**LABVIEW**:

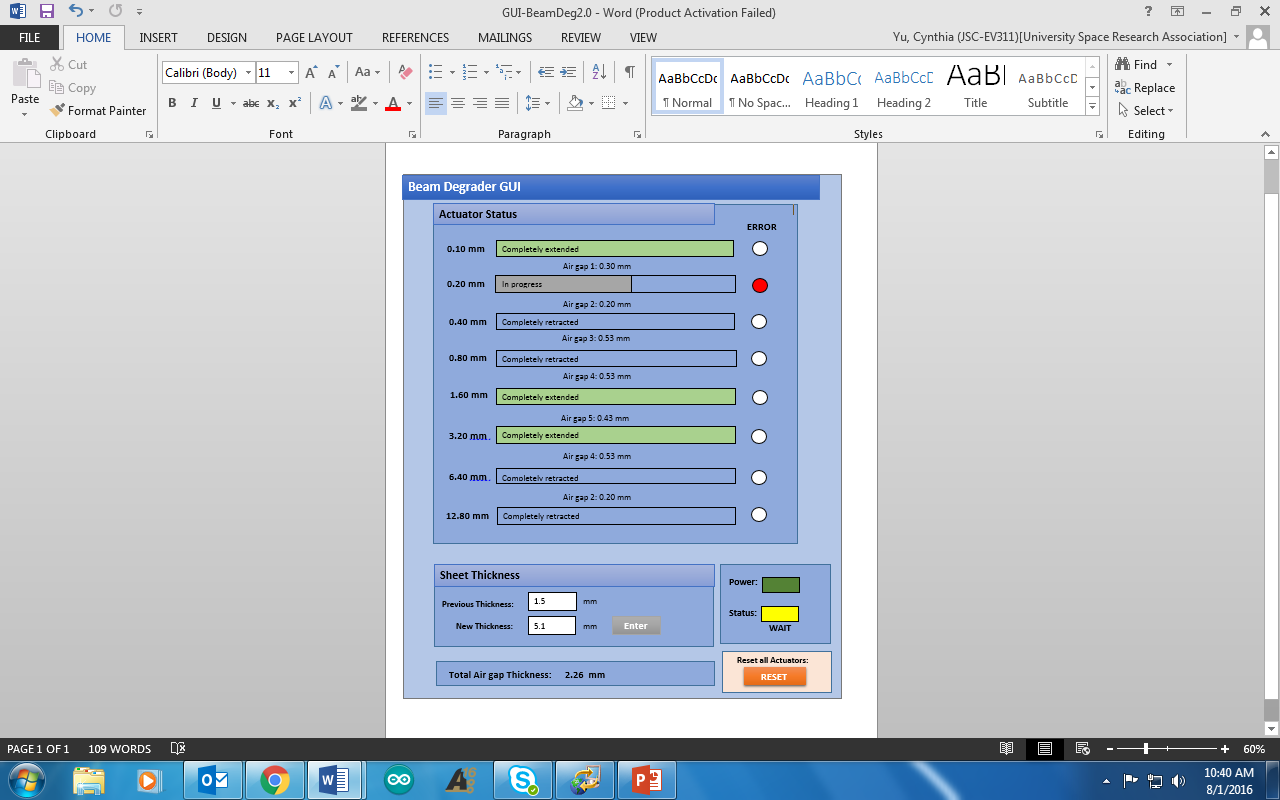
\*\*Another quick overview of what we are trying to accomplish, we want the user to input in a desired thickness and based off of that thickness the actuators will extend and retract to accommodate that thickness.

So far I’ve talked about the electrical design but not about the User interface which is really important, because we want the overall design for the user to be VERY EASY and STRAIGHTFORWARD.

We want the user to be able to see the position of each actuator and if there are any errors with each actuator. Also we want an area where the user can input in a desired thickness.

-- I made a sample GUI on word and attached it in a word document on the disc. It should be titled, “**GUI-BEAMDEG\_2.0**” (please note the 2.0 version is the update and much better version).

Here I’ll attach a screenshot of what it looks like and explain the design.



This is the sample GUI I created on word. I just threw together something that looks very straightforward and easy to understand. The 8 bars you see above are showing the user the “Actuator Status” It will update instantaneously the position of each actuator and show when each actuator is fully extended or retracted and process while it is happening. The red bubble on the right shows if there is an error detected on each actuator. If there is a detected error, the bubble will light up red. In between each Actuator status bar is telling the user the air gap thickness between each actuator. This is what Andy recommended I add in, and is completely open to alternatives if not needed in the final design.

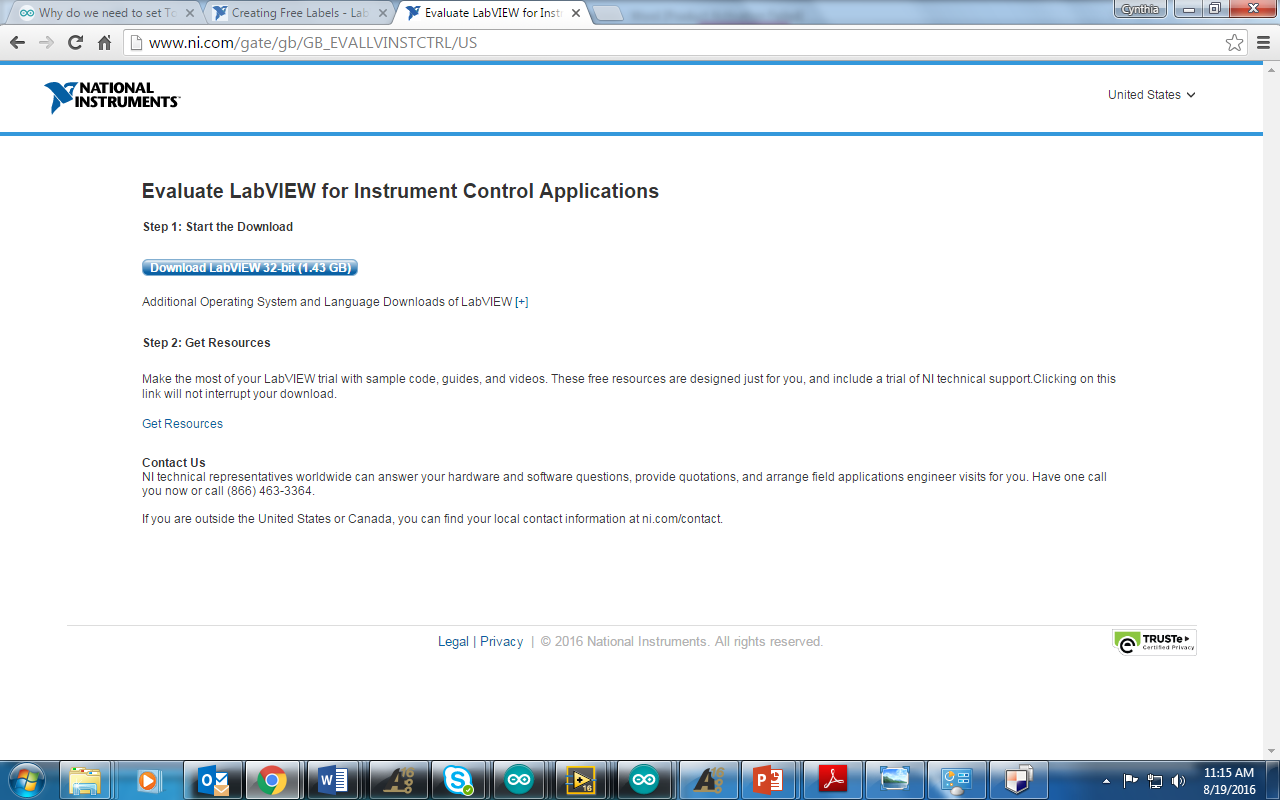
The bottom portion is showing the Desired sheet thickness and an enter button after the thickness is entered. There is also a button to RESET all the actuators, which will pull back all the actuators if pressed.

The two square LED’s are showing the Power and Status of the actuator. So the Power is referring to whether there is power supplied to the mechanism and board. The Status LED is showing the status of the actuators in accordance to the final desired actuator position. This LED will vary from Yellow to Green to indicate it is in its final destination. It also has a corresponding text associated with it that says “WAIT” and “READY” to indicate it is ready to have the radiation beams sent through the sheets.

**DOWNLOAD LABVIEW:**

To download LabVIEW go here: <http://www.ni.com/download-labview/>

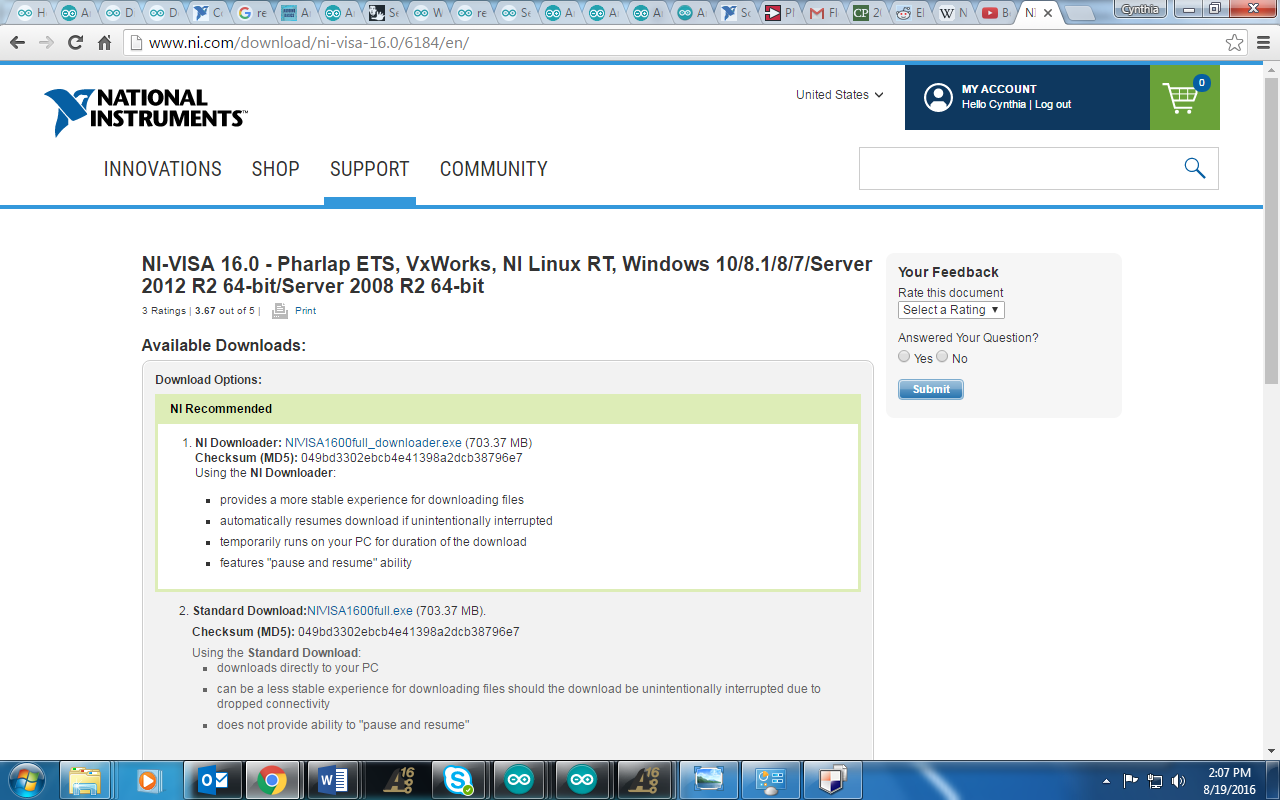
And download LabVIEW. It will give you a 7 day trial, but once you have it fully downloaded you can update the trial to 45 days.



\*\*\*IMPORTANT: \*\*\*Also be sure to download the **NIVISA1600full**, this allows you to interface between LabVIEW and the Arudino Serial port.

Download it here: <http://www.ni.com/download/ni-visa-16.0/6184/en/>

The picture below shows what the download screen looks like.



So **LABVIEW** is what I used to create the GUI for the User. The thing I just showed you was just a sample of what I intended to create, but isn’t what the “true” GUI actually looks like. So my Labview version GUI so far is still in the process of being configured, but it looks like this:

\*This is known as the **Labview Front Panel**, and is what the user will see displayed\*

\*\*I will try my best to explain the LabVIEW program to you, but I really recommend that you watch some YouTube videos and read over some online tutorials before reading this any further. Because watching the videos/tutorials will give you a better understanding of what I am talking about rather than just reading my explanation without any background information.

\*\*Some **YOUTUBE** videos I recommend watching that helped me were:

<https://www.youtube.com/watch?v=PqxStfwjQoQ&list=PL085A358B79B37463>

You should watch all of his videos because each of them are extremely short and easy to understand. He explains each of the processes very well and it helped me a lot.

This one kind of helped: <https://www.youtube.com/watch?v=I8pc8-VcVFo>

Honestly the most helpful thing was to play around in LabVIEW and read tutorials to get more familiar with it.



**ABOVE**: **FRONT PANEL (GUI**) this is what I created so far and it actually interfaces with the Beam Degrader prototype that we have so far. But there are only 2 actuator’s connected so we can only output and control two actuators. But I have already configured the entire design to be compatible with all 8 actuators. Everything that is outside of the box is not going to be included in the final GUI for the user. That stuff is simply for testing purposes so that I can see what exactly is being sent to the Arduino and so forth. So don’t mind those. But basically on the left it is telling me the digital value of the actuator position and the box on the right are showing me the output to the error flag. (Remember 1 = no error, and 0 = error). The bottom left output is showing me the output that is actually being sent to the Arduino, which is being multiplied by 10 because we don’t want decimals being read in Arduino for the reasons I told you before.

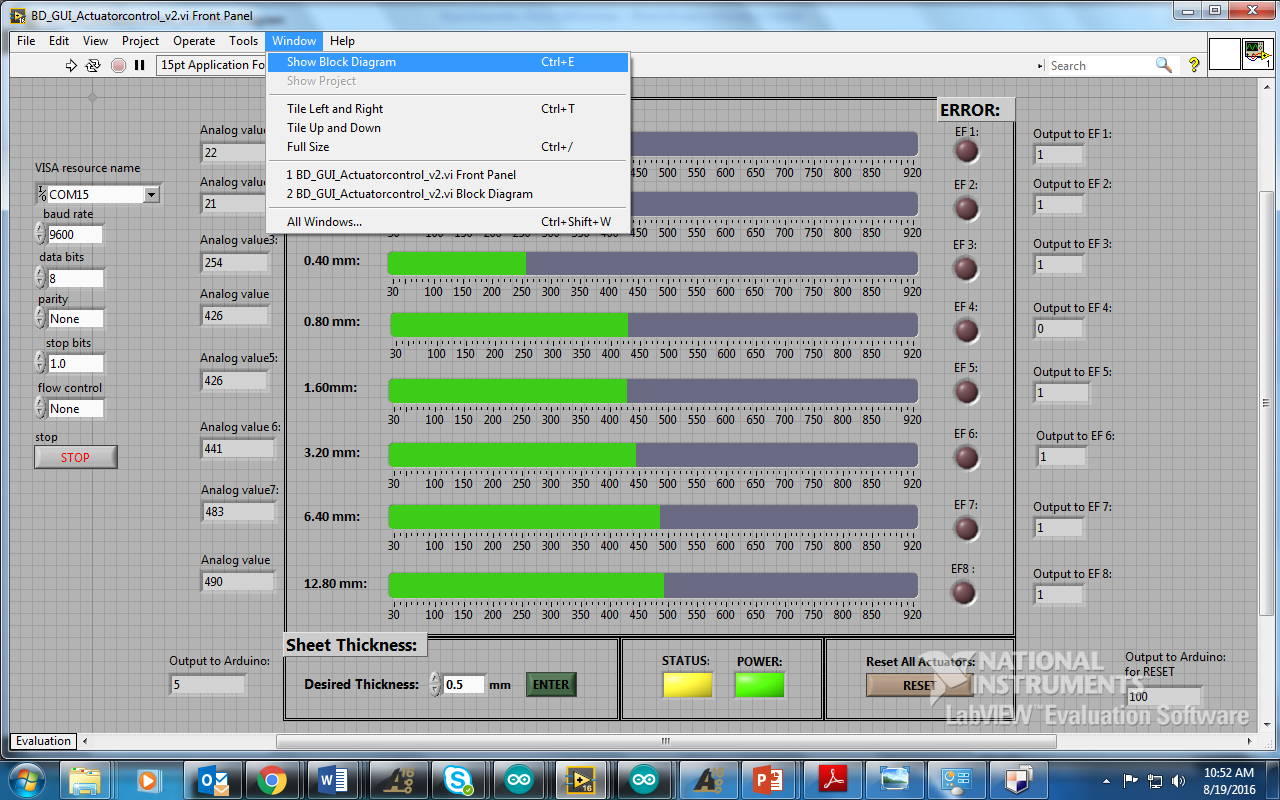
That is why I multiplied the incoming desired thickness by 10.

The box on the right is the output to the Arduino when the user selects he wants to reset all the actuators. By clicking that button, we basically send that we want an overall thickness of 0 to be applied to the actuators and to pull all the actuators back. So that output should read 0 when the RESET button is pressed.

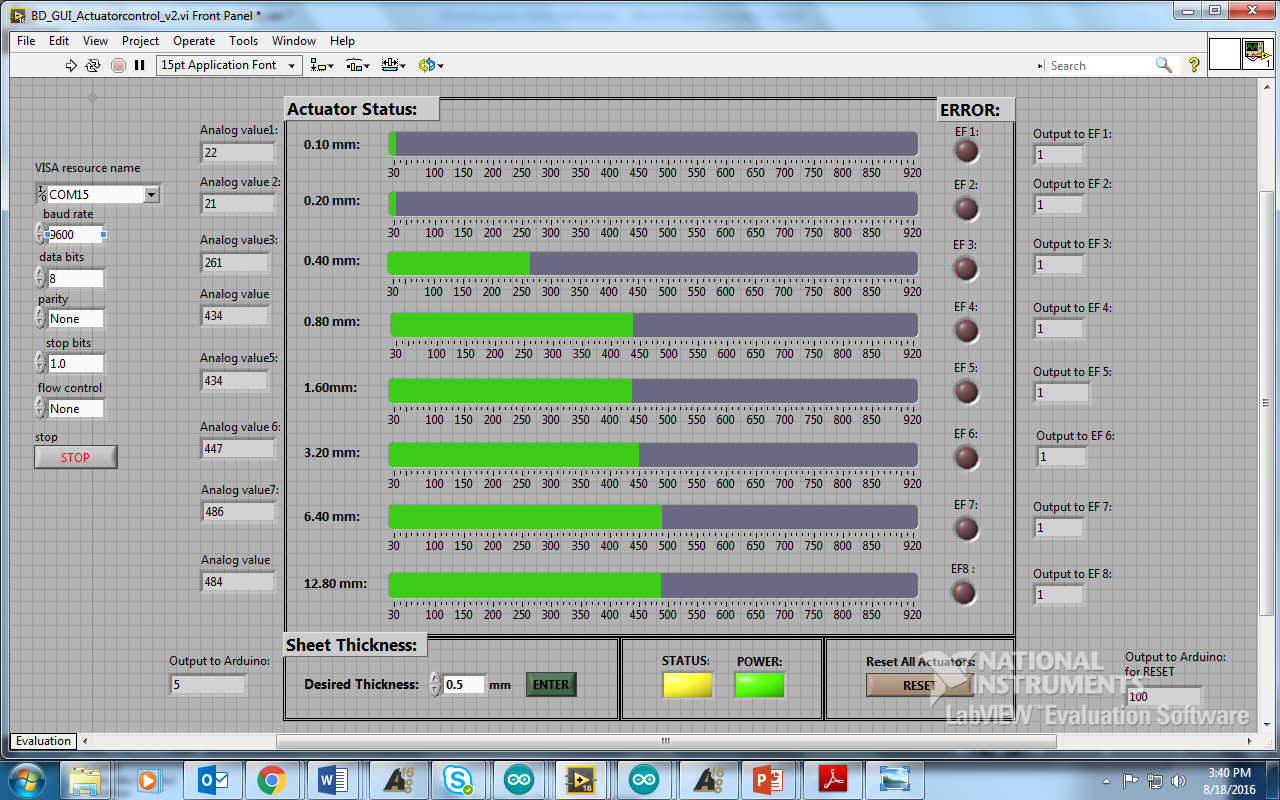
**BLOCK DIAGRAM:**

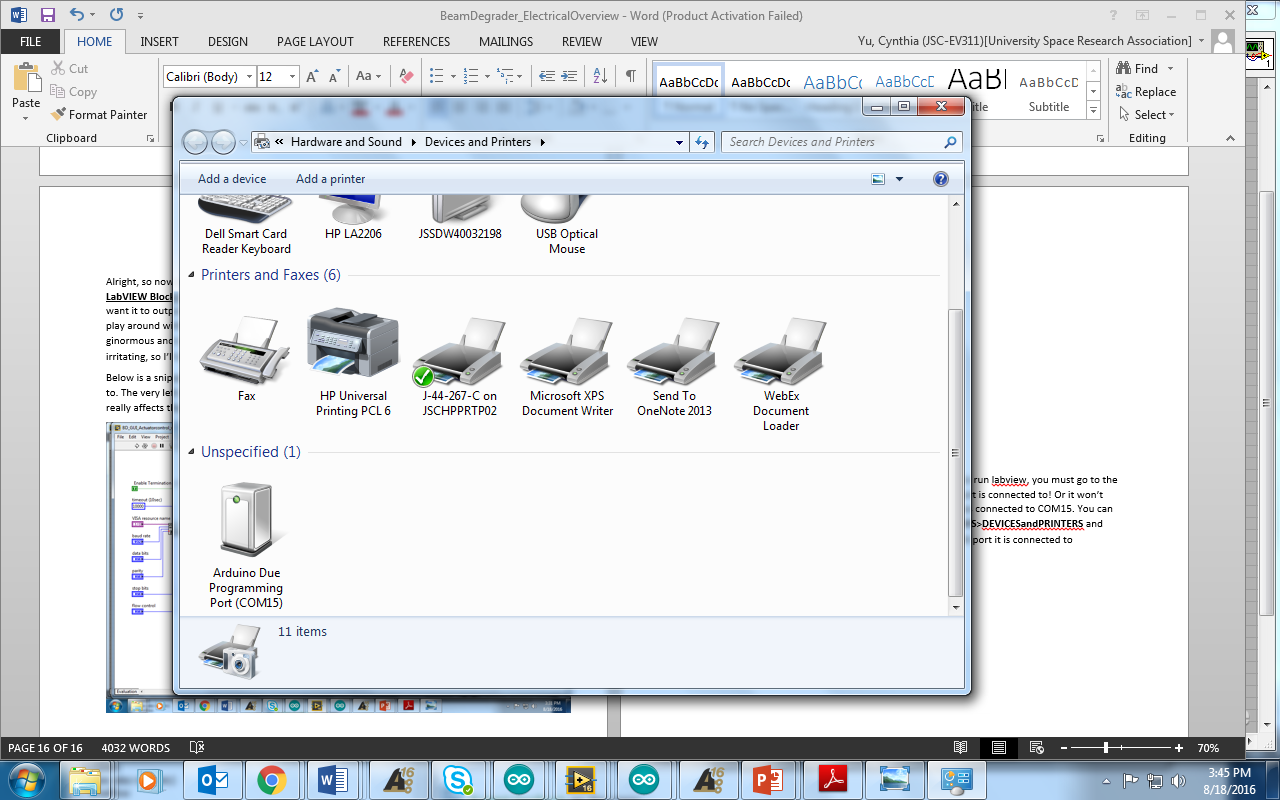
Alright, so now you have a good understanding of the Front Panel, I’m going to show you the **LabVIEW Block Diagram** which is basically connecting the GUI front panel to everything we want it to output to the Arduino. It might look a little intimidating because it is, but after you play around with it, it’s actually really easy and straightforward. So the whole block diagram is ginormous and won’t fit on one page, and LabVIEW doesn’t let you zoom out which is pretty irritating, so I’ll show you everything in multiple screens.

\*In order to access the Block Diagram simply press Ctrl+E, or go to WINDOW>SHOW BLOCK DIAGRAM. Then the block diagram window will pop up.



\*\*However DO NOTE that on the front panel, you must change the “VISA resource name” to connect to the write COM port that your Arduino is connected to.

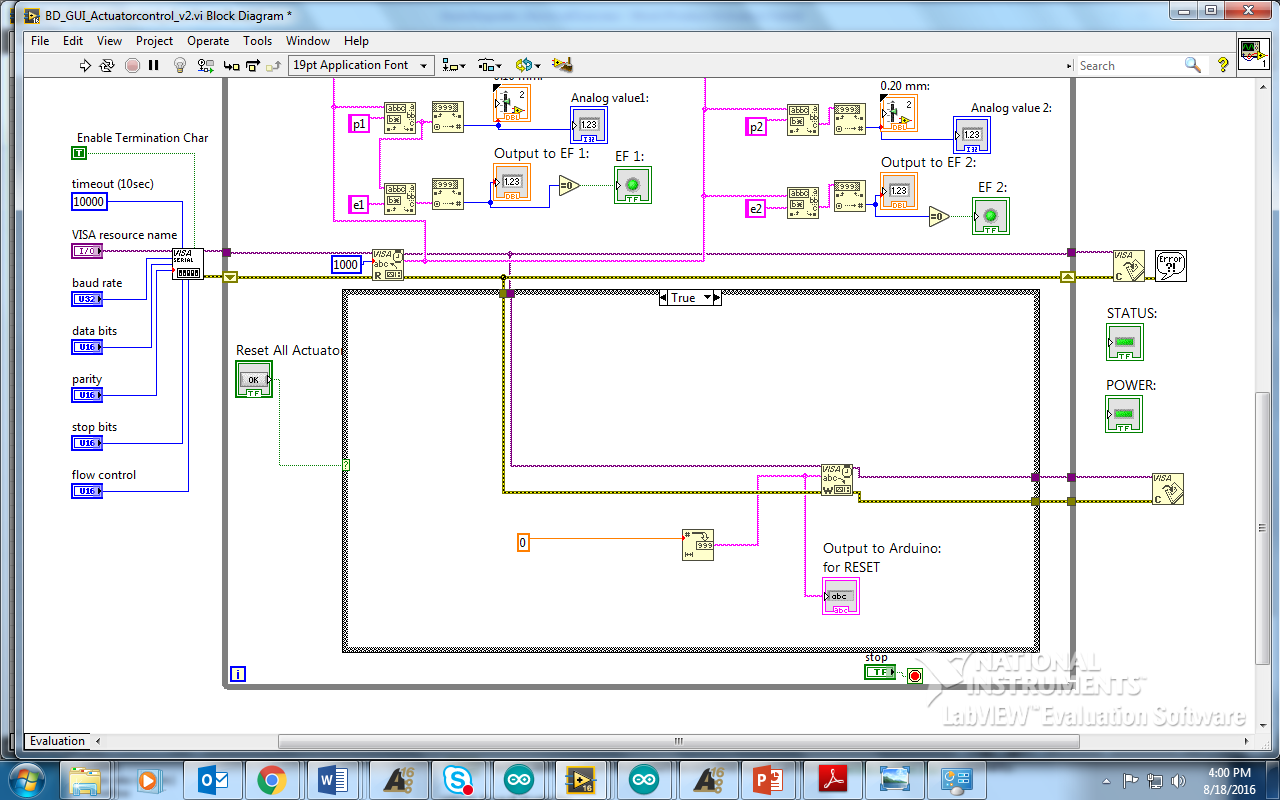
 **REMEMBER**: when you are about to run labview, you must go to the front panel and select the VISA resource name the Arduino port is connected to! Or it won’t run!! REMEMBER THIS!! In my case the Arduino Serial port was connected to COM15. You can check which Serial port your Arduino is connected to **WINDOWS>DEVICES and PRINTERS** and then looking for “Arduino” and look for the corresponding COM port the Arduino DUE is connected to.



**Figure 1:** Below is a snippet of the Block Diagram showing you what each of the inputs will connect to. The very left side (circled in **RED**) is stuff that you don’t really need to worry about because none of that really affects the outputs to the serial port right now.

**HELP FEATURE**

\*If at any point you don’t understand a little box in the Block Diagram, right click the box and press **HELP**. This will give you a good understanding of what exactly that block will do and how it works. It also will sometimes provide you with a few examples of that block in use.



**Figure 1**

The little box that I circled in **ORANGE** is also very important. That little box is the VISA SERIAL box. So that box is basically where all the information and data coming from the Arduino to be interfaced in LabVIEW. If you don’t have that box then you won’t be able to get anything coming from the Arduino. Basically the Arduino is sending data the Serial port to LabVIEW! T

The purple wire coming out of it is where the data is being pulled out from, the yellow looking wire coming out is any errors that are detected form the Serial Port.

The next thing that I circled in YELLOW is also VERY IMPORTANT! This thing allows you to Read the data being sent form the Serial port. Without this box we would be unable to Read the data being sent. So remember this thing. Right click and press HELP at any point you want a better understanding of the box features.

The box circled in **GREEN** is the Serial WRITE box that enables us to write an output to the Arduino Serial port.

But before this which is what I circled in **BLUE**, is the output that is sent to the Arduino, so the little Orange 0 that you see inside the circled blue area, is showing you that I want a value of 0 sent to the next box which is the box that converts a NUMBER to a DECIMAL STRING. So a string is being written to the Serial WRITE box and then outputted to the Arduino.

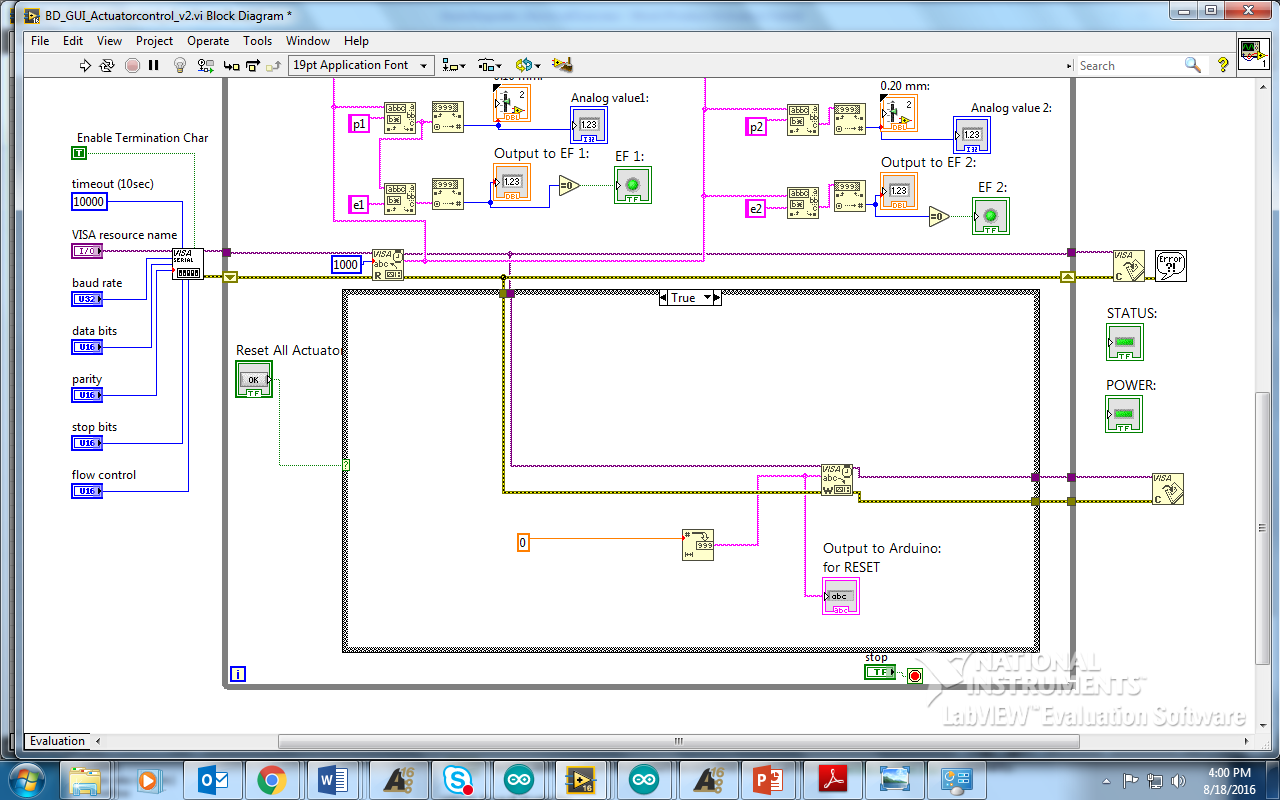
The box circled in **PURPLE**, is the VISA close box that basically closes the serial port to the Arduino so that after everything is written and sent out the serial port to the Arduino, we close the port so that it can be used by something else if needed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Alright, so **Figure 2** shown below is the most straightforward program I can explain so I am going to start with that. What this block diagram is showing is that the Boolean case structure (**PINK circle**) is showing, is that LabVIEW is WRITING a 0 to the Arduino’s serial port whenever the case statement is TRUE.

The GREEN circle is circling the input of the Boolean case structure. That is connected to the RESET all ACTUATORS button, therefore whenever that button is pressed on the Front panel (input to Boolean case structure = TRUE) then 0 will be written to the Arduino (ORANGE).

The **RED** is branching off of the string that is being sent to the Arduino Serial port block because this block is simply for testing purposes. I wanted to see a display of what was actually being sent to the Arduino Serial port. (Which in this case should read a 0 on the front panel).



**Figure 2**

Be prepared the next diagrams are going to get a bit more complicated, so if you aren’t familiar with LabVIEW yet, please go watch tutorials or it will confuse you.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Figure3**: I just explained what will happen to the outside Boolean case structure when the input is TRUE, now I’m going to explain what happens when the **Boolean case structure = FALSE**(when no one presses the ‘RESET all ACTUATORS’ button). Where I circled **PINK** is the Boolean case structure for the RESET all ACTUATORS case statement.

When **RESET = FALSE**, the case statement moves into the next casement within it to test to see if it is TRUE or FALSE (shown by the **GREEN** circle).

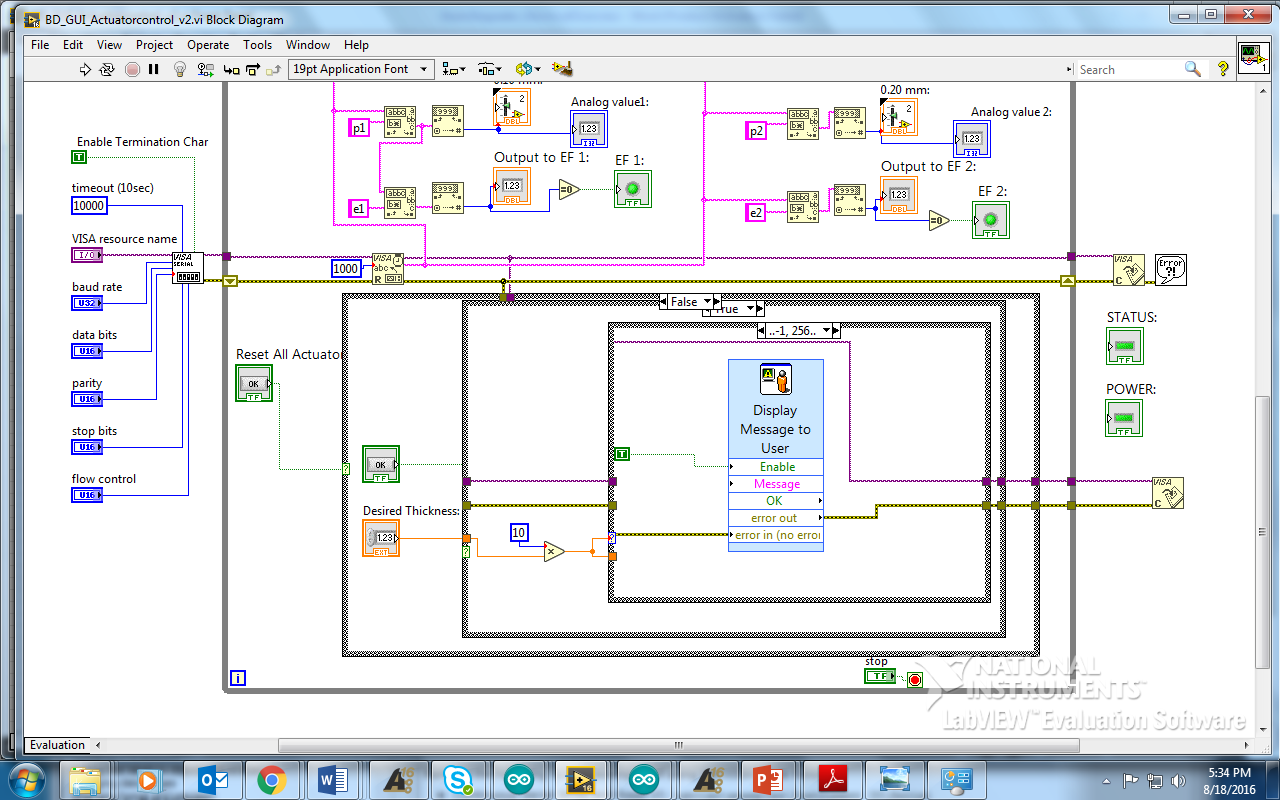
If this GREEN Boolean statement = TRUE, then we can see that the stuff circled in **BLUE** will be sent to the 3rd Boolean statement (shown by the **PURPLE**).

The 2nd Boolean statement circled in GREEN, has an input of an “**OK**” button and a “**Desired Thickness**” input (circled in YELLOW). This Boolean statement is controlled by the input of the “OK” button, if this button is CLICKED, then it equals TRUE, then it sends the “Desired Thickness” the user inputted in to the 3rd Boolean statement(**PURPLE**).

Within the **BLUE** circle, we can see that the “Desired Thickness” number gets multiplied by 10(REMEMBER that I did this because we don’t want any decimals to be inputted to the Arduino Serial Port because we are testing in binary from 0 to 255). Multiplying by 10 to the user input thickness ranges from (0.1mm to 25.5mm) allows us to change the values from 1 to 255.

The **PURPLE** circled Boolean statement looks a little different than the other two. So Boolean statements are pretty cool because they don’t just limit you to true and false possibilities. LabVIEW allows you to output a certain something if certain ranges of numbers, letters or words are inputted into the case statement. So in this PURPLE Boolean statement, I am saying that if any number is -1 or less or 256 or greater (represented in LabVIEW by ..-1, 256..) don’t ask me that’s just how you write it in LabVIEW. If you want more examples remember to right click and click HELP then look at more examples. This helped me a lot.

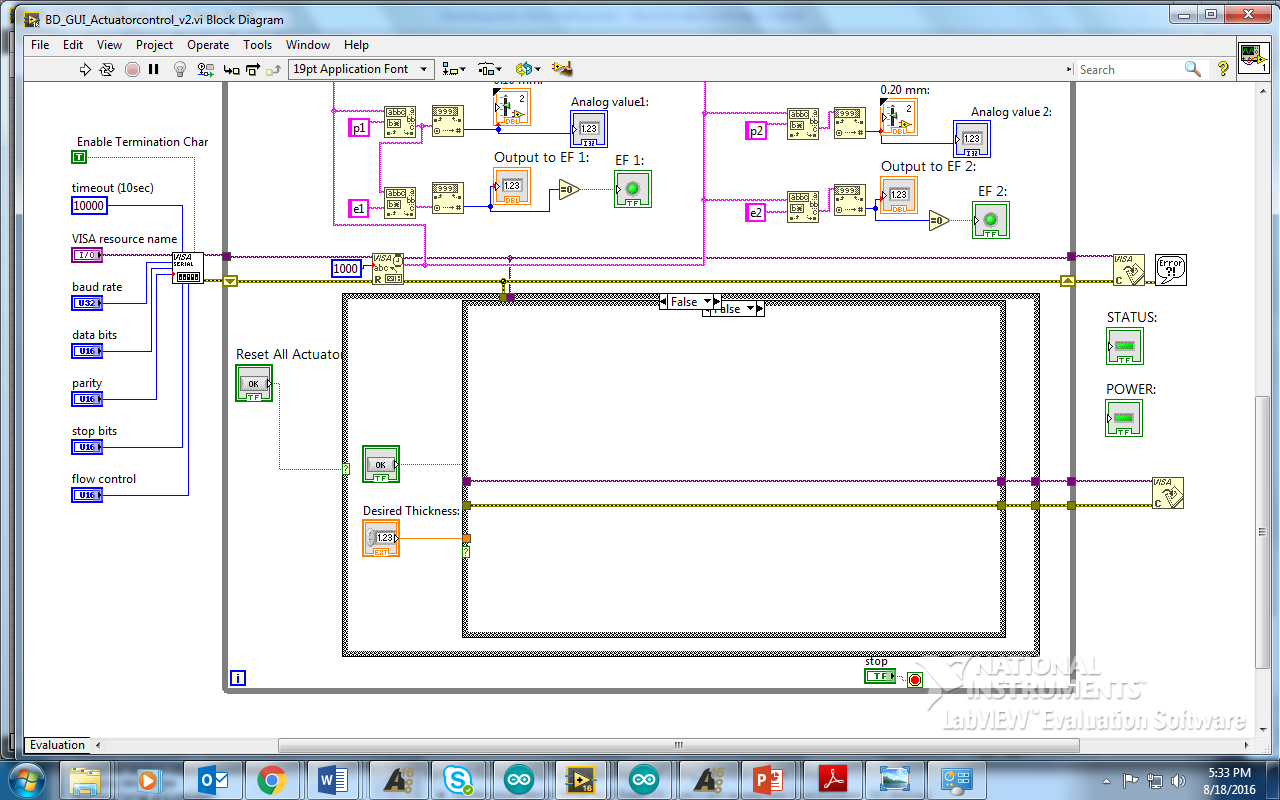
So if any number inputted into the “Desired Thickness” input and the “OK” button is pressed, and the number is outside of the range of 0 to 255, then an ERROR Message box(**ORANGE** circle) will pop up to the user, telling them to input a value that is within the correct range. Note that nothing gets written to the Arduino Serial port in this case.



**Figure 3**

Figure 4:

Now let us move on to what happens when the second Boolean statement = FALSE (**GREEN** circle), when the “OK” button is not pressed. In this case we can see all that stuff from before disappear. This is because when the “OK” button is not pressed on the front panel, nothing gets written to the Arduino Serial port. We want this because we only want a desired thickness to be sent when the user WANTS it to be sent! Which is done by pressing the “OK” button.

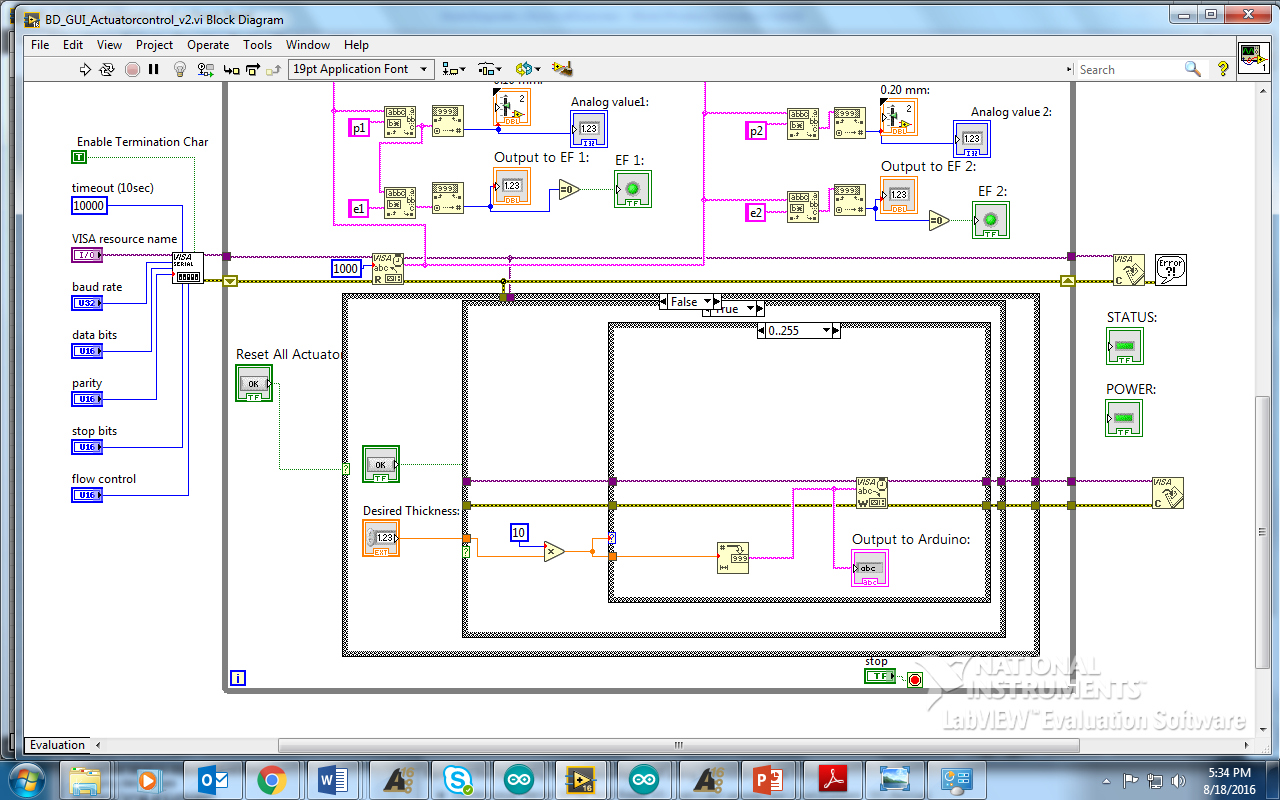


**Figure 4**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**FIGURE 5:** Alrighty, so now I’ll explain what happens when the “Desired Thickness” inputted is within the range of 0 to 255 and the “OK” button is pressed, which allows us to enter into the final 3rd Boolean case statement (**PURPLE** circle)!

So when this 3rd statement is within the desired ranges we want, then we WANT this number to be sent and WRITTEN to the Arduino Serial port. So first we multiply the desired thickness by 10, then that number gets converted to a “String” (**ORANGE**) so that it is readable in Arduino. The “Output to Arduino” block(**PINK**) is simply for means of testing to let us see what is being sent out to the Arduino Serial port. This output should read the desired thickness number multiplied by 10.



**Figure 5**

**FIGURE 6:** I’d like to apologize beforehand that I can’t input a picture of the entire block diagram on one screenshot. If you can figure that out props to you! So that’s why I had to show you images in little chunks at a time.

Below the image is showing you all the stuff on top of the case statements in the block diagram. So what each of these blocks are doing is basically it is OUTPUTTING the information READ from the Arduino Serial port and displaying that information to the “Position bar” and “Error Flag LED display” on the front panel.

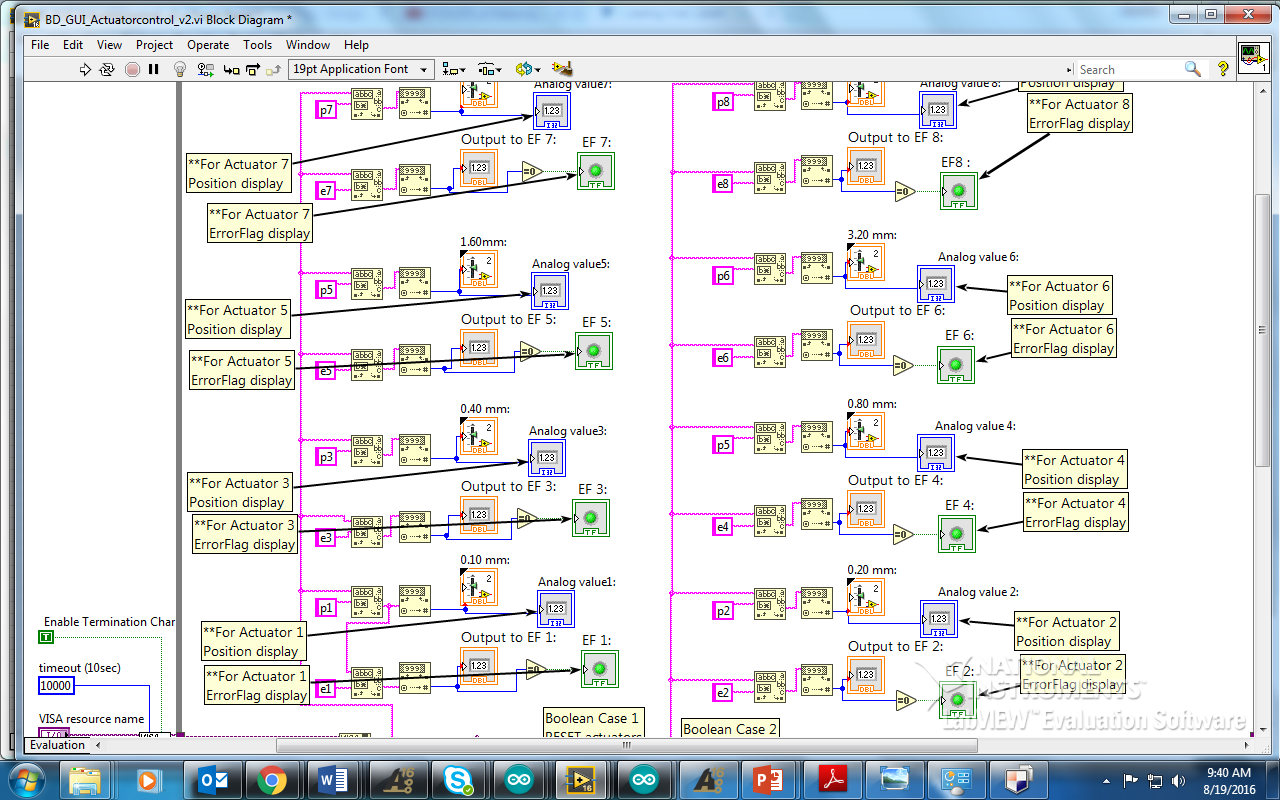
**FIGURE7**: This is the front panel and I circled things on the front panel and block diagram to show where it matches up.

The big RED circle, is everything for Actuator 1(0.10mm). I circled the same area on both front and back panels to show how it works. So within the RED circle there is a little ORANGE circle, this circle is shown on both figures and this is outputting the position of the actuator read from the Arduino to the LabVIEW GUI.

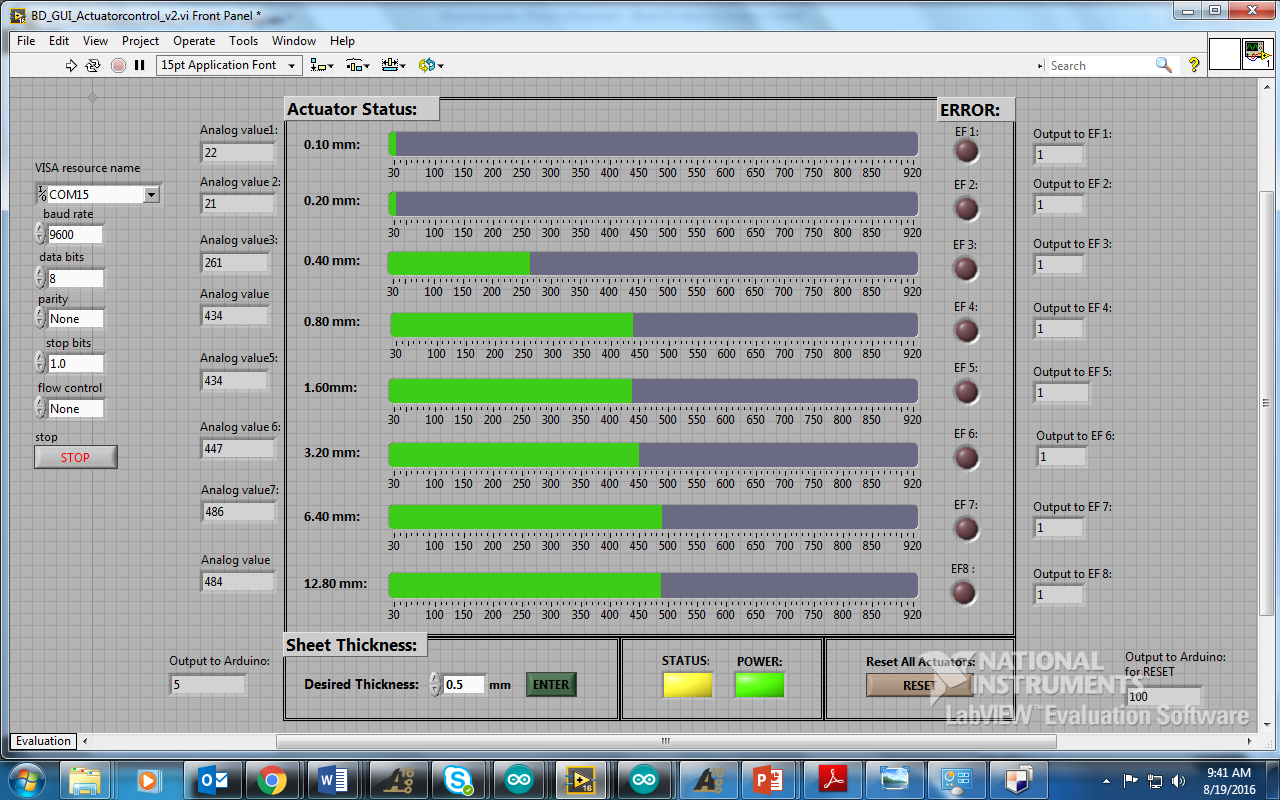
The GREEN circle is displaying the ErrorFlag for Actuator 1 as a circle LED.

The **PINK** circle is just tapping into that value that is being sent to the display and showing us a digital value which was for testing purposes. This is the same case with the YELLOW circle.

The big **BLUE** circle is just showing you another Actuator and how everything inside of that circle is connected to everything circled on the FRONT panel.



**Figure 6**



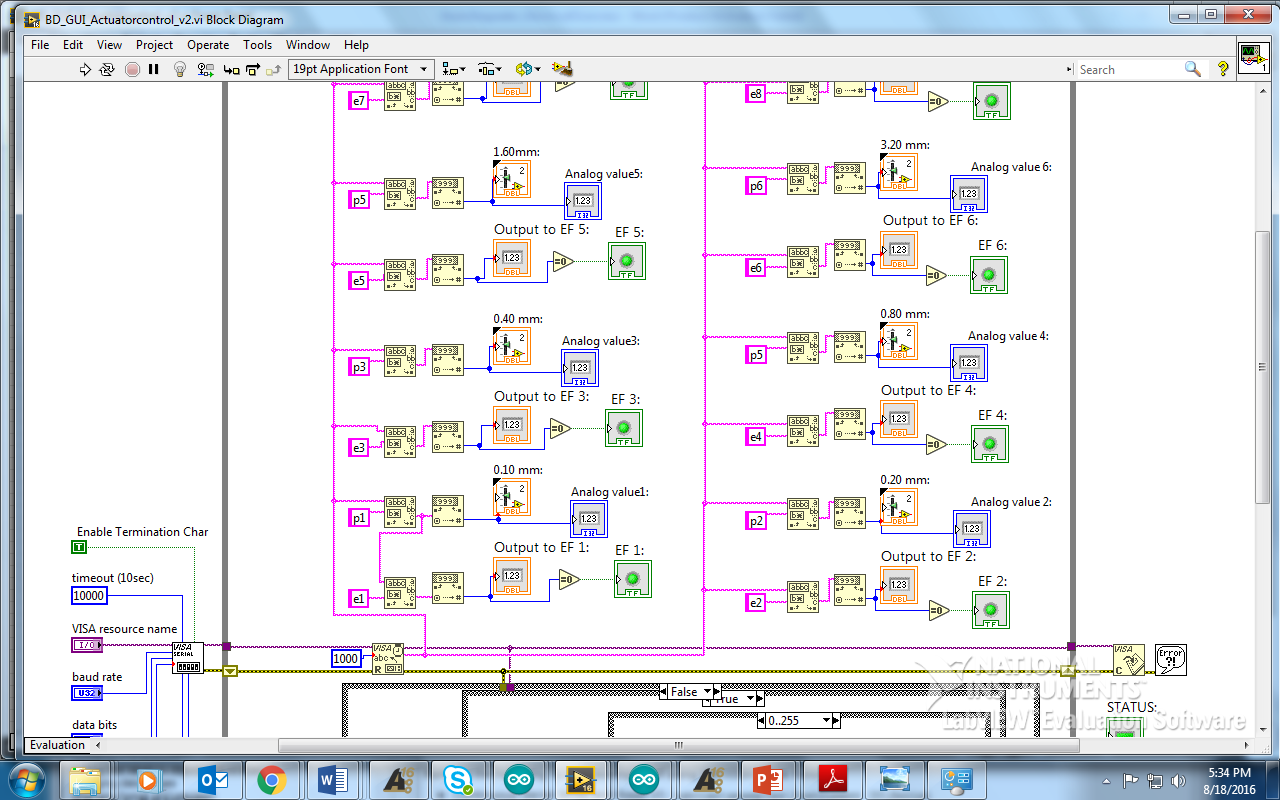
**Figure 7**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**FIGURE 8:**

So another crucial part of this design is that you see a “p1” (**RED circle**) and a “e1” (**RED circle)** are being inserted into a block for Actuator1 and they vary with each Actuator they are controlling. The block that it is being inserted is called the “Match Pattern Function” (**GREEN circle**).

This “Match Pattern Function” allows us to read the incoming String from the Arduino and separate out everything that has a “p1” in front of it, after that data is filtered out, it sends the rest of the string through the next Match Pattern Function (BLUE) to filter out everything that has a “e1” in front of it and sends this value to the error flag LED output. We are saying that if the value = 0, that means that there is an ERROR and the LED will light up RED.



**Figure 8**

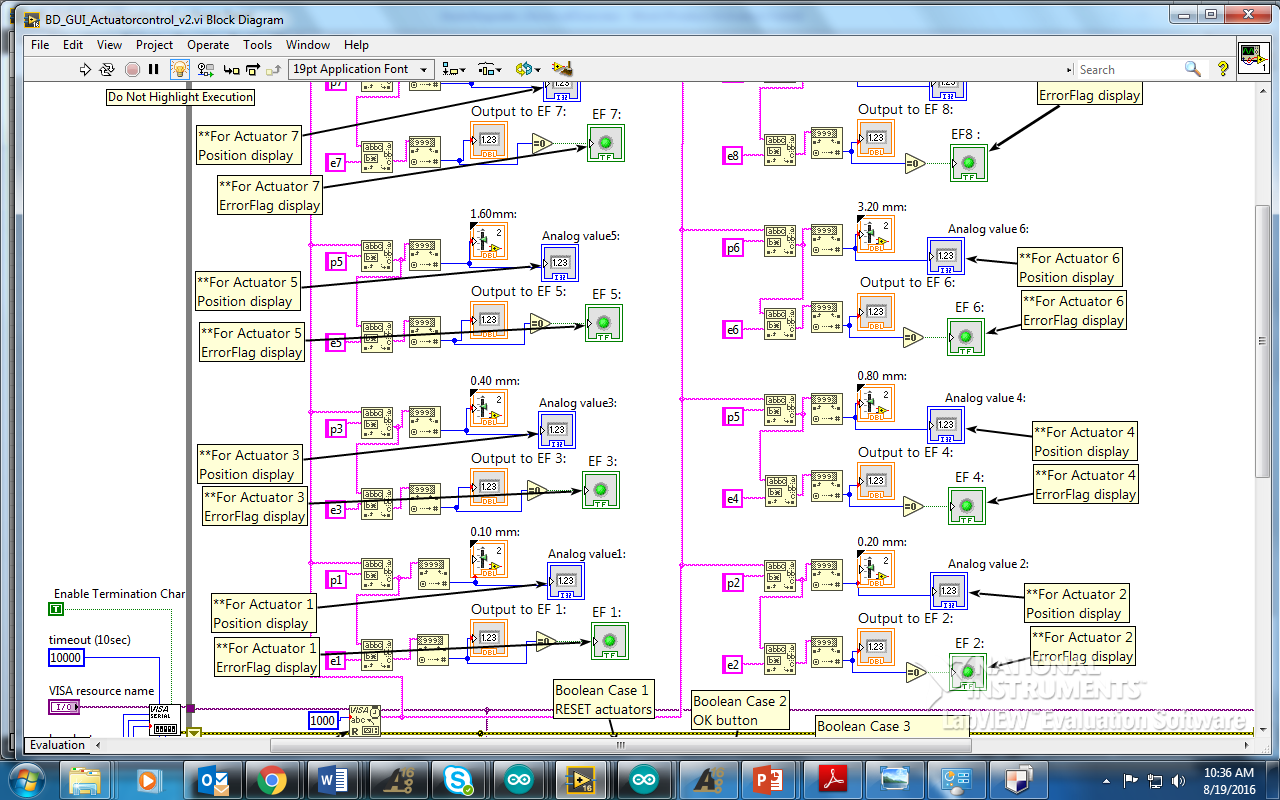
\*\***EXAMPLE**\*\*Here’s an example of what I just explained, hope it helps\*\*

The output from the Arduino will send a code like this to LabVIEW. “p1125e11p234e21p3900e30….” And it will go on all the way to actuator 8.

Let me explain what the above code means, from that code we can see that position 1 of actuator 1 is 125, no error, position of actuator 2 is 34, with no error, position of actuator 3 is 900 with error.

“**p1** 125 **e1** 1**p2** 34 **e2** 1 **p3** 900 **e3** 0….”

Hopefully separating it out this way helps you understand it. The “p1” in front tells you which Actuator it is associated with, the e1 tells you which error it is associated with. The number displayed after the variable indicator tells you which actuator the data should be displayed to. This goes on to all eight actuators and sent instantaneously.

**\*\*Figure 9:** Note you can also press the **LIGHTBULB** shown below to slow down the whole process of what is happening in LabVIEW. But be sure to turn it off again when you want it to run at normal speed. Otherwise it will run extremely slowly.

**Figure 9**

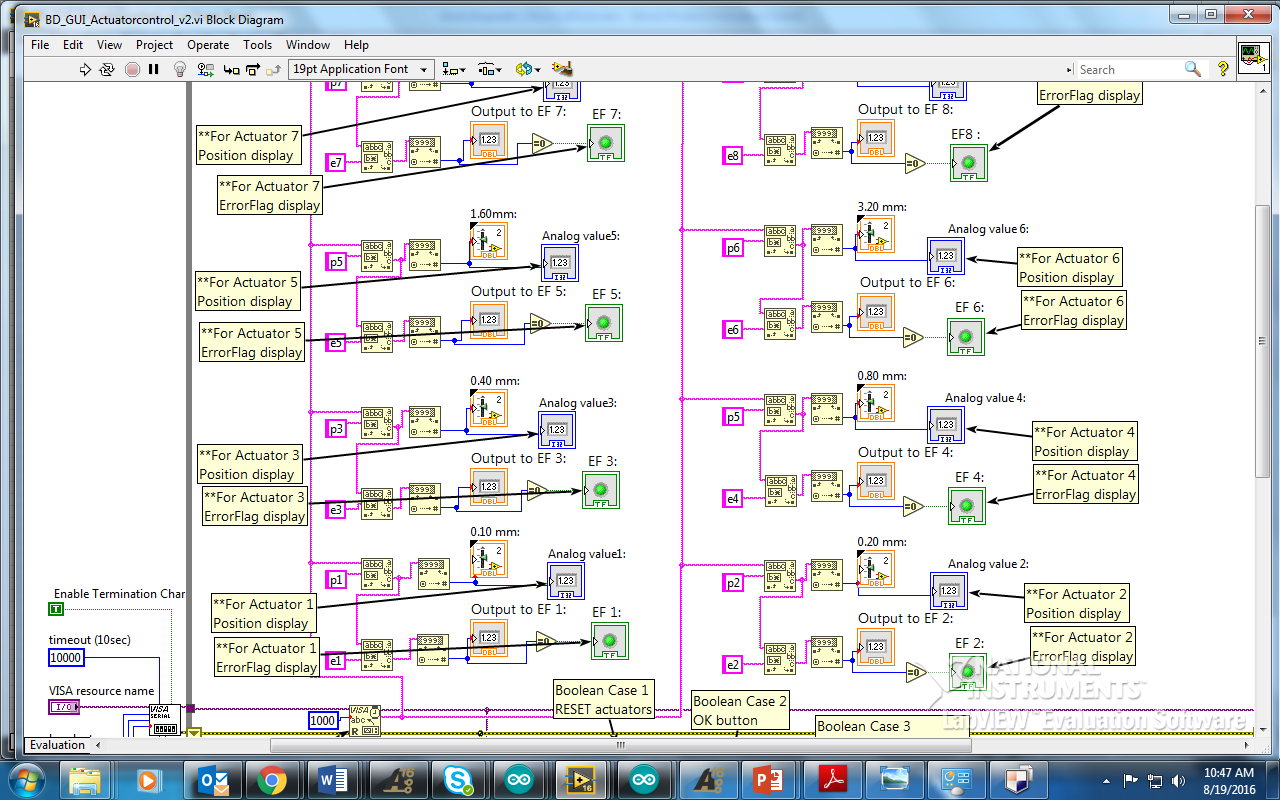
**\*\*TO RUN LABVIEW: \*\*\***

**FIGURE 10:** In order to Run LabVIEW, you want to have initially uploaded the Arduino code to the Arduino DUE board. Once that is uploaded, we can run LabVIEW.

\*\*NOTE: The Arduino Code should be titled **“BD\_8\_Acc\_control\_v3”**

To do so, press the little white arrow shown below. (**PINK**)

Also for good gracious sake remember to turn off the Lightbulb (BLUE) so that it is no longer yellow so it doesn’t take like 5 minutes to run the whole program. (This took us a few days to figure out -\_\_-)



Figure

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**FIGURE 11:** Once LabVIEW is running you will be able to see the current positions of the Actuators coming from the Arduino analog input ports and the error flag inputs. The Slide and EF LED should be displaying accurate results and should look something like this.



**Figure 11**

In this case it is reading accurate results of the 2 actuators we have set up. Both of the actuators are in the retracted position, hence the 22 and 23 analog value reading. The error flag is OFF (set at a HIGH), so no errors are detected.

To **STOP** the LabVIEW program press the RED circle shown above (**RED**).