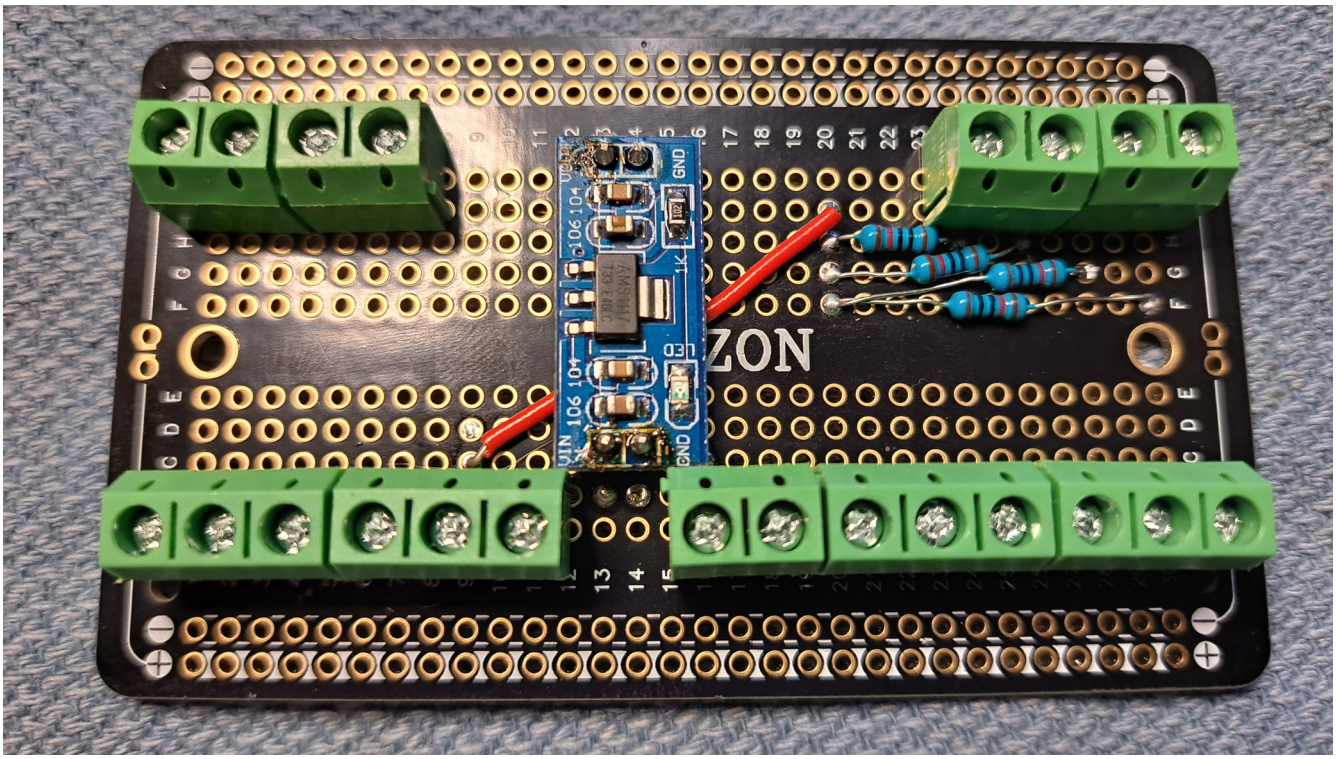
I almost always base my projects on PCB clones of breadboards to make planning easier. In rare cases, I will use plain old point-to-point boards if it doesn't involve a lot of jumper connections. Here's a link to the boards I used in the photos.

https://www.amazon.com/gp/product/B0BP28GYTV/

There are four PCBs that require soldering and that's the power break out board (pictured above), the valve control board, the heating stepper control board, and the interface board. Luckily, none of them involve anything real complicated.

Power Break Out Board



The power break out board receives 5 volts from the buck regulator provides a set of power distribution terminals. Six for 5 volts, four for 3.3 volts, eight grounds, and set of four pull up resistors for the GPIO pins to the servo valve limit switches.

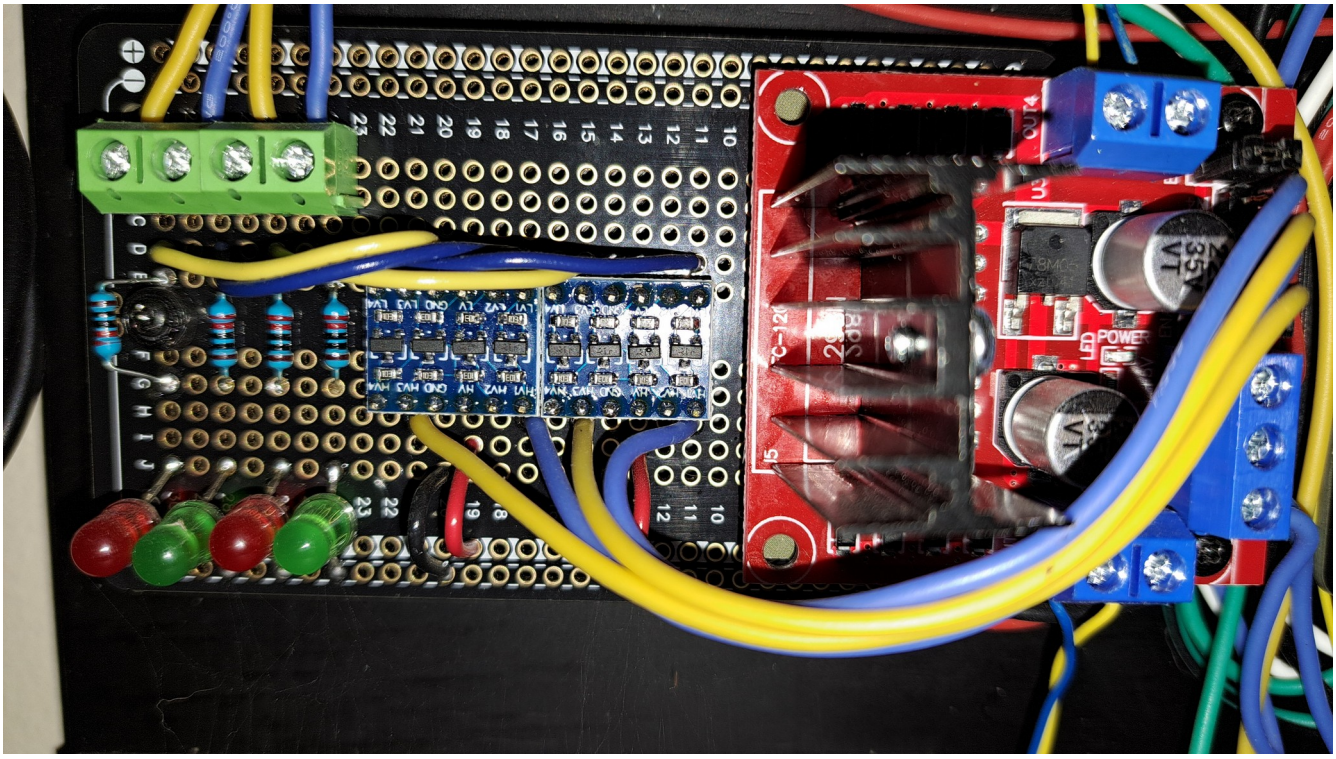
The pull up resistors are 10K and feed from the 5 volt supply. The 3.3 volt supply was an after thought, or I would have fed 4.7K resistors from there.

The little blue board in the middle is an AMS1117 3.3 volt 800 mA regulator that feeds from the 5 volt supply and outputs to the four terminals in the upper left. Below is a link to these regulators.

https://www.amazon.com/gp/product/B074FDLCLB/

Thus far, I am only using the 3.3 volt supply to power the heating stepper motor control board and a small speaker that I use for the Raspberry PI audio output.

Valve Control Board

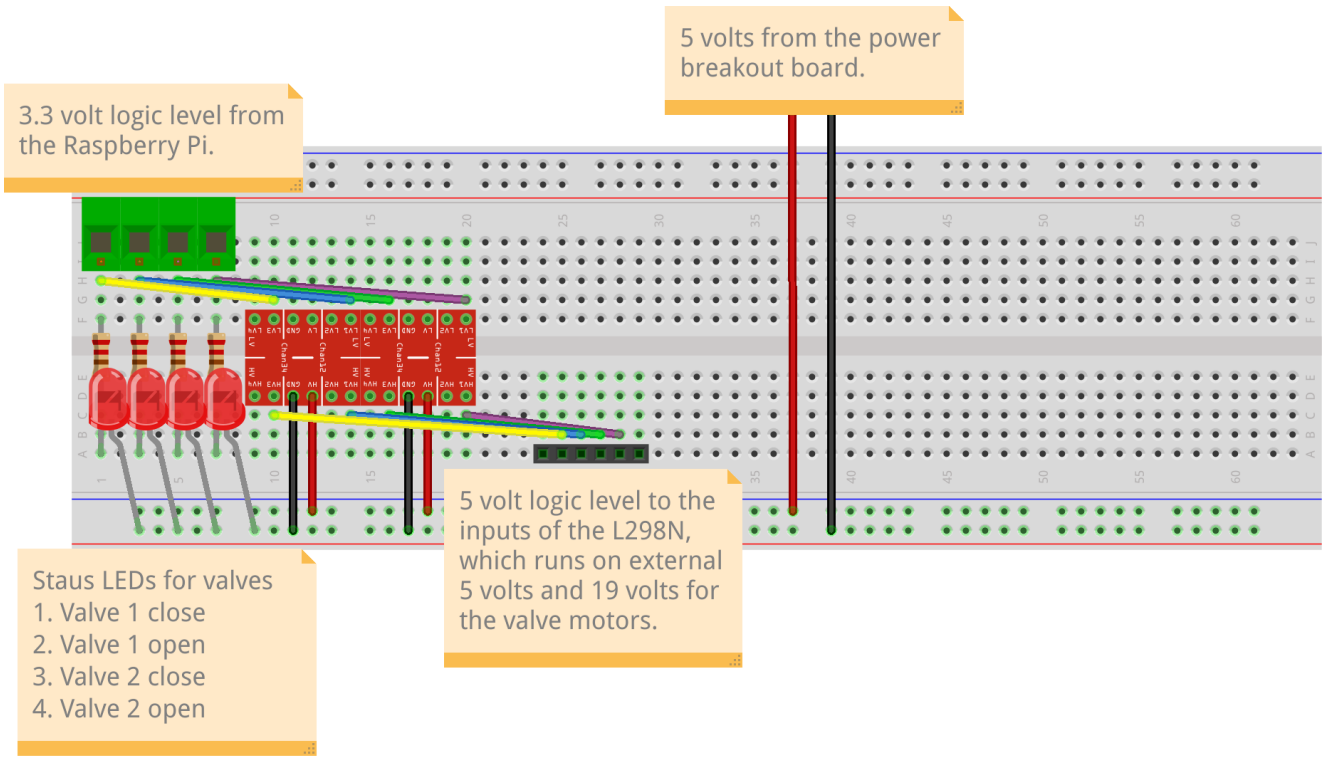


The condenser and dephlegmator valves don’t use stepper motors, but they have upper and lower limit switches. These use plain old DC motors and their position is merely tracked by how many milliseconds that they get power in either direction. This is the reason for the “Calibrate Valves” function of the smart still controller.

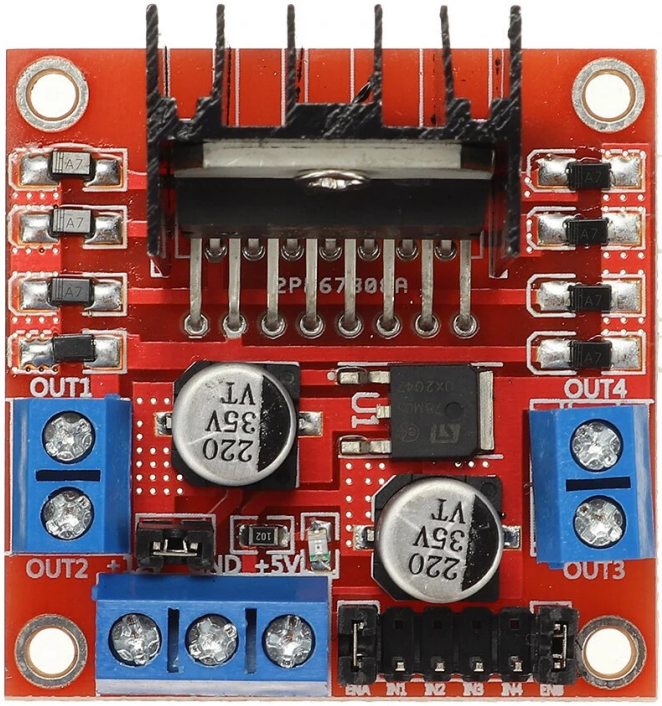
The motors are controlled by an L298N “H-Bridge” driver which is commonly used in robotics and RC cars. These drivers use 5 volt logic, so I2C logic level shifters are used in order for the 3.3 volt logic from the Raspberry PI GPIO pins to control them. This could also be done with a diode/resistor pair per pin, but I like simple.

If you look at the source code in valve.c you will see the GPIO pin assignments for the valve motors. Looking at the diagram below, the green header connects to the GPIO pins 25, 24, 21, and 20 from left to right.

The 4 pull-up resistors on the power break out board in the above connect to GPIO pins 23, 22, 19, and 18 which are for the limit switches in the valves. When these pins are pulled low by the limit switches, it tells the code in valve.c that the valve is at its full-open or full-closed position. I will cover the valve connections later.



Below, you can see the L298N dual H-Bridge driver board. The black strip of pins at the bottom are the logic inputs and match up with the 4 level shifted pins from the I2C logic level shifter outputs. I just soldered the wires directly to these pins instead of crimping a plug onto the wires.



As you look to the left of the logic pins, you will see the power connections for the driver board. Directly above those terminals is a jumper that you will want to remove. With the jumper in place, the driver can only be run by 12 volts since the onboard linear 5 volt regulator (LM7805) can’t handle any more than that.

The valve motors run on 9 to 24 volts and I use a 19 volt laptop charger brick to run everything. So I connect +19 volts to the left terminal, ground in the middle, and +5 volts to the last one. This allows the valves run at a higher rate of speed which is way more reliable than making them struggle along on 12 volts.

The operation of this driver is rather simple. If all the logic pins are low, nothing is output. If pin 1 is high, the outputs on the left put out 19 volts in one polarity, if 2 is high, then the polarity is reversed. Logic pins 3 and 4 do the same thing with the outputs on the right hand side. Only one logic input can be high at any time.

If the manufacturer changes the color coding of the wires, you will just need to do test runs to make sure that your valves are moving the correct direction. The wire colors are not standardized, different brands will likely use totally different colors. The yellow and blue wires on the outputs in the photo may not match yours at all.

Parts Needed

4 LEDs and 220 ohm resistors (if you want activity indicators)

(Your choice of source, these aren’t specialized parts)

Screw terminal strips for PCBs

https://www.amazon.com/gp/product/B088LSS14J/

I2C logic level shifters

https://www.amazon.com/gp/product/B07F7W91LC/

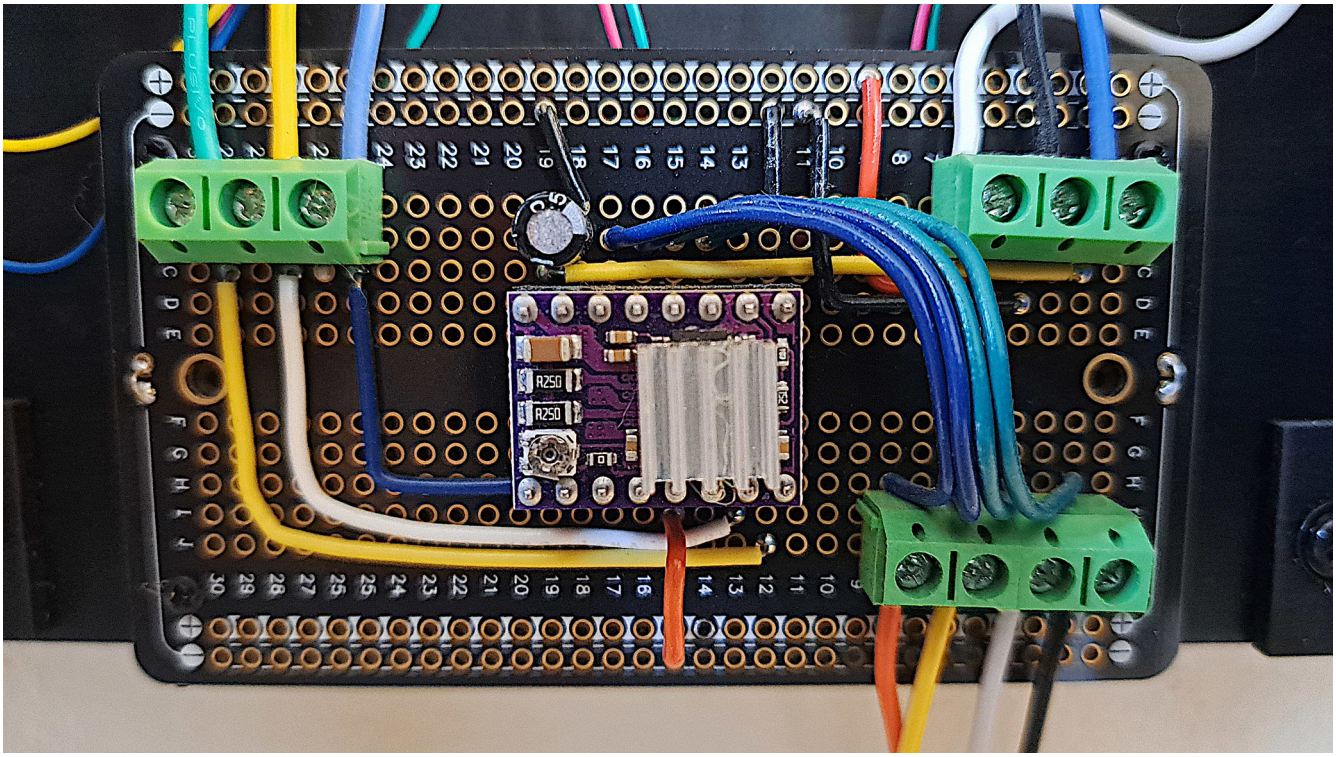
L298N dual H-Bridge module

https://www.amazon.com/gp/product/B014KMHSW6/

Motorized stainless steel ball valve with limit switches (x2)

https://www.amazon.com/gp/product/B06Y16BDFG/

Heating Stepper Control Board



The first thing I would suggest that you do is watch the following tutorial video on the Raspberry PI and stepper motors, it will take all of the mystery out of this one. This smart still controller expects you to have a heating system that is controlled by rotary adjustment, whether it's an SCR controller or gas fired burner and valve.

https://www.youtube.com/watch?v=LUbhPKBL_IU – Seriously, watch this video!

As you can see, there's a whole lot of nothing to this board because it's just point to point wiring from screw terminals to the stepper motor driver module and just a 100 uF electrolytic capacitor to smooth out any voltage drops. This is identical to what is used in the tutorial video.

The DRV-8825 is socketed rather than being soldered to the PCB. This is just one of those habits of mine because swapping out modules is the easiest way for me to troubleshoot them. I honestly can't say if these are prone to burning out, but I would rather be able to just swap them out without any tools if one burns out.

Since there isn't much point in a diagram here, due to the fact that's it's all just point to point wiring to the DRV-8825 module, I'll just explain what each screw terminal is connecting to. It wouldn't take a whole lot of effort to build things in a much more attractive fashion than I did with breadboard PCB clones.

Upper left terminal strip (left to right)

1. Raspberry PI GPIO 11, stepper direction control, module “dir” pin.

2. Raspberry PI GPIO 12, stepper pulse line, module “step” pin.

3. Raspberry PI GPIO 13, stepper enable/disable, module “sleep” pin.

Upper right terminal strip (left to right)

1. +3.3 volt supply, module “reset” pin.

2. Ground to both module “gnd” pins.

3. +19 volt supply, module “vmot” pin.

NOTE: A 100 uF, 50 volt electrolytic capacitor is connected between the “vmot” pin and ground. This is a polarized capacitor, don’t connect it backwards!

Lower right terminal strip (left to right)

1. Stepper motor coil 1a, module “a1” pin.

2. Stepper motor coil 1b, module “b1” pin.

3. Stepper motor coil 2a, module “a2” pin.

4. Stepper motor coil 2b, module “b2” pin.

The stepper motor coils aren’t critical as far as polarity, but connecting the coils to the wrong output pins of the module will make it run in reverse. If your motor runs the wrong direction when you increase your heating, just switch the coils or change the “Polarity” setting in the smart still controller, your choice.

DRV-8825 module

https://www.amazon.com/gp/product/B07XF2LYC8/

Nema 17 stepper motor

https://www.amazon.com/gp/product/B088BLQYDX/