 255 for motor control. The third block named '**change units**' performs this conversion.

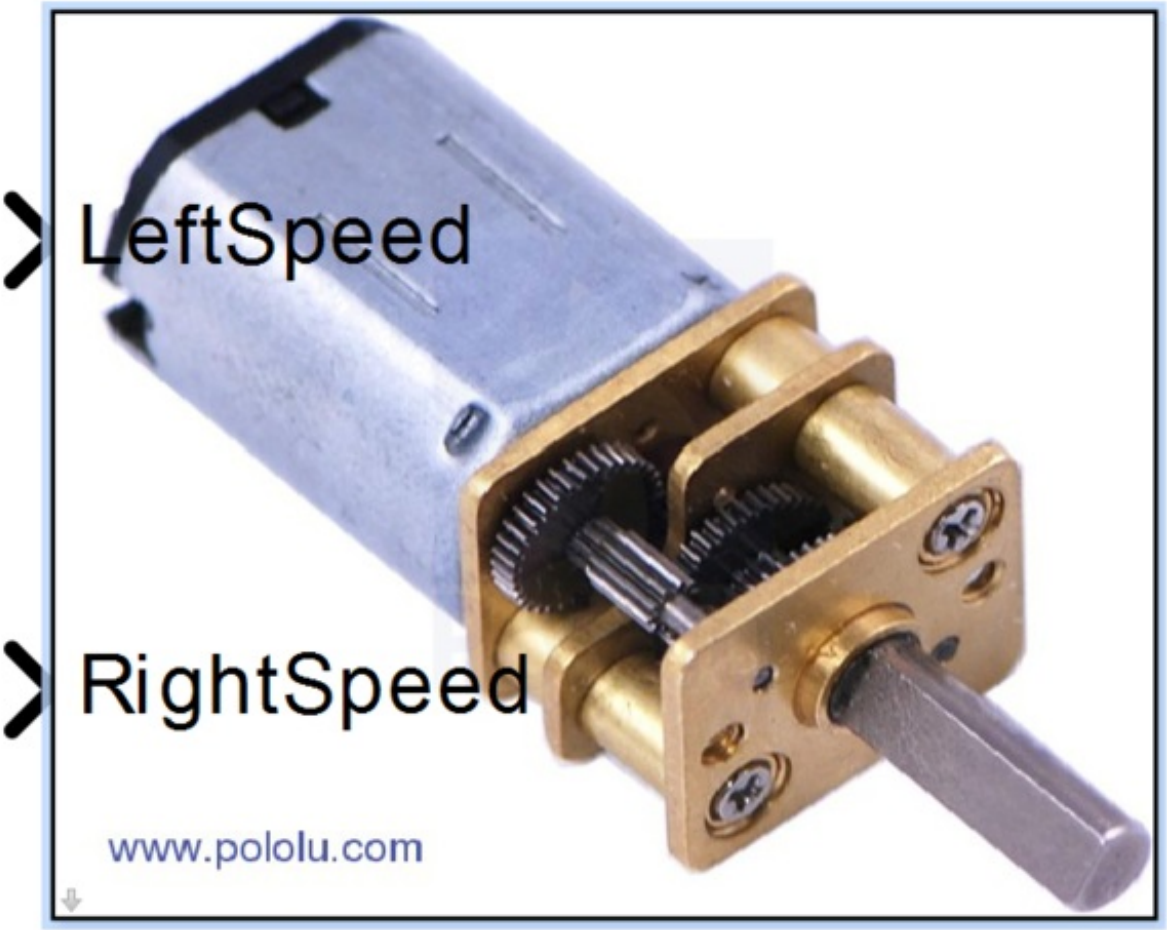


The parameters **axleLength, symmetrizeMotors,** and **commandPercmPerSecond** are all model

parameters and can differ slightly from Zumo Robot to Zumo Robot. Thus even though the model currently has some default values, these can be tuned to fit your specific needs. These parameters can be modified through the model workspace which can be accessed through the button highlighted below.



The outputs of the motor command conversion subsystem which are the left and right wheel velocities are fed to the '**ZumoMotors**' block. The '**ZumoMotors**' block represents the motors on the assembled Zumo Robot.



ZumoMotors

Generate code, load and run

Now that we have understood the model, let us download it to the robot and see it in action.

Open the Motors model from the examples directory that came with the ZumoBot library. To start off, use the default values in the model for velocity 'v' (20 cm/s) and angle of turn 'omega' (0.1 rad/s).

To build and download the model on the ZumoBot, click on the Deploy on hardware button in the right top corner of the model.

Read the messages at the bottom status bar of the model to confirm that the model is successfully downloaded to the target. See the robot moving in the input trajectory.

In this case, the radius of curvature is quite large, and for short distances, the Zumo Robot almost appears to move in a straight line.

Now let us modify the input velocity 'v' to 10 cm/s and 'omega' to 1.5 rad/s. Note that you can change the value of 'v' and 'omega' by double clicking on the respective blocks and modifying the constant value parameter. Generate and download the code for the modified model into the Arduino once again.

In this situation the radius of curvature is comparatively smaller, and the Zumo Robot appears to constantly make a sharp turn.