Syringe Pump System Guide

Clayton Little

August 15, 2016

Contents

1 Introduction

The Syringe Pumps System (SPS) is a set of three programmable syringe pumps, designed to help conduct cell signaling experiments. More generally, the System regulates the concentration of ligand in a solution over a long period of time. It consists of two relatively low-precision syringe pumps (LPPs) and one high-precision syringe pump (HPP). The high-precision pump delivers concentrated ligand, while the other two LPPs flush through fresh media. The pumps are driven by stepper motors, which are controlled with a single Arduino Due, which communicates with a Python program on a connected computer.

This guide will help the reader **understand**, **operate**, or **modify** the Syringe Pump System. Throughout the guide, links to online resources are embedded in the text.

2 Physical Components

In this section, we give an overview of the non-electrical and non-fluidic parts of the SPS.

2.1 Casing

The Arduino Due, three driver boards for the stepper motors, and breadboard are all housed in a small black thermoplastic box. The breadboard makes prototyping changes simpler. The casing has five ports. One is for power, one is for controlling the Arduino, and the last three connect to the three syringe pumps.

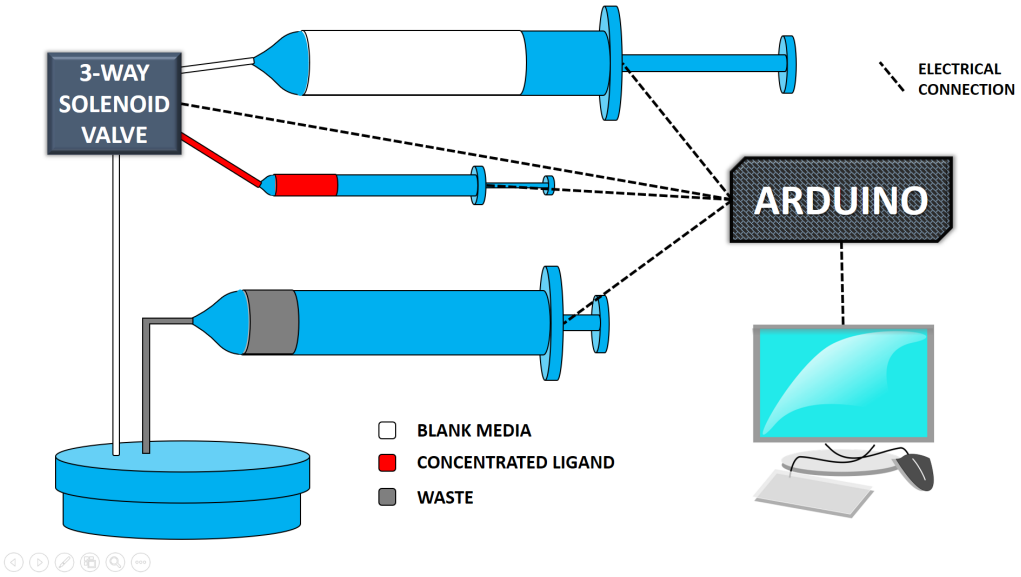

Figure 1: Diagram of all components

2.2 Pumps

All three pumps have the same stepper motor and are 3D printed with ABS plastic at the Rice University OEDK. The two low-precision pumps are mounted on aluminum rails, which are bolted together with a couple strips of thermoplastic. Our syringe pumps almost identically mimic a couple of online plans: low-precision pump, high-precision pump. We made a few changes, however. The LPP screws are turned down to 1/4in. instead of 8mm (with an appropriate coupler change), and we use a 5mm to 3mm metal coupler for the HPP instead of flexible tubing. We also use a glass syringe in the HPP, which required some changes in the the 3D printed model to fit the different dimensions. We also used a different lead screw for the HPP.

3 Electronics

4 Fluidics

5 System Code

Python sends commands to the Arduino over serial USB. The Arduino code primarily controls the motor driving and sensor reading, while the Python code controls the ligand stimulation schemes.

5.1 Arduino

There are several iterations of the Arduino code, all for different hardware setups. I will focus on the current and final iteration. Currently, the Arduino code (1) receives serial commands, (2) drives the three motors, (3) controls the 3-way solenoid valve, and (4) limits the movement of the HPP motor. Furthermore, the code is practically independent of the Python code, and can be run separately, for debugging purposes. Simply press CTRL+SHIFT+M in the Arduino window to send individual serial commands. The code is commented, and should be intuitive to understand. However, I believe this more explicit documentation to be beneficial. The structure of the code is:

1. *Defining of assigned pin numbers and declaring of variables.* Every pin used on the Arduino is given its own definition. This is useful if one ever wants to use different pins on the the Due, or perhaps use a different Arduino model. Constants that relate to the phsysical properties of the stepper motor, lead screw, etc. are all in the initial declaration. Constants

are different for the LPP vs. the HPP. The LPP has an error correction constant that was derived empirically. To add a motor, simply increase the numMtrs definition, and add a row to the pins array. This will default to a LPP.

2. *Initial device setup.* The pins on the Arduino are set up, including the interrupt pin (it is directed to an ISR called "checkLS"). The serial connection to the computer is established, for communication.

3. *The main loop.* Repeatedly checks for serial commands, and directs the Arduino to the correct routines accordingly (LPP vs. HPP). Multiple commands CAN be queued up. That is, if a command is sent while a previous command routine is still running, it is not lost, and will proceed after the previous command is complete.

4. *Motor drive routines.* The LPP and HPP routines are very similar. Both use 1/16 microsteps (to maximize precision). However, the HPP takes advantage of a limit switch (by repeatedly checking the volatile output of the ISR), and charges the solenoid in the three-way valve. There is a third routine, which drives the blank and waste syringes at the same time, but in opposite directions.

5. *Limit switch ISR.* Simply checks to see if the limit switch has been triggered. Every time the intterupt pin changes state, this ISR runs, and stores the state in a global variable, which the HPP drive routine can access.

5.2 Python

5.3 Setting up a control computer

Here I detail how to set up a computer to control the SPS. First, install the following programs:

- Arduino
- Anaconda (make sure to get the Python 3.5 version)
- Github (optional)

Then complete the following steps:

1. *Install the Due package for Arduino.* Start Arduino, navigate to Tools¿Board¿Board manager and select the Due package.

2. *Install PySerial.* This Python module will enable communication with the Arduino over serial USB. Open the Anaconda command prompt. Enter "conda install pyserial". Enter "y" for yes. Enter "conda list" to check if it installed.

3. *Gain access to the control files.* You can do this through a Github repository, or just manually copy the files onto your computer.

4. *Set the Python current directory.* Anaconda comes with a Python IDE called Spyder. This is what we will be using. Open up a console in Spyder and enter "import os" and then "os.getcwd()". This should give you some idea of how the string path is structured, regradless what OS you are using. Find the path to your control files, and change the current directory to that with "os.chdir(string path)". You only have to do this once.

6 Improvements

There are several opportunities for improvement on the SPS design:

7 Setting Up an Experiment

Make sure you have already completed the steps in Section 5.3 before attempting to set up an experiment.

1. *Make a Pluronic solution.* That is, if you do not already have some. Take the powder and dissolve it in DI water at a 5 percent mass ratio. Place the solution in 4 degrees for half an hour to increase the solubility. Vortex, and store in 4 degrees. After sterilizing the fluidics, you can flush the solution through the ligand tubing, let it sit for an hour, and then flush through with DI water.

2. *Sterlize the fluidics.* For all components, we flush with 70 percent ethanol, then flush with MilliQ. I use a sterilized syringe to drive the fluid through the tubing. To make it easier to switch the 3-way valve, short the red wires on the relay (on the breadboard). Now when you switch on the power to the device, the valve will change states, without the aid of the Arduino controls. Make sure to replace the red wires properly when you are done sterilizing. First, I switch on the power and sterilize the concentrated ligand tubing. Then I switch off the power and sterilize the blank media tubing. Finally, I sterilize the waste tubing. I take care to sterilize all the luer connectors, and immediately seal them with the luer caps (sterilize those too!) I also cap the ligand and blank syringes after sterilizing them. I am less cautious with the waste syringe. Finally, I seal the dish cap and syringes in a bag and spray ethanol inside (storing until I am ready for the experiment). During this whole process, I never disconnect the tubing at any spot other than the luer connectors.

3. *Assemble the parts in the MS room.* First, find a stable spot to place the blank and waste syringes. Ideally, they should be about level with the stage. Currently, they are in a waterproof tub to prevent leaks from damaging MS equipment. I recommend positioning the pumps so that their electrical connectors face outward. Second, place the ligand pump on the edge of the stage and route all tubing out of the chamber. Make sure the pump is placed so that the dish lid can comfortably reach the stage insert, and so that the pumps will not hit the walls of the chamber while the stage moves. I highly recommend securing the pump with putty. Third, connect the pumps to the Arduino casing. On the casing, the port closest to the power connector is the ligand pump, and the port furthest from the power connector is the blank pump. In the middle is the waste pump. Fourth, connect the power adapter to the port on the casing and connect the Arduino to the computer with a USB cable. Turn on the power and make sure the LEDs on the Big Easy Drivers light up.

4. *Testing computer controls.* Start Spyder and open the fluidmaster.py file. Upon running the file, you will be presented with a GUI to control the SPS. I recommend running a dummy experiment to see if the controls are working. Make sure the motors are positioned far from their limits, so you don't overextend the pump.

5. *Setting up the fluidics.* First, connect the ligand syringe to its tubing. Power the solenoid valve. Pump the ligand through the tubing until you are confident it has filled the valve cavities. Unpower the solenoid valve. Remove the syringe and retract the plunger (you probably have no more ligand left in the syringe, but you still need air volume to drive it). Second, flush the blank media through its tubing until it comes out of the inlet. Place all syringes in their pumps.

6. *Integrating the cells.* At this point, you may want to sterilize the inlets, because they probably have been bumping around and hitting things. Place the lid on the dish with your cells.

7. *Final tasks.* Set up and start your imaging run before starting the pumps controls. This way you can precisely time the pump transfers and washes between the imagings.