**Software Setup**

Sensor

* The sensor likely won’t work on the first run. On the main Mega with the sensor, run New\_ManualInput\_Stepper\_Control\_Mega\_NEMA.ino and remember to uncomment sendCommand(baud\_9600, sizeof(baud\_9600)). If the serial monitor is still displaying “Checksum validation failed” after a while, unplug the 5V wire and replug it. After this, the sensor should be stable.
  + The sensor is stable only when the serial monitor displays three non-zero values on each line and keeps on updating new values. When it only displays three 0’s per line, the sensor is still unstable and doesn’t read data. In this case, run the code again with sendCommand(baud\_9600, sizeof(baud\_9600)) uncommented, and replug the 5V wire as above.
  + After the sensor is stable, comment out sendCommand(baud\_9600, sizeof(baud\_9600)) and run it again. The sensor should fetch data normally, and the serial monitor should display three non-zero values on each line and keep updating. There should not be any message with “Checksum validation failed.”
  + The sensor must be working before anything else can be run.

Code To Upload

* On the Mega with the LCD screen (the “UI Mega”), upload the code UI\_Mega.ino. This code provides the user interface (including the rotary encoders for updating setpoints and the screen).
* On the Mega with the sensor (the main Mega), upload the code New\_Master\_Code.ino. This code is the master code that moves through the different stages to reach the setpoints.

Serial communication

* Make sure the two Megas are connected via their Serials. The main Mega should have the sensor connected to its Serial 1, so it communicates with the UI Mega via Serial 2. On the other hand, UI Mega communicates with the main Mega via Serial 1. These are all noted in the code. Make sure that the Rx of one Mega is connected to the Tx of the other.

**Notes**

How often does the sensor malfunction?

* Pretty much every time the system is shut down for a while. If running the master code does not work, it is likely that the sensor isn’t communicating properly. To ensure that the sensor can run every time, make sure to run New\_ManualInput\_Stepper\_Control\_Mega\_NEMA.ino before the rest of the code.
* We still can’t figure out why the sensor malfunctions this frequently. In Total\_PID.ino, which only includes the PID algorithm (stages 2 and 3 in the master code), the sensor rarely encounters “Checksum Validation failed” errors.

Reset

* The system is stable if the desired setpoints are reached, the motors stop turning, and the screen displays the sensor readings (which should be similar to the setpoint values).
* To reset the system after the system stabilizes, press the air knob once and the oxygen knob once. After both knobs are pressed, the system should reset by closing both the air and oxygen valves.
* Do NOT reset the system when the system is not stable yet. Do not press the knobs when the motors are still moving and the desired setpoints are not reached.

Stage 0 & Sensor Reading Delay

* This is the stage where the sensor fetches the actual O2 concentration of the sources. The air source typically has an O2 concentration of ~21%, while the O2 source should be close to 100%.
* The sensor may need to be further configured. When checking the O2 source, it displays anywhere from 95% to 98% O2 but never 100%, even though we use a tank of pure oxygen.
* Stage 0 is currently commented out in New\_Master\_Code.ino. It does not work all the time (in fact it fails to work more often than it works).
  + The sensor still has some communication issues even after the configuration above. The configuration allows the Mega to read and get the sensor’s data. However, there is a delay of a few readings.
    - For example, when the sensor fetches the O2 % in air, it displays 20.8%, 20.8%, 20.8%, 21.0%, 21.0% (it fetches five readings at once then averages them). When it fetches the O2 % in the O2 source after a while, it displays 21.0%, 21.0%, 21.0%, 97.2%, 96.8%. Clearly, the first three readings here are from before, when the air valve is still open. The last two readings are the readings for O2.
    - The sensor readings should not have jumped from 21% to 97.2%. When the oxygen starts flowing in, the readings should increase incrementally (e.g. 21%, 45%, 65%, 85%, 97%). If it jumps from 21% to 97.2%, then there must be some delay in detecting the readings.
    - This makes it difficult to fetch the correct readings. For now, Stage 0 is commented out, and the concentrations used for the calculations in Stage 1 are set to 21% for air and 100% for oxygen.
  + When running Stage 2 (which is when it adjusts the air motor to the calculated air flow setpoint), there are sometimes “Checksum validation failed” errors. The algorithm runs well even with these errors and can still reach the setpoints.
    - The minor issue is that the algorithm doesn’t detect that the motor has already turned, at least in the first 3 or 4 movements. For example, the sensor reads 0 LPM even though the motor is already opened by 200 steps. After another 200 steps, the sensor reading is still at 0 LPM, and after another 200 it is still at 0 LPM. The Serial monitor then prints “Checksum validation failed,” which indicates some communication error with the sensor, and the code automatically fixes the issue before the next motor movement. After another 200 steps of opening, the sensor reads, say, 35 LPM (because it has already opened 800 steps). So, the readings are 0 LPM, 0 LPM, 0 LPM, and 35 LPM. When it jumps all the way to 35 LPM, it may have already overshot the setpoint. But the algorithm will correct for that and close the motor until the setpoint is reached.
    - This doesn’t really affect how the overall algorithm works, but now it is less efficient than it should have been. (In one case, with 60 LPM and 60% O2 as setpoints, the sensor failed to get readings at first, so the total flow rate shot all the way up to 100 LPM before the algorithm started detecting readings and closed the valve. This may actually be dangerous to the patient.)
  + This issue is also present in the Reset stage. The reset stage cannot print out the correct readings because of this sensor delay. When both the valves are closed, the sensor should read 0 LPM and 0% O2, but when we display the actual readings the LPM is still non-zero, which means it is displaying data from the previous reading.

Troubleshooting

* If New\_Master\_Code.ino doesn’t work, follow the sensor configuration instructions above. If that still doesn’t work, go to Prototyping > Arduino > Modified Artemis, and run the Total\_PID file. It only includes Stages 2 and 3 in the master code, where the motors are adjusted to their flow setpoints.
* Total\_PID.ino never encounters any “Checksum validation failed” issues, but New\_Master\_Code.ino frequently does. They do pretty much the same thing, although New\_Master\_Code.ino includes user inputs, communication between two Megas, and LCD screen displays. All these other things may have disrupted communication between the sensor and the main Mega, or there may be delays when these other things are running.

John’s Code

* We tried to modify and run some of John’s code from before. In our version of the project, we discarded mostly everything that John did. Our algorithm is much simpler than John’s PID algorithm (which didn’t work well at all). Don’t go through John’s code. They aren’t worth your time. We already spent 4 weeks doing that before coming up with a new, simplified algorithm.
* [Motor & Valve Testings](https://docs.google.com/document/d/1b_SWOcd7VZTsdrpVrNKoZO5QYNt-mqXgGez2lnKgZk8/edit) details a lot of our efforts to try and decipher John’s code. We did a lot of testing on his existing code files, but they ended up being unnecessary for our new master code.

Valve Characteristics Curves

* In the Modified Artemis folder, there is code in Test\_characteristics that allows us to plot the flow rate vs the % of the valve opened. These curves can also be found in Prototyping > Valve Characteristics Curves.
* They ended up being quite useless for the final algorithm. The master code can automatically adjust to the right setpoints without relying on this curve.
* The curves may be helpful when testing out new valves. They can determine how much of the valve needs to be opened to reach certain flow rates. However, we didn’t end up trying new valves.
* The curves should have three distinct regions with different slopes. Each region should have a different Kp value in the PID algorithm ([Motor & Valve Testings](https://docs.google.com/document/d/1b_SWOcd7VZTsdrpVrNKoZO5QYNt-mqXgGez2lnKgZk8/edit) may have more explanation on this). We did adjust for the Kp values: Kp is 3 for flows greater than 40 LPM, while Kp is 10 for flows less than 40 LPM. But this adjustment is merely based on experimenting with different values. We did not rely on the curves at all.
* John also had code for finding the optimal Kp values (Kc\_Ku\_Testing\_Mega.ino in Modified Artemis). We didn’t use his code to find the best Kp value. We tested out a few random values and found the ones that work well enough.