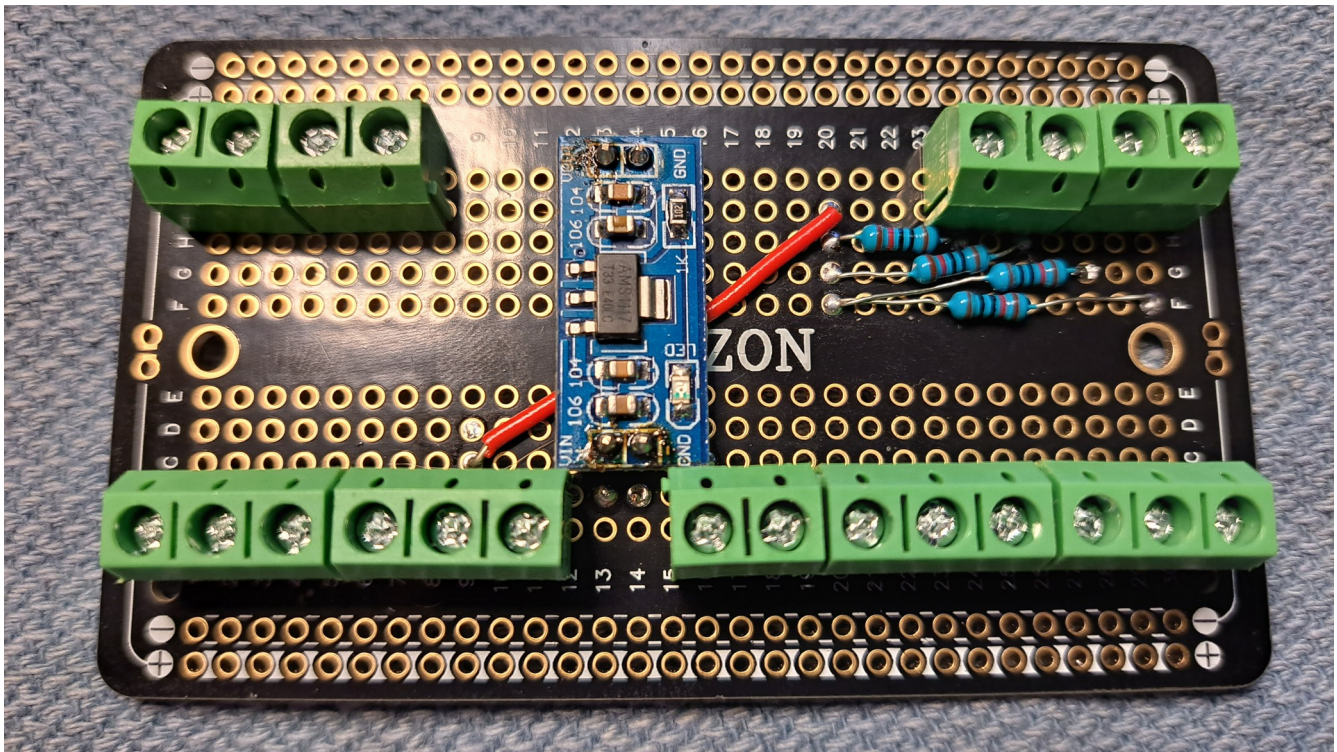


I almost always base my projects on PCB clones of breadboards to make planning easier. In rare cases, I will use plain old point-to-point boards if it doesn't involve a lot of jumper connections. Here's a link to the boards I used in the photos.

<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.

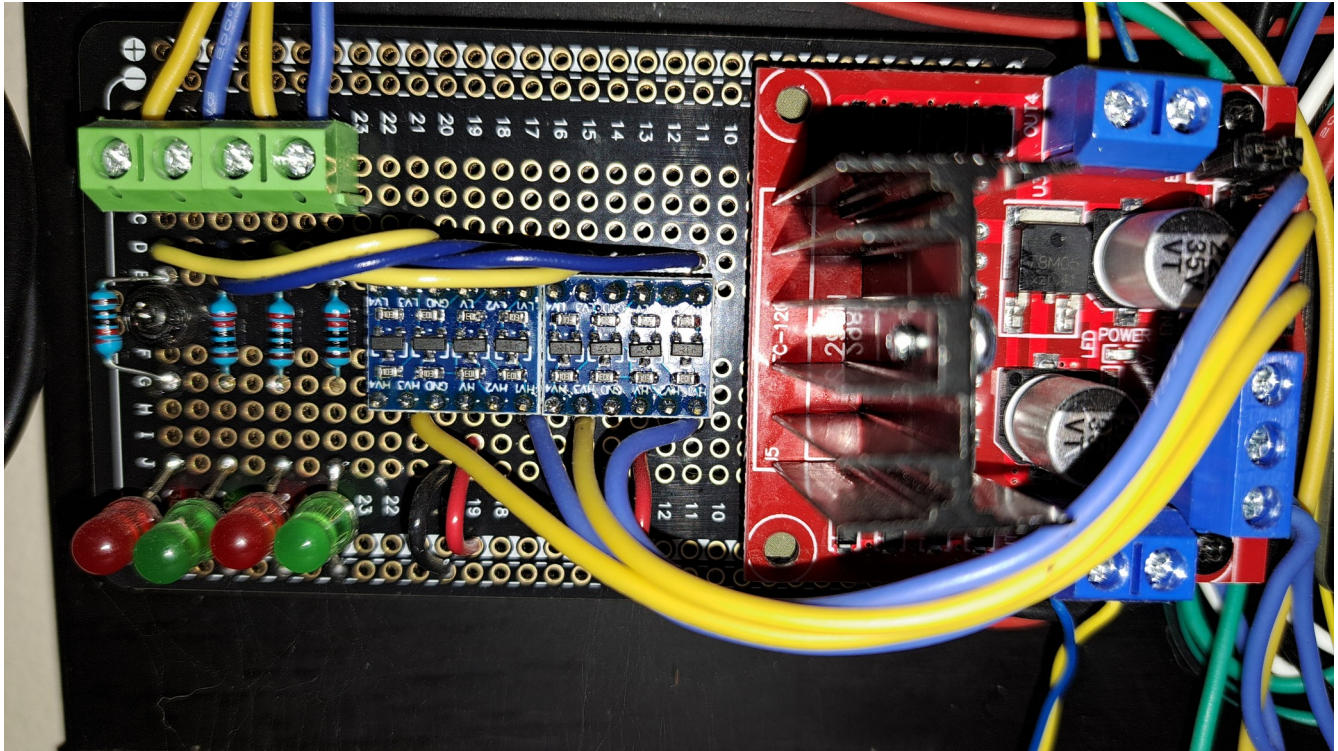
The pull up resistors are 10K and feed from the 5 volt supply. The 3.3 volt supply was an after thought, or I would have fed 4.7K resistors from there.

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.

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 5 volt regulator can't take any more and your valve motors will run slow.

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.