SYNTHIAM

synthiam.com

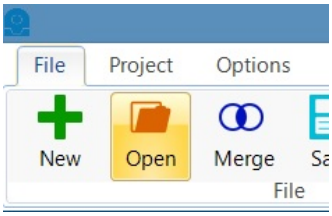
Reading and Writing with Files

This tutorial is an introduction to working with text files in EZ-Builder. EZ-Builder gives you the ability to write information to text files and read information from text files. You can use this for logging data, giving your robot a memory, a way to read in data you might not want to manually enter in a script and anything else that your imagination might lead you to use text files for.

Last Updated: 11/20/2017

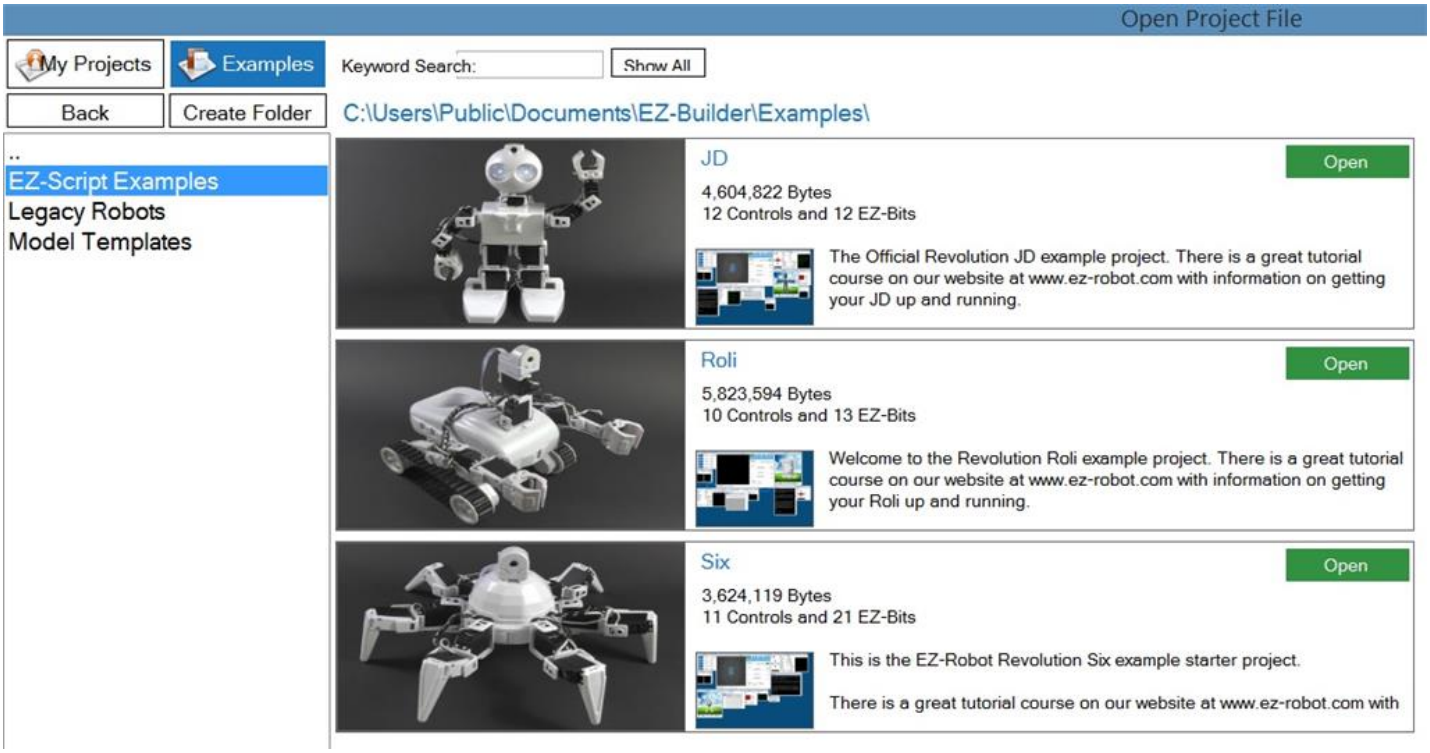
Step 1. Open a project

With EZ-Builder, locate the "Open" icon and click it



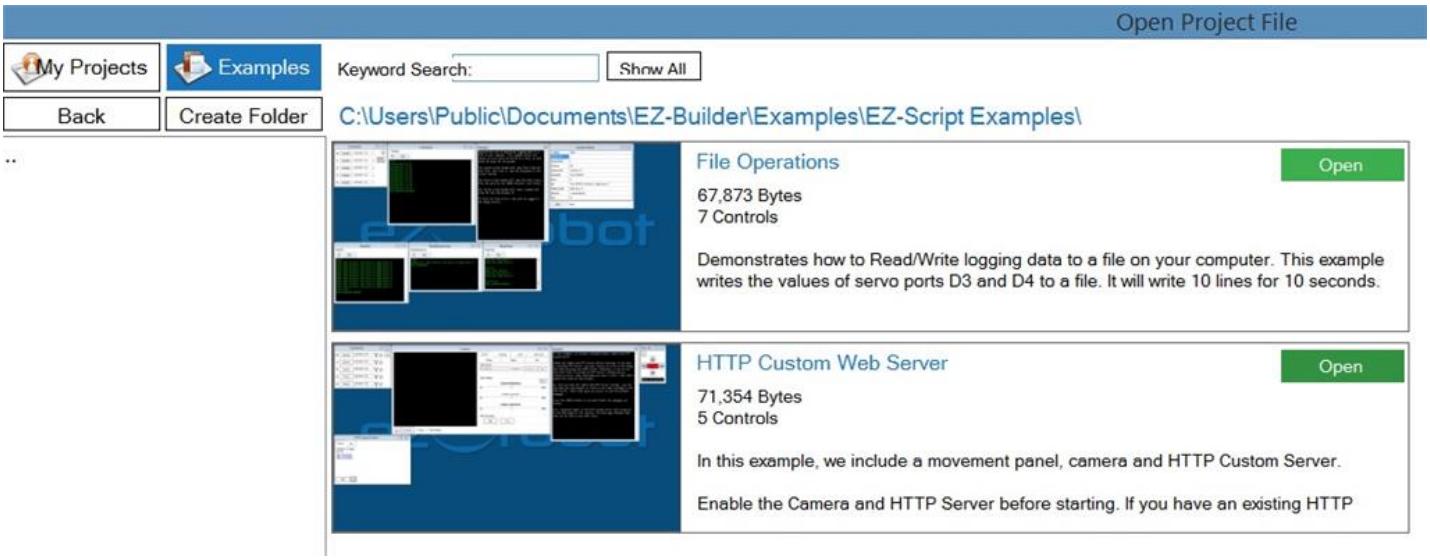
Step 2. Select EZ-Script Examples

Click Examples then EZ-Script Examples.



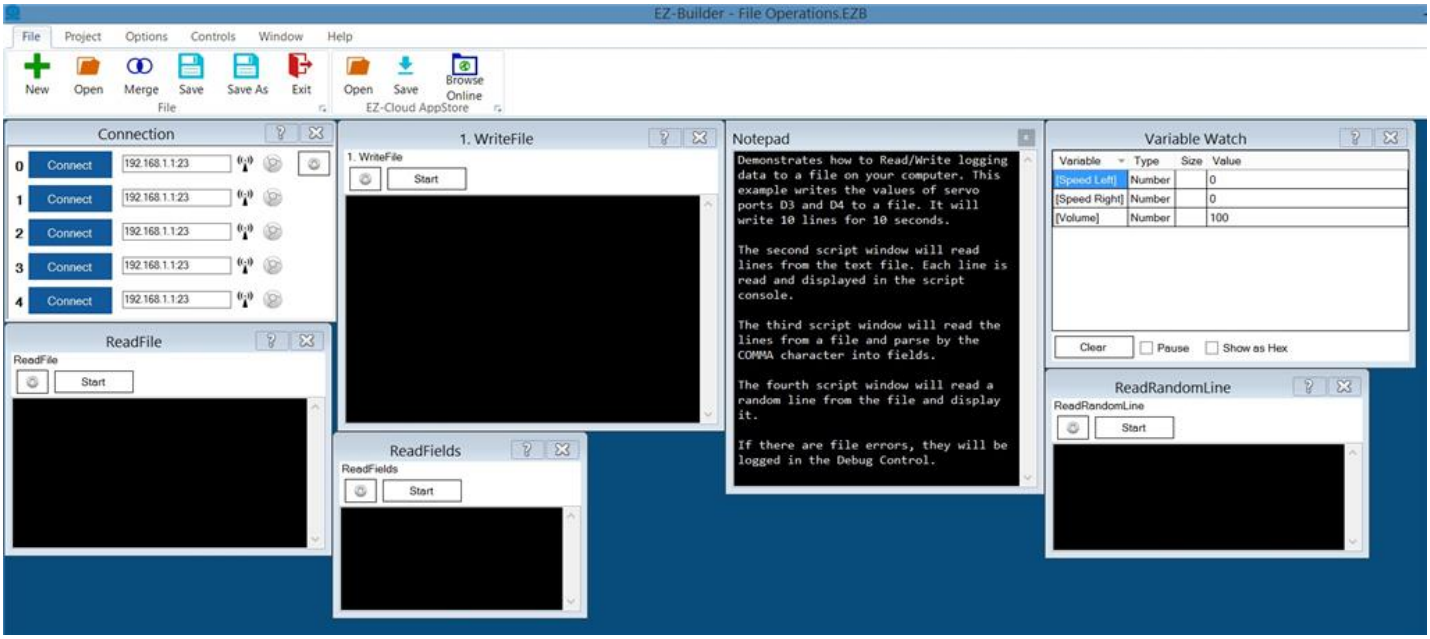
Step 3. locate File Operations example

Locate the File Operations script example. You can type "file" in the Keyword Search window to help you find it. Once you find it, click Open.



Step 4. with the File Operations example open

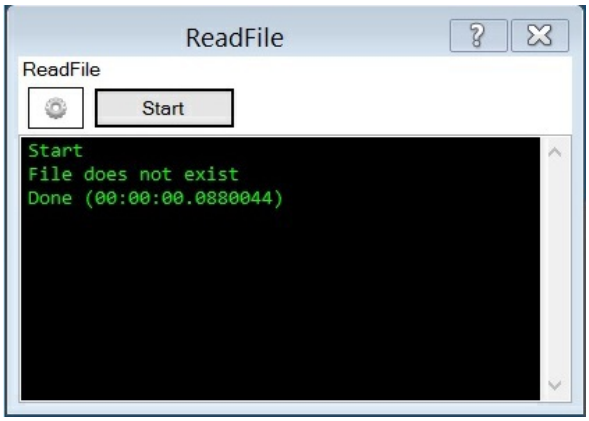
Once the File Operations EZ-Builder script example is open your screen should look similar to this:



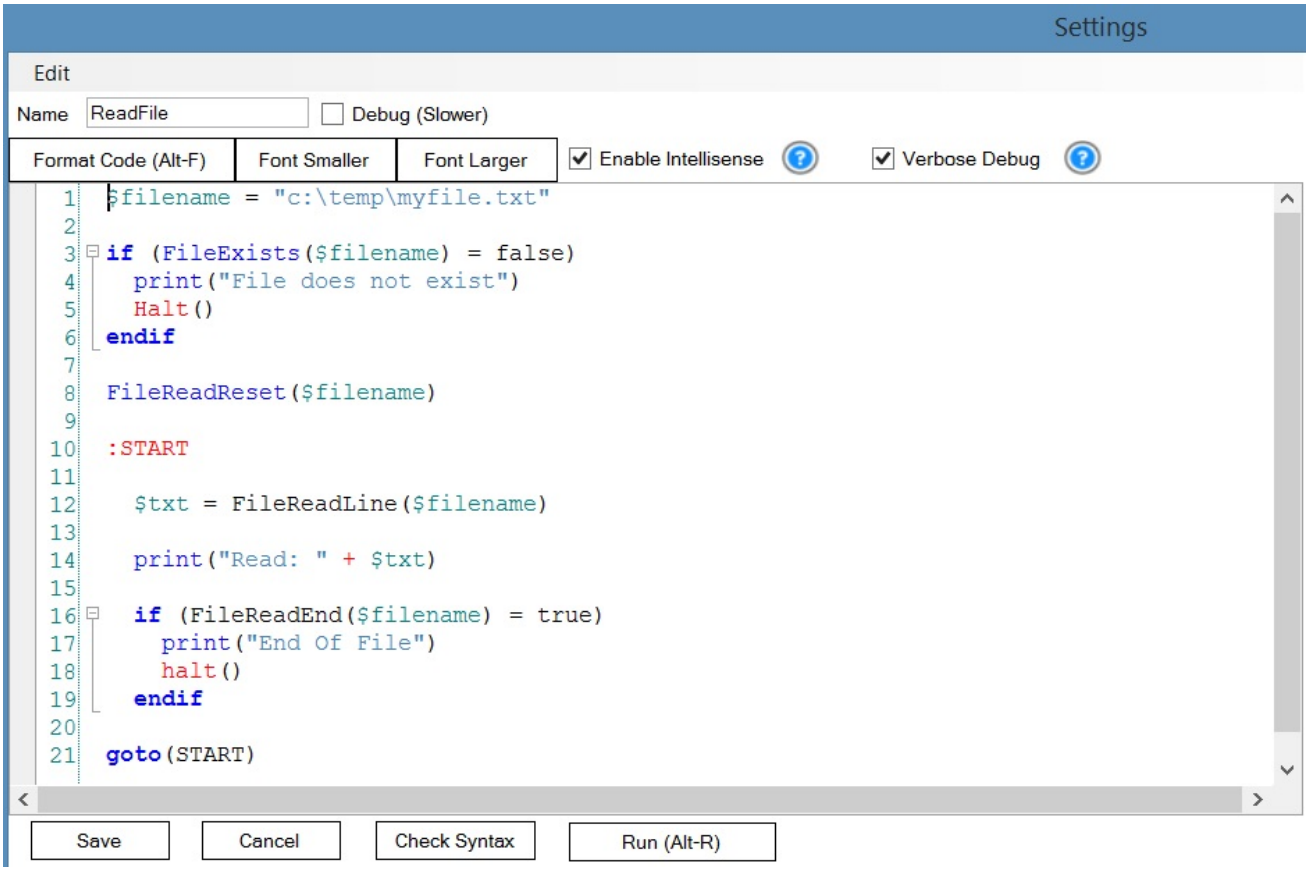
Step 5. Make a temp folder, file and read a file

To work with text files, you'll probably want a folder or primary location where you are going to store files that you read and write. In this example we want to create a folder on the c:/ drive named "temp".

Let me show you why, if you Click Start on the "ReadFile" script (without the "c:/temp" folder and a text file to read, you will get an error message stating "File does not exist".



Lets look at the code of this script to see how our file is called up to read it. Click on the gear icon (settings) for the "ReadFile" script.

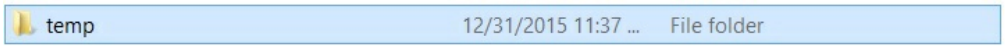


The first line in our script sets a variable named $filename equal to the path and file name of "c:\temp\myfile.txt" Then we see and "IF" clause that if the file and path can not be found, then print out on the screen "File does not exist", just as we saw when we ran the script a moment ago. The script is then halted.

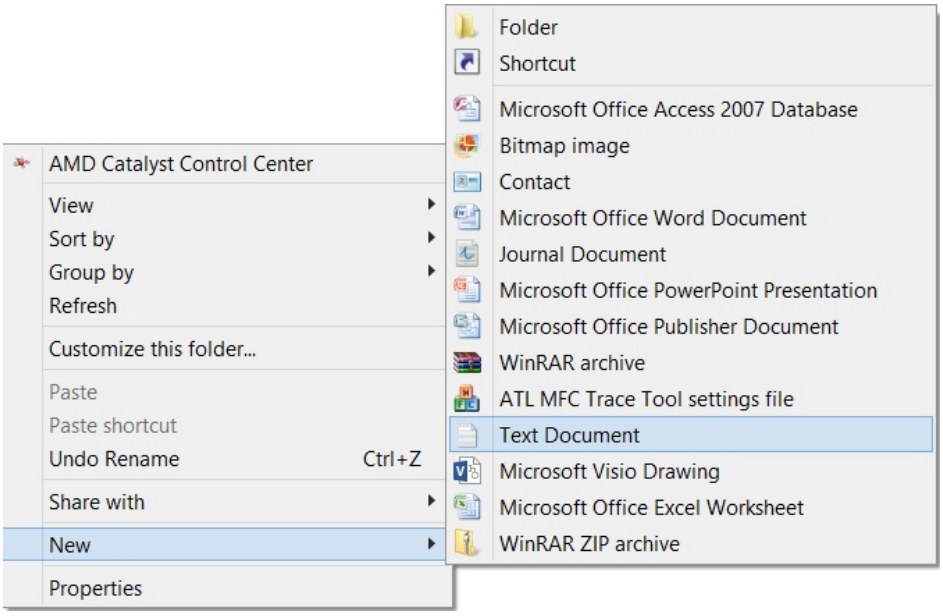
This is a very good function to include when reading or writing files, because text files might get deleted or moved or just stuck depending on what you might be doing with files and how much you

are using them. So making sure your script has an "out" method such as this to check the path and halt the program. It is a smart way to address any issues that might come up when working with text files.

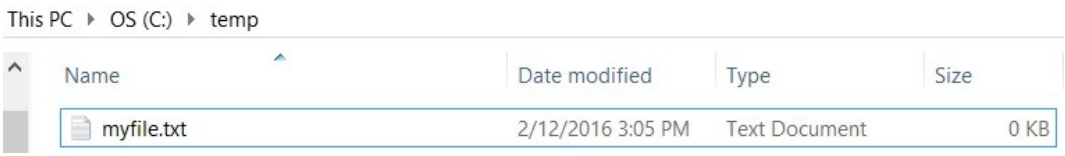
Before going further, lets make sure you have created a temp folder on your c:\ drive.



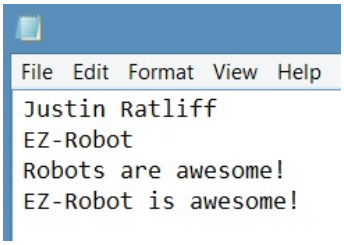
Now create a text file in the folder, by right clicking in the folder and select Text Document.



Name the file "myfile".



Now enter some lines of text in the file and be sure to save it!



Now lets run the "ReadFile" script again. You should get the following successfully outcome from reading the file. Each line should be read and output until the end of the file is reached.

Step 6. Writing to a file

Now comes the fun part, writing to a file. The example script "1. WriteFile" will write data to your "myfile" for servo positions if your EZ-B is connected. If it is not connected or an error occurs it will delete your "myfile.txt" file.

Lets create new script, by going to Project/Add Controls/Scripting and select EZ-Script. Your new script control will pop up with the default name of "Script". Click on the gear "setting" for the script and rename the script "WriteTest".

Then enter the following code, then save the script: ``` #WriteTest

$filename = "c:\temp\myfile.txt"

FileReadClose($filename) #FileDelete($filename)

$TestValue1 = 24 $TestValue2 = 10 $cnt = 0 $max = 3

:START

print("Writing line " + $cnt + " of " + $max)

FileWrite($filename, "Time: " + $time) FileWrite($filename, ", Left Servo: " + $TestValue1) FileWriteLine($filename, ", Right Servo: " + $TestValue2)

$cnt = $cnt + 1

if ($cnt = $max) halt() endif

sleep(100)

goto(START) ```

What we are doing here is creating a new version of the same write script included in the example but we are manually adding in test values for our servos so that we don't have to have the EZ-B connected or servos connected to certain ports.

We have also commented out the line to delete the file. Deleting the file is handy if you wanted to log data with a refreshed (or clean) log file. With the delete file line commented out with the # we are ignoring this option and each time you run this script it will keep adding to the file, retaining what was previously written to the file.

the $cnt and $max variables are used to tell the script to loop through the number set as $max, in this example 3 times.

This is a great example of a few different ways you can write data to a file all in one script, but it might be a little busy. You can greatly simplify writing to a file.

Add a new script control and name it "WriteTest2" and enter the following code, the save the script:

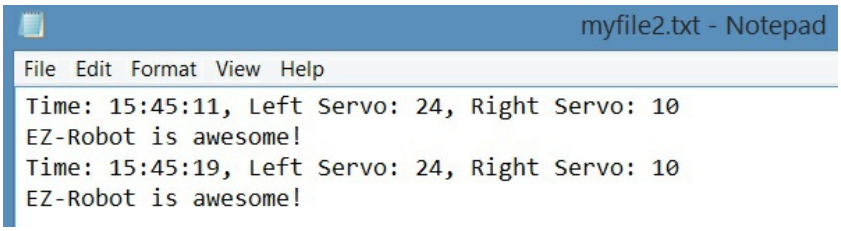
``` #WriteTest2

$filename = "c:\temp\myfile2.txt"

FileReadClose($filename)

$TestValue1 = 24 $TestValue2 = 10

FileWrite($filename, "Time: " + $time) FileWrite($filename, ", Left Servo: " + $TestValue1) FileWriteLine($filename, ", Right Servo: " + $TestValue2) FileWriteLine($filename, "EZ-Robot is awesome!")
```

sleep(100) ```

This code will write a single file line with the time, Left Servo value and Right Servo value (because it does not write the line until we tell it to do so with FileWriteLine) and then it will add a second line that says "EZ-Robot is awesome!".

We also added a new file name. We do not have to create this file ahead of time. If the file is not there, it will be created. If the file is there, it will be added to with new lines each time you run this script.

If you run this script once, it will write this data once and you'll have two lines. Each time you run this example, 2 new lines will be added to the file.



You can simplify the write process even further in this example, add another script control and name it "WriteTest3" with this code, then save the script:

``` #WriteTest3 $filename = "c:\temp\myfile2.txt" FileReadClose($filename)

$PingValue = 124 FileWriteLine($filename," Ping ," + $PingValue)

sleep(100) ```

With this code, we are just writing the word Ping with a comma and then the variable we set as the value. As you can see, this could easily be the actual variable you use for your Ping sensor values.

Lets say you wanted to add a time stamp to this example. Would this work? Give it a try by modifying the line as seen below.

``` FileWriteLine($filename,$time + " Ping ," + $PingValue) ```

This will allow you add a time stamp to the line.

Another option is to comma separate all the items you write in a single line:

``` FileWriteLine($filename,"Time:," +$time + ",Ping," + $PingValue) ```

Writing data to a file is all well and good. It can be handy if you wanted to log data and review it yourself. But if you want to make use of it within EZ-Builder, you need to write it in a specific way that your can script read it so your robot can make use of it.
```

Step 7. Making real use of data from a file

If you write to a file "Time Ping 124" you are going to have a very hard time using it in another script. If you separate it with commas so that it is written as "Time,Ping,124" it is going to make it easier for a script to read that line of data and make sense of it. Because you could script reading data after the second comma, or any other place in the line because the data is separate by commas which is what allows a script to read the data in an understandable way.

But be careful how you use the commas. If your file output ends up being "Time , Ping , 124 " those spaces will probably throw off the usability of those values, because while you know the value for the ping sensor is "124", what your robot will read it as is " 124 " which renders the data nearly useless because of the extra spaces before and after the numerical value of 124.

Of course you could simply write ping data like this to a single line with only the ping sensor value, just "124" with nothing else.

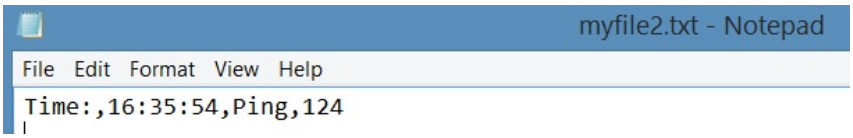
Having said all that, let's make sure your file output from "WriteTest3" script is clean with a slight modification. Lets clean up our script with the FileDelete line add back. Modify the WriteTest3 script as shown, save it and run it.

``` #WriteTest3 $filename = "c:\temp\myfile2.txt" FileReadClose($filename) FileDelete($filename)

$PingValue = 124 FileWriteLine($filename,"Time:," +$time + ",Ping," + $PingValue)

sleep(100) ```

After running the script, open the "myfile2" and make sure it looks like this:




If it does, close the file. If it does not, review the previous steps to see what you might have missed.

There is a sample script named "ReadFields" that we could use to see how to read comma separated text files, but I'm going to show you a slightly different example. Add a new script control and name it "ReadCommaFile" and enter the following code, then save the script.

```

Reads a file and seperates each field by a comma.

The Split() function will return a

specified field index

$filename = "c:\temp\myfile2.txt"

if (FileExists($filename) = false) print("File does not exist") Halt() endif

FileReadReset($filename)

$cnt = 1

:START

$txt = FileReadLine($filename)

$TimeField = Split($txt, ",", 1) $SensorField = Split($txt, ",", 2) $SensorValueField = Split($txt, ",", 3)

print("Row: " + $cnt) print("Time: " + $TimeField) print("Sensor: " + $SensorField) print("SensorValue: " + $SensorValueField)

if (FileReadEnd($filename) = true) FileReadClose($filename) print("End Of File") goto(END) endif

$cnt = $cnt + 1

goto(START)

:END ```

This reads the line of data from our file and separates out the data by commas so it can be used. The data in our file was:

Time:,16:35:54,Ping,124

Going by comma position, the data breaks down as follows: Time: = 0 16:35:54 = 1 Ping = 2 124 = 3

So the line: $SensorValueField = Split($txt, ",", 3) is reading the data after the 3rd comma. And $SensorValueField is the variable we are using to store that value.

Now you can use this value elsewhere in other scripts if you need to. Lets add a new Script control and name it "NearOrFar", then enter the following code and save the script.

``` if ($SensorValueField > 100) Print("I was far away from something.") else Print("I was near something.") endif ```

Run the script. Based on the value being 124, the script should print "I was far away from something.".

Now go back to your "WriteTest3" script and change the value from 124 to 80. Save the script and run it.

Now run the "ReadCommaFile" script again.

Now run the "NearOrFar" script again. This time the script should print "I was near something.".
```

Step 8. Reading Random Lines

There is another example script named "ReadRandomLine", go ahead and run that script. It will read a random line from your "myfile" text file.

One use I could see for this would be to use it to read a text file and pick a quote, greeting or joke at random. Lets create a random joke teller to see how we can us this:

1. Create a new text file in your c:\temp folder named "jokes"
2. Lets add some jokes, but they have to fit on one line. Here are a few I found online:

A recent study has found that women who carry a little extra weight live longer than the men who mention it.

I find it ironic that the colors red, white, and blue stand for freedom until they are flashing behind you.

Today a man knocked on my door and asked for a small donation towards the local swimming pool. I gave him a glass of water.

Add those to your joke text file and save it. You should end up with 3 lines of jokes with no spaces between the lines.

3. Now add a new Script control and name it "RandomJokes" and enter the following code and save the script:

```
# Reads random line from the specified file

$filename = "c:\temp\jokes.txt"

if (FileExists($filename) = false) print("File does not exist") Halt() endif

$cnt = 1

$txt = FileReadLineRandom($filename)

print("Random Line: " + $txt) Say($txt) ```

This will print the joke in the output of the script debug window as well as speak it from the computer speakers.
```

Step 9. Save your Project

The last step is to save your project as new project (if you want to). Click on File, Select Save As and give your project a new name so you can save it and not overwrite the example project.

You never want to overwrite the included example projects, for one you'll lose them as good examples. And two, those example files will be overwritten when an updated version of EZ-Builder is installed in the future, which means your work would be lost.

So never save changes to an example project.

Happy EZ-Roboting!!

Download the latest Example Project which includes additional sample scripts not covered in this written tutorial:JrFileReadandWriteSamplev4.EZB

Related YouTube videos: