Electric Paintbrush Cleaner

By

Luis Bibian

John Kriston

Yael Legaria

Final Report for ECE 445, Senior Design, Spring 2020

TA: Madison Hedlund

8 May 2020

Project No. 65

Abstract

Painters often struggle with keeping their paintbrushes clean throughout the duration of the painting process, as the usual method of rinsing a paintbrush in a cup of water decreases in effectiveness as the water becomes more contaminated with residual paint over time. In our first solution, we aimed to fix this problem by designing a device that could clean a paintbrush by directing clean water from a storage tank down onto the paintbrush via a spray valve, with the dirty water collected in a removable tank sitting directly below the paintbrush. Our first design also implemented a rotating cleaning brush system for the purpose of cleaning the paintbrush bristles. For our second solution, we focused on designing a system that would allow users to manually rinse their paintbrushes in a refillable paintbrush cleaning receptacle, while also filtering acrylic paint particles out of the used water. Compared to our original solution, our second solution allows for a more efficient use of clean water, and also takes a more environmentally-friendly approach with the handling of the contaminated water.

Contents

1 Second Project Motivation

1.1 Updated Problem Statement

When painting, it is commonplace for painters to have a cup or jar of water used to clean their paintbrush when switching between colors. However, after only a handful of times of dipping the paintbrush in the water to clean it, the water becomes heavily stained with residual paint. As the unclean water starts seeping into the brush, this leads to the unwanted mixing of colors on the canvas. This is especially problematic when working with acrylic paints due to their thick consistency compared to other types of paint such as watercolor [1]. In addition to quickly contaminating the painter's rinse water, acrylic paints also require the proper disposal of acrylic paint wastewater due to the plastic particles present in the paint. Although having a designated cup of water to clean the paintbrush is a simple way to combat unwanted color mixing, the overall problem is that it is highly inconvenient for painters to continuously get up to empty and refill their cup, as this ultimately disrupts the painting process.

To mitigate the problem of contaminated rinse water interfering with the quality of a painting, painters will usually either use two separate containers of water to clean their brush (one for getting most of the paint off the brush, and the other for rinsing the brush after the first clean), or a very large tupperware/bucket of water such that any paint that is rinsed into the water will achieve a higher degree of dilution with the larger volume of water. However, neither of these solutions fix the problem of the cleaning water itself getting heavily contaminated with different colors. Additionally, the regular jars/mugs that painters use to clean their paintbrushes are prone to spilling since they are easily knocked over.

Furthermore, there is a need for a product that can ease the painting process for painters working with acrylic paint. Acrylic paints are popular amongst beginner, intermediate, and advanced painters alike for their longevity once put on a canvas and their 'forgiveness of mistakes' [1]. But because of their thicker consistency when compared to watercolor paints, acrylic paints contaminate rinse water at a faster rate, and the plastic particles present in acrylic paint require proper disposal. Acrylic paint particles that contaminate rinse water are harmful to septic systems due to their potential to clog sewage pipes [2], but many painters neglect the treatment of acrylic wastewater because it is a seemingly tedious process. For these reasons, the need for a product that makes the filtration process for painters hassle-free cannot be overstated.

1.2 Updated Solution

Our proposed solution is to create a device that allows the user to easily get a fresh supply of clean water while properly handling the disposal of the contaminated acrylic wastewater. To achieve this, our solution consists of three major parts for handling the flow of water. The first part of our solution is a clean water storage tank that sits at the bottom of the device, holding up to 2.5 liters of water. A dc water pump will sit in the clean water storage tank and direct clean water to a paintbrush cleaning receptacle, which is the second major part of our solution and is positioned at the top of the device. The paintbrush cleaning receptacle is the designated area where users can clean their paintbrushes and receive refills of fresh cleaning water in 0.25 liter amounts. When the user requests a new refill of water, the used water in the paintbrush cleaning receptacle is emptied into the third major component of our solution: a filtration tank that sits between the clean water storage tank and the paintbrush cleaning receptacle. The filtration tank is responsible for filtering used water from the paintbrush cleaning receptacle by separating the acrylic paint particles from the water through a chemical process known as flocculation, where small particulates present in water are treated to form larger clumps that can easily be removed by passing the water through a coffee filter. To treat the filtered water, aluminum sulfate and hydrated lime powder are required to trigger the flocculation process, as outlined by Golden Artist Colors Inc. [3]. The filtration tank also comes equipped with a pH sensor that detects the pH of the wastewater to determine the proper amount of chemicals needed to treat the water, as well as mixing blades that are spun by a motor to evenly distribute the chemicals throughout the wastewater. After the acrylic paint particles are separated from the used water, the water is directed back into the clean water storage tank through a funnel with coffee filters that are used to catch the flocculated paint solids. These coffee filters can later be removed and replaced by the user. To control the flow of water between the three compartments, solenoid valves are positioned between the paintbrush cleaning receptacle and the filtration tank, as well as between the filtration tank and the funnel leading to the clean water storage tank.

The painter controls the device through a user interface that sits beside the clean water storage tank. The user interface has push buttons that allow the user to request a new refill of clean water, as well as to start the filtration process at the end of a painting session. The user interface also includes an LCD display that provides the user with information on the current pH of the wastewater, how many refills of clean water the user has left, and the quantities of each chemical that the user needs to add for filtering. Additionally, a piezo buzzer is built into the user interface to alert the user once the filtration process is complete. Figure 7 (located in Appendix A) illustrates the overall physical design of our solution. Our solution directly addresses the problem statement by providing users with a way to continuously rinse their paintbrushes in clean water without disrupting the flow of the painting process, while also providing a way for users to properly dispose of acrylic paint waste. Due to its physical size, our solution also remedies the lack of stability that comes with using regular cups/jars for cleaning, meaning that the cleaning water is less prone to spilling from being knocked over.

For our original solution, we proposed a device that would clean paintbrushes by spraying clean water down onto the paintbrush, allowing dirty water to be collected in a removable water collection receptacle sitting below the paintbrush. Upon the insertion of the paintbrush into the device, a passive infrared sensor would detect the motion of the paintbrush and trigger a dc water pump, directing clean water from a storage tank on top of the device down to a spray valve positioned over the paintbrush. After rinsing the paintbrush, a rotating brush system actuated forward by a stepper motor would clean the bristles of the paintbrush to remove any dried paint particles.

Our new solution differs from our previous solution in three major aspects. The first major change for our new design is that we have removed the motorized brush system that was proposed in our initial design entirely. Realistically, we were not sure how effective a rotating cleaning brush would be at cleaning a paintbrush given the flexibility and density of paintbrush bristles. Additionally, the wait time experienced by the user during the deployment of the brush system would require more time for cleaning than if the user were to just swish their paintbrush around in a regular cup of water, making the cleaning process less convenient than it should be.

The second major change for our new design is that paintbrushes will now be inserted vertically into a cleaning receptacle for cleaning instead of horizontally into a cleaning compartment where a spray valve directs water onto the paintbrushes. While our previous solution was not equipped to handle brushes of various sizes due to the constraints of directing water from a spray valve of a fixed size, our new solution is designed to work with any paintbrush that can have its bristles dipped into the circumference of a regular mug. Replacing the spray valve with a cleaning receptacle also allows for a more efficient use of cleaning water compared to the previous design.

Finally, the most significant change for our new solution is that its use is now focused around the implementation of a filtration system. Because our previous design had no way of filtering or recycling the contaminated rinse water, it was intended for use among painters working with watercolor paints. However, since the new design filters the rinse water, allowing water to be recycled and acrylic particles to be disposed of, this product is targeted towards painters working with acrylic paints, as this market has an intrinsically higher need/demand compared to watercolor paints due to reasons previously mentioned.

Our new design excels over our previous design in its ability to make efficient use of the reserved cleaning water and the ease with which users can clean their paintbrushes. In our original solution, every paintbrush would receive the same amount of water from the spray valve regardless of its size, meaning that paintbrushes with finer tips would use up more water than necessary, and paintbrushes with larger tips would potentially require multiple rinses. Because our new solution implements a clean water receptacle that gives the user full control over when to receive a new supply of clean water, this allows the user to replace the water as frequently as they desire. This design aspect of the new solution is especially important considering that when working with multiple colors, it makes sense for painters to only use a fresh supply of clean water when switching between colors, as this prevents the unwanted mixing of colors on the canvas. Additionally, our design's filtration systems fulfills an educational role in teaching users about the methods required to treat contaminated water.

The closest product that currently exists on the market for brush cleaning is the STYLPRO makeup brush cleaner [4], which cleans makeup brushes with a motorized handheld device that spins the makeup brush around in a small bowl of water to get any residual makeup out of it. However, this product does not address our initial problem of the water easily getting stained during the painting process since the STYLPRO is only intended for a single use before the user has to manually refill the bowl. Additionally, the STYLPRO is not designed to handle brushes that are stained with acrylic paint, as brushes that are stained with makeup do not require the filtration of any residual particles that exist in the rinse water.

1.3 Updated High-Level Requirements

- The user can request at least eight refills of water to the paintbrush cleaning receptacle during each painting session for a water storage tank filled with 2.5 liters of water.
- The treated water's pH at the end of the filtration cycle must sit between a pH of 6.5 and 8.5 to be considered acceptable for recycling.
- The filtration system must demonstrate the ability to remove acrylic paint particles from the treated wastewater, indicated by the transition of the wastewater from opaque to transparent.

1.4 Updated Visual Aid

Figure 1 illustrates the two main functions of the proposed solution. Through a user interface module, the user can request a refill of clean water, causing the wastewater to drain into the filtration tank to be replaced by clean water routed to the paintbrush cleaning receptacle. The user can also request to filter the wastewater, causing the acrylic paint particles to flocculate as they separate from the water and begin to form solid clumps easy for filtering.

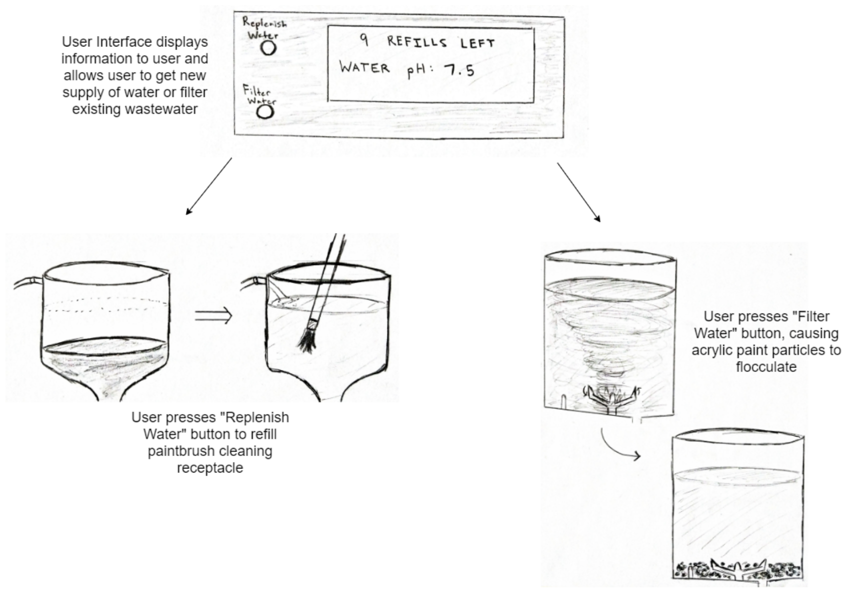

User Interface displays information to user and allows user to get new supply of water or filter existing wastewater
Replenish Water
Filter Water
9 REFILLS LEFT
WATER pH: 7.5
User presses "Replenish Water" button to refill paintbrush cleaning receptacle
User presses "Filter Water" button, causing acrylic paint particles to flocculate

Figure 1: Visual Aid for Electric Paintbrush Cleaner

1.5 Updated Block Diagram

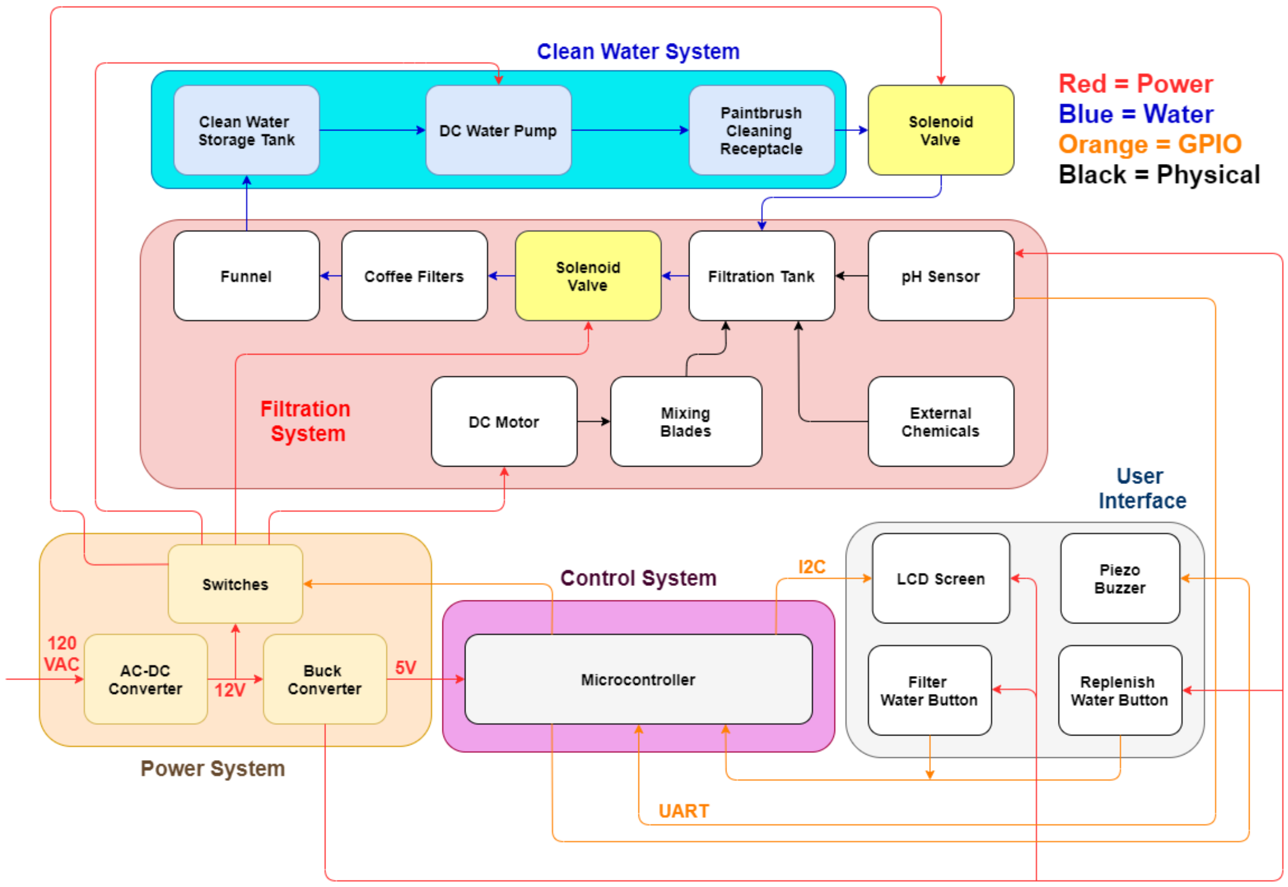


Figure 2: Block Diagram for Electric Paintbrush Cleaner

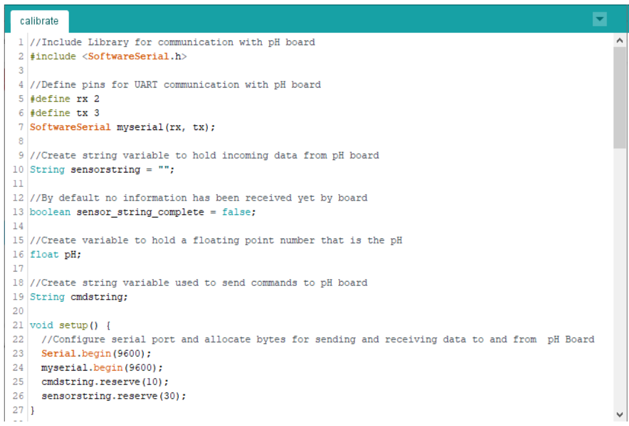
Figure 2 depicts the functional block diagram of the device. On startup, the user presses the ‘replenish water’ button to fill the cleaning receptacle with fresh water from the clean water storage tank, with the clean water directed up to the cleaning receptacle via a dc water pump and tubing system. When the user wants to get a new supply of water, pressing the ‘replenish water’ button again empties the dirty water from the receptacle into the filtration tank through a solenoid valve before refilling the clean water receptacle with new water. As contaminated water pours into the filtration tank, the microcontroller keeps track of the wastewater’s pH using the pH sensor, while also keeping track of the amount of water in the filtration tank based on how many times the user has requested a refill of clean water.

From the pH of the wastewater along with its volume, the microcontroller computes the amount of aluminum sulfate solution the user should add to the contaminated water and prints this value to the LCD display. To filter the wastewater, the user adds the amount of aluminum sulfate solution specified by the LCD display to the filtration tank and then presses the ‘filter water’ button, causing the dc motor to turn the mixing blades to begin the flocculation process. Once enough time has passed for the aluminum sulfate to take effect, the piezo buzzer will beep two times to signal the user to add the correct amount of hydrated lime powder as specified by the LCD display. After adding the hydrated lime powder, the user presses the ‘filter water’ button once again to resume the treatment of the wastewater, causing the dc motor and mixing blades to mix the solution to ensure even distribution of the hydrated lime powder. When sufficient time has passed for the acrylic paint particles to fully separate from the water, the microcontroller switches on a solenoid valve to release the treated water from the filtration tank, allowing the recycled water to fill into the clean water storage tank via a funnel as the flocculated paint particles are collected in the coffee filters. The piezo buzzer will beep three times when the filtration is complete.

2 Second Project Implementation

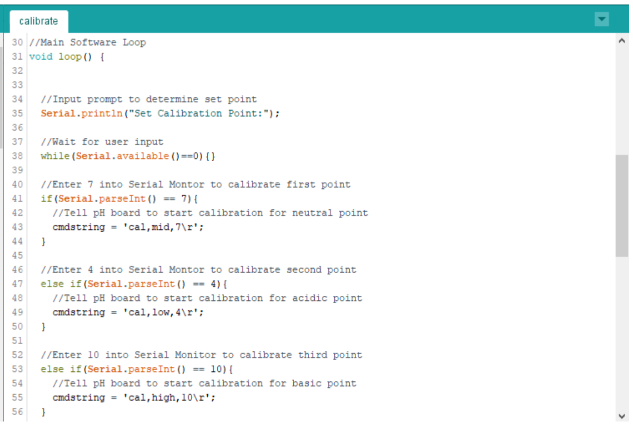
2.1 Software for pH Sensor Calibration

Before integrating the Atlas Scientific pH sensor with our project, we must first calibrate the sensor such that it will output accurate pH readings. According to the EZO-pH datasheet [5], three-point calibration of the pH sensor board is necessary to provide the most accurate reading possible, and this involves submerging the sensor in a neutral solution, an acidic solution, and then a basic solution. Because aluminum sulfate lowers the pH of the wastewater below the neutral point, whereas hydrated lime powder raises the pH of the wastewater above the neutral point, three-point calibration is favorable over two-point or one-point calibration of the sensor since we want the pH sensor to cover the full range of the pH scale.



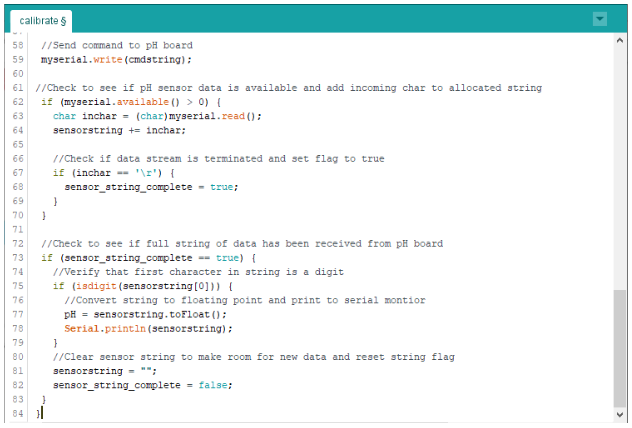
```
//Include Library for communication with pH board
#include <SoftwareSerial.h>

//Define pins for UART communication with pH board
#define rx 2
#define tx 3
SoftwareSerial myserial(rx, tx);

//Create string variable to hold incoming data from pH board
String sensorstring = "";

//By default no information has been received yet by board
boolean sensor_string_complete = false;

//Create variable to hold a floating point number that is the pH
float pH;

//Create string variable used to send commands to pH board
String cmdstring;

void setup() {
  //Configure serial port and allocate bytes for sending and receiving data to and from  pH Board
  Serial.begin(9600);
  myserial.begin(9600);
  cmdstring.reserve(10);
  sensorstring.reserve(30);
}
```

(a) First Part of Code



```
//Main Software Loop
void loop() {



  //Input prompt to determine set point
  Serial.println("Set Calibration Point:");

  //Wait for user input
  while(Serial.available()==0){}

  //Enter 7 into Serial Montor to calibrate first point
  if(Serial.parseInt() == 7){
    //Tell pH board to start calibration for neutral point
    cmdstring = 'cal,mid,7\r';
  }

  //Enter 4 into Serial Montor to calibrate second point
  else if(Serial.parseInt() == 4){
    //Tell pH board to start calibration for acidic point
    cmdstring = 'cal,low,4\r';
  }

  //Enter 10 into Serial Monitor to calibrate third point
  else if(Serial.parseInt() == 10){
    //Tell pH board to start calibration for basic point
    cmdstring = 'cal,high,10\r';
  }
```

(b) Second Part of Code



```
  //Send command to pH board
  myserial.write(cmdstring);

//Check to see if pH sensor data is available and add incoming char to allocated string
  if (myserial.available() > 0) {
    char inchar = (char)myserial.read();
    sensorstring += inchar;

    //Check if data stream is terminated and set flag to true
    if (inchar == '\r') {
      sensor_string_complete = true;
    }
  }

  //Check to see if full string of data has been received from pH board
  if (sensor_string_complete == true) {
    //Verify that first character in string is a digit
    if (isdigit(sensorstring[0])) {
      //Convert string to floating point and print to serial montior
      pH = sensorstring.toFloat();
      Serial.println(sensorstring);
    }
    //Clear sensor string to make room for new data and reset string flag
    sensorstring = "";
    sensor_string_complete = false;
  }
}
```

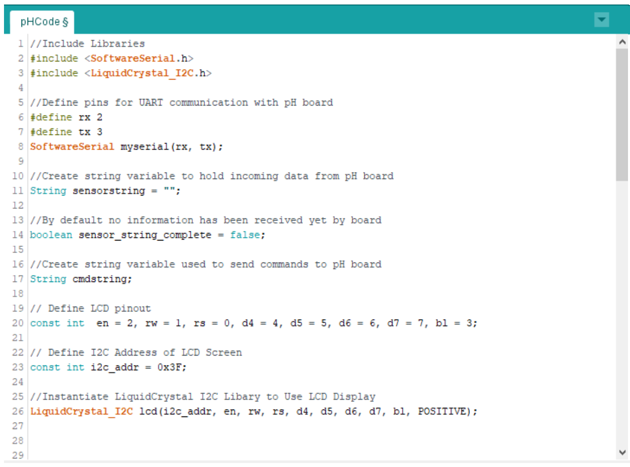
(c) Third Part of Code

Figure 3: Code for Calibrating pH Sensor

Figure 3 outlines the code required to calibrate the pH board. In Figure 3a, UART communication is set up to allow the microcontroller to send commands to the pH board, as well as receive incoming data. In Figure 3b, we set up a prompt that will wait for us to enter the pH of the solution that we wish to test. In Figure 3c, the calibration command with the known pH of the solution is sent to the pH board, and the microcontroller reads from the incoming data stream until a carriage return is detected, indicating that the pH board has sent a complete reading, at which point the pH reading is printed to the serial monitor.

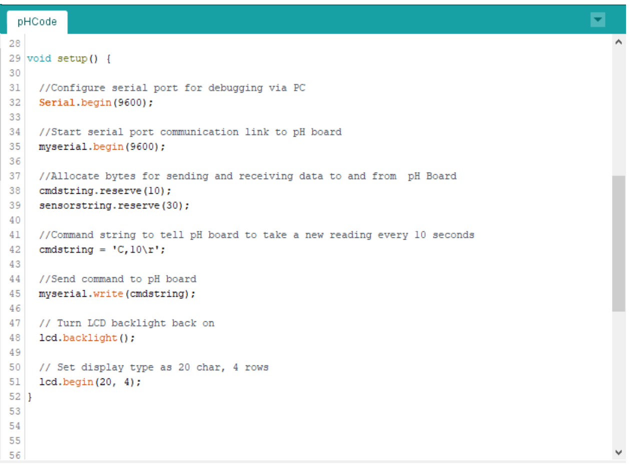
2.2 Software for Displaying pH Sensor Data to LCD Display

After calibrating the pH sensor, we will want to output the pH readings to the LCD display so the user can monitor the pH of the wastewater. Figure 4 outlines the code needed to print the pH sensor data to the LCD display. In Figure 4a, the microcontroller is configured for UART communication with the pH sensor and I2C communication with the LCD display. In Figure 4b, we open the serial connection and configure the pH board to transmit a new reading from the pH sensor every ten seconds. In Figure 4c, the microcontroller polls the serial connection for incoming data from the pH sensor and prints the measurement to the LCD display. Before printing the new reading, the LCD display is cleared and the position of the cursor is reset.



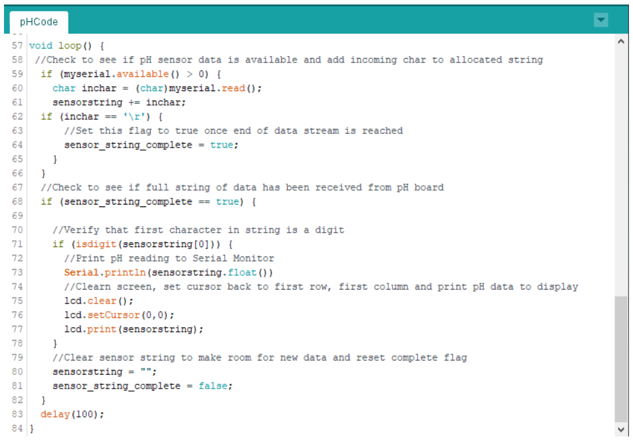
```
//Include Libraries
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>

//Define pins for UART communication with pH board
#define rx 2
#define tx 3
SoftwareSerial myserial(rx, tx);

//Create string variable to hold incoming data from pH board
String sensorstring = "";

//By default no information has been received yet by board
boolean sensor_string_complete = false;

//Create string variable used to send commands to pH board
String cmdstring;

// Define LCD pinout
const int  en = 2, rw = 1, rs = 0, d4 = 4, d5 = 5, d6 = 6, d7 = 7, bl = 3;

// Define I2C Address of LCD Screen
const int i2c_addr = 0x3F;

//Instantiate LiquidCrystal I2C Libary to Use LCD Display
LiquidCrystal_I2C lcd(i2c_addr, en, rw, rs, d4, d5, d6, d7, bl, POSITIVE);
```

(a) First Part of Code



```

void setup() {

  //Configure serial port for debugging via PC
  Serial.begin(9600);

  //Start serial port communication link to pH board
  myserial.begin(9600);

  //Allocate bytes for sending and receiving data to and from  pH Board
  cmdstring.reserve(10);
  sensorstring.reserve(30);

  //Command string to tell pH board to take a new reading every 10 seconds
  cmdstring = 'C,10\r';

  //Send command to pH board
  myserial.write(cmdstring);

  // Turn LCD backlight back on
  lcd.backlight();

  // Set display type as 20 char, 4 rows
  lcd.begin(20, 4);
}
```

(b) Second Part of Code



```
void loop() {
 //Check to see if pH sensor data is available and add incoming char to allocated string
  if (myserial.available() > 0) {
    char inchar = (char)myserial.read();
    sensorstring += inchar;
  if (inchar == '\r') {
      //Set this flag to true once end of data stream is reached
      sensor_string_complete = true;
    }
  }
  //Check to see if full string of data has been received from pH board
  if (sensor_string_complete == true) {

    //Verify that first character in string is a digit
    if (isdigit(sensorstring[0])) {
      //Print pH reading to Serial Monitor
      Serial.println(sensorstring.float())
      //Clearn screen, set cursor back to first row, first column and print pH data to display
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print(sensorstring);
    }
    //Clear sensor string to make room for new data and reset complete flag
    sensorstring = "";
    sensor_string_complete = false;
  }
  delay(100);
}
```

(c) Third Part of Code

Figure 4: Code for Displaying pH Sensor Reading to LCD Display

2.3 Tank/Receptacle Sizing

For our project, it is important for the design to have a physical structure that remains stable while sitting on a table/desk without compromising the ease of use by taking up too much space. In order to satisfy these constraints, the water storage tank and filtration tank need to be sized accordingly.

Because the water storage tank sits at the bottom of the device and serves as the base, the water storage tank should naturally have a larger diameter than the filtration tank. Additionally, the tank should have a smaller height compared to the filtration tank to prevent the height of the overall design from being too large. In order to provide enough room for the 2.5 liter storage capacity without spilling any water, the clean water storage tank will have an extra one liter of clearance space to account for the movement of water during transport. To limit the amount of vertical space that the clean water storage tank takes up, it is designed to satisfy a height of three inches and a total volume of 3.5 liters. Based on Equation (1), a 3.5 liter volume corresponds to 213.6 cubic inches of volume to work with for sizing the tank.

$$3.5\ L \cdot \frac{1000\ cm^3}{1\ L} \cdot \frac{0.0610237\ in^3}{1\ cm^3} = 213.6\ in^3 \tag{1}$$

Using a three-inch height constraint, the clean water storage tank should have a radius of approximately 4.77 inches (or a diameter of 9.54 inches) by Equation (2).

$$r = \sqrt{\frac{V}{\pi h}} = \sqrt{\frac{213.6\ in^3}{\pi(3\ in)}} = 4.77\ in. \tag{2}$$

Working with these dimensions for the clean water storage tank, when filled to maximum capacity there is a total of 0.866 inches of free space above the water level to account for any sudden movements, as shown by Equations (3), (4), and (5).

$$2.5\ L \cdot \frac{1000\ cm^3}{1\ L} \cdot \frac{0.0610237\ in^3}{1\ cm^3} = 152.56\ in^3 \tag{3}$$

$$h_{full} = \frac{152.56\ in^3}{\pi(4.77\ in)^2} = 2.134\ in \tag{4}$$

$$h_{free} = h_{tank} - h_{full} = 3\ in - 2.134\ in = 0.866\ in \tag{5}$$

For sizing the filtration tank, two liters of clearance will be provided in addition to the 2.5 liter storage capacity to account for any splashing of the water during the mixing process. Because the filtration tank sits directly on top of the clean water storage tank, a constraint of 3.5 inches for the radius is imposed to ensure that the diameter of the filtration tank doesn't completely cover the width of the clean water tank, as we want to leave enough space to run tubing through the top of the clean water tank to the paintbrush cleaning receptacle. Using Equations (6) and (7), the filtration tank is sized to a height of 7.14 inches.

$$4.5\ L \cdot \frac{1000\ cm^3}{1\ L} \cdot \frac{0.0610237\ in^3}{1\ cm^3} = 274.63\ in^3 \tag{6}$$

$$h = \frac{V}{\pi r^2} = \frac{274.63\ in^3}{\pi(3.5\ in)^2} = 7.14\ in \tag{7}$$

The size of the paintbrush cleaning receptacle is modeled after the size of a typical mug that a painter would use to clean their brushes. From our observations, a radius of 1.625 inches (or a diameter of 3.5 inches) and a height of 4 inches is suitable for the cleaning receptacle. To prevent users from spilling the 0.25 liters of water during their normal rinsing motions, the cleaning receptacle should have an extra 0.15 liters of clearance above the water level at minimum. Based on Equation (8), the cleaning receptacle requires a minimum volume of 24.41 cubic inches to accommodate for the 0.25 liters of water plus 0.15 liters of clearance space. Equation (9) shows that our selected radius of 1.625 inches and height of 4.0 inches meet the minimum sizing required for the paintbrush cleaning receptacle.

$$0.4\ L \cdot \frac{1000\ cm^3}{1\ L} \cdot \frac{0.0610237\ in^3}{1\ cm^3} = 24.41\ in^3 \tag{8}$$

$$V = \pi r^2 h = \pi(1.625\ in)^2(4\ in) = 33.183\ in^3 \tag{9}$$

2.4 Simulation of Solenoid Valve Switching

Because the solenoid valves used to control the flow of water have inductance, it is important to provide a path for current to flow when the solenoids transition from on to off to deal with the large amounts of back-emf generated across the solenoids. When the MOSFETs used to switch the solenoids on and off undergo a switch-state transition, unless a snubber diode is attached across the solenoids to provide a path for the current to dissipate, arcing can occur across the switches due to the instantaneous voltage buildup across the solenoid when the current conducting path is removed.

The Nexperia PSMN022-30PL MOSFET that we have selected for our project has a maximum rated drain-to-source voltage of 30 V, with a rise time of 29 ns and fall time of 7 ns [6]. Figure 11 (located in Appendix C) illustrates the solenoid switching circuit simulation created in LTspice accounting for the switch transition times. Because the solenoid draws 550 mA of current, it is represented in the LTspice simulation by an ideal inductor in series with a 21.81 Ω resistor. From Figure 11c, it is clear that a solenoid valve with a 5 μH inductance is enough to damage the MOSFETs. Additionally, switching a solenoid without a snubber diode introduces significant ringing to the voltage waveform seen across the MOSFET.

Figure 12 (located in Appendix C) illustrates the solenoid switching circuit simulation created in LTspice with the addition of the snubber diode. Once the snubber diode is added, the maximum voltage seen across the MOSFET during a switch-state transition becomes the supply voltage, as shown in Figure 12c. In Figure 12d, the inductor current slowly dissipates through the path formed by the snubber diode and the solenoid's resistance, instead of rapidly dropping off to zero as in Figure 11d.

2.5 Software for Switching MOSFETs and Handling User Input

In order to implement the software for driving the MOSFETs, it is important to first determine how long each electromechanical component needs to be switched on for. For the dc water pump, the amount of time it remains switched on is determined by its flow rate. Given that the flow rate of our selected water pump is 0.1527 liters per second when operated at 12 V [7], it should remain on for 1.637 seconds in order to move 0.25 liters from the clean water storage tank to the cleaning receptacle, according to Equation (10). The dc motor that spins the filtration tank's mixing blades should remain on for two minutes, or 120 seconds, to thoroughly distribute the chemicals throughout the wastewater.

$$\frac{1\ sec}{0.1527\ L} \cdot 0.25\ L = 1.637\ sec \tag{10}$$

Determining the amount of time each of the solenoid valves should remain open requires the use of Equation (11), which describes how the height of a liquid sitting in a container changes over time when drained through a given area. In Equation (11), A_p refers to the area of the opening that the water is falling through, C_v is the flow coefficient associated with the solenoid valve, g is the acceleration due to gravity, $h(0)$ is the initial height of the water, and A_t is the area of the tank that is holding the water. The derivation of Equation (11) is covered in Appendix D.

$$h(t) = \left(\frac{-A_p C_v \sqrt{2g}}{2A_t} t + h(0)\right)^2 \tag{11}$$

Our selected solenoid valves have a British Standard Pipe (BSP) specification of G1/4" [8], which corresponds to a thread diameter of 11.445 millimeters. The solenoid valves also have an associated flow coefficient of 0.23 [8]. Using the given thread diameter, Equation (12) is used to calculate the area of the solenoid valve's opening.

$$A_p = \pi(5.7225 \cdot 10^{-3}\ m)^2 = 1.0288 \cdot 10^{-4}\ m^2 \tag{12}$$

The area of the paintbrush cleaning receptacle is given by Equation (13). When filled with 0.25 liters of water, the initial height of the liquid in the paintbrush cleaning receptacle is given by Equation (14). Plugging in the known values of A_p, C_v, g, A_t, and $h(0)$ into Equation (11) results in Equation (15). By setting Equation (15) equal to zero and solving for t, it takes 7.269 seconds for 0.25 liters of water to drain from the paintbrush cleaning receptacle into the filtration tank.

$$A_t = \pi(1.625in)^2 \cdot \frac{0.000645m^2}{1\ in^2} = 5.3521 \cdot 10^{-3}\ m^2 \tag{13}$$

$$h(0) = \frac{24.41\ in^3}{\pi(1.625\ in^2)} \cdot \frac{0.0254\ m}{1\ in} = 0.0747\ m \tag{14}$$

$$h(t) = \left(\frac{-(1.0288 \cdot 10^{-4}\ m^2)(0.23)\sqrt{2(9.81\ m/s)}}{2(5.3521 \cdot 10^{-3}\ m^2)} t + 0.0747\ m \right) \tag{15}$$

The same process is used to find out how long the solenoid valve connecting the filtration tank to the clean water storage tank should remain open. The area of the filtration tank is given by Equation (16), and the initial height for 2.5 liters of wastewater is given by Equation (17). Plugging in the known values into Equation (11) results in Equation (18). By setting Equation (18) equal to zero and solving for t, it takes 47.705 seconds for 2.5 liters of water to drain from the filtration tank into the clean water storage tank.

$$A_t = \pi(3.5\ in)^2 \cdot \frac{0.000645m^2}{1\ in^2} = 0.02483\ m^2 \tag{16}$$

$$h(0) = \frac{152.56\ in^3}{\pi(3.5\ in^2)} \cdot \frac{0.0254\ m}{1\ in} = 0.1006856\ m \tag{17}$$

$$h(t) = \left(\frac{-(1.0288 \cdot 10^{-4}\ m^2)(0.23)\sqrt{2(9.81\ m/s)}}{2(0.02483\ m^2)} t + 0.1006856\ m \right) \tag{18}$$

Figures 13, 14, 15, 16, 17, and 18 (located in Appendix E) display the software implemented to drive the switches and handle user input from the push buttons. In Figure 13, constants are defined corresponding to the time in milliseconds that the software needs to activate each switch, along with the wait times between each activation. To be safe, an extra second has been added to the calculated time constants that were determined for the solenoid valves. In Figure 14, the pin mappings corresponding to the schematics shown in Appendix B are defined. In Figures 15, 16, 17, and 18, the main software loop handles the user input and executes the water-refill and filtration processes accordingly.

For switching on the dc motor and the solenoid connecting the filtration tank to the clean water storage tank, the *millis()* function is used over the *delay()* function since these pins will be on for a longer period of time. While using *delay()* completely halts the execution of the program, using *millis()* allows the program to continue running while keeping track of time in the background.

3 Second Project Conclusions

3.1 Implementation Summary

The first thing we accomplished in Chapter 2 was writing the software needed to use the pH sensor. Using the software from Section 2.1, the pH sensor board can be calibrated to measure across a pH range of 0.0 to 14.0. Ensuring that the pH sensor is properly calibrated is essential to our project since two of our high-level requirements rely directly on the feedback from the pH sensor. In an actual lab environment, we would use this software in conjunction with three solutions with known pH levels of 4.0, 7.0, and 10.0, leaving the pH sensor submerged in each solution for two minutes as recommended by the manufacturer to achieve proper calibration [5]. We also managed to write the software used for printing the pH data directly to the LCD display, covered in Section 2.2. This allows users to monitor the pH of the wastewater on the user interface throughout their painting sessions and during the filtration process. Furthermore, this software can be easily repurposed to output instructions to the user detailing how much aluminum sulfate and hydrated lime powder to add to the filtration tank based on the pH readings.

In Section 2.3, we determined the dimensions of the three main water containers for our project, ensuring that there was ample clearance space for the maximum water levels while still taking the stability of the overall design into consideration. Knowing the dimensions of the water containers ahead of time is important since this information is also used to determine how long the solenoid valves need to remain open. The calculations completed in Section 2.3 also serve as a basis for any 3D modeling of the physical construction that we would perform when working with the machine shop on this project.

In Section 2.4, we performed simulations in LTspice to verify the working implementation of our solenoid valve switching circuits. Specifically, we observed how operating the solenoids without a snubber diode attached in parallel would damage our selected MOSFETs by exceeding the rated drain-to-source voltage. Verifying that the protection diode mitigated this problem is important since the solenoids are responsible for controlling the flow of the used water throughout our system, and any damage sustained by the MOSFETs would compromise our project's ability to allow the user to replenish or filter the water.

Finally, in Section 2.5 we accomplished building the software required to process user input and drive the electromechanical components. This piece of software is important for controlling the time that each MOSFET is switched on in order to ensure that the water replenishment and filtration processes occur in a cohesive cycle. Realistically, this piece of software would be integrated with the code written in Section 2.2 to fully implement the pseudocode illustrated in Figure 10 (located in Appendix B).

3.2 Unknowns, Uncertainties, and Testing Needed

Unfortunately, due to the current state of the nation, we are left with only simulations and designs which could not be put to test, thus leaving us with many uncertainties. Among these uncertainties lies many points of verification for the functionality of our project. To begin with, our ac/dc power converter was meant to be tested using an oscilloscope in the lab to assure that our output voltage would remain within 5% of 12 V while supplying a maximum current of 3.5 A. In addition, similar measurements would need to be conducted for our buck converter to ensure that it could output 5 V at $\pm$ 5% while supplying a maximum of 300 mA of current.

Another important aspect of our project that requires lab access for testing is our control unit, which includes our microcontroller, MOSFETs, and all of the other I/O components associated with the microcontroller. To start, we would need to verify that our microcontroller could supply the MOSFETs with 5 V at $\pm$ 10% for at least two minutes using a multimeter to ensure that stable amounts of power could be delivered to our components. These MOSFETs would also need to have their drain-source voltages verified to be less than 0.2 V while conducting to minimize losses. We would also need to verify that our digital inputs would register as logical LOW signals at voltages less than 1 V, and logical HIGH signals at voltages greater than 4 V. This could be done in the lab using a bench power supply to send various voltage levels to the microcontroller while connecting LEDs to the digital output pins of the microcontroller that would turn on to verify a registered HIGH signal and off to verify a registered LOW signal. Furthermore, we would also test the microcontroller to ensure that it could print text to our LCD display. This would be achieved by connecting the microcontroller to a PC via a USB-to-TTL adapter and starting a PuTTY session with the serial speed set to 9600 baud. We would then use an Arduino sketch to test the read/write capabilities of the LCD display by processing user input from the PC and ensuring that the appropriate text appears visible on the LCD display via the I2C protocol.

Once starting the overall product assembly, we would need to test our electromechanical components. Specifically, we would have to verify that the DC water pump we ordered can actually deliver 0.25 liters of water within five seconds of receiving its power from the MOSFET. This would be tested in the lab simply by submerging the pump in a water tank containing more than 0.25 liters of water, then powering the pump with 12 V from the lab bench supply and using a stopwatch to time how long it takes to fill the cleaning receptacle to its 0.25 liter mark. We would have to perform similar testing on our solenoid valves to ensure that they could open and close within 1.5 seconds of receiving power from their associated MOSFETs. This would also be tested using power from the lab bench and a stopwatch to time how long it takes for the solenoid valves to open and close.

Finally, after assembling the filtration system, we would need to verify its operation in the lab as well. Specifically, we would check that the DC motor used for turning the mixing blades could maintain a speed of 100 RPM with the filtration tank at full capacity. To test this, we would fill the filtration tank with 2.5 liters of water and attach the motor to the mixing blades. We would then apply a 12 V dc signal from the lab bench to the motor terminals and record the speed of the motor using a stroboscope. Along with this, we would also test our pH sensor's ability to supply our microcontroller with accurate information. This would be accomplished by creating an aluminum sulfate solution of a known pH in the lab through dilution, and then measuring the pH of the solution using the pH sensor and printing that data to the LCD display. Additionally, we would need to test our coffee filters to make sure that they can withstand the filtration process. This would simply be done by performing the filtration process several times and analyzing whether or not our coffee filters were tearing under the pressure of the water and acrylic particles. To bring everything together, we would need to verify that our microcontroller could properly execute the timing cycles set for our water-refill and filtration processes while continuously updating the LCD display with data from the pH sensor.

Overall, apart from testing to make sure that all of our components are working, we are aware that there would most likely be some troubleshooting problems that would arise even after the initial testing phases, and most of the troubleshooting would need to be done using the equipment in the ECE 445 lab. These issues could include things such as software errors, mechanical failures, faulty parts, power restrictions, and any other problems that require the use of the lab benches to probe our components and get appropriate measurements to identify where the problem is occurring. We understand that it is unreasonable to believe that our current plans and designs are completely flawless.

3.3 Ethics and Safety

The first safety issue we must address is one referenced by Section 7.8.1 in the IEEE code of ethics. We must face the possibility of water coming into contact with our electrical subsystems if proper precautions are not taken [9]. In order to prevent this, we have sealed each subsystem off from one another, so no liquid can pass or leak uncontrollably. This way, users are protected from electrical shock, and the electrical components are also protected.

Furthermore, in Section 7.87 of the IEEE code of ethics, it states "we must seek, accept, and offer honest criticism of technical work" [9]. With a product like this, it is imperative to get user feedback so we can actually help artists make painting a more enjoyable experience. For a product like ours that can be marketed as an educational tool for kids, we think it is important that our product ensures the ease of use for painters of all ages. Thus, our group wants to try out our prototype with parents and their kids first to see how they react to the product.

A safety manual will inform users of the safety precautions that should be taken when working with the new chemical agents for the filtration process as well as first aid measures in case of emergencies. The aluminum sulfate and hydrated lime powder can be sensitive to the skin or eyes, so wearing the proper protective equipment (such as goggles, gloves, and aprons) is necessary. If the aluminum sulfate solution is accidentally exposed to the skin or eyes, one should immediately flush/rinse the exposed area with water for up to 20 minutes [10]. Furthermore, if the solution is accidentally ingested, a person should drink a large quantity of water or milk and should not induce vomiting [10]. If the hydrated lime powder is exposed to the skin or eyes, the same measures should be taken as for the aluminum sulfate. However, unlike with aluminum sulfate, if hydrated lime powder is ingested one should at most only drink small sips of water, and instead focus on flushing out their mouth [11]. We will make it clear in the saftey manual that the filtration process should only be carried out by a responsible adult following the important safety guidelines. These safety precautions are in support of the IEEE guidelines set in place to hold paramount the safety of the consumer [9].

Finally, as stated by the IEEE code of ethics, our group must "be honest and realistic in stating claims or estimates" [9]. This pertains mainly to the amount of brushes that can be cleaned from a water storage tank holding 2.5 liters of clean water. As of now, we estimate that the user will be able to rinse their brush a minimum of ten times (0.25 liters of water per rinse). However, if this changes in the future, our users must be made aware of it since maintaining an adequate water supply is a major concern for the ease of use of our project.

3.4 Project Improvements

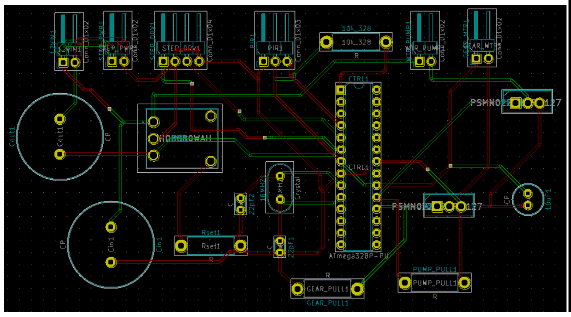
If given a full year to work on this project, the first improvement we would look into is refining the chemical treatment process of the acrylic wastewater by incorporating a temperature sensor into the design. The Atlas Scientific pH sensor that we chose for our project bases its readings on an assumed operating temperature of 25° Celsius, meaning that variations from this temperature introduce slight errors to the pH measurements. The implications that this has on our current design is that inaccurate pH readings could potentially put the final pH of the treated water outside of the acceptable range of 6.5 and 8.5 if the user treats the water with improper chemical quantities due to skewed pH readings. Another risk is that if the computed chemical quantities are too small based on the pH readings, the water may not transition from opaque to transparent, or if they are too large then the user would be unnecessarily wasting chemicals. By adding a temperature sensor to our design, we could process the temperature of the painter's environment and continuously update the operating temperature of the pH sensor using that data. Overall, this would ensure the functionality of our project over a wider range of temperature conditions, resulting in a more efficient use of the chemical treatments.

The second step we would take to improve this project in the future is implementing a fully-automated filtration process. With our current project, the user is responsible for manually adding the aluminum sulfate and hydrated lime powder to treat the wastewater for every filtration cycle. While our project still provides an easier method for disposing of acrylic paint particles than if users were to attempt to filter the acrylic wastewater on their own without the assistance of the pH sensor and mixing blades, it is still slightly inconvenient for the user to measure out the chemical quantities and wait for the aluminum sulfate to take effect before adding the hydrated lime powder. By adding a mechanism that can dispense the correct amounts of chemical treatment based on feedback from the pH sensor and microcontroller, this allows the user to completely walk away from the device after starting the filtration process. This also lessens the hassle of the user having to put on safety equipment to handle the chemicals each time they wish to filter the water, and would eliminate the possibility of user error in measuring out the chemical quantities.

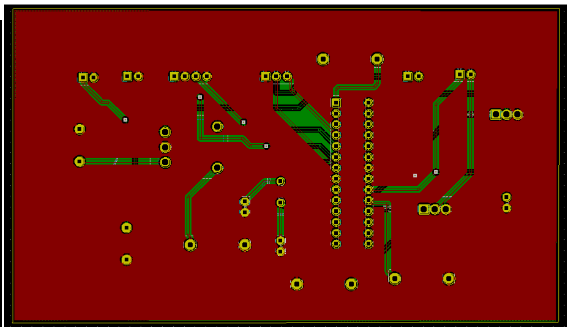
The third improvement we would make for our project is giving users total control over how much clean water they want to route to the paintbrush cleaning receptacle, meaning that clean water would be dispensed in variable amounts rather than in fixed amounts of 0.25 liters. The motivation behind this improvement is that users who are painting with a diverse palette of colors may not require the full 0.25 liters of water to clean off their brush, especially when using some of those colors in smaller amounts. For example, if a painter wants to use a red paint color to sparingly touch up some details in the background of the painting, but then wants to clean their brush before painting the foreground with a green color, the user would be wasting water if they used the full 0.25 liter refill to rinse the red paint off the brush before applying the green paint. Therefore, by allowing users to control how much water gets sent to the paintbrush cleaning receptacle, our project would end up saving more water over the long run.

4 Progress Made on First Project

The progress we made on our first project consists of the printed circuit board design outlined in Figure 5, and the 3D model outlined in Figure 6. The circuit board design contains the necessary footprints to accommodate for the microcontroller, smoothing capacitors, pull-down resistors, the 12 V to 5 V buck converter, the MOSFETs used to drive the dc water pump and cleaning brush motor, and the 16 MHz crystal for the ATMEGA328p. Figure 6 illustrates the physical design of our original project, with the removable water disposal compartment pictured in Figure 6a and the outer casing that it slides into pictured in Figure 6b. We intended to share these designs with the machine shop as a starting point to build a working prototype for our device. From Figure 6, it is clear that the design of our first solution is vastly different from that of our second solution in terms of both shape and functionality due to the lack of a filtration system in the first design iteration. To provide context for the size of our original design, the outer casing pictured in Figure 6b was designed with a length of 11.325 inches, a width of 6.375 inches, and a height of 8.0 inches. Meanwhile, the removable water disposal tank pictured in Figure 6a was designed with a length of 11.125 inches, a width of 4.125 inches, a height of 2.625 inches, and square protrusions of 0.375 inches running across the length.

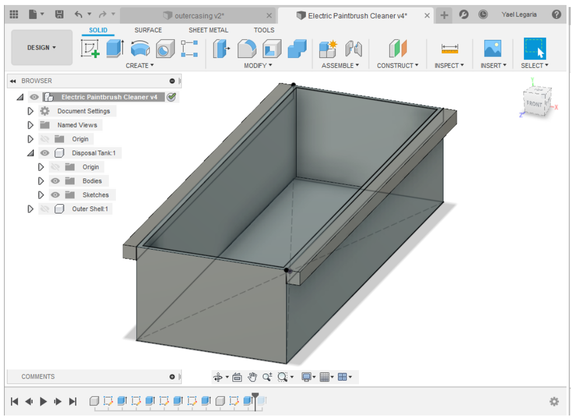


(a) Layout of PCB Traces

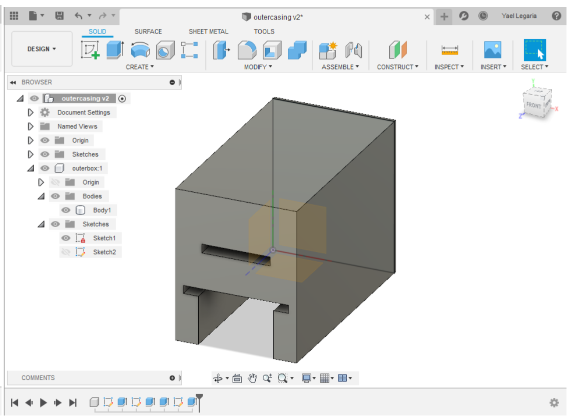


(b) PCB With Added Ground Plane

Figure 5: PCB Design for First Project



(a) Water Disposal Compartment



(b) Outer Casing

Figure 6: Fusion360 CAD Model for First Project

References

[1] "12 Crucial Differences Between Acrylic and Watercolor Paint", *Art Passion*, 2019. [Online]. Available: https://artpassiononline.com/differences-between-acrylic-and-watercolor-paint/ [Accessed: Apr. 2, 2020].

[2] S. Stewart, "3 Steps to an Environmentally Sound Art Studio", *Medium.com*, 2018. [Online]. Available: https://medium.com/@susanlstewart/3-steps-to-an-environmentally-sound-art-studio-d05504d36a22 [Accessed: Apr. 2, 2020]

[3] "Removing Water-Based Paint Solids from Rinse Water", *Golden Artist Colors Inc.*, 2019. [Online]. Available: https://www.justpaint.org/removing-water-based-paint-solids-from-rinse-water [Accessed: Apr. 2, 2020]

[4] "STYLPRO Original", *StylTom*. [Online]. Available: https://styltom.co.uk/products/stylpro [Accessed: Apr. 2, 2020]

[5] Atlas Scientific, "EZO-pH Embedded pH Circuit", 2020. [Online]. Available: https://www.atlas-scientific.com/_files/_datasheets/_circuit/pH_EZO_Datasheet.pdf [Accessed: May 6, 2020]

[6] Nexperia, "PSMN022-30PL", 2010. [Online]. Available: https://www.mouser.com/datasheet/2/916/PSMN022-30PL-1600234.pdf [Accessed: May 6, 2020]

[7] DFRobot, "DC Mini Immersible Water Pump". [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/FIT0563_Web.pdf [Accessed: May 6, 2020]

[8] Amazon.com, "1/4inch DC 12V 2 Way Normally Closed Electric Solenoid Air Valve". [Online]. Available: https://www.amazon.com/4inch-Normally-Closed-Electric-Solenoid/dp/B074Z5SDG3 [Accessed: May 6, 2020]

[9] IEEE.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: Apr. 2, 2020].

[10] General Chemical,"Liquid Alum MSDS", 2008. [Online]. Available: http://www.sawpa.org/wp-content/uploads/2012/05/Aluminum-Sulfate-General-Chemical.pdf [Accessed: Apr. 2, 2020]

[11] Fisher Science Education, "Calcium Hydroxide SDS", 2014. [Online]. Available: https://beta-static.fishersci.com/content/dam/fishersci/en_US/documents/programs/education/regulatory-documents/sds/chemicals/chemicals-c/S25225.pdf [Accessed: Apr. 2, 2020]

Appendix A Physical Design of Electric Paintbrush Cleaner

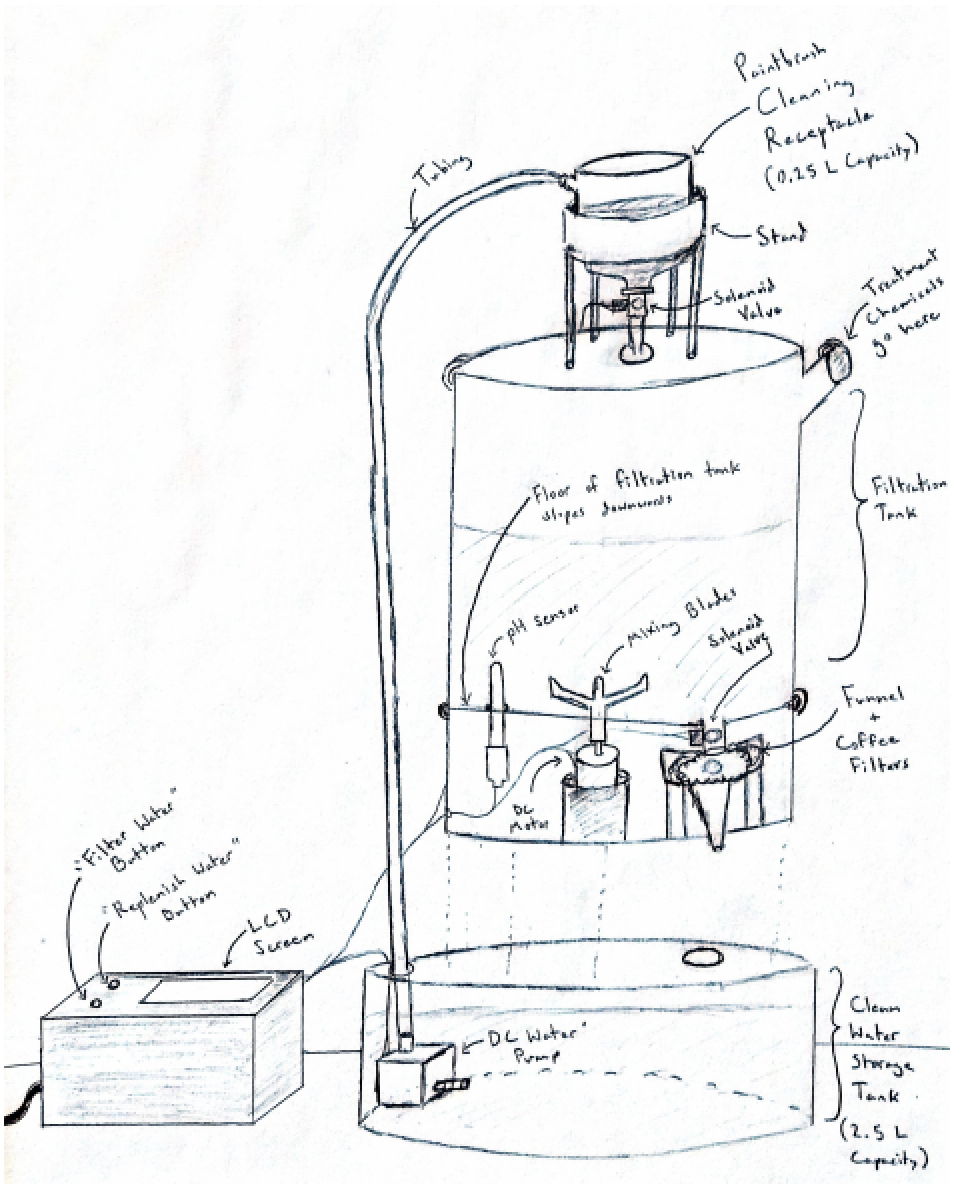


Figure 7: Electric Paintbrush Cleaner Physical Design

Appendix B Schematic, Pseudocode, and Bill of Materials

Figure 8 depicts the schematic of the control system for the electric paintbrush cleaner, consisting of the microcontroller, the LCD screen, the buck converter, the pH sensor board, and the piezo buzzer. Figure 9 depicts the schematic layout for the wiring of the electromechanical components (i.e. dc motor, dc water pump, and two solenoids). The device psuedocode is shown in Figure 10.

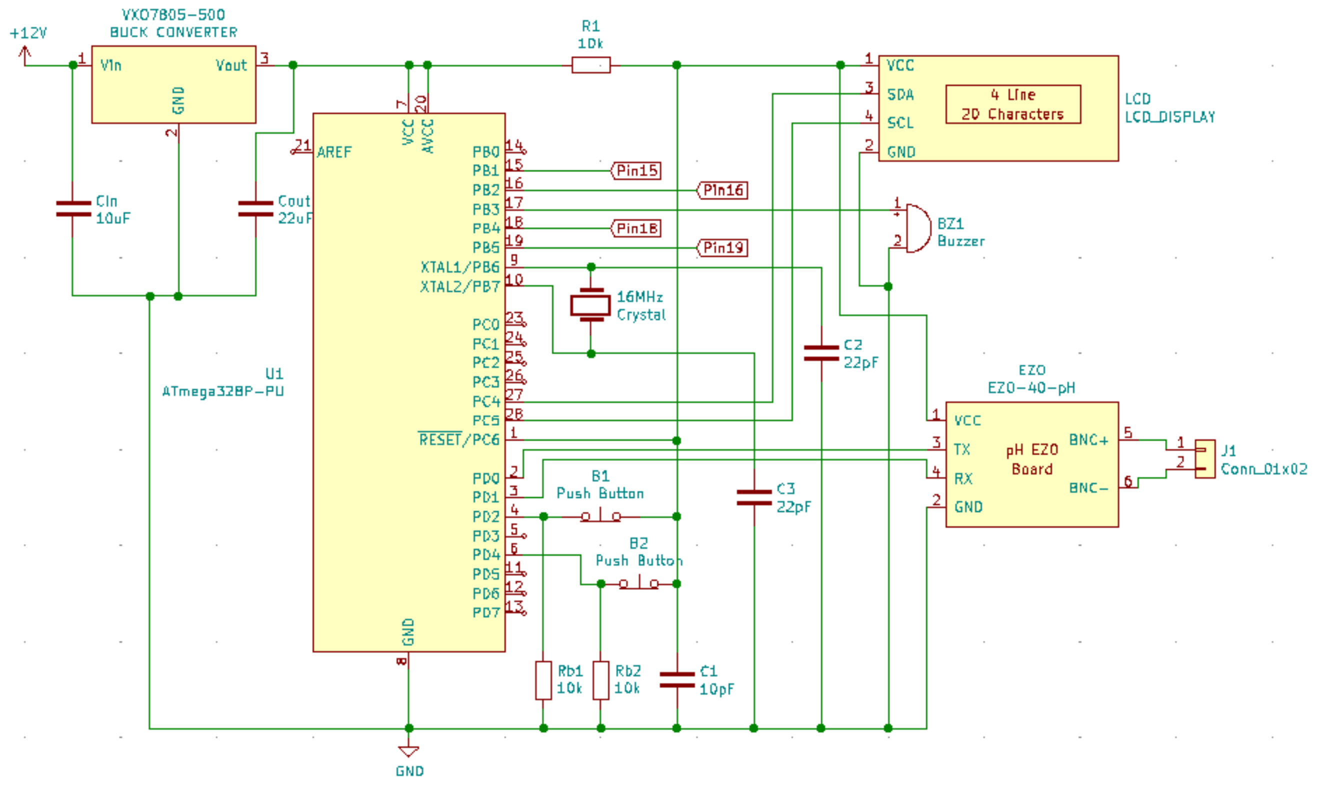


Figure 8: Control System Schematic for Electric Paintbrush Cleaner

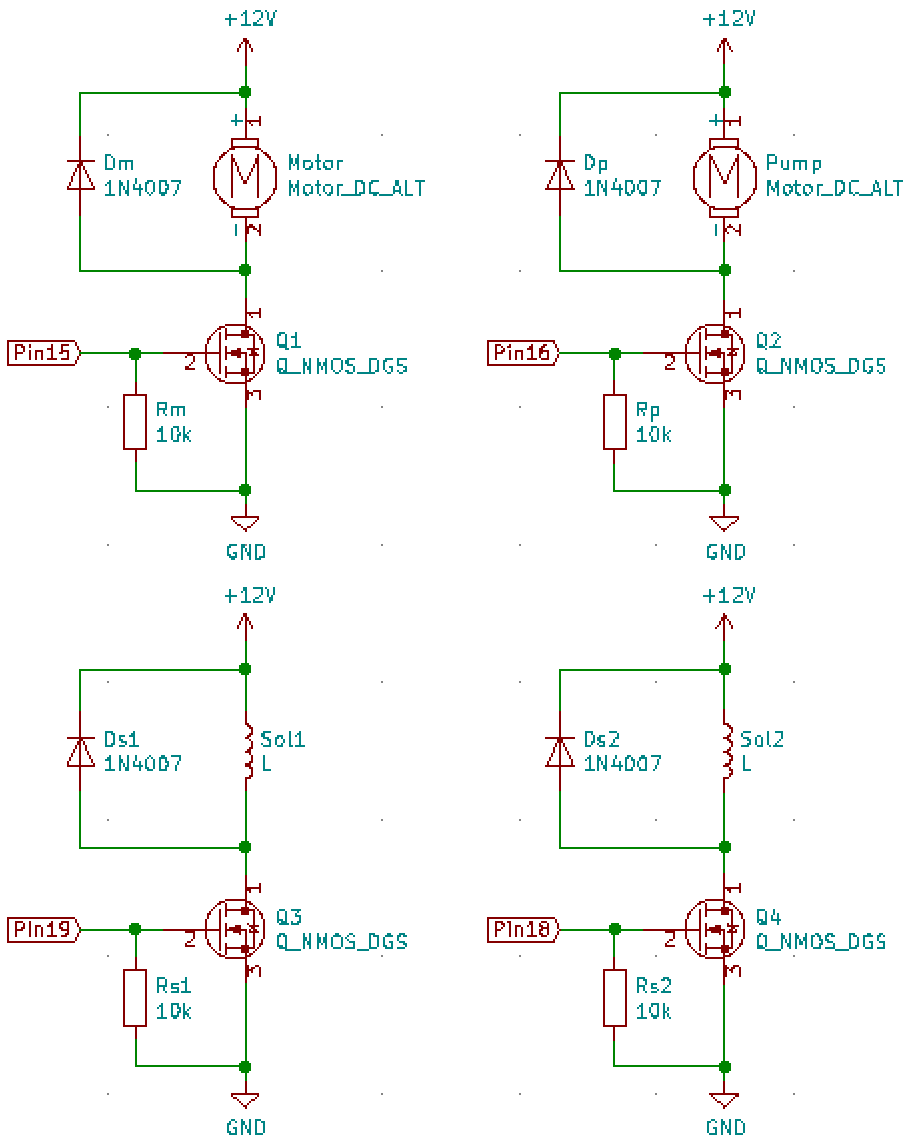


Figure 9: Electromechanical Components Schematic for Electric Paintbrush Cleaner

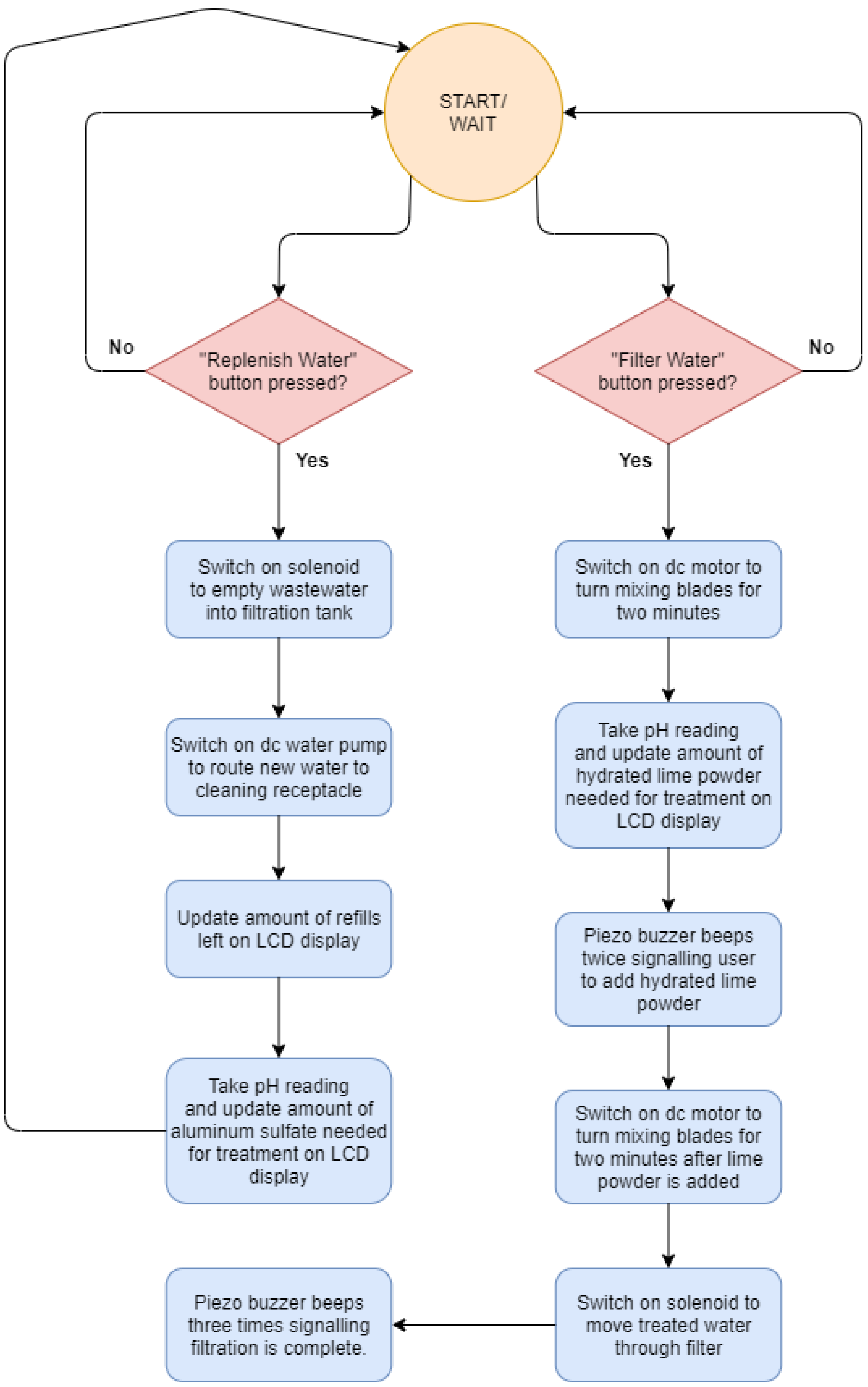

START/ WAIT
"Replenish Water" button pressed?
No
Yes
"Filter Water" button pressed?
No
Yes
Switch on solenoid to empty wastewater into filtration tank
Switch on dc water pump to route new water to cleaning receptacle
Update amount of refills left on LCD display
Take pH reading and update amount of aluminum sulfate needed for treatment on LCD display
Switch on dc motor to turn mixing blades for two minutes
Take pH reading and update amount of hydrated lime powder needed for treatment on LCD display
Piezo buzzer beeps twice signalling user to add hydrated lime powder
Switch on dc motor to turn mixing blades for two minutes after lime powder is added
Switch on solenoid to move treated water through filter
Piezo buzzer beeps three times signalling filtration is complete.

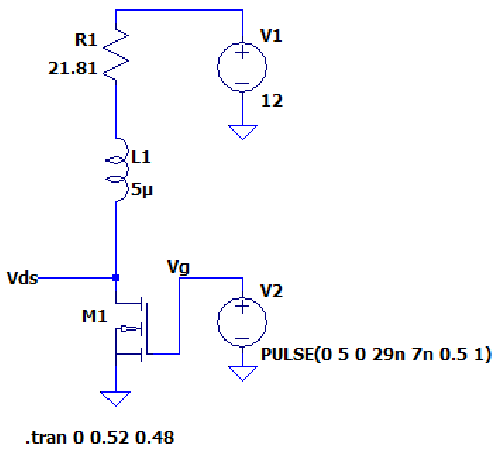
Figure 10: Electric Paintbrush Cleaner Pseudocode

The parts required to complete our project, along with their respective costs are outlined in Table 1. Overall, the estimated cost for parts is \$288.82.

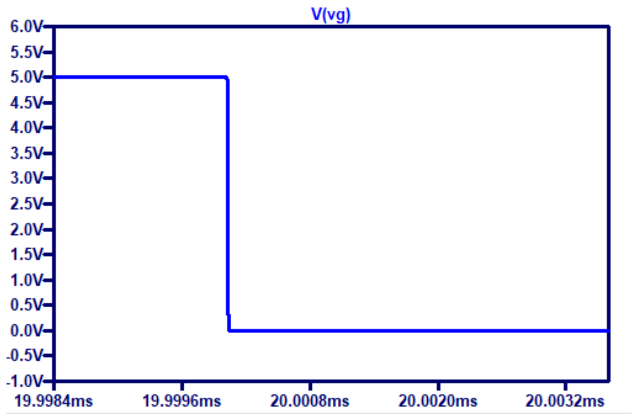
Table 1: Cost Breakdown of Required Parts

Description	**Manufacturer**	**Part Number**	**Quantity**	**Unit Price**
MOSFET	Nexperia	PSMN022-30PL	4	\$0.90
Buck Converter	CUI Inc.	VXO7805-500	1	\$2.36
AC-DC Converter	Signcomplex	ZF120A-1204000	1	\$15.58
pH Sensor	Atlas Scientific	ENV-40-pH	1	\$75.00
pH Sensor Board	Atlas Scientific	EZO-pH	1	\$40.00
Solenoid Valve	Plum Garden	PL-220101	2	\$9.55
DC Water Pump	DFRobot	FIT0563	1	\$9.29
DC Motor	Greartisan	12V 200RPM	1	\$15.99
LCD Screen	SunFounder	I2C LCD2004	1	\$12.99
Microcontroller	Atmel	ATMEGA328P	1	\$2.08
Pack of Pushbuttons	Sparkfun	COM-10302	1	\$5.85
Aluminum Sulfate (2 lbs)	FDC	16828-12-9	1	\$11.99
Hydrated Lime Powder (2 lbs)	Hi-Yield	33362	1	\$14.99
PCB & Manufacturing	Various	Various	N/A	\$60.00
Total				**\$288.82**

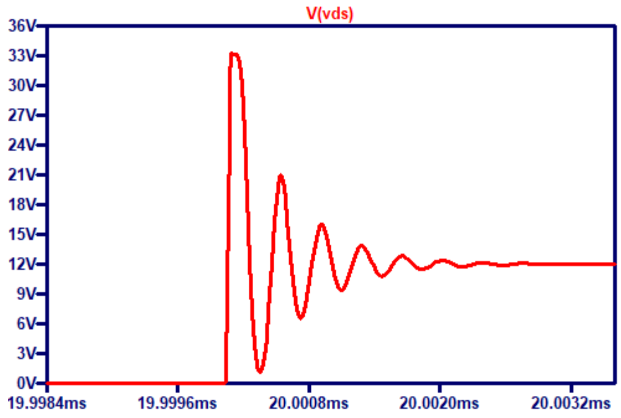
Appendix C LTspice Simulations for Section 2.4



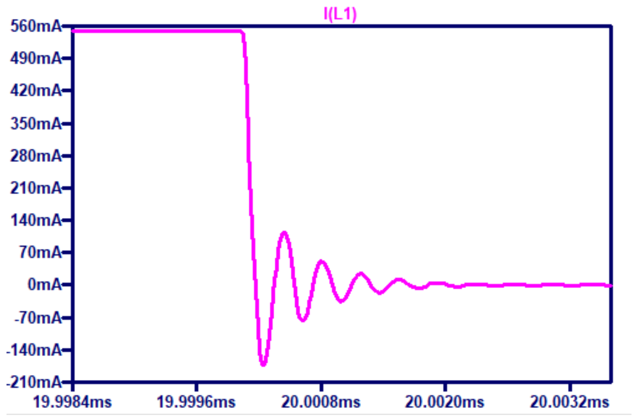
(a) Simulated Circuit



(b) MOSFET Gate Signal

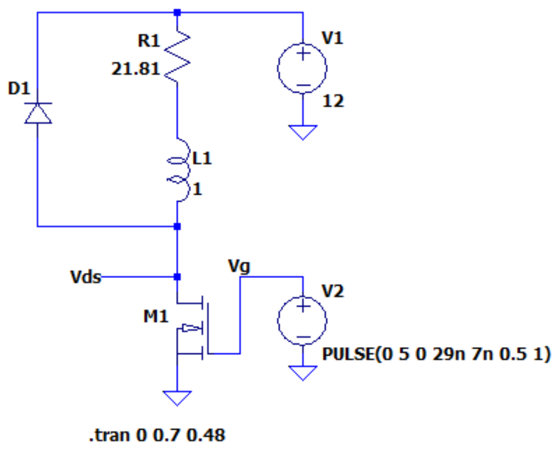


(c) MOSFET Drain-Source Voltage

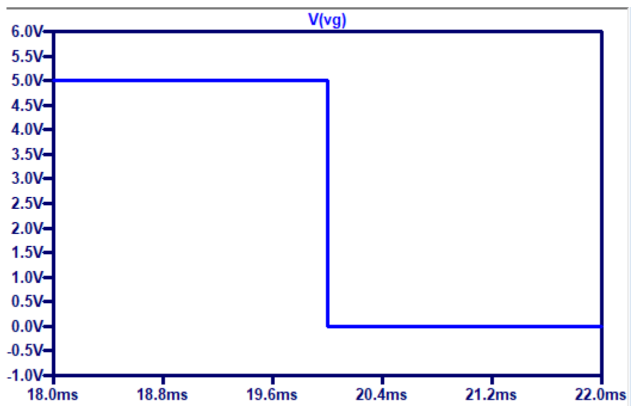


(d) Solenoid Current

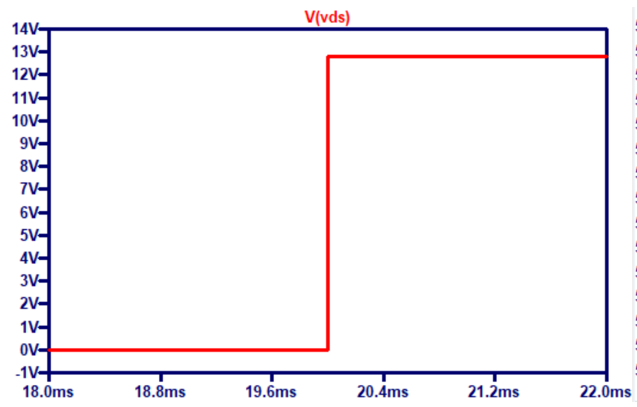
Figure 11: Simulation For Solenoid Switching Circuit Without Freewheeling Diode



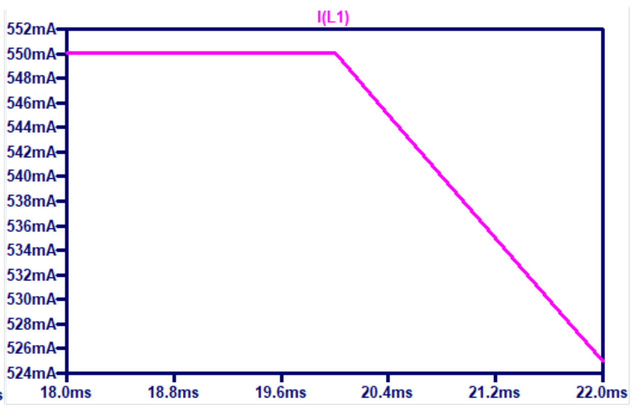
(a) Simulated Circuit



(b) MOSFET Gate Signal



(c) MOSFET Drain-Source Voltage



(d) Solenoid Current

Figure 12: Simulation For Solenoid Switching Circuit With Freewheeling Diode

Appendix D Derivation of Height Equation for Draining Liquid

For a liquid sitting in a tank, Toricelli's law allows us to relate the velocity of the liquid as it is drained from the tank to the change in height of the liquid in the tank. The velocity of a water falling through an area with a flow coefficient of C_v is given by Equation (19), where g is the gravitational acceleration, and the height h is a function of t.

$$v = C_v\sqrt{2gh} \tag{19}$$

Meanwhile, the change in volume of the liquid as it is drained is equal to the velocity of the liquid multiplied by the area of the opening (A_p) and the change in time. This is also equal to the area of the tank (A_t) multiplied by the change in height of the liquid, as shown in Equation (20).

$$\Delta V_t = A_p v \Delta t = -A_t \Delta h \tag{20}$$

Combining Equations (19) and (20) gives Equation (21), which can be rearranged to form the differential equation shown in Equation (22).

$$A_p C_v \sqrt{2gh} \cdot \Delta t = -A_t \Delta h \tag{21}$$

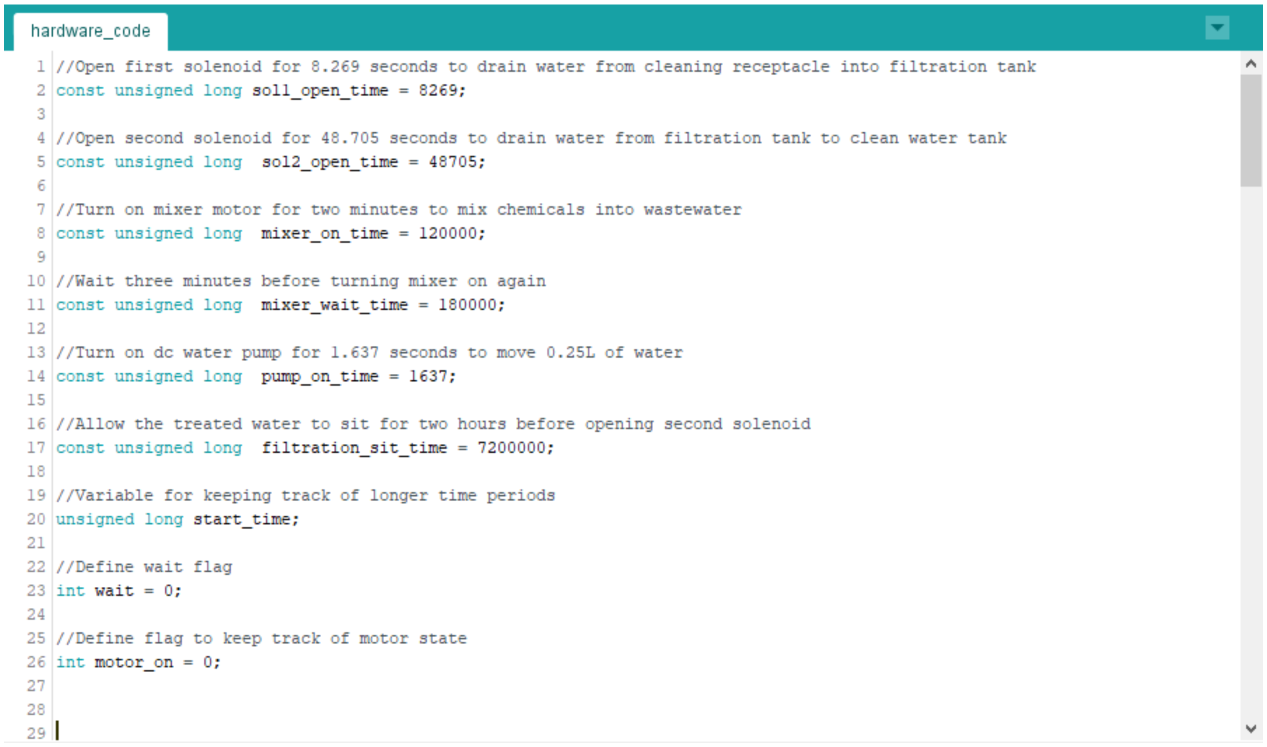
$$\frac{dh}{dt} = -\frac{A_p C_v \sqrt{2g}}{A_t} h^{0.5} \tag{22}$$

Solving Equation (22) by integrating both sides (as shown in Equation (23)), the expression for the height of the liquid in the tank as a function of time is given by Equation (24). The constant C corresponds to the height of the liquid at time $t = 0$.

$$\int h^{-0.5}\, dh = -\int \frac{A_p C_v \sqrt{2g}}{A_t}\, dt \tag{23}$$

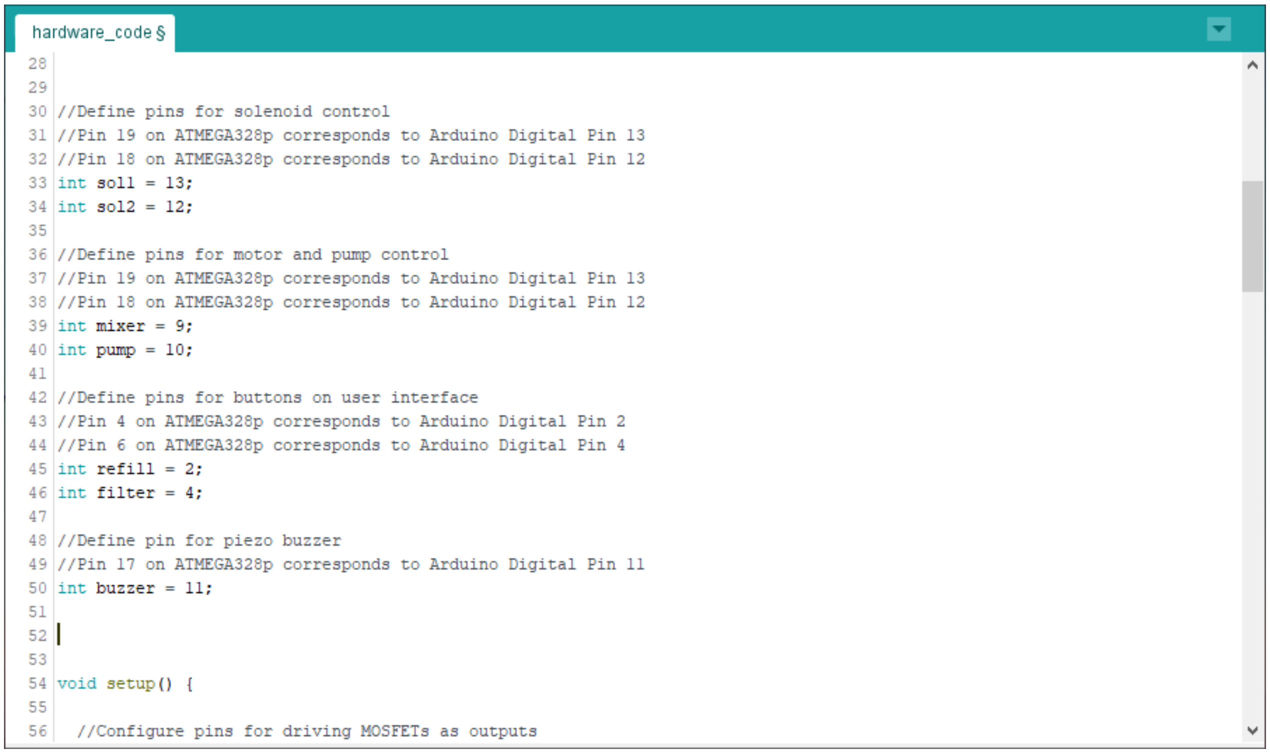
$$h(t) = \left(\frac{-A_p C_v \sqrt{2g}}{2A_t} t + C \right)^2 \tag{24}$$

Appendix E Software Images for Section 2.5



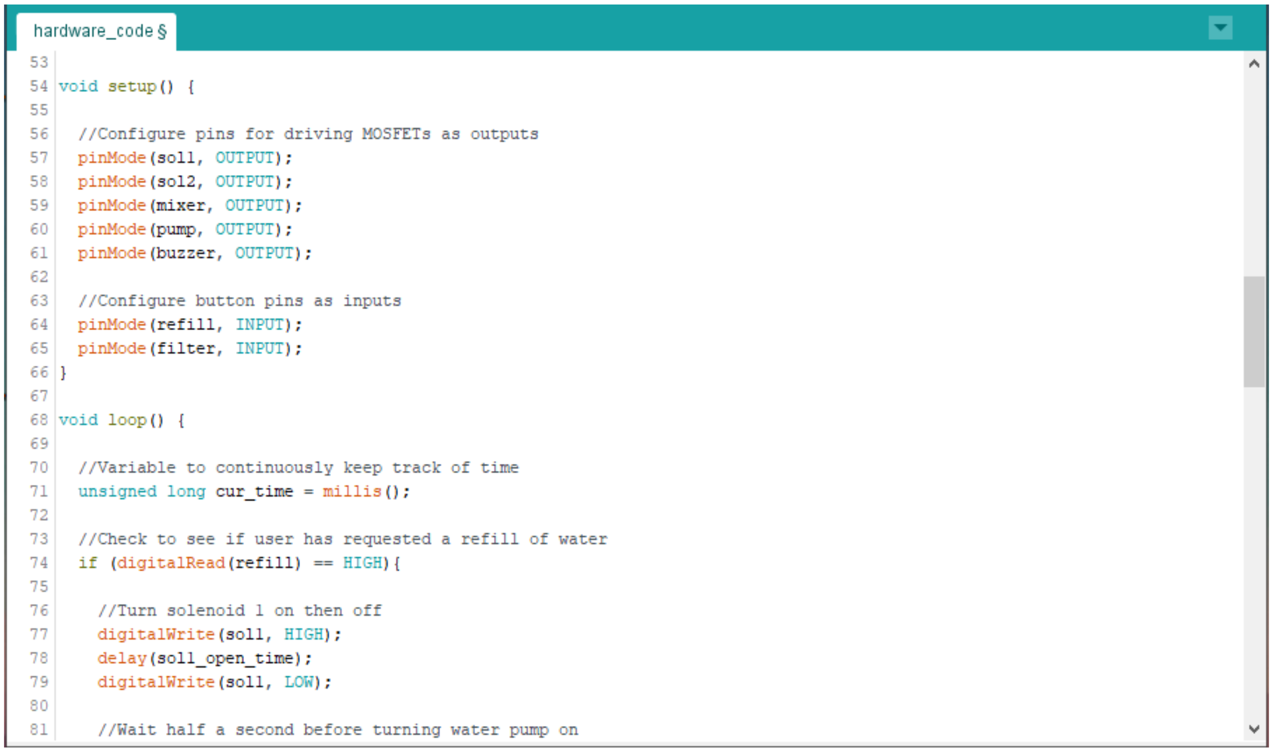
```
//Open first solenoid for 8.269 seconds to drain water from cleaning receptacle into filtration tank
const unsigned long sol1_open_time = 8269;

//Open second solenoid for 48.705 seconds to drain water from filtration tank to clean water tank
const unsigned long  sol2_open_time = 48705;

//Turn on mixer motor for two minutes to mix chemicals into wastewater
const unsigned long  mixer_on_time = 120000;

//Wait three minutes before turning mixer on again
const unsigned long  mixer_wait_time = 180000;

//Turn on dc water pump for 1.637 seconds to move 0.25L of water
const unsigned long  pump_on_time = 1637;

//Allow the treated water to sit for two hours before opening second solenoid
const unsigned long  filtration_sit_time = 7200000;

//Variable for keeping track of longer time periods
unsigned long start_time;

//Define wait flag
int wait = 0;

//Define flag to keep track of motor state
int motor_on = 0;


```

Figure 13: First Part of Software Outlined in Section 2.5



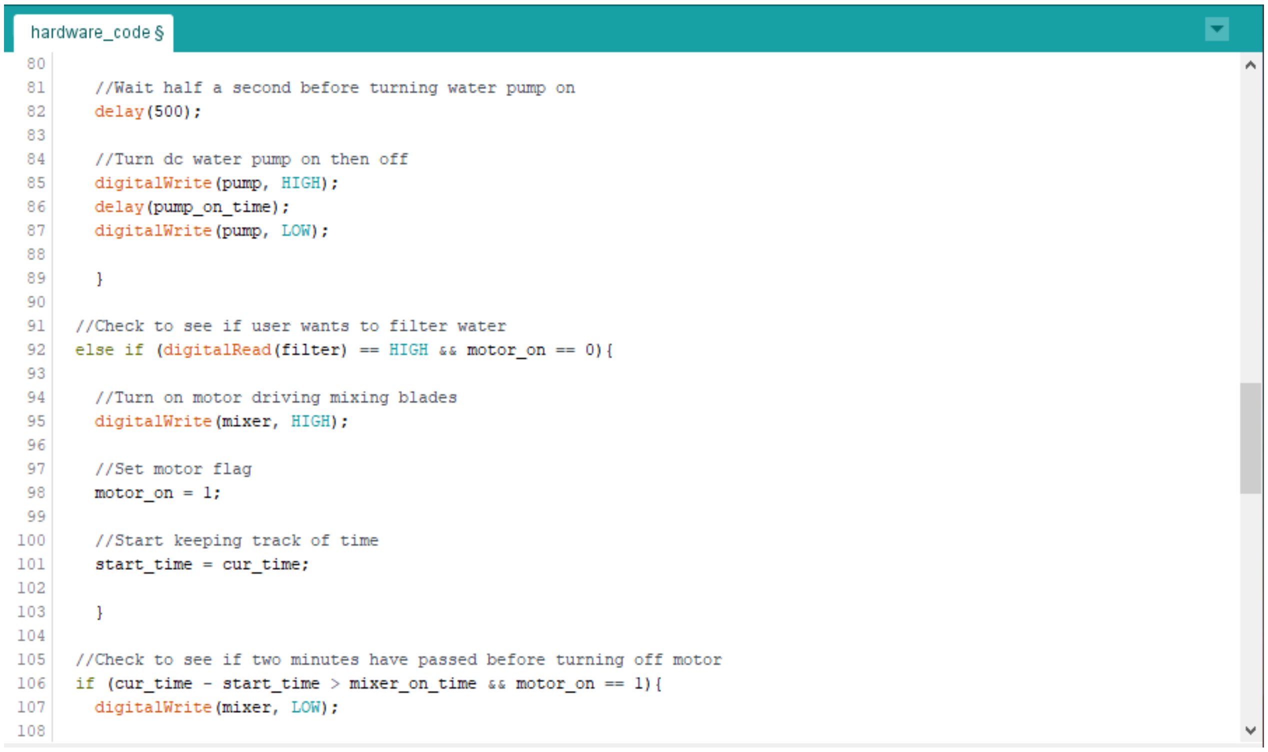
```

//Define pins for solenoid control
//Pin 19 on ATMEGA328p corresponds to Arduino Digital Pin 13
//Pin 18 on ATMEGA328p corresponds to Arduino Digital Pin 12
int sol1 = 13;
int sol2 = 12;

//Define pins for motor and pump control
//Pin 19 on ATMEGA328p corresponds to Arduino Digital Pin 13
//Pin 18 on ATMEGA328p corresponds to Arduino Digital Pin 12
int mixer = 9;
int pump = 10;

//Define pins for buttons on user interface
//Pin 4 on ATMEGA328p corresponds to Arduino Digital Pin 2
//Pin 6 on ATMEGA328p corresponds to Arduino Digital Pin 4
int refill = 2;
int filter = 4;

//Define pin for piezo buzzer
//Pin 17 on ATMEGA328p corresponds to Arduino Digital Pin 11
int buzzer = 11;



void setup() {

  //Configure pins for driving MOSFETs as outputs
```

Figure 14: Second Part of Software Outlined in Section 2.5



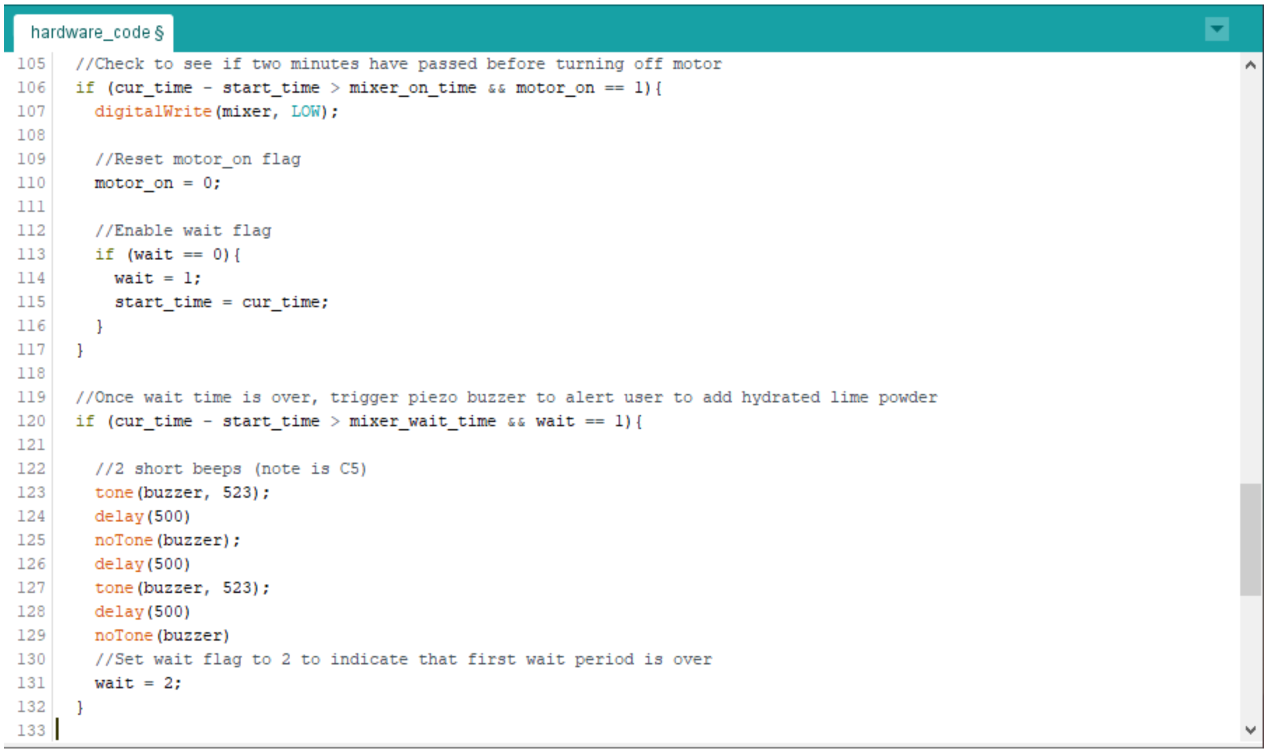
```

void setup() {

  //Configure pins for driving MOSFETs as outputs
  pinMode(sol1, OUTPUT);
  pinMode(sol2, OUTPUT);
  pinMode(mixer, OUTPUT);
  pinMode(pump, OUTPUT);
  pinMode(buzzer, OUTPUT);

  //Configure button pins as inputs
  pinMode(refill, INPUT);
  pinMode(filter, INPUT);
}

void loop() {

  //Variable to continuously keep track of time
  unsigned long cur_time = millis();

  //Check to see if user has requested a refill of water
  if (digitalRead(refill) == HIGH){

    //Turn solenoid 1 on then off
    digitalWrite(sol1, HIGH);
    delay(sol1_open_time);
    digitalWrite(sol1, LOW);

    //Wait half a second before turning water pump on
```

Figure 15: Third Part of Software Outlined in Section 2.5



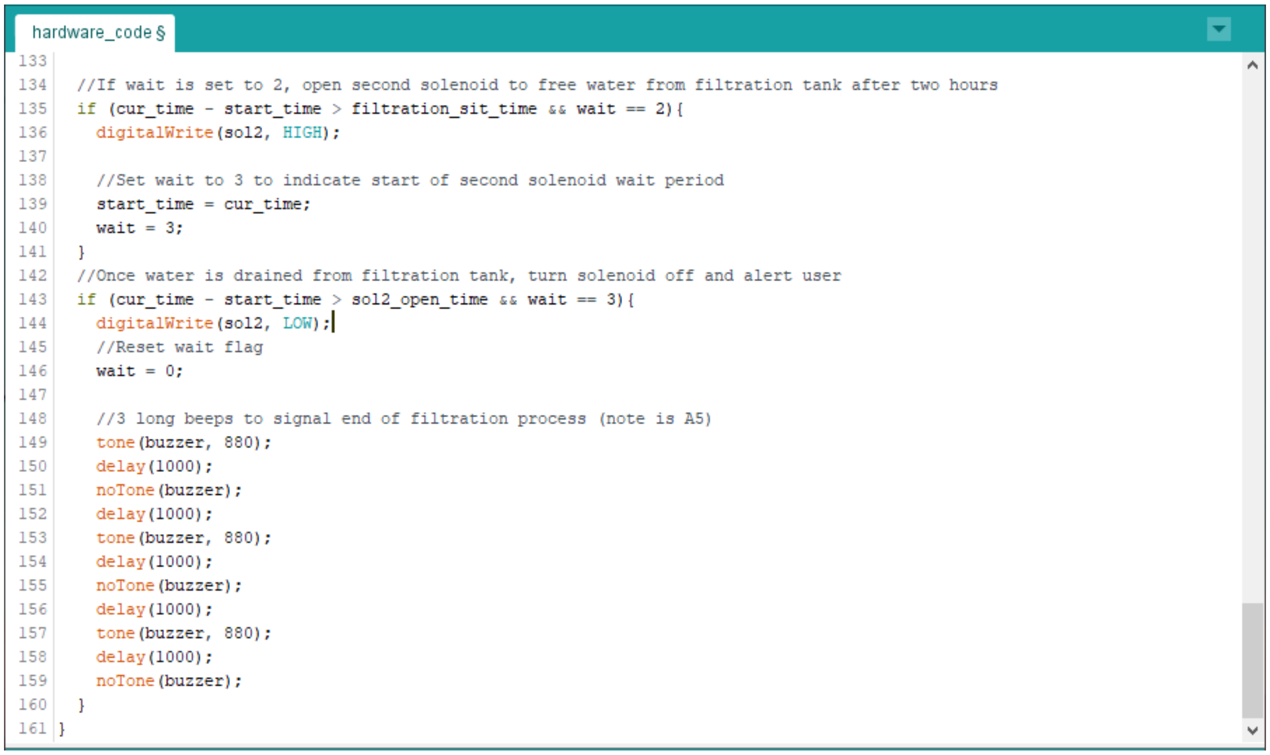
```
  //Wait half a second before turning water pump on
  delay(500);

  //Turn dc water pump on then off
  digitalWrite(pump, HIGH);
  delay(pump_on_time);
  digitalWrite(pump, LOW);

  }

//Check to see if user wants to filter water
else if (digitalRead(filter) == HIGH && motor_on == 0){

  //Turn on motor driving mixing blades
  digitalWrite(mixer, HIGH);

  //Set motor flag
  motor_on = 1;

  //Start keeping track of time
  start_time = cur_time;

  }

//Check to see if two minutes have passed before turning off motor
if (cur_time - start_time > mixer_on_time && motor_on == 1){
  digitalWrite(mixer, LOW);

```

Figure 16: Fourth Part of Software Outlined in Section 2.5



```
//Check to see if two minutes have passed before turning off motor
if (cur_time - start_time > mixer_on_time && motor_on == 1){
  digitalWrite(mixer, LOW);

  //Reset motor_on flag
  motor_on = 0;

  //Enable wait flag
  if (wait == 0){
    wait = 1;
    start_time = cur_time;
  }
}

//Once wait time is over, trigger piezo buzzer to alert user to add hydrated lime powder
if (cur_time - start_time > mixer_wait_time && wait == 1){

  //2 short beeps (note is C5)
  tone(buzzer, 523);
  delay(500)
  noTone(buzzer);
  delay(500)
  tone(buzzer, 523);
  delay(500)
  noTone(buzzer)
  //Set wait flag to 2 to indicate that first wait period is over
  wait = 2;
}

```

Figure 17: Fifth Part of Software Outlined in Section 2.5



hardware_code §

```

  //If wait is set to 2, open second solenoid to free water from filtration tank after two hours
  if (cur_time - start_time > filtration_sit_time && wait == 2){
    digitalWrite(sol2, HIGH);

    //Set wait to 3 to indicate start of second solenoid wait period
    start_time = cur_time;
    wait = 3;
  }
  //Once water is drained from filtration tank, turn solenoid off and alert user
  if (cur_time - start_time > sol2_open_time && wait == 3){
    digitalWrite(sol2, LOW);
    //Reset wait flag
    wait = 0;

    //3 long beeps to signal end of filtration process (note is A5)
    tone(buzzer, 880);
    delay(1000);
    noTone(buzzer);
    delay(1000);
    tone(buzzer, 880);
    delay(1000);
    noTone(buzzer);
    delay(1000);
    tone(buzzer, 880);
    delay(1000);
    noTone(buzzer);
  }
}
```

Figure 18: Sixth Part of Software Outlined in Section 2.5