AUTOMATIC TITRATION MACHINE

By

Jason Flanagan

Jack Viebrock

Matthew Weyrich

Final Report for ECE 445, Senior Design, Spring 2024

TA: Selva Subramaniam

1 May 2024

Project No. 59

Abstract

This automatic titration machine will reproduce the mechanisms and functionality needed to perform a manual titration. The system will dispense chemicals into a beaker through a syringe driver, record the pH of the solution with an electrode, and display this information back to the user.

The final design of the titration machine successfully includes the following subsystems: a working motor, microprocessor control, power, and user interface. However, the pH sensor was not fully implemented and intended functionality can be replicated with a voltage source.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1 Statement of Purpose

Titration is used in many areas of industry, research, and for home projects. The market is very limited on the types of titration devices, with on one end we have manual titration, and the other is a several thousand-dollar automatic machine. The goal of our project is to bring a product to market to bridge this gap with something on the low end of the hundreds of dollars range and a device that has more accuracy, precision, and is quicker than manual titration techniques.

1.2 High-level Requirements

- 1st Requirement (Precision): Repeat titration with only +/-0.5% deviation between measured titration endpoints
- 2nd Requirement (Speed): Perform a titration as fast or faster than five minutes
- 3rd Requirement (Accuracy): Measure pH with the pH probe within +/-0.056 (+/- 0.02 V of the correlated voltage; 0 V +/- 0.02 V for a pH of 0, and 5 V +/- 0.02V for a pH of 14)

These high-level requirements come from looking at the current products in the titration machine market. We wanted our design to be competitive with the high precision offered by them, thus the tolerances given in the first and third requirements have small ranges. The timing requirement is to ensure that this product can provide a benefit over manual titration, which can take a long time for setup and constant manual readings of data.

1.3 Block Diagram and Visual Aid

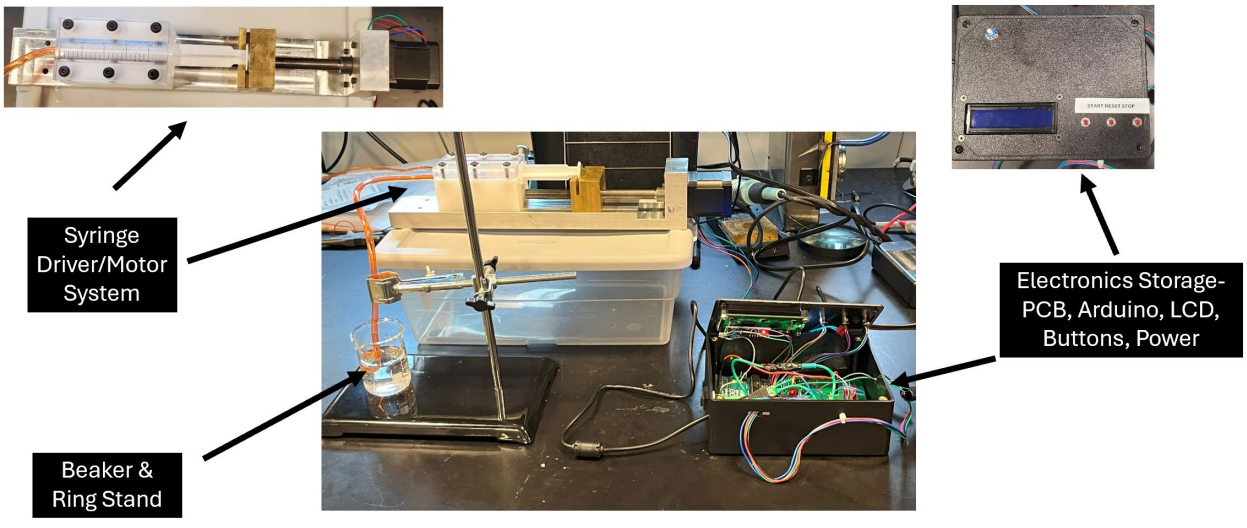

Figure 1: Final Demonstration Setup

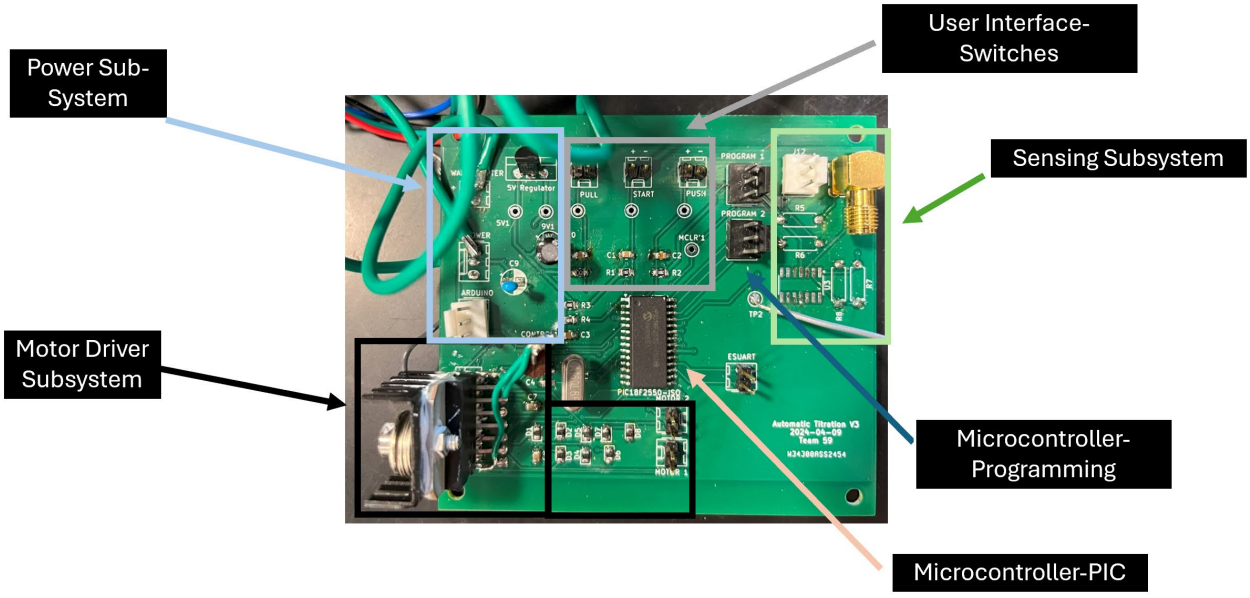

Figure 2: Final Assembled PCB

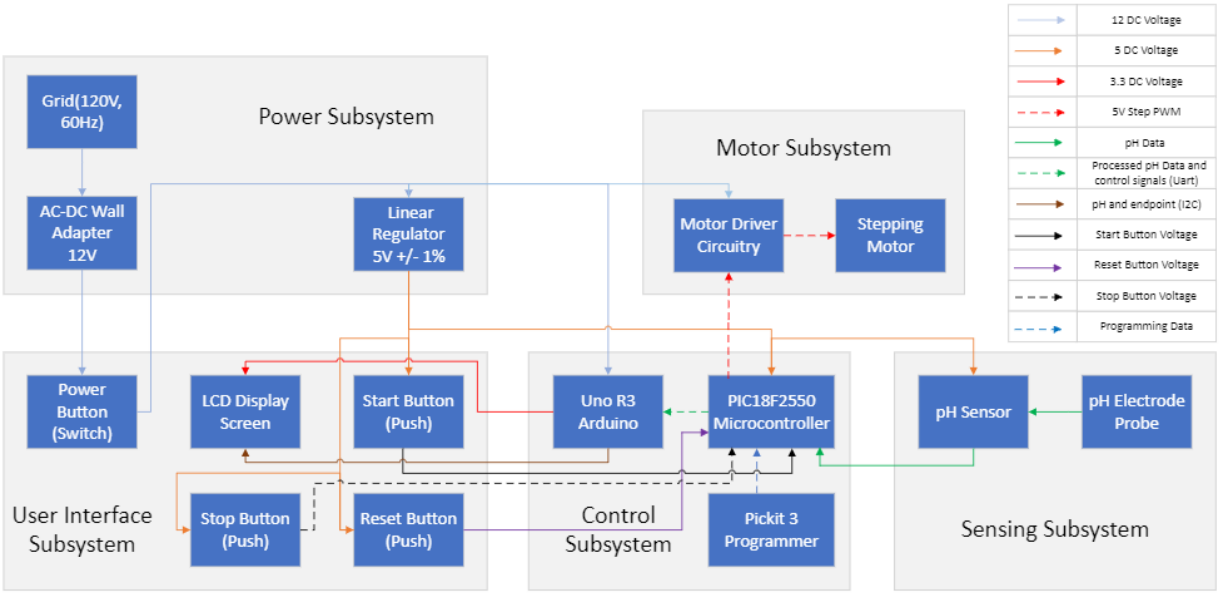

Figure 3: Diagram of Titration System Consisting of The Five Subsystems: Power, Motor, Sensing, Control and User Interface

1.4 Subsystem Overview

1.4.1 pH Sensing

The sensing system's main purpose is to read the pH of the current solution. This will be achieved by using an electrode probe to determine the pH of the solution. Since the electrode outputs miniscule amounts of voltage, a pH sensor is needed to amplify the signal and add a DC offset. The pH sensor will

be powered by the power subsystem and will send the data from the pH sensor to the microcontroller in the control subsystem. In Figure 1, the pH electrode is missing from the demonstration setup because it was not working as intended.

1.4.2 Power

This system converts the grid's voltage to 12 V used to power the Arduino motor, and connects to the linear regulator. The system's linear regulator is implemented to step-down the DC voltage to 5V, allowing the power subsystem to provide power to the microcontroller and pH sensing circuitry.

1.4.3 Control

This subsystem contains the pic18f2550 microcontroller and an Arduino Uno r3. The microcontroller is used to acknowledge the push buttons from the sensing subsystem and to control the motor through four output signals. The microcontroller receives a 0 V to 5 V value from the pH subsystem, converts it into a 10-bit digital value, and then sends the data to the Arduino through UART. The Arduino is used to store that pH data and to display it on the LCD screen from the User Interface subsystem through I2C.

1.4.4 Motor

Contains the stepping motor used to control the speed of release from the syringe containing the titrate. This system also utilizes a half bridge driver and other circuitry to amplify the voltage and current that is being supplied by the microcontroller in the control subsystem for the stepper motor as seen in Figure 2.

1.4.5 User Interface

Shown in Figure 3, this module contains an LCD screen, a start button, a reset button, a stop button, and a power button (Emergency stop). The reset button is used to fill the syringe by pulling the plunger of the syringe out of the barrel until it reaches the pre-determined endpoint. The start button is used to initiate the titration and to stop moving once the syringe has completely closed. The stop button is used to interrupt the start and reset motion of the syringe driver. The user is then able to decide to continue the previous act or return to the initial position by pressing the opposite button. The LCD screen is used to display the data being stored by the Arduino. The power button is used as an emergency stop switch in case anything goes wrong or to cut the power when the system is not in use.

2. Design

2.1 pH Sensing

2.1.1 Design Procedure

A titration is performed to calculate the equivalence point of the reaction—the point in which there are equal moles of acid and base present. This causes a sharp change in pH from acidic to basic, where the equivalence point is defined as the point of maximal change in pH. Manual titrations approximate this point by using an indicator that changes color to a “light pink” when the solution transitions from acidic to basic. This introduces inaccuracies in a person’s choice of what color is acceptable as “light pink”, so we considered using an optical sensor to detect this color change. Another way to determine the pH is to directly measure it with a pH electrode. This would give us a more exact equivalence point, but electrodes are costly and extremely sensitive. We decided to go ahead with using an electrode over an optical sensor because our project is focused on precision and the extra cost is worth the additional functionality.

A pH electrode works by emitting a voltage in the range -400 mV to +400 mV on top of a supplied reference voltage [1]. If the reference voltage was ground, then a pH of 0 would be +400 mV, a pH of 7 would be 0 mV, and a pH of 14 would be -400 mV. This voltage swing is a small range and needs to be amplified to match the 5 V maximum input on the microcontroller’s analog-to-digital (ADC) pins. The wider the range of voltages we can get from the pH electrode, the more precise our pH reading can be.

2.1.2 Design Details

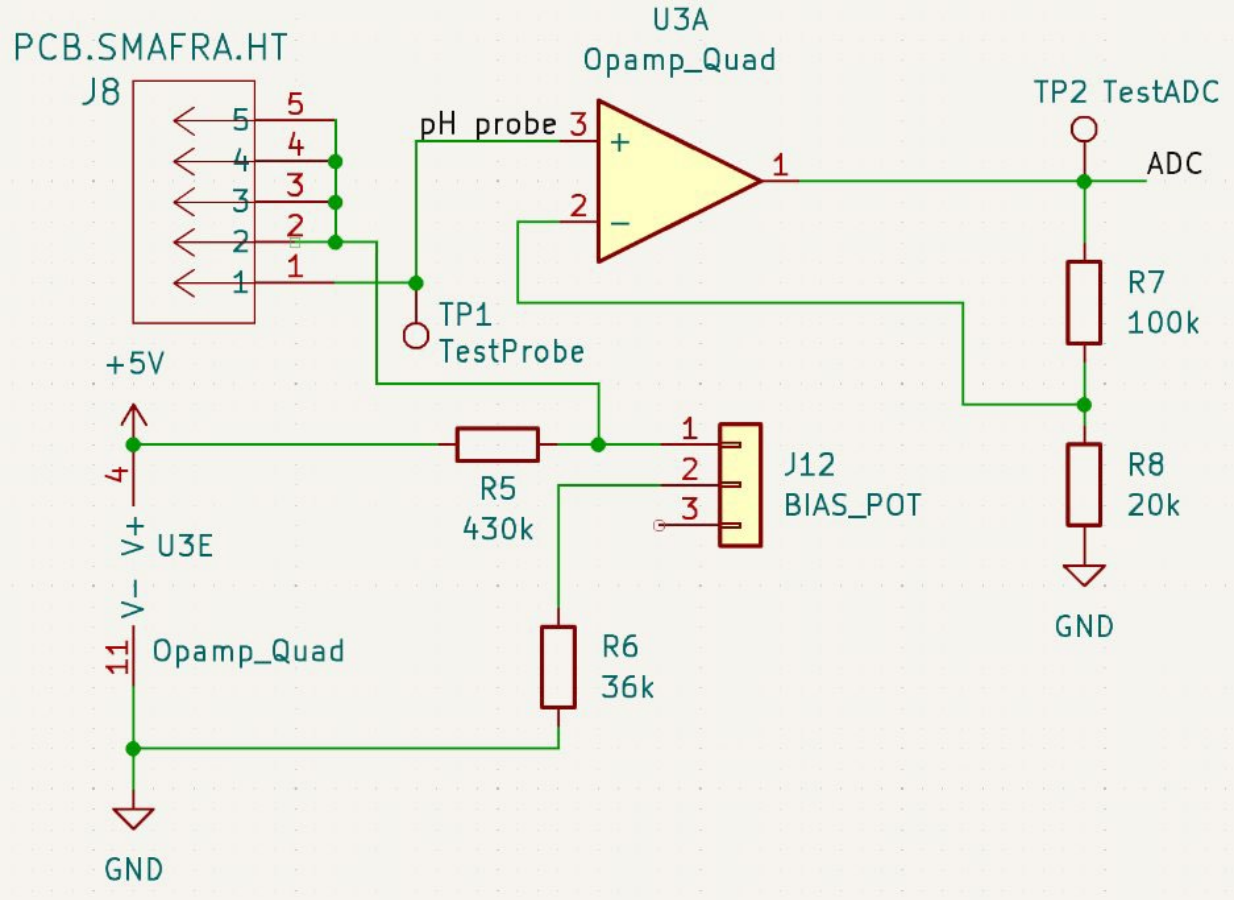

Figure 4: pH Senor Subsystem Schematic

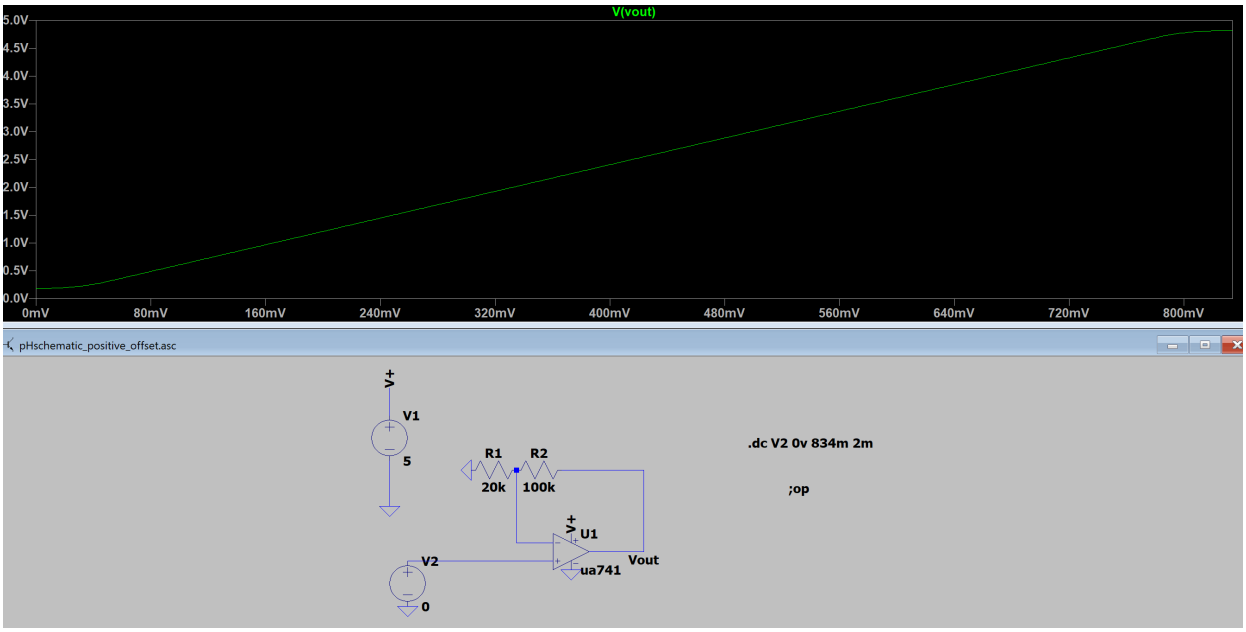

Figure 5: Simulation of pH amplifying circuitry

From Figure 4, there is a voltage divider circuit to supply the reference voltage to the electrode. This voltage divider uses a potentiometer that is routed to the top-left of the electronics enclosure, shown in Figure 1. Then, using calibration solutions of pH 4.00 and 7.00, the electrode's reference voltage can be tuned to account for any temperature variance in solutions when performing experiments.

The circuit also features an amplifying circuit using an operational amplifier (op-amp) to maximize the use of the ADC's 10 bits. If we kept the voltage output as 0 mV to 800 mV, then that would only correspond to 164 unique values for pH over the entire 0 to 14 range. However, as shown in Figure 5, using a non-inverting op-amp configuration with the resistance values 100 kΩ and 20 kΩ, the 800 mV gets amplified to 4.816 V. It can also be seen that a 0 V input does not equal a 0 V output. This is because op-amps cannot perfectly output voltages near the supply voltages, 0 V and 5 V. Thus, we made sure to specifically look for a product that could have an output swing better than the provided op-amp model in simulation. This resulted in us finding the LPC660 op amp, which features a typical output swing of 0.040 V to 4.940 V. This will enable us to make full use of the 10-bit ADC to achieve the precision we want with our high-level requirements.

2.2 Power

2.2.1 Design Procedure

The power subsystem is necessary to power the entire PCB and the Arduino Uno r3. The main design choices for this subsystem relate to what voltage and current the wall adapter should output and how much current the 5 V linear regulator should output. Since the Arduino can safely accept between 7 V and 12 V, it was preferable to stay within this margin to avoid using another linear regulator. This also meant that the stepper motor had to be able to operate at or under 12 V. The wall adapter also must be able to supplement the extra current draw that comes from the motor operating. The linear regulator also had to act as a current buffer to the microcontroller.

2.2.2 Design Details

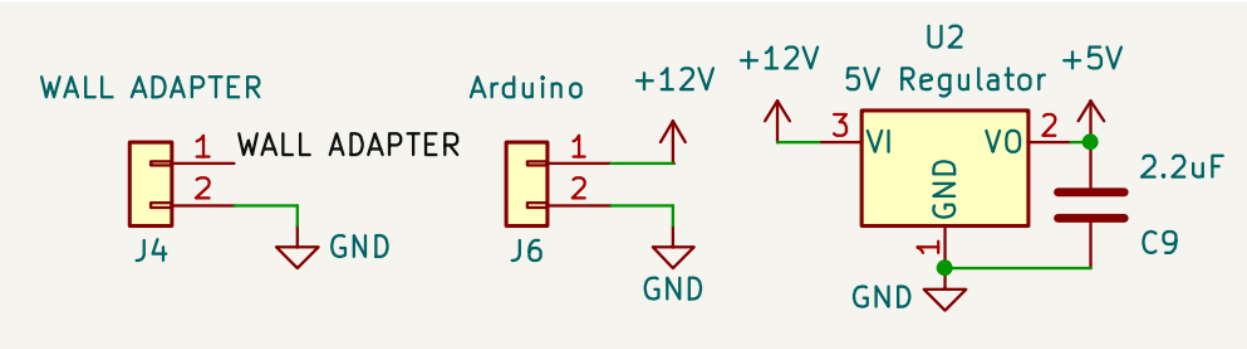

Figure 6: Power Subsystem Schematic

A 12 V wall adapter was selected to power the PCB and Arduino. Most of the stepper motors that the team looked at had a minimum voltage of 12 V, allowing the wall adapter to power the Arduino and the motor. In the end, a 12 V, 4 A wall adapter was settled on. A previous 12 V, 2 A wall adapter was purchased, but didn't output enough current for the stepper motor. This was unfortunate since the wall

adapter wouldn't surpass 0.5 A, even with its 24 W power rating. To be safe, a more robust wall adapter was selected.

The PIC18f2550 requires 5 V to operate, so that was the voltage the linear regulator had to output. On top of that, the VDD pin on the PIC18f2550 can accept a maximum current of 250 mA. To avoid any possible damage to the microcontroller, a linear regulator that outputs at most 250 mA was selected. The 5 V source was also connected to the three push buttons and the pH circuitry, which each took in at most 0.5 mA of current at one time. The original design did not include the 2.2 µF capacitor in Figure 6. Without this capacitor connected to the output of the linear regulator, the voltage wouldn't stabilize, resulting in readings of about 4.5 V.

2.3 Control

2.3.1 Design Procedure

The control subsystem is composed of two main components: The PIC18F2550 and the Arduino Uno r3. The PIC18f2550 was selected because it was one of the provided microcontrollers in the electronic services shop. The microcontroller needs to be able to take in four outside inputs: the start button, push button, pull button, and the current pH reading from the sensing subsystem. When one of the buttons is pressed, the PIC will begin controlling the motor through the stepper driver circuitry. The microcontroller also needs to be able to output four control signals to the motor subsystem to dictate the type of steps being taken and when they are taken. The PIC uses its 10-bit Analog-to-Digital converter to create a pH value that can be sent from the PIC to the Arduino for further processing and storage. The plan was originally to use I2C to connect the two components, but that transfer protocol is needed to connect the Arduino to the the LCD screen. The Uno r3 was picked since the Arduino was already owned by one of the group members. The Arduino also interfaces with the LCD screen through I2C to display the current pH of the solution.

2.3.2 Design Details

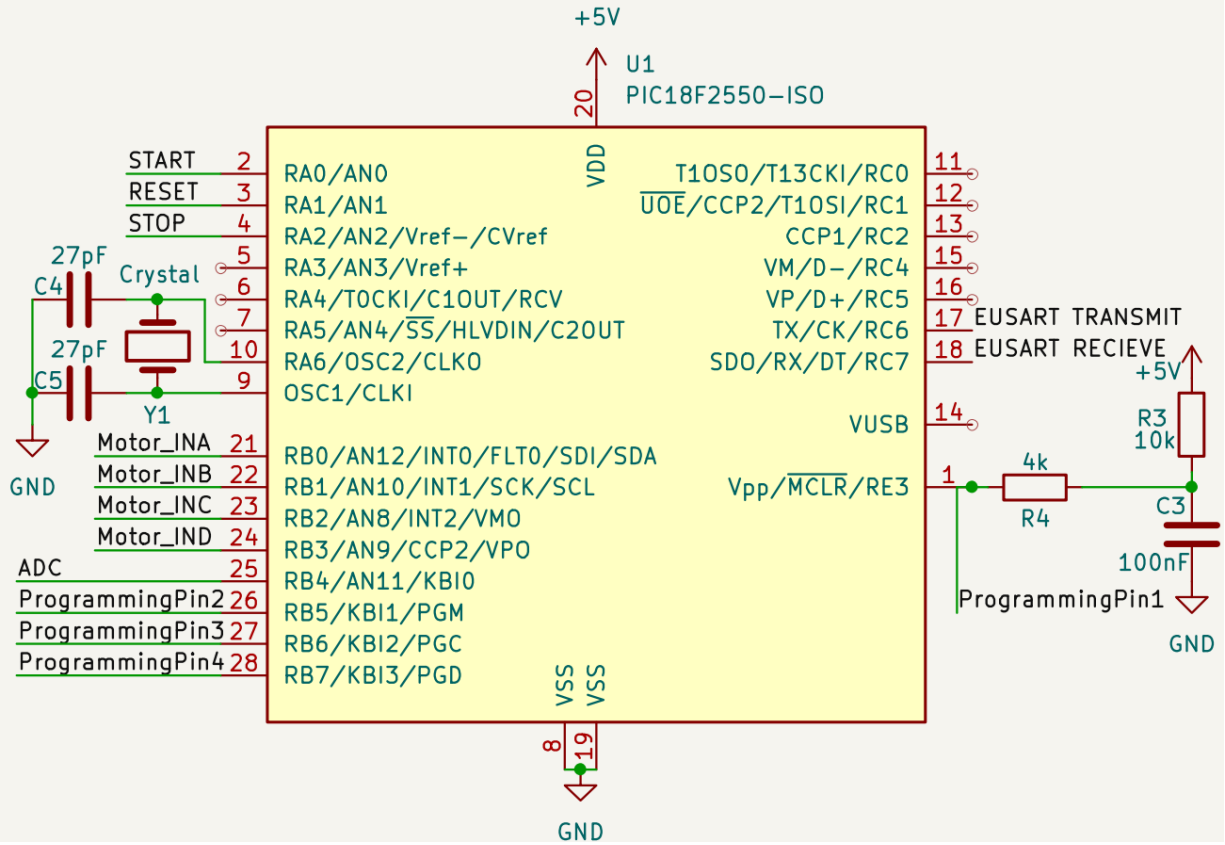

Figure 7: Control Subsystem Schematic: PIC Microcontroller

The microcontroller controls the motor by sending out four signals in a piecewise manner to the stepper driver circuitry. At one point, only two consecutive motor pins will be on at a time. The main code on the PIC revolves around how the system receives outside inputs, how long the syringe driver is operating for, and for sending the pH data through UART to the Arduino. When the system is in the idle state, pH data is sent to the Arduino every second. The PIC is also checking to see if either the start or reset button has been pressed. If the syringe is completely closed, then the only button that will change the state is if the reset button is pressed. When that happens, the syringe plunger is slowly pulled out of the syringe barrel. This act will either execute to the full extent of the syringe and stop when it reaches the 20 mL mark, or it can be stopped by clicking the stop button. At this point, the reset button can be clicked to finish the process, or the start button can be clicked to initiate a titration that doesn't need the full 20 mL of titrant. Whatever button is pressed will return the syringe driver to either the open or closed state, without pushing the plunger too far into or too far out of the barrel.

The Arduino is connected to the PIC through UART. The microcontroller transmits the 10-bit pH data that was converted using the Analog-to-Digital converter. The easiest way of connecting the two components was by using a baud rate of 9600 and a frequency of 16 MHz. As seen in Figure 7, an external 16 MHz crystal is used to generate this frequency. Once the Arduino receives the data, it is stored into a buffer if the syringe driver is currently in the start state. These states are sent in extra UART packages

when they are entered and exited. The 10-bit pH value is manipulated into the range of 0-14 pH before being displayed to the LCD screen through I2C.

2.4 Motor

2.4.1 Design Procedure

There are many design approaches to the motor subsystem because at its simplest approach it needs to deliver varying chemicals and quantities to a beaker setup. We first explored whether a pump or motor driver would be ideal for this application, due to the budgetary and mechanical restrictions, a precise and accurate pump would be too costly. With our sights now set on a motor inclined machine we understood any design would require a mechanism to convert the motor spin to a linear motion. Additionally, we needed to decide what type of motor would fit our application best, our research cornered the stepper motor as we could control its movements with precision as we direct movements with each pulse. This configuration also made the resource of control lighter as we could use an open loop design by coding in pulses per movement. These limitations and decisions led us to a syringe driver which would need to push and pull liquid, we brainstormed a variety of setups from vertically mounted right above the beaker to use gravity to our benefit or a horizontal driver over the beaker or horizontal but looped in with tubing. Our ultimate design was influenced by the machine shop who came back to us after finding an old driver mechanism, which came with its own issues, and set us up for a horizontal driver routed with tubing. We would discover that the motor implemented was geared and the syringe too large for the application and more prone to get air bubbles, fortunately the machine shop was able to transform it into the driver seen in Figure 1 with a smaller syringe and stepper motor.

Our next design consideration was driving the stepper motor control, we had decided to purchase an external motor control module due to complexity. When conferring with our TA we discovered the necessity to get this circuitry on our PCB for wave five. This is when we switched to the L298N Half Bridge IC to drive the system for our final design in Figure 2.

2.4.2 Design Details

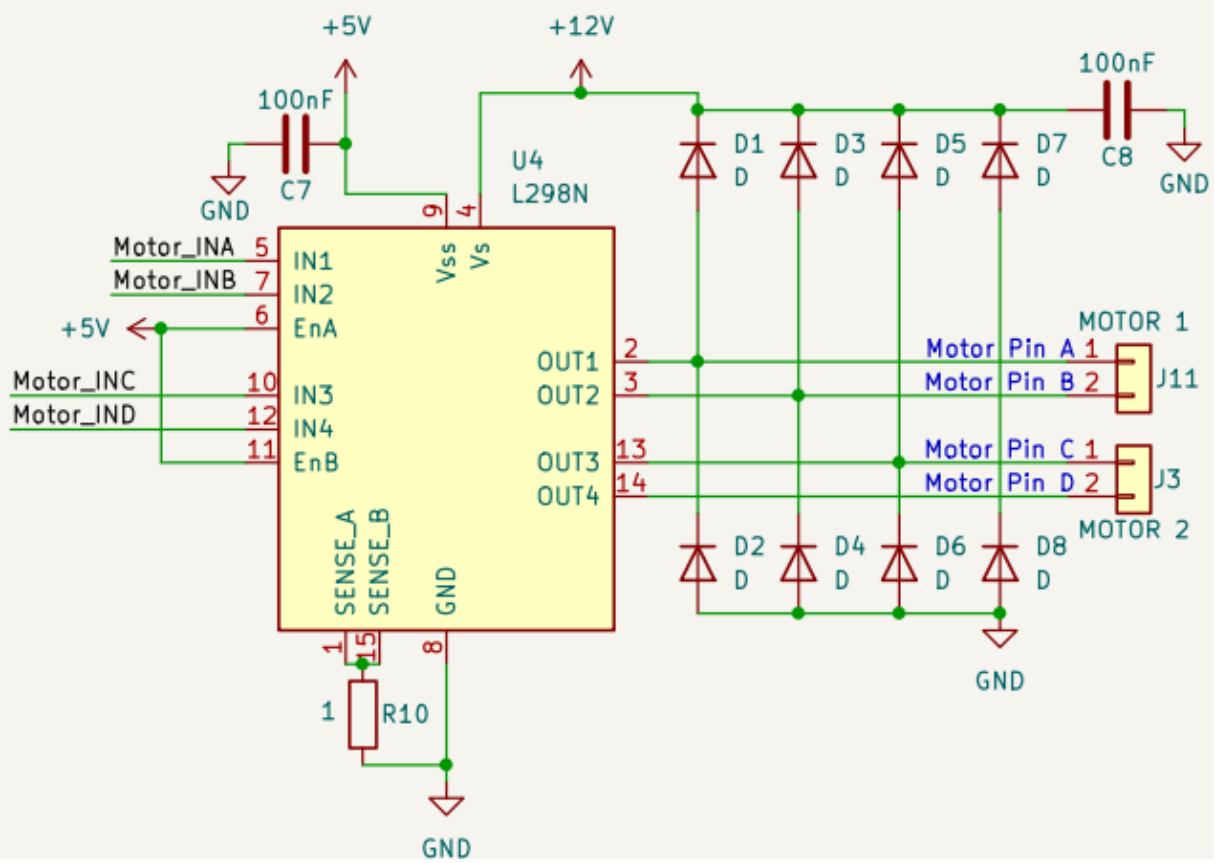

Figure 8: Motor Subsystem Schematic

The finalized stepper motor driver circuitry can be seen in the schematic featured in Figure 8 which features the L298N half bridge chipset, the design of circuitry was heavily influenced by the application section of the data sheet [2]. In determining the motor, we attempted to calculate/find the force required to push a syringe and try to account for the torque needed to overcome the components that supply the linear motion as well. This led to us research syringe size and force. We had decided on the 20ml syringe due to the tolerance analysis seen in our design document, which led to a study which indicated that 33N of force was necessary for the plunger [3]. Additionally, we had to account for some of the resistance in the linear components which would be difficult without specifications handled by the machine shop, so we added at least 10 N of force as F=ma with the addition of friction we are looking at a threaded shaft around one pound and the block it is driving around .75 pounds, as seen by similar products. Using the force equation, we see that with a relatively slow acceleration (seen due to high level requirements) we get a relatively low force of less than one and the aluminum having a coefficient of friction of friction around 0.47 we can determine that the 10 N of force is overkill in having enough cushion for whatever the machine shop supplies. While the force to torque equation is force times radius we were unsure of the specs of the mechanism we would receive and applied a small radius due to the shaft and the power would be force times distances to get us around twenty watts needed and the main motors having power close to twenty-four watts in the lower range. This led to us researching the varying stepper

motors out there that could fit our voltage and power needs to see what options were available, we saw that main applications dictated a lot of the supply with most for 3D printing and CNC machines of varying size and noted that the cost was relatively low for this component. With this in mind it led us to the NEMA 17 Stepper motor rated at 12V to 24 V, 2 A, 200 steps per revolution and 59 Ncm holding torque and the power rating would fit our needs. We knew that the calculations would be off due to the linear driver and that the only way to be sure about the motor was for testing with the machine shop supplied device. Thus, for our needs it led me to choose the lower motor used for 3D printing and low power CNC providing a good middle ground in power concerns and noted that more options were easily assessable if testing proved more power was necessary.

2.5 User Interface

2.5.1 Design Procedure

This subsystem consists of three push buttons, a switch, and an LCD screen. The three push buttons were designed to be active-low. If the buttons were active-high, then the PIC would run into a few problems. There isn't a direct path for the voltage to drain once an active-high button has been pressed. To reduce any complexity, the push buttons were designed to be active-low. The switch acts as an emergency shutoff and just a basic power button. The LCD screen has already been discussed in the control subsystem but was selected to display the current data of the pH for user ease. That specific 12x2 character screen was selected because it was obtainable from the lab.

2.5.2 Design Details

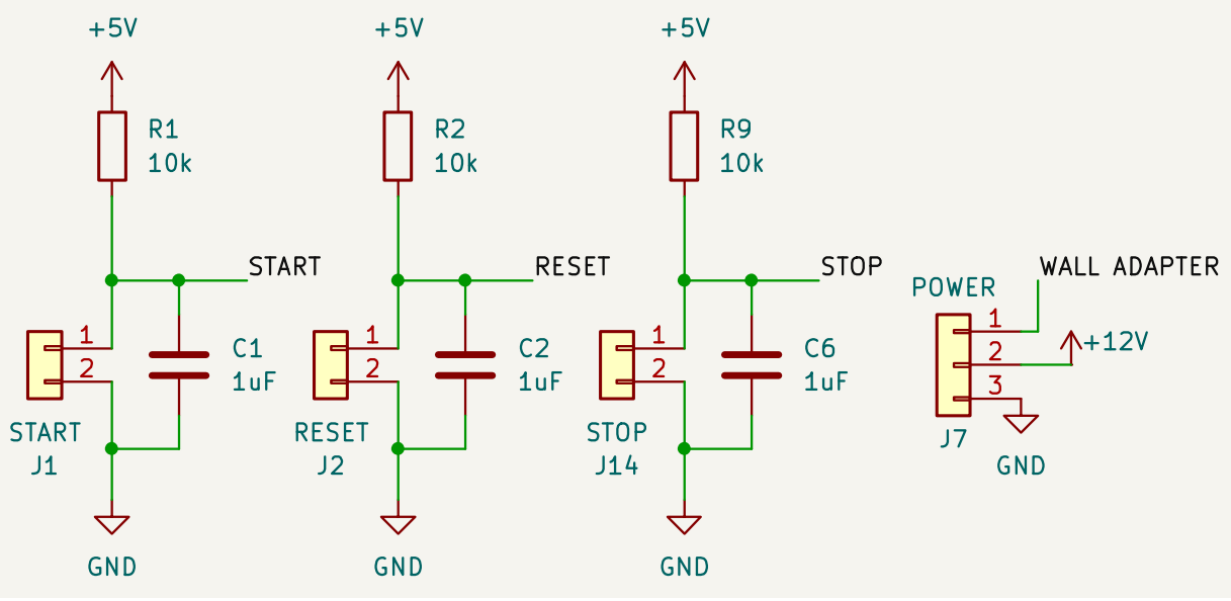

Figure 9: User Interface Subsystem

Other than why the push buttons are active-low, the other main component for the push buttons were the capacitors that can be seen in Figure 9. This capacitor is there to help with a bouncing signal. When a push button is pressed, it may bounce multiple times before settling. By adding in this capacitor, the bouncing signal will be rounded out, reducing any chance of a false signal.

3. Design Verification

Each subsystem was verified by performing measurements and checks outlined in the Requirements & Verification Tables 3-7 in Appendix A.

3.1 pH Sensing

The pH sensing subsystem was first verified by checking if the amplifying circuitry performed as expected from the simulation in Figure 5. This aspect did work as intended, with an input of 0 V outputting 0.05 V and an input of 0.8 V outputting 4.9 V. This is within the tolerances we wanted our amplifying circuit to output and matches the expected output swing from the op-amp's datasheet.

After this initial verification, we switched focus to motor operation and left the pH circuitry alone. Then, when we tried reading the voltage values reaching the microcontroller, we saw a constant low voltage of 0 V or 0.53 V on the amplifying circuitry output regardless of the input voltage. This shows that the op-amp was getting damaged somehow during normal operation and required resoldering. A new op-amp chip was soldered on and the circuit still exhibited the same behavior. This indicated to us that the connection from the op-amp to the ADC of the microcontroller was causing the op-amps to be damaged.

To prevent further damage on op-amps, we connected the output of the pH electrode straight into the ADC of the microcontroller. This would no longer amplify the voltage, but we wanted the project to maintain pH sensing functionality if possible. However, we were not able to verify any voltage readings from the pH electrode. Using a multi-meter showed a constant 0 V output from the electrode regardless of the solution it was submerged in or the supplied reference voltage. The final design for demonstration purposes did not include any pH electrode because of these issues and instead featured a voltage source that could go from 0 V to 5 V to mimic the intended functionality of the amplifying circuitry when it was working independently.

3.2 Power

To verify that the power subsystem was working, we took voltage and current measurements throughout the PCB. Voltage measurements are taken in parallel with components, and current measurements are taken in series with components. We verified that the wall adapter was able to output up to 12 V with a peak output of 12.24 V and supplied up to 0.8 A of current while all components were on and active. The last verification for the power subsystem is checking that the voltage regulator worked as intended. We need the 12 V to get stepped down to 5 V, and the output of the 5 V regulator was measured at 5.04 V. This is within the tolerances we needed for the microcontroller and other chips so this subsystem passed verification.

3.3 Control

The first part of verifying the control subsystem was making sure the plunger was never pulled out of the barrel of the syringe. Since the start and reset states are based on an internal counter, the value of that counter must be manually adjusted to find the optimal time. To do this, the counter was set to a relatively small value. Since the syringe was in the closed state, the reset button was pressed. The distance the syringe moves was recorded and compared against the total distance the syringe needs to move. The counter was set to 500 and the syringe moved about 2 mL during that time. The number was then

multiplied by 10 to get the full 20 mL of the syringe. In the end, this was a little low, which meant the counter had to be slightly altered to utilize the complete syringe.

Table 1: Calibration Test

Steps per Revolution	Sets of Revolutions per 30 seconds	Drops per 30 seconds	Operating cycles per Revolution	Wait cycles per Revolution	Duty Ratio (%)	Volume (mL)
10	32	17	16,500	330,000	5	3.10
10	33	16	16,500	330,000	5	
10	31	14	16,500	330,000	5	
20	32	33	33,000	316,800	10.42	N/A
20	32	33	33,000	316,800	10.42	2.20
18	32	43/44	29,700	285,120	10.42	2.40
16	39	43	26,400	253,440	10.42	2.60
14	44	44	23,100	221,760	10.42	2.59
14	44	42	23,100	221,760	10.42	2.00
14	44	27	23,100	221,760	10.42	1.90
14	44	43	23,100	221,760	10.42	2.50
14	44	38	23,100	221,760	10.42	N/A

The next requirement was to get 30 drops of water for 30 sets of revolutions. To test this, we set up a 30 second timer to get an average number of drops and revolutions for that timeframe. All this data is recorded in Table 1. When referencing cycles, this is referring to the number of “for loop” cycles that are being used as delays. Testing over 30 seconds was necessary because there are instances of two drops happening during one revolution, and other instances where zero drops during one revolution. By adjusting the number of steps taken in one revolution, the number and or size of drops would also change. We started out at about half the number of drops for the number of sets. By doubling the duty ratio, the number of drops was around the number of revolutions, except in a few strange cases. The first strange case was when there were 18 steps per revolution. There was a large increase in the number of double drops per revolution. Although there were so many more drops, the size of each drop decreased. We settled on 14 steps per revolution. We have some recordings that were in our favor. We either recorded the same number of revolutions to the number of drops, or an amount within +/- 2 drops. However, there was one case where the number of drops decreased drastically. This is due to the inconsistency of the syringe driver. Depending on where the threaded block is in relation to the motor, the amount of torque needed to push the block changes. There is also varying amount of friction along the blocks track. We attempted to decrease resistance by applying WD-40 to the threaded rod and the base. In the end, we had instances of having one drop per revolution, but the inconsistency wouldn’t allow us to find an optimal value.

After receiving the pH data from the PIC, the data is displayed on the LCD screen. The LCD screen must be verified by sending test messages to the screen. This is all done using I2C. Setting up I2C wasn't too hard to implement since we used the Liquid Crystal Library and the Wire library.

UART was used to connect the PIC to the Arduino. This is the pH reading that will be displayed on the screen. Since the LCD screen was already connected, the value is able to be printed on the screen for easier debugging. This was tested out by sending test messages from the PIC along with the pH to test the refresh rate. Since the pH value was 10-bits, it also had to be sent in two different packets and re-assembled by the Arduino.

3.4 Motor

This was a multipart verification process, we had to first ensure that we would get enough voltage and current draw from the supply, ensure the motor had enough power to ensure movement, and calibrate the system for chemical drop sizes and timing. The machine shop also had concerns over the strength of the stepper motor to replace the geared motor in the driver they found so we set up a simple experiment using the motor, shoelace, motor driver with Arduino control, power supply, and syringe. We hooked up the lace to the motor's rotor and the other end to the syringe plunger. We started the system using a 12 V supply and saw the motor pull the syringe with every test. We implemented varying loads by adding resistance at the syringe opening to ensure it could overcome the pressure and set the syringe up to where we added more opposing force to the motor to account for the linear motion driver. Additionally, we considered how hard the motor was working, under its heaviest load test we saw it use 12 V and a maximum of 0.6 A which left a lot of room for the load's power draw as it is rated for 2 A. Thus, every test yielded the same result, that the motor had sufficient power and torque to move our syringe and could manage some extra force for the linear mechanism.

Once the machine shop implemented the changes, our next test of concern was when the motor was mounted to the linear syringe driver. We used the 12 V power supply from our locker, the stepper motor driver external module, 20 mL syringe and microcontroller. This allowed us to confirm the functionality of the motor over the full load, as we also pulled water into the syringe and pushed it out to see the drop functionality, in addition to ensuring our microcontroller could handle the control. We in fact saw the expected function of the motor system and saw a high draw of current up to 0.8 A, we marked the highest current as our only data point as this was a qualitative experiment.

At this point we needed to change the driving and control scheme of the stepper motor to be included on the PCB which is where the L298N chip came in. When first implementing the same test as in the last paragraph we noted the chip getting too hot. Due to this we decided to apply thermal paste and add a heatsink, as seen in Figure 2, in order to alleviate the thermal buildup. This in fact bought us more time for movement but still the IC was holding too much heat and eventually gave up and started staling the motor. This led to some more research about the chip which was built with older BJT technology having more current flowing through and being lost in the chip, the way to combat this problem was to change how the motor was run. We needed to do more bursts with faster speeds to ensure the current was passed to the motor. When implementing this we saw a huge alleviation of strain in the IC chip which was cool to the touch.

With this issue averted we were able to start in on calibrations, this led us to run the circuit to fit figure out steps to distance ratio which is detailed in Table 5. Additionally, we wanted to calibrate the drop sized to the step rate, the quantitative date is detailed in Table 1. With this we see variations in the drop rate even with the same revolutions which presented a new problem. We noticed that the change occurred around the same spot of the linear movement, we tried lubricating the threaded shaft and gliding surface but found that the issue came from the way the linear motion was created. Due to the crude setup the block being driven was pushing into the metal bottom in combination with the threaded rod moving left-right and up-down as only one side was secured. Due to limitations with the machine shop, our idea to put stabilization bars over the top to better guide the block was not able to be implemented.

3.5 User Interface

We verified the user interface from the correct response of the buttons and the ability to display information on the screen. All three buttons (Start, Reset, and Stop) showed working functionality with the motor subsystem. The screen was able to display text and any data sent from the microcontroller to the Arduino.

4. Costs

4.1 Parts

Table 2: Parts Costs

Part Description	Manufacturer	Part #	Quantity	Cost
Gen 2 Mini Lab Grade pH Probe	Atlas Scientific	#ENV-20-pH	1	$60.99
20 mL Plastic Syringe	Betevie	BETUS-LTL	10	$8.48
12V DC 4A POWER SUPPLY Wall Adapter	Signcomplex	TZF120A-1204000-F	1	$17.98
+5 Voltage Regulator	STMicroelectronics	L4931CZ50-AP	1	$0.93
LCD 12x2	ECE 445 Lab Cabinet	N/A	1	$0.00
STEPPERONLINE Nema 17 Stepper Motor Bipolar 2A 59Ncm(84oz.in) 48mm Body 4-Lead W/ 1m Cable and Connector Compatible with 3D Printer/CNC	STEPPERONLINE	17HS19-2004S	1	$13.95
PCB.SMAFRA.HT	Taoglas Limited	931-1361-ND	1	$3.54
Quad Op-Amp	Texas Instruments	LPC660	1	$0 (Electronic Services Shop)
External potentiometer	Bourns Inc.	3310Y-001-103L-ND	1	$3.82
PIC® 18F Microcontroller IC 8-Bit 48MHz 32KB (16K x 16) FLASH 28-SOIC	Microchip	PIC18F2550	1	$0 (Electronic Services Shop)
Half Bridge Driver	STMicroelectronics	L298n	1	$11.20
Push Button	N/A	N/A	3	$0 (Electronic Services Shop)
Toggle Switch	N/A	N/A	1	$0 (Electronic Services Shop)
Misc. PCB Components	N/A	N/A	X	$0 (Electronic Services Shop)
SUPERLELE Glass Graduated Cylinder Set 10ml 25ml 50ml 100ml, Thick Glass Beaker Set	SUPERLELE	B07PDVSL9N	1	$19.99

Sodium Hydroxide Solution, 0.1M, 1L	Innovating Science	N/A	1	$19.99
pH Calibration Buffer Solution Kit, 4.00 and 7.00	Apera Instruments	N/A	1	$25.00
Laboratory Retort Support Stand for Titration Extraction	XMWangzi	N/A	1	$21.99

4.2 Labor

The average starting salary for a UIUC Electrical Engineer Graduate is $42.20/hour. With an overhead factor of 2.5 and having worked an estimated 120 hours on this project throughout the semester, the cost comes to $12,660 per group member. Since there are three group members, the cost of our labor comes to $37,980. For the machine shop, the average salary of a machine shop worker is around $20/hour. With our design taking an estimated 14 hours to complete, the total cost of the machine shop would be $280. The total costs of the parts from Table 2 come to $207.86. This results in the complete cost of our project being $38,467.86.

5. Conclusion

5.1 Accomplishments

The final demonstration of the automatic titration machine showed full control of the motor subsystem drawing in liquid and dispensing it out. This indicated that the control, power, and motor subsystems were working as intended. When using a voltage source to replace the broken pH subsystem, the user interface correctly showed a changing pH value from 0 to 14 when supplied with a varying voltage from 0 V to 5 V. This indicates that our final design would have met the high-level functionality if we had a working pH sensing subsystem. The design achieved the second high-level requirement of completing a titration within 5 minutes because we were able to dispense the entire syringe's volume within 5 minutes.

5.2 Uncertainties

The biggest uncertainty with this project is the precision we set out to achieve with the first and third high-level requirements. The pH electrode and circuitry components all had tolerances to support meeting our third high-level requirement of reading changes in voltage of 0.020 V, we just did not have a working subsystem to verify it. For the first requirement, there was variance in repeat titrations coming from the syringe chosen and syringe driver provided by the machine shop. The syringe had slight variances in force required to push the plunger, and the linear motion of the syringe driver was inconsistent throughout the full range of motion. Trying to further reduce these inconsistencies would help us narrow the product towards the intended precision goals.

There also is uncertainty in whether the pH electrode we purchased was faulty, or if it was the circuitry that did not work as intended. A pH electrode only has a shelf life of around 3 years [4], and we purchased an older model of electrode to save money. There is a chance that this electrode was not stored properly and has become defective. This would match the verification tests we tried with the electrode because we were not able to read a voltage output from it regardless of the setup.

5.3 Ethical considerations

There are many ethics and safety concerns when dealing with our project of titration, especially with the use of varying chemicals. We acknowledge that the IEEE ethics code [5] I.1 to "hold paramount the safety, health, and welfare of the public" will be chief in ensuring that our product will provide safe manipulation of chemicals in multiple applications and environments. This will be kept in mind for all design aspects of the project.

We also believe that the ACM's code of ethics 1.3 "Be honest and trustworthy" [6] is one that plays into all factors of our project from the design, build and functionality. It should provide a reminder to always cite sources, check patents and be honest about the origins of ideas. Additionally, when showcasing our project, it is important to be honest about the limitations and features. We must also be trustworthy between teammates to ensure we keep a good working relationship and continue working efficiently and effectively.

Safety is of utmost importance when considering the design and execution of this product. We stand behind the procedures set by the Division of Research Safety. This includes standards for chemical

waste management making sure that the chemical compounds used in titration are not directly poured down the drain unless authorized which includes rinses that do not include heavy metals or acutely toxic waste with others on a case-by-case basis [7]. Additionally, all other waste should be placed in a proper container and use DRS waste tags for identification.

We also must keep in mind the possibilities of spills in the lab area, and we need to understand the risks and the ability to clean up any issues. The 445 lab already has a bright yellow spill kit that we intend to have present during the titration demo. The Division of Research Safety [8] outlines chemically resistant gloves, googles, acid & base neutralizers, and chemical spill absorption powder as essential when working with chemicals. You must use the PPE to protect yourself from the effects of these chemicals and properly use the absorption powder and understanding the classifications of chemicals to do so properly. For the demo we have planned, the chemicals used will be acetic acid (vinegar) and sodium hydroxide. The vinegar used will be food grade and can be used without PPE, but we will still have a safety data sheet on hand for acetic acid in case of emergencies. Sodium hydroxide will be used at 0.1 M concentration, which is dilute but still can cause damage to the user so nitrile gloves, googles and a lab coat will be worn for the demo. The volume of NaOH used will be 20 mL or less because we will use a 20 mL syringe, and the volume of acetic acid that can be neutralized with 20 mL of NaOH is around 2.4 mL, so 3 mL or less of acetic acid will be needed for the demo. For chemical storage, the base will need to be in its own container explicitly labeling the presence of the base, and the acid can be stored in the red box of flammable materials already in the lab. The safety data sheets for both chemicals will be added to the yellow safety folder located at the entrance of the lab. Lastly, because of the nature of the demo, any observers for the demo will need to be seated or standing at least 6 feet away from the demo to prevent any splashes or spills from reaching people not wearing PPE.

Additionally, the risk of fire is possible with the testing of electronics and in real world applications when flammable chemicals are in use. It is important to understand the difference in fire extinguishers that ABC extinguishers can be used for electrical fires and many chemical fires. Although some chemical fires from combustible metals would require a class D extinguisher. Making note of all emergency exits and fire extinguishers near the demo location will be done before demoing.

The use of moving pieces includes a motor and moving syringe provide a need to have rules set in place to keep hands, cords, long clothes, and hair out of reach of these moving components and chemicals/liquids. Additionally, you want to avoid these types of objects as to not pull the lab setup over which would result in spilled chemicals and possibility for fire.

5.4 Future work

The pH subsystem was the biggest challenge with this project, and further work is needed to debug why the op-amp chips kept breaking and the electrode was not working. To help prevent op-amp damage, a buffer should be added between the ADC of the microcontroller and the output of the amplifying op-amp circuitry. A buffer could also be added between the reference voltage circuitry and the pH electrode to ensure that there is no current leaking into the electrode and causing interference.

References

[1] “AN-1852 Designing With pH Electrodes”, Texas Instruments, https://www.ti.com/lit/an/snoa529a/snoa529a.pdf?ts=1714496882841&ref_url=https%253A%252F%252Fwww.google.com%252F

[2] L298_H_Bridge.PDF, https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

[3] M. S. Read, “Pressure effects of syringes,” *Anaesthesia*, vol. 48, no. 11, pp. 1017–1017, Nov. 1993. doi:10.1111/j.1365-2044.1993.tb07504.x

[4] “pH Sensor Shelf Life and Expiration”, Hamilton, https://www.hamiltoncompany.com/process-analytics/ph-and-orp-knowledge/ph-shelf-life-and-expiration-questions

[5] “IEEE code of Ethics,” IEEE, https://www.ieee.org/about/corporate/governance/p7-8.html

[6] “The code affirms an obligation of computing professions to use their skills for the benefit of society.,” Code of Ethics, https://www.acm.org/code-of-ethics

[7] “Division of Research Safety,” Illinois, https://drs.illinois.edu/Page/Waste/ChemicalWasteProcedures

[8] “Division of Research Safety,” Illinois https://drs.illinois.edu/Page/IncidentResponse/EmergencyPreparedness

[9] “How to calibrate a pH meter?,” Royal Brinkman International - Global Specialist in Horticulture, https://royalbrinkman.com/knowledge-center/mechanical-equipment/ph-meter-calibration

[10] Sourav Gupta, “Interfacing Stepper Motor with PIC Microcontroller”, Circuit Digest, https://circuitdigest.com/microcontroller-projects/interfacing-stepper-motor-with-pic16f877a.

Appendix A Requirement and Verification Table

Table 3: Sensing Subsystem Requirements and Verification

Requirements	Verification	Results
• The pH sensor will utilize a 5 +/- 0.1 V input	• Place multimeter probes on the Vdd and GND pins on the pH sensor PCB to determine the voltage	• The linear regulator outputs 5.04 volts to the pH circuitry
• The pH sensor will output voltages between 0 and 5 +/- 0.1V	• Place multimeter probes on the output to ADC and GND pins on the pH sensor circuitry to determine the voltage • If the current or voltage is too large, the microcontroller will not be able to reliably read the data. Adjust the output current by altering the resistance at the output.	• When testing the pH circuitry with an external voltage, it properly amplified the 0-0.8 volts to 0-5 volts. • At an input of 0.8 volts, the multimeter was reading 4.9 volts
• Measure pH with the pH probe within +/- 0.056 (+/- 0.02V of the correlated voltage; 0 V +/- 0.02V for 0 pH, and 5 V +/- 0.02V for 14 pH)	• Obtain demineralized water, pH4 and pH7 solutions • Place the electrode in the pH7 solution • After the measurement becomes stable (~ one minute), adjust the pH sensor to output 2.5 volts • Rinse the electrode well with demineralized water • Repeat the previous three steps with pH4 solution (achieve a reading of 1.429V), until the pH sensors output voltage is within +/- 0.02V for both solutions [9]	• Obtained the solutions • Cleaned the electrode with distilled water • Placed the electrode in the pH7 solution • Measurement never became stable. Constantly read out 17mV when changing solutions
• Reduce the amount of electrical interference around the instrument	• Remove any unnecessary machinery in the immediate area to reduce electrical noise	• All nearby electronics and equipment were moved out of the way

Table 4: Power Subsystem Requirements and Verification

Requirements	Verification	Results
• System Protection	• When deciding what wall adapter to use, make sure it can protect the system against power surges, inserting the plug in backwards, and other safety features	• The selected wall adapter, 12V 4A Signcomplex wall adapter protects our system
• AC-DC wall adapter that produces a DC voltage between 7-12 volts to power the Arduino	• Use a multimeter to ensure the wall adapter is producing a voltage between the Arduinos specified values	• The wall adapter provided a voltage 12.26 volts
• Route the wall adaptors DC voltage through an emergency shutoff button	• Connect the wall adapter to the power button in the user interface subsystem • Ensure the button is operating correctly by using a multimeter to ensure no voltage is being supplied to the system when the switch is open	• A multimeter was connected to each wire soldered onto the switch to check continuity • After connecting the switch to the PCB, it was known when power is supplied of the circuit by the LEDs on the Arduino
• Subsystem produces a 5 +/- 0.1 V output to power the pH sensor and microcontroller	• Validate the 5 V regulator can accept the voltage produced by the wall adapter. • Ensure the regulator is outputting a voltage at five volts and within tolerance	• The linear regulator outputs 5.04 volts to the pH circuitry

Table 5: Control Subsystem Requirements and Verifications

Requirements	Verification	Results
• The microcontroller will operate the motor to avoid pushing the plunger too far into the barrel, or by pulling the plunger out of the barrel.	• Attach a closed syringe to the stepper motor's rod • Set the timer/counter to a relatively small value • Record the distance and time traveled while the timer/counter is active • Check how far the plunger moved inside of the barrel • Divide the total distance or time by the specified value and multiply it against the current timer/counter • Reset the plunger to the initial closed state and click start to check your new timer/counter • Make minor adjustments to get the desired results	• Placed syringe and motor driver in the closed state • Set counter to 500 and recorded the time and distance moved • Determined that the plunger moved just about 2mL • Multiplied 500 by 10 to get the final timer of 5000 • This value worked for our needs
• The microcontroller will create a four wired signal to control the motor dependent on step size. For a design specific time interval, the syringe will need to produce one drop of titrate.	• Adjust the wire outputs based on the driving mode of the motor [10] until one drop comes out per cycle • Achieve at least 30 drops for 30 duty cycles to confirm that no more or less drops dispended out of the syringe	• The motor was operating properly with the four control signals • It was not possible to get 30 drops for 30 sets of steps. The syringe driver didn't have constant torque across the rod, resulting in a different number of drops and drop sizes
• The Arduino will be used manipulate the pH data and to display to the LCD screen.	• Data will be retrieved and connected to the LCD screen in the User Interface • Check that the pH by using the pH solutions that were used to calibrate the electrode	• The LCD screen was properly displaying the pH for the correct analog value being read by the microcontroller
• The microcontroller will communicate to the LCD screen through I2C	• Setup I2C using the LiquidCrystal_I2C library	• The LCD screen was setup to display a bootup message

	• Display the pH value by dividing the 10bit signal by 1024 and then multiplying by 14	• After UART was connected, the signal was able to be manipulated into an accurate pH value
• The microcontroller will communicate with the Arduino through UART	• Set the baud rate to 9600 • Connect the Transmit pin of the microcontroller to the receive pin of the Arduino, along with ground • Test out UART by sending character messages that can get displayed on the LCD screen • Send the data as two characters since UART is 8-bits	• After setting up the baud rate and the rest of the microcontroller control signals, the UART was properly sending bootup messages • The 10-bit signal was recombined after being received by the Arduino and displayed to the screen

Table 6: Motor Subsystem Requirements and Verification

Requirements	Verification	Results
• The motor driver will amplify the microcontrollers signals, into ones that the motor can operate at by boosting current and voltage	• Observe the motors movement of the motor when the start or reset button is pressed in the user interface subsystem	• The driver circuitry amplified the voltage and current to run the motor driver
• The driver circuit must not overheat	• The L298n half bridge driver is used to amplify the voltage and current for the motor • Since the L298n is BJT logic, it cannot run the motor at slow speeds without heating up. • Run the stepper motor in a piecewise fashion, short bursts with waiting time, to reduce heating problems • A heat sink can also be applied to the chip if necessary	• Running a continuous stepping signal resulted in major heating problems. • Running the motor in a piecewise fashion, ~10% of the time, the heating was greatly reduced to virtually nothing
• The motor must produce at least 33 Newtons of force [3] to pull and push the plunger out of the barrel	• When deciding what motor to use, guarantee it can produce enough force to operate the syringe • Since the torque is being converted from angular to linear, a part of the force will be used in this conversion, resulting in the necessity for a motor that can produce more than 33 newtons of force	• The motor was tested to see if it had the necessary torque needed to operate the syringe. This was done by attaching a shoestring to the plunger of the syringe and turning on the stepper motor. The torque was more than enough

Table 7: User Interface Subsystem Requirements and Verification

Requirements	Verification	Results
• The DC power from the wall adapter is fed through an emergency/power button	• Connect the wall adapter to the power button • Ensure the button is operating correctly by using a multimeter to ensure no voltage is being supplied to the system when the switch is open	• A multimeter was connected to each wire soldered onto the switch to check continuity • After connecting the switch to the PCB, it was known when power is supplied of the circuit by the LEDs on the Arduino
• The start button initiates the motor to start pressing the plunger into the syringe	• When the start button is pressed, the syringe will start to close • The syringe will stop moving when it has reached its approximate endpoint that was determined in the microcontroller's code	• After clicking the start button, clicking it again, or the reset button, won't do anything • The start button moves the syringe to the proper position
• The reset button initiates the motor to start pulling the plunger out of the syringe	• When the reset button is pressed, the syringe will pull open • The syringe will stop moving when it has reached its approximate endpoint that was determined in the microcontroller's code	• After clicking the reset button, clicking it again, or the start button, won't do anything • The start button moves the syringe to the proper position
• The stop button will result in the start or reset state being instantly stalled	• When the stop button is pressed, the syringe driver will stop moving if it was • If the system was in the start state and the stop button was pressed, two things can happen. If start is pressed again, it will continue the current titration. If reset were pressed, then the syringe will return to the reset state with the plunger fully pulled out • The opposite is true for the reset state	• The stop button was tested to make sure the syringe will never be pulled out of the barrel or pushed into it too far • Start, reset, and stop were interchangeably pressed to check the functionality of stop