Notes on Dust Collector System

This will add linear actuators to new custom-made blast gates and sensors to detect when a machine is turned on or off and control blast gates and the dust collector motor appropriately.

Gate/Motor Controller

This controller is implemented on a Raspberry pi zero, much like this one by adafruit. I've got plenty of suitable wall-plug 5V supplies available, so one of them will do for power. It's got more than enough GPIO for my needs. I'll need an external A/D multi-channel converter such as this PCF8591 for sensing current in the shop tools (saw, drill press and lathe power cords).

Components

I have on order from Amazon:

1. A linear actuator that travels 4 inches in a couple of seconds, and has a 4.5 lb push, which should be more than enough. See this information.
2. A 12V power supply for the actuator. This one can handle power needs for all 3 gates. See this information.
3. A current sensor module. See this information.
4. A 40 Amp AC solid state relay to control power to the dust collector motor. See this information.

Basic Logic

The controller measures the current sensor on each machine. If there is current flowing (that is, the machine is on) then the blast gate for that is opened. [Since the actuator has limit switches at each end, it should OK to keep telling it to open. Otherwise some sort of state machine is in order.] In addition, the relay for the dust collector motor is activated, turning on the dust collector. [This too is an idempotent operation.] If a sensor indicates that the machine is not running (no current) then the corresponding blast gate is closed [again, idempotent]. If no machines are running, the dust collector relay is turned off, and the dust collector stops.

There may be a need to introduce a short delay between starting to open the blast gate and turning on the dust collector, since the actuator needs a couple of seconds to work. This is easy to do if it becomes necessary, but I don't think it will – it takes the dust collector motor a couple of seconds to spin up anyways.

Input and Outputs

Inputs

The input from the current sensors is an analog voltage I can read with the multi-channel A/D.

Outputs

- Output to the actuators is just a 12-volt signal with polarity reversed to reverse the direction. Here's a link to a ChatGPT discussion about H-bridges to implement this. Might be able to use this device.
- Output to the solid-state relay is simple.
- I might want to add some LED outputs to show status, just for coolness. For example, green for open, red for closed, similar for the motor.

Testing

Just for coolness it would be neat to add a push-button that would initiate a sequence of events to verify correct operation. For example, one such sequence would be:

1. open gate 1, wait 5 seconds,
2. close gate 1, wait 5 seconds
3. same for gates 2 and 3
4. open all gates, wait for 5 seconds
5. close all gates, wait 5 seconds
6. open all gates, wait 5 seconds
7. turn on dust collector, wait 60 seconds
8. turn off dust collector
9. close all gates

… and so on.

Development methodology

Python is the language of choice here. ChatGPT can do a decent job of generating skeleton code given suitable prompts. I'll try to develop separate Python apps for checking out each component.

- Current sensor input: a program to read the current sensor at 1 Hz or so and print out the value.
- Linear actuator: a program to move the actuator all the way out, wait 3 seconds, then all the way back, wait 3 seconds, and repeat.
- Solid-state relay: turn an incandescent lamp on and off.
- Basic logic: a program that will exercise the logic of the controller by simulating inputs via a script. Doing this early on will make implementation of the logic easier.