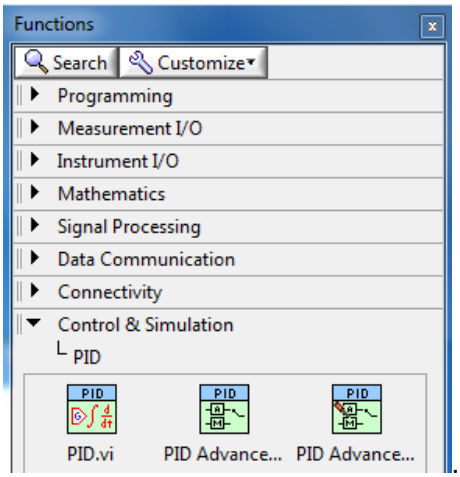
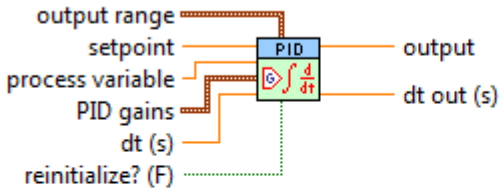
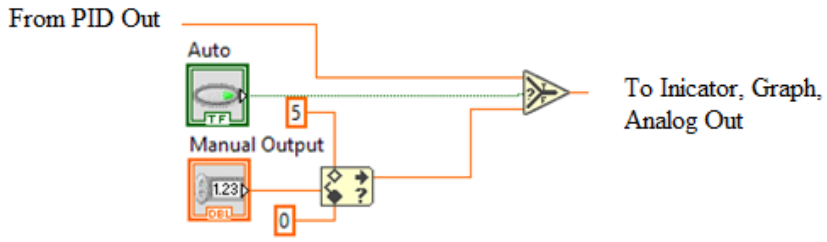
End of Semester Project
Part 2
ver 2019-1.5

Objective	To gain firsthand experience with PID Control
Background	See notes from classes on PID and Video links from class website
Procedure	In LabVIEW we will create a PID control program. Starting from Lab 9a, we will add the PID Control. Now goto the Functions Menu | Control & Simulation | PID and select the first Block (PID.vi) from the menu and place it in your VI.  Below is a help for the block: **NI_PID_pid.lvlib:PID.vi**  Implements a PID controller using a PID algorithm for simple PID applications or high speed control applications that require an efficient algorithm. The PID algorithm features control output range limiting with integrator anti-windup and bumpless controller output for PID gain changes. Use the DBL instance of this VI to implement a single control loop. Use the DBL Array instance to implement parallel multi-loop control.

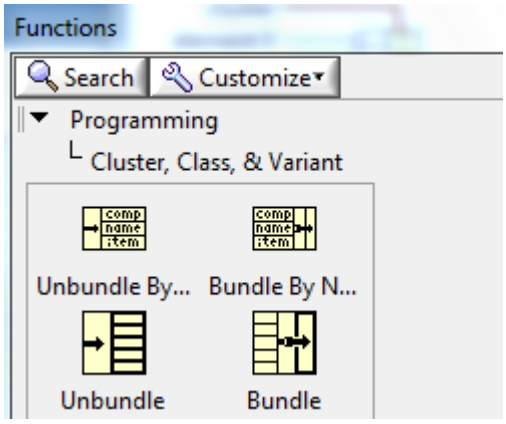
You want your Process Variable (PV) to be your speed in RPM and your output to go to the data port of the analog write. Create Controls for Setpoint (SP), PID Gains and dt(s). Create Constant for Output Range and make Output High 3 and Output Low 0. You will want indicators for PV and Output. Without running the VI, set dt(s) on the front panel to 1 and do a Edit | Make Selected Values Default.

Now we will add the ability to do Automatic or Manual Control: Add the following:



When AUTO is on, the PID will be controlling the output, when off, the manual output (in volts) will be placed on the Analog Output (allowing for manual control).

Create a Graph with three lines: the first being your PV, the second should be the SP and the last being the output. To do this graph, place a waveform chart on the VI front panel. Now go to the Block diagram, the waveform chart should be there. But to bring in three data points to the graph you need to use a "BUNDLE" block to merge the three signals. This can be found on the menu shown below:

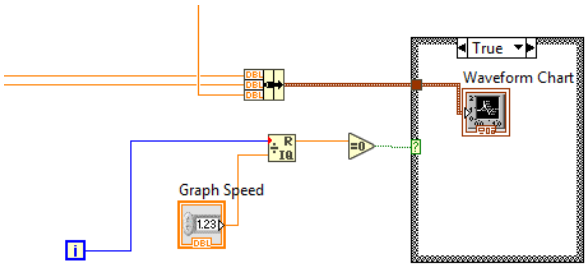


(Note – use Bundle, not Bundle by name, for this).

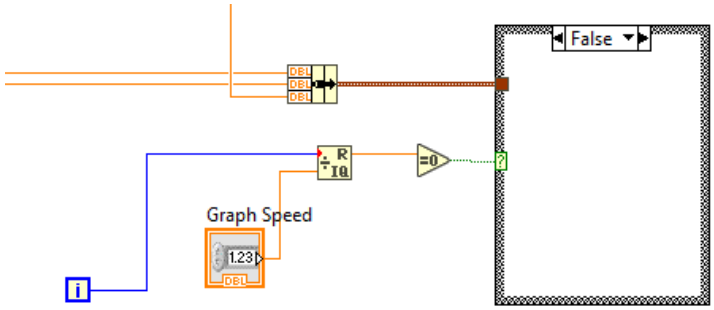
It should look as follows:

You will need to put the cursor on the bundle block and right click. Two blue dots (as shown above) will appear. Move the cursor over the lower dot, until the cursor changes to up/down arrow. Then hold the left mouse button and drag down until one more box appears (going from 2 to 3 inputs). It should now look like:

It would be beneficial if you could adjust the time the graphs update and make it slower than the time it takes to execute the PID code. To do this use the following code (placing the two Waveform Charts created above in a Case Structure (found next to the While Loop in the Function menu under Structures)



What it does is it takes the loop iteration count (ie the number of times the loop has executed) and divides it by the “Graph Speed” if the REMAINDER is zero it will then update the graph (TRUE Condition). The FALSE Condition is left blank



You will want to have everything in a While Loop. You will want to loop to have a 10ms wait timer within the loop.

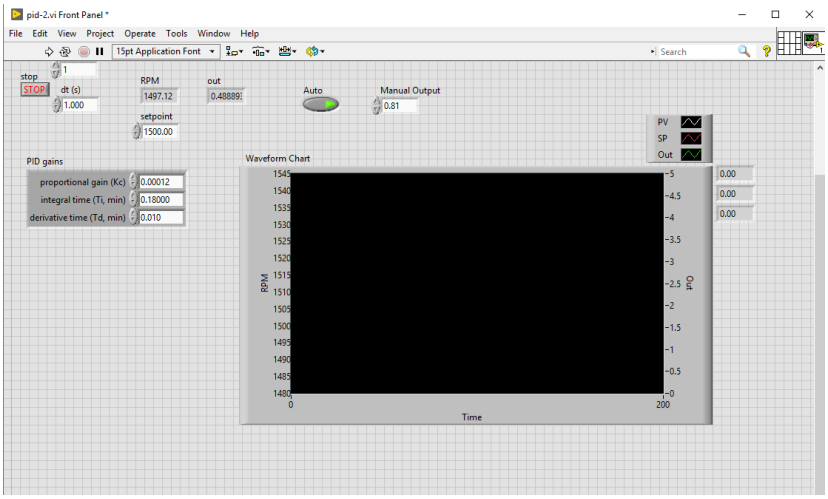
Lastly go to the front panel. First double right click on any value on the Y scale. Left click and select Duplicate Scale. Then Right Click on the Waveform Chart and select Properties. You will need to make the following changes:

In Appearance Tab	In Scales Tab	In Plots Tab

• Set plots shown to 3	• Select Time (X-axis)
	  ○ Uncheck Autoscale and set Min to 0 Max to 200)
	• Select 1st Y scale
	  ○ Name it RPM
	  ○ Make sure it is in Autoscale
	• Select 2nd Y scale
	  ○ Name it OUT
	  ○ Unselect Autoscale and set Min to 0 and Max to 5
		• Use the top pull down to select each channel. Use the 2nd box (NAME) to rename to PV, SP, OUT respectively

Press OK to exit the box. Lastly Click on the OUT Y scale and select Swap Sides.

The display should look something like the following:



Now you will need to tune the loop (see next lab for procedure)