*Suggested Progression of Lesson*

|  |  |
| --- | --- |
| **Step** | **Content** |
| 1 | [Introduction](#_Introduction) |
| 2 | [MATLAB Function Blocks](#_MATLAB_Function_Blocks) |
| 3 | [MATLAB Functions and LED Light](#_MATLAB_Functions_and) |
| 4 | [Optional: Button with Lights “Switch Style”](#_Optional:_Button_with) |
| 5 | [Study 2](#_Study_2) |
| 6 | [Simulink with Motors](#_Simulink_with_Motors) |

**Introduction**

Video Reference: [Unit2a\_SimulinkBasicMath.mov](https://www.youtube.com/watch?v=uLheskCv3Ds)

The MATLAB command window allows us to experiment with input and output pins on the miniQ robot. However, through the MATLAB command window we cannot deploy our code to hardware and this is important if we want our robot to function without being connected to a computer. Therefore, we must use another standalone MathWorks product, Simulink.

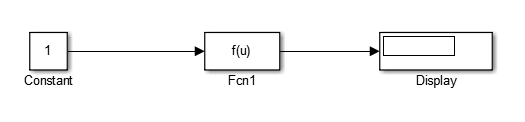
Simulink provides a graphical programming interface and is an industry standard in the fields of engineering and the sciences. Oftentime it is easier to represent code in Simulink than in the form of MATLAB scripts. This is because images resonate more with people as a communicative tool which is consistent across cultures. In this unit we will explore combining the skills we have learned with regards to formatting MATLAB code but we will connect this knowledge to the graphical programming environment, Simulink.

Sequence:

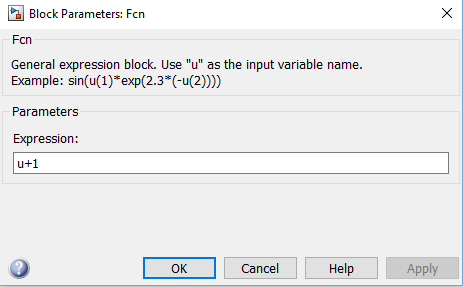
1. Open Simulink to create the model in the picture below
2. Open the Simulink Library Browser
3. The "Constant" block is in the 'Commonly Used blocks' tab
4. The “Display” block is in the Sinks tab and
5. The "Function" block in the “User Defined Functions” tab

“Addition Machine**”**

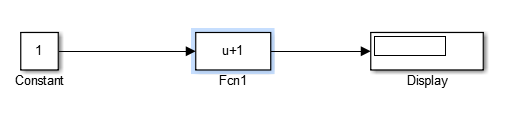
Example Model: AdditionMachine.slx



1. Double clicking into the function block (“Fcn1” in the picture above) and you should rewrite the mathematical equation so it is the following:



1. After you write in the function, which can be expressed in one line, the completed Simulink model should be the following:



1. Press the green “play” button in MATLAB toolbar and notice the value that the display block now reads.

**MATLAB Function Blocks**

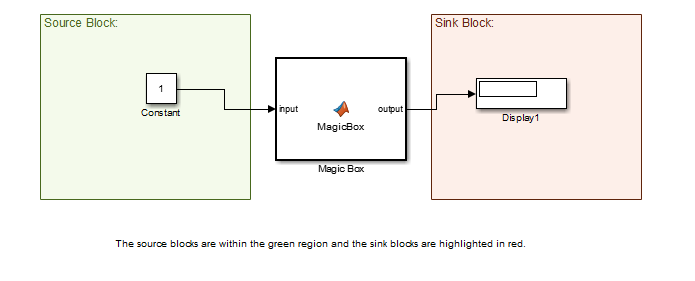
*For reference, see the**“Instructor Reference: Lesson 1 Unit 2”**document in the Unit 2 Instructor Reference folder*

The addition function of the “Addition Machine” can also be represented in Simulink through a “MATLAB function block” which can be found in the “User Defined Function” tab. This block is necessary when you need more than one line to express things by using control structures such as “if”, “elseif”, “while” and “for” to name a few. Create the following model below:

Video Reference: [Unit2b\_SimulinkMagicBox.mov](https://www.youtube.com/watch?v=czuAarO-4cE)

“Magic Box”

Example Model: MagicBox1.slx



To build the model:

1. Begin by finding the constant blocks in the “Commonly Used Blocks” tab in the Simulink blockbuilder. Drag the following blocks into your Simulink platform to begin developing your first model.
2. Specifically, drag in one constant block, one MATLAB function block, and one “Display” block.
3. Double click into the constant and change the number value it holds. For this example though, set the constant block to the value, 1.
4. Double click theMATLAB function block found in the “User Defined Function” taband drag this into the model and change the name at the bottom of the box to “Magic Box” as in the picture above.
5. Double click into the “Magic Box” and write the script:

function output = MagicBox(input)

number = input;

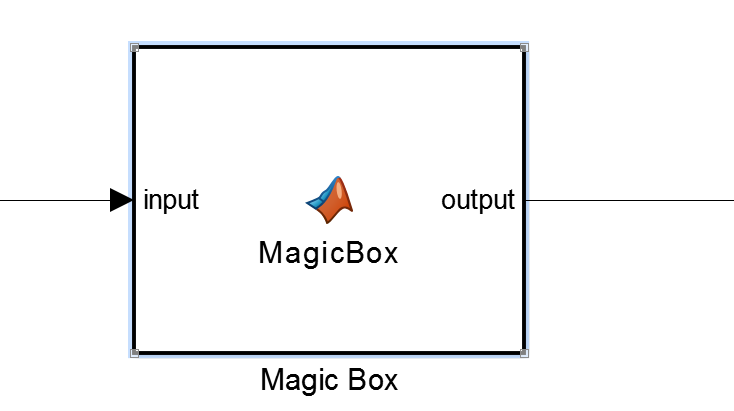
output = number+1;

end

1. Go back to the Simulink block selector and in the “Sinks” tab drag in a “Display” block. Press the green “play” symbol at the top of the Simulink workspace window to see what number your Simulink “Display” block changes to.

*Questions to consider:*

1. Is this what you expected to happen in the model?
2. Can you create a few magic boxes to model other mathematical concepts? For example, the equation for the hypotenuse of a triangle or the area of a square.



Inside *this* “Magic Box”, a MATLAB function, is the following code. *When writing your scripts don’t forget to end statements with a semicolon:*

function output = MagicBox(input)

number = input;

output = number+1;

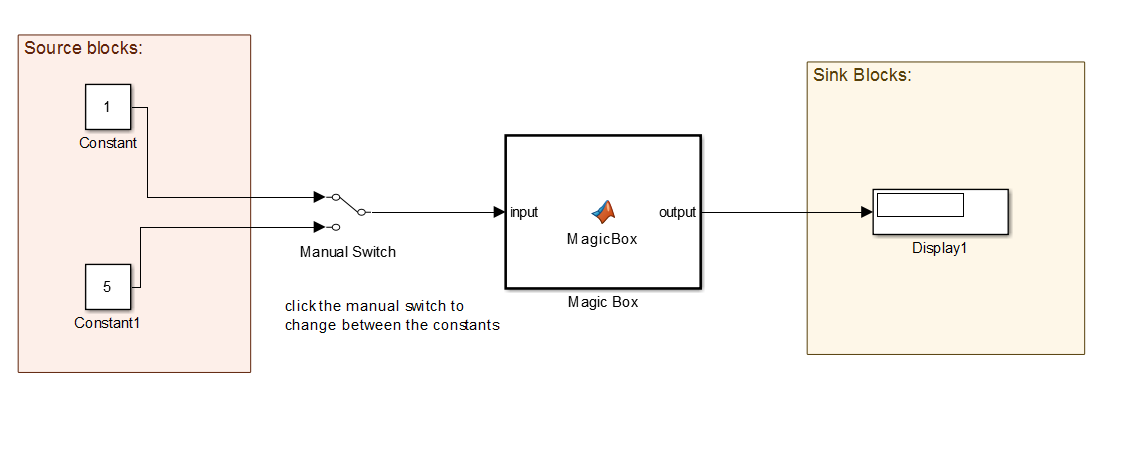
end

The code above takes a variable, adds +1, and then outputs a returned value.

The model below, with a switch block, resembles the MATLAB with motors unit exercise. By clicking on the switch block two different options for constant blocks are displayed. Create the model below and click the run tab which is a green colored play button at the top of the toolbar.

1. Set the time frame for the simulation to “inf” (which stands for “infinite”) to run the simulation continually without it stopping. Once you have the model running, click the switch block to see what values the change to in your model:

Example Model: MagicBox2.slx

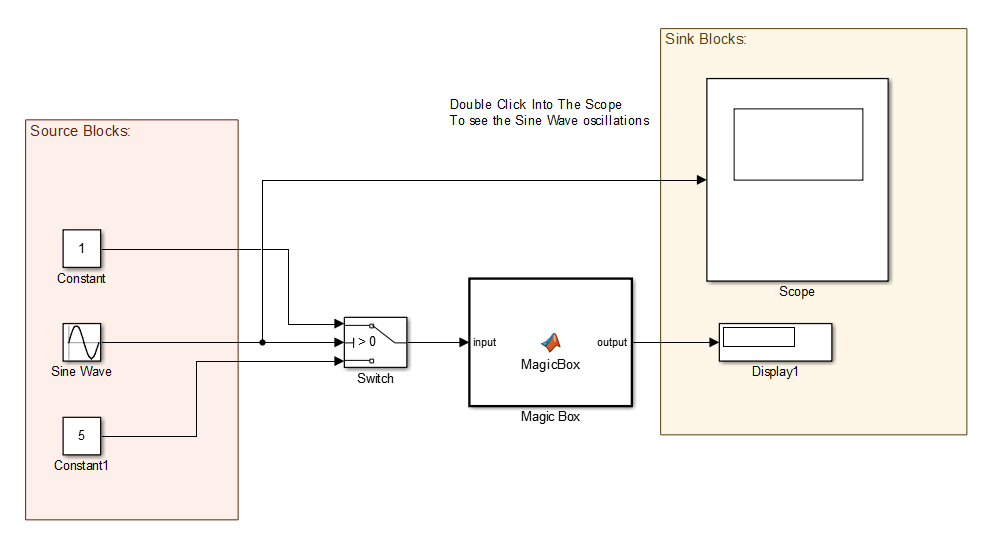


We can adapt the model above specifically, instead of manual control by way of mouse click, on the manual switch block, we can have an oscillating sine wave control the switch between constant values.

Attach a sine wave Simulink block and connect it to the middle of the switch block. Two constant values, 1 and 5, are branched to the switch block. Observe the values which occur in the display block and why these are the particular numbers displayed relative to the “Magic Box.” Click into the scope to further investigate.

Also, consider clicking into the switch to investigate or the “scope” box which will give you a visual of the actual sine wave itself.

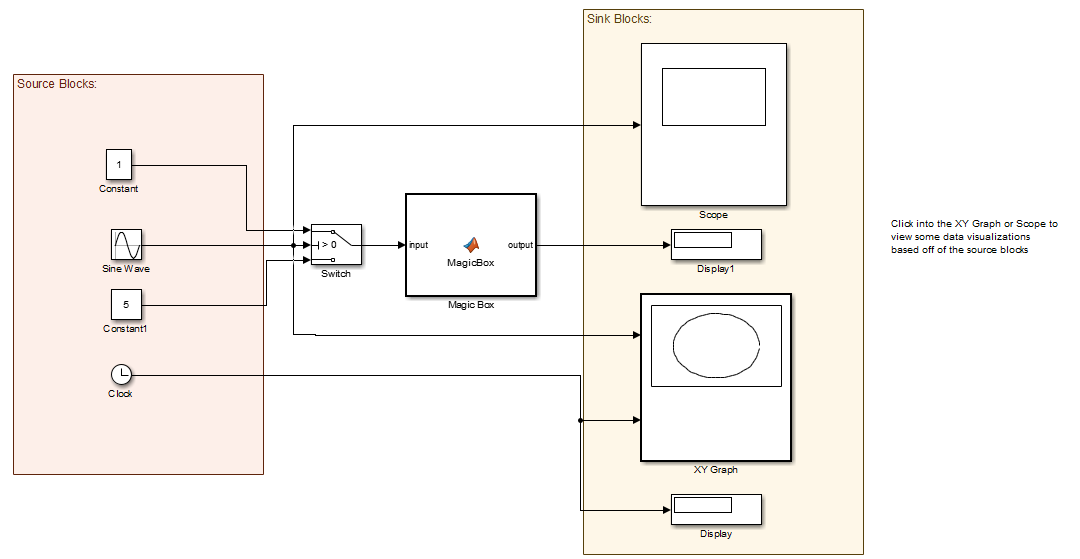
Example Model: MagicBox3.slx



*Intermediate:*

Below we have created a model with the addition of an XY Graph found in the Simulink blocks toolbar, under the “Sinks tab.” A clock block is attached to the lower XY Graph tab while the sine wave signal is connected to the top connector of the XY block.

Example Model: MagicBox4.slx



With respect to programming a MiniQ robot a Simulink library exists where the functions needed to control our hardware is bundled into Simulink block form. This simplifies our efforts by not having to program these blocks for ourselves as in the MATLAB scripts were composed during Unit 1.

**MATLAB Functions and LED Light**

*For reference, see “Instructor Reference: Lesson 2 Unit 2” in the Unit 2 Instructor Reference folder.*

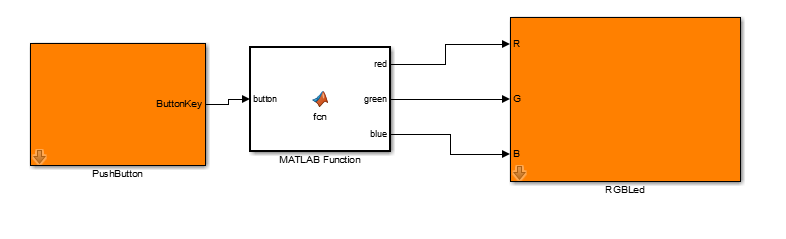
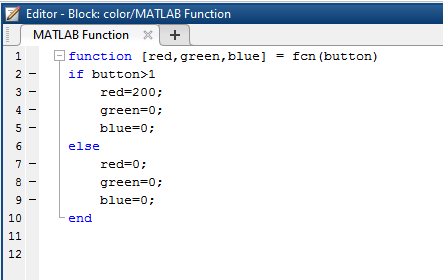
Video Reference: [Unit2c\_SimulinkFunctionsLights.mov](https://www.youtube.com/watch?v=UVssb7datAo)

Open up the MiniQ library in the Simulink Block Builder and open the following model:

“Button With Lights Function Style”

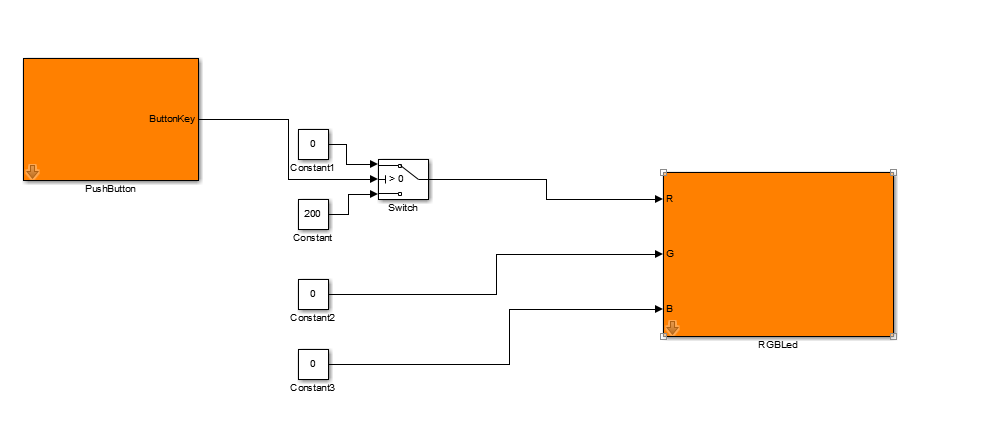
Example Model: ButtonWithLightsFunctionStyle.slx

*Constructing a MATLAB Function block for Simulink:*

1. Install the MiniQ library and drag in a push button block.
2. Also drag in an RGB led block and set them apart from each other.
3. In the middle you’ll need to drag in a MATLAB function block: 
4. Double click into the MATLAB function block and enter the code below as a script. 
5. Our objective is to get a red light to turn on when the button is pressed otherwise, there should be no light shown on the MiniQ:

**Optional: Button with Lights “Switch Style”**

*For reference, see the “Instructor Reference: Lesson 3 Unit 2” document in the Unit 2 Instructor Reference folder*

Example Model: ButtonWithLightsSwitchStyle.slx

Here we have another example of a red light being light up depending on a button press. This time however, we are not relying on the construction of a function to achieve the same purpose.

If you click into the PushButton block you’ll read outputs as an integer relative to what key, and there are three, is pressed. Click into the switch block and notice how the top and bottom values shift relative to the incoming value being more or the same as 0. Each constant sends a different value for “R” to the Red,Green Blue (RGB) LED.

Basically, the color red happens or it doesn’t depending upon a button press or not. In this case, a button press turns the light off. Try mixing up some of the constants in external mode and see what results with the lights.

**Study 2**

*Note: Below is a copy of Study 2. For print-outs, the original document can be found in the Unit 2 folder*

Standards Covered: (a), (b), (c)**,** (d)**,** (e), (f)

1. Split into 4 groups and evaluate and discuss the following URLs as a group. Do this after reading the URL individually *(15 min)*

History of Morse Code: <http://www.history.com/topics/inventions/telegraph>

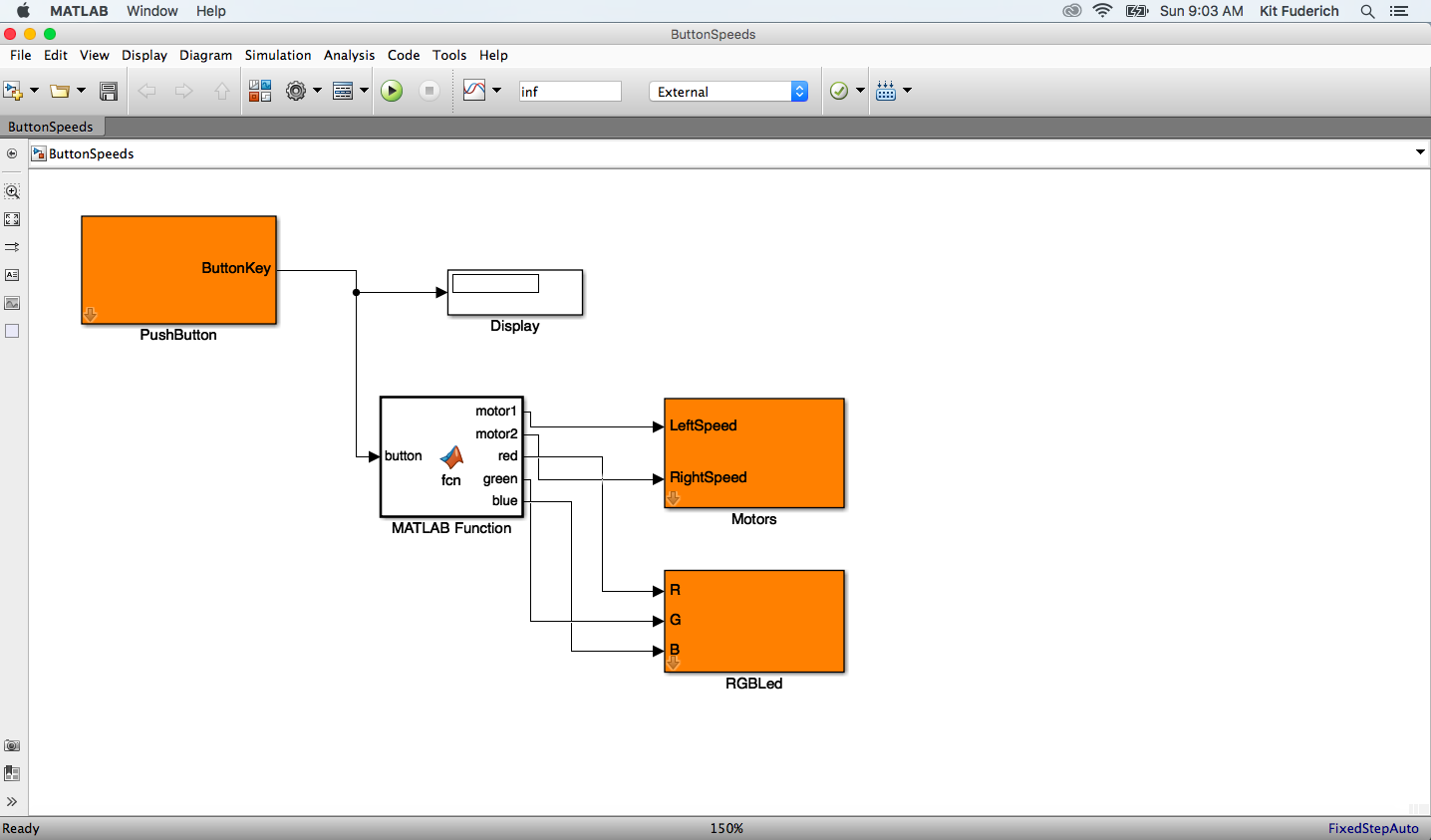
1. Share what you have learned with the rest of the class by electing a leader to summarize your research. Prompt: Include one interesting fact and question you have after reading the article and discussing things among your groups *(10 min)*
2. Load the Simulink file: ButtonWithLightsFunctionStyle.slx in your miniQ robots if you don’t already have it there.
3. Watch:
4. Video of SOS with a flashlight: <https://www.youtube.com/watch?v=CETGq7sqytE>
5. Other Letters in Morse Code: <http://rsgb.org/main/files/2012/10/Morse_Code_Sheet_01.pdf>
6. Activity *(30 min)*
   1. Individually and in private, think of a word you’d like to share with a partner. You will learn how to communicate this word in Morse code.
   2. Practice communicating this code with your miniQ robot by button press. Do so in private.
   3. Circulate around the room and share your Morse Code pattern with others. See if they can decode the results. Note: use the Morse Code handout to help with the decoding process.
7. Group Discussion *(30 min)*
   1. How does Morse code relate to the code we write for a computer?
   2. What are the differences between how we communicate information to machines vs. people.
   3. What are some of the ways in which people communicate with each other daily? Between cultures what is a potential barrier to communication?
   4. Is there a form of communication which transcends language? Idea: think of a paintbrush.

**Simulink with Motors**

*For reference, see the “Instructor Reference: Lesson 3 Unit 2” document in the Unit 2 Instructor Reference folder*

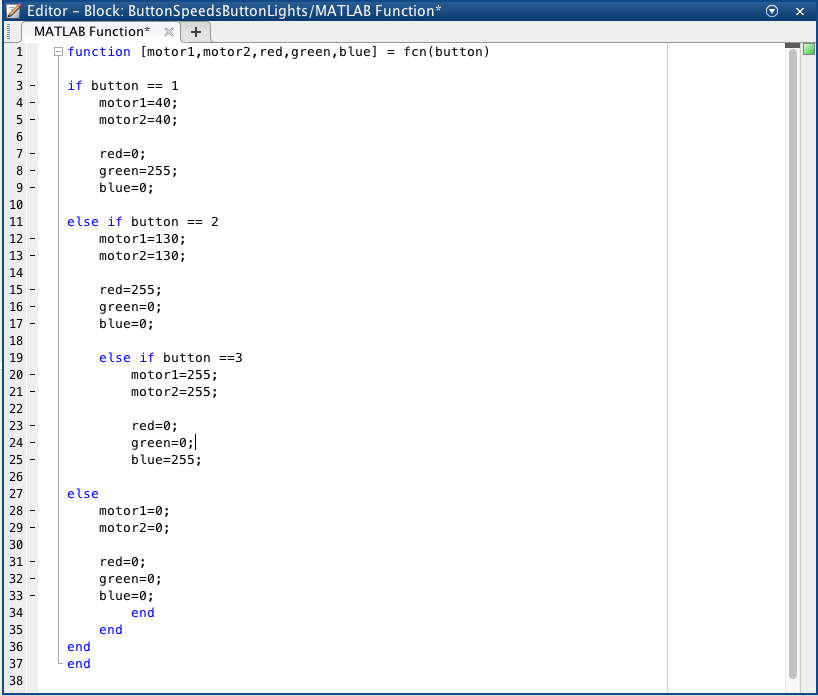
Open up the following file: Example Model: ButtonSpeedsButtonLights.slx

Video Reference: [Unit2d\_SimulinkWithMotors.mov](https://www.youtube.com/watch?v=Os2-6kis2ZA)



In the example above we have a MATLAB function which controls the colors of the LED lights and speed of the motors. Both the colors and speed outputs vary depending upon which of the three buttons on the miniQ robot are pressed. The display block is there so students can first interpret how each of the buttons on the miniQ outputs its value numerically.

Double click into the MATLAB function and this is what you will see:



Notice how there are variables for the motors (motor1 and motor2) as well as variables for each of the output colors of the RGB led (red,green,blue). Study the construction of the function especially the branching from “if”, “else if” and “else” control structures.

*Activity:*

1. Experiment with changing variables to get different colors and different speeds to occur for each button.

Creative Commons License

This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](http://creativecommons.org/licenses/by-sa/4.0/).