CIDAR Lab DepMaM Project

Automatization and Subsystems Integration Technical Documentation

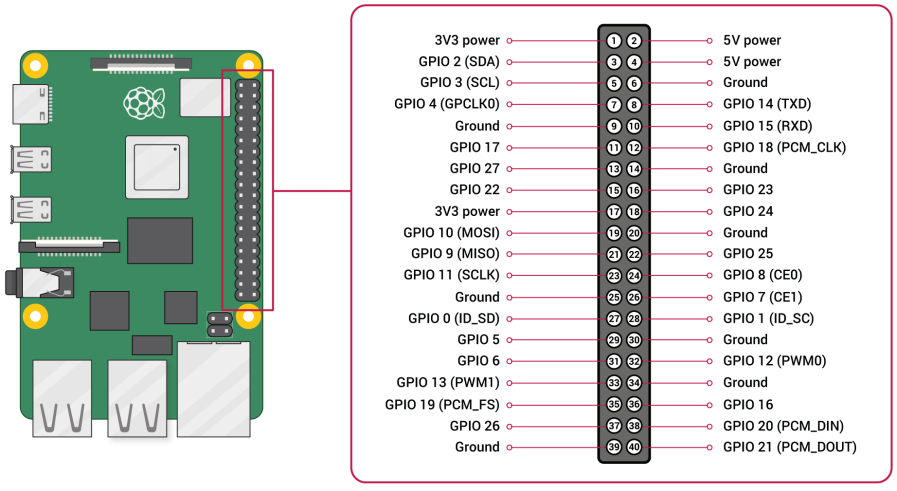
Setup:

- Raspberry Pi
- NEMA 17 Bipolar Motor
- Motor Driver TB6600

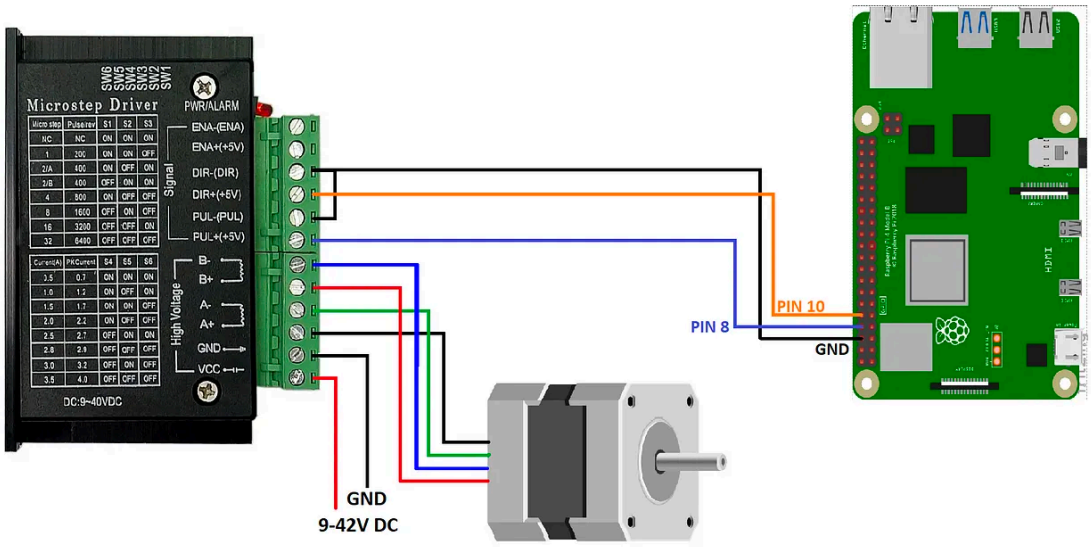
NEMA 17 Bipolar Motor:
Model: 17HS16-2004S1
Link: NEMA 17 Bipolar 45Ncm
Step Angle: 1.8 deg
Rated Current/phase: 2A
Connection:

Black - A+
Green - A-
Red - B+
Blue - B-

Raspberry Pi GPIO Pin Diagram:



Wiring Diagram:

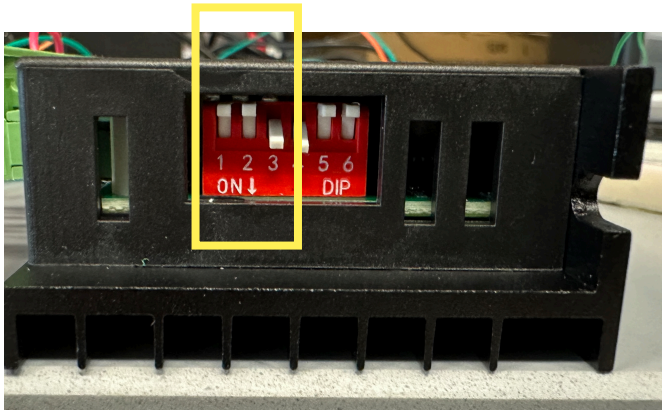


Wiring Diagram to Control NEMA 17 with Raspberry Pi and TB6600

TB6600 Micro Step Setting:

- Set the motor micro step via the first three DIP Switch
- Step Angle = Motor Step Angle/Micro Step
- Microstep 4 = 1.8 deg / 4 = 0.45 deg

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

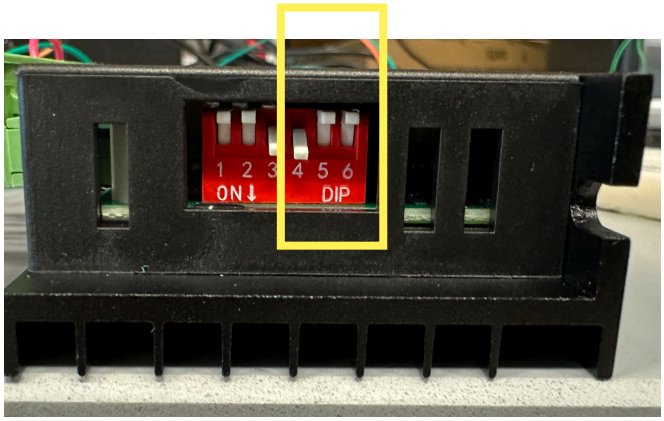


Use the first 3 DIP switch to control micro step setting. Currently it is set at 16 micro steps per step. meaning that a full revolution is 200 * 16 = 3200 steps

TB6600 Current Control Setting:

- Current needs to be controlled to match the motor
- NEMA 17 uses 2A, so we need to set the current control on the motor driver accordingly.
- Use 4-6 DIP Pins to control current:

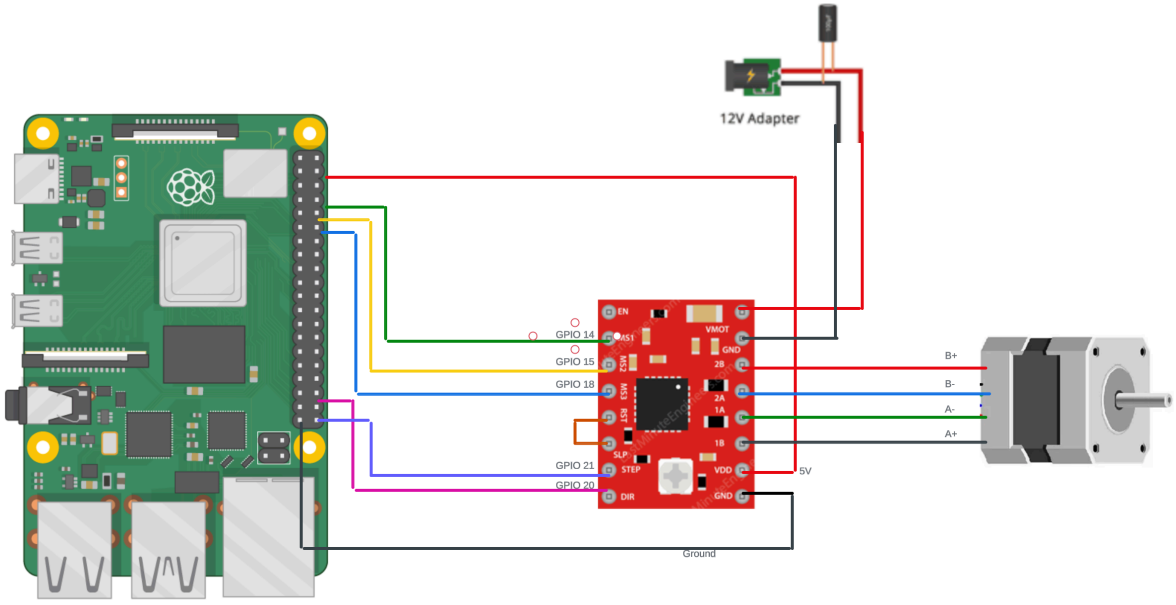
Current (A)	S4	S5	S6
0.5	ON	ON	ON
1.0	ON	OFF	ON
1.5	ON	ON	OFF
2.0	ON	OFF	OFF
2.5	OFF	ON	ON
2.8	OFF	OFF	ON
3.0	OFF	ON	OFF
3.5	OFF	OFF	OFF



Current is set at 2A for NEMA 17

A4988 Motor Driver Setup (Recommended):

- Small Unit
- Allows for micro-stepping
- Current control on the Unit
- Wiring diagram shown below



Problems and Trouble Shootings:

- If the motor overheats or twitches when turning, the motor is not getting enough current from the motor driver.
- https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/
- This link details how the motor driver should be set up with the correct voltage by turning the knot on the driver.

Useful Links:

NEMA 17 and Motor Driver TB6600 RP setup:
https://danielwilczak101.medium.com/control-a-stepper-motor-using-python-and-a-raspberry-pi-11f67d5a8d6d

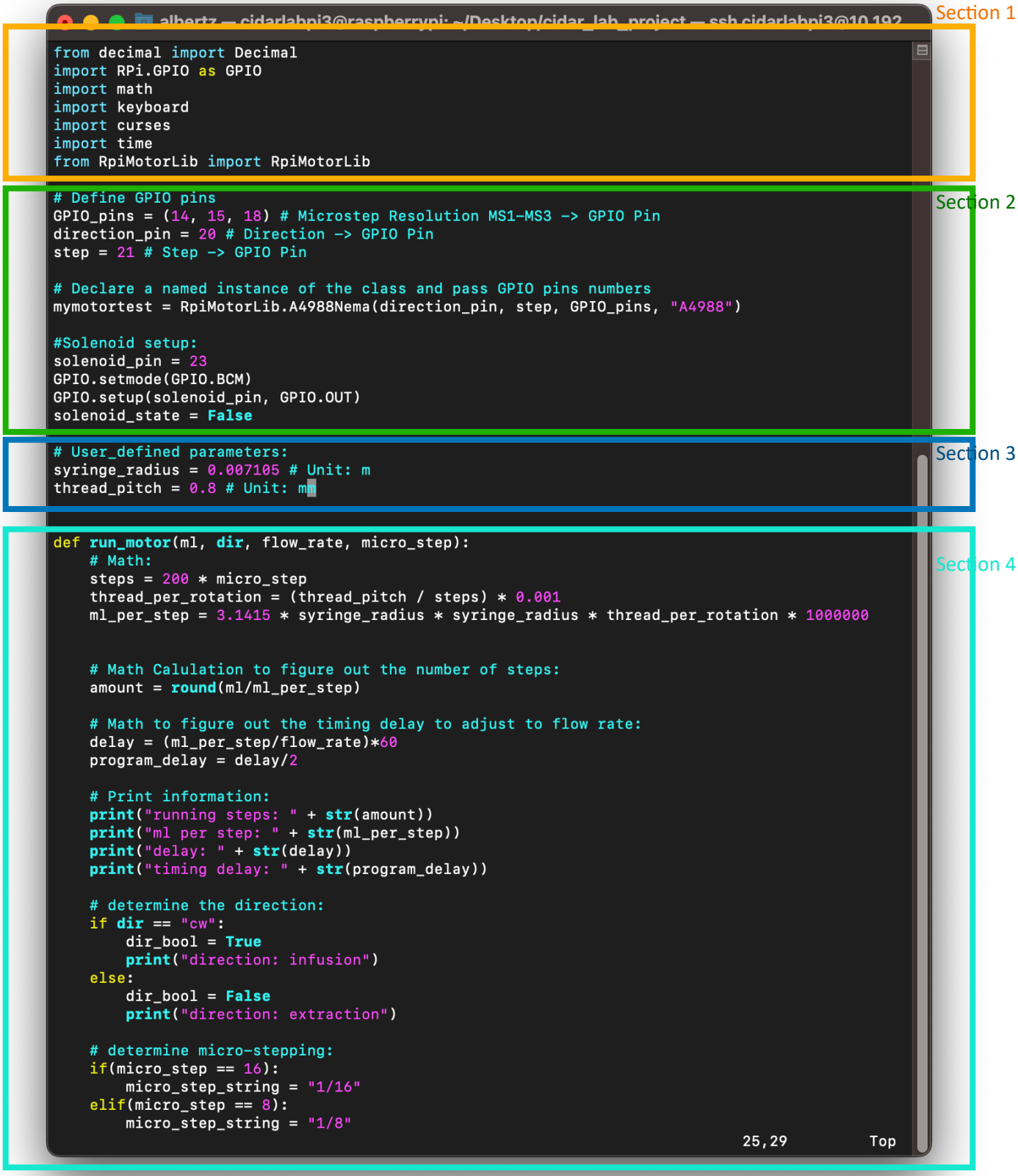
NEMA 17 and Motor Driver A4988 Setup:
https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/

Nema 17 Bipolar 45Ncm Specifications:
https://www.omc-stepperonline.com/nema-17-bipolar-45ncm-64oz-in-2a-42x42x40mm-4-wires-w-1m-cable-connector-17hs16-2004s1

3D Printed Syringe Pump Rack Instruction:
https://www.instructables.com/3D-Printed-Syringe-Pump-Rack/

Software Explanation:

(software should already be working without any installing or changes)



albertz — cidarlabpi3@raspberrypi: ~/Desktop/cidar_lab_project — ssh cidarlabpi3@10.192...

```
    elif(micro_step == 4):
        micro_step_string = "1/4"
    elif(micro_step == 2):
        micro_step_string = "Half"
    elif(micro_step == 1):
        micro_step_string = "Full"
```

Section 5

```
    # run the motor:
    mymotortest.motor_go(dir_bool, micro_step_string, amount, program_delay, False, 0.05)
```

Section 6

```
def toggle_solenoid():
    global solenoid_state
    solenoid_state = not solenoid_state
    GPIO.output(solenoid_pin, solenoid_state)
    print("Switchd")
```

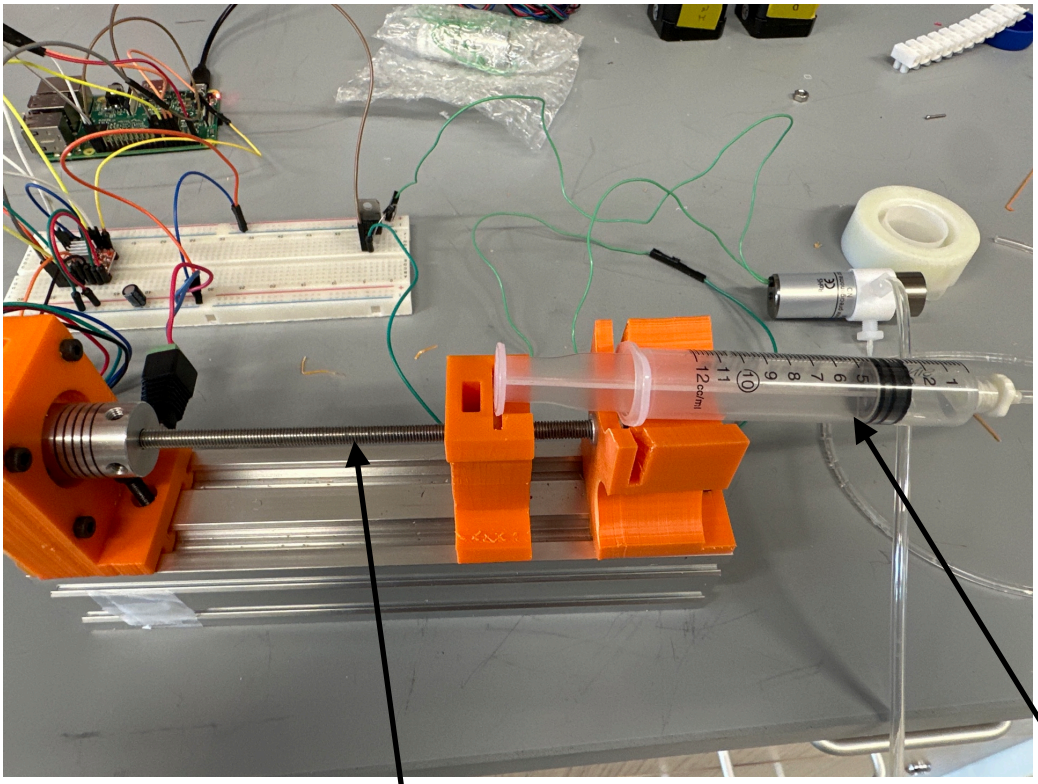
Section 7

```
while True:
    try:

        # Ask user for input
        ml = float(input("Enter ml: "))

        dir = input("Enter direction (cw/ccw): ").lower()
        if dir not in ["cw", "ccw"]:
            print("Invalid direction. Please enter 'cw' or 'ccw'.")
            continue

        flow_rate = float(input("Enter flow rate ml/min: "))

        if flow_rate > 0 and flow_rate < 1:
            micro_step = 16
        elif  flow_rate >= 1 and flow_rate < 2.37879:
            micro_step = 8
        elif flow_rate >= 2.37879 and flow_rate < 4.758:
            micro_step = 4
        elif flow_rate >= 4.758 and flow_rate < 9.516:
            micro_step = 2
        else:
            print("Invalid flow_rate. Please enter >0 and < 9.516 ml/min. ")
            continue

        solenoid_toggle = str(input("Please enter 'yes'to toggle solenoid. "))
        if solenoid_toggle == "yes":
            toggle_solenoid()
            time.sleep(0.2)



        # Run the motor with the provided input:
        run_motor(ml, dir, flow_rate, micro_step)


    except ValueError:

        print("Invalid Input. Please enter a valid number.")
~
~
~
-- INSERT --                                                    116,61        Bot
```

Section 1:
import all necessary libraries
all libraries should already come pre-installed if using the same raspberry pi
to install:
sudo pip3 ___ install

Section 2:
Define GPIO Pins
Pins are interchangeable
This code currently follows the wiring diagram above

Section 3:
Define equipment parameters:
Thread pitch in mm
Inner radius of the syringe in m
THE RECOMMENDED THREAD PITCH IS 0.8mm, AND THE RECOMMENDED SYRINGE RADIUS IS 0.007105m. THOSE ARE CURRENTLY TESTED AND SHOULD PROVIDE THE MOST ACCURATE RESULT.



Section 4:

Section 4 contains the math to calculate the number of steps (or turns) the motor needs to dispense/extract the requested amount of liquid:

The relations:

microstepping
This is a variable in program

200 steps per revolution × 16 microsteps = 0.8 mm (thread pitch)

3600 steps per revolution using 1/16 microsteps = 0.8 mm

1 step = x mm

$x = 0.8/3200$

1 motor step x = 0.00025 mm

= 250 nm (nanometers)

This means that for 1 step on the motor, the thread moves about 250 nm.

The syringe inner diameter:

$1\ m^3 = 1 \times 10^6\ ml$

Area of a circle: πR^2

Volume of cylinder: $\pi R^2 h$

Syringe diameters:

10 mL (12 mL) 14.21 mm

radius:

$14.21\ mm/2 = 7.105\ mm$

so if the motor moves 1 step (250 nm), the total volume dislocated is:

$3.14 \times 0.007105\ m^2 \times 0.00000025\ m = 3.962760546 \times 10^{-11}\ m^3$

$= 3.962760546 \times 10^{-5}\ ml$

convert all units to m

if I want to dispense 2 ml of water:

$\frac{2\ ml}{3.962760546 \times 10^{-5}} = 50469.86867 = 50470$ steps

Flow rate calculation:

Flow rate Q is given by $Q = \frac{D_x A_x}{t}$ or $Q = \frac{V}{t}$ change in volume over change in time.

From our previous equation, we know that the change in volume per step (Ex. for 1/16 micro-step) is $3.962760486 \times 10^{-5}$ mL. We just need to find t, the delay time for our raspberry pi. Rearrange the equation, we get

$t = \frac{V}{Q}$

$$t = \frac{3.96465 \times 10^{-5}\ mL}{Q} = \frac{3.96465 \times 10^{-5}\ mL}{1\ mL/min} \times \frac{3.96465 \times 10^{-5}\ min}{4} \times \frac{60\ secs}{1\ min}$$

↖ plug in the flow rate you want

$= 0.002379$ secs

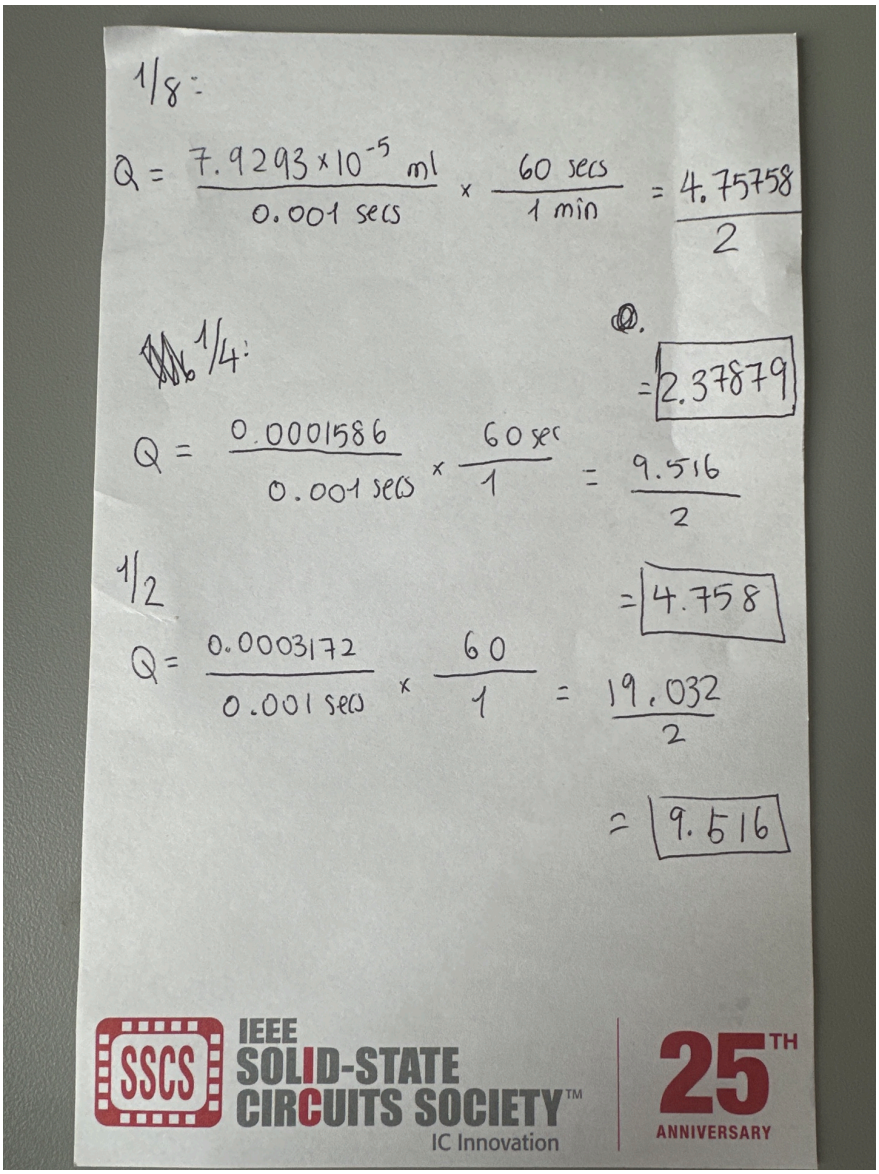
And obviously, the t delay time would change depending on the volume dispense per step(& radius, thread pitch, and microstepping).

Leave the final delay time in secs as the motor library takes in secs as unit parameter.

Note:
To use the delay in the program. We need to divide this delay by 2: program_delay = delay / 2
This is because the stepper motor uses two cycles to make one turn. So we need to divide this delay by 2 for the motor to turn for the correct flow rate.

There is also part of the code that determines the amount of micro-stepping it needs for the motor to turn at the correct speed. This is because the least delay GPIO pins could output is 0.001 secs (or 1 ms). So, we need to adjust micro-stepping based on the request flow rate.

Below is the math to determine the upper bound of flow rate for each micro-stepping setting:



Section 5:

Section 5 instantiates the motor, and this is what makes the motor run. Below are the inputs and parameters. A lot are variables because it depends on ml, direction, and flow rate from user inputs.

The main method that moves motor is called motor_go, takes 6 inputs.

```
motor_go(clockwise, steptype, steps, stepdelay, verbose, initdelay)
```

ID	Name	Type	Default	Help
(1)	clockwise	bool	false	Turn stepper counterclockwise
(2)	steptype	string	Full	type of drive to step motor 5 options, Full, Half, 1/4, 1/8, 1/16
(3)	steps	int	200	Number of steps sequence's to execute. Default is one revolution , 200 in Full mode.
(4)	stepdelay	float	0.05	Time to wait (in seconds) between steps.
(5)	verbose	bool	false	Write pin actions
(6)	initdelay	float	1mS	Intial delay after GPIO pins initialized but before motor is moved

for more information on this python library, visit:
https://github.com/gavinlyonsrepo/RpiMotorLib/blob/master/Documentation/Nema11A4988.md

Section 6:

This function is for solenoid switching.
Whenever this function is called, it switches the solenoid from high -> low to low -> high, therefore switching the solenoid.

Section 7:

This section asks the user to input the parameters that they want:

- ml: the amount of liquid to infuse or extract
- direction: ‘cw’ for infusion and ‘ccw’ for extraction
- ml per min: for flow rate
- type ‘yes’ to switch solenoid. nothing or anything else to skip

Type [Control ^ C] to stop the program

Running the Program:

(Required to have Linux Terminal)

Step 1:
Power Raspberry Pi

Step 2:
Connect a laptop or PC wifi to BU Guest

Step 3:
In your terminal, type:

```
(base) ➜  ~ ssh cidarlabpi3@10.192.47.10
```

ssh cidarlabpi3@10.192.47.10

Troubleshooting:
For some reason, if you cannot remote ssh into the raspberry pi, connect the raspberry pi to a monitor and check and ensure that the IP address of the pi did not change.

Step 4:
Enter password: **CIDARLAB_$$$123**

Step 5:
Once connected to the raspberry pi, we just need to access the Python file and run the program.

```
cidarlabpi3@raspberrypi:~ $ ls
Bookshelf  Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
cidarlabpi3@raspberrypi:~ $ cd Desktop
cidarlabpi3@raspberrypi:~/Desktop $ ls
cidar_lab_project  test5.py
cidarlabpi3@raspberrypi:~/Desktop $
```

ls: to show the list of files in the current directory
cd Desktop: go to the desktop directory
ls: show current files in the directory
we need to run test5.py

Step 6:
Example running:

```
cidarlabpi3@raspberrypi:~/Desktop $ sudo python3 test5.py
Enter ml: 2
Enter direction (cw/ccw): cw
Enter flow rate ml/min: 2
Please enter 'yes'to toggle solenoid. yes
Switchd
running steps: 25223
ml per step: 7.929307001875001e-05
delay: 0.0023787921005625004
timing delay: 0.0011893960502812502
direction: infusion
^CUser Keyboard Interrupt : RpiMotorLib:
Enter ml: ^CTraceback (most recent call last):
  File "/home/cidarlabpi3/Desktop/test5.py", line 82, in <module>
    ml = float(input("Enter ml: "))
KeyboardInterrupt
```

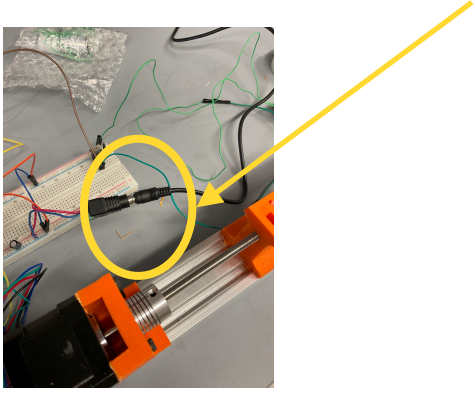
sudo python3 test5.py **<to run the program>**
Enter ml: 2 **<enter the ml you want>**
Enter direction (cw/ccw): cw **<enter cw for infusion and ccw for extration>**
Enter flow rate ml/min: 2 **<2 ml per minute. The limit of flow rate is 0 < flow_rate < 9.516 ml/min>**
Please enter 'yes' to toggle solenoid **<if nothing is entered, or anything but a yes, solenoid will not toggle>**

Then, the program will display all the helpful information.

***
If the program is running and you want the program to stop running, type **<Control + C>** to stop.
***
You may enter another round of commands to start once again. To completely exit the program:

Type <Control + C> again to completely exit the program.
Note: there may be an error message when you exit the program completely, but it is normal. The error is just telling you that you did not enter the correct value.

If the motor starts to run un-controllably after exiting the program, unplug motor power source to stop it from running.

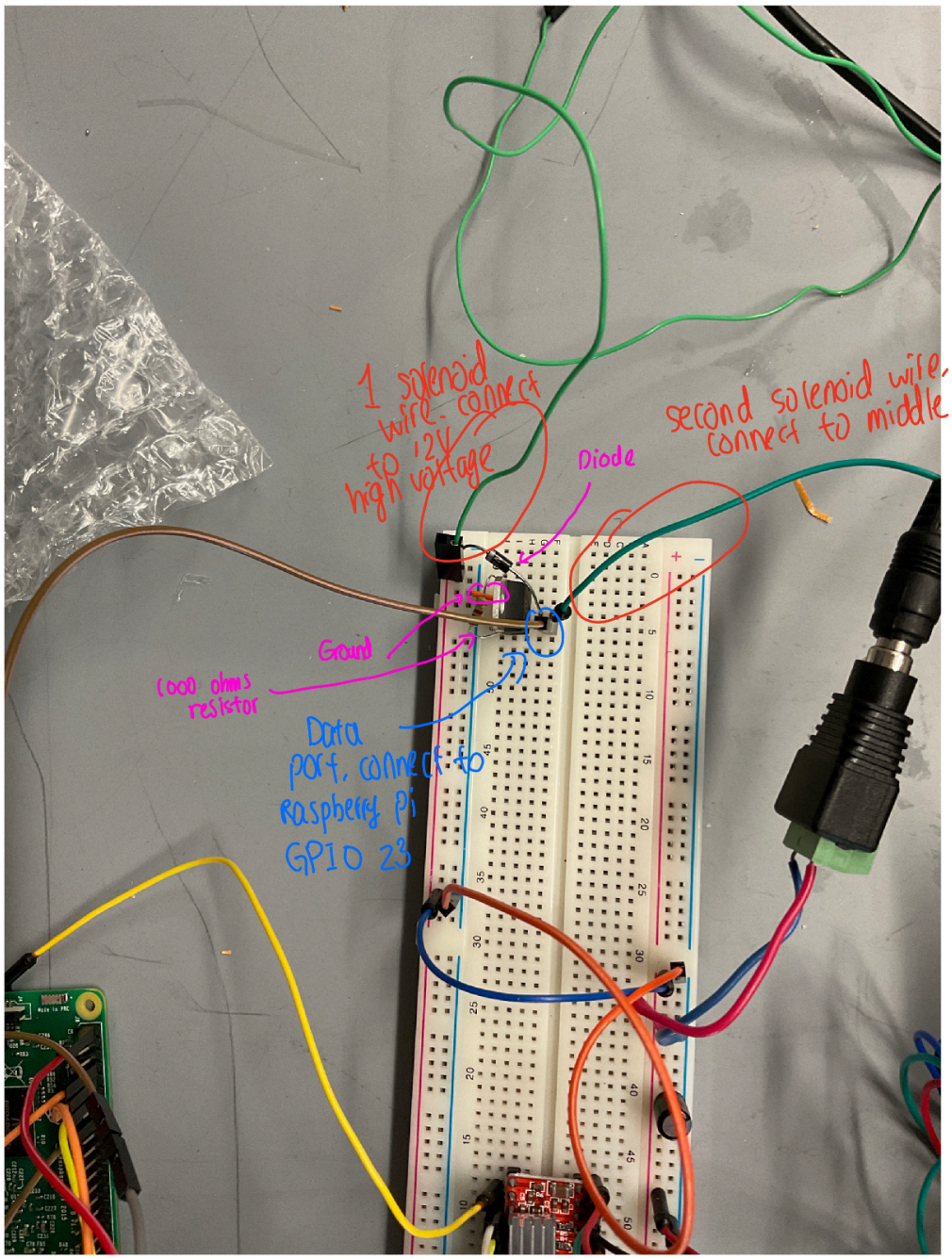


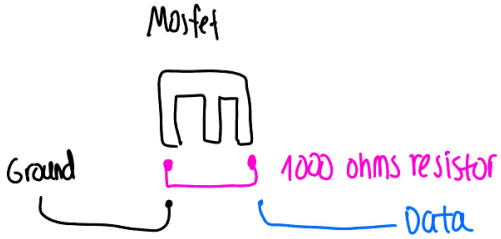
Step 7 (Shut-down):
To shutdown the raspberry pi, type:
sudo shutdown -h now

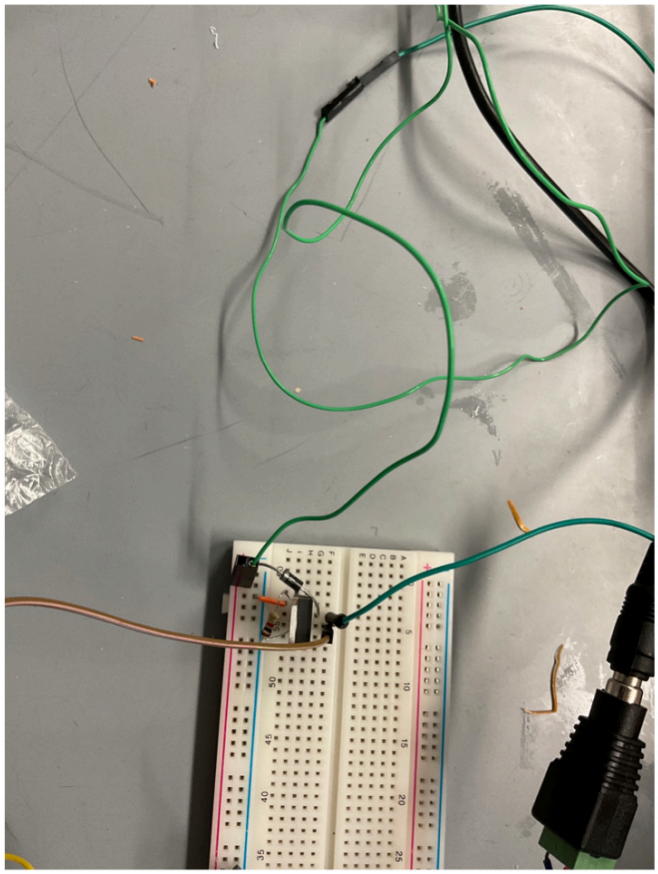
Then, unplug the raspberry pi from the power

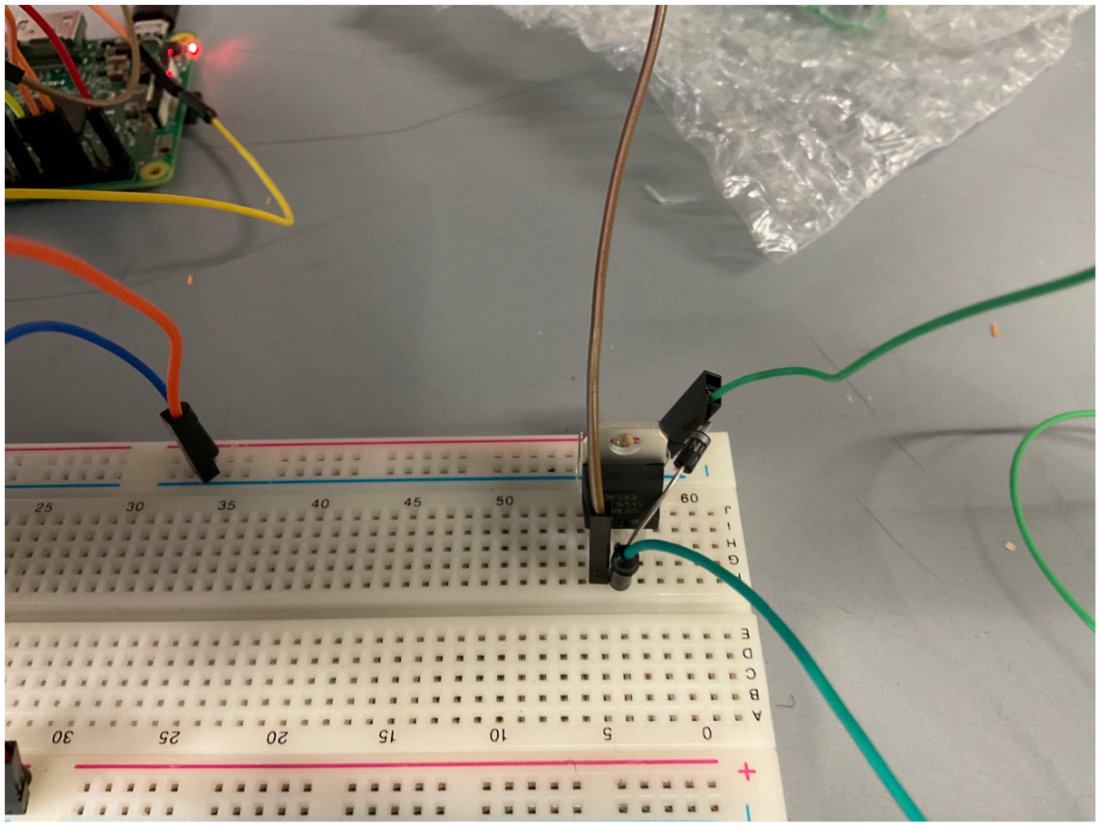
Solenoid Setup:

This solenoid uses 12V high-voltage to trigger the toggle.









Recommendations:

- Using the **same syringe and thread pitch I tested with** is recommended as it produces the most accurate result.
- It is recommended to ensure all 3D-printed parts are tight and not loose, as it will make the system lose its accuracy.
- For solenoids, the current setup makes the solenoid run super hot:
- Potential Solutions: You might use a PWM (Pulse Width Modulation) signal to effectively lower the average voltage to the solenoid once it's activated. Another approach is to use an external circuit (like a relay or a MOSFET) controlled by the Pi, which can switch between the activation and holding voltage based on your requirements. Here's a basic example using the RPi.GPIO library's PWM functionality:

Here is an example of what the code might look like in python. However, this code must be tested and implemented into the current program. If the solenoid gets too hot while testing the motor, then unplug the solenoid from the breadboard system.

```
import RPi.GPIO as GPIO
import time

solenoid_pin = 23

GPIO.setmode(GPIO.BCM)
GPIO.setup(solenoid_pin, GPIO.OUT)

# Create a PWM instance
pwm = GPIO.PWM(solenoid_pin, 100)  # 100Hz frequency

try:
    while True:
        # Turn solenoid ON with 100% duty cycle (full voltage)
        pwm.start(100)
        print("Solenoid activated")
        time.sleep(1)  # Assuming it takes 1 second to activate

        # Reduce to holding voltage (let's say 50% of activation voltage)
        pwm.ChangeDutyCycle(50)
        print("Solenoid on holding voltage")
        time.sleep(5)  # Hold for 5 seconds

        # Turn off the solenoid
        pwm.stop()
        print("Solenoid is OFF")
        time.sleep(1)

except KeyboardInterrupt:
    print("\nCleaning up...")
    pwm.stop()
    GPIO.cleanup()
```