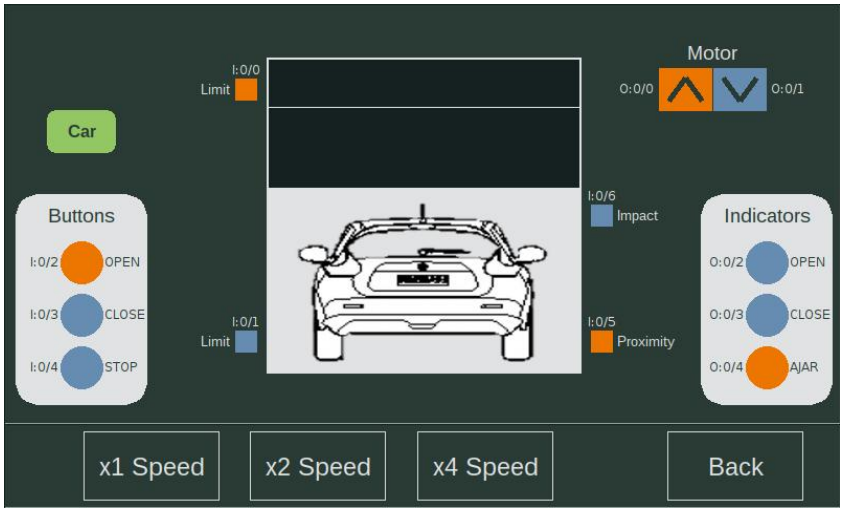
Programmable Logic Controllers (PLCs)
Lab #1 Intro to Ladder Logic and Garage Door

Intro:

During this lab you will learn about programming PLCs using ladder logic. In order to demonstrate this, you will write two different ladder logic applications. First, you will make a simple modification to the IO_Test program you used in the previous lab. Second, you will write a fully functioning garage door controller.

IO Test Modification:

- Read this tutorial on how ladder logic works with PLCs. https://www.allaboutcircuits.com/textbook/digital/chpt-6/programmable-logic-controllers-plc/
- Take the IO_Test program that you flashed to the PLC in the previous lab and make some modifications. The new IO_Test program should have the following features (It is recommended to copy the existing program before making modifications)
    - If analog input I:0.4 is above 5.0V, then digital output O:0/5 should be high.
    - If digital inputs I:0/7, I:0/8 and I:0/9 are all high, then analog output O:1.3 should be set to 7.0V.
    - The PLC sees the analog inputs and outputs as integers not as floating point numbers. See the existing ladder logic for an example of how to interpret the values.
    - NOTE: It is best practice to only read inputs or write outputs in one place. Inputs can change any time causing conflicting values and writing to the same output in multiple places can cause the output to misbehave.



Garage Door

- In this section, you will practice digital logic by writing code to control a garage door.
- **Make a copy of the IO_Test project file and use it as a baseline to get started on this project.**
- Write a ladder logic program that fulfills the following criteria:
    - The motor must never be driven up and down at the same time.
    - The PLC must activate the appropriate indicator for the current state of the system. Only one indicator should be active at a time.
    - The “OPEN” and “CLOSE” buttons should work when they are activated.
    - If the “OPEN” and “CLOSE” buttons are both active the door should stop.
    - When the “STOP” button is pressed, the door should stop.
    - When the door reaches a limit switch it should stop.
    - When the “Proximity” sensor is triggered, the door should not be able to move down, but may continue to move up. This is to avoid crashing into a car.
    - If the “Impact” sensor is triggered, the motor should stop completely.
    - The inputs and outputs are labeled just like in the IO_Test program.

- If time permits, add logic to the program to enforce one final criteria.
  <u>The motor must not reverse direction until it has come to a complete stop first.</u>

  Since no feedback is available, a timer may be used to delay 2 seconds before reversing direction. This should provide enough time for the door to stop fully.

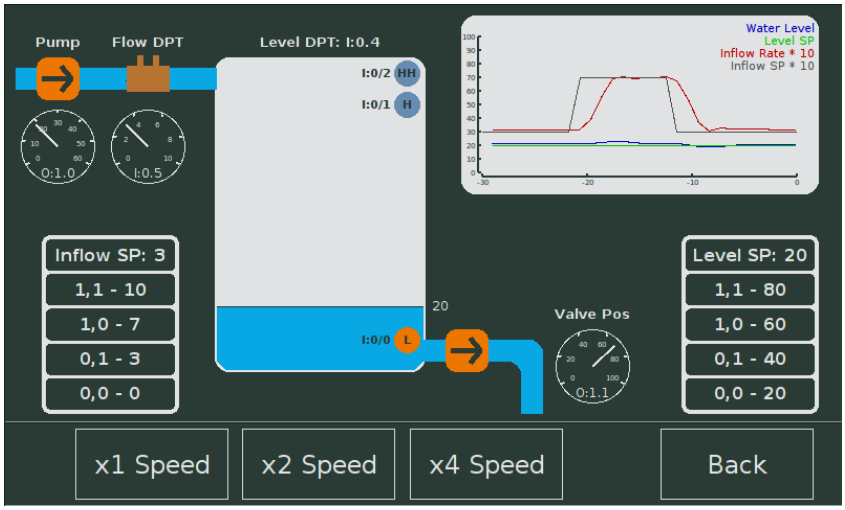
  Note: See this tutorial for an example of using timers in RSLogix 500:
  https://www.youtube.com/watch?v=kONI5e9f6Qk

PID Tuning

- PID controllers are extremely common tools for controlling analog physical processes. For example, quadcopters use 3 or more PIDs to vary the speed of the 4 propellers to achieve specific yaw, pitch and roll values and stay in the air.
- PID stands for “proportional–integral–derivative” and has three primary tuning parameters: P, I and D.
- Read the summary and fundamental operation of PID controllers here to get a general idea of how they work:
  https://en.wikipedia.org/wiki/PID_controller

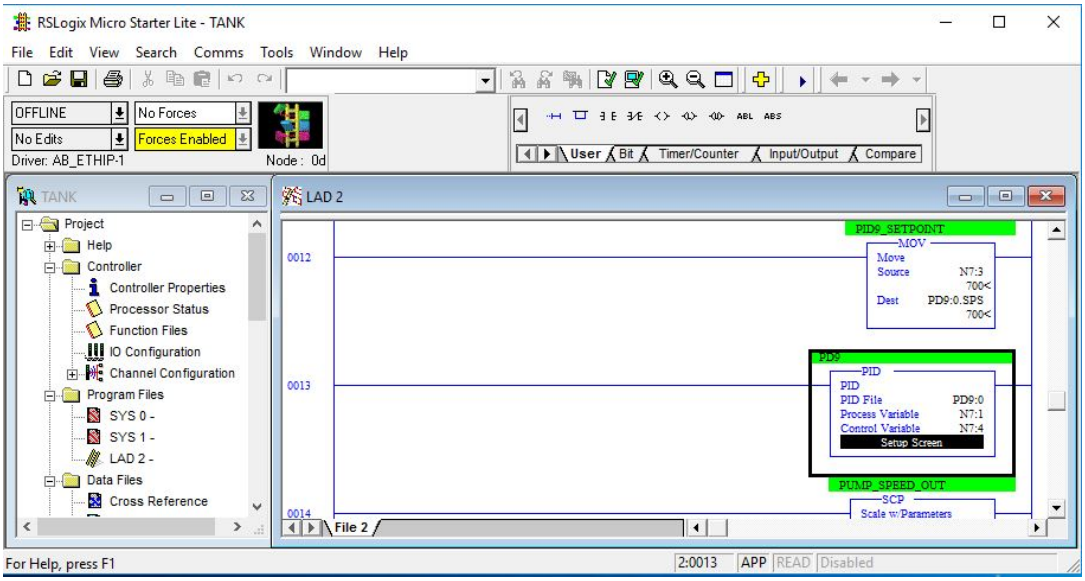
- **PIDs funcion by measuring a “Process Variable” (PV), calculating error based on a “Setpoint” (SP) and using the error and PID parameters to calculate a “Control Variable” (CV).**

- In this lab, you will use the Water Tank simulation to experiment with PIDs. The water tank uses two PID controllers:
  - PID 1 - PD9:
    - PV - Inflow rate
    - CV - Pump speed
  - PID 2 - PD10:
    - PV - Water level
    - CV - Valve position

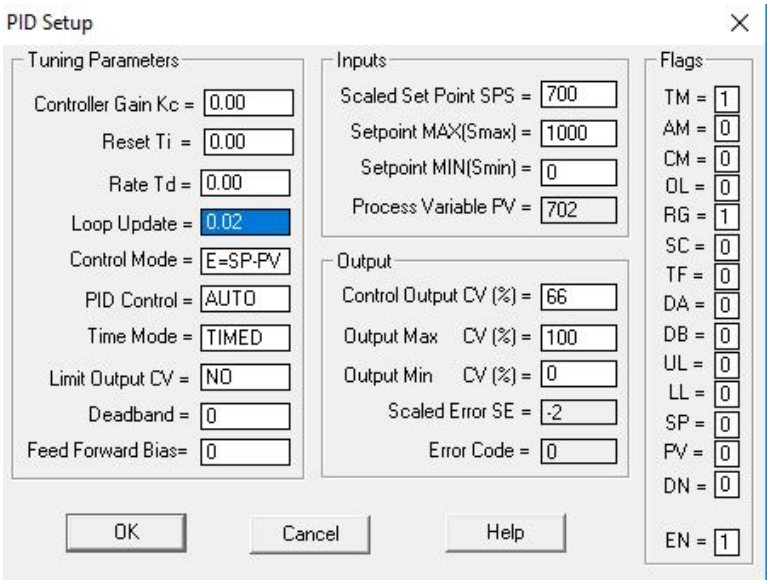


Getting Started:

1. Open the Tank project file in RSLogix.
2. Find the PID block for PD9:0.

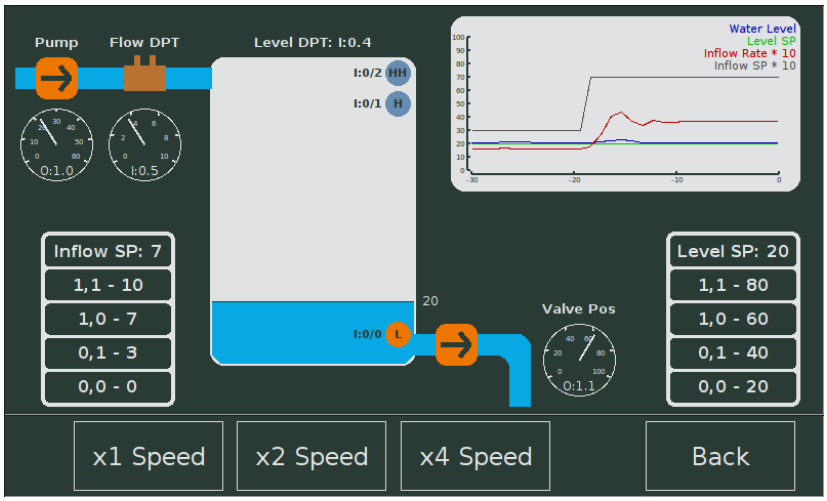


3. Open the PID “Setup Screen” and verify that it looks like this. All of the values under Tuning Parameters and most of the Flags should be the same. SPS and CV may be different, but the max and min values should be the same.



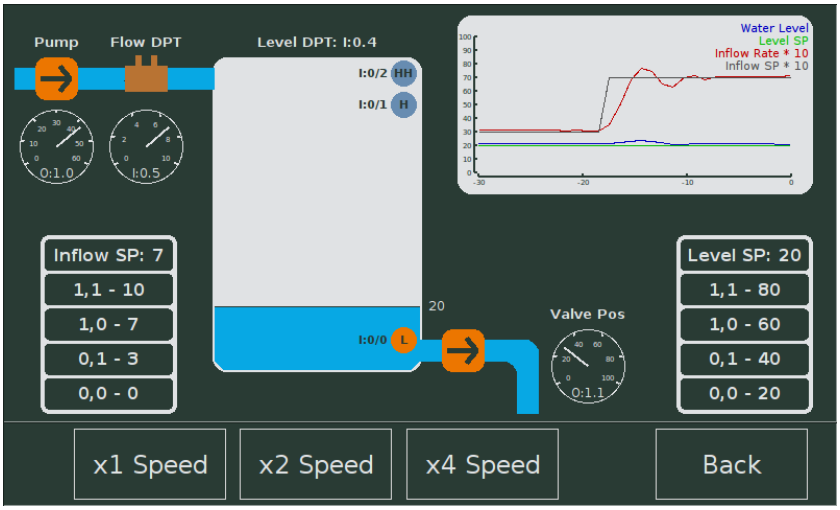
Tuning:

1. PID tuning is the process of selecting the three tuning parameters to achieve optimal results. PID tuning is 1% science, 9% art and 90% random guessing.
2. You will tune PID 1 to accurately control the pump to achieve a specific inflow rate.
3. **Start by increasing P while keeping I and D at 0.0**
4. Notice how the process responds to SP changes.



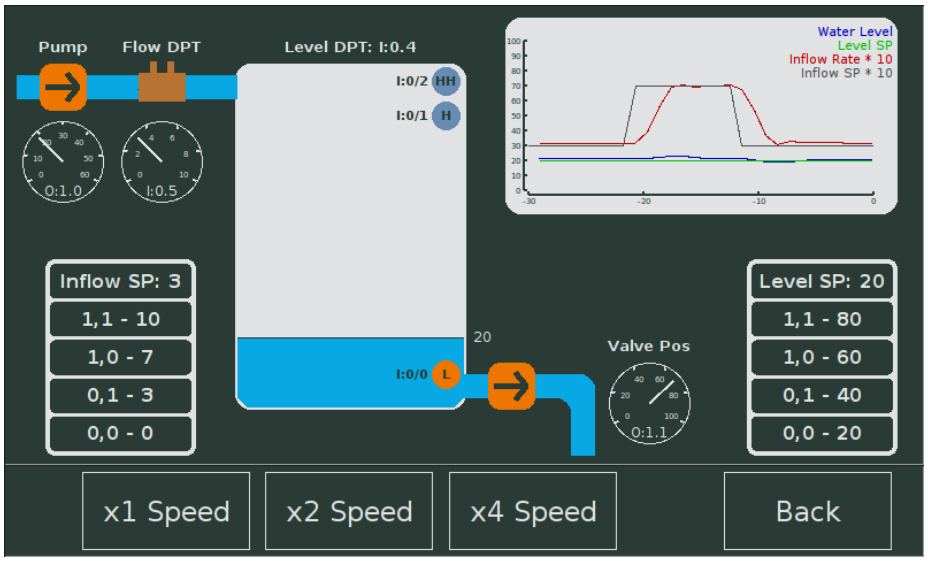
P only

5. **Next, add I. In this program, the smaller the I-value, the larger the impact. Start with an I of 0.10 and decrease it to see the impact.**



P & I

6. **Lastly, add in some D to reduce the overshoot. Note how the system reacts when you have too much D vs too little D. Also note how the CV behaves as you change D.**



P & I & D