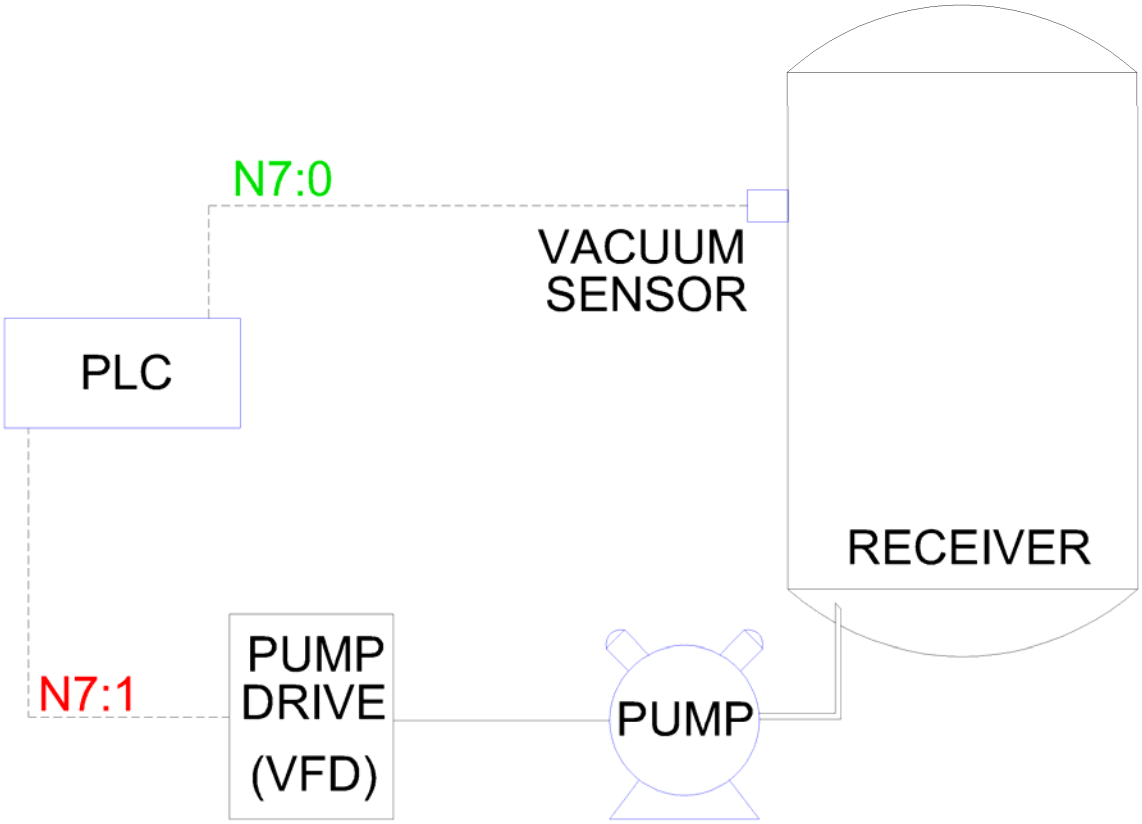
Project 8

PROCESS:



SUMMARY:

We need to maintain a vacuum in our system in spite of rapidly changing environmental conditions and to make matters worse - we can't use a PID to do it! So, we need to write our own logic which will speed the pump up or slow it down as necessary in order to get the vacuum input to attain and maintain the desired setpoint. (Sensor range 0-30 in/Hg, PID module scaling, setpoint = 15)

What we want to create is a program that compares our measured vacuum with our setpoint every 10 seconds (which is too slow for real life, but great for testing on Emulate). If we are within 4% of the setpoint, we don't want to change anything. If we are between 4-15%, we want to add or subtract a little bit (20) from our pump speed. If we are between 15-30%, we want to make an even larger adjustment (60). If we are greater than 30%, we want to make our largest adjustment yet (100).

And remember – a LOWER inHg value means MORE vacuum! (Very important!)

IO / ASSIGNED MEMORY:

N7:0 - Vacuum sensor in
N7:1 - Vacuum pump out
N7:2 - Vacuum setpoint
N7:3 – Measured vacuum

TEST CRITERIA:

First, run your program on Emulate. Set N7:0 = 8192 and N7:2 = 15. N7:3 should be right around 15. Now set N7:1 = 6000. N7:1 should not be changing (watch it for more than 10 seconds to be sure).

Next, set N7:0 = 8750. N7:1 should begin to increase by 20 every ten seconds. Now set N7:0 = 7650. N7:1 should begin to decrease by 20 every ten seconds.

To test the next range, set N7:0 = 10000. N7:1 should begin to increase by 60 every ten seconds. Now set N7:0 = 6200. N7:1 should begin to decrease by 60 every ten seconds.

Last bit - set N7:0 = 13000. N7:1 should begin to increase by 100 every ten seconds. Now set N7:0 = 4000. N7:1 should begin to decrease by 100 every ten seconds.

NOTES:

After the last project, this one will be a walk in the park. Of course the park is infested with grizzly bears, but you wouldn't expect anything less from one of my projects by now. There are a LOT of ways to approach this problem. There isn't a "right" way, and there really aren't "wrong" ways either. There are only ways that work and ways that don't work. The goal is to maintain the vacuum. We can assume the rest of the system outside of what we have on our diagram is working against us and creating wild fluctuations making it all very unruly. All we have to do… is our best.

There is another bummer with this one: testing it on emulate will be very hard. By hard, I mean impossible. In the real world, logic like this would need to be tweaked on-the-fly in order to "tune" the program according to the actual system and conditions its controlling. Not to mention, our ranges and time base for these adjustments would be TERRIBLE for a vacuum application!!!!!!! But that being said, the logic is fine. To use this in real life, we'd just have to change our values.

Speaking of trolling… sorry for this one. You can thank me later. ☺