critique

- Hardware

  We are using EL terminals. If not using good panels this is not the best setup.

  - pressure sensor

    When it comes to sensors, 0-10v is alright. It's nothing fancy but it gets the job done. It's common, straight forward, easy to troubleshoot, and nearly every industrial controller will accept this type of signal.

    There are some downsides. "Noisy" devices such as relays, motors, and power supplies can induce voltages onto signal lines that can degrade the 0-10v sensor signal. Also, a 0-10v signal is susceptible to voltage drops caused by wire resistance, especially over long cable runs.

  - hydraulic system

    I would imagine that this type of hydraulic system would be difficult to control and not much ability to control is given. The operation would be less than optimal. The pump would probably cycle many times per day depending on the hydraulic system (bleed off due to lack of check valves?) and the control loop type. If using PID loop it will more likely get you thru day without many interruptions but I can't imagine the pump's life would be long and if using bang bang (like I did for this project) you'd probably run into chasing the right pressure with some need to tune the hysteresis that will probably change as the pump wears.

  - safety

    There's no mention of safety. I would feel terrible if someone was injured with something built me (I would feel terrible about anyone getting injured, period) my safety is a concern too. Don't skip out on safety, it could be your life.

- Code

  - my code

    Sure it's built with a modular system in mind but it was put together really quickly. With more time, more thought would be put into the system's architecture to greatly improve it. Could it be used in production with a bit more tweaking... probably, but I wouldn't recommend. I tried iXlinker for the first time, you did say fun, and it works good except there's little control as to what it produces (global vs local struct and where it places it). It is an open source project so maybe I should volunteer some of my free time to improve it! I do like the idea of linking absolutely everything and only buffering what you need that way variable link issues that are susceptible to twincat would be minimized. Check out iXlinker here to learn more and here's a quick video also.

improvements

- Hardware

  I recommend using EP blocks if no panels are available. They are better suited for harsh environments.

  - pressure sensor

    I would use a 4..20mA sensor. It offers increased immunity to both electrical interference and signal loss over long cable runs. A 4..20mA signal provides inherent error condition detection since the signal, even at its lowest value, is still active. At its minimum ("zero") position the sensor will still provide a 4mA signal, if the signal ever goes to 0mA, something isn't right. For a 0-10v, 0v could mean it's at "zero" position or the sensor quit on us.

  - hydraulic system

    I would recommend using a hydraulic power unit (HPU) and proportional valves (Nice D03 valves come to mind) but I'm no hydraulics expert! I would imagine an HPU would handle multiple cells within reach and proportional valves would allow for finer control (PID loops) of pressure for fixture clamps (or whatever needs hydro juice).

  - safety

    Beckhoff makes nice safety hardware. Buy what you can afford. I like safe cells, safety is no joke!

  - code

    With a wee bit more time I know I can make a better Twincat magic. Just for fun I added a few bits: The Twincat repository is broken up into many folders. The source code is kept in a solution of its own and referenced in two variants 'Simulation' and 'V1_{00}'. The idea behind this is that when hardware definition and their links to software change due to supply chain issues or improvements to design (or whatever the reason) you keep things nicely in order and the source code remains as single entity. Changes to the repository are minimized and when making changes in a version control environment (git/github) things are less tightly coupled and are easier to make changes. A JSON file is written/read onto each Beckhoff IPC which holds information about its hardware and sections of code it should execute. The 'Simulation' holds no hardware so no more mucking around disabling/enabling hardware when switching between working offline and online. Even the HMI project (and if I was TcScopes) those are seperate projects too where versioning can be applied to them all. Variant Immunity (sort of)!

    It was my first time really digging this deep into Twincat Scope and by the end of creation I learned that there are way better methods of creating what I did like 'user controls' where you can parameterize the containers filled with widgets where you can replicate them infinitely with minimal workload. Miles better that Twincat Visualization and any other HMI package I've used to date. I will keep learning it on my free time for sure.

```
IF stConfig.bIsSimulation THEN
    // do some cool simulation stuff
ELSIF stConfig.bHasRobot THEN
    // do some cool robot things
END_IF
```

Project Startup

A window will pop up with pre-populated information click done and a file named MachineConfig.json will be generated at ’C:’ in case of error delete MachineConfig.json and restart the simulated PLC.

An error is produced on a timer that simulate a pressure drop. If the robot is in cycle it will error otherwise only the hydraulic system will. When in ’Auto’ and you press the button to go to ’Manual’ if the robot is in cycle it will wait until it’s finished before switching over. The signal is not incorrect at 2v… using a 0-10v to work on a 4..20mA terminal I would use a 500ohm resister to convert it to 4..20mA. We would lose some resolution as when sensor produces 4mA signal it is actually 2v. The robot cycles as if there’s a robot scheduler managing jobs for it so you’ll see it start and stop on its own while cell is in ’Auto’. For the pressure signal it rises quickly and drops slowly mimicking a slow leak.