Variable
back pressure regulator

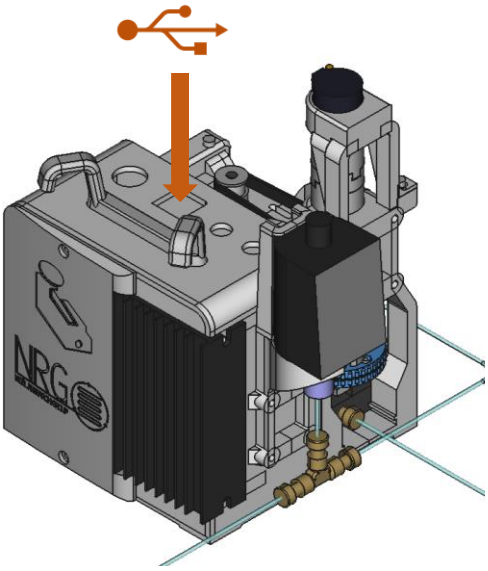
Technical Specifications

Simone Pilon <s.pilon @ uva.nl> - Noël Research Group - 2023

Table of Contents

Usage

Make sure the following conditions are verified before operating the device:



1. The device is appropriately connected to the flow system, as described in the device overview section.
2. The device is receiving both 12v and 24v power, as described in the device overview section.

Note: The device will not operate correctly without dedicated 12v power: most computers do not provide a stable power supply for the Arduino, leading to incorrect pressure readings, as the analogue signal is internally compared to the supply voltage.

3. The device is connected to a computer via USB.

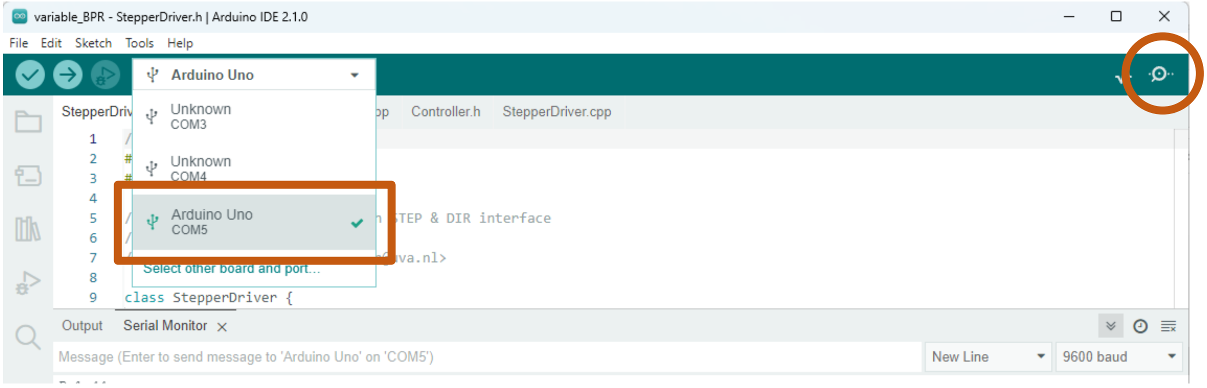
The device is controlled by connecting to a computer via the USB cable. The Arduino USB port is located behind the square hole on the top of the box. Once connected, communication with the device can be achieved via any serial interface on the computer.

Arduino Serial Monitor

To control the device remotely via the Arduino Serial Monitor, start the Arduino IDE (tested on version 2.1.0), then connect the device to the USB port of the computer. Make sure the “Serial Monitor” is open, by clicking on the top-right button to activate it. Then, select the correct USB port from the menu on the left. If asked, select “Arduino UNO” as the board.

Once connected, it is possible to send and receive data from the device.

Note: if you are assembling a new device or replacing the Arduino board, the firmware will need to be uploaded to the board before sending any commands: open the firmware sketch (solar_simulator/solar_simulator.ino file) in the Arduino IDE and press the ‘upload’ button on the top left. If there are no errors, the message ‘upload successful’ should appear at the bottom of the screen.



variable_BPR - StepperDriver.h | Arduino IDE 2.1.0

```
STEP & DIR interface
va.nl>
class StepperDriver {
```

Message (Enter to send message to 'Arduino Uno' on 'COM5')

To test if the Arduino board is working as expected, type ‘R1’ in the Serial monitor message bar and press enter. The message ‘PRESSURE_CONTROL_0’ should appear right below it.

List of Commands

This is a list of commands which can be sent to the device via the serial interface:

'Sx=y' Set variable x to value y

Sending the ASCII character 'S' tells the microcontroller that a variable value needs to be changed. An integer number must follow the 'S' character, indicating the variable number (see table below). After a delimiter character (any non-digit except '.' Will do), the value for the variable must be given. The type depends on the variable.
Note: in case of a string variable, the command must terminate with a newline.

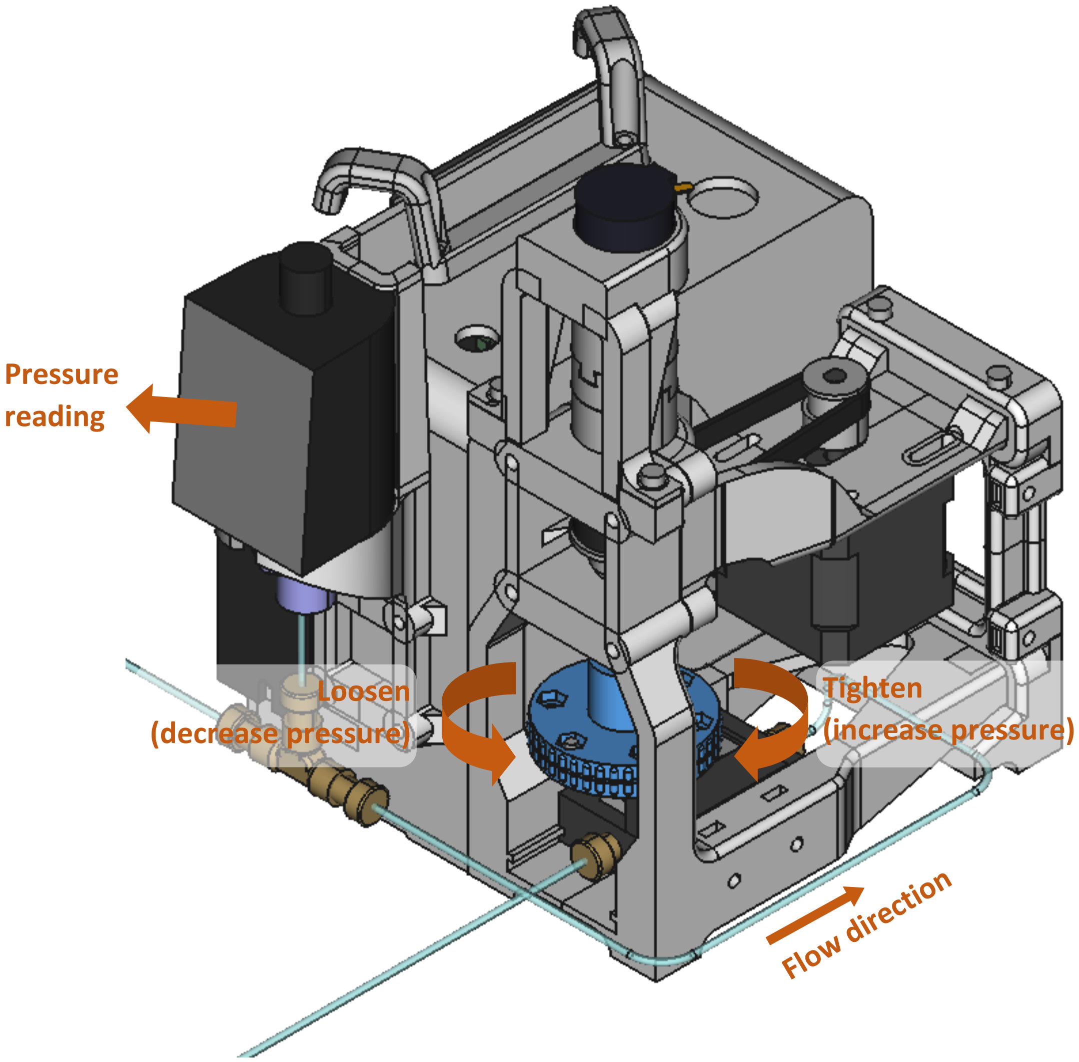
'Rx' Read variable x

Sending the ASCII character 'R' tells the microcontroller that a variable value needs to be printed to serial. An integer number must follow the 'R' character, indicating the variable number (see table below).

Variable number	Access	Type	Description
0	RESERVED	-	Serial.parseInt returns 0 on error To avoid undesired variable access, this value is not used.
1	READ/WRITE	String [20]	Device identifier This can be used to distinguish similar serial devices.
2	READ/WRITE	Int [0-1]	Device enable On/_Off state Setting to 1 enables the device. Setting to 0 disables it. Once the device is enabled, it attempts to achieve the target pressure by moving the back pressure regulator valve. In this state the valve should not be adjusted manually, as this may damage the device and interfere with its operation. If the device is disabled, the valve position can be adjusted manually.
3	READ/WRITE	Float	Pressure setpoint [Bar, relative] Access this variable to set or read the desired pressure value.
4	READ_ONLY	Float	Pressure measurement [Bar, relative] Reading this variable provides the latest pressure measured by the device.
5	READ_ONLY	Int [0-1024]	Absolute valve position Read the absolute position of the valve as an arbitrary number.

Variable number	Access	Type	Description
6	COMMAND	Float	Begin calibration procedure [Bar relative] The device reads the pressure as an analogue signal. For this reason, it stores a 2 points calibration curve to convert the signal intensity into the desired units. To set this curve, disable the device (variable 2) and manually bring the system pressure close to the minimum by operating the adjustable valve (1.0 bar is a good value). Then send this command using the known pressure value in Bars as argument. The accurate pressure value can be read from the screen of the pressure sensor instrument. Note: this command alone will have no effect on the behaviour of the device, as it must be followed by command 7.
7	COMMAND	Float	End calibration procedure [Bar relative] Stores the second calibration point, causing the calibration data to be changed and stored on the device. Before the command is issued, the valve should be adjusted to achieve a higher pressure than the one recorded for the previously issued command 6, the larger the difference, the more accurate the calibration will be. Note: issuing this command before command 6 will result in undefined behaviour.
8	COMMAND	None*	Set current valve position as minimum Sets the lower limit for the actuation of the valve. The device will not loosen the valve beyond this position. Before issuing the command, the valve should be manually loosened until the pressure reading is zero, or a few turns after that.
9	COMMAND	None*	Set current valve position as maximum Sets the higher limit for the actuation of the valve. The device will not tighten the valve beyond this position. Before issuing the command, the valve should be manually screwed tightly (as much as it will go without strong resistance), then half a turn back.
10	COMMAND	Int	Move motor Moves the motor by the specified number of steps (both positive and negative values are allowed, 2560 steps correspond to 1 turn of the valve). The direction of rotation is given by the sign of the number. This command should be issued only in the disabled state (S2=0).

Device overview



Construction

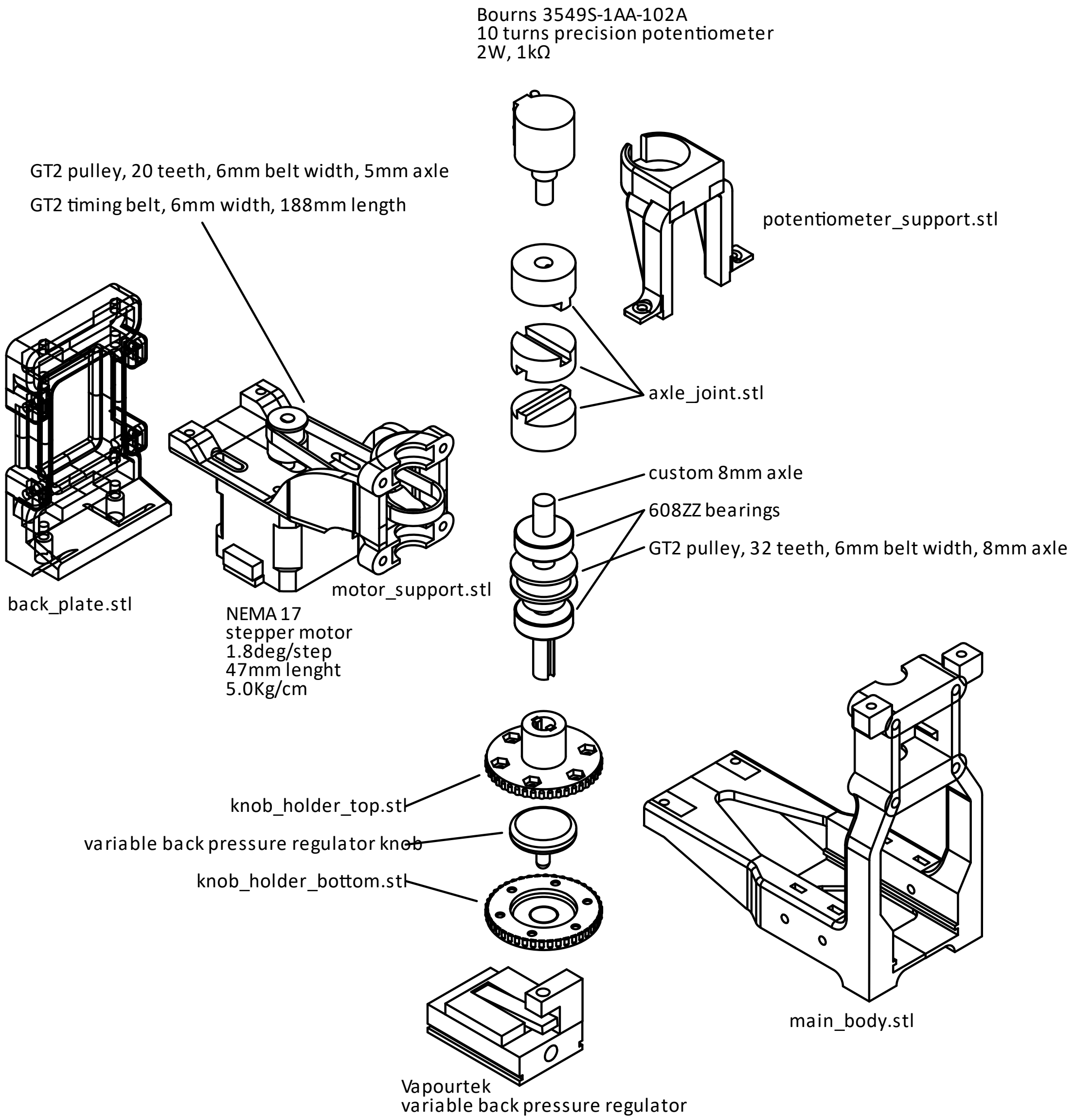
Principle of operation

A commercial variable back pressure regulator is controlled automatically by a motor turning its knob.

The knob is mechanically linked to an 8mm axle, which imparts rotation but allows some degree of motion along the axle direction (the knob screws up and down by a few millimeters). The axle in turn is linked to a NEMA 17 stepper motor (200 steps/revolution) via a GT2 timing belt with an 8 to 5 reduction factor. The axle is also connected to a 10 turns precision potentiometer, which allows the microcontroller to evaluate the absolute position of the valve even in the event that this is manually adjusted by the user.

A pressure sensor is connected to the microcontroller, which gives a reading of the pressure before the back pressure regulator. By moving the back pressure regulator knob and reading the resulting pressure, the microcontroller can find an optimal position to achieve the desired pressure.

Main mechanical components

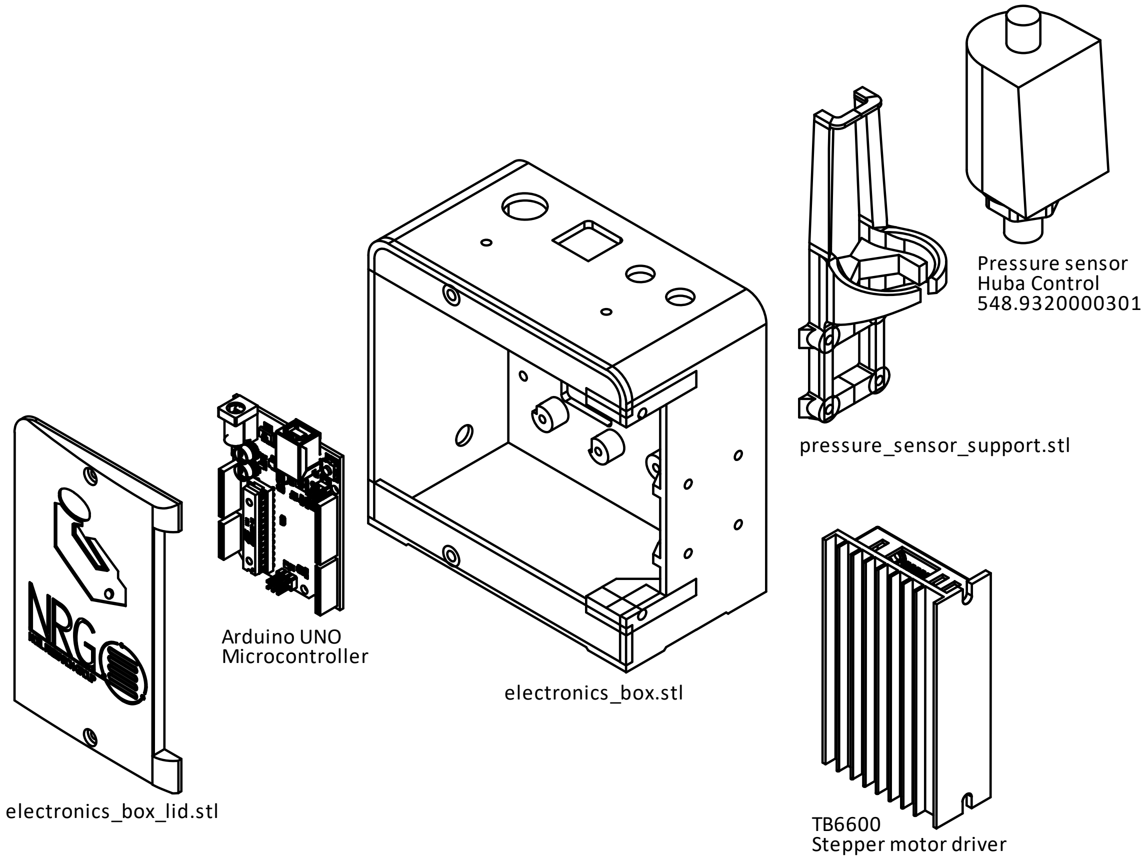

Bourns 3549S-1AA-102A
10 turns precision potentiometer
2W, 1kΩ
GT2 pulley, 20 teeth, 6mm belt width, 5mm axle
GT2 timing belt, 6mm width, 188mm length
potentiometer_support.stl
axle_joint.stl
custom 8mm axle
608ZZ bearings
GT2 pulley, 32 teeth, 6mm belt width, 8mm axle
back_plate.stl
motor_support.stl
NEMA 17
stepper motor
1.8deg/step
47mm lenght
5.0Kg/cm
knob_holder_top.stl
variable back pressure regulator knob
knob_holder_bottom.stl
main_body.stl
Vapourtek
variable back pressure regulator

All 3D printed components (*.stl) were printed from PLA plastic.

1. Unscrew the knob of the pressure regulator completely and fit it between the knob_holder elements. Six M3x6mm screws are used to fix the two halves together, with the knob in between.
2. Re-screw the knob and slide the back pressure regulator into the main_body.
3. Mount the NEMA 17 motor to the motor_support with 4 M3x12mm screws loosely (should be able to move back and forth).
4. Fix the 20 teeth pulley to the motor axle and place the timing belt on it.
5. Assemble the custom 8mm axle, 608ZZ ball bearings and larger pulley in a stack as shown in the picture (bearing – pulley – bearing).
6. Slip the whole axle stack into the motor support, making sure the belt is around both motor and axle pulley. Push the motor towards the axle to make this easier.

7. Pull the motor away, putting tension on the belt. While in tension, tighten the motor screws, so that the belt remains in tension.
8. Connect the motor_support to the main_body with 4 M3x35mm screws, making sure that the axle also fits inside the knob_holder_top.
9. Slide the 3 components of the axle_joint together, they should slide freely. The bottom component needs to be fitted on the 8mm axle, while the top will fit the potentiometer axle.
10. Mount the potentiometer on the potentiometer support (using the nut and washer provided with the potentiometer).
11. Press the potentiometer axle onto the top of the axle joint, which in turns presses on the 8mm axle. The axle should be able to turn the potentiometer.
12. Mount the potentiometer_support to the top of the main_body using two M3x12 screws.

Electronics

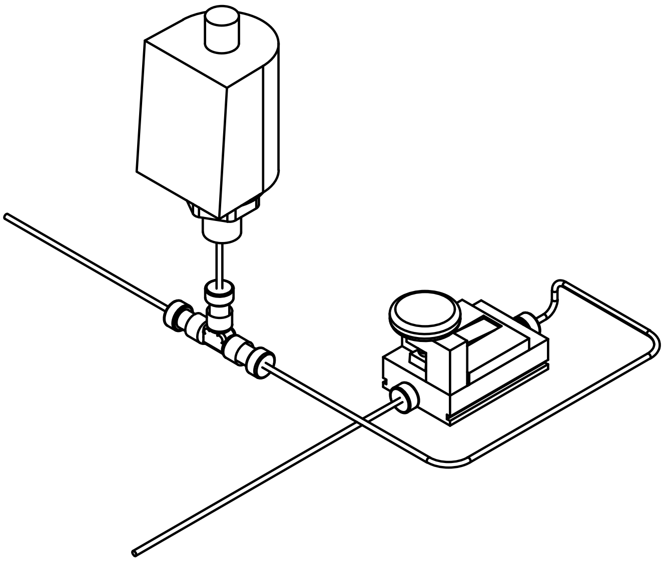


A custom box can be printed for all electronics components to be neatly packed, but this is not required for the operati

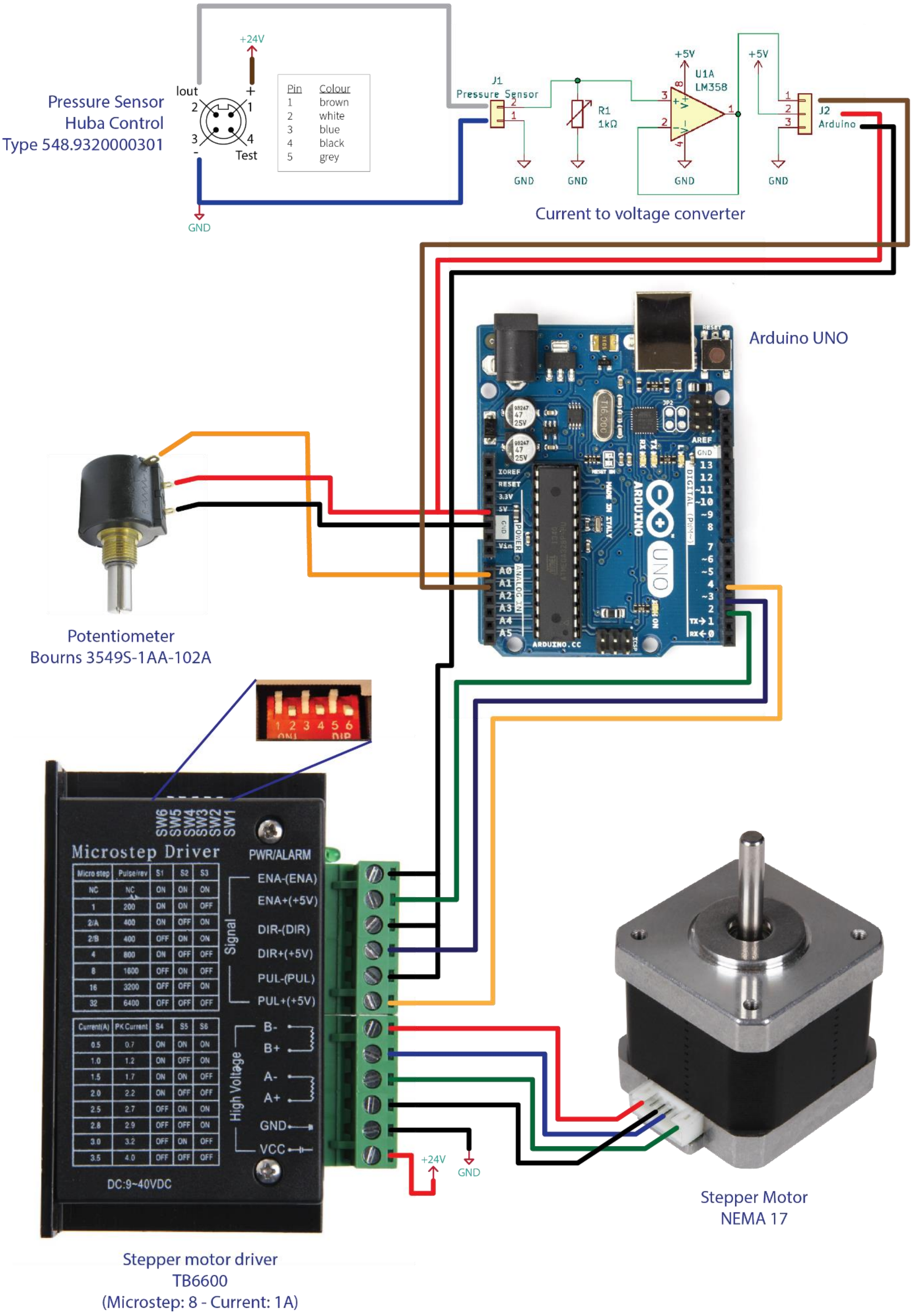
on of the device:

1. Fix the Arduino microcontroller and the current-to-voltage conversion board inside the electronics_box.
2. Fix the pressure_sensor_support to the left side of the electronics_box and place the pressure sensor inside of it.
3. It is advised to make all electronics connections at this stage. The electronics_box has a hole in the back for the motor wires to come through (the box can be attached to the mechanical components assembly). See the electrical connections section for detailed information.

4. Mount the stepper motor driver in front of the right of the electronics_box with two M4x6mm screws.
5. Attach the electronics_box_lid with 2 M3 screws.
6. See the serial monitor section for information on programming the Arduino for the first time.
7. Connect the flow tubing to the device as shown below (flow starts at top left and ends at bottom left).
8. After programming the Arduino for the first time, some steps are required before commencing operations:
    a. Set max and min valve positions for motors (commands 8 and 9 in the serial interface).
    b. Calibrate pressure sensor reading (commands 6 and 7 in the serial interface).



Electrical connections



Pin	Colour
1	brown
2	white
3	blue
4	black
5	grey

Micro step	Pulse/rev	S1	S2	S3
NC	NC	ON	ON	ON
1	200	ON	ON	OFF
2/A	400	ON	OFF	ON
2/B	400	OFF	ON	ON
4	800	ON	OFF	OFF
8	1600	OFF	ON	OFF
16	3200	OFF	OFF	ON
32	6400	OFF	OFF	OFF

Current(A)	PK Current	S4	S5	S6
0.5	0.7	ON	ON	ON
1.0	1.2	ON	OFF	ON
1.5	1.7	ON	ON	OFF
2.0	2.2	ON	OFF	OFF
2.5	2.7	OFF	ON	ON
2.8	2.9	OFF	OFF	ON
3.0	3.2	OFF	ON	OFF
3.5	4.0	OFF	OFF	OFF

Bill of Materials

1. 10 turns precision potentiometer 2W, 1kΩ (Bourns 3549S-1AA-102A)
2. Pot_holder (3D-printed part)
3. Axle Joint (3D-printed part, 3 components)
4. 608ZZ bearings (x2) 8mm ID, 22mm OD, 7mm thickness
5. GT2 pulley 32 teeth, 6mm belt width, 8mm axle diameter
6. 8mm axle (dimensions and features in CAD file)
7. Knob holder top (3D-printed part)
8. Knob holder bottom (3D-printed part)
9. Back pressure regulation valve (Vapourtek)
10. Axle holder (3D-printed part)
11. NEMA 17 stepper motor, 1.8deg/step, 47mm lenght, 5.0Kg/cm
12. GT2 timing belt, 6mm width, 188mm length, closed
13. GT2 pulley 20 teeth, 6mm belt width, 5mm axle diameter
14. Motor holder (3D-printed part)
15. Counterplate (3D-printed part)
16. Ebox lid (3D-printed part)
17. Stepper driver (TB6600)
18. Arduino UNO board
19. Ebox (3D-printed part)
20. Sensor cable holder (3D-printed part)
21. Sensor holder (3D-printed part, 2 parts)
22. Pressure sensor (Huba Control Type 548.9320000301)