|  |  |
| --- | --- |
| **Step** | **Content** |
| 1 | [Study 1](#_Study_1) |
| 2 | [Introduction](#_Introduction) |
| 3 | [Interfacing with the MiniQ](#_Interfacing_with_the) |
| 4 | [Writing Functions](#_Writing_Functions) |
| 5 | [Intermediate Programming Activity](#_Intermediate_Programming_Activity) |
| 6 | [Taking it further](#_Taking_It_Further) |

*Suggested Progression of Lesson*

**Study 1**

*Note: Below is a copy of Study 1. For print-outs, the original document can be found in the Unit 1 folder*

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

Sequence:

1. Icebreaker, get to know everyone’s name and one of their interests, academic or extracurricular. See if anyone can remember everyone’s name and interest. *(10 min)*
2. Evaluate the following URLs as a group. Elect one person on the group to summarize what the URL was about. Use the questions as prompts to help you present your findings to the class. You may have to do additional research to answer some questions *(20 min, presentations 10 min)*
3. Group Discussion:
   1. What are some of the essential skills you need to have if you want to be a robotics engineer?
   2. What are some useful “tools” for developing robots? *(10 min)*
4. Video: What is it like to be a robotics engineer through the Darpa Robotics Challenge. Notice how they test the robot and how it is “tethered” to be controlled: *(10 min)* <https://www.youtube.com/watch?v=Gk_zADMORzo>
   1. Extra (If Interested): Automatons as the beginning of the robot renaissance: <https://www.youtube.com/watch?v=MZMeQI1V1Ow>

Group 1:

<http://www.princetonreview.com/careers/139/robotics-engineer>

<http://www.princetonreview.com/careers/139/robotics-engineer>

<http://www.princetonreview.com/careers/139/robotics-engineer>

1. What does it mean to be a robotics engineer?
2. What kinds of applications do we see robots in? And in what capacity?

-Agriculture

-Mining

-Nuclear power plant maintenance, and a variety of other fields

Group 2:

<http://study.com/articles/Robotics_Engineers_Information_About_Starting_a_Career_in_Robotic_Science_and_Robotic_Engineering.html>

“Robotics engineers should be highly creative, self-motivated individuals with an ability to think outside the box. Their advanced mathematics, applied physical science, and computer science skills should be solid, and they should enjoy collaborating with a team and be adept at communicating with others. Because the programming of new robots is considered by some to be the most challenging aspect of robotics engineering, particular expertise or a degree in software engineering is highly desirable and may expand a robotics engineer career options considerably.”

1. What does it mean to think outside of the box?
2. What does collaboration and “adept” communication mean?
3. What is programming? And do you know any programming languages? What are these languages used for?

Group 3:

<http://www.wired.com/2014/10/robotic-followers/>

[http://www.learnaboutrobots.com/roboticsEngineer.htmhttp://www.learnaboutrobots.com/roboticsEngineer.htm](http://www.learnaboutrobots.com/roboticsEngineer.htm)

“So you want to be a robotics engineer? Software engineering is probably the Achilles heel of robotics. The mechanical, electrical and computer engineers have built awesome machines, but they still are extremely difficult to put into production. This is because they are so difficult to teach. An expert technician `has to program the robot's every motion down to the tiniest minutia. In my opinion, the biggest contributions yet to be made in robotics will come from the software engineers. Companies are hiring robotics engineers to develop everything from automated vacuum cleaners to robot dogs.”

Group 4:

Robotics Companies to look at (evaluate):

[http://www.fanucamerica.com/http://www.fanucamerica.com/](http://www.fanucamerica.com/)

[http://www.adept.com/http://www.adept.com/](http://www.adept.com/)

[http://www.sarcos.com/http://www.sarcos.com/](http://www.sarcos.com/)

[http://www.kuka-robotics.com/usa/en/http://www.kuka-robotics.com/usa/en/](http://www.kuka-robotics.com/usa/en/)

<http://www.rethinkrobotics.com/>

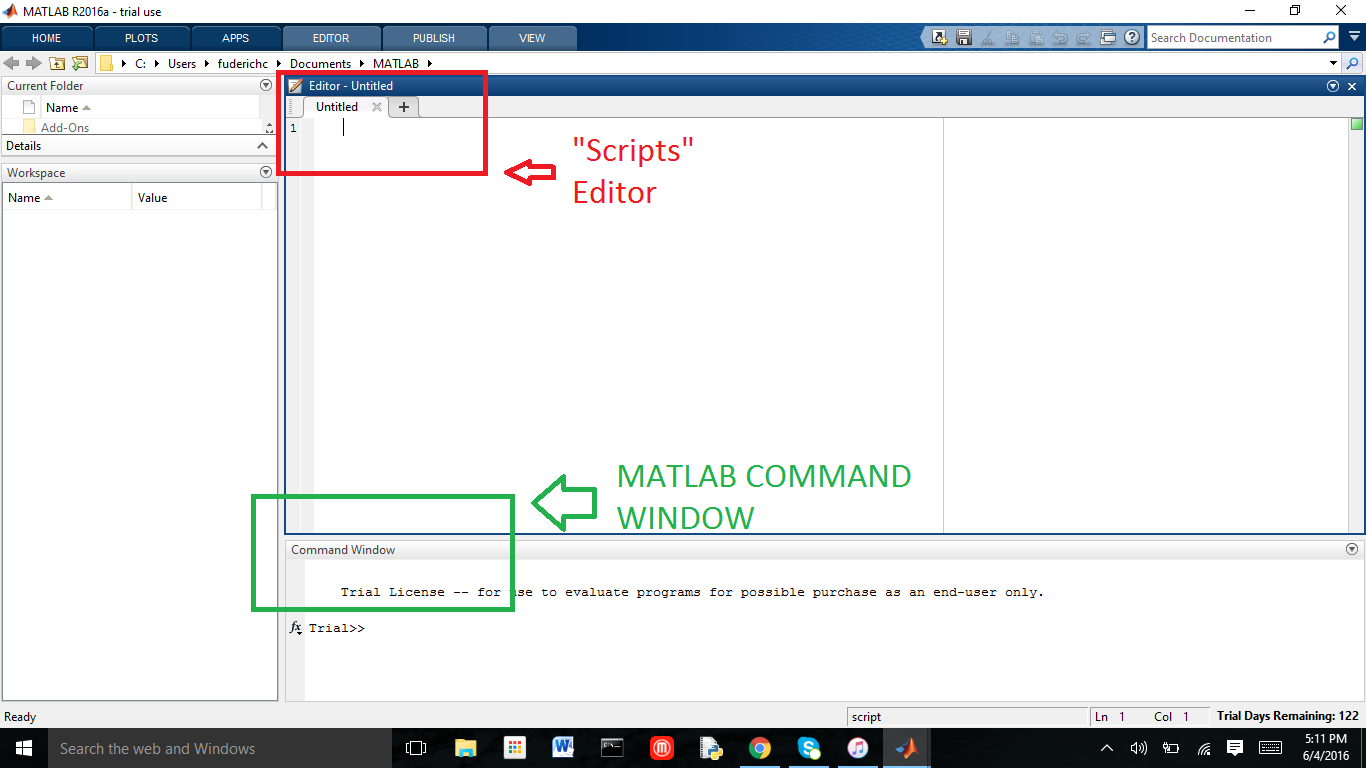
1. What are some interesting “products” (robots) are produced by this company?
2. Do they have a flagship product?
3. How are they similar?
4. How are they different?

**Introduction**

*For more information, reference the “Instructor Reference Lesson 1 Unit 1” document in the Unit 1 Instructor Reference folder*

Video: [Unit 1a\_BasicMath.mov](https://www.youtube.com/watch?v=HxX2UZShVN4)

For our exploration of programming there is a difference between the location where our code is written. There is the MATLAB command window which can execute code as it is typed thus, acting as a “interpreter” for what we input on a keyboard. There are also tabs in the top right corner, which are used for opening and writing ‘scripts’ in the MATLAB interface.



Let’s begin by looking at the MATLAB command window:

Exploring variables:

Type into the command window, line by line:

A = 10

B=12

A+B= 22

**Interfacing with the MiniQ (Arduino based robot)**

Video: [Unit1b\_MATLABwithMotors.mov](https://www.youtube.com/watch?v=IBor4kHFt0g)

Using the MATLAB command window let’s make our first Arduino object using the MATLAB programming language. The miniQ is equipped with an Arduino Leonardo as it’s microcontroller which is basically the “brains” for our robot.

Check your device manager to figure out what the name of the USB port the MiniQ is connected to. On my PC computer it is COM14, therefore:

a = arduino(‘COM14’,’leonardo’)

On a Mac the address is likely to follow a different COM port address, instead of ‘COM14’ it will look more like something along the lines of ‘/dev/tty.usbmodem1421’

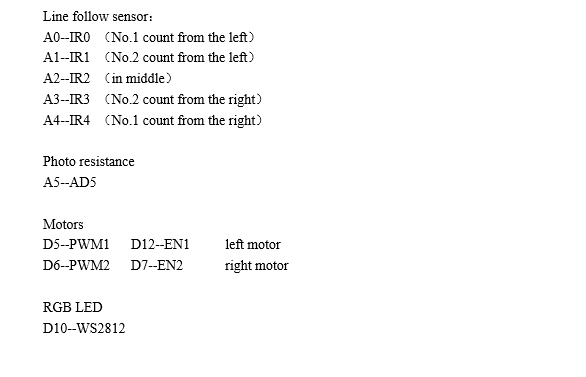
Once the above command is complete, an Arduino “object” allows us to control and experiment with the different pins and sensors attached to our miniQ robot. Here is a table for your sampling and experimentation using the MATLAB command window:

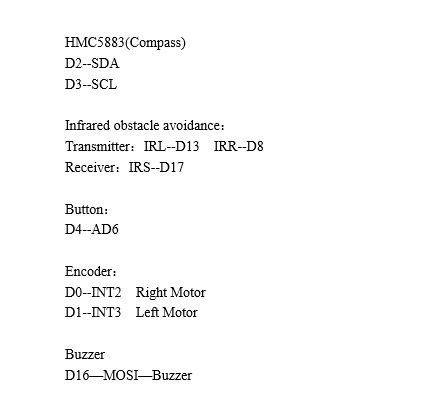
*Pin Map of Robot:*

Reference URL:

<http://www.dfrobot.com/index.php?route=product/product&product_id=555#.VyCwoDArLIU>

For Experimentation:





Let’s experiment with the left motor on the miniQ robot. The pins used on this motor are D5 (PWM) and D12. Calling different values to pin “D5,” which supports pulse width modulation (PWM), allows us to control the direction of the motor. This is a concept we will explore further through writing code in the MATLAB command line.

This is the template for the code we will write:

writePWMDutyCycle(a, 'pin', (enter a value ranging from 0.0 to 1.0);

writeDigitalPin(a, 'pin', (1 or 0 to control direction));

Here is an application of the above template:

writePWMDutyCycle(a, 'D5', 0.5);

writeDigitalPin(a, 'D12', 0);

Try this one:

writePWMDutyCycle(a, 'D5', 1.0);

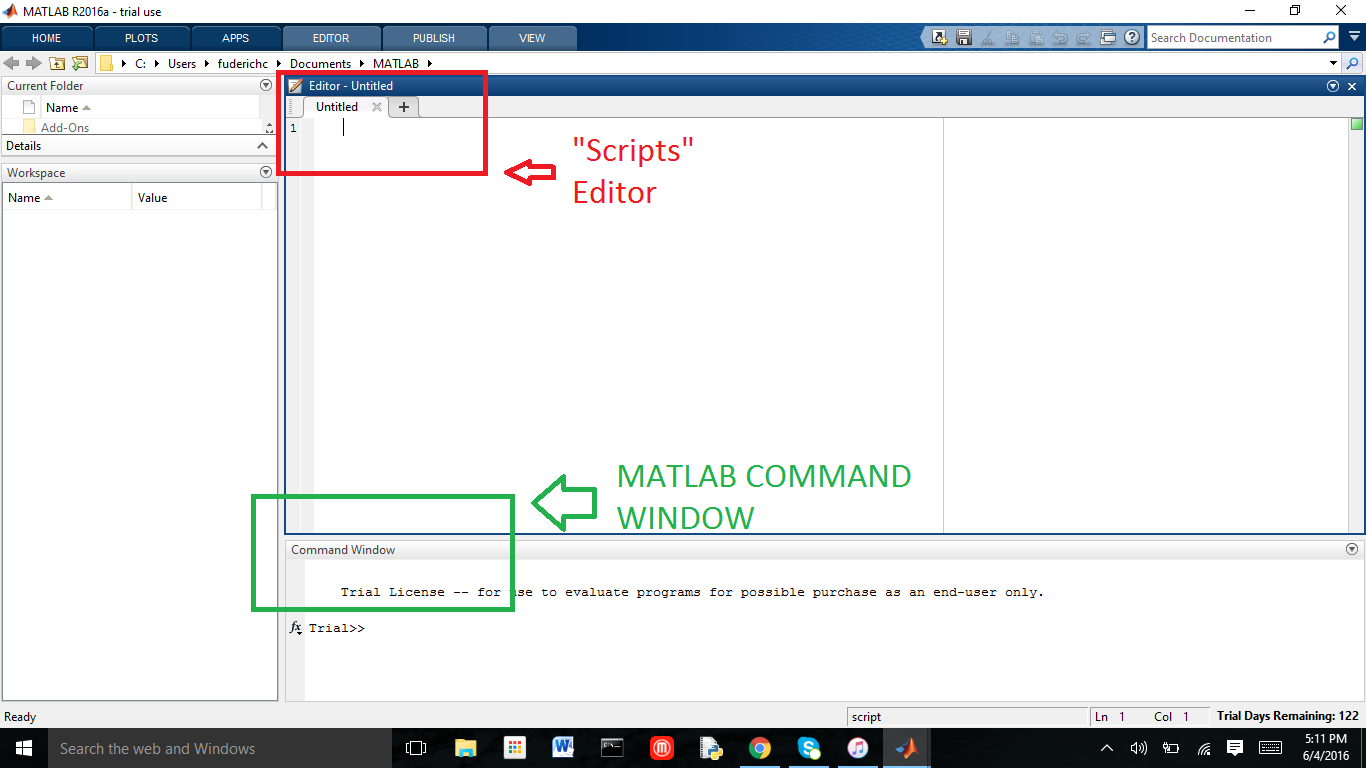
writeDigitalPin(a, 'D12', 1);

What do you think will happen if you type this line in?

writePWMDutyCycle(a, 'D5', 0);

writeDigitalPin(a, 'D12', 0);

**Writing Functions**

Video: [Unit1c\_MATLABFunctionsAndCommandWindow.mov](https://www.youtube.com/watch?v=mha1G8z3qi4)

Click the editor tab in MATLAB and select the “scripts” editor. Paste the code below into your script (not including the *Pure Forward* title as you save the file for execution). This code demonstrates how to build a simple function in MATLAB’s editor.

Note: Make sure “a” is still equal to your Arduino object as done previously. Also, before running this script please make sure that your robot is on a raised platform so it does not drive off of the table. To help me cope with this issue I 3D printed a stand which can be downloaded from the following link as a .STL file:

[*https://drive.google.com/folderview?id=0B8Kb7tLLB0yzMV9jVk9HNGg2VHc&usp=sharing*](https://drive.google.com/folderview?id=0B8Kb7tLLB0yzMV9jVk9HNGg2VHc&usp=sharing)

Save this script under the name “pureforward” then execute the function by writing the function name “pureforward(a)” in the MATLAB command window. Check that the wheels are travelling in the direction you desire.

*“Pure Forward”:*

function k = pureforward(a)

k = 0;

while (k < 10)

writePWMDutyCycle(a, 'D5', .2);

writeDigitalPin(a, 'D12', 0);

writePWMDutyCycle(a, 'D6', .2);

writeDigitalPin(a, 'D7', 0);

k = k+ 1;

writePWMDutyCycle(a, 'D5', 0);

writePWMDutyCycle(a, 'D6', 0);

end

end

*Task to consider:*

1. Manipulate the digitalWritePin() function by entering in a 1 instead of a 0. The wheels will now turn backwards.
2. Change the k value and see what happens? What effect occurs if we change the code to (k<5)? Experiment with all these situations to gain a deeper understanding of the code relative to the function of the motors.

Let’s write a new function. Create another script in the MATLAB scripts editor and save this following file under then name “pureforward2.” Paste the code below into this script:

*“pureforward2”*

function y = pureforward2(a, x)

%this is going forward if x is 0 and reverse if x is 1

k = 0;

while (k < 2)

writePWMDutyCycle(a, 'D5', .2);

writeDigitalPin(a, 'D12', x);

writePWMDutyCycle(a, 'D6', .2);

writeDigitalPin(a, 'D7', x);

pause(0.5)

In the code above we can substitute a value of 1 or 0 for “x” as an input parameter for our function. This saves us time because we don’t have to type in as many lines of code to get a motor to turn forward or backwards.

To test “pureforward2” type this into the MATLAB command window:

pureforward2(a, 1)

Try typing in this one:

pureforward2(a, 0)

Observe the difference between these two lines of code. How does this translate into the rest of the script above?

**Intermediate Programming Activity**

Video: [Unit1d\_CommandWindowWithMotors.mov](https://www.youtube.com/watch?v=8bZJqgUQSrQ) (end of video)

1. Open the “forward,” “left”, “right”, “backward”, “stop” and “motor command window” script examples.
2. Run the “motor command window” script.
3. In the command window experiment with sending the character values “w”, “s”,”d”,”a”.
4. Experiment with changing the character values which execute a particular movement and also if you can get the motors to run for a longer period of time.

**Taking It Further**

Part 1

Video: [Unit1d\_CommandWindowWithMotors.mov](https://www.youtube.com/watch?v=8bZJqgUQSrQ)

Create MATLAB functions for going left, right, backwards,forwards. Click the “+” Icon in the “home” tab to create a new script in MATLAB. If you’ve used the previous examples and they are already open so copy and paste the lines previously written to speed up your processes. Test these functions using the command window till you get the behavior you desire.

Note: you must call the functions you create “forward”, “backward”, “left”,”right” and save them under these names as well. The “forward” function is already available as it is exactly the same code as in the “pureforward” example and can be structured as such.

Nonetheless, here are the functions I created:

*“Forward”*

function k = forward(a)

k = 0;

while (k < 10)

writePWMDutyCycle(a, 'D5', .2);

writeDigitalPin(a, 'D12', 0);

writePWMDutyCycle(a, 'D6', .2);

writeDigitalPin(a, 'D7', 0);

k = k+ 1;

writePWMDutyCycle(a, 'D5', 0);

writePWMDutyCycle(a, 'D6', 0);

end

end

*“Backward”*

function k = backward(a)

k = 0;

while (k < 2)

writePWMDutyCycle(a, 'D5', .2);

writeDigitalPin(a, 'D12', 1);

writePWMDutyCycle(a, 'D6', .2);

writeDigitalPin(a, 'D7', 1);

k = k+ 1;

writePWMDutyCycle(a, 'D5', 0);

writePWMDutyCycle(a, 'D6', 0);

end

end

*“Right”*

function k = right(a)

k = 0;

while (k < 2)

writePWMDutyCycle(a, 'D5', .2);

writeDigitalPin(a, 'D12', 1);

writePWMDutyCycle(a, 'D6', .2);

writeDigitalPin(a, 'D7', 0);

k = k+ 1;

writePWMDutyCycle(a, 'D5', 0);

writePWMDutyCycle(a, 'D6', 0);

end

end

*“Left”*

function k = left(a)

k = 0;

while (k < 2)

writePWMDutyCycle(a, 'D5', .2);

writeDigitalPin(a, 'D12', 0);

writePWMDutyCycle(a, 'D6', .2);

writeDigitalPin(a, 'D7', 1);

k = k+ 1;

writePWMDutyCycle(a, 'D5', 0);

writePWMDutyCycle(a, 'D6', 0);

end

end

Question: What are the similarities and differences between the functions above?

*Note: test all your functions by typing in the following commands into the MATLAB command window*

forward(a)

left(a)

right(a)

backward(a)

Create a new script and run the following code:

*“Motor Command Window Control”*

x = 2;

while (x>1)

prompt = 'which direction to go? w for forward: ';

answer = input(prompt,'s');

if answer == 'w'

forward(a);

answer = input(prompt,'s');

end

if answer == 's'

backward(a);

end

if answer == 'a'

right(a);

end

if answer == 'd'

left(a);

end

if answer == 'q'

break;

end

end

-----------------

Test this code by typing in the characters: w,s,a,d into the MATLAB command window. Remember to press q to stop the program from running. If you don’t do this you’ll get an error.

Part 2

*For more information, reference the “Taking It Further Instructor Reference” document in the Unit 1 Instructor Reference folder*

Previously we took user input to control the robot. Let’s move on to figuring out how to do this with a light sensor:

The robot photoresistor (light sensor) is on pin ‘A5’. Type this into the MATLAB command window:

*readVoltage(a, ‘A5’)*

Execute this command once while covering the “photosensor” and then once with your hand away.

You can use the script below to check sensor data and graph the results:

1. Run the following scripts, first collect data:

*Photo Sensor Reading With Button Control*

button = readDigitalPin(a,'D4');

while (button>0)

reading = readVoltage(a,'A5');

button = readDigitalPin(a,'D4');

disp(reading);

end

1. Press the button on the side of the robot to stop the voltage readings being printed to the command window.
2. Analyze the data gained by scrolling up and down the command window work plane.

*Graphing Sensor Data*

Save the following script as “sensordata” and type sensordata(a) into the command window to execute the graphing of data:

*“sensordata”*

function sensordata(a)

data = [];

for i = 1:20

data = [data ;readVoltage(a, 'A5')];

pause(0.2);

end

plot(data);

xlabel('Time');

ylabel('Sensor Value');

grid on;

assignin('base', 'maxVal', max(data));

assignin('base', 'serialData', data);

disp(['Max Value is:', num2str(max(data))]);

From your graph, identify a baseline line follower sensor value you want to trigger the robot to go on and off. For me it was the following. For you the values may very well be different:

*LightSensorBot*

button = readDigitalPin(a,'D4');

while (button>0)

button = readDigitalPin(a,'D4');

reading = readVoltage(a,'A5');

if reading < 1

forward(a);

else

backward(a)

end

end

Additional Tasks to Consider

1. Create a program that controls the robot wheels to move when a button is pressed
2. A light seeking robot which follows your hand around; this will be dependent upon a shadow being cast over the photoresistor sensor.

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/).